**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**Тернопільський національний економічний університет**
**Факультет комп'ютерних інформаційних технологій**
Кафедра комп'ютерних наук

**БОРІВЕЦЬ Ігор Іванович**

# Android-додаток для обліку відвідування занять студентами/ Android-application for the students' attendance accounting

напрям підготовки: 6.050103 - Програмна інженерія
фахове спрямування - Програмне забезпечення систем

Бакалаврська дипломна робота

Виконав студент групи ПЗС-41
І. І. Борівець

_____

Науковий керівник:
викладач ПОРПЛИЦЯ Н.П.

_____

Бакалаврську дипломну роботу
допущено до захисту:

"___" _____ 20___ р.

Завідувач кафедри
_____ **А. В. Пукас**

**ТЕРНОПІЛЬ - 2016**

Зміст

# ВСТУП

У сучасній системі освіти постійно відбуваються зміни, що зумовлені процесами економічного, політичного чи соціального характеру. Змінюються, як пріоритети в системі освіти і у системі управління освітою, так і, зміст та структура самого освітнього процесу. Крім цього, у галузі освіти, досить часто, випробовуються й новітні технології.

Зокрема, істотні зміни в інформаційній сфері діяльності людини призводять до суттєвого зниження ефективності застарілих підходів до контролю відвідуваності студентами занять. Саме тому, сьогодні є доцільною розробка спеціальної системи, яка дозволить автоматизувати процес обліку відвідування занять студентами та зробити його швидшим, зручнішим та надійнішим.

Прогули, запізнення, відсутність учнів на заняттях - це проблема кожного навчального закладу. Такі речі роблять процес навчання непередбачуваним, що надалі позначається на загальних показниках успішності. Автоматизація процесу контролю відвідуваності, може вирішити ці проблеми надавши гнучкі засоби для моніторингу за пропусками занять студентами. Така система, дозволить швидко виявляти проблеми з безпричинними пропусками занять студентом та наддасть певні засоби для запобігання прогулам. Крім того, такий засіб, буде дозволяти студенту стежити за власною відвідуваністю, дозволяючи пересвідчитися в тому, що він повною мірою використовує можливості навчання, які доступні для нього.

Найкращим рішенням для автоматизації процесу контролю відвідуваності є система, яка інтегрує у собі веб-орієнтоване програмне рішення та програмні рішення для мобільних платформ. На сьогоднішній день уже існує достатньо велика кількість веб-орієнтованих продуктів для контролю відвідуваності, фактично такі системи вже є у більшості навчальних закладів. Тому, основним предметом розробки дипломної роботи є програмний продукт для мобільних платформ, який надаватиме зручні засоби для контролю відвідуваності та інтегруватиметься із аналогічними веб-орієнтованими системами.

# РЕЗЮМЕ

Дипломна робота містить хх сторінок, хх таблиць, хх рисунків, список використаних джерел із хх найменувань та хх додатків.

Метою дипломної роботи є розробка Android-додатку для обліку відвідування занять студентами із можливістю інтеграції із веб-орієнтованим аналогом.

Об’єктом дослідження є засоби контролю та обліку відвідуваності занять у вищому навчальному закладі.

Предметом дослідження є шляхи автоматизації обліку та контролю відвідування занять студентами за допомогою інформаційних технологій.

Для розробки програмного продукту було використано технології JAVA, PHP, архітектуру REST та РСУБД MySQL.

Результатом є готовий Android-додаток для обліку відвідування занять студентами.

Ключові слова: Android-додаток, облік відвідування занять студентами, інтеграція із веб-орієнтованими продуктами, архітектура REST.

# SUMMARY

Thesis contains xx pages, xx tables, xx figures, list of sources with xx titles and xx applications.

The aim of the thesis is the development of Android-app for the students' attendance accounting with the possibility of integration with web-based analog.

The object of research is a means of control and accounting of attendance in institution of higher education.

The subject of research is ways to automate accounting and control of students' attendance through information technology.

Technologies used to develop software are JAVA, PHP, REST architecture and RDBMS MySQL.

The result is ready Android-application for the students' attendance accounting.



Keywords: Android-application, students' attendance accounting, integration with web-oriented products, architecture REST.

# РОЗДІЛ І

# АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ «ОБЛІК ВІДВІДУВАНОСТІ СТУДЕНТІВ»

## 1.1. Коротка характеристика об'єкту управління «Вищого навчального закладу»

Вищий навчальний заклад (далі ВНЗ) – освітньо-науковий заклад, що заснований та діє у відповідності із освітнім законодавством, маючи один із чотирьох ступенів акредитації, реалізує освітньо-професійні програми, забезпечує виховання, навчання та професійну підготовку осіб відповідно до їх покликання.

На сьогодні фактично кожний навчальний заклад потребує систему для обліку відвідуваності своїх студентів.

Управління ВНЗ здійснюють за допомогою багатьох відділів та підрозділів. Усі вони певним чином пов’язані між собою. У залежності від типу закладу та його внутрішнього устрою ці підрозділи можуть називатися по різному, але їх сутність є досить схожою. Далі розглянемо основні структурні підрозділи, які мають зв'язок із предметною областю курсової роботи.

Колегіальне керівництво закладом, тобто вирішення ключових питань щодо його діяльності, здійснюється найвищим адміністративним органом (переважно ректоратом). Фактично, цей орган приймає усі основні рішення щодо будь-яких питань пов’язаних з роботою ВНЗ. Таким чином, він здійснює керування усіма структурними підрозділами вищого навчального закладу та забезпечує їх ефективну роботу.

Наступним, у схемі управління навчальними процесами є «навчальний підрозділ». Навчальний підрозділ також складається із організаційних і навчально-наукових структурних підрозділів (наприклад факультетів). Кожний із них має відповідний освітній напрямок, а їхнім основним завданням є підготовка студентів по обраних спеціальностях. Кожний із організаційних і

навчально-наукових структурних підрозділів має власний керуючий орган(наприклад деканат) та власні навчальні та наукові-дослідні підрозділи(кафедри), основною метою, яких є проведення навчально-виховної та методичної діяльності. Останні із згаданих органів та підрозділів фактично є тими, які безпосередньо керують навчальним процесом та забезпечують його виконання.

На рисунку 1.1 зображено загальну схему структурної організації управління ВНЗ, яка забезпечує контроль навчального процесу, основні елементи схеми були описані вище.


Рис. 1.1. Схема організаційної структури управління

Питання відвідуваності студентами занять завжди було проблематичним для будь-якого ВНЗ. У сучасному світі є безліч найрізноманітніших причин та факторів, які можуть спонукати студентів до пропуску занять. Цікаво, що проведені дослідження свідчать про те, що причини невідвідуваності можуть

змінюватися в залежності від того на якому курсі навчається студент. Наприклад на першому курсі студентам завжди важко звикнути до нової форми навчання й вони часто пропускають заняття через проблеми з адаптацією, на другому – причиною пропусків можуть проблемні стосунки з однокурсниками, на третьому у студента може знизитися мотивація(думки про неправильно обрану професію) або з'явитися самовпевненість у власних знаннях, на четвертому – проблеми з відвідуваністю зазвичай пов'язані з намаганням поєднати навчання та роботу.

Отже, у студентів можуть бути різні причини пропуску занять, крім того, ці причини можуть змінюватися. Незмінним залишається одне – потреба у обліку відвідуваності занять. Правильний підхід до цього процесу дозволить зробити його максимально швидким та надійним. Що у свою чергу дозволить не просто виявляти студентів, які повинні бути виключені через систематичні пропуски та заборгованості, а вчасно відстежити проблеми з відвідуваністю у студента й запобігти подальшим пропускам.

Розглянемо процес обліку відвідуваності студентами занять у вищих навчальних закладах детальніше. Староста групи отримує журнал обліку та особисто вносить в нього дані про відсутність студентів групи для кожного заняття. Вигляд журналу обліку відображено у таблиці 1.1.

Таблиця 1.1

Вигляд журналу обліку

| | Дні тижня | Понеділок | Вівторок | Середа |
|---|---|---|---|---|
| | Дата заняття | | | |

| № | Прізвище та ініціали | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1. | | | | | | | | | | | | | |
| ….. | | | | | | | | | | | | | |
| Підпис Викладача | | | | | | | | | | | | | |

Староста групи зобов’язується передавати відомості про відвідуваність з певною регулярністю, переважно щоденно. Цей процес може відбуватися одним із наступних шляхів:

- староста передає журнал у відповідний керуючий орган(далі деканат), який має свій журнал і переносить в нього потрібні дані;
- староста вносить дані в спеціальну систему(базу даних) самостійно або це роблять відповідні працівники деканату.

На рисунку 1.2 зображено варіант коли староста сам вносить дані у електронний журнал відвідуваності.

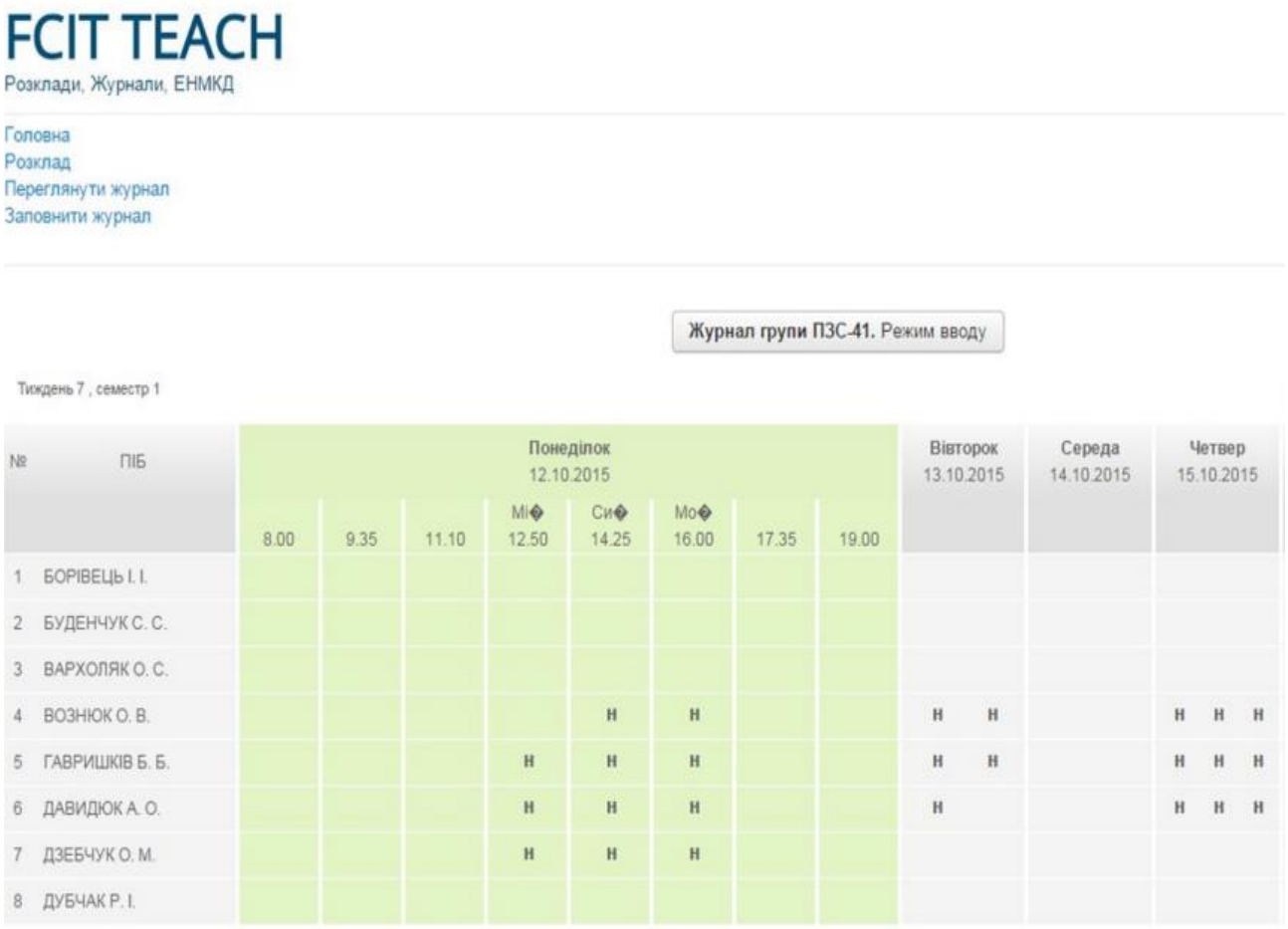


Рис. 1.2. Вигляд електронного журналу

Далі відповідно до використовуваного методу проводять облік та визначають потрібні статистичні дані. Наприклад, обраховують кількість

пропусків без поважних причин окремо для кожного студента. Таким чином, студенти, які перевищили встановлений навчальним закладом ліміт пропусків, потрапляють у списки на виключення. Сутність процесу обліку відвідуваності студентами занять описує діаграма IDEF0, яка зображена на рисунку 1.3.

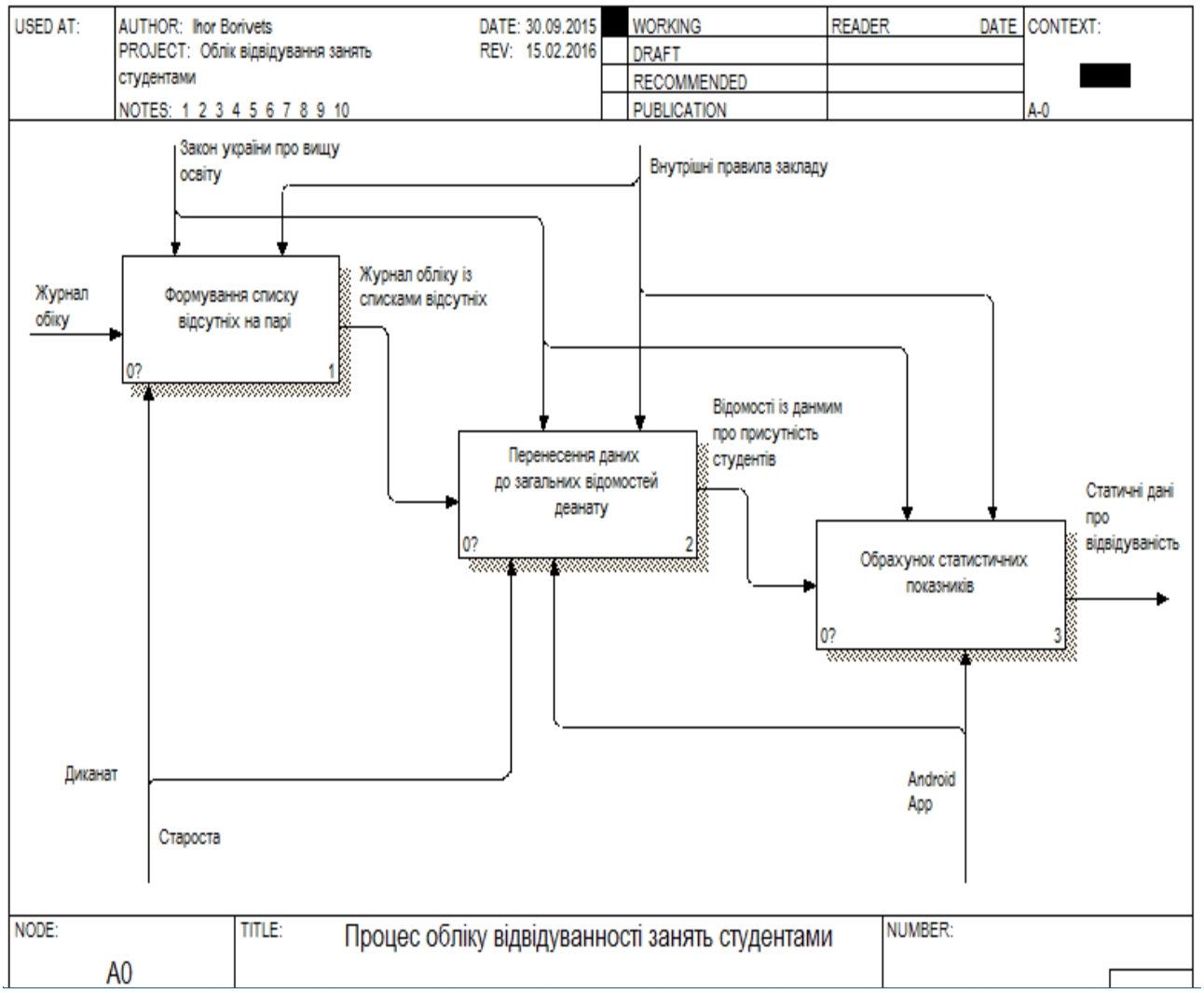


Рис. 1.3. Декомпозована діаграма IDEF0

Процес обліку відвідуваності за такою схемою є складним та містить велику кількість недоліків. На будь-якому з вищезгаданих етапів може трапитися помилка, особливо, якщо вся інформація обробляється вручну. Деякі старости можуть не завжди подавати точні дані, бажаючи «прикрити» товаришів. Фактично, кожний із викладачів веде власний журнал обліку відвідуваності, що є повторенням процесу. Крім того, процес отримання та

аналізу статистичних даних про відвідуваність за описаною схемою, є досить повільним і, як наслідок, процес повідомлення про проблеми з відвідуваністю є відсутнім, аж до того моменту, поки студент не перевищить ліміт пропусків.

Після розгляду основної проблеми, очевидно, що для її розв'язку, для певних класів користувачів, необхідно буде використовувати мобільну платформу. Проте найкраще буде використовувати мобільну платформу інтегровану із веб-орієнтованою програмною системою. У більшості навчальних закладів уже є веб-сайти для обліку відвідуваності студентів. Проте, вони здатні лише на часткове вирішення проблеми. Вони є хорошим рішення для працівників деканату, які отримують автоматизований засіб для швидкого керування, який дозволяє миттєво обраховувати статистичні показники та працювати з великими об'ємами даних. Однак веб-сайт аж ніяк не вирішує проблему для інших користувачів системи. Зрозуміло, що практично не можливим є використання веб-орієнтованого продукту студентами та викладачами під час навчального процесу. Це демонструє, що веб-орієнтоване вирішення проблеми має велику кількість недоліків. Хорошим прикладом є те, що викладач не зможе вводити дані у систему безпосередньо під час заняття. При інтеграції з мобільною платформою майже усі недоліки веб-орієнтованої системи нівелюватимуться перевагами системи на мобільній платформі. Щодо вибору мобільної платформи, очевидно, що найкраще вибрати найпопулярнішу, якою на сьогодні є Android.

## 1.2. Опис предметної області «Облік відвідуваності студентів»

Для належного функціонування Android-додатку для обліку відвідуваності студентів потрібно забезпечити виконання таких основних бізнес-процесів (рис. 1.4):

1) процес збору даних про відвідуваність;
2) процес обробки та аналізу даних;
3) процес перегляду статистичних даних;

4) процес повідомлення про проблеми з відвідуваністю.

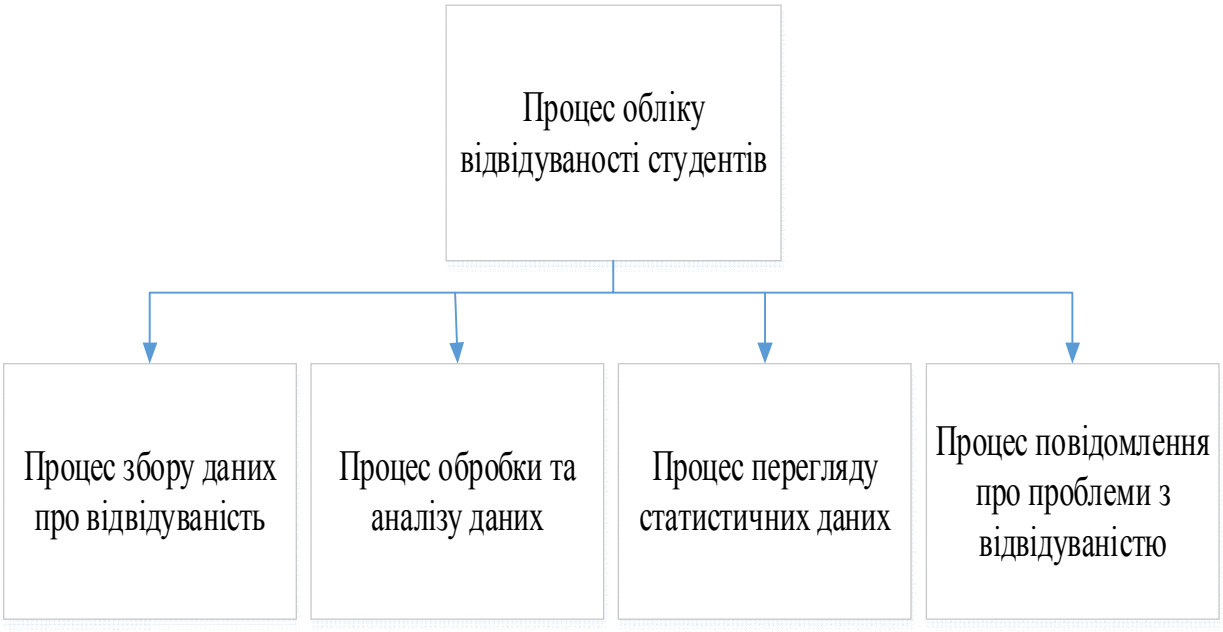


Рис. 1.4. Діаграма бізнес-процесів розроблюваного програмного продукту

Розглянемо детальніше кожен з вище представлених бізнес-процесів. На рисунку 1.5 бачимо діаграму функцій, які потрібні для забезпечнення виконання процесу збору даних.

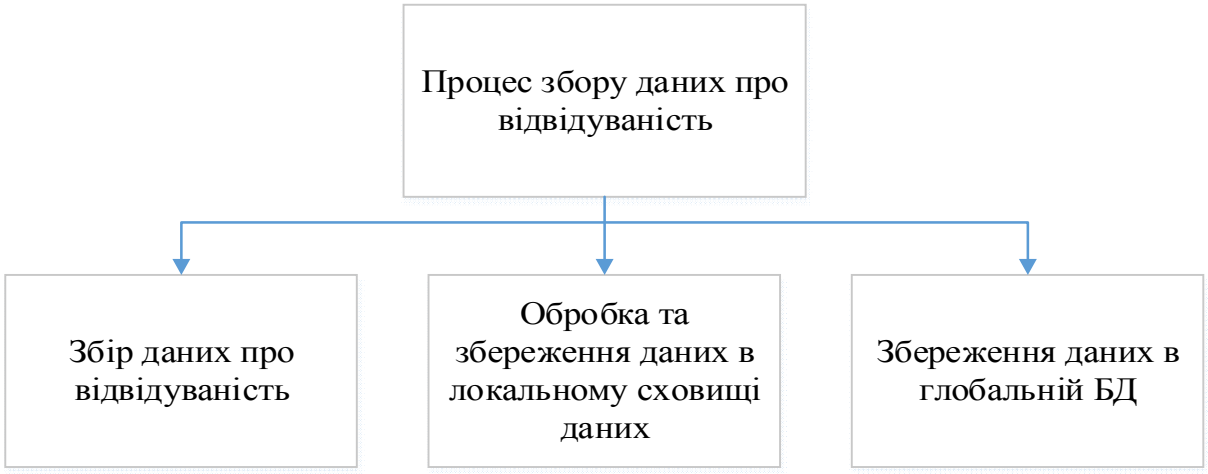


Рис. 1.5. Діаграма функцій процесу збору даних про відвідуваність

Першим етапом для обліку відвідуваності студентів є збір даних про відвідуваність. Цей процес є дуже важливим, оскільки від ного залежатиме якість та достовірність результатів всіх інших бізнес-процесів. Тому, процес

має бути організованим таким чином, щоб забезпечити максимальну точність при мінімальній кількості етапів.

Характеристику бізнес-процесу збору даних наведено в таблиці 1.2.

Таблиця 1.2.

Характеристика бізнес-процесу збору даних про відвідуваність

| Назва характеристики | Значення характеристики |
|---|---|
| Ім'я бізнес-процесу | Збір даних про відвідуваність |
| Основні учасники | Староста, викладач |
| Вхідна подія | Проведення заняття |
| Вхідні документи | Список студентів |
| Вихідна подія | Проведене заняття та зібрані дані про присутність студентів |
| Вихідні документи | Список відсутніх студентів |
| Клієнт бізнес-процесу | Обробка та аналіз даних |

На рисунку 1.6 бачимо діаграму функцій необхідних для виконання процесу обробки та аналізу даних.


Рис. 1.6. Діаграма функцій процесу обробки та аналізу даних

На етапі обробки та аналізу даних виконується певне опрацювання даних, яке забезпечує можливість проведення різноманітних аналітичних розрахунків

та збереження їх результатів. Фактично, саме цей бізнес-процес відповідальний за створення різноманітних статистичних показників щодо відвідуваності.

Характеристику бізнес-процесу обробки та аналізу даних наведено в таблиці 1.3.

Таблиця 1.3.

Характеристика бізнес-процесу обробки та аналізу даних

| Назва характеристики | Значення характеристики |
|---|---|
| Ім'я бізнес-процесу | Обробка та аналіз даних |
| Основні учасники | Працівник деканату, електрона система |
| Вхідна подія | Зібрані дані про відвідуваність передані на подальше опрацювання |
| Вхідні документи | Журнали із даними про відвідуваність |
| Вихідна подія | Створено статистичні дані про відвідуваність |
| Вихідні документи | Документи із статистичними показниками |
| Клієнт бізнес-процесу | Перегляд статистичних даних |

На рисунку 1.7 бачимо діаграму функцій процесу перегляду статистичних даних.


Рис. 1.7. Діаграма функцій процесу перегляду статистичних даних

На етапі обробки та аналізу даних про відвідуваність отримують статистичні показники, які зберігають у певному вигляді. Бізнес-процес перегляду статистичних даних дозволяє використовувати ці дані для того щоб

отримати потрібну інформацію щодо статистики відвідування занять студентами.

Характеристику бізнес-процесу перегляду статистичних даних наведено в таблиці 1.4.

Таблиця 1.4.

Характеристика бізнес-процесу перегляду статистичних даних

| Назва характеристики | Значення характеристики |
|---|---|
| Ім'я бізнес-процесу | Перегляду статистичних даних |
| Основні учасники | Викладач, працівник деканату |
| Вхідна подія | Запит про статистичні показники |
| Вхідні документи | Документи із статистичними показниками |
| Вихідна подія | Отримано статистичні показники |
| Вихідні документи | - |
| Клієнт бізнес-процесу | Повідомлення про проблеми з відвідуваністю |

На рисунку 1.8 бачимо діаграму функцій процесу повідомлення про проблеми з відвідуваністю.

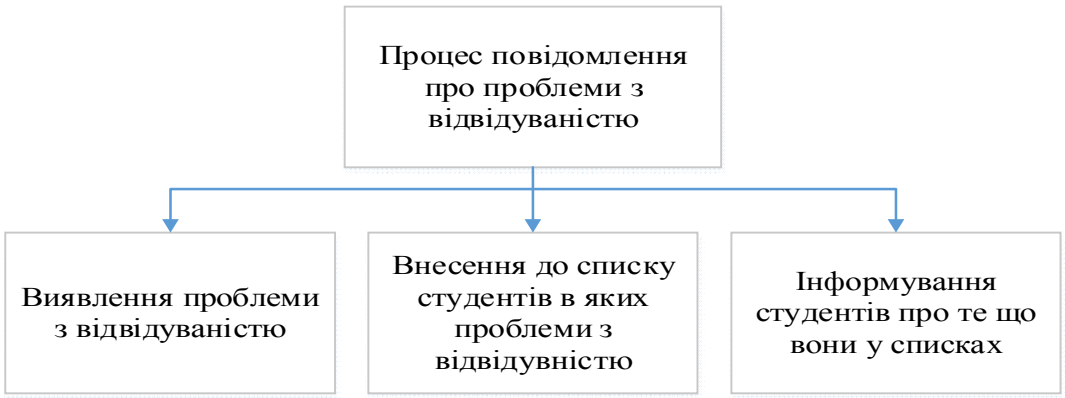


Рис. 1.8. Діаграма функцій процесу повідомлення про проблеми з відвідуваністю

Для виявлення проблем з відвідуваністю переглядаються статистичні показники. Таким чином, якщо в якогось із студентів є значні порушення

допустимої норми пропущених занять він потрапляє у список студентів, яких можуть виключити та має бути проінформованим про це.

Характеристику бізнес-процесу процесу повідомлення про проблеми з відвідуваністю наведено в таблиці 1.5.

Таблиця 1.5.

Характеристика бізнес-процесу процесу повідомлення про проблеми з відвідуваністю

| Назва характеристики | Значення характеристики |
|---|---|
| Ім'я бізнес-процесу | Повідомлення про проблеми з відвідуваністю |
| Основні учасники | Працівник деканату |
| Вхідна подія | Запит на виявлення проблем з відвідуваності |
| Вхідні документи | Документи із статистичними показниками |
| Вихідна подія | Проблеми з відвідуваністю виявлені та опрацьовані |
| Вихідні документи | Списки студентів з проблемами з відвідуваністю |
| Клієнт бізнес-процесу | - |

## 1.3. Огляд існуючих аналогів

На сучасному IT-ринку уже існує досить широкий спектр програмних продуктів, які дозволяють частково автоматизувати процес обліку відвідуваності у навчальному закладі. Проте, потрібно відзначити, що кожен програмний продукт має свої методи для вирішення проблеми обліку відвідуваності і є рішенням для конкретного навчального закладу. Незалежно від платформи, кожна система для моніторингу відвідуваності може представляти унікальні способи реалізації функціоналу, які можливо адаптувати під будь-яку платформу. Саме тому, надзвичайно важливо провести аналіз ринку існуючих програмних засобів не лише для мобільних платформ.

У межах дипломної роботи було розглянуто широкий асортимент програмних продуктів для розв’язання проблеми обліку відвідуваності

студентів. При цьому, для детального огляду та аналізу, було обрано найпопулярніші програмні системи, такі як: «BioTime EDU», «MyAttendanceTracker» (режим доступу: https://www.myattendancetracker.com/), «Attendance», «Attendance Tracker». Варто зазначити, що кожен із цих засобів використовує унікальні методи та технології для реалізації основного функціоналу.

Розглянемо детальніше систему «BioTime EDU». Ця програмна система розроблена британською компанією «M3 Biometrics». Вона майже цілком забезпечує автоматизацію процесу моніторингу за відвідуваністю у навчальному закладі. Програма має інтуїтивно зрозумілий інтерфейс, але для роботи з деякими функціями персоналу потрібно пройти підготовку. Система «BioTime EDU» передбачає встановлення біометричних терміналів на вході у кожну аудиторію та наявність засобів для комунікації з центральними серверами. Таким чином, кожний студент у якого має відбутися заняття повинний отримати доступ при вході у аудиторію через термінал, приставивши палець до сканера і зробити теж саме після завершення заняття. Приклад біометричного сканеру та процес ідентифікації студента зображено на рисунку 1.9.

Рис. 1.9. Процес ідентифікації студента при вході (виході) в аудиторію

Таким чином, система «BioTime EDU» повністю звільняє викладача від процесу перевірки присутності студентів та автоматично формує списки

присутніх, відзначаючи час коли студент увійшов та вийшов. Також, система дозволяє складати розклади занять (рис. 1.10).

