Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка мобільного додатку для проведення презентацій

Виконав: студент IV курсу, групи СН-41
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)

(підпис) Лесяк В.І. (прізвище та ініціали)

Керівник (підпис) Матійчук Л.П. (прізвище та ініціали)

Нормоконтроль (підпис) Шимчук Г.В. (прізвище та ініціали)

Завідувач кафедри (підпис) Боднарчук І.О. (прізвище та ініціали)

Рецензент (підпис) Крамар О.І. (прізвище та ініціали)

Тернопіль
2021

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«__» _____ 2021 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Лесяк Віталій Ігорович
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка мобільного додатку для проведення презентацій

Керівник роботи Матійчук Любомир Павлович, к.е.н, доцент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від 02 » березня 2021 року № 4/7-171

2. Термін подання студентом завершеної роботи 23 червня 2021р.

3. Вихідні дані до роботи Літературні джерела за тематикою дослідження

4. Зміст роботи (перелік питань, які потрібно розробити)
1. Аналіз предметної області та проектування додатку проведення онлайн презентацій. 1.1. Опис та аналіз існуючих аналогів, що реалізують функції предметної області. 1.2. Специфікація вимог до системи. 1.3. Розроблення архітектури програмної системи.1.4. Реалізація асинхронності роботи додатку. 2. Програмна реалізація тестування та длслідна експлуатація додатку для проведення презентацій. 2.1. Обґрунтування вибору засобів реалізації мобільного додатку. 2.2. Програмна реалізація серверної частини додатку. 2.3.Програмна реалізація бази даних. 2.4. Тестування додатку. 2.5. Інструкція користувача.
3. Безпека життєдіяльності, основи хорони праці. Висновки. Перелік джерел.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
Актуальність дослідження. Діаграма варіантів використання. Огляд і аналіз існуючих аналогів. Діаграма послідовності. Ескіз екранних форм. Компоненти MPV. Покрокова схема роботи мобільного додатку. Діаграма класів екрану «Файли». Діаграма класів в патерні «Спостерігач». Процес компіляції програми на мові Kotlin. Зв'язок між медіа сервером і Flash плеєром. Процес сигналінгу. Платформа OpenTok. Інструкція користувача. Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи хорони праці	Гурик О.Я., доцент кафедри МТ		

7. Дата видачі завдання 7 червня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	07.06.2021	*Виконано*
2.	Підбір джерел предметної області та проектування	08.06.2021-09.06.2021	*Виконано*
3.	Переклад та опрацювання джерел щодо розробки мобільного додатку для проведення презентацій.	10.06.2021-11.06.2021	*Виконано*
4.	Виконання дослідження щодо розробка мобільного додатку для проведення презентацій.	12.06.2021-13.06.2021	*Виконано*
5.	Оформлення розділу «Аналіз предметної області та проектування додатку проведення онлайн презентацій»	14.06.2021-15.06.2021	*Виконано*
6.	Оформлення розділу «Програмна реалізація тестування та длслідна експлуатація додатку для проведення презентацій.»	16.06.2021-17.06.2021	*Виконано*
7.	Виконання завдання до підрозділу «Безпека життєдіяльності»		*Виконано*
8.	Виконання завдання до підрозділу «Основи хорони	17.06.2021	*Виконано*
	праці»		
9.	Оформлення кваліфікаційної роботи	18.06.2021	*Виконано*
10.	Нормоконтроль	19.06.2021	*Виконано*
11.	Перевірка на плагіат	19.06.2021	*Виконано*
12.	Попередній захист кваліфікаційної роботи	19.06.2021	*Виконано*
13.	Захист кваліфікаційної роботи	23.06.2021	

Студент _____ Лесяк В.І.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Матійчук Л.П.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка мобільного додатку для проведення презентацій // Кваліфікаційна робота освітнього рівня «Бакалавр»// Лесяк Віталій Ігорович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СН-41// Тернопіль, 2021 // сторінок 59, рисунки – 36, таблиць – 9, джерел – 39.

Ключові слова: мобільний додаток, технологія WebRTC, розробка Androidдодатку, Java, Activity, Layout.