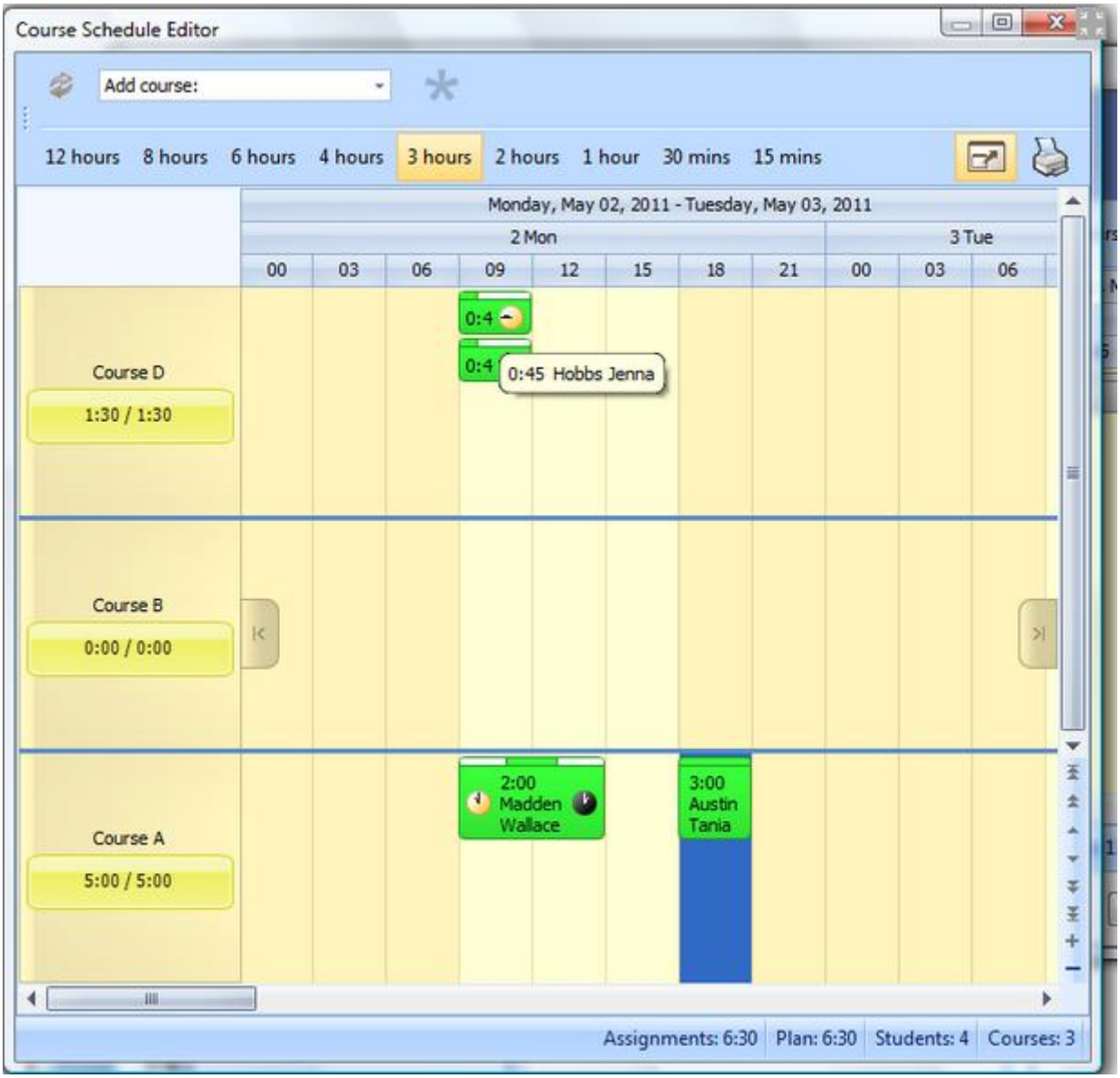


Рис. 1.10. Вікно для побудови розкладів занять

Крім того, програмна система «BioTime EDU» реалізує такі функції: додавання, перегляд та редагування персональних даних кожного студента, перегляд різноманітних звітів (звіт по відвідуваності, звіт по присутності в аудиторіях, звіт по студентах які перевищили ліміт пропусків та ін.), контроль доступу в аудиторії студентів та персоналу навчального закладу. Важливою особливістю даного продукту є те, що після досягнення певним студентом

встановленого ліміту пропусків, він отримає на свій e-mail автоматично згенероване попередження про можливість того, що він потрапить у списки студентів на виключення.

На рисунку 1.11 проілюстровано загальний вигляд вікна для перегляду персональних даних студента та вікна програмної системи для їх редагування.

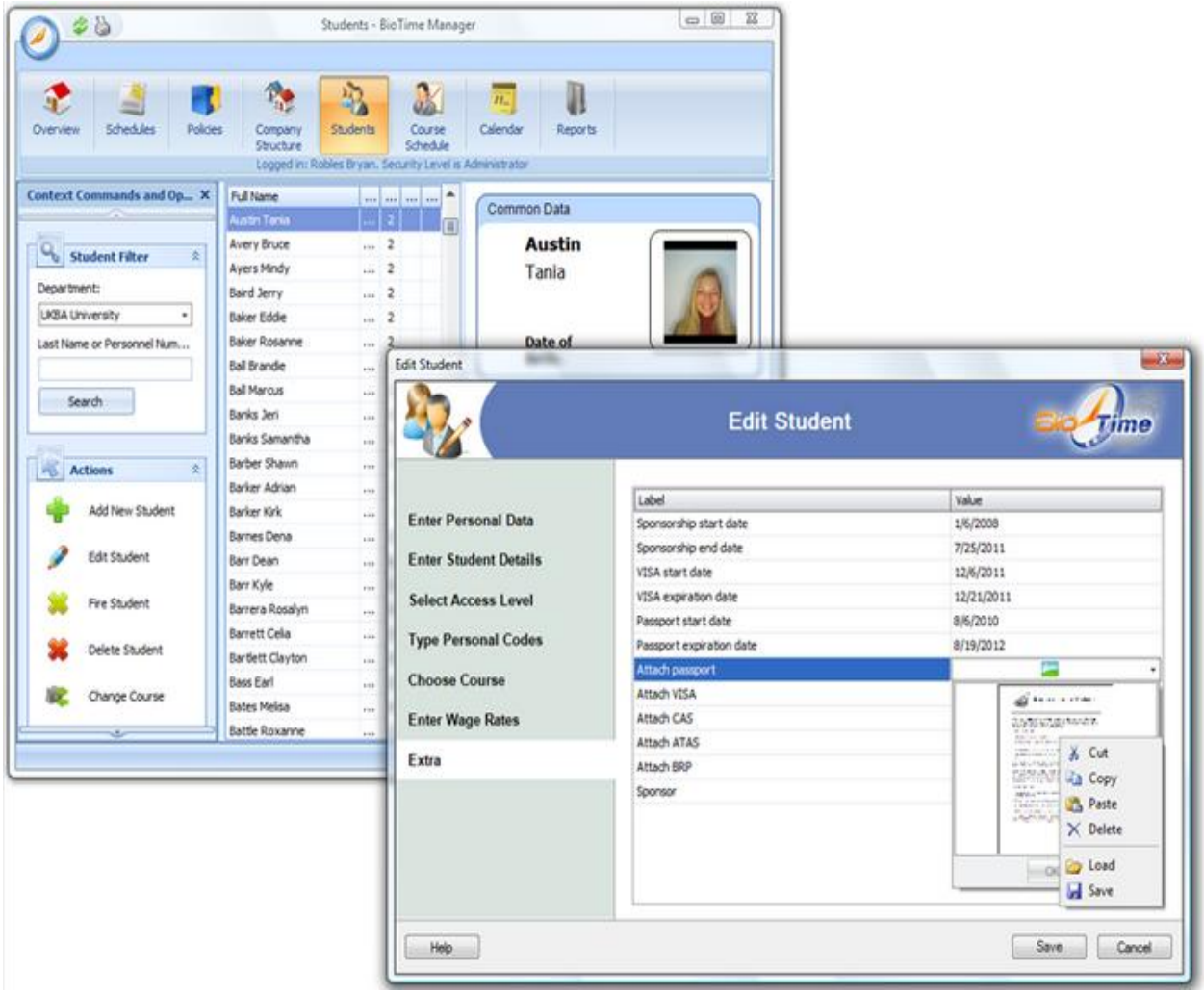


Рис. 1.11. Вікна для перегляду та редагування персональних даних студента

Далі розглянемо детальніше веб-орієнтовану програмну систему «MyAttendanceTracker» (режим доступу: https://www.myattendancetracker.com/). Веб-сайт було створено, як персональний помічник викладача для обліку відвідуваності його занять студентами. Однак, з часом програмна система доопрацьовувалася та удосконалювалася, що в свою чергу забезпечило значне

розширення її функціоналу, який включає досить цікаві особливості. Зокрема, інтерфейс системи є досить зручним, а також він максимально оптимізований для запуску на мобільних платформах. При першому використанні програмного засобу, автоматично відкривається помічник, який допомагає розпочати роботу та ознайомитися з веб-сайтом (рис.1.12), що безумовно є перевагою цієї програмної системи. Оскільки, продукт розроблявся, як персональний помічник, викладач сам повинен формувати списки студентів, розклади занять та вносити персональні дані студентів.

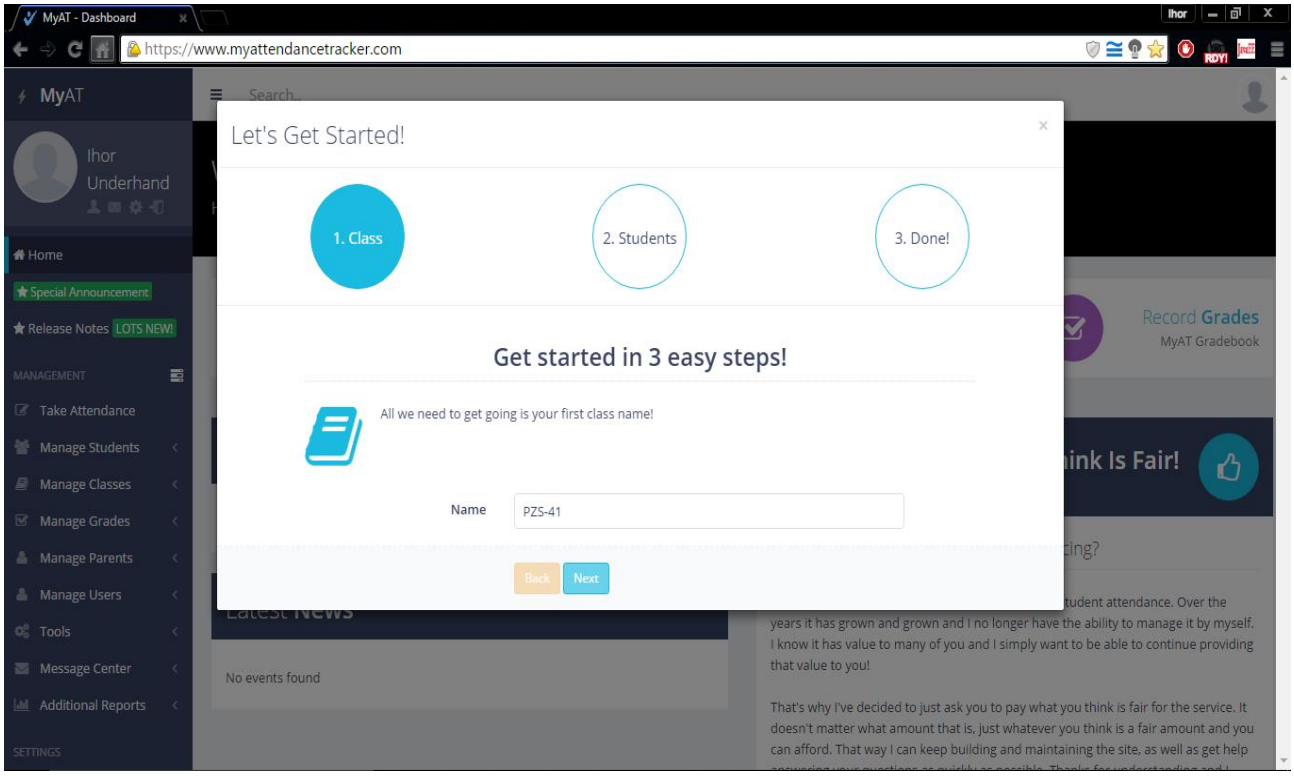


Рис. 1.12. Помічник для початку роботи та знайомства з веб-сайтом

Цікавою особливістю системи є те, що він передбачає заповнення сторінки персональної інформації батьків студента(рис.1.13). Таким чином, викладач зможе надсилати скаргу на відвідуваність чи погану успішність безпосередньо батькам студента.

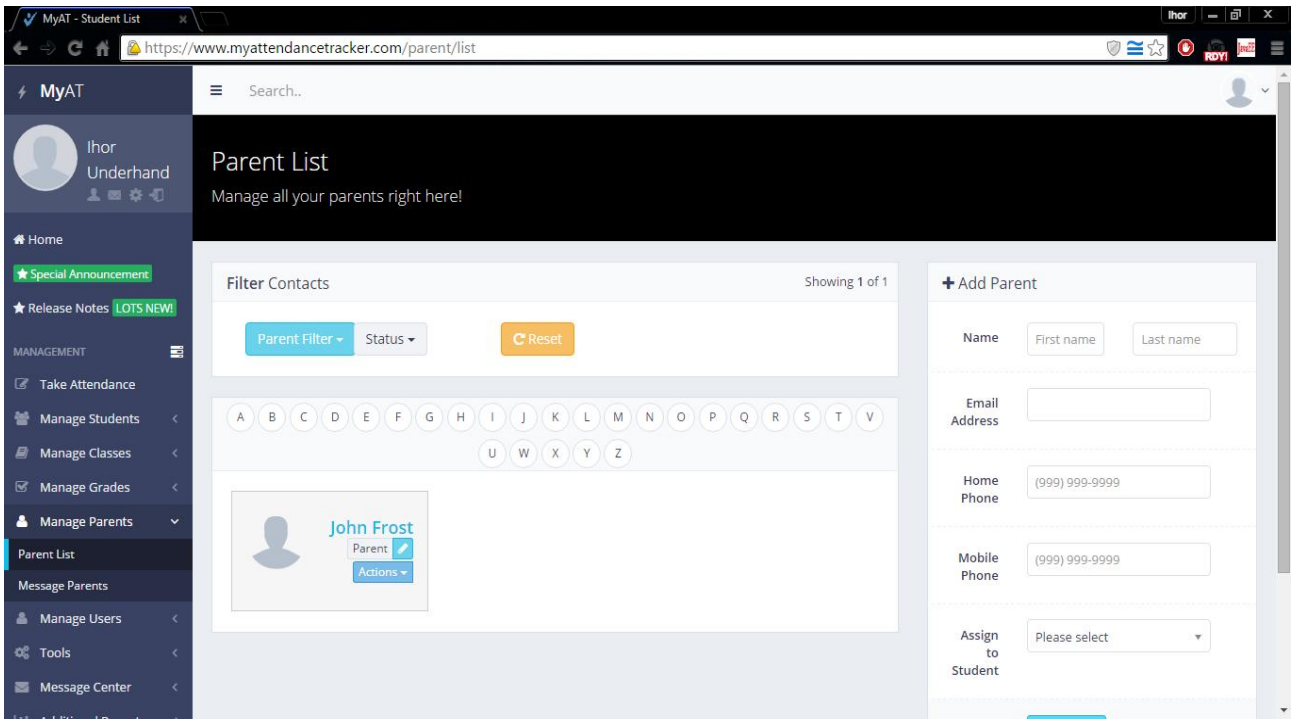


Рис. 1.13. Сторінка для введення персональної інформації батьків студента

Крім того, система, також, дозволяє переглядати різноманітні звіти (рис.1.14).

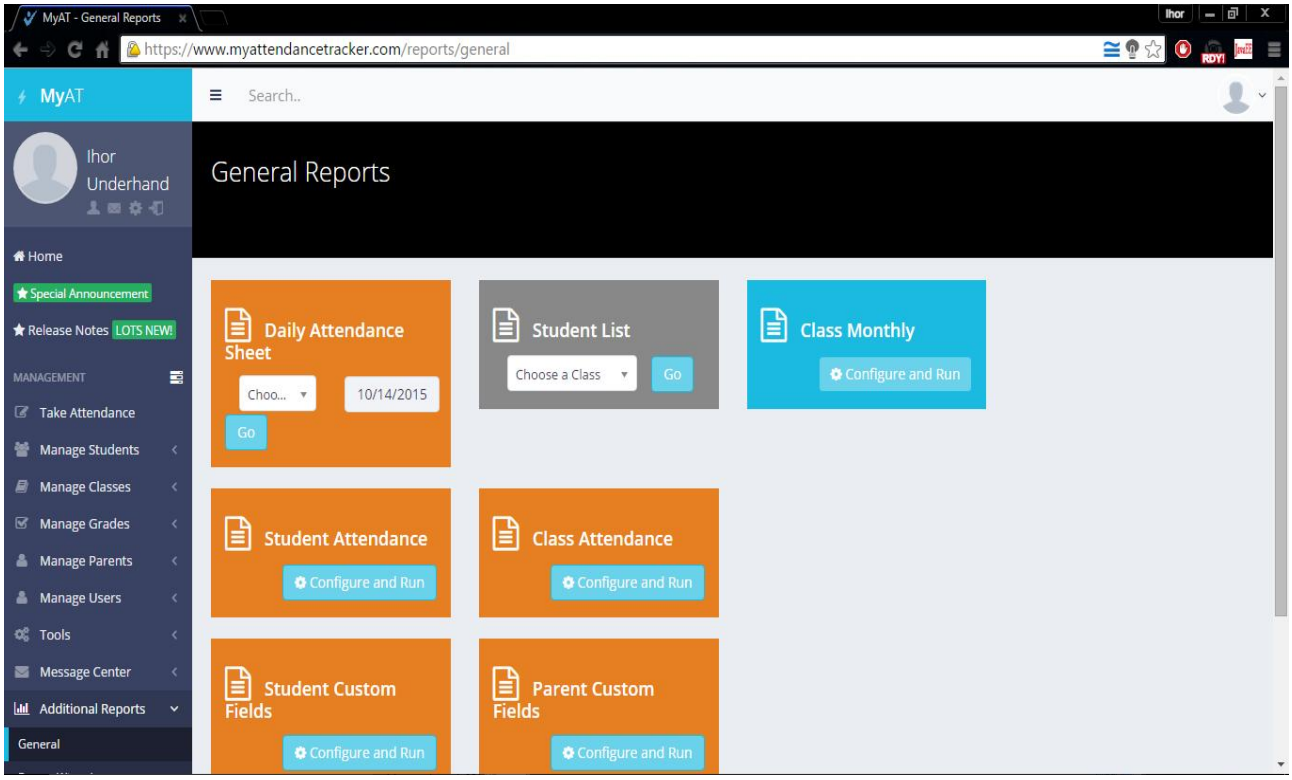


Рис. 1.14. Сторінка для вибору типу звіту

При цьому, продукт надає користувачу унікальну можливість – керувати структурою звітів та створювати нові види звітів (рис.1.15). Ще однією особливістю системи є те, що викладач сам може вирішувати, яким користувачам надати доступ до звітів про відвідуваність.

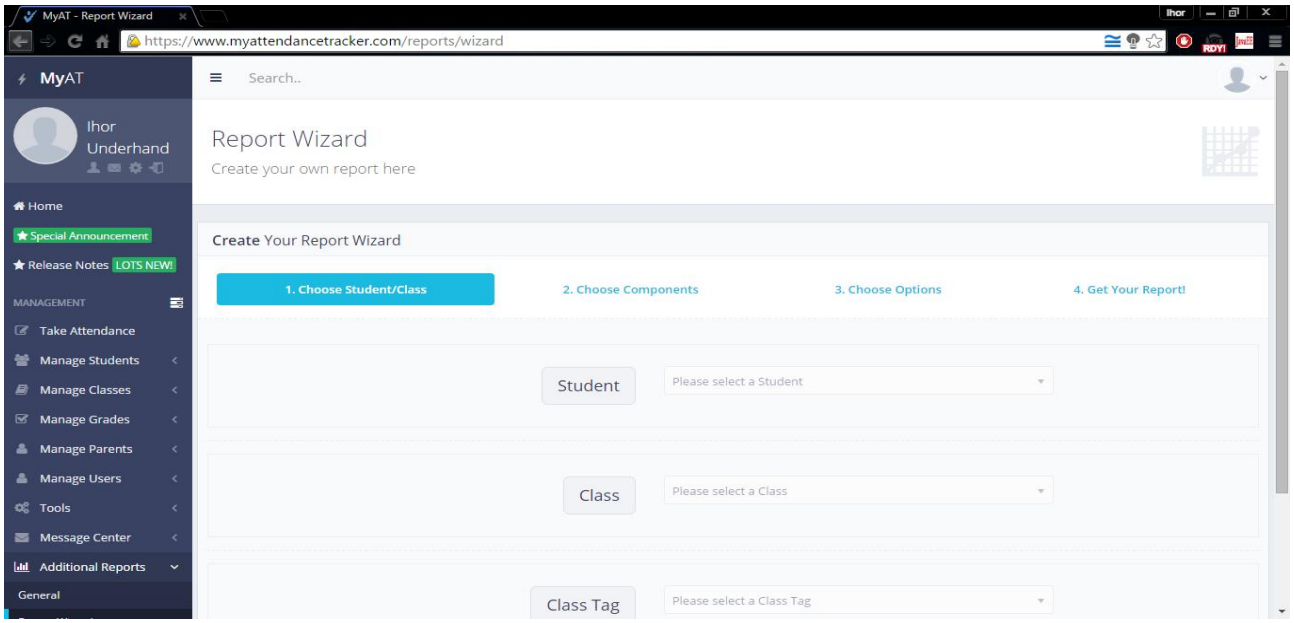


Рис. 1.15. Сторінка для створення нового виду звіту

На рисунку 1.16 зображено основну сторінку веб-орієнтованої програмної системи для введення даних про відвідуваність.

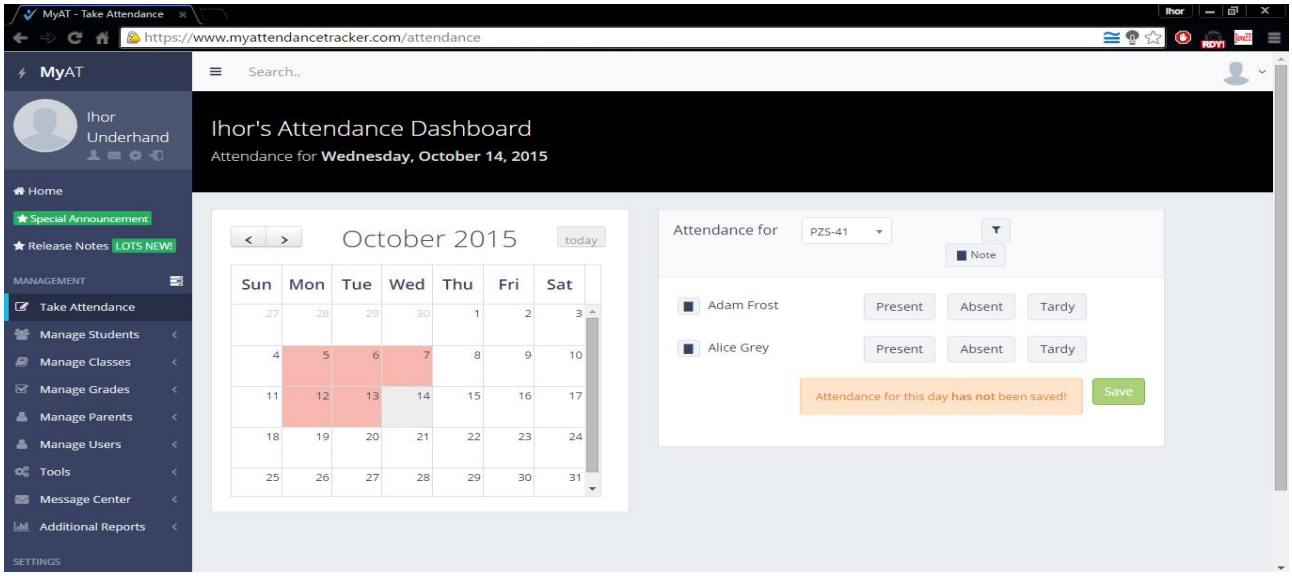

Рис. 1.16. Сторінка для введення даних про відвідуваність

Далі розглянемо систему для моніторингу відвідуваності у навчальному закладі «Attendance». «Attendance» – це мобільний додаток для платформи Android, який позиціонує себе як систему для викладачів, які хочуть мати можливість вести облік присутності своїх студентів використовуючи телефони. Інтерфейс додатку є досить простим та функціональним. Для того, щоб допомогти користувачу швидко навчитися користуватися системою, передбачено підказки, які вбудовані безпосередньо у інтерфейс. Додаток орієнтований на одного користувача, який сам вноситиме дані про студентів та налаштовуватиме свій розклад.

У головному вікні додатку відображається перелік занять для поточної дати, що проілюстровано на рисунку 1.17.

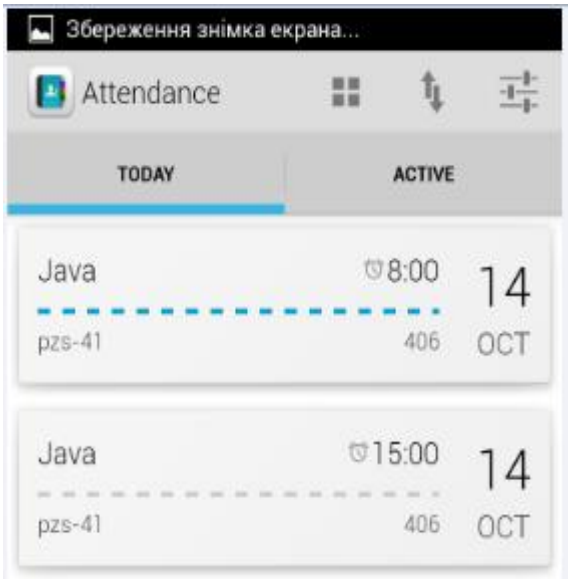


Рис. 1.17 Основне вікно додатку

Після вибору конкретного заняття, користувачу відображається список студентів, які повинні бути на ньому присутніми. Для кожного студента викладач може обрати відповідний стан із контекстного меню, що проілюстровано на рисунку 1.18. Після цього список оновиться і буде відображати усі введені зміни. Вибір стану присутності для кожного студента із контекстного меню є не надто вдалим рішення, оскільки при великій кількості

студентів такий підхід забиратиме у викладача досить велику кількість часу. Крім того, це не зручно для викладача тим, що вимагатиме від нього повторення великої кількості однотипних дій.


Рис. 1.18. Вікно вибору стану присутності

Далі програма автоматично зберігає введену інформацію, щоб користувач мав змогу переглядати списки присутніх на кожному занятті для певного предмету (рис.1.19).


Рис. 1.19 – Вікно зі списком занять з обраного предмету

Однією із переваг системи, безумовно є можливість імпорту списків студентів із текстового документу із розширенням .txt. Для цього потрібно буде

створити документ за наданим форматом(рис.1.20). Після чого вибрати функцію імпорту списку із документу та вибрати потрібний документ знайшовши його розташування на диску.

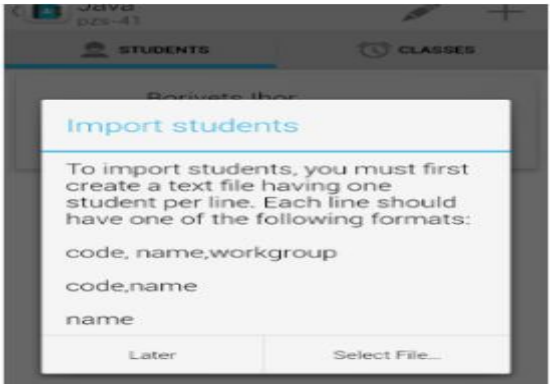


Рис. 1.20. Вікно імпорту списку студентів

Наступним розглянемо Android-додаток «Attendance Tracker», який дозволяє проводити моніторинг відвідуваності фактично для будь-яких подій, зокрема, й відвідуваності занять студентами. Інтерфейс програми є простим, але не зовсім зрозумілим, що можна відзначити, як недолік цієї програмної системи. Розробники вирішили цю проблему за допомогою детальної інструкції користувача (рис.1.21).


Рис. 1.21 – Вікно інструкції користувача

Як і у попередньому розглянутому програмному продукті, тут користувачу потрібно самому створити свій розклад та внести список учасників подій. Після цього, він зможе вибрати потрібне заняття та ввести дані про відсутність чи присутність студентів. Важливо зазначити, що додаток забезпечує можливість відправки e-mail повідомлення групі людей з певним станом присутності (рис.1.22), що безумовно є перевагою.

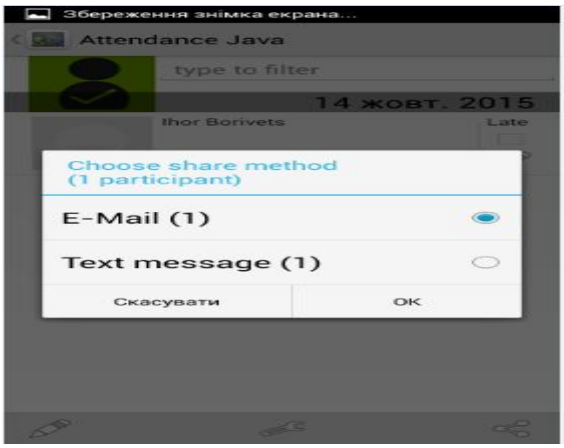


Рис. 1.22. Вікно для надсилання повідомлення

Крім того, додаток дозволяє переглядати звіти відвідуваності в обраному форматі та експортувати потрібний користувачу звіт у файл (рис.1.23). Основною перевагою додатку є можливість зробити резервне копіювання усіх даних, для подальшого використання на інших пристроях.


Рис. 1.23. Вікно для експорту статистики відвідуваності

Проведення порівняльної характеристики розглянутих програмних систем-аналогів дало можливість виокремити їх основні переваги та недоліки, що відображено в таблиці 1.6.

Таблиця 1.6

Порівняльна характеристика програмних продуктів

| Назва програмного продукту | «BioTime EDU» | «MyAttendanceTracker.com» | «Attendance» | «Attendance Tracker» |
|---|---|---|---|---|
| Інтерфейс користувача | Зрозумілий інтерфейс, для роботи з деякими функціями потрібна спеціальна підготовка | Зрозумілий та зручний інтерфейс, максимально адаптований під мобільні платформи | Зрозумілий та зручний інтерфейс | Інтерфейс є складним для розуміння |
| Допомога користувачу | Користувачі системи потребуватимуть постійної підтримки спеціалістів та техніків | Присутні підказки для початку роботи з системою | Присутні підказки, які вбудовано безпосередньо в користувацький інтерфейс | Присутня інструкція користувача |
| Функціонал для студентів | Реалізована можливість доступу та надано певний функціонал | Реалізована можливість доступу та надано певний функціонал | Доступ має лише один користувач (тобто тільки викладач) | Доступ має лише один користувач (тобто тільки викладач) |

Продовження таблиці 1.6

| | | | | |
|---|---|---|---|---|
| Робота основних функції без доступу до Інтернету | Неможлива, оскільки потрібне підключення до зовнішньої БД на сервері | Неможлива, оскільки це веб-орієнтована система | Можлива, оскільки реалізовано локальне сховище даних | Можлива, оскільки реалізовано локальне сховище даних |
| Можливість зв'язку із студентами | Відсутня | Присутня | Відсутня | Відсутня |
| Потреба у додаткових апаратних інтерфейсах | Потрібні дорогі біометричні сканери та засоби для їх підтримки | Не потрібно додаткових апаратних інтерфейсів | Не потрібно додаткових апаратних інтерфейсів | Не потрібно додаткових апаратних інтерфейсів |

Розглянемо основні переваги, які можна запозичити із кожного продукти та недоліки яких слід уникнути. Система «BioTime EDU» надає засоби для автоматичного попередження студента про переміщення ліміту пропусків, що однозначно слід реалізувати у розроблюваній системі. Обов'язково потрібно уникнути основних недоліків «BioTime EDU» – використання занадто дорогих компонентів та складного інтерфейсу. Продукт «MyAttendanceTracker» не має суттєвих недоліків, проте представляє цікавий механізм використання фільтрів для статистичних даних. Єдиним корисним рішення, яке можна запозичити з додатку «Attendance» є детальна інструкція користувача, проте, додаток «Attendance Tracker» має цікаву функцію, яка дозволить викладачу надсилати повідомлення своїм студентам. Основними недопустимими для розроблюваної системи недоліками двох вищезгаданих додатків є те, що вони не використовують зовнішньої БД та призначені для одного користувача.

## 1.4. Специфікація вимог до Android-додатку для обліку відвідуваності занять студентами

Після детального дослідження предметної області можна перейти до визначення основних вимог щодо системи, яка розробляється в дипломній роботі. Очевидно, що Android-додаток надає багато цікавих рішень та переваг для вирішення проблеми обліку відвідуваності студентів. Проте необхідно відмітити, що розв'язання деяких завдань на платформі Android є недоцільним через певну обмеженість мобільних платформ. Тому, Android-додаток для обліку відвідуваності занять студентами слід розглядати, як підсистему інтегровану у продукти, які функціонують не на мобільних платформах. Така інтеграція дозволить значно розширити систему для обліку відвідуваності. Найпоширенішими та найбільш сумісними з мобільними платформами є веб-орієнтовані продукти. Фактично, Android-додаток забезпечить виконання майже всього основного функціоналу пов'язаного із певними класами користувачів, надавши для цього зручніші та більш гнучкі рішення та засоби. Крім того, використання мобільної платформи дозволить розв'язати багато проблем та незручностей, які неможливо або дуже складно вирішити за допомогою інших платформ.

Аналіз існуючої продукції допоміг побачити непотрібність певних технічних рішень, що зробили б систему занадто складною та невиправдано дорогою. Варто відзначити, що даний етап, також, показав й основні проблеми систем пов'язані з виглядом інтерфейсу користувача та допоміг відсіяти деякий непотрібний функціонал.

Далі можливе використання специфічної термінології, тому доцільно розглянути глосарій проекту, який наведено в додатку А.

Розглянемо функціональні вимоги до розроблюваної системи. В системі передбачається два типи користувачів: студент і викладач. Кожному із них додаток надаватиме різний функціонал. Спільними для обох видів користувачів буде функція для перегляду повідомлень та функція авторизації. Функції для

перегляду статистики відвідуваності та перегляду розкладу, також будуть спільними для студента та викладача, проте, надаватимуть їм різні можливості та матимуть деякі відмінності у інтерфейсі. Інші функції будуть доступними тільки викладачу. Відношення між користувачами та варіантами використання зображено на рисунку 1.24.


Рис. 1.24. Діаграма варіантів використання

Розглянемо більш детально кожний із варіантів використання. Робота з додатком розпочнеться із функції «Авторизація». Для того щоб отримати можливість користуватися додатком користувачу потрібно буде пройти процес авторизації. Лише після того як система розпізнає користувача він зможе отримати доступ до основного функціоналу користувача. Основним завданням цієї функції є визначення чи користувач має доступ до системи та визначення типу користувача. Також важливо щоб функція забезпечувала безпечне використання даних для авторизації та неможливість їх викрадення.

У таблиці 1.7 представлено опис варіанту використання «Авторизація».

Таблиця 1.7

Варіант використання «Авторизація»

| | |
|---|---|
| Контекст використання | Метою функції є ідентифікація користувача та надання або ненадання йому відповідних прав |
| Дійові особи | Студент або викладач |
| Передумова | – |
| Тригер | Перший запуск додатку |
| Сценарій | 1 – Надання даних потрібних для авторизації<br>2 – Натиснення кнопки підтвердження авторизації |
| Пост умова | Toast про успішність/неуспішність авторизації |

Логічний і концептуальний опис функціональних можливостей системи для сценарію «Авторизація», відображено на ескізі екранної форми, яка представлена на рисунку 1.25.

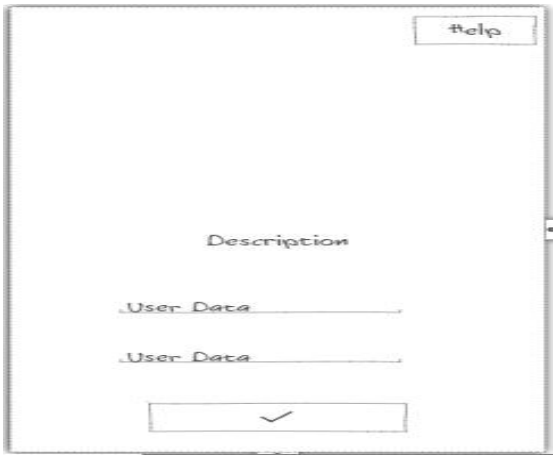


Рис. 1.25. Кадр для сценарію «Авторизація»

Далі розглянемо варіант використання «Переглянути розклад». Розклад має відображатися в основному вікні програми. В залежності від типу користувача розклад відображатиме потрібні для нього дані. Головною задачею цієї функції є формування та відображення особистого графіку занять користувача. У випадку викладача вибір заняття із списку передбачатиме відкриття нового вікна для додавання даних про присутність. Важливо забезпечити зручність перегляду розкладу.

У таблиці 1.8 представлено опис варіанту використання «Переглянути розклад».

Таблиця 1.8

Варіант використання «Переглянути розклад»

| | |
|---|---|
| Контекст використання | Метою функції є формування та відображення персонального розкладу користувача |
| Дійові особи | Студент або викладач |
| Передумова | Успішна авторизація |
| Тригер | Запуск додатку |
| Сценарій | 1 – Вибір дати та дня тижня<br>2 – Перегляд розкладу |
| Пост умова | Відображений розклад |

Логічний і концептуальний опис функціональних можливостей системи для сценарію «Переглянути розклад», відображено на ескізі екранної форми, яка представлена на рисунку 1.26.


Рис. 1.26. Кадр для сценарію «Переглянути розклад»

Варіант використання «Переглянути повідомлення» має бути реалізованим у багатьох аспектах. Основною задачею цієї функції є відображення користувачу повідомлень отриманих від системи. Можливість використання цієї функції повинна бути реалізована в основному меню. Крім

того, важливо реалізувати підтримку миттєвих нотифікацій, які дозволятимуть проінформувати користувача що в додатку для нього є нове повідомлення.

У таблиці 1.9 представлено опис варіанту використання «Переглянути повідомлення».

Таблиця 1.9

Варіант використання «Переглянути повідомлення»

| | |
|---|---|
| Контекст використання | Метою функції є показ повідомлень та механізм інформування користувача про нове повідомлення |
| Дійові особи | Студент або викладач |
| Передумова | Успішна авторизація |
| Тригер | Нове повідомлення |
| Сценарій | 1 – Відкрити список усіх повідомлень<br>2 – Переглянути необхідне повідомлення |
| Пост умова | Після перегляду повідомлення стають прочитаними |

Логічний і концептуальний опис функціональних можливостей системи для сценарію «Переглянути повідомлення», відображено на ескізі екранної форми, яка представлена на рисунку 1.27.


Рис. 1.27. Кадр для сценарію «Переглянути повідомлення»

На рисунку 1.28 має бути зображено кадр, який описує частину функції, що відповідатиме за показ безпосередньо самого повідомлення.


Рис. 1.29 – Кадр для сценарію «Переглянути повідомлення»

Варіант використання «Надіслати повідомлення», буде доступний лише викладачу. Ця функція має забезпечити можливість надіслати повідомлення будь-якому студенту якого навчає викладач.

У таблиці 1.10 представлено опис варіанту використання «Надіслати повідомлення».

Таблиця 1.10

Варіант використання «Надіслати повідомлення»

| | |
|---|---|
| Контекст використання | Метою функції є надсилання повідомлення студенту |
| Дійові особи | Викладач |
| Передумова | Успішна авторизація |
| Тригер | Вибір функції в меню |
| Сценарій | 1 – Надання даних потрібних для авторизації<br>2 – Натиснення кнопки підтвердження авторизації |
| Пост умова | Toast про успішність/неуспішність надсилання |

Логічний і концептуальний опис функціональних можливостей системи для сценарію «Надіслати повідомлення», відображено на ескізі екранної форми, яка представлена на рисунку 1.30.


Рис. 1.30. Кадр для сценарію «Надіслати повідомлення»

Далі розглянемо варіант використання «Додати дані про присутність студентів». Ця функція буде доступна лише для викладача, після вибору предмета із розкладу. Вона відображатиме список студентів із даними про них відносно вибраного предмету. Також, функція повинна надавати зручні засоби для внесення даних про присутність студентів. Крім того, функція має забезпечити обробку та збереження цих даних.

У таблиці 1.11 представлено опис варіанту використання «Додати дані про присутність студентів».

Таблиця 1.11

Варіант використання «Додати дані про присутність студентів»

| | |
|---|---|
| Контекст використання | Метою функції є внесення даних про присутність студентів на занятті |
| Дійові особи | Викладач |
| Передумова | Успішна авторизації |
| Тригер | Вибір предмету або запуск додатку під час заняття |
| Сценарій | 1 – Ввести всі потрібні дані<br>2 – Натиснути на кнопку підтвердження |
| Пост умова | Введені дані автоматично збережені, toast про успішність/неуспішність дії |

Логічний і концептуальний опис функціональних можливостей системи для сценарію «Додати дані про присутність студентів», відображено на ескізі екранної форми, яка представлена на рисунку 1.31.

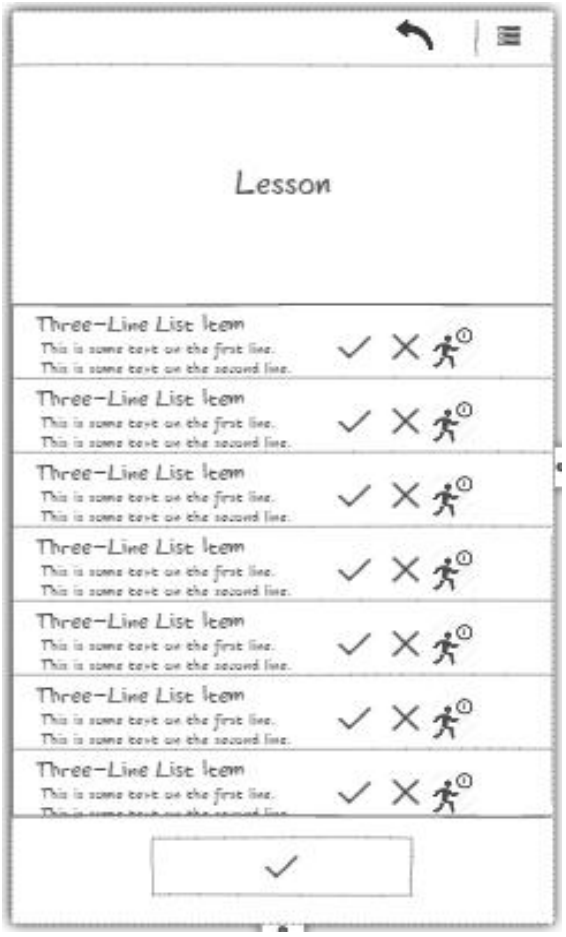


Рис. 1.31. Кадр для сценарію «Додати дані про присутність студентів»

Варіант використання «Редагувати дані про присутність студентів» буде схожим до попереднього, проте матиме свої особливості. Функція дозволятиме редагувати уже збережені дані та перезаписуватиме їх. Щоб функція була активною дані про присутність студентів на відповідному занятті мають бути уже введеними та збереженими. При редагуванні даних викладачу буде потрібно вказати причину внесення змін. Таким чином, викладач не матиме права редагувати дані коли йому заманеться без обґрунтованої на це причини.

У таблиці 1.12 представлено опис варіанту використання «Редагувати дані про присутність студентів».

Таблиця 1.12

Варіант використання «Редагувати дані про присутність студентів»

| | |
|---|---|
| Контекст | Метою функції є надання можливості редагування уже |

| | |
|---|---|
| використання | збережених даних |
| Дійові особи | Викладач |
| Передумова | Додано дані про присутність студентів» |
| Тригер | Вибір предмету із збереженими даними |
| Сценарій | 1 – Введення потрібних даних<br>2 – Натиснення кнопки підтвердження<br>3 – Введення причини змін<br>4 – Натиснення кнопки підтвердження |
| Пост умова | Toast про успішність/неуспішність дії |

Логічний і концептуальний опис функціональних можливостей системи для сценарію «Редагувати дані про присутність студентів», збігається з ескізом екранної форми для варіанту використання «Додати дані про присутність студентів». Тобто, при редагуванні даних користувач здійснить новий ввід даних, а старі данні при цьому будуть перезаписані. Відмінність функції редагування від варіанту використання «Додати дані про присутність студентів» полягатиме в тому, що користувач буде змушений пояснити причину внесення правок, яка відповідно теж буде збереженою разом із новими даними.

Відмінність між варіантом використання «Редагувати дані про присутність студентів» та варіантом використання «Додати дані про присутність студентів» відображено на рисунку 1.32.


Рис. 1.32. Кадр для сценарію «Authorization»

Варіант використання «Переглянути статистику відвідуваності» має бути доступним для викладача та студента. Основним завданням функції є надання засобів для перегляду статистичних даних різного вмісту. Користувачу має

надаватися можливість вибрати вид статистики. Також, функція має забезпечити коректне відображення статистичних даних.

У таблиці 1.13 представлено опис варіанту використання «Переглянути статистику відвідуваності».

Таблиця 1.13

Варіант використання «Переглянути статистику відвідуваності»

| | |
|---|---|
| Контекст використання | Метою функції є можливість перегляду різних статистичних даних |
| Дійові особи | Студент або викладач |
| Передумова | - |
| Тригер | Перший запуск додатку |
| Сценарій | 1 – Надання даних потрібних для авторизації<br>2 – Натиснення кнопки підтвердження авторизації |
| Пост умова | Toast про успішність/неуспішність авторизації |

На рисунку 1.33 зображено ескіз що описує частину функції «Переглянути статистику відвідуваності» та дозволяє вибрати вид потрібної статистики.

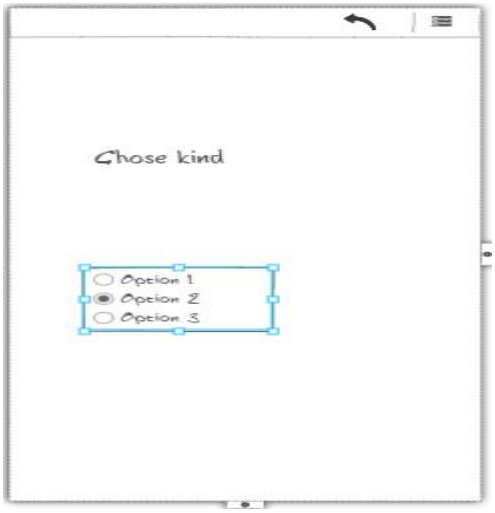


Рис. 1.33. Кадр для сценарію «Переглянути статистику відвідуваності»

На рисунку 1.34 зображено кадр що відповідає за відображення статистичних показників.

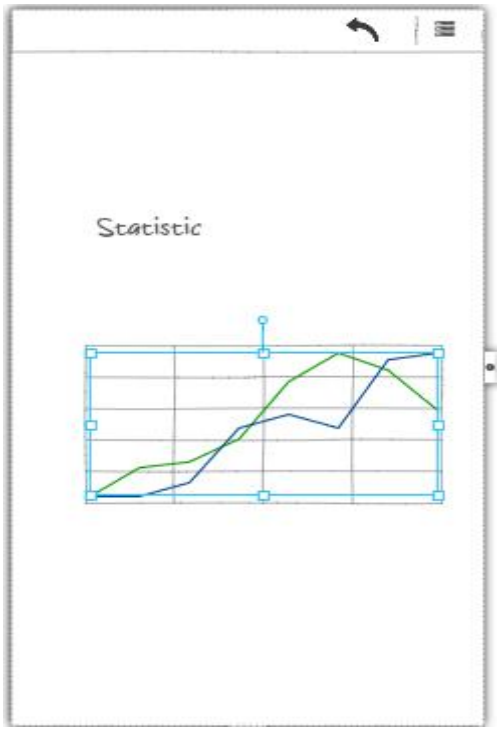


Рис. 1.34. Кадр для сценарію «Переглянути статистику відвідуваності»

Після детального розгляду варіантів та їх розкадрування можна скласти специфікацію функціональних вимог, яка наведена в таблиці 1.14.

Таблиця 1.14

Специфікація функціональних вимог

| Ідентифікатор вимоги | Назва вимоги | Атрибути вимог | | |
|---|---|---|---|---|
| | | Пріоритет | Складність | Контакт |
| 1. | Авторизація | Обов’язково | Середня | Викладач, студент |
| 2. | Перегляд розкладу | Обов’язково | Низька | Викладач, студент |
| 3. | Переглянути повідомлення | Обов’язково | Висока | Викладач, студент |
| 4. | Надіслати повідомлення | Обов’язково | Висока | Викладач, студент |

Продовження таблиці 1.14

| | | | | |
|---|---|---|---|---|
| 5. | Додати дані про присутність студентів | Обов’язково | Висока | Викладач, студент |
| 6. | Редагувати дані про присутність студентів | Обов’язково | Висока | Викладач, студент |
| 7. | Переглянути статистику відвідуваності | Обов’язково | Висока | Викладач, студент |

Специфікація нефункціональних вимог наведена у таблиці 1.15.

Таблиця 1.15

Специфікація нефункціональних вимог

| Ідентифікатор вимоги | Назва вимоги | Атрибути вимог | | |
|---|---|---|---|---|
| | | Пріоритет | Складність | Контакт |
| 1. | Застосовність | Рекомендовано | Висока | Викладач, студент |
| 1.1. | Час, необхідний для навчання звичайних і досвідчених користувачів | Обов’язково | Середня | Викладач, студент |
| 1.2. | Час відгуку для типових завдань | Рекомендовано | Висока | Викладач, студент |
| 1.3. | Вимоги по відповідності загальним стандартам застосовності | Рекомендовано | Висока | Викладач, студент |
| 2. | Надійність | Обов’язково | Висока | Викладач, студент |
| 2.1. | Точність | Обов’язково | Висока | Викладач, студент |
| 3. | Робочі характеристики | Обов’язково | Середня | Викладач, студент |

Продовження таблиці 1.15

| | | | | |
|---|---|---|---|---|
| 3.1. | Швидкодія для транзакції | Обов’язково | Висока | Викладач, студент |
| 3.2. | Продуктивність | Обов’язково | Середня | Викладач, студент |
| 3.3. | Використання ресурсів | Опційно | середня | Викладач, студент |
| 4. | Вимоги до документації | Обов’язково | середня | Викладач, студент |

Результатом визначення та розгляду функціональних та нефункціональних вимоги є сформована специфікація вимог до розроблюваного Android-додатку для обліку відвідуваності занять студентами. Після виконання цього етапу можна переходити безпосередньо до проектування системи.

# РОЗДІЛ ІІ

# ПРОЕКТУВАННЯ

## 2.1. Розробка архітектури програмної системи

Після розгляду функціональних та нефункціональних вимог і формування специфікації можна перейти до етапу проектування. Оскільки, планується, що розроблюваний Android-додаток буде підсистемою веб-орієнтованого продукту, то, перш за все, потрібно розглянути механізми, які будуть використовуватися для інтеграції двох систем. Враховуючи особливості платформи Android, у такому випадку найкраще буде використовувати архітектуру, яка буде наслідувати архітектуру REST або принаймні буде схожою на неї. Такий підхід дозволить Android-додатку отримати доступ до бази даних не через пряме підключення, а через HTTP протокол. При цьому для отримання даних з БД та їх обробки будуть використовуватися ресурси сервера веб-орієнтованої системи. Таким чином, підхід до проектування, що базуватиметься на RESTful архітектурі дозволить легко відділити клієнтський додаток від програмних інтерфейсів для роботи з базою даних, що є беззаперечною перевагою для системи, яка передбачає інтеграцію з іншими системами[1]. Фактично, розроблюваний додаток стане не прив’язаним ані до структури БД, ані до використовуваної РСУБД. Що, в свою чергу наддасть можливість легко перепроектовувати систему для аналогічних веб-орієнтованих програмних систем, що вирішують проблему обліку відвідуваності студентів. Для цього потрібно буде лише внести певні коригування у API для роботи з БД. В перспективі розроблювану систему, можна було б зробити більш універсальною та гнучкою і забезпечити її швидку інтеграцію фактично з будь-якою веб-орієнтованою системою для обліку відвідуваності. Слід зазначити, що використання RESTful підходу має й багато мінусів основним з яких є використання посередника між клієнтом та сервером БД. Для розроблюваного Android-додатку ці мінуси не є критичними чи дуже важливими, оскільки додаток має забезпечити роботу основного функціоналу й

у режимі «Offline», а для цього доведеться зберігати усі потрібні данні у локальному сховищі.

Ще одною надзвичайно важливою причиною для вибору RESTful архітектури є те, що для забезпечення стабільної роботи функціоналу потрібне використання певного серверу БД та сервісу Google Cloud Messaging. Для забезпечення функцій надсилання, отримання та генерування миттєвих повідомлень GCM повинен мати доступ до бази даних, який фактично не можливо надати без використання веб-серверу.

Отже для забезпечення потрібного функціоналу та стабільної роботи Android-додатку для обліку відвідуваності занять студентами потрібне використання трьох серверів. Зв'язок між усіма вузлами та компонентами проектованої системи відображено на рисунку 2.1.

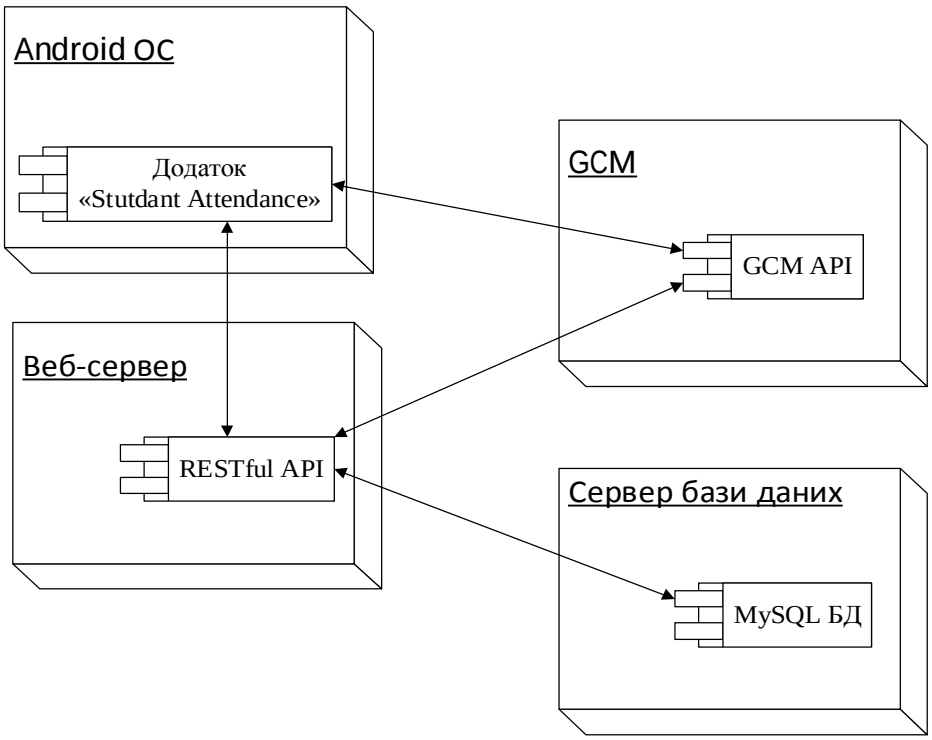


Рис. 2.1. Діаграма розгортання розроблюваної програмної системи

Розглянемо детальніше залежності між компонентами програмного забезпечення для повнішого представлення статичного аспекту архітектури додатку.

В системі буде два пакети, кожний із яких відповідатиме за функціонал для відповідного класу користувача. Крім того, в системі будуть окремі модулі відповідальні за більшість операцій з даними та за можливість роботи з веб-сервером через використання мережевого протоколу. Ці модулі будуть використовуватися обома вищезгаданими пакетами. Залежності між описаними компонентами відображено на рисунку 2.2.

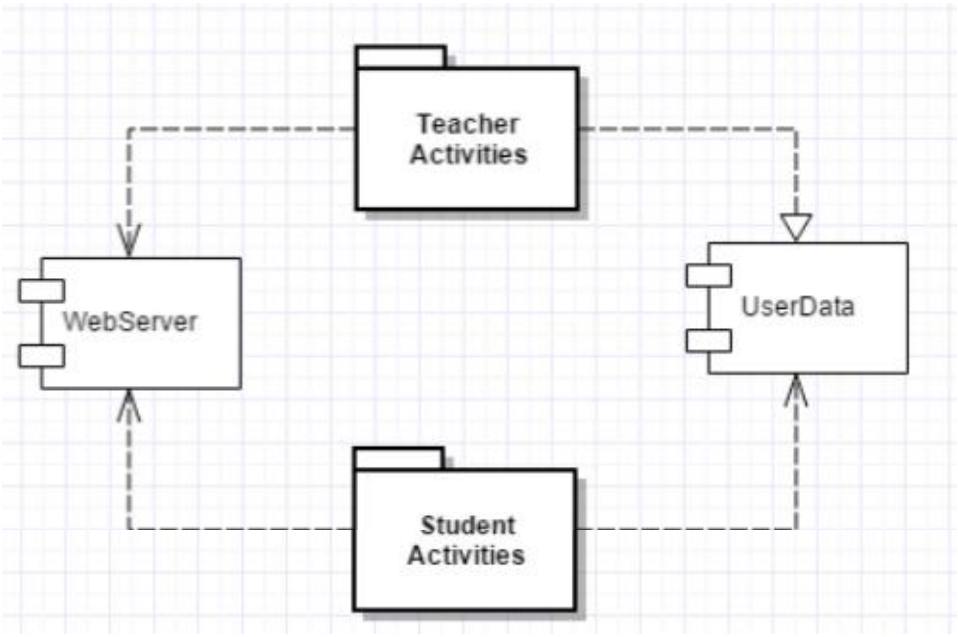


Рис. 2.2. Діаграма компонентів розроблюваної програмної системи

Структурна модель додатку є досить складною. Більшість модулів системи відповідатимуть за інтерфейс або стосуватимуться особливостей забезпечення роботи з платформою Android. Тому їх відображення на даному етапі, є не потрібним оскільки воно зробить діаграму класів занадто громіздкою та важкою для читання та розуміння. Отже, на рисунку 2.3 діаграма відображена лише тими класами які забезпечуватимуть виконання бізнес-логіки, яку можливо відділити від графічної частини та частини для організації роботи з даними.

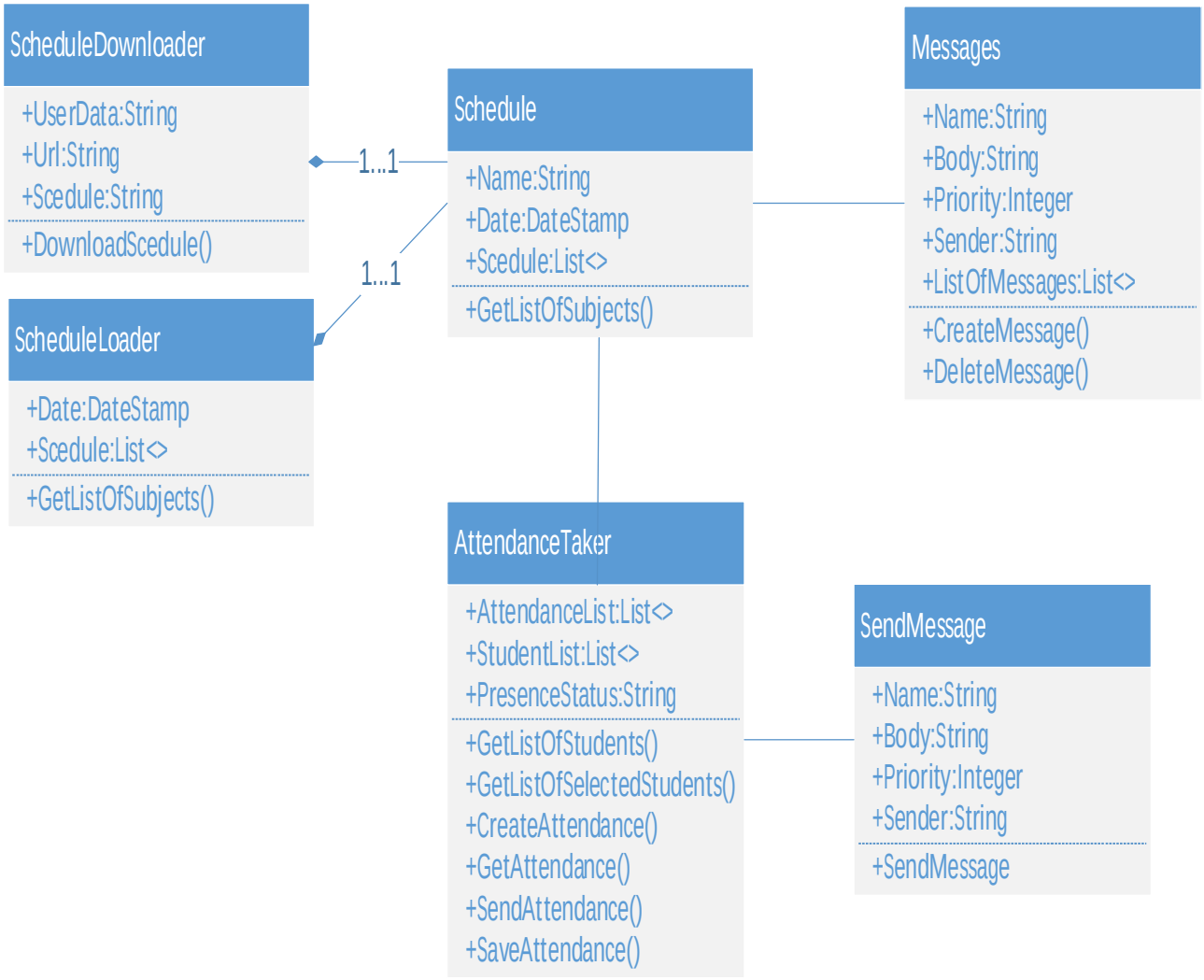


Рис. 2.3. Діаграма класів

Наведена вище діаграма класів фактично показує структуру модуля UserData. Кожний із класів буде використовуватися для роботи з даними та забезпечуватиме певну бізнес-логіку, яку можливо відділити від елементів, що забезпечуватимуть графічне представлення. Відображення інших класів, що відповідатимуть за частину функціоналу не доцільне, оскільки вони будуть тісно пов'язані із класами що відповідатимуть за графічний інтерфейс користувача.

Розглянемо детальніше динамічну складову архітектури системи. Для цього графічно опишемо основні бізнес-процеси та бізнес-правила, закладені в програмну систему, за допомогою засобів мови моделювання UML[2].

На рисунку 2.4, зображена діаграма послідовності для варіанту використання «Надсилання повідомлення», яка відображає взаємодії об'єктів впорядкованих за часом.

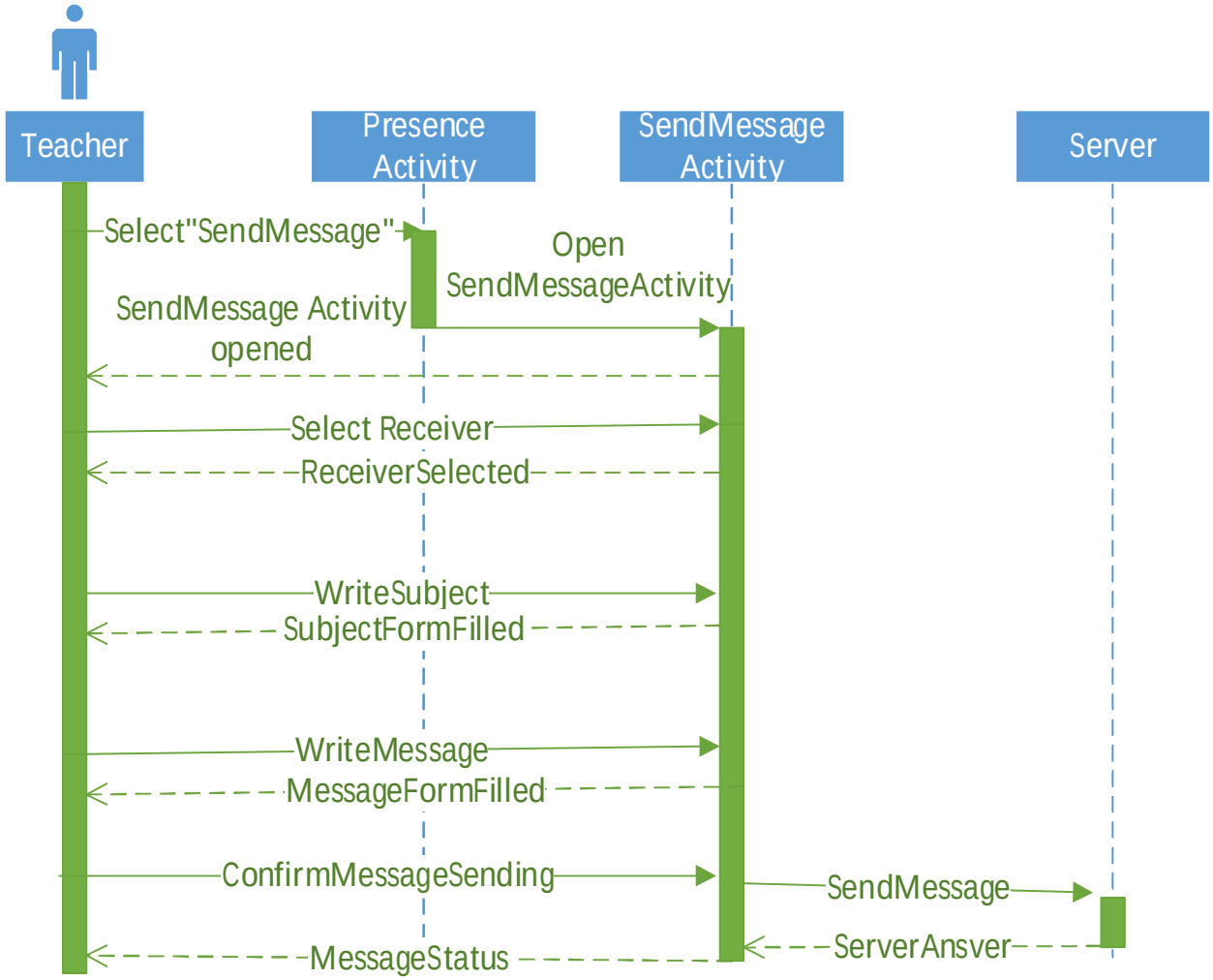


Рис. 2.4. Діаграма послідовності для функції «Надсилання повідомлення»

Ця діаграма ілюструє процес надсилання повідомлення викладачем та показує коли і протягом якого часу будуть активні графічні інтерфейси. Вданому випадку зображено перехід до activity «SendMessage», не із основного меню а з activity «Precense», яке відповідає за збір даних про відвідуваність.