Робота присвячена впровадженню трансляцій в реальному часі в мобільний додаток для проведення презентацій. Це дозволить проводити дистанційне навчання, а також дозволить збільшити продуктивність навчання, тому що інформація краще сприймається, коли надходить з різних джерел.

У першому розділі наведено аналіз існуючих додатків для проведення онлайн презентацій, наводиться порівняння технологій для передачі потокового мультимедіа, а також розглянуто принцип роботи обраного рішення. Другий розділ присвячений розробці мобільного додатку, вибору архітектури, описано процедуру тестування додатку, створення екранних форм, відомості про робота сервера.

ANNOTATION

Mobile application development for presentation delivering // Qualification work of the educational level "Bachelor" // Lesyak Vitaliy Ihorovych // Ivan PuliyuTernopil National Technical University, Faculty of Computer Information System and Software Engineering, Department of Computer Science,group. CH-41 // Ternopil, 2021 // pages – 59, figures – 36, tables – 9, sources – 39.

Keywords: mobile application, WebRTC technology, Android application development, Java, Activity, Layout.

The work is devoted to the implementation of real-time broadcasts in the mobile application for presentations. This will allow for distance learning, as well as increase learning productivity, because information is better perceived when coming from different sources.

The first section provides an analysis of existing applications for online presentations, compares technologies for streaming multimedia, and considers the principle of operation of the selected solution. The second section is devoted to the development of a mobile application, the choice of architecture, describes the procedure for testing the application, creating screen forms, information about the server.

ЗМІСТ

ВСТУП

На поточний момент більшої популярності набирають проекти, пов'язані із застосуванням трансляцій в реальному часі. Область застосування трансляцій в реальному часі широка: консультації, відеоконференції, дзвінки, дистанційне навчання, інтернет речей, ігрова індустрія.

У зв'язку з цим велику популярність отримала технологія WebRTC, що дозволяє з мінімальними затримками передавати медіа дані.

Компанія Google вже давно представило свій додаток Hangouts для проведення відеоконференцій до 50 людей. Все більше компаній намагається впровадити в свої продукти функціональність, пов'язану із застосуванням трансляцій в реальному часі.

Метою даної роботи є впровадження трансляцій в реальному часі в мобільний додаток для проведення презентацій. Це дозволить проводити дистанційне навчання, а також дозволить збільшити продуктивність навчання, тому що інформація краще сприймається, коли надходить з різних джерел.

У першому розділі наведено аналіз існуючих додатків для проведення онлайн презентацій, наводиться порівняння технологій для передачі потокового мультимедіа, а також розглянуто принцип роботи обраного рішення. Другий розділ присвячений розробці мобільного додатку, вибору архітектури, описано процедуру тестування додатку, створення екранних форм, відомості про робота сервера.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПРОЕКТУВАННЯ ДОДАТКУ ПРОВЕДЕННЯ ОНЛАЙН ПРЕЗЕНТАЦІЙ

1.1. Опис та аналіз існуючих аналогів, що реалізують функції предметної області

Сучасні комунікатори (айфони, планшети, смартфони, КПК та ін.)надають своїм власникам різні функціональні можливості наприклад: робота в телефоних мережах з виходом в Інтернет; підтримка переносних носіїв даних; підтримка різнокольорових дисплеїв; потужні мобільні процесори; тривалість роботи без підзарядки; зручний форм-фактор. Цілком зрозуміло, що цих можливостей вистачає для повноцінної мобільної роботи в різних сферах праці – бізнесі, освіті та науці.Особливо перспективним і перспективним є використання мобільних пристроїв в напрямку дистанційного навчання, як мобільного навчання.

Проблема використання нових технічних засобів і дидактичних матеріалів у навчанні не є новою, питання використання стільникових мобільних пристроїв виникло ще у 80-х роках 20-го століття. Зрозвитком мобільних технологій зросла потреба і можливість швидкого доступу до інформації і особливо іншомовної, адже зросла мобільність населення і потужність мобільних пристроїв.