Після основного вікна для надсилання повідомлення викладачу буде потрібно ввести необхідні дані та підтвердити надсилання повідомлення. Якщо всі поля будуть заповнені вірно, activity «SendMessage» відправить повідомлення на сервер.

Процес надсилання повідомлення демонструє діаграма станів яка зображена на рисунку 2.5.

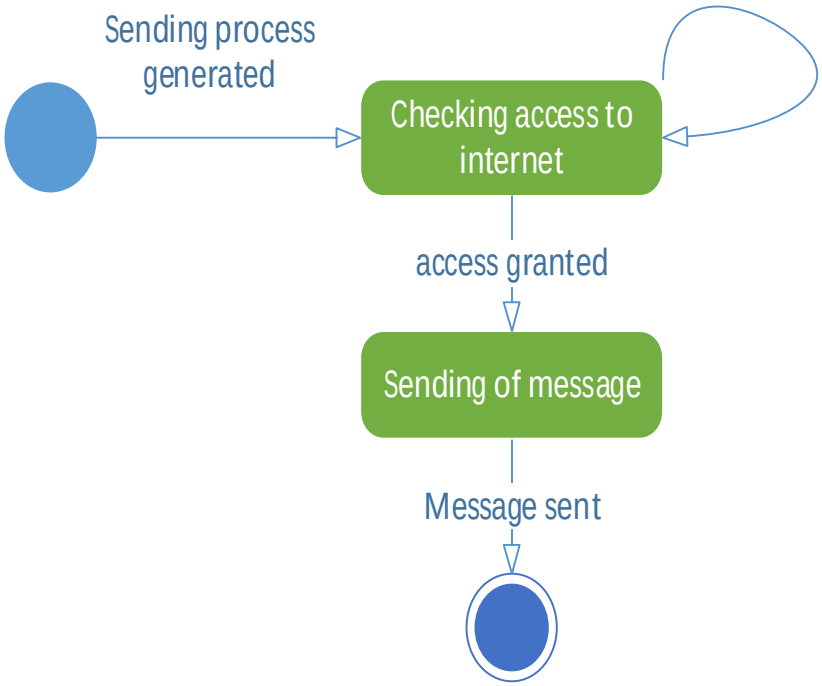


Рисунок 2.5 –Діаграма станів для системного процесу «Надсилання повідомлення»

Важливим етапом є відображення нових повідомлень. При отримані сповіщення про нові повідомлення користувач матиме змогу прочитати повідомлення безпосередньо відкривши додаток або натиснувши на сповіщення, що автоматично запустить додаток. Цей процес описано за допомогою діаграми станів зображеної на рисунку 2.6.

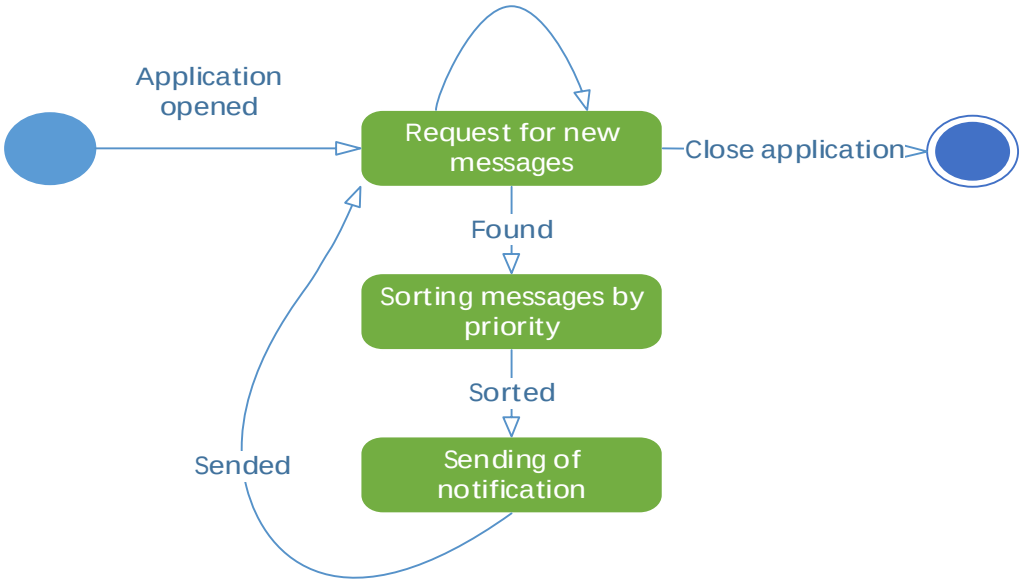


Рис. 2.6. Діаграма станів для функції «Перевірка наявності нових повідомлень»

На рисунку 2.7 зображена діаграма послідовності, яка відображає взаємодії графічних об'єктів впорядкованих за часом для варіанту використання «Перегляд статистики» та його можливості. Функція буде доступною тільки з головного меню та працюватиме лише тоді, коли буде доступ до мережі Інтернет.

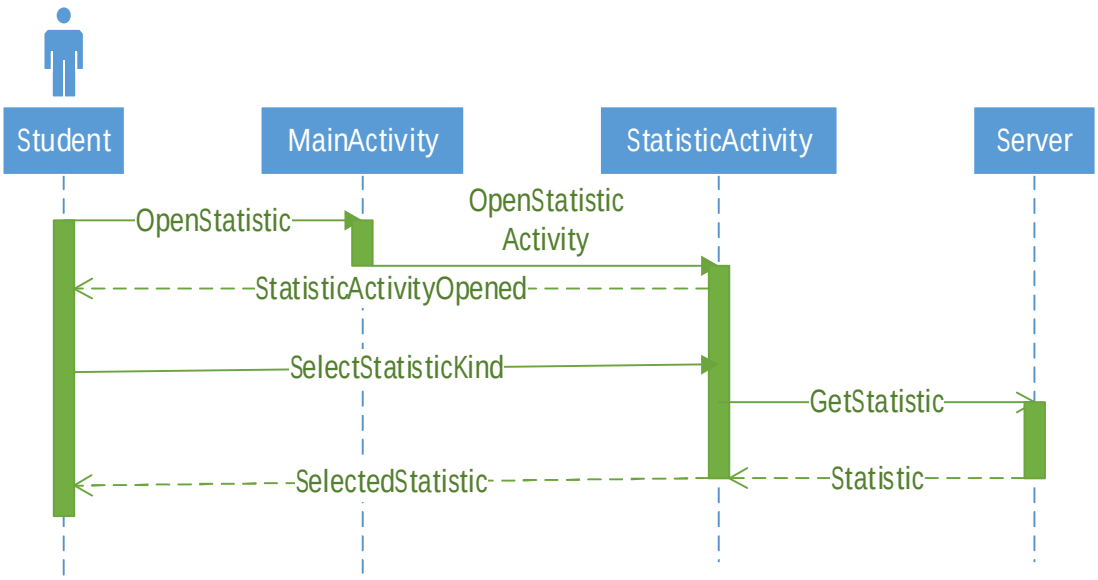


Рис. 2.7. Діаграма послідовності для функції «Перегляд статистики»

Далі розглянемо одну із найважливіших функцій для викладача «Створення списку присутніх», яка відповідатиме за збір даних про відвідуваність. Доступ до цієї опції можна буде отримати при виборі предмету із персонального розкладу у головному вікні програми. Функція відображатиме список студентів для обраного заняття та забезпечуватиме збір даних. Дані про студентів будуть завантажуватися із локального сховища. При відкриті програми під час заняття автоматично відкриватиметься список студентів для відповідного заняття, щоб викладач міг швидко відмітити присутніх. Цей процес відображено за допомогою діаграми послідовності на рисунку 2.8.

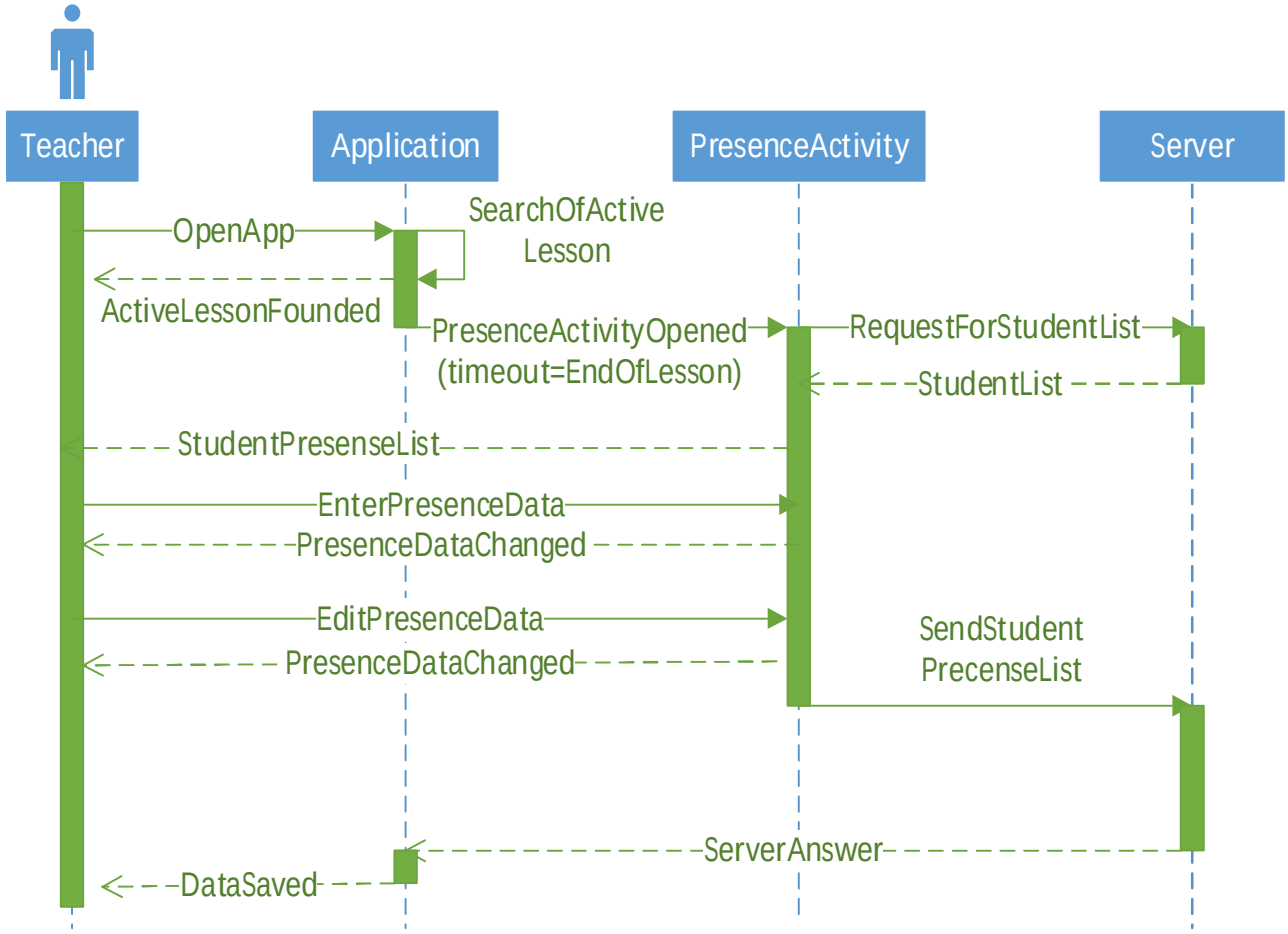


Рис. 2.8. Діаграма послідовності для функцій «Створення списку присутніх»

Для кращого розуміння механізму «Створення списку присутніх», його слід розглянути із боку системних процесів, що будуть відповідальні за його

реалізацію. Дії системних процесів показує діаграма станів, яка зображена на рисунку 2.9.

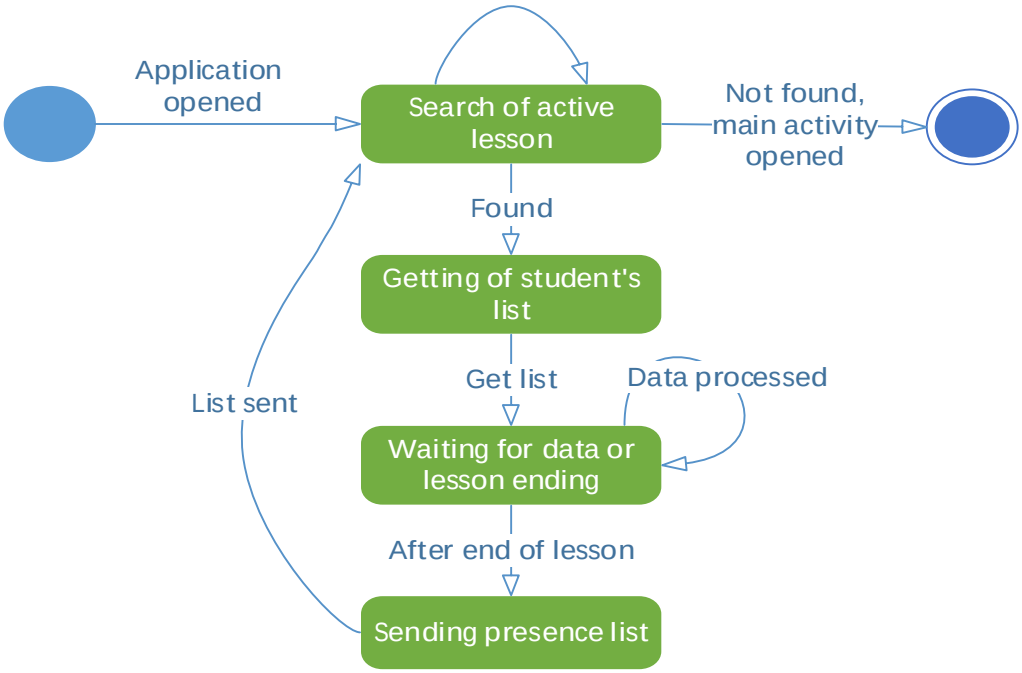


Рис. 2.9. Діаграма станів для функції «Створення списку присутності»

Механізм надсилання списку відображено на рисунку 2.10

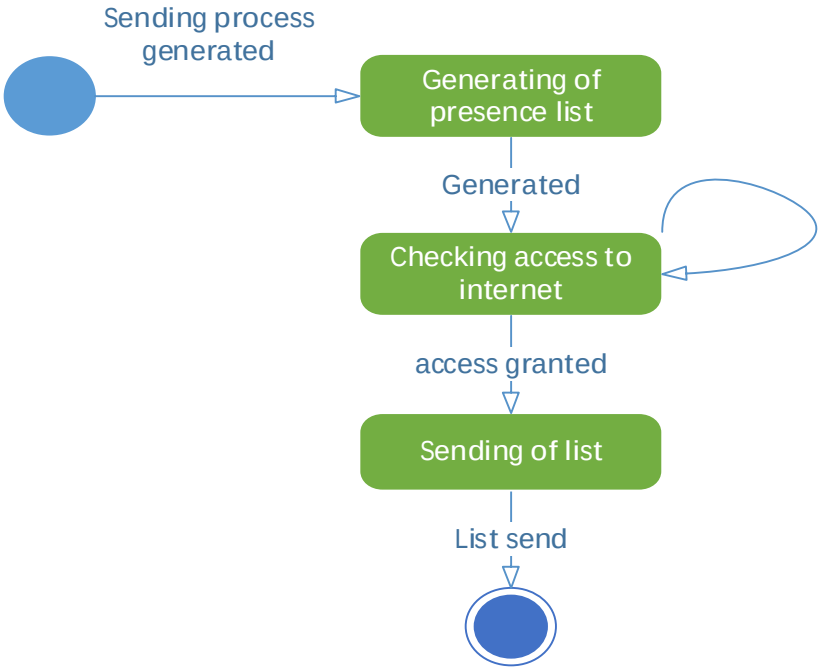


Рис. 2.10. Діаграма станів для функції «Надсилання списку присутності»

Важливою вимогою є забезпечення безпечної авторизації користувачів. Для цього найкраще буде реалізувати механізм подібний до подвійної автентифікації. Після введення користувачем персонального ідентифікатора на його зареєстрований у системі email буде відправлятися тимчасовий пароль, який використовуватиметься для підтвердження прав доступу до додатку. Цей процес відображено за допомогою діаграми послідовності на рисунку 2.11.

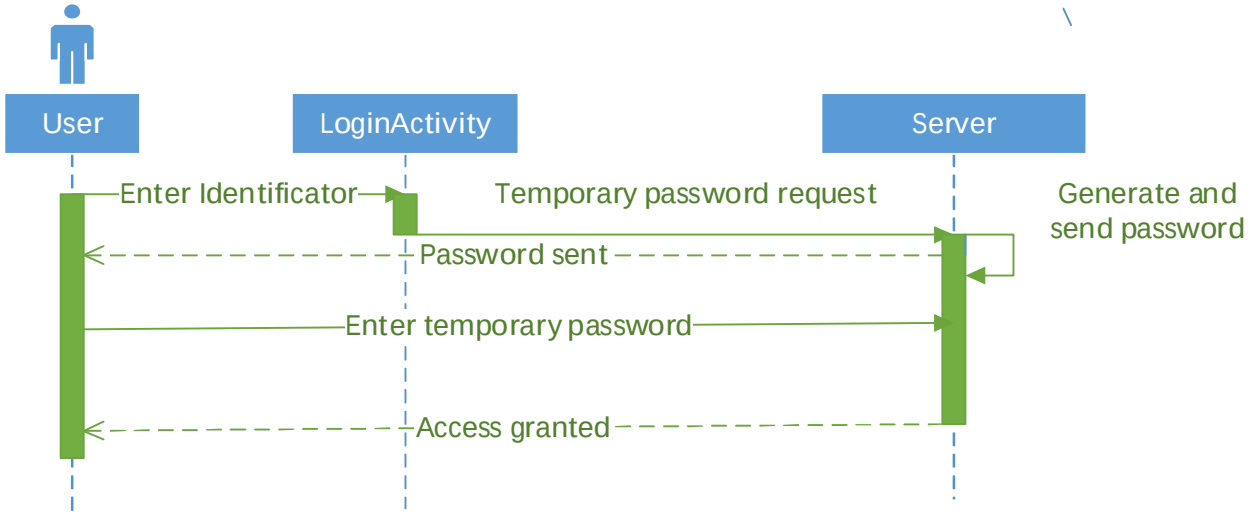


Рис. 2.11. Діаграма послідовності для функції «Авторизація»

Основні функції, що забезпечуватимуть чітке виконання вимог до системи було розглянуто вище. Решта функціоналу системи буде відповідати за обробку даних та представлення даних користувачу, тому його детальний розгляд є недоцільним. Оскільки, реалізація цих функцій практично не матиме ніякого впливу на бізнес-логіку та бізнес-правила механізми їх реалізації будуть обрані безпосередньо розробниками проекту.

## 2.2. Проектування структури бази даних

Зважаючи на те, що розроблювана система повинна бути інтегрованою із веб-орієнтованим продуктом, потрібно розробити рекомендаційну структуру бази даних, яка буде містити дані які будуть необхідні для роботи Android-

додатку. Оскільки, система буде побудована на базі RESTful архітектури, зрозуміло, що навіть та частина БД, яка буде потрібна для підтримки функціонування додатку допускатимуться певні відмінності від спроектованої бази. Майже усі відмінності можна буде нівелювати, як засобами самої РСУБД, так і засобами веб-сервера, який надаватиме інтерфейс для доступу до БД. У якості РСУБД для цього проекту буде використовуватися MySQL.

Враховуючи той фактор, що більшість інформації, яку використовуватиме Android-додаток буде зберігатися у локальному сховищі даних потрібно детальніше розглянути методи доступу до глобальної бази даних.

Для того щоб отримати доступ до глобальної БД додаток надсилатиме запити через протокол HTML до спеціального веб-застосунку, який розташований на веб-сервері, передаючи йому при цьому необхідну інформацію. Веб-застосунок, опрацьовуючи запити додатку, підключатиметься до сервера з БД для отримання необхідних даних, після чого, оброблятиме дані перетворивши їх у потрібний формат та надсилатиме клієнту-додатку відповідь із даними у вигляді JSON. Отримані дані додаток збереже у Shared Preferences.

Після проведеного аналізу предметної області можна визначити дані, які будуть необхідні додатку та визначити, які дані додаток буде повинний зберігати в локальному сховищі.

Для функціонування додатку у режимі «Offline» обов’язково потрібно зберігати у локальному сховищі дані що репрезентуватимуть персональний розклад користувача та дані про уже отримані повідомлення. У випадку викладача необхідно також буде зберігати списки студентів. Усі операції із статистичними даними працюватимуть тільки в режимі «Online». Дані про відвідуваність, які вводитиме викладач будуть зберігатися локально та відправлятися на сервер при появі стабільно підключення до мережі Інтернет.

На етапі експлуатації системи для функціонування Android-додатку будуть використовуватися наступні об’єкти та дані про них:

1) Предмет у розкладі:

- Назва;
- Тип;
- Викладач;
- Група;
- Аудиторія;
- День тижня коли проходять заняття;
- Дати занять;
- Час коли відбувається заняття;
- Тривалість заняття;
- Дані про парність/непарність тижня.

2) Студент:
   - Ім’я;
   - Прізвище;
   - Email;
   - Курс;
   - Дані про присутність на заняттях;
   - Назва групи;
   - Назва факультету.
3) Викладач:
   - Ім’я;
   - Прізвище;
   - Email.
4) Повідомлення:
   - Тема;
   - Час створення;
   - Статус;
   - Пріоритет;
   - Відправник;
   - Отримувач.
5) Користувач;

- GCM ID;
- Тип доступу;
- Пароль.

Після опису вхідної та вихідної інформації, яка обробляється в рамках функцій предметної області розроблюваної програмної системи можна зобразити глобальну даталогічну модель даних у вигляді логічної моделі ERD на рисунку 2.12.

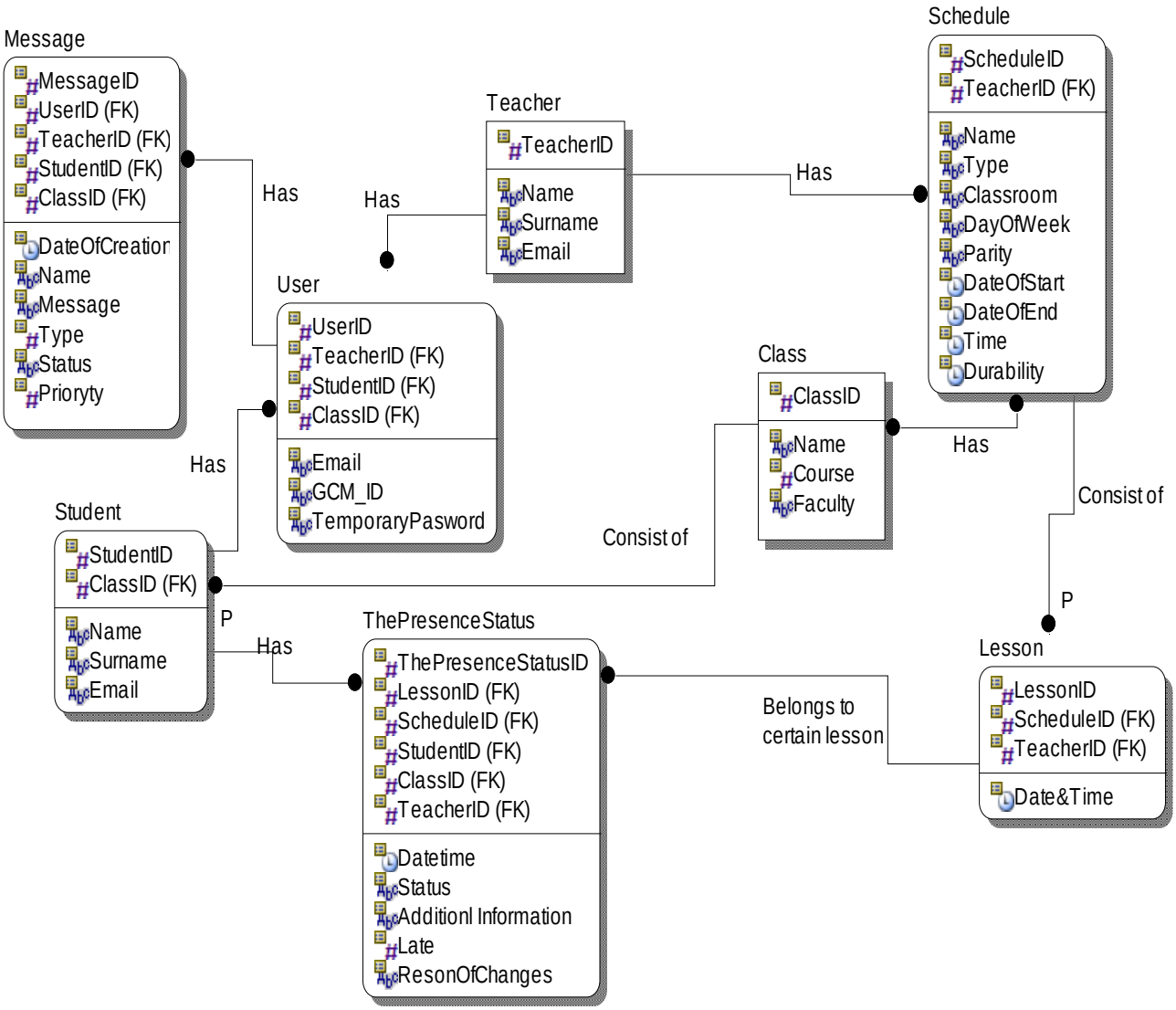


Рис. 2.12. Логічна модель ERD

Для всіх відношень автоматично створюються індекси по первинному ключі.

Індексування значно пришвидшує пошук даних у БД. Проте надмірне використання індексування призводить до втрати продуктивності виконання запитів на поновлення та знищення[3].

Для ефективного пошуку передбачено створення foreign keys для наступних атрибутів:

1) ClassID - зовнішній ключ
2) TeacherID- зовнішній ключ
3) LessonID- зовнішній ключ
4) ScheduleID- зовнішній ключ
5) PresenceStatusID - зовнішній ключ

Також передбачається, що атрибут Email буде індексованим.

Для забезпечення правильного збереження даних на етапі надсилання даних про відвідуваність передбачається використання транзакцій.

При написані інтерфейсу комунікації БД із веб-сервером розробники повинні будуть використовувати Stored Procedures, для нестандартних операцій з даними БД.Для можливості генерування автоматичних повідомлень базою даних будуть створені спеціальні тригери.Результатом фізичного проектування є DDL код, який надано в додатку X.

Після побудови логічної моделі та аналізу даних можна побудувати фізичну модель ERD, яка зображена на рисунку 2.13.

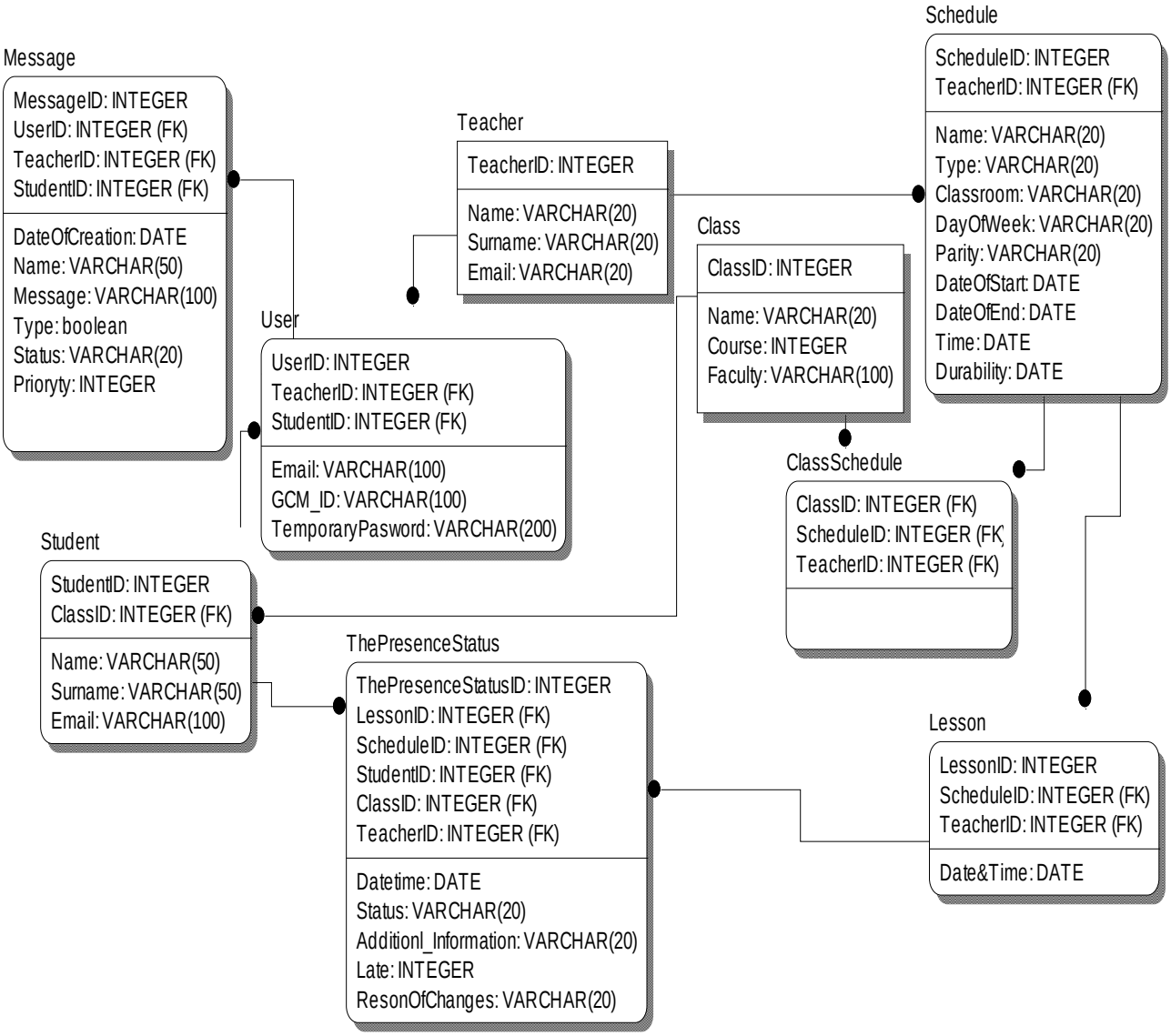


Рис. 2.12. Фізична модель ERD

# РОЗДІЛ ІІІ

# ПРОГРАМНА РЕАЛІЗАЦІЯ

## 3.1. Програмна реалізація проекту

Найкращим вибором серед мов для написання додатку для мобільної платформи Android сьогодні є Java. Java є широко розповсюдженою мовою програмування, яка має багато переваг. Основною перевагою використання мови Java у платформі Android є те, що вона виконується на віртуальній машині і тому немає необхідності перекомпільовувати додатки для кожного телефону.

Додаток, що розробляється, буде підтримуватися на усіх версіях операційної системи Android починаючи із SDK 16(тобто версії Android 4.1.2). Це рішення можна обґрунтувати тим, що підтримувати сумісність усіх функцій системи із попередніми застарілими версіями Android буде досить складно та недоцільно, оскільки за останніми статистичними даними ці версії використовують менш ніж 5% користувачів платформи Android і число таких користувачів продовжує суттєво скорочуватися.

Для написання веб-сервісів, які забезпечуватимуть доступ до глобальної бази даних, оптимальним вибором є використання мови PHP, яка має достатню кількість засобів, щоб швидко та якісно розв’язати проблему доступу до БД та має достатню кількість уже готових фреймворків для створення REST API та роботи з ним. Для написання REST API використано мікрофреймворк Slim.

Основним завданням веб-сервісів, які надають додатку засоби для комунікацій із БД на сервері, є можливість отримання та надсилання даних через мережевий протокол HTTP. Для забезпечення зручної та швидкої роботи із даними, усі дані пересилаються у форматі JSON. Частина коду, яка наведена нижче демонструє частину API для роботи із базою даних та забезпечує можливість отримання викладачем списку студентів із глобальної БД. За допомогою методу POST додаток пересилає потрібні ідентифікаційні дані, після чого скрипт витягує потрібні дані із БД, форматує їх та надсилає додатку.

```
$con=connect();
if ($con!=null) {
mysqli_set_charset($con,"utf8");
$ID=mysqli_real_escape_string($con,$_POST['ID']);

    $stmt = mysqli_prepare($con, 'CALL GetStudents(?)');
    mysqli_stmt_bind_param($stmt, 'i', $ID);
    mysqli_stmt_execute($stmt);
    $result = $stmt->get_result();
close($con);
if (mysqli_num_rows($result) > 0) {
    $response["Classes"] = array();
    $className=null;
    while ($row = mysqli_fetch_array($result)) {
        if($className==null)
        {
            $className=$row["ClassID"];
            $Class[$className]=array();
        }
        else if($className!=$row["ClassID"])
        {
            $className=$row["ClassID"];
            $Class[$className]=array();
        }
        $student=array();
        $student["ID"]=$row["ID"];
        $student["Name"]=$row["Name"];
        $student["Surname"]=$row["Surname"];
        $student["Email"]=$row["Email"];
        array_push($Class[$className] , $student );
```

```
    }
     array_push($response["Classes"] , $Class);
    $response["success"] = 1;

    echo json_encode($response,JSON_UNESCAPED_UNICODE);
} else {
    $response["success"] = 0;
    $response["message"] = "User not exist";
    echo json_encode($response,JSON_UNESCAPED_UNICODE);
    }
}
else {
    $response["success"] = 0;
    $response["message"] ="Connection failed ";
    echo json_encode($response,JSON_UNESCAPED_UNICODE);
}}
else {
    $response["success"] = 0;
    $response["message"] = "Required field(s) is missing";
    echo json_encode($response,JSON_UNESCAPED_UNICODE);   }
```

Усі процеси надсилання та отримання даних додатком виконуються у бекграунді за допомогою спеціального native класу AsyncTask. Під час процесу надсилання чи завантаження даних користувач бачитиме повідомлення про зайнятість додатку "Завантаження даних, зачекайте будь ласка...".

Для забезпечення стабільної роботи із проколом HTTP використовується спеціальна бібліотека OKHTTP. Ця бібліотека підтримує усі сучасні засоби для зручної роботи із протоколом HTTP. Для отримання даних додатком та

коректного їх опрацювання у форматі JSON був розроблений спеціальний клас, лістинг коду якого наведено нижче.

```
public class JSONParser {
     public static final MediaType JSON =
MediaType.parse("application/json; charset=utf-8");

     OkHttpClient client = new OkHttpClient();
     static InputStream is = null;
     static JSONObject jObj = null;
     static String jsonAns = null;

     JSONObject postRequest(String url, RequestBody json) throws
IOException {
          try {

               Request request = new
Request.Builder().url(url).post(json).build();
               Response response =
client.newCall(request).execute();
               jsonAns = response.body().string();
          } catch (UnsupportedEncodingException e) {
               Log.e("Er:", Log.getStackTraceString(e));
          } catch (IOException e) {
               Log.e("Er:", Log.getStackTraceString(e));
          } catch (Exception e) {
               Log.e("Er:", Log.getStackTraceString(e));
          }
          try {
               jObj = new JSONObject(jsonAns);
          } catch (JSONException e) {
```

```
                Log.e("JSON Parser", "Error parsing data " +
e.toString());
            }
            return jObj;
        }
        }
```

Для локального збереження даних використовується спеціальний файл SharedPrefernce формату xml. Цей файл зберігається у захищеній пам'яті пристрою Android та дозволяє зберігати дані у форматі key-value. У фрагменті коду наведеному нижче продемонстровано завантаження даних із SharedPreferences та їх обробка для подальшої можливості роботи із потрібними даними.

```
protected String doInBackground(String... args) {
try {
sPref = this.getSharedPreferences("com.studentattendance",
Context.MODE_PRIVATE);
            try {
                days = new JSONArray(sPref.getString("Schedule",
""));
            } catch (JSONException e) {
                Log.e("Er:", Log.getStackTraceString(e));
            }

                    for (int i = 0; i < days.length(); i++) {
                        JSONObject day = days.getJSONObject(i);
                        Log.e("Er:", day.toString());
                        JSONArray d = day.getJSONArray(dayOfWeek);
                        for (int j = 0; j < d.length(); j++) {
```

```
                                JSONObject subject =
d.getJSONObject(j);


                                String id = subject.getString("ID");
                                String name =
subject.getString("Name");
                                String dayOfWeek =
subject.getString("DayOfWeek");
                                String classroom = "Classroom:" +
subject.getString("Classroom");
                                String time =
subject.getString("Time");

HashMap<String, String> map = new HashMap<String, String>();
                                map.put("ID", id);
                                map.put("Name", name);
                                map.put("Classroom", classroom);
                                map.put("DayOfWeek", dayOfWeek);
                                map.put("Time", time);
                                scheduleList.add(map);


                            }
                        }
                    } catch (JSONException e) {
                        Log.e("Er:", Log.getStackTraceString(e));
                    }
                    return null;
                }
```

Для організації зручного інтерфейсу користувача додаток використовуватиме стандартні елементи інтерфейсу, які прийнято використовувати для платформи Android. Для забезпечення зручності

інтерфейсу та його презентабельного вигляду практично усі елементи було кастомізовано. Усі іконки, які використовуються в додатку є стандартними та знайомими для користувачів Android. Основне вікно додатку, яке дозволяє переглянути користувачам їх особистий розклад, містить елементи керування у меню, які дозволяють переключатися між днями розкладу. Крім того, у основному вікні можна переміщуватися між днями за допомогою swipe або можна вибрати конкретну дату розклад для якої потрібно відобразити. У тому випадку, коли користувач є викладачем він зможе вибрати предмет в розкладі та заповнити дані про відвідуваність. У всіх вікнах додатку присутнє меню у формі Action Bar та працює кнопка «назад», яка дозволяє повернутися до попереднього вікна або попереднього стану вікна.

Вікна, які дозволятимуть користувачам переглядати статистику, функція отримання нових повідомлень та функція надсилання даних про відвідуваність буде працювати тільки в тому випадку якщо пристрій матиме стабільне підключення до мережі Інтернет. За перевірку підключення до мережі відповідає клас InternetConnectionChecker, лістинг коду цього класу наведено нижче.

```
public class InternetConnectionChecker {
     Context _context;
     public InternetConnectionChecker(Context context) {
          _context = context;
     }
     public boolean isOnline() {
          ConnectivityManager cm = (ConnectivityManager)
_context.getSystemService(Context.CONNECTIVITY_SERVICE);
          NetworkInfo netInfo = cm.getActiveNetworkInfo();
          return netInfo != null &&
netInfo.isConnectedOrConnecting();
     }
     }
```

Для того, щоб працювали push notification, які сповіщатимуть про нове повідомлення миттєво, використовується сервіс GCM. Нижче відображено структуру повідомлення, яке надсилатиметься через GCM.

'message' => 'here is a message. message',

'title' => 'This is a title. title',

'vibrate' => 1,

'sound' => 1,

'created_at' => '2016-05-25 12:00:00'

Після відкриття отриманої нотифікації автоматично буде запущено додаток де користувач зможе отримати нотифікацію. Нотифікації отримуватимуть навіть коли додаток або пристрій знаходитимуться в неактивному стані. Єдиною вимою для роботи миттєвих повідомлень є підключення до Інтернету. Нижче наведено код функції, яка виконує обробку отриманого повідомлення.

```
@Override
public void onMessageReceived(String from, Bundle bundle) {
    String title = bundle.getString("title");
    String message = bundle.getString("message");
    String image = bundle.getString("image");
    String timestamp = bundle.getString("created_at");


    if
(!NotificationUtils.isAppIsInBackground(getApplicationContext()))
{
        Intent pushNotification = new
Intent(Config.PUSH_NOTIFICATION);
        pushNotification.putExtra("message", message);

LocalBroadcastManager.getInstance(this).sendBroadcast(pushNotifica
tion);

        NotificationUtils notificationUtils = new
NotificationUtils();
        notificationUtils.playNotificationSound();
    } else {

        Intent resultIntent = new Intent(getApplicationContext(),
MainActivity.class);
        resultIntent.putExtra("message", message);

        if (TextUtils.isEmpty(image)) {
            showNotificationMessage(getApplicationContext(),
```

```
title, message, timestamp, resultIntent);
        }
  }
```

## 3.2. Програмна реалізація бази даних

Оскільки розроблюваний додаток націлений на інтеграцію із веб-продуктами, то зрозуміло, що бізнес-правила, інструменти для роботи з БД та послідовність дій створення самої бази даних може різнитися в залежності від продукту з яким додаток буде інтегруватися. Фактично, ані вибір інших інструментів, ані СУБД та середовища її функціонування ніяк не вплине на роботу додатку, адже за зв'язок із БД на сервері відповідатимуть веб-служби, які можна буде переписати потрібним чином, при цьому не вносячи ніяких змін у самий Android-додаток.

Можливість використання Android-додатку передбачає, що усі потрібні для його роботи таблиці уже створені та наповнені даними, а REST API при потребі переписано та адаптовано у відповідності до веб-продукту та СУБД.

Єдиною вимогою при інтеграції додатку із веб-системою є використання рекомендованої структури бази даних, тобто забезпечення додатку усіма потрібними даними.

Для можливості надходження повідомлень від системи, потрібна реалізація спеціальних тригерів, які генеруватимуть потрібні повідомлення. Найкращим варіантом для підрахунку статичних даних теж є використання тригерів.

Для отримання та запису даних Android-додаток використовуватиме веб-служби REST API, які в свою чергу отримуватимуть дані з БД та записуватимуть дані в БД. Для роботи веб-служб із БД найкраще буде використовувати stored procedures.

Передбачаються наступні stored procedures для забезпечення роботи додатку:

1) «TemporaryUserPassword» – після авторизації користувачу надсилатимуеться тимчасовий пароль, ця процедура зберігатиме його в базу даних;
2) «GetTeacherSubjects» – отримує із БД персональний розклад для викладача;
3) «GetStudentSubjects» – отримує із БД персональний розклад для студента;
4) «GetStudents» – отримує із БД список студентів, для викладача, у яких він веде заняття;
5) «GetClassSubjects» – отримує таблицю зв’язків між класами та предметами;
6) «PushLesson» – записує в БД дані про проведене заняття;
7) «PushPresenceStatus» – записує в БД дані про відвідуваність занять студентами;
8) «GetStatForTecherAllSubjects» – отримує з БД статистичні дані про відвідуваність по всіх предметах викладача;
9) «GetStatForTecherSpecificSubjects» – отримує з БД статистичні дані про відвідуваність по певному предметі викладача;
10) «GetStatForTecherSpecificStudent» – отримує з БД статистичні дані про відвідуваність по всіх предметах викладача для певного студента;
11) «GetStatForStudentAllSubjects» – отримує з БД статистичні дані про відвідуваність по всіх предметах студента;

12) «GetStatForStudentSpecificSubjects» – отримує з БД статистичні дані про відвідуваність по певному предметі студента;

13) «GetMessages» – отримує з БД повідомлення для користувача;

14) «CreateMessage» – записує в БД створене повідомлення;

15) «ChangeMessage» – змінює в системі певні дані повідомлення.

Усі згадані stored procedures наведені в DDL коді Додатку Х.

Важливо щоб надсилання даних про відвідуваність проведеного викладачем заняття записувалися у базу даних у межах транзакції.

# РОЗДІЛ IV
# ТЕСТУВАННЯ ТА ДОСЛІДНА ЕКСПЛУАТАЦІЯ

## 4.1. Тестування

При розробці програмного продукту було проведено модульне та системне тестування. Було проведено функціональне тестування та тестування графічного інтерфейсу. Під час розробки системи та її тестування було налаштовано та наповнено тестову бузу даних.

Під час написання програмної реалізації більша частина функціоналу була покрита модульними тестами. Для модульного тестування було використано фреймворк JUnit. Загалом було написано 240 тестових випадків. Результати тестування відображено на рисунку 4.1.

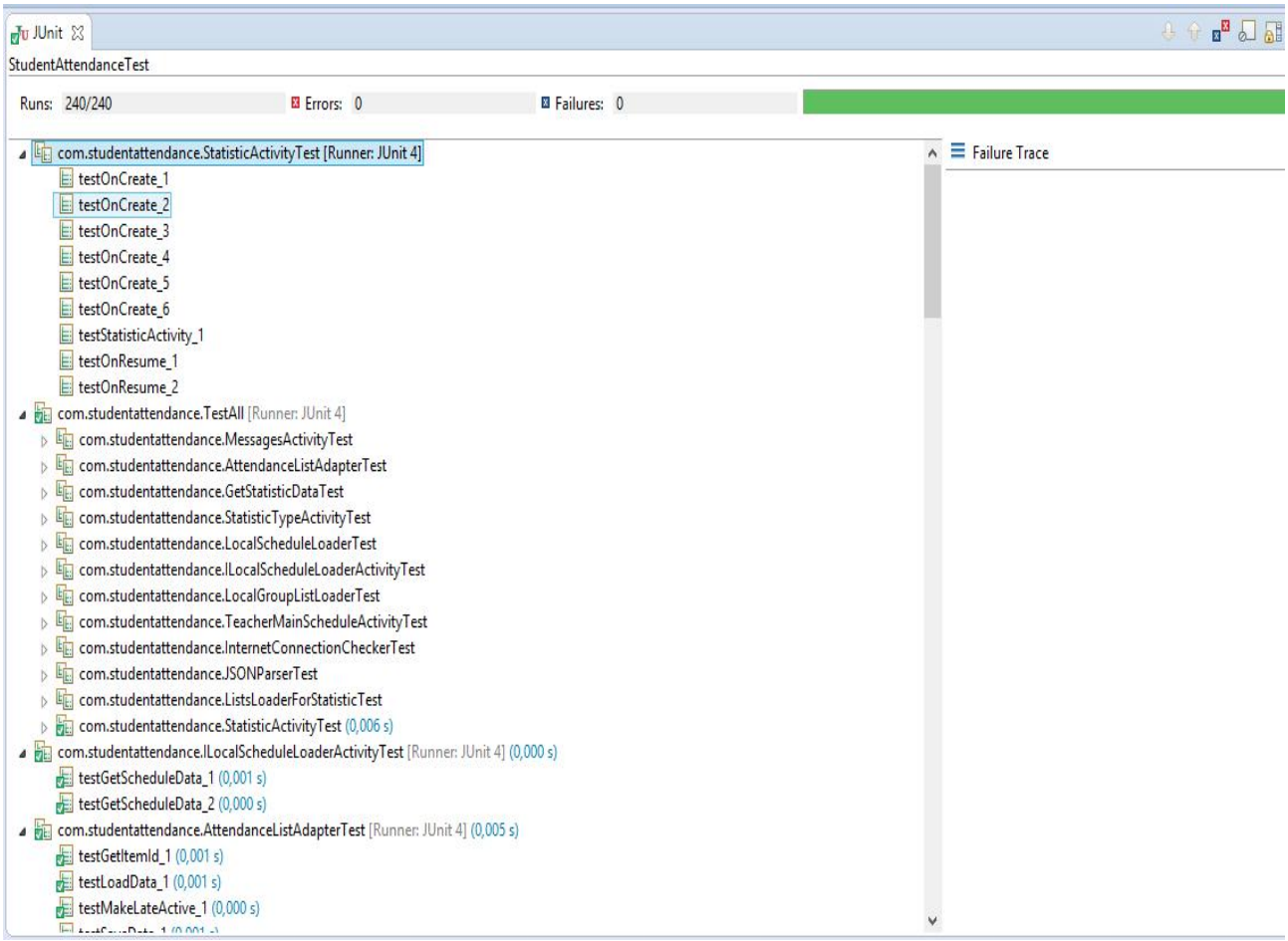


Рис. 4.1. Результат модульного тестування

Під час фази розробки тестів було спроектовано функціональні тестові випадки. Таблиця показує розподіл функціональних тестових випадків і наборів тестових даних для цих випадків за варіантами використання.

Таблиця 4.1

Розподіл функціональних тестових

| Варіанти використання | Тестові випадки | Тестові данні |
|---|---|---|
| Check logging | 2 | 2 |
| Check option “Exit” | 2 | 2 |
| Check option “TakeAttendance” | 2 | 2 |
| Check option “Send Message” | 3 | 3 |
| Check option “Get Statistic” | 2 | 2 |
| Check option “Send attendance data” | 2 | 2 |
| Загалом | 12 | 12 |

Тестування графічного інтерфейсу Android-додатку проводилось вручну на пристрої Huawei G700 із версією Android 4.2.1. Під час тестування було перевірено коректність розміщення усіх елементів інтерфейсу та їх відповідність заданим стилям й призначеному функціоналу.

## 4.2. Розгортання програмного продукту

Розроблений програмний продукт орієнтований на інтеграцію з функціонуючим веб-орієнтовним працюючим аналогом. Для коректної роботи клієнтської частини, серверна частина системи має бути повністю налаштована. Усі дані, які потрібні для роботи додатку мають бути присутніми у БД веб-орієнтованого продукту. Для роботи Android-додатку може бути використана фактично будь яка РСУБД та може бути використана основна БД або проміжна база даних синхронізована для імпорту й експорту з основною БД . Єдиною вимогою до структури БД є наявність усіх даних, які потрібні для роботи основного функціоналу Android-додатку. Основним завданням при налаштуванні серверної частини системи є забезпечення рекомендованої структури БД з усіма механізмами для обробки даних або адаптація REST API для БД з іншою структурою. Налаштування серверної частини є досить складним та потребує використання відповідних технічних спеціалістів.

Для функціонування клієнтської частини на мобільний пристрій потрібно завантажити та встановити Android apk файл надавши йому потрібні дозволи. Встановлення Android-додатку є стандартною процедурою для користувачів мобільних пристроїв із операційною системою Android та не потребує спеціальних знань чи вмінь.

Розроблений додаток буде повністю сумісний із усіма пристроями починаючи із SDK 16, що відповідає версії 4.1.2 операційної системи Android. На пристроях із більш ранім SDK додаток не працюватиме.

## 4.3. Інструкція користувача

Для авторизації у додатку користувачу потрібно буде ввести свій email та натиснути кнопку підтвердження. Після цього, система надішле йому на вказаний email тимчасовий пароль, який потрібно буде ввести у вікні, що відкриється. Повторна авторизація не буде потрібною поки користувач не змінить пристрій або не вийде із системи.

На рисунку 4.2 зображено вікно, яке дозволяє користувачу авторизацію в додатку.


Рис. 4.2. Вікно для авторизації в додатку

Після успішної авторизації, додаток сформує ваш персональний розклад, який буде відображатися в головному вікні програми. Для навігації по розкладу можна використовувати відповідні елементи меню, або свайп слайдер. Головне вікно та способи навігації у ньому зображено на рисунку 4.3.

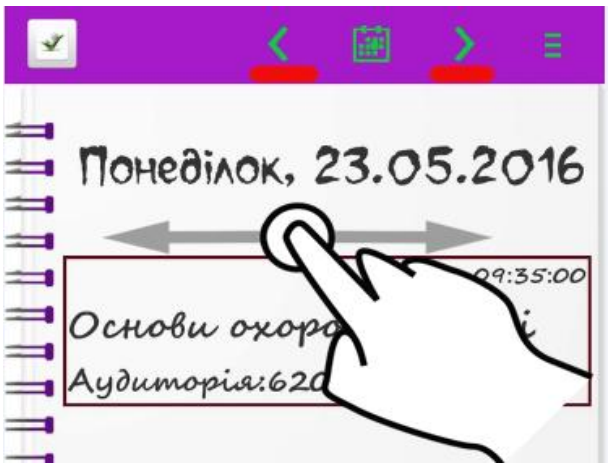


Рис. 4.3. Основне вікно додатку

Крім того користувач завжди має змогу повернутися до поточної дати вибравши елемент навігації «Календар». Якщо відкрито розклад для поточної дати, то натискання елементу навігації «Календар» відкриє календар який дозволить перейти до розкладу для вибраної дати, що відображено на рисунку 4.4.

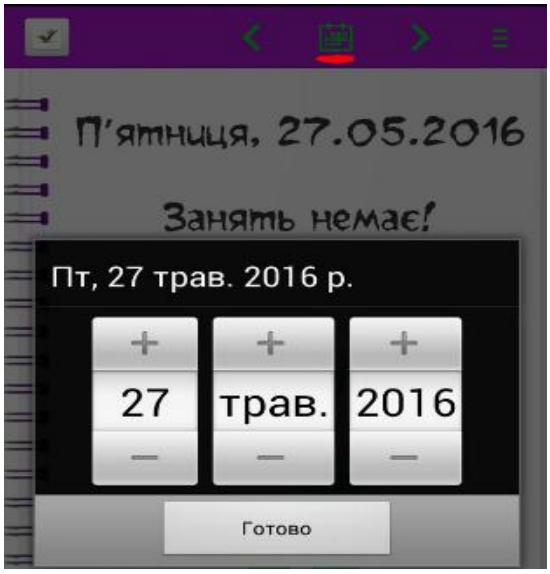


Рис. 4.4. Вікно «Календар»

Якщо користувач має права «Викладач», то він зможе відкрити вікно для заповнення даних про відвідуваність натиснувши на потрібний предмет у розкладі, що відображено на рисунку 4.5.

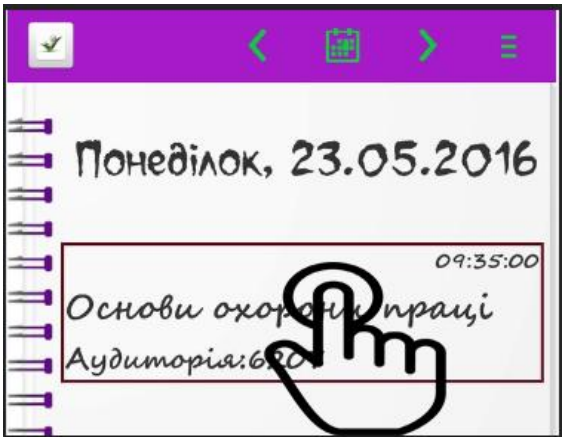


Рис. 4.5. Перехід до вікна для заповнення даних про відвідуваність

У вікні для заповнення даних про відвідуваність, яке зображено на рисунку 4.6, викладач зможе ввести дані як за допомогою елементів меню, так і за допомогою елементів керування у списку студентів.

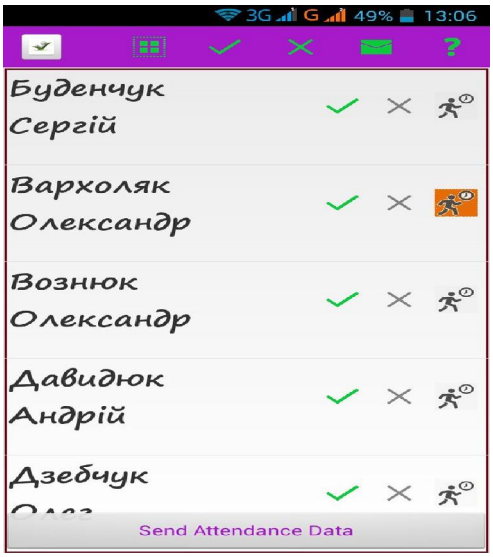


Рис. 4.6. Вікно для заповнення даних про відвідуваність

Вибравши зі списку студентів викладач зможе відправити їм повідомлення що відображено на рисунку 4.7

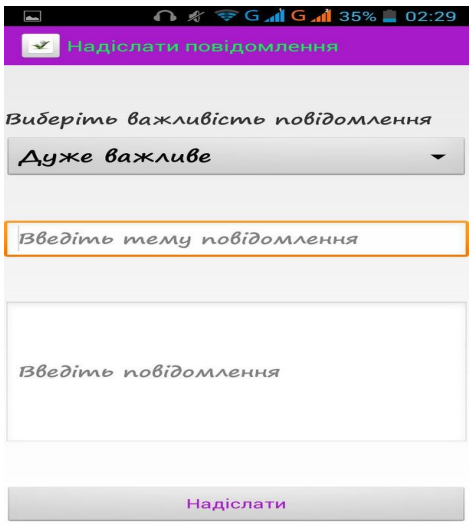


Рис. 4.7. Вікно для відправки повідомлення

В основному вікні знаходиться головне меню додатку, яке зображено на рисунку 4.8.

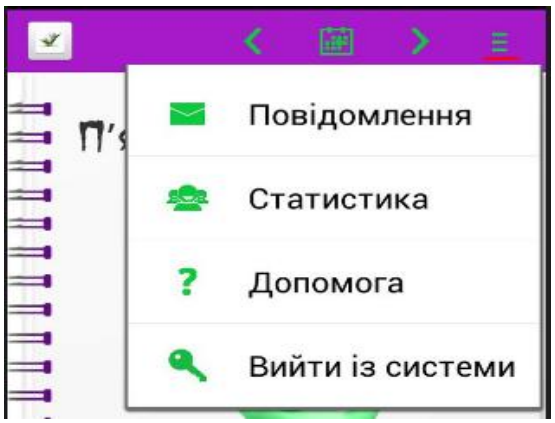


Рис. 4.8. Головне меню додатку

При виборі опції головного меню «Статистика» перед користувачем відкриється вікно, в якому він зможе вибрати необхідні фільтри, що відображено на рисунку 4.9.

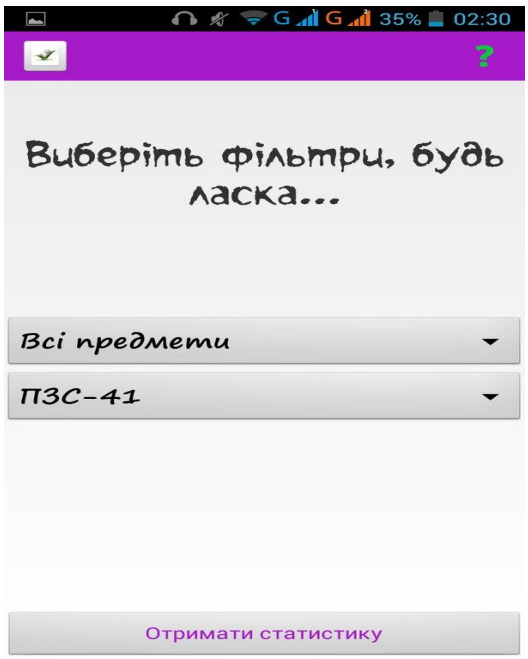


Рис. 4.9. Вікно для вибору фільтрів для статистики

Вибравши потрібні фільтри користувач зможе переглянути статистику натиснувши на відповідну кнопку. Вікно для перегляду статистики відображено на рисунку 4.10.

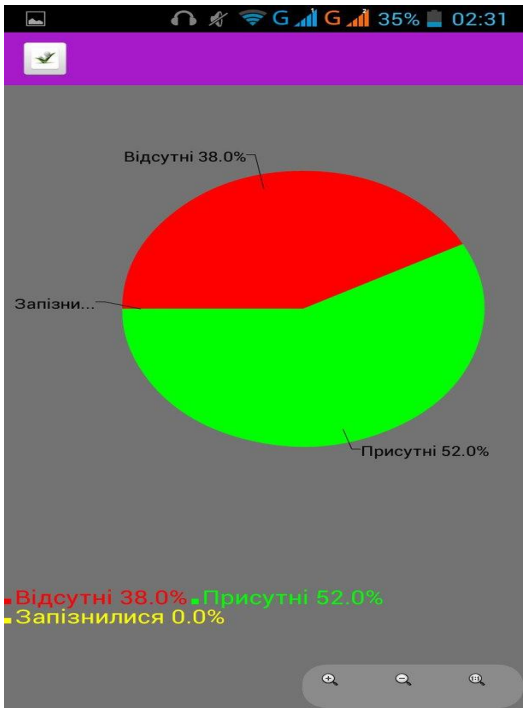

Рис. 4.10. Вікно із статистикою

При виборі опції головного меню «Повідомлення» перед користувачем відкриється вікно, в якому він зможе переглянути отримані повідомлення, що відображено на рисунку 4.11.


Рис. 4.11. Вікно із повідомленнями

Опція меню «Допомога» доступна із будь якого вікна програми. У вікні допомоги яке зображено на рисунку 4.12 відображено повну інструкцію користування додатком.

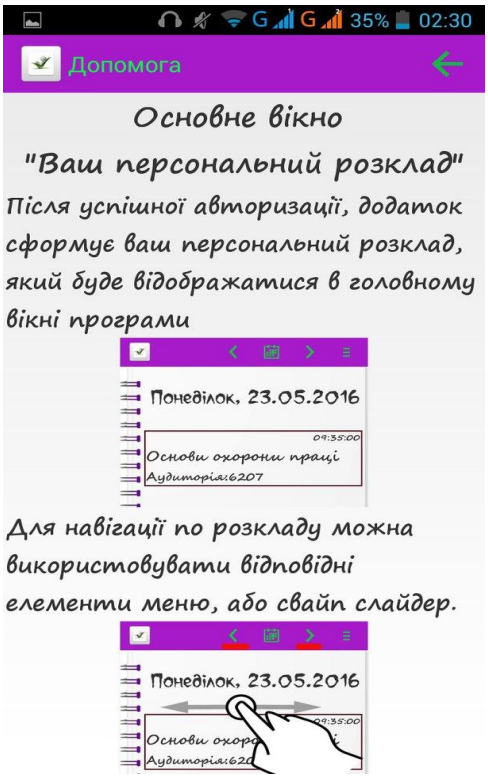


Рис. 4.12. Вікно «Допомога»

# ВИСНОВОК

Результатом написання дипломної роботи є готовий програмний продукт, який являє собою Android-додаток для обліку відвідування занять студентами. Однією із основних переваг розробленої програмної системи є можливість інтеграції із веб-орієнтованим аналогом. Така інтеграція дозволить вирішити одну із основних проблем навчальних закладів – проблему контролю відвідування студентами занять.

Розроблений програмний продукт майже повністю автоматизує процес заповнення та перегляду даних про відвідуваність. Також, розроблена програмна система надає засоби для інформування студентів про проблеми із відвідуванням занять та дозволяє студентам стежити за власною статистикою відвідування занять. Крім того, додаток гадає користувачу можливість перегляду свого персонального розкладу. За допомогою Android-додатку викладачі зможуть заповнювати журнал відвідування занять студентами безпосередньо на заняті та відправляти дані коли їм зручно. Відправлені дані одразу ж будуть оброблені на сервері та доступні для усіх учасників контролю процесу відвідування занять. Таким чином, виключаються зайві етапи збору даних про відвідуваність, що зменшує кількість помилок та виключає можливість недостовірності даних.

Важливою особливістю Android-додатку є те, що більша частин його функціоналу працює у режимі «Offline». Також, додаток надає викладачу можливість надсилання миттєвих повідомлень студентам, що дозволяє йому за потреби швидко передати студентам потрібну інформацію.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Subbu Allamaraju RESTful Web Services Cookbook / Subbu Allamaraju// O'Reilly Media / Yahoo Press. – 2010. – 316.
2. Martin Fowler UML Distilled: A Brief Guide to the Standard Object Modeling Language/ Martin Fowler // – Addison-Wesley Professional – 2003. – 208.
3. C.J. DateAn Introduction to Database / C.J. DateAn // – Pearson – 2003. – 1024.
4. Kathy Sierra Head First Java /Kathy Sierra// – O'Reilly Media – 2005. – 688.
1. Ryan Hodson Android Programming Succinctly / Ryan Hodson // – Syncfusion Inc.. – 2014. – 113.

ДОДАТОК А

Таблиця

Глосарій

| Термін | Опис терміну |
|---|---|
| Основні поняття та категорії предметної області та проекту | |
| Action Bar | Панель керування додатком, яка знаходиться в верхній частині екрану |
| Activity | Аналог вікна у ОС Android |
| Android | Найпопулярніша платформа та операційна система для мобільних пристроїв, яка створена на ядрі Linux й належить корпорації Google Inc. |
| APK | Інсталяційний файл програми для Android, який містить байт-код програми, ресурси, маніфест |
| AsyncTask | Клас, що забезпечує простий і зручний механізм для переміщення трудомістких операцій в фоновий потік. |
| Google Cloud Messaging (GCM) | Сервіс для надсилання миттєвих нотифікацій до клієнта-додатка Android |
| JSON | Це текстовий формат обміну даними між комп'ютерами. JSON базується на тексті, і може бути з легкістю прочитаним людиною. |
| Layout | Макет який визначає візуальну структуру користувацького інтерфейсу |
| REST | Підхід до архітектури мережевих протоколів, які забезпечують доступ до інформаційних ресурсів |
| RESTful | Такий, що відповідає архітектурі REST та підтримує її |
| Shared Preferences | Спеціальний файл для конкретного додатку який зберігається в захищеній внутрішній пам’яті та дозволяє зберігати дані у форматі |

| | |
|---|---|
| | key-value pair |

Продовження таблиці

| | |
|---|---|
| Stored procedure | Об'єкт бази даних , який представляє собою набір SQL інструкцій, який компілюється один раз і зберігається на сервері |
| Toast | Спливаюче повідомлення в ОС Android |
| Trigger | Це stored procedure особливого типу, яку користувач не викликає явно, а використання якої обумовлено настанням визначеної події (дії) у реляційній базі даних |
| Нотифікація | Тип повідомлення в Android-додатку, яке відображається в системному рядку стану |
| Транзакція | Група послідовних операцій з базою даних, яка є логічною одиницею роботи з даними. Транзакція може бути виконана успішно, або не виконана зовсім. |
| Користувачі системи | |
| Викладач | Користувач, який матиме повний доступ до функціоналу додатку та можливість внесення даних до головної БД |
| Студент | Користувач, який матиме обмежений доступ до функціоналу додатку лише в інформаційних цілях |
| Вхідні та вихідні документи | |
| Розклад занять | Документ, який є персональним графіком занять для викладача чи студента |
| Список із даними про відвідуваність | Документ, який формується викладачем при проведенні занять та містить дані про відвідуваність |
| Список із статистичними даними | Документ, який містить певні статистичні відомості |

| Список студентів | Документ із даними про студентів |
|---|---|

ДОДАТОК Б

```
CREATE TABLE Class
(
        ClassID          INTEGER NOT NULL,
        Name             VARCHAR(20) NULL,
        Course           INTEGER NULL,
        Faculty          VARCHAR(100) NULL
);
ALTER TABLE Class
ADD PRIMARY KEY (ClassID);
CREATE TABLE ClassSchedule
(
        ClassID          INTEGER NOT NULL,
        ScheduleID       INTEGER NOT NULL,
        TeacherID        INTEGER NOT NULL
);
ALTER TABLE ClassSchedule
ADD PRIMARY KEY (ClassID,ScheduleID,TeacherID);
CREATE TABLE Lesson
(
        LessonID         INTEGER NOT NULL,
        ScheduleID       INTEGER NOT NULL,
        TeacherID        INTEGER NOT NULL,
        Date&Time        DATE NULL
);
ALTER TABLE Lesson
ADD PRIMARY KEY (LessonID,ScheduleID,TeacherID);
CREATE TABLE Message
(
        MessageID        INTEGER NOT NULL,
        DateOfCreation   DATE NULL,
        Name             VARCHAR(50) NULL,
        Message          VARCHAR(100) NULL,
        Type             boolean NULL,
```

```
        UserID          INTEGER NOT NULL,
        TeacherID       INTEGER NOT NULL,
        StudentID       INTEGER NOT NULL,
        Status          VARCHAR(20) NULL,
        Prioryty        INTEGER NULL
    );
    ALTER TABLE Message
    ADD PRIMARY KEY (MessageID,UserID,TeacherID,StudentID);
    CREATE TABLE Schedule
    (
        ScheduleID      INTEGER NOT NULL,
        TeacherID       INTEGER NOT NULL,
        Name            VARCHAR(20) NULL,
        Type            VARCHAR(20) NULL,
        Classroom       VARCHAR(20) NULL,
        DayOfWeek       VARCHAR(20) NULL,
        Parity          VARCHAR(20) NULL,
        DateOfStart     DATE NULL,
        DateOfEnd       DATE NULL,
        Time            DATE NULL,
        Durability      DATE NULL
    );
    ALTER TABLE Schedule
ADD PRIMARY KEY (ScheduleID,TeacherID);
    CREATE TABLE Student
    (
        StudentID       INTEGER NOT NULL,
        ClassID         INTEGER NOT NULL,
        Name            VARCHAR(50) NULL,
        Surname         VARCHAR(50) NULL,
        Email           VARCHAR(100) NULL
    );
    ALTER TABLE Student
    ADD PRIMARY KEY (StudentID,ClassID);
    CREATE TABLE Teacher
```

```
(
        TeacherID          INTEGER NOT NULL,
        Name               VARCHAR(20) NULL,
        Surname            VARCHAR(20) NULL,
        Email              VARCHAR(20) NULL
);
ALTER TABLE Teacher
ADD PRIMARY KEY (TeacherID);
CREATE TABLE ThePresenceStatus
(
        ThePresenceStatusID  INTEGER NOT NULL,
        LessonID           INTEGER NOT NULL,
        ScheduleID          INTEGER NOT NULL,
        StudentID          INTEGER NOT NULL,
        ClassID            INTEGER NOT NULL,
        TeacherID          INTEGER NOT NULL,
        Datetime           DATE NULL,
        Status             VARCHAR(20) NULL,
        Additionl_Information VARCHAR(20) NULL,
        Late               INTEGER NULL,
        ResonOfChanges       VARCHAR(20) NULL
);
ALTER TABLE ThePresenceStatus
ADD                                    PRIMARY                                    KEY
(ThePresenceStatusID,LessonID,ScheduleID,StudentID,ClassID,TeacherID);
CREATE TABLE User
(
        UserID             INTEGER NOT NULL,
        Email              VARCHAR(100) NULL,
        TeacherID          INTEGER NOT NULL,
        StudentID          INTEGER NOT NULL,
        GCM_ID              VARCHAR(100) NULL,
        TemporaryPasword     VARCHAR(200) NULL
);
ALTER TABLE User
```

```
ADD PRIMARY KEY (UserID,TeacherID,StudentID);
ALTER TABLE ClassSchedule
ADD FOREIGN KEY Has (ClassID) REFERENCES Class (ClassID);
ALTER TABLE ClassSchedule
ADD FOREIGN KEY R_22 (ScheduleID, TeacherID) REFERENCES Schedule (ScheduleID, TeacherID);
ALTER TABLE Lesson
ADD FOREIGN KEY R_37 (ScheduleID, TeacherID) REFERENCES Schedule (ScheduleID, TeacherID);
ALTER TABLE Message
ADD FOREIGN KEY R_20 (UserID, TeacherID, StudentID) REFERENCES User (UserID, TeacherID, StudentID);
ALTER TABLE Schedule
ADD FOREIGN KEY R_50 (TeacherID) REFERENCES Teacher (TeacherID);
ALTER TABLE Student
ADD FOREIGN KEY R_35 (ClassID) REFERENCES Class (ClassID);
ALTER TABLE ThePresenceStatus
ADD FOREIGN KEY R_45 (LessonID, ScheduleID, TeacherID) REFERENCES Lesson (LessonID, ScheduleID, TeacherID);
ALTER TABLE ThePresenceStatus
ADD FOREIGN KEY R_46 (StudentID, ClassID) REFERENCES Student (StudentID, ClassID);
```

ДОДАТОК В

| № | Description | Steps to reproduce | Expected result | Actual result | Status | Environment |
|---|---|---|---|---|---|---|
| 1. | Check logging with registered email | 1.Enter registered email<br>2.Click “ok” | System creates temporary password and send this password on entered email. | System creates temporary password and send this password on entered email. | Passed | Android 4.2.1 |
| 2. | Check logging with unregistered email | 1.Enter unregistered email<br>2.Click “ok” | System shows toast “This email is not registered”. | System shows toast for unregistered email. | Passed | Android 4.2.1 |
| 3. | Check option “Exit” | 1. Open main menu<br>5.Click on option “Exit” | System deletes all user data. App is closed. | System deletes all user data. App is closed. | Passed | Android 4.2.1 |
| 4. | Check option “TakeAttendance” with teacher permissions | 1.Click on subject in schedule | Attendance window is opened | Attendance window is opened | Passed | Android 4.2.1 |
| 5. | Check option “TakeAttendance” with student permissions | 1.Click on subject in schedule | Nothing happened | Nothing happened | Passed | Android 4.2.1 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 6. | Check option “Send Message” with entered message | 1. Click on subject in schedule<br>2.Select some students<br>3. Click on option “Send Message”<br>4. Enter some message<br>5. Click “Send” | Message is sent. System shows toast “Message is sent successfully”. | Message is sent. System shows toast “Message is sent successfully”. | Passed | Android 4.2.1 |
| 7. | Check option “Send Message” with not entered message | 1. Click on subject in schedule<br>2.Select some students<br>3. Click on option “Send Message”<br>4. Click “Send” | System shows toast “Enter message please”. | System shows toast “Enter message please”. | Passed | Android 4.2.1 |
| 8. | Check option “Send Message” without Internet connection | 1. Click on subject in schedule<br>2.Select some students<br>3. Click on option “Send Message” | System shows toast “Internet connection is not available” | System shows toast “Internet connection is not available” | Passed | Android 4.2.1 |
| 9. | Check option “Get Statistic” | 1. Open main menu<br>2. Click on option “Statistic” | Statistic is showed | Statistic is showed | Passed | Android 4.2.1 |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | 3. Click on “Get Statistic” | | | | |
| 10. | Check option “Get Statistic” without Internet connection | 1. Open main menu<br>2. Click on option “Statistic” | System shows toast “Internet connection is not available” | System shows toast “Internet connection is not available” | Passed | Android 4.2.1 |
| 11. | Check option “Send attendance data” | 1. Click on subject in schedule<br>2. Click on “Send data” | Data is sent. System shows toast “Data is sent successfully”. | Data is sent. System shows toast “Data is sent successfully”. | Passed | Android 4.2.1 |
| 12. | Check option “Send attendance data” without Internet connection | 1. Click on subject in schedule<br>2. Click on “Send data” | System shows toast “Internet connection is not available, please try later” | System shows toast “Internet connection is not available, please try later” | Passed | Android 4.2.1 |

ДОДАТОК Г

AttendanceListAdapter.java

```
package com.studentattendance;

import java.util.ArrayList;
import java.util.HashMap;

import android.content.Context;
import android.graphics.Color;
import android.graphics.Typeface;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.Button;
import android.widget.ListAdapter;
import android.widget.TextView;

public class AttendanceListAdapter extends BaseAdapter implements ListAdapter {
  private ArrayList<HashMap<String, String>> list = new
ArrayList<HashMap<String, String>>();
  private Context context;

  public AttendanceListAdapter(ArrayList<HashMap<String, String>> list, Context
context) {
    this.list = list;
    this.context = context;
  }

  @Override
  public int getCount() {
    return list.size();
  }

  @Override
  public Object getItem(int pos) {
    return list.get(pos);
  }

  @Override
  public long getItemId(int pos) {
        return 0;
```

```
  }

  public void butonsNotActive(Button presentBtn, Button absentBtn, Button lateBtn)
{
    presentBtn.setBackgroundResource(R.drawable.ic_action_accept);
    absentBtn.setBackgroundResource(R.drawable.ic_action_cancel);
    lateBtn.setBackgroundResource(R.drawable.ic_late);
  }

  public void makePresentActive(Button presentBtn, Button absentBtn, Button
lateBtn, int pos) {
    butonsNotActive(presentBtn, absentBtn, lateBtn);
    presentBtn.setBackgroundResource(R.drawable.ic_action_accept_active);
    list.get(pos).put("late", "");
    list.get(pos).put("status", "p");
  }

  public void makeAbsentActive(Button presentBtn, Button absentBtn, Button
lateBtn, int pos) {
    butonsNotActive(presentBtn, absentBtn, lateBtn);
    absentBtn.setBackgroundResource(R.drawable.ic_action_cancel_active);
    list.get(pos).put("late", "");
    list.get(pos).put("status", "a");
  }

  public void makeLateActive(Button presentBtn, Button absentBtn, Button lateBtn,
int pos) {
    list.get(pos).put("status", "p");
    makePresentActive(presentBtn, absentBtn, lateBtn, pos);
    list.get(pos).put("late", "y");
    lateBtn.setBackgroundResource(R.drawable.ic_late_active);
  }

  public void loadData() {
  }

  public void saveData() {

  }

  @Override
  public View getView(final int position, View convertView, ViewGroup parent) {
    View view = convertView;
    if (view == null) {
      LayoutInflater inflater = (LayoutInflater)
```

```
context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        view = inflater.inflate(R.layout.attendance_list_item, null);
    }
    if (list.get(position).get("selected").equals("y"))
        view.setBackgroundColor(Color.parseColor("#20b12baf"));
    else
        view.setBackgroundColor(Color.TRANSPARENT);

    // Handle TextView and display string from your list

    Typeface myTypeface = Typeface.createFromAsset(context.getAssets(),
"fonts/segoe_print_bold.ttf");
    TextView stdName = (TextView) view.findViewById(R.id.studName);
    stdName.setTypeface(myTypeface);
    stdName.setText((String) ((list.get(position)).get("stdName")));
    // TextView className = (TextView) view.findViewById(R.id.className);
    // className.setTypeface(myTypeface);
    // className.setText((String) (list.get(position)).get("className"));

    // Handle buttons and add onClickListeners
    final Button presentBtn = (Button) view.findViewById(R.id.present_btn);
    final Button absentBtn = (Button) view.findViewById(R.id.absent_btn);
    final Button lateBtn = (Button) view.findViewById(R.id.late_btn);

    if (list.get(position).get("status").equals("p") &&
list.get(position).get("late").equals(""))
        makePresentActive(presentBtn, absentBtn, lateBtn, position);
    else if (list.get(position).get("status").equals("a"))
        makeAbsentActive(presentBtn, absentBtn, lateBtn, position);
    else if (!list.get(position).get("late").equals(""))
        makeLateActive(presentBtn, absentBtn, lateBtn, position);

    presentBtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            makePresentActive(presentBtn, absentBtn, lateBtn, position);
            notifyDataSetChanged();
        }
    });
    absentBtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            makeAbsentActive(presentBtn, absentBtn, lateBtn, position);
            notifyDataSetChanged();
        }
```

```
    });
    lateBtn.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        makeLateActive(presentBtn, absentBtn, lateBtn, position);
        notifyDataSetChanged();
      }
    });

    return view;
  }
}
```

AttendanceTaker.java

```
package com.studentattendance;

import android.app.ActionBar;
import android.app.Activity;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.SharedPreferences.Editor;
import android.graphics.Color;
import android.graphics.drawable.ColorDrawable;
import android.os.AsyncTask;
import android.os.Bundle;
import android.text.Html;
import android.util.Log;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.Window;
import android.widget.AdapterView;
import android.widget.Button;
import android.widget.ListView;
import android.widget.Toast;

import com.google.gson.Gson;
import com.google.gson.reflect.TypeToken;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;

import okhttp3.FormBody;
import okhttp3.RequestBody;

public class AttendanceTaker extends Activity {

   SharedPreferences sPref;
   private ProgressDialog pDialog;
   JSONObject jsonSchedule;
   Intent intent;
```

```
   String id, idOfSubject;
   ArrayList<HashMap<String, String>> listAttendance;
   AttendanceListAdapter atd;
   Boolean allSelected = false;
   ListView lView;
   Button Ok;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      getWindow().requestFeature(Window.FEATURE_ACTION_BAR);
      setContentView(R.layout.attendance_taker);
      ActionBar actionBar = getActionBar();
      actionBar.setBackgroundDrawable(new
ColorDrawable(Color.parseColor("#a71aca")));
      actionBar.setTitle(Html.fromHtml("<font color='#0bf748'></font>"));
      actionBar.show();
      sPref = this.getSharedPreferences("com.studentattendance",
Context.MODE_PRIVATE);
      intent = getIntent();
      id = intent.getStringExtra("ID");
      idOfSubject = id.substring(0, id.indexOf(' '));
      listAttendance = new ArrayList<HashMap<String, String>>();

      Ok = (Button) findViewById(R.id.ok);
      Ok.setTextColor(Color.parseColor("#a71aca"));
      // handle listview and assign adapter
      lView = (ListView) findViewById(R.id.attList);
      new LoadAttendanceList().execute();
      lView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
         @Override
         public void onItemClick(AdapterView<?> parent, View view, int position,
long id) {
            if (listAttendance.get(position).get("selected").equals("n"))
               listAttendance.get(position).put("selected", "y");
            else
               listAttendance.get(position).put("selected", "n");
            atd.notifyDataSetChanged();
            Log.e("Click", (String) listAttendance.get(position).get("status"));
            Log.e("Selected", (String)
listAttendance.get(position).get("selected"));
         }
      });

   }

   public void onOKStart(View v) {
      if (!listAttendance.isEmpty()) {
         String jsonAttendanceList = new Gson().toJson(listAttendance);
         sPref = this.getSharedPreferences("com.studentattendance",
Context.MODE_PRIVATE);
         Editor ed = sPref.edit();
         ed.putString(id, jsonAttendanceList);
         ed.commit();
         if (new InternetConnectionChecker(this).isOnline()) {
            new SendAttendance().execute();
         } else {
            Toast.makeText(this,
                  "Дані збережені, але не надіслані! Будь ласка повторіть
надсилання із підключеням до Інтернету!",
                  Toast.LENGTH_LONG).show();
         }
      } else
         Toast.makeText(this, "Щось погане трапилось :(",
```

```
Toast.LENGTH_LONG).show();
   }

   public void onBackPressed() {
      if (!listAttendance.isEmpty()) {
         String jsonAttendanceList = new Gson().toJson(listAttendance);

         sPref = this.getSharedPreferences("com.studentattendance",
Context.MODE_PRIVATE);
         Editor ed = sPref.edit();
         ed.putString(id, jsonAttendanceList);
         ed.commit();
         Toast.makeText(this, "Дані збережено, не забудьте їх надіслати!",
Toast.LENGTH_LONG).show();
      }
      finish();
   }

   @Override
   public boolean onCreateOptionsMenu(Menu menu) {
      MenuInflater inflater = getMenuInflater();
      inflater.inflate(R.menu.attendance_taker_actions, menu);
      return super.onCreateOptionsMenu(menu);
   }

   @Override
   public boolean onMenuItemSelected(int featureId, MenuItem item) {
      int itemId = item.getItemId();
      switch (itemId) {
      case R.id.selectAll:
         if (!allSelected) {
            for (int i = 0; i < listAttendance.size(); i++)
               listAttendance.get(i).put("selected", "y");
            allSelected = true;
         } else {
            for (int i = 0; i < listAttendance.size(); i++)
               listAttendance.get(i).put("selected", "n");
            allSelected = false;
         }
         atd.notifyDataSetChanged();
         break;
      case R.id.AllPresent:
         for (int i = 0; i < listAttendance.size(); i++)
            if (listAttendance.get(i).get("selected") == "y")
               listAttendance.get(i).put("status", "p");
         atd.notifyDataSetChanged();
         break;
      case R.id.AllAbsent:
         for (int i = 0; i < listAttendance.size(); i++)
            if (listAttendance.get(i).get("selected") == "y")
               listAttendance.get(i).put("status", "a");
         atd.notifyDataSetChanged();
         break;
      case R.id.action_help:
         Intent helpActivity = new Intent(getApplicationContext(),
HelpActivity.class);
         startActivity(helpActivity);
         break;
      case R.id.action_send_messages:
         Intent messageActivity = new Intent(getApplicationContext(),
SendMessageActivity.class);
         startActivity(messageActivity);
         break;
```

```
      }
      return true;
   }

   class LoadAttendanceList extends AsyncTask<String, String, String> {

      @Override
      protected void onPreExecute() {
         super.onPreExecute();
         pDialog = new ProgressDialog(AttendanceTaker.this);
         pDialog.setMessage("Завантаження даних, зачекайте будь ласка...");
         pDialog.setIndeterminate(false);
         pDialog.setCancelable(false);
         pDialog.show();
      }

      protected String doInBackground(String... args) {
         if (sPref.contains(id)) {
            Log.e("Er:", "ISA");
            Gson gson = new Gson();
            listAttendance = gson.fromJson(sPref.getString(id.toString(), ""),
                  new TypeToken<ArrayList<HashMap<String, String>>>() {
                  }.getType());
         } else {
            Gson gson = new Gson();
            ArrayList<HashMap<String, String>> classSubjects = new
ArrayList<HashMap<String, String>>();
            JSONArray studentsList = new JSONArray();
            String classID = null;
            classSubjects = gson.fromJson(sPref.getString("ClassSubjects", ""),
                  new TypeToken<ArrayList<HashMap<String, String>>>() {
                  }.getType());
            try {
               studentsList = new JSONArray(sPref.getString("StudentList", ""));

               for (int i = 0; i < classSubjects.size(); i++) {
                  Log.e("QQQQ:",
classSubjects.get(i).get("SubjectID").toString());
                  Log.e("QQQQ:", idOfSubject);
                  if
(idOfSubject.equals(classSubjects.get(i).get("SubjectID").toString())) {
                     Log.e("Er:", "ISsss");
                     classID = classSubjects.get(i).get("ClassID").toString();
                  }
               }
               if (classID != null)
                  for (int i = 0; i < studentsList.length(); i++) {
                     Log.e("Er:", "ISsss");
                     JSONObject classes = studentsList.getJSONObject(i);
                     JSONArray students = classes.getJSONArray(classID);
                     for (int j = 0; j < students.length(); j++) {
                        JSONObject subject = students.getJSONObject(j);

                        String id = subject.getString("ID");
                        String name = subject.getString("Surname").concat("\n")
                              .concat(subject.getString("Name"));
                        String email = subject.getString("Email");

                        HashMap<String, String> map = new HashMap<String,
String>();

                        map.put("ID", id);
                        map.put("classID", classID);
                        map.put("stdName", name);
```

```
                        map.put("email", email);
                        map.put("status", "p");
                        map.put("selected", "n");
                        map.put("late", "");

                        listAttendance.add(map);
                     }
                  }
               } catch (JSONException e) {
                  Log.e("Er:", Log.getStackTraceString(e));
               }
            }

            return null;
         }

         protected void onPostExecute(String file_url) {

            pDialog.dismiss();
            if (!listAttendance.isEmpty()) {
               atd = new AttendanceListAdapter(listAttendance,
AttendanceTaker.this);
               lView.setAdapter(atd);
            }

         }
      }

      class SendAttendance extends AsyncTask<String, String, String> {

         @Override
         protected void onPreExecute() {
            super.onPreExecute();
            pDialog = new ProgressDialog(AttendanceTaker.this);
            pDialog.setMessage("Завантаження даних, зачекайте будь ласка...");
            pDialog.setIndeterminate(false);
            pDialog.setCancelable(false);
            pDialog.show();
         }

         protected String doInBackground(String... args) {
            String attendanceData = sPref.getString(id.toString(),
"").replace("\n", " ");
            Log.e("AttData:", attendanceData);
            String[] lessonData = id.split(" ");
            JSONParser jsonParser = new JSONParser();
            String url_checkIfExist =
"http://192.168.0.103/StudentAttandanceWSTest/pushAttendance.php";
            RequestBody formBody = new FormBody.Builder().add("SubjectID",
lessonData[0])
                  .add("DateAndTime", lessonData[2]).add("AttendanceData",
attendanceData).build();
            JSONObject json = null;
            try {
               json = jsonParser.postRequest(url_checkIfExist, formBody);
            } catch (IOException e) {
               Log.e("Er:", Log.getStackTraceString(e));
            }
            try {
               int success = json.getInt("success");
               if (success == 1) {
                  // Toast.makeText(AttendanceTaker.this, "Data is send
                  // successfuly.:)", Toast.LENGTH_SHORT).show();
                  Log.e("J:", json.getString("message"));
```

```
            } else {
               Toast.makeText(AttendanceTaker.this, "Проблема із підключенням до
сервера.:( Спробуйте пізніше будь ласка!",
                     Toast.LENGTH_SHORT).show();
               Log.e("J:", json.getString("message"));
            }
         } catch (JSONException e) {
            Log.e("Er:", Log.getStackTraceString(e));
         }
         return null;
      }

      protected void onPostExecute(String file_url) {
         pDialog.dismiss();
      }

   }
}
```

GetStatisticData.java

```
package com.studentattendance;

import java.io.IOException;

import org.json.JSONException;
import org.json.JSONObject;

import android.app.ProgressDialog;
import android.content.Context;
import android.os.AsyncTask;
import android.util.Log;
import okhttp3.FormBody;
import okhttp3.RequestBody;

public class GetStatisticData extends AsyncTask<String, String, String> {

  private StatisticTypeActivity activity;
  private Context context;
  private ProgressDialog pDialog;
  private String subjectsAndGroups, result;

  public GetStatisticData(Context c, StatisticTypeActivity a, String
subjectsAndGroups) {
    context = c;
    activity = a;
    this.subjectsAndGroups = subjectsAndGroups;
    context.getSharedPreferences("com.studentattendance",
Context.MODE_PRIVATE);
  }
```

```
    @Override
    protected void onPreExecute() {
        super.onPreExecute();
        pDialog = new ProgressDialog(context);
        pDialog.setMessage("Завантаження даних, зачекайте будь ласка...");
        pDialog.setIndeterminate(false);
        pDialog.setCancelable(false);
        pDialog.show();
    }

    protected String doInBackground(String... args) {
        JSONParser jsonParser = new JSONParser();
        String url_checkIfExist =
"http://192.168.0.103/StudentAttandanceWSTest/getStatisticForTeacher.php";
        RequestBody formBody = new
FormBody.Builder().add("SubjectsAndGroupsList", subjectsAndGroups).build();
        JSONObject json = null;
        try {
            json = jsonParser.postRequest(url_checkIfExist, formBody);
        } catch (IOException e) {
            Log.e("Er:", Log.getStackTraceString(e));
        }
        try {
            int success = json.getInt("success");
            if (success == 1) {
                result = json.getString("stat");
                Log.e("J:", json.getString("message"));
            } else {
                Log.e("J:", json.getString("message"));
            }
        } catch (JSONException e) {
            Log.e("Er:", Log.getStackTraceString(e));
        }
        return null;
    }

    protected void onPostExecute(String file_url) {
        activity.CallActivityForSelectedFilters(result);
        pDialog.dismiss();
    }

}
```

HelpActivity.java

```
package com.studentattendance;
```

```
import android.app.ActionBar;
import android.app.Activity;
import android.graphics.Color;
import android.graphics.Typeface;
import android.graphics.drawable.ColorDrawable;
import android.os.Bundle;
import android.text.Html;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.Window;
import android.widget.TextView;

public class HelpActivity extends Activity {

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    getWindow().requestFeature(Window.FEATURE_ACTION_BAR);
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_help);
    ActionBar actionBar = getActionBar();
    actionBar.setBackgroundDrawable(new
ColorDrawable(Color.parseColor("#a71aca")));
    actionBar.setTitle(Html.fromHtml("<font
color='#0bf748'>Допомога</font>"));
    actionBar.show();
    Typeface tp = Typeface.createFromAsset(getAssets(),
"fonts/segoe_print_bold.ttf");
    TextView mainWindowTV=(TextView) findViewById(R.id.mainWindowTV);
    mainWindowTV.setTypeface(tp);
    TextView mainWindowText1=(TextView)
findViewById(R.id.mainWindowText1);
    mainWindowText1.setTypeface(tp);
    TextView mainWindowText2=(TextView)
findViewById(R.id.mainWindowText2);
    mainWindowText2.setTypeface(tp);
  }

  @Override
  public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.help_activity_actions, menu);
    return super.onCreateOptionsMenu(menu);
  }
```

```
  @Override
  public boolean onMenuItemSelected(int featureId, MenuItem item) {
    int itemId = item.getItemId();
    switch (itemId) {
    case R.id.action_back:
      finish();
      break;
    }
    return true;
  }
}
```

IlocalScheduleLoaderActivity.java

```
package com.studentattendance;

import java.util.ArrayList;
import java.util.HashMap;

import android.app.Activity;

public class ILocalScheduleLoaderActivity extends Activity {
  ArrayList<HashMap<String, String>> scheduleList;

  void getScheduleData(ArrayList<HashMap<String, String>> scheduleList) {
    if (!scheduleList.isEmpty()) {
      this.scheduleList = scheduleList;
    }
  }
}
```

InternetConnectionChecker.java

```
package com.studentattendance;

import android.content.Context;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;

public class InternetConnectionChecker {
  Context _context;

  public InternetConnectionChecker(Context context) {
    _context = context;
  }

  public boolean isOnline() {
    ConnectivityManager cm = (ConnectivityManager)
```

```
_context.getSystemService(Context.CONNECTIVITY_SERVICE);
      NetworkInfo netInfo = cm.getActiveNetworkInfo();
      return netInfo != null && netInfo.isConnectedOrConnecting();
    }
}
```

JSONParser.java

```
package com.studentattendance;

import java.io.IOException;
import java.io.InputStream;
import java.io.UnsupportedEncodingException;

import org.json.JSONException;
import org.json.JSONObject;

import android.util.Log;
import okhttp3.MediaType;
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.RequestBody;
import okhttp3.Response;

public class JSONParser {
  public static final MediaType JSON = MediaType.parse("application/json;
charset=utf-8");

  OkHttpClient client = new OkHttpClient();
  static InputStream is = null;
  static JSONObject jObj = null;
  static String jsonAns = null;

  JSONObject postRequest(String url, RequestBody json) throws IOException {
    try {

      Request request = new Request.Builder().url(url).post(json).build();
      Response response = client.newCall(request).execute();
      jsonAns = response.body().string();
    } catch (UnsupportedEncodingException e) {
      Log.e("Er:", Log.getStackTraceString(e));
    } catch (IOException e) {
      Log.e("Er:", Log.getStackTraceString(e));
    } catch (Exception e) {
      Log.e("Er:", Log.getStackTraceString(e));
    }
```

```
    Log.e("var-", jsonAns);
    // try parse the string to a JSON object
    try {
      jObj = new JSONObject(jsonAns);
    } catch (JSONException e) {
      Log.e("JSON Parser", "Error parsing data " + e.toString());
    }
    // return JSON String
    return jObj;
  }
}
```

LocalGroupListLoader.java

```
package com.studentattendance;

import java.util.ArrayList;
import java.util.HashMap;

import com.google.gson.Gson;
import com.google.gson.reflect.TypeToken;

import android.app.ProgressDialog;
import android.content.Context;
import android.content.SharedPreferences;
import android.os.AsyncTask;

public class LocalGroupListLoader extends AsyncTask<String, String, String> {
   private StatisticTypeActivity activity;
   private Context context;
   private ProgressDialog pDialog;
   private SharedPreferences sPref;
   private ArrayList<HashMap<String, String>> groupeList = new
ArrayList<HashMap<String, String>>();
   private String subjectId;
   String IdOrNames;

   public LocalGroupListLoader(Context c, StatisticTypeActivity a, String
subjectId, String IdOrNames) {
      context = c;
      activity = a;
      this.subjectId = subjectId;
      this.IdOrNames = IdOrNames;
      sPref = context.getSharedPreferences("com.studentattendance",
Context.MODE_PRIVATE);
   }

   @Override
   protected void onPreExecute() {
      super.onPreExecute();
      pDialog = new ProgressDialog(context);
      pDialog.setMessage("Завантаження даних, зачекайте будь ласка...");
      pDialog.setIndeterminate(false);
      pDialog.setCancelable(false);
      pDialog.show();
   }

   protected String doInBackground(String... args) {
      Gson gson = new Gson();
      ArrayList<HashMap<String, String>> classSubjects = new
```

```
ArrayList<HashMap<String, String>>();
      classSubjects = gson.fromJson(sPref.getString("ClassSubjects", ""),
            new TypeToken<ArrayList<HashMap<String, String>>>() {
            }.getType());
      if (subjectId.equals("All_Subjects")) {
         for (int i = 0; i < classSubjects.size(); i++) {
            HashMap<String, String> hm = new HashMap<String, String>();
            hm.put("Name", classSubjects.get(i).get("Name").toString());
            hm.put("ClassID", classSubjects.get(i).get("ClassID").toString());
            hm.put("SubjectID",
classSubjects.get(i).get("SubjectID").toString());
            groupeList.add(hm);
         }
      } else {
         for (int i = 0; i < classSubjects.size(); i++) {
            if
(subjectId.equals(classSubjects.get(i).get("SubjectID").toString())) {
               HashMap<String, String> hm = new HashMap<String, String>();
               hm.put("Name", classSubjects.get(i).get("Name").toString());
               hm.put("ClassID",
classSubjects.get(i).get("ClassID").toString());
               hm.put("SubjectID",
classSubjects.get(i).get("SubjectID").toString());
               groupeList.add(hm);
            }
         }
      }

      return null;
   }

   protected void onPostExecute(String file_url) {
      pDialog.dismiss();
      if (!(IdOrNames.equals("Names"))) {
         activity.setGroupeSpinnerData(groupeList);
      }
   }

}
```

MainActivity.java

```
package com.studentattendance;

import android.app.Activity;
import android.app.ProgressDialog;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.SharedPreferences;
import android.content.SharedPreferences.Editor;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.Handler;
import android.support.v4.content.LocalBroadcastManager;
import android.util.Log;
import android.view.Gravity;
import android.widget.Toast;

import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.GoogleApiAvailability;

import org.json.JSONException;
import org.json.JSONObject;
```

```
import java.io.IOException;

import okhttp3.FormBody;
import okhttp3.RequestBody;

public class MainActivity extends Activity {
   SharedPreferences sPref;
   Toast toast;
   private ProgressDialog pDialog;
   JSONParser jsonParser = new JSONParser();
   private static final String TAG_SUCCESS = "success";
   private static String url_GetSchedule =
"http://192.168.0.103/StudentAttandanceWSTest/getSchedule.php";
   private static String url_GetStudents =
"http://192.168.0.103/StudentAttandanceWSTest/getStudentList.php";
   private static String url_GetClassSubjects =
"http://192.168.0.103/StudentAttandanceWSTest/getClass_subjects.php";
   String UserID;
   String UserType;
   Handler handler = new Handler();
   private String TAG = MainActivity.class.getSimpleName();
   private static final int PLAY_SERVICES_RESOLUTION_REQUEST = 9000;
   private BroadcastReceiver mRegistrationBroadcastReceiver;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      sPref = this.getSharedPreferences("com.studentattendance",
Context.MODE_PRIVATE);
      final Intent myIntent = new Intent(this, LoginActivity.class);
      if (!sPref.contains("UserID"))
         handler.postDelayed(new Runnable() {
            public void run() {
               startActivity(myIntent);
               finish();
            }
         }, 2000);
      else {
         if (!sPref.contains("Schedule"))
            new GetDataForSchedule().execute();
         activitySelector();
      }
      mRegistrationBroadcastReceiver = new BroadcastReceiver() {
         @Override
         public void onReceive(Context context, Intent intent) {

            // checking for type intent filter
            if (intent.getAction().equals(Config.REGISTRATION_COMPLETE)) {
               // gcm successfully registered
               // now subscribe to `global` topic to receive app wide
notifications
               String token = intent.getStringExtra("token");

               Toast.makeText(getApplicationContext(), "GCM registration token:
" + token, Toast.LENGTH_LONG).show();

            } else if (intent.getAction().equals(Config.SENT_TOKEN_TO_SERVER)) {
               // gcm registration id is stored in our server's MySQL

               Toast.makeText(getApplicationContext(), "GCM registration token
is stored in server!", Toast.LENGTH_LONG).show();
```

```
            } else if (intent.getAction().equals(Config.PUSH_NOTIFICATION)) {
               // new push notification is received

               Toast.makeText(getApplicationContext(), "Push notification is
received!", Toast.LENGTH_LONG).show();
            }
         }
      };

      if (checkPlayServices()) {
         registerGCM();
      }
   }

   public void onBackPressed() {
      if (toast != null)
         toast.cancel();
      System.exit(0);
   }

   void activitySelector() {
      if ((sPref.getString("UserType", "")).equals("t")) {
         final Intent myIntent = new Intent(this,
TeacherMainScheduleActivity.class);
         handler.postDelayed(new Runnable() {
            public void run() {
               startActivity(myIntent);
               finish();
            }
         }, 1000);
      } else if ((sPref.getString("UserType", "")).equals("s")) {
         final Intent myIntent = new Intent(this,
StudentMainScheduleActivity.class);
         handler.postDelayed(new Runnable() {
            public void run() {
               startActivity(myIntent);
               finish();
            }
         }, 1000);
      }

   }

   class GetDataForSchedule extends AsyncTask<String, String, String> {

      @Override
      protected void onPreExecute() {
         super.onPreExecute();
         pDialog = new ProgressDialog(MainActivity.this);
         pDialog.setMessage("Loading data...");
         pDialog.setIndeterminate(false);
         pDialog.setCancelable(true);
         pDialog.show();
      }

      protected String doInBackground(String... args) {
         UserID = (sPref.getString("UserID", ""));
         UserType = (sPref.getString("UserType", ""));
         RequestBody formBodyToGetScedule = new FormBody.Builder().add("ID",
UserID).add("UserType", UserType)
               .build();
         JSONObject scheduleJson = null;
         try {
            scheduleJson = jsonParser.postRequest(url_GetSchedule,
```

```
formBodyToGetScedule);
         } catch (IOException e) {
            Log.e("Er:", Log.getStackTraceString(e));
         }
         try {
            int success = scheduleJson.getInt(TAG_SUCCESS);
            if (success == 1) {
               final String Schedule = scheduleJson.getString("Schedule");
               sPref =
MainActivity.this.getSharedPreferences("com.studentattendance",
Context.MODE_PRIVATE);
               Editor ed = sPref.edit();
               ed.putString("Schedule", Schedule);
               ed.commit();
            } else {
               final String message = scheduleJson.getString("message");
               MainActivity.this.runOnUiThread(new Runnable() {
                  public void run() {
                     toast = Toast.makeText(MainActivity.this, message,
Toast.LENGTH_LONG);
                     toast.setGravity(Gravity.CENTER, 0, 0);
                     toast.show();
                  }
               });

               Log.e("J:", scheduleJson.getString("message"));
            }
         } catch (JSONException e) {
            Log.e("Er:", Log.getStackTraceString(e));
         }
         if (UserType.equals("t")) {
            RequestBody formBodyToGetStudentList = new
FormBody.Builder().add("ID", UserID).build();
            JSONObject jsonStudentList = null;
            JSONObject class_subjects = null;
            try {
               jsonStudentList = jsonParser.postRequest(url_GetStudents,
formBodyToGetStudentList);
               class_subjects = jsonParser.postRequest(url_GetClassSubjects,
formBodyToGetStudentList);
            } catch (IOException e) {
               Log.e("Er:", Log.getStackTraceString(e));
            }
            try {
               if (jsonStudentList.getInt(TAG_SUCCESS) == 1) {
                  final String studentList =
jsonStudentList.getString("Classes");
                  sPref =
MainActivity.this.getSharedPreferences("com.studentattendance",
Context.MODE_PRIVATE);
                  Editor ed = sPref.edit();
                  ed.putString("StudentList", studentList);
                  ed.commit();
               } else {
                  final String message = jsonStudentList.getString("message");
                  MainActivity.this.runOnUiThread(new Runnable() {
                     public void run() {
                        toast = Toast.makeText(MainActivity.this, message,
Toast.LENGTH_LONG);
                        toast.setGravity(Gravity.CENTER, 0, 0);
                        toast.show();
                     }
                  });
```

```
                Log.e("J:", jsonStudentList.getString("message"));
            }
            if (class_subjects.getInt(TAG_SUCCESS) == 1) {
                final String classSubjects =
class_subjects.getString("ClassSubjects");
                sPref =
MainActivity.this.getSharedPreferences("com.studentattendance",
Context.MODE_PRIVATE);
                Editor ed = sPref.edit();
                ed.putString("ClassSubjects", classSubjects);
                ed.commit();
            } else {
                final String message = class_subjects.getString("message");
                MainActivity.this.runOnUiThread(new Runnable() {
                    public void run() {
                        toast = Toast.makeText(MainActivity.this, message,
Toast.LENGTH_LONG);
                        toast.setGravity(Gravity.CENTER, 0, 0);
                        toast.show();
                    }
                });

                Log.e("J:", jsonStudentList.getString("message"));
            }
         } catch (JSONException e) {
            Log.e("Error!:", Log.getStackTraceString(e));
         }

      }

      return null;
   }

   protected void onPostExecute(String file_url) {
      pDialog.dismiss();
   }

}

// starting the service to register with GCM
private void registerGCM() {
   Intent intent = new Intent(this, GcmIntentService.class);
   intent.putExtra("key", "register");
   startService(intent);
}

private boolean checkPlayServices() {
   GoogleApiAvailability apiAvailability =
GoogleApiAvailability.getInstance();
   int resultCode = apiAvailability.isGooglePlayServicesAvailable(this);
   if (resultCode != ConnectionResult.SUCCESS) {
      if (apiAvailability.isUserResolvableError(resultCode)) {
         apiAvailability.getErrorDialog(this, resultCode,
PLAY_SERVICES_RESOLUTION_REQUEST)
               .show();
      } else {
         Log.i(TAG, "This device is not supported. Google Play Services not
installed!");
         Toast.makeText(getApplicationContext(), "This device is not
supported. Google Play Services not installed!", Toast.LENGTH_LONG).show();
         finish();
      }
      return false;
   }
```

```
        return true;
    }

    @Override
    protected void onResume() {
        super.onResume();

        // register GCM registration complete receiver

LocalBroadcastManager.getInstance(this).registerReceiver(mRegistrationBroadcastR
eceiver,
                new IntentFilter(Config.REGISTRATION_COMPLETE));

        // register new push message receiver
        // by doing this, the activity will be notified each time a new message
arrives

LocalBroadcastManager.getInstance(this).registerReceiver(mRegistrationBroadcastR
eceiver,
                new IntentFilter(Config.PUSH_NOTIFICATION));
    }

    @Override
    protected void onPause() {

LocalBroadcastManager.getInstance(this).unregisterReceiver(mRegistrationBroadcas
tReceiver);
        super.onPause();
    }
}
```

MessagesListAdapter.java

```
package com.studentattendance;

import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.HashMap;

import android.content.Context;
import android.graphics.Typeface;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.ImageView;
import android.widget.ListAdapter;
import android.widget.TextView;

public class MessagesListAdapter extends BaseAdapter implements ListAdapter {
    private ArrayList<HashMap<String, String>> list = new
ArrayList<HashMap<String, String>>();
    private Context context;

    public MessagesListAdapter(ArrayList<HashMap<String, String>> list, Context
context) {
        this.list = list;
        this.context = context;
    }

    @Override
    public int getCount() {
        return list.size();
    }
```

```
   @Override
   public Object getItem(int pos) {
      return list.get(pos);
   }

   @Override
   public long getItemId(int pos) {
      // return list.get(pos).getId();
      return 0;
      // just return 0 if your list items do not have an Id variable.
   }

   class HashMapComparator implements Comparator<HashMap<String, String>> {
      private final String key;

      public HashMapComparator(String key) {
         this.key = key;
      }

      public int compare(HashMap<String, String> first, HashMap<String, String>
second) {
         // TODO: Null checking, both for maps and values
         return second.get(key).compareTo(first.get(key));
      }

   }

   public void loadData() {
   }

   public void saveData() {
   }

   @Override
   public View getView(final int position, View convertView, ViewGroup parent) {
      View view = convertView;
      if (view == null) {
         LayoutInflater inflater = (LayoutInflater)
context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
         view = inflater.inflate(R.layout.messages_list_item, null);
      }

      Collections.sort(list, new HashMapComparator("priority"));
      Typeface mssgNameFont = Typeface.createFromAsset(context.getAssets(),
"fonts/comic_sans_ms.ttf");
      TextView mssgName = (TextView) view.findViewById(R.id.messageSubject);
      mssgName.setTypeface(mssgNameFont);
      mssgName.setText(list.get(position).get("messageSubject"));
      ImageView mssgImg = (ImageView) view.findViewById(R.id.message_img);
      TextView mssgBody = (TextView) view.findViewById(R.id.messageBody);
      mssgBody.setText(list.get(position).get("messageBody").length() > 20
            ? list.get(position).get("messageBody").substring(0,
20).concat("...")
            : list.get(position).get("messageBody"));
      TextView dateAndTime = (TextView) view.findViewById(R.id.dateAndTime);

dateAndTime.setText(list.get(position).get("date").concat("\n").concat(list.get(
position).get("time")));
      ImageView priorityImg = (ImageView) view.findViewById(R.id.priority_img);

      if (list.get(position).get("status") == "new")
         mssgImg.setImageResource(R.drawable.ic_action_email);
      else
```

```
            mssgImg.setImageResource(R.drawable.ic_action_read);

        if (list.get(position).get("priority") == "2")
            priorityImg.setImageResource(R.drawable.ic_action_important);
        else if (list.get(position).get("priority") == "1")
            priorityImg.setImageResource(R.drawable.ic_action_half_important);
        else
            priorityImg.setImageResource(R.drawable.ic_action_not_important);

        return view;
    }
}
```

MyGcmPushReceiver.java

```
package com.studentattendance;

import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.support.v4.content.LocalBroadcastManager;
import android.text.TextUtils;
import android.util.Log;

import com.google.android.gms.gcm.GcmListenerService;

/**
 * Created by underhand on 25.05.2016.
 */

public class MyGcmPushReceiver extends GcmListenerService {

    private static final String TAG = MyGcmPushReceiver.class.getSimpleName();

    private NotificationUtils notificationUtils;

    /**
     * Called when message is received.
     *
     * @param from   SenderID of the sender.
     * @param bundle Data bundle containing message data as key/value pairs.
     *               For Set of keys use data.keySet().
     */

        @Override
        public void onMessageReceived(String from, Bundle bundle) {
            String title = bundle.getString("title");
            String message = bundle.getString("message");
            String image = bundle.getString("image");
            String timestamp = bundle.getString("created_at");
            Log.e(TAG, "From: " + from);
            Log.e(TAG, "Title: " + title);
            Log.e(TAG, "message: " + message);
            Log.e(TAG, "image: " + image);
            Log.e(TAG, "timestamp: " + timestamp);

            if (!NotificationUtils.isAppIsInBackground(getApplicationContext()))
{

                // app is in foreground, broadcast the push message
                Intent pushNotification = new Intent(Config.PUSH_NOTIFICATION);
                pushNotification.putExtra("message", message);

LocalBroadcastManager.getInstance(this).sendBroadcast(pushNotification);
```

```
                // play notification sound
                NotificationUtils notificationUtils = new NotificationUtils();
                notificationUtils.playNotificationSound();
            } else {

                Intent resultIntent = new Intent(getApplicationContext(),
MainActivity.class);
                resultIntent.putExtra("message", message);

                if (TextUtils.isEmpty(image)) {
                    showNotificationMessage(getApplicationContext(), title,
message, timestamp, resultIntent);
                } else {
                    showNotificationMessageWithBigImage(getApplicationContext(),
title, message, timestamp, resultIntent, image);
                }
        }
    }

    /**
     * Showing notification with text only
     */
    private void showNotificationMessage(Context context, String title, String
message, String timeStamp, Intent intent) {
        notificationUtils = new NotificationUtils(context);
        intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
        notificationUtils.showNotificationMessage(title, message, timeStamp,
intent);
    }

    /**
     * Showing notification with text and image
     */
    private void showNotificationMessageWithBigImage(Context context, String
title, String message, String timeStamp, Intent intent, String imageUrl) {
        notificationUtils = new NotificationUtils(context);
        intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
        notificationUtils.showNotificationMessage(title, message, timeStamp,
intent, imageUrl);
    }
}
```

NotificationUtils.java

```
package com.studentattendance;

import android.app.ActivityManager;
import android.app.Notification;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.media.Ringtone;
import android.media.RingtoneManager;
import android.net.Uri;
import android.os.Build;
import android.support.v7.app.NotificationCompat;
import android.text.Html;
import android.text.TextUtils;
import android.util.Patterns;
```

```
import java.io.IOException;
import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.URL;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Arrays;
import java.util.Date;
import java.util.List;

/**
 * Created by underhand on 25.05.2016.
 */
public class NotificationUtils {

    private static String TAG = NotificationUtils.class.getSimpleName();

    private Context mContext;

    public NotificationUtils() {
    }

    public NotificationUtils(Context mContext) {
        this.mContext = mContext;
    }

    public void showNotificationMessage(String title, String message, String
timeStamp, Intent intent) {
        showNotificationMessage(title, message, timeStamp, intent, null);
    }

    public void showNotificationMessage(final String title, final String
message, final String timeStamp, Intent intent, String imageUrl) {
        // Check for empty push message
        if (TextUtils.isEmpty(message))
            return;


        // notification icon
        final int icon = R.drawable.ic_launcher;

        intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP |
Intent.FLAG_ACTIVITY_SINGLE_TOP);
        final PendingIntent resultPendingIntent =
                PendingIntent.getActivity(
                        mContext,
                        0,
                        intent,
                        PendingIntent.FLAG_CANCEL_CURRENT
                );

        final NotificationCompat.Builder mBuilder = new
NotificationCompat.Builder(
                mContext);

        final Uri alarmSound =
RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);

        if (!TextUtils.isEmpty(imageUrl)) {

            if (imageUrl != null && imageUrl.length() > 4 &&
Patterns.WEB_URL.matcher(imageUrl).matches()) {
```

```
                Bitmap bitmap = getBitmapFromURL(imageUrl);

                if (bitmap != null) {
                    showBigNotification(bitmap, mBuilder, icon, title, message,
timeStamp, resultPendingIntent, alarmSound);
                } else {
                    showSmallNotification(mBuilder, icon, title, message,
timeStamp, resultPendingIntent, alarmSound);
                }
            }
        } else {
            showSmallNotification(mBuilder, icon, title, message, timeStamp,
resultPendingIntent, alarmSound);
            playNotificationSound();
        }
    }


    private void showSmallNotification(NotificationCompat.Builder mBuilder, int
icon, String title, String message, String timeStamp, PendingIntent
resultPendingIntent, Uri alarmSound) {

        NotificationCompat.InboxStyle inboxStyle = new
NotificationCompat.InboxStyle();

        if (Config.appendNotificationMessages) {
            // store the notification in shared pref first

MyApplication.getInstance().getPrefManager().addNotification(message);

            // get the notifications from shared preferences
            String oldNotification =
MyApplication.getInstance().getPrefManager().getNotifications();

            List<String> messages = Arrays.asList(oldNotification.split("\\|"));

            for (int i = messages.size() - 1; i >= 0; i--) {
                inboxStyle.addLine(messages.get(i));
            }
        } else {
            inboxStyle.addLine(message);
        }


        Notification notification;
        notification = mBuilder.setSmallIcon(icon).setTicker(title).setWhen(0)
                .setAutoCancel(true)
                .setContentTitle(title)
                .setContentIntent(resultPendingIntent)
                .setSound(alarmSound)
                .setStyle(inboxStyle)
                .setWhen(getTimeMilliSec(timeStamp))
                .setSmallIcon(R.drawable.ic_launcher)

.setLargeIcon(BitmapFactory.decodeResource(mContext.getResources(), icon))
                .setContentText(message)
                .build();

        NotificationManager notificationManager = (NotificationManager)
mContext.getSystemService(Context.NOTIFICATION_SERVICE);
        notificationManager.notify(Config.NOTIFICATION_ID, notification);
    }

    private void showBigNotification(Bitmap bitmap, NotificationCompat.Builder
```

```
mBuilder, int icon, String title, String message, String timeStamp,
PendingIntent resultPendingIntent, Uri alarmSound) {
        NotificationCompat.BigPictureStyle bigPictureStyle = new
NotificationCompat.BigPictureStyle();
        bigPictureStyle.setBigContentTitle(title);
        bigPictureStyle.setSummaryText(Html.fromHtml(message).toString());
        bigPictureStyle.bigPicture(bitmap);
        Notification notification;
        notification = mBuilder.setSmallIcon(icon).setTicker(title).setWhen(0)
                .setAutoCancel(true)
                .setContentTitle(title)
                .setContentIntent(resultPendingIntent)
                .setSound(alarmSound)
                .setStyle(bigPictureStyle)
                .setWhen(getTimeMilliSec(timeStamp))
                .setSmallIcon(R.drawable.ic_launcher)

.setLargeIcon(BitmapFactory.decodeResource(mContext.getResources(), icon))
                .setContentText(message)
                .build();

        NotificationManager notificationManager = (NotificationManager)
mContext.getSystemService(Context.NOTIFICATION_SERVICE);
        notificationManager.notify(Config.NOTIFICATION_ID_BIG_IMAGE,
notification);
    }

    /**
     * Downloading push notification image before displaying it in
     * the notification tray
     */
    public Bitmap getBitmapFromURL(String strURL) {
        try {
            URL url = new URL(strURL);
            HttpURLConnection connection = (HttpURLConnection)
url.openConnection();
            connection.setDoInput(true);
            connection.connect();
            InputStream input = connection.getInputStream();
            Bitmap myBitmap = BitmapFactory.decodeStream(input);
            return myBitmap;
        } catch (IOException e) {
            e.printStackTrace();
            return null;
        }
    }

    // Playing notification sound
    public void playNotificationSound() {
        try {
            Uri notification =
RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);
            Ringtone r =
RingtoneManager.getRingtone(MyApplication.getInstance().getApplicationContext(),
notification);
            r.play();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    /**
     * Method checks if the app is in background or not
     */
```

```
    public static boolean isAppIsInBackground(Context context) {
        boolean isInBackground = true;
        ActivityManager am = (ActivityManager)
context.getSystemService(Context.ACTIVITY_SERVICE);
        if (Build.VERSION.SDK_INT > Build.VERSION_CODES.KITKAT_WATCH) {
            List<ActivityManager.RunningAppProcessInfo> runningProcesses =
am.getRunningAppProcesses();
            for (ActivityManager.RunningAppProcessInfo processInfo :
runningProcesses) {
                if (processInfo.importance ==
ActivityManager.RunningAppProcessInfo.IMPORTANCE_FOREGROUND) {
                    for (String activeProcess : processInfo.pkgList) {
                        if (activeProcess.equals(context.getPackageName())) {
                            isInBackground = false;
                        }
                    }
                }
            }
        } else {
            List<ActivityManager.RunningTaskInfo> taskInfo =
am.getRunningTasks(1);
            ComponentName componentInfo = taskInfo.get(0).topActivity;
            if (componentInfo.getPackageName().equals(context.getPackageName()))
{
                isInBackground = false;
            }
        }

        return isInBackground;
    }

    // Clears notification tray messages
    public static void clearNotifications() {
        NotificationManager notificationManager = (NotificationManager)
MyApplication.getInstance().getSystemService(Context.NOTIFICATION_SERVICE);
        notificationManager.cancelAll();
    }

    public static long getTimeMilliSec(String timeStamp) {
        SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
        try {
            Date date = format.parse(timeStamp);
            return date.getTime();
        } catch (ParseException e) {
            e.printStackTrace();
        }
        return 0;
    }
}
```

OnSwipeTouchListener.java

```
package com.studentattendance;

import android.content.Context;
import android.view.GestureDetector;
import android.view.GestureDetector.SimpleOnGestureListener;
import android.view.MotionEvent;
import android.view.View;
import android.view.View.OnTouchListener;

public class OnSwipeTouchListener implements OnTouchListener {

   private final GestureDetector gestureDetector;
```

```
   public OnSwipeTouchListener(Context ctx) {
      gestureDetector = new GestureDetector(ctx, new GestureListener());
   }

   @Override
   public boolean onTouch(View v, MotionEvent event) {
      return gestureDetector.onTouchEvent(event);
   }

   private final class GestureListener extends SimpleOnGestureListener {

      private static final int SWIPE_THRESHOLD = 100;
      private static final int SWIPE_VELOCITY_THRESHOLD = 100;

      @Override
      public boolean onDown(MotionEvent e) {
         return true;
      }

      @Override
      public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX,
float velocityY) {
         boolean result = false;
         try {
            float diffY = e2.getY() - e1.getY();
            float diffX = e2.getX() - e1.getX();
            if (Math.abs(diffX) > Math.abs(diffY)) {
               if (Math.abs(diffX) > SWIPE_THRESHOLD && Math.abs(velocityX) >
SWIPE_VELOCITY_THRESHOLD) {
                  if (diffX > 0) {
                     onSwipeRight();
                  } else {
                     onSwipeLeft();
                  }
               }
               result = true;
            } else if (Math.abs(diffY) > SWIPE_THRESHOLD && Math.abs(velocityY)
> SWIPE_VELOCITY_THRESHOLD) {
               if (diffY > 0) {
                  onSwipeBottom();
               } else {
                  onSwipeTop();
               }
            }
            result = true;

         } catch (Exception exception) {
            exception.printStackTrace();
         }
         return result;
      }
   }

   public void onSwipeRight() {
   }

   public void onSwipeLeft() {
   }

   public void onSwipeTop() {
   }

   public void onSwipeBottom() {
```

```
   }
}
```

ScheduleListAdapter.java

```
package com.studentattendance;

import android.content.Context;
import android.graphics.Typeface;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.ListAdapter;
import android.widget.TextView;

import java.util.ArrayList;
import java.util.HashMap;

public class ScheduleListAdapter extends BaseAdapter implements ListAdapter {
   private ArrayList<HashMap<String, String>> list = new ArrayList<>();
   private Context context;

   public ScheduleListAdapter(ArrayList<HashMap<String, String>> list, Context
context) {
      this.list = list;
      this.context = context;
   }

   @Override
   public int getCount() {
      return list.size();
   }

   @Override
   public Object getItem(int pos) {
      return list.get(pos);
   }

   @Override
   public long getItemId(int pos) {
      // return list.get(pos).getId();
      return 0;
   }

   @Override
   public View getView(final int position, View convertView, ViewGroup parent) {
      View view = convertView;
      if (view == null) {
         LayoutInflater inflater = (LayoutInflater)
context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
         view = inflater.inflate(R.layout.schedule_item, null);
      }

      Typeface subjectFont = Typeface.createFromAsset(context.getAssets(),
"fonts/segoescb.ttf");
      TextView subjectName = (TextView) view.findViewById(R.id.name);
      subjectName.setTypeface(subjectFont);
      subjectName.setText(list.get(position).get("Name"));
      TextView classroom = (TextView) view.findViewById(R.id.classroom);
      classroom.setTypeface(subjectFont);
      classroom.setText(list.get(position).get("Classroom"));
      TextView time = (TextView) view.findViewById(R.id.time);
      time.setTypeface(subjectFont);
      time.setText(list.get(position).get("Time"));
```

```
        return view;
    }
}
```

SpinnerAdapterForStatActivity.java

```
package com.studentattendance;

import android.content.Context;
import android.graphics.Typeface;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.TextView;

public class SpinnerAdapterForStatActivity extends ArrayAdapter<String> {
    Typeface font;

    SpinnerAdapterForStatActivity(Context context, int resource, String[]
spinnerArray) {
        super(context, resource, spinnerArray);
        font = Typeface.createFromAsset(context.getAssets(),
"fonts/segoe_print_bold.ttf");
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        TextView view = (TextView) super.getView(position, convertView, parent);
        return redesignTextView(view);
    }

    @Override
    public View getDropDownView(int position, View convertView, ViewGroup parent)
{
        TextView view = (TextView) super.getDropDownView(position, convertView,
parent);
        return redesignTextView(view);
    }

    private TextView redesignTextView(TextView view) {
        view.setTextSize(20);
        view.setSingleLine(false);
        view.setTypeface(font);
        return view;
    }
}
```

StatisticActivity.java

```
package com.studentattendance;

import android.app.ActionBar;
import android.app.Activity;
import android.graphics.Color;
import android.graphics.drawable.ColorDrawable;
import android.os.Bundle;
import android.text.Html;
import android.view.View;
import android.view.Window;
import android.widget.LinearLayout;
import android.widget.LinearLayout.LayoutParams;
import android.widget.Toast;

import org.achartengine.ChartFactory;
import org.achartengine.GraphicalView;
import org.achartengine.model.CategorySeries;
```

```
import org.achartengine.model.SeriesSelection;
import org.achartengine.renderer.DefaultRenderer;
import org.achartengine.renderer.SimpleSeriesRenderer;

public class StatisticActivity extends Activity {

   private static int[] COLORS = new int[] { Color.RED, Color.GREEN,
Color.YELLOW };

   private static double[] VALUES = new double[] { 75, 25, 15 };

   private static String[] NAME_LIST = new String[] { "Відсутні", "Присутні",
"Запізнилися" };

   private CategorySeries mSeries = new CategorySeries("");

   private DefaultRenderer mRenderer = new DefaultRenderer();

   private GraphicalView mChartView;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      getWindow().requestFeature(Window.FEATURE_ACTION_BAR);
      setContentView(R.layout.activity_statistic_all_subjects);
      ActionBar actionBar = getActionBar();
      actionBar.setBackgroundDrawable(new
ColorDrawable(Color.parseColor("#a71aca")));
      actionBar.setTitle(Html.fromHtml("<font color='#0bf748'></font>"));
      actionBar.show();
      String[] values = (getIntent().getStringExtra("stat")).split(";");
      for (int i = 0; i < values.length; i++) {
         VALUES[i] = Double.parseDouble(values[i]);
      }
      mRenderer.setApplyBackgroundColor(true);
      mRenderer.setBackgroundColor(Color.parseColor("#737373"));
      mRenderer.setChartTitleTextSize(25);
      mRenderer.setLabelsTextSize(25);
      mRenderer.setLabelsColor(Color.BLACK);
      mRenderer.setLegendTextSize(35);
      mRenderer.setMargins(new int[] { 20, 30, 15, 0 });
      mRenderer.setZoomButtonsVisible(true);
      mRenderer.setStartAngle(180);

      for (int i = 0; i < VALUES.length; i++) {
         mSeries.add(NAME_LIST[i] + " " + VALUES[i] + "%", VALUES[i]);
         SimpleSeriesRenderer renderer = new SimpleSeriesRenderer();
         renderer.setColor(COLORS[(mSeries.getItemCount() - 1) %
COLORS.length]);
         mRenderer.addSeriesRenderer(renderer);
      }

      if (mChartView != null) {
         mChartView.repaint();
      }
   }

   @Override
   protected void onResume() {
      super.onResume();
      if (mChartView == null) {
         LinearLayout layout = (LinearLayout) findViewById(R.id.chart);
         mChartView = ChartFactory.getPieChartView(this, mSeries, mRenderer);
         mRenderer.setClickEnabled(true);
```

```
         mRenderer.setSelectableBuffer(10);

         mChartView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
               SeriesSelection seriesSelection =
mChartView.getCurrentSeriesAndPoint();

               if (seriesSelection == null) {
                  Toast.makeText(StatisticActivity.this, "No chart element was
clicked", Toast.LENGTH_SHORT)
                        .show();
               } else {
                  Toast.makeText(StatisticActivity.this,
                        "Chart element data point index " +
(seriesSelection.getPointIndex() + 1)
                              + " was clicked" + " point value=" +
seriesSelection.getValue(),
                        Toast.LENGTH_SHORT).show();
               }
            }
         });

         mChartView.setOnLongClickListener(new View.OnLongClickListener() {
            @Override
            public boolean onLongClick(View v) {
               SeriesSelection seriesSelection =
mChartView.getCurrentSeriesAndPoint();
               if (seriesSelection == null) {
                  // Toast.makeText(StatisticActivity.this, "No chart
                  // element was long pressed", Toast.LENGTH_SHORT);
                  return false;
               } else {
                  // Toast.makeText(StatisticActivity.this, "Chart element
                  // data point index "
                  // + seriesSelection.getPointIndex() + " was long
                  // pressed", Toast.LENGTH_SHORT);
                  return true;
               }
            }
         });
         layout.addView(mChartView, new LayoutParams(LayoutParams.MATCH_PARENT,
LayoutParams.MATCH_PARENT));
      } else {
         mChartView.repaint();
      }
   }
}
```

StatisticTypeActivity.java

```
package com.studentattendance;

import android.app.ActionBar;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.graphics.Color;
import android.graphics.Typeface;
import android.graphics.drawable.ColorDrawable;
import android.os.Bundle;
import android.text.Html;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
```

```
import android.view.View;
import android.view.Window;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemSelectedListener;
import android.widget.Button;
import android.widget.Spinner;
import android.widget.TextView;

import com.google.gson.Gson;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Set;

public class StatisticTypeActivity extends ILocalScheduleLoaderActivity
implements View.OnClickListener {

   SharedPreferences sPref;
   Button Ok;
   Spinner subjectSpinner;
   Spinner groupeSpinner;
   ArrayList<HashMap<String, String>> scheduleList, groupsList,
filteredGroupsList;
   String statData = "";

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      getWindow().requestFeature(Window.FEATURE_ACTION_BAR);
      setContentView(R.layout.activity_statistic_type);
      ActionBar actionBar = getActionBar();
      actionBar.setBackgroundDrawable(new
ColorDrawable(Color.parseColor("#a71aca")));
      actionBar.setTitle(Html.fromHtml("<font color='#0bf748'> </font>"));
      actionBar.show();
      Typeface myTypeface = Typeface.createFromAsset(getAssets(),
"fonts/v_CCForkedTongue.ttf");
      TextView selectFiltersTV = (TextView) findViewById(R.id.selectFiltersTV);
      selectFiltersTV.setTypeface(myTypeface);
      subjectSpinner = (Spinner) findViewById(R.id.subjectsSpinner);
      setOnSpinnerItemSelected(subjectSpinner);
      sPref = this.getSharedPreferences("com.studentattendance",
Context.MODE_PRIVATE);
      if ((sPref.getString("UserType", "")).equals("t")) {
         groupeSpinner = (Spinner) findViewById(R.id.groupeSpinner);
         groupeSpinner.setEnabled(true);
         groupeSpinner.setVisibility(View.VISIBLE);
      }
      Ok = (Button) findViewById(R.id.ok);
      Ok.setOnClickListener(this);
      new LocalScheduleLoader(StatisticTypeActivity.this, this, "").execute();
      new LocalGroupListLoader(StatisticTypeActivity.this, this, "All_Subjects",
"").execute();
   }

   public void onClick(View v) {
      new GetStatisticData(this, this,
getSelectedSubjectsAndGroupsJson()).execute();
   }

   public void setOnSpinnerItemSelected(Spinner spinner) {
      spinner.setOnItemSelectedListener(new OnItemSelectedListener() {
```

```
         @Override
         public void onItemSelected(AdapterView<?> parentView, View
selectedItemView, int position, long id) {
            refreshGroupsSpinnerOnSubjectChange();
         }

         @Override
         public void onNothingSelected(AdapterView<?> parent) {
            // TODO Auto-generated method stub
         }
      });
   }

   @Override
   public boolean onCreateOptionsMenu(Menu menu) {
      MenuInflater inflater = getMenuInflater();
      inflater.inflate(R.menu.statistic_type_activity_actions, menu);
      return super.onCreateOptionsMenu(menu);
   }

   @Override
   public boolean onMenuItemSelected(int featureId, MenuItem item) {
      int itemId = item.getItemId();
      switch (itemId) {
      case R.id.action_help:
         Intent helpActivity = new Intent(getApplicationContext(),
HelpActivity.class);
         startActivity(helpActivity);
         break;
      }
      return true;
   }

   void CallActivityForSelectedFilters(String statData) {
      startActivity(new Intent(getApplicationContext(),
StatisticActivity.class).putExtra("stat", statData));
   }

   String getSelectedSubjectsAndGroupsJson() {
      return new Gson().toJson(getSlectedSubjectsAndGroupsList());
   }

   ArrayList<HashMap<String, String>> getSlectedSubjectsAndGroupsList() {
      ArrayList<HashMap<String, String>> subjectAndGroupePairs = new
ArrayList<>();
      if (subjectSpinner.getSelectedItem().equals("Всі предмети"))
         for (int i = 0; i < scheduleList.size(); i++)
            addAllGroupseForSubject(subjectAndGroupePairs, scheduleList.get(i));
      else
         for (int i = 0; i < scheduleList.size(); i++)
            if
(scheduleList.get(i).get("Name").equals(subjectSpinner.getSelectedItem())) {
               addAllGroupseForSubject(subjectAndGroupePairs,
scheduleList.get(i));
            }
      return subjectAndGroupePairs;
   }

   void addAllGroupseForSubject(ArrayList<HashMap<String, String>>
subjectAndGroupePairs,
         HashMap<String, String> subject) {
      for (int j = 0; j < groupsList.size(); j++) {
         if (groupsList.get(j).get("SubjectID").equals(subject.get("ID"))) {
            HashMap<String, String> hm = new HashMap<String, String>();
```

```
            hm.put("SubjectID", subject.get("ID").toString());
            hm.put("ClassID", groupsList.get(j).get("ClassID").toString());
            subjectAndGroupePairs.add(hm);
         }
      }
   }

   @Override
   void getScheduleData(ArrayList<HashMap<String, String>> scheduleList) {
      if (!scheduleList.isEmpty()) {
         this.scheduleList = scheduleList;
         setSubjectSpinnerAdapter(scheduleList);
      }
   }

   void setGroupeSpinnerData(ArrayList<HashMap<String, String>> groupsList) {
      this.groupsList = groupsList;
      setGroupeSpinnerAdapter(groupsList);
   }

   void refreshGroupsSpinnerOnSubjectChange() {
      if (!subjectSpinner.getSelectedItem().equals("Всі предмети")) {
         getGroupsForSpecificSubject();
         groupsList = filteredGroupsList;
         setGroupeSpinnerAdapter(filteredGroupsList);
      } else {
         new LocalGroupListLoader(StatisticTypeActivity.this, this,
"All_Subjects", "").execute();
         setGroupeSpinnerAdapter(groupsList);
      }
   }

   void getGroupsForSpecificSubject() {
      filteredGroupsList = new ArrayList<>();
      for (int i = 0; i < scheduleList.size(); i++)
         if
(scheduleList.get(i).get("Name").equals(subjectSpinner.getSelectedItem())) {
            new LocalGroupListLoader(StatisticTypeActivity.this, this,
scheduleList.get(i).get("ID"), "").execute();
            filteredGroupsList.addAll(groupsList);
         }
   }

   void setSubjectSpinnerAdapter(ArrayList<HashMap<String, String>> subjectList)
{
      SpinnerAdapterForStatActivity adapter = new
SpinnerAdapterForStatActivity(StatisticTypeActivity.this,
            android.R.layout.simple_spinner_item,
            getArrayWithFilteredDataForSpinnerAdapter(subjectList, "Всі
предмети"));

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
      subjectSpinner.setAdapter(adapter);
   }

   void setGroupeSpinnerAdapter(ArrayList<HashMap<String, String>> groupsList) {
      SpinnerAdapterForStatActivity adapter = new
SpinnerAdapterForStatActivity(StatisticTypeActivity.this,
            android.R.layout.simple_spinner_item,
            getArrayWithFilteredDataForSpinnerAdapter(groupsList, "Всі групи"));

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
      groupeSpinner.setAdapter(adapter);
   }
```

```
   String[] getArrayWithFilteredDataForSpinnerAdapter(ArrayList<HashMap<String,
String>> list, String title) {
      Set<String> hs = new HashSet<>();
      hs.add(title);
      for (int i = 0; i < list.size(); i++) {
         hs.add(list.get(i).get("Name"));
      }
      String[] spinnerArray = new String[hs.size()];
      hs.toArray(spinnerArray);
      Arrays.sort(spinnerArray);
      return spinnerArray;
   }

}
```

StudentMainScheduleActivity.java

```
package com.studentattendance;

import android.app.ActionBar;
import android.app.Activity;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.SharedPreferences;
import android.graphics.Color;
import android.graphics.Typeface;
import android.graphics.drawable.ColorDrawable;
import android.os.AsyncTask;
import android.os.Bundle;
import android.text.Html;
import android.util.Log;
import android.widget.ImageView;
import android.widget.ListView;
import android.widget.RelativeLayout;
import android.widget.TextView;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.HashMap;
import java.util.Locale;

public class StudentMainScheduleActivity extends Activity {

   SharedPreferences sPref;
   private ProgressDialog pDialog;
   JSONObject jsonSchedule;
   ArrayList<HashMap<String, String>> scheduleList;
   JSONArray days = null;
   ListView lv;
   String dayOfWeek;
   static Calendar dateChosen;
   TextView dayAndDateStatus;
   ImageView youFreeImg;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.student_main_schedule_activity);
      ActionBar actionBar = getActionBar();
```

```
      actionBar.setBackgroundDrawable(new
ColorDrawable(Color.parseColor("#a71aca")));
      actionBar.setTitle(Html.fromHtml("<font color='#0bf748'> </font>"));
      actionBar.show();
      youFreeImg = (ImageView) findViewById(R.id.youFreeImage);
      dayAndDateStatus = (TextView) findViewById(R.id.day_of_week);
      Typeface myTypeface = Typeface.createFromAsset(getAssets(),
"fonts/v_CCForkedTongue.ttf");
      dayAndDateStatus.setTypeface(myTypeface);
      dayOfWeek =
Calendar.getInstance(Locale.US).getDisplayName(Calendar.DAY_OF_WEEK,
Calendar.LONG, Locale.US);
      dateChosen = Calendar.getInstance();
      dayAndDateStatus

.setText(Calendar.getInstance(Locale.US).getDisplayName(Calendar.DAY_OF_WEEK,
Calendar.LONG, Locale.getDefault())
                  + ", " + new
SimpleDateFormat("dd.MM.yyyy").format(dateChosen.getTime()));
      sPref = this.getSharedPreferences("com.studentattendance",
Context.MODE_PRIVATE);
      try {
         days = new JSONArray(sPref.getString("Schedule", ""));
      } catch (JSONException e) {
         Log.e("Er:", Log.getStackTraceString(e));
      }
      scheduleList = new ArrayList<HashMap<String, String>>();
      lv = (ListView) findViewById(R.id.schedule);
      new LoadSchedule().execute();

      RelativeLayout rLayout = (RelativeLayout) findViewById(R.id.relLayout);
      rLayout.setOnTouchListener(new
OnSwipeTouchListener(StudentMainScheduleActivity.this) {
         public void onSwipeTop() {
            DateDayChanger(7);
         }

         public void onSwipeRight() {
            DateDayChanger(-1);

         }

         public void onSwipeLeft() {
            DateDayChanger(1);
         }

         public void onSwipeBottom() {
            DateDayChanger(-7);
         }

      });

      lv.setOnTouchListener(new
OnSwipeTouchListener(StudentMainScheduleActivity.this) {

         public void onSwipeRight() {
            DateDayChanger(-1);
         }

         public void onSwipeLeft() {
            DateDayChanger(1);
         }

      });
```

```
    }

    public void DateDayChanger(int daysToChange) {
        youFreeImg.setBackgroundResource(0);
        dateChosen.add(Calendar.DATE, daysToChange);
        dayOfWeek = dateChosen.getDisplayName(Calendar.DAY_OF_WEEK, Calendar.LONG,
Locale.US);
        dayAndDateStatus.setText(dateChosen.getDisplayName(Calendar.DAY_OF_WEEK,
Calendar.LONG, Locale.getDefault()) + ", "
                + new SimpleDateFormat("dd.MM.yyyy").format(dateChosen.getTime()));
        lv.setAdapter(null);
        scheduleList.clear();
        new LoadSchedule().execute();
    }

    class LoadSchedule extends AsyncTask<String, String, String> {

        @Override
        protected void onPreExecute() {
            super.onPreExecute();
            pDialog = new ProgressDialog(StudentMainScheduleActivity.this);
            pDialog.setMessage("Завантаження даних, зачекайте будь ласка...");
            pDialog.setIndeterminate(false);
            pDialog.setCancelable(false);
            pDialog.show();
        }

        protected String doInBackground(String... args) {
            try {

                for (int i = 0; i < days.length(); i++) {
                    JSONObject day = days.getJSONObject(i);
                    Log.e("Er:", day.toString());
                    JSONArray d = day.getJSONArray(dayOfWeek);
                    for (int j = 0; j < d.length(); j++) {
                        JSONObject subject = d.getJSONObject(j);

                        String id = subject.getString("ID");
                        String name = subject.getString("Name");
                        String dayOfWeek = subject.getString("DayOfWeek");
                        String classroom = "Classroom:" +
subject.getString("Classroom");
                        String time = subject.getString("Time");

                        HashMap<String, String> map = new HashMap<String, String>();

                        map.put("ID", id);
                        map.put("Name", name);
                        map.put("Classroom", classroom);
                        map.put("DayOfWeek", dayOfWeek);
                        map.put("Time", time);

                        scheduleList.add(map);

                    }

                }
            } catch (JSONException e) {
                Log.e("Er:", Log.getStackTraceString(e));
            }

            return null;
        }
```

```
      protected void onPostExecute(String file_url) {
         pDialog.dismiss();
         runOnUiThread(new Runnable() {
            public void run() {

               if (!scheduleList.isEmpty()) {
                  ScheduleListAdapter scheduleAdapter = new
ScheduleListAdapter(scheduleList, StudentMainScheduleActivity.this);
                  lv.setAdapter(scheduleAdapter);
               } else {
                  dayAndDateStatus.append("\n\n" + "Занять немає!\n");

youFreeImg.setBackgroundResource(R.drawable.ic_green_happy_man);
               }
            }
         });

      }

   }
}
```

TeacherMainScheduleActivity.java

```
package com.studentattendance;

import android.annotation.SuppressLint;
import android.app.ActionBar;
import android.app.DatePickerDialog;
import android.app.Dialog;
import android.app.DialogFragment;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.graphics.Color;
import android.graphics.Typeface;
import android.graphics.drawable.ColorDrawable;
import android.os.Bundle;
import android.text.Html;
import android.util.Log;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.Window;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.DatePicker;
import android.widget.ImageView;
import android.widget.ListView;
import android.widget.RelativeLayout;
import android.widget.TextView;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.HashMap;
import java.util.Locale;
```

```
public class TeacherMainScheduleActivity extends ILocalScheduleLoaderActivity {

   SharedPreferences sPref;
   private ProgressDialog pDialog;
   JSONObject jsonSchedule;
   ArrayList<HashMap<String, String>> scheduleList;
   JSONArray days = null;
   ListView lv;
   String dayOfWeek;
   static Calendar dateChosen;
   TextView dayAndDateStatus;
   ImageView youFreeImg;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      getWindow().requestFeature(Window.FEATURE_ACTION_BAR);
      setContentView(R.layout.teacher_main_schedule_activity);
      ActionBar actionBar = getActionBar();
      actionBar.setBackgroundDrawable(new
ColorDrawable(Color.parseColor("#a71aca")));
      actionBar.setTitle(Html.fromHtml("<font color='#0bf748'> </font>"));
      actionBar.show();
      youFreeImg = (ImageView) findViewById(R.id.youFreeImage);
      dayAndDateStatus = (TextView) findViewById(R.id.day_of_week);
      Typeface myTypeface = Typeface.createFromAsset(getAssets(),
"fonts/v_CCForkedTongue.ttf");
      dayAndDateStatus.setTypeface(myTypeface);
      dayOfWeek =
Calendar.getInstance(Locale.US).getDisplayName(Calendar.DAY_OF_WEEK,
Calendar.LONG,  Locale.US);
      dateChosen = Calendar.getInstance();
      dayAndDateStatus

.setText(Calendar.getInstance(Locale.US).getDisplayName(Calendar.DAY_OF_WEEK,
Calendar.LONG, Locale.getDefault())
                  + ", " + new
SimpleDateFormat("dd.MM.yyyy").format(dateChosen.getTime()));
      sPref = this.getSharedPreferences("com.studentattendance",
Context.MODE_PRIVATE);
      try {
         days = new JSONArray(sPref.getString("Schedule", ""));
      } catch (JSONException e) {
         Log.e("Er:", Log.getStackTraceString(e));
      }
      scheduleList = new ArrayList<HashMap<String, String>>();
      lv = (ListView) findViewById(R.id.schedule);
      new LocalScheduleLoader(TeacherMainScheduleActivity.this, this,
dayOfWeek).execute();
      lv.setOnItemClickListener(new OnItemClickListener() {
         public void onItemClick(AdapterView<?> parent, View view, int position,
long id) {
            String pid = scheduleList.get(position).get("ID");
            String dayAndTime =
scheduleList.get(position).get("DayOfWeek").concat(" ")
                  .concat(new
SimpleDateFormat("yyyy.MM.dd").format(dateChosen.getTime()))
                  .concat(scheduleList.get(position).get("Time"));
            Intent in = new Intent(getApplicationContext(),
AttendanceTaker.class);
            in.putExtra("ID", pid.concat(" ").concat(dayAndTime));

            startActivityForResult(in, 100);
         }
```

```
      });

      RelativeLayout rLayout = (RelativeLayout) findViewById(R.id.relLayout);
      rLayout.setOnTouchListener(new
OnSwipeTouchListener(TeacherMainScheduleActivity.this) {
         public void onSwipeTop() {
            DateDayChanger(7);
         }

         public void onSwipeRight() {
            DateDayChanger(-1);

         }

         public void onSwipeLeft() {
            DateDayChanger(1);
         }

         public void onSwipeBottom() {
            DateDayChanger(-7);
         }

      });

      lv.setOnTouchListener(new
OnSwipeTouchListener(TeacherMainScheduleActivity.this) {

         public void onSwipeRight() {
            DateDayChanger(-1);
         }

         public void onSwipeLeft() {
            DateDayChanger(1);
         }

      });

   }

   @Override
   public boolean onCreateOptionsMenu(Menu menu) {
      MenuInflater inflater = getMenuInflater();
      inflater.inflate(R.menu.teacher_main_actions, menu);
      return super.onCreateOptionsMenu(menu);
   }

   @Override
   public boolean onMenuItemSelected(int featureId, MenuItem item) {
      int itemId = item.getItemId();
      switch (itemId) {
      case R.id.previous:
         DateDayChanger(-1);
         break;
      case R.id.next:
         DateDayChanger(1);
         break;
      case R.id.calendar:
         Calendar c1 = Calendar.getInstance(Locale.US);
         c1.setTime(new Date());
         if (dateChosen.get(Calendar.DAY_OF_YEAR) !=
c1.get(Calendar.DAY_OF_YEAR)
               || dateChosen.get(Calendar.YEAR) != c1.get(Calendar.YEAR)) {
            dateChosen = c1;
            DateDayChanger(0);
```

```
            } else {
               showDatePickerDialog(this.findViewById(android.R.id.content));
            }
            break;
         case R.id.action_messages:
            Intent messageActivvity = new Intent(getApplicationContext(),
MessagesActivity.class);
            startActivity(messageActivvity);
            break;
         case R.id.action_help:
            Intent helpActivity = new Intent(getApplicationContext(),
HelpActivity.class);
            startActivity(helpActivity);
            break;
         case R.id.action_statistic:
            Intent statisticActivity = new Intent(getApplicationContext(),
StatisticTypeActivity.class);
            startActivity(statisticActivity);
            break;
         }
         return true;
      }

      public void showDatePickerDialog(View v) {
         DialogFragment newFragment = new DatePickerFragment();
         newFragment.show(getFragmentManager(), "datePicker");
      }

      public void DateDayChanger(int daysToChange) {
         youFreeImg.setBackgroundResource(0);
         dateChosen.add(Calendar.DATE, daysToChange);
         dayOfWeek = dateChosen.getDisplayName(Calendar.DAY_OF_WEEK, Calendar.LONG,
Locale.US);
         dayAndDateStatus.setText(dateChosen.getDisplayName(Calendar.DAY_OF_WEEK,
Calendar.LONG, Locale.getDefault()) + ", "
               + new SimpleDateFormat("dd.MM.yyyy").format(dateChosen.getTime()));
         lv.setAdapter(null);
         scheduleList.clear();
         new LocalScheduleLoader(TeacherMainScheduleActivity.this,
TeacherMainScheduleActivity.this, dayOfWeek)
               .execute();
      }

      @Override
      void getScheduleData(ArrayList<HashMap<String, String>> scheduleList) {
         this.scheduleList = scheduleList;
         OnScheduleLoading();
         Log.e("Name:", "12131313");
      }

      void OnScheduleLoading() {
         if (!scheduleList.isEmpty()) {
            ScheduleListAdapter scheduleAdapter = new
ScheduleListAdapter(scheduleList,
                  TeacherMainScheduleActivity.this);
            lv.setAdapter(scheduleAdapter);
         } else {
            dayAndDateStatus.append("\n\n" + "Занять немає!\n");
            youFreeImg.setBackgroundResource(R.drawable.ic_green_happy_man);
         }
      }

      @SuppressLint("ValidFragment")
      public class DatePickerFragment extends DialogFragment implements
```

```
DatePickerDialog.OnDateSetListener {

      @Override
      public Dialog onCreateDialog(Bundle savedInstanceState) {
         final Calendar c = Calendar.getInstance();
         int year = c.get(Calendar.YEAR);
         int month = c.get(Calendar.MONTH);
         int day = c.get(Calendar.DAY_OF_MONTH);
         return new DatePickerDialog(getActivity(), this, year, month, day);
      }

      public void onDateSet(DatePicker view, int year, int month, int day) {
         dateChosen.set(year, month, day);
         DateDayChanger(0);
      }
   }
}
```

Schedule.php

```
<?php
header('Content-type: text/plain; charset=utf-8');
if (isset($_POST['ID']) and isset($_POST['UserType'])){
$response = array();
// include db connect class
require_once __DIR__ . '/connector.php';
// connecting to db
$con=connect();
   // echo $con;
if ($con!=null) {
// get all  from  table
mysqli_set_charset($con,"utf8");
$ID=mysqli_real_escape_string($con,$_POST['ID']);
If($_POST['UserType']=="s")
{
    $stmt = mysqli_prepare($con, 'CALL GetStudentSubjects(?)');
    mysqli_stmt_bind_param($stmt, 'i', $ID);
    mysqli_stmt_execute($stmt);
    $result = $stmt->get_result();
}
    else
    {
        $stmt = mysqli_prepare($con, 'SELECT * FROM Subject WHERE TeacherID like
? ORDER BY Subject.DayOfWeek');
        mysqli_stmt_bind_param($stmt, 'i', $ID);
        mysqli_stmt_execute($stmt);
        $result = $stmt->get_result();
    }
close($con);
// check for empty result
if (mysqli_num_rows($result) > 0) {
    $response["Schedule"] = array();
    $dayName=null;
    while ($row = mysqli_fetch_array($result)) {
         if($dayName==null)
        {
            $dayName=$row["DayOfWeek"];
            $day[$dayName]=array();
        }
        else if($dayName!=$row["DayOfWeek"])
        {
            $dayName=$row["DayOfWeek"];
            $day[$dayName]=array();
        }
```

```
        $subject=array();
        $subject["ID"]=$row["ID"];
        $subject["Name"]=$row["Name"];
        $subject["Type"]=$row["Type"];
        $subject["Classroom"]=$row["Classroom"];
        $subject["DayOfWeek"]=$row["DayOfWeek"];
        $subject["Time"]=$row["Time"];
        $subject["WeekParity"]=$row["WeekParity"];
        $subject["Durability"]=$row["Durability"];
        $subject["TeacherID"]=$row["TeacherID"];
        array_push($day[$dayName] , $subject);
    }
    array_push($response["Schedule"] , $day);
    // success
    $response["success"] = 1;

    // echoing JSON response
    echo json_encode($response,JSON_UNESCAPED_UNICODE);
} else {
    // no products found
    $response["success"] = 0;
    $response["message"] = "User not exist";
    // echo no users JSON
    echo json_encode($response,JSON_UNESCAPED_UNICODE);
    }
}
else {
    // required field is missing
    $response["success"] = 0;
    $response["message"] ="Connection failed ";
    // echoing JSON response
    echo json_encode($response,JSON_UNESCAPED_UNICODE);
}}
else {
    // required field is missing
    $response["success"] = 0;
    $response["message"] = "Required field(s) is missing";

    // echoing JSON response
    echo json_encode($response,JSON_UNESCAPED_UNICODE);
}
?>
```

studentList.php

```
<?php
header('Content-type: text/plain; charset=utf-8');
if (isset($_POST['ID'])){
$response = array();
// include db connect class
require_once __DIR__ . '/connector.php';
// connecting to db
$con=connect();
   // echo $con;
if ($con!=null) {
// get all  from  table
mysqli_set_charset($con,"utf8");
$ID=mysqli_real_escape_string($con,$_POST['ID']);
    $stmt = mysqli_prepare($con, 'CALL GetStudents(?)');
    mysqli_stmt_bind_param($stmt, 'i', $ID);
    mysqli_stmt_execute($stmt);
    $result = $stmt->get_result();
close($con);
// check for empty result
if (mysqli_num_rows($result) > 0) {
```

```
    $response["Classes"] = array();
    $className=null;
    while ($row = mysqli_fetch_array($result)) {
        if($className==null)
        {
            $className=$row["ClassID"];
            $Class[$className]=array();
        }
        else if($className!=$row["ClassID"])
        {
            $className=$row["ClassID"];
            $Class[$className]=array();
        }
        $student=array();
        $student["ID"]=$row["ID"];
        $student["Name"]=$row["Name"];
        $student["Surname"]=$row["Surname"];
        $student["Email"]=$row["Email"];
        array_push($Class[$className] , $student );

    }
     array_push($response["Classes"] , $Class);
    // success
    $response["success"] = 1;

    // echoing JSON response
    echo json_encode($response,JSON_UNESCAPED_UNICODE);
} else {
    // no found
    $response["success"] = 0;
    $response["message"] = "User not exist";
    // echo no users JSON
    echo json_encode($response,JSON_UNESCAPED_UNICODE);
    }
}
else {
    // required field is missing
    $response["success"] = 0;
    $response["message"] ="Connection failed ";
    // echoing JSON response
    echo json_encode($response,JSON_UNESCAPED_UNICODE);
}}
else {
    // required field is missing
    $response["success"] = 0;
    $response["message"] = "Required field(s) is missing";

    // echoing JSON response
    echo json_encode($response,JSON_UNESCAPED_UNICODE);
}
?>
```

Class_subject.php

```
<?php
header('Content-type: text/plain; charset=utf-8');
if (isset($_POST['ID'])){
$response = array();
// include db connect class
require_once __DIR__ . '/connector.php';
// connecting to db
$con=connect();
   // echo $con;
if ($con!=null) {
// get all  from  table
```

```
mysqli_set_charset($con,"utf8");
$ID=mysqli_real_escape_string($con,$_POST['ID']);
    $stmt = mysqli_prepare($con, 'CALL GetClassSubjects(?)');
    mysqli_stmt_bind_param($stmt, 'i', $ID);
    mysqli_stmt_execute($stmt);
    $result = $stmt->get_result();
close($con);
// check for empty result
if (mysqli_num_rows($result) > 0) {
    $response["ClassSubjects"] = array();
    while ($row = mysqli_fetch_array($result)) {
        $ClassSubject=array();
        $ClassSubject["ClassID"]=$row["ClassID"];
        $ClassSubject["Name"]=$row["Name"];
        $ClassSubject["SubjectID"]=$row["SubjectID"];
        array_push($response["ClassSubjects"] , $ClassSubject);
    }

    // success
    $response["success"] = 1;

    // echoing JSON response
    echo json_encode($response,JSON_UNESCAPED_UNICODE);
} else {
    // no found
    $response["success"] = 0;
    $response["message"] = "User not exist";
    // echo no users JSON
    echo json_encode($response,JSON_UNESCAPED_UNICODE);
    }
}
else {
    // required field is missing
    $response["success"] = 0;
    $response["message"] ="Connection failed ";
    // echoing JSON response
    echo json_encode($response,JSON_UNESCAPED_UNICODE);
}}
else {
    // required field is missing
    $response["success"] = 0;
    $response["message"] = "Required field(s) is missing";

    // echoing JSON response
    echo json_encode($response,JSON_UNESCAPED_UNICODE);
}
?>
```

statisticForTeacher.php

```
<?php
header('Content-type: text/plain; charset=utf-8');
if (isset($_POST['SubjectsAndGroupsList'])){
$response = array();
$jsonData = array();
$SubjectsAndGroupsList=json_decode($_POST['SubjectsAndGroupsList'], true);


require_once __DIR__ . '/connector.php';

$con=connect();
$absent=0;
$present=0;
$late=0;
```

```
if ($con!=null) {
mysqli_set_charset($con,"utf8");
        foreach($SubjectsAndGroupsList as $subjectAndGroupe)
        {
        $stmt = mysqli_prepare($con, 'CALL GetStatForTeacher(?,?)');
        mysqli_stmt_bind_param($stmt, 'ii',
$subjectAndGroupe["SubjectID"],$subjectAndGroupe["ClassID"]);
        $stmt->execute();
        $result = $stmt->get_result();
        $row=mysqli_fetch_assoc($result);
        $absent+=$row['Absent'];
        $present+=$row['Present'];
        $late+=$row['Late'];
        $stmt->close();
        }
        close($con);
        $stat=$absent.";".$present.";".$late;
        $response["success"] = 1;
        $response["message"] = "Transaction Successful";
        $response["stat"]=$stat;
        echo json_encode($response,JSON_UNESCAPED_UNICODE);

}
else {
    $response["success"] = 0;
    $response["message"] ="Connection failed ";
    echo json_encode($response,JSON_UNESCAPED_UNICODE);
}}
else {
    $response["success"] = 0;
    $response["message"] = "Required field(s) is missing";

    echo json_encode($response,JSON_UNESCAPED_UNICODE);
}
?>
```

Help_activity.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="${relativePackage}.${activityClass}" >
    <ScrollView
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:fillViewport="true"> <!--IMPORTANT otherwise backgrnd img. will not fill
the whole screen -->

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context="${relativePackage}.${activityClass}" >
    <TextView
        android:id="@+id/mainWindowTV"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:gravity="center"
        android:text="@string/mainWindowHelpTitle"
        android:textSize="22sp"/>

    <TextView
        android:id="@+id/mainWindowText1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/mainWindowTV"
        android:layout_centerHorizontal="true"
```

```
        android:layout_marginLeft="2dp"
        android:layout_marginRight="2dp"
        android:gravity="fill_horizontal"
        android:text="@string/mainWindowText1"
        android:textSize="18sp"/>
    <ImageView
        android:id="@+id/mainWindowImg1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/mainWindowText1"
        android:src="@drawable/main_window_schedule"/>
    <TextView
        android:id="@+id/mainWindowText2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/mainWindowImg1"
        android:layout_marginLeft="2dp"
        android:layout_marginRight="2dp"
        android:layout_centerHorizontal="true"
        android:gravity="fill_horizontal"
        android:text="@string/mainWindowText2"
        android:textSize="18sp"/>
        <ImageView
            android:id="@+id/mainWindowImg2"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_below="@+id/mainWindowText2"
            android:src="@drawable/main_window_schedule_navigation"/>
</RelativeLayout>
    </ScrollView>
</RelativeLayout>
```

List_loader_for_statistic.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="${relativePackage}.${activityClass}" >

  <ListView
        android:id="@+id/listV"
        android:background="@drawable/lv_background"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="1dp"
        android:layout_marginLeft="1dp"
        android:layout_marginRight="1dp"/>
<Button
         android:id="@+id/ok"
         android:onClick="onOKStart"
         android:text="Chose"
         android:textSize="16sp"
           android:layout_alignParentBottom="true"
           android:layout_centerHorizontal="true"
           android:layout_width="fill_parent"
           android:layout_height="wrap_content"
           />

</RelativeLayout>
```

activity_login.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/lbg"
    tools:context="${relativePackage}.${activityClass}" >
   <Button
      android:id="@+id/help"
      android:layout_width="50dp"
      android:layout_height="60dp"
```

```
        android:layout_alignParentRight="true"
        android:background="@android:color/transparent"/>

    <TextView
        android:id="@+id/enter_emailTV"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:gravity="center"
        android:text="Введіть Ваш емейл,\n будь ласка :)"
        android:textSize="32sp"
        android:layout_marginTop="170dp"
        android:layout_marginBottom="20dp"
        />
        <EditText
              android:id="@+id/id"
              android:layout_above="@+id/ok"
              android:layout_marginBottom="120dp"
              android:layout_width="match_parent"
              android:layout_height="wrap_content"
              android:hint="@string/enterID"
              android:inputType="textEmailAddress" />
        <Button
           android:id="@+id/ok"
             android:layout_alignParentBottom="true"
             android:layout_centerHorizontal="true"
             android:layout_width="110dp"
             android:layout_height="114dp"
             android:background="@android:color/transparent"/>
</RelativeLayout>
```

Teacher_main_schedule_activity.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/relLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/notepad"
    tools:context="${relativePackage}.${activityClass}" >
    <RelativeLayout
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:orientation="horizontal"
   android:layout_marginLeft="35dp"
   android:layout_marginTop="30dp"
   android:layout_marginRight="3dp">
   <TextView
   android:id="@+id/day_of_week"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
    android:gravity="center"
    android:layout_centerHorizontal="true"
   android:layout_marginBottom="35dp"
   android:textSize="28sp">
   </TextView>
    <ListView
        android:id="@+id/schedule"
        android:background="@drawable/lv_background"
        android:layout_below="@+id/day_of_week"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginRight="5dp"/>
    <ImageView
        android:id="@+id/youFreeImage"
        android:layout_below="@+id/day_of_week"
        android:layout_width="wrap_content"
        android:gravity="center"
        android:layout_centerHorizontal="true"
        android:layout_height="wrap_content"
        android:layout_marginLeft="35dp"
```

```
            android:layout_marginRight="5dp"/>
                </RelativeLayout>

    </RelativeLayout>
```