Тому особливої актуальності набуває пошук нових підходів до організації навчального процесу і створення навчальних матеріалів і технологій, які б враховували можливості мобільних пристроїв. За останні роки були різні думки, про те що саме є мобільним навчанням. Різноманітність думок частково пов’язано з швидким розвитком мобільного навчання.

Європейська гільдія з електронного навчання визначає його так: будь-яка діяльність, яка дозволяє людям бути більш продуктивними у споживанні, взаємодії або створенні інформації компактними цифровими

пристроями, якщо людина призводить ці дії на регулярній основі, має надійний зв’язок і пристрій поміщається в кишені або сумочці.

Отже, термін «мобільне навчання» (м-навчання), або mobile learning(m-learning), відноситься до використання у викладанні та навчанні мобільних і портативних ІТ-пристроїв, таких, як кишенькові комп’ютери PDA(Personal Digital Assistants), мобільні телефони, ноутбуки, нетбуки, планшетні ПК, iPhone, iPadта інше.

Мобільне навчання відноситься до використання у викладанні та навчанні мобільних та портативних інформаційно технологічних пристроїв, таких, як КПК, мобільні телефони, планшети, айфони,смартфони, ноутбуки, нетбуки та інше. Мобільний AssistedLearning. Мова (MALL) – технологія навчання мовам за допомогою портативних мобільних пристроїв, таких як мобільні телефони (стільникові телефони), MP3- і MP4-плеєри, КПК, Айфонабо Айпадта інше. MALL є підмножиною мобільного навчання і комп’ютерного навчання мовам (CALL).

Засоби мобільних інформаційно-комунікайних технологій навчання можна поділити на апаратні та програмні. Існують різні програмні мобільні засоби навчання такі як: MobileELDIT, MLEX, MLE-Moodle, AmadeusLMSMobile, LearnCast, Mobl21та ін. Сучасні апаратні пристрої на які встановлені різні операційні системи такі як:Android, BlackBerry, iPhone, WindowsCE, безкоштовно підтримують різні мовні функціональні можливості без додаткового ПЗ.

Під функціональними можливостями ми розуміємо різні сервіси, які пропонують операційній системі і самі пристрої своїм користувачам. У таких систем спільним є те, шо на їх платформи можна встановити безкоштовні додатки браузер GoogleChrome та програму GoogleTranslate.

Браузер GoogleChrome, встановлений на платформі ОС Android, має доступ до різних додатків на 30 мовах і в ньому присутні нові фунції, також користувач може запитати настільну версію web-сайту, якщо він

відмовляється переглядати мобільну версію сайту, також можемо додавати закладки що значно спрощує пошук улюблених сайтів.

Так як Інтернет-ресурси та комп'ютери стали невід'ємними освітніми інструментами і з'являються більш прості та ефективні пристрої, відкриваються ширші можливості для розширення сфер використання ІКТ. Варто відзначити, що мобільні пристрої коштують дешевше ніж настільні ПК і є більш дешевим засобом доступу до Інтернет ресурсів.

Тому особливої актуальності набуває пошук нових підходів до організації навчального процесу і створення навчальних матеріалів і технологій, які б враховували можливості мобільних пристроїв.

В якості розглянутих додатків були обрані два додатки з магазину додатків Google Play: Google Презентації, Presentain. Вибрані додатки мають високий рейтинг і велику кількість завантажень, а також надають можливість проведення онлайн презентацій.

Додаток Google Презентації. Google Презентації [1] - додаток розробляється компанією Google, доступний для таким платформ як iOS, Android і Web (див. рис. 1.1).

Дана програма надає найширшу функціональність серед конкурентів: створення і редагування презентації, спільна робота над презентацією, робота без інтернету, автоматичне збереження, показ презентацій з мобільного пристрою, планування автоматично програвання презентації.

Переваги:

- багаті можливості по створенню і зміні слайдів презентації;

- сумісність з файлами PowerPoint;

- показ слайдів на будь-яких екранах (підтримка Chromecast, Hangouts, Air Play);

Недоліки:

- немає відео і аудіо супроводу;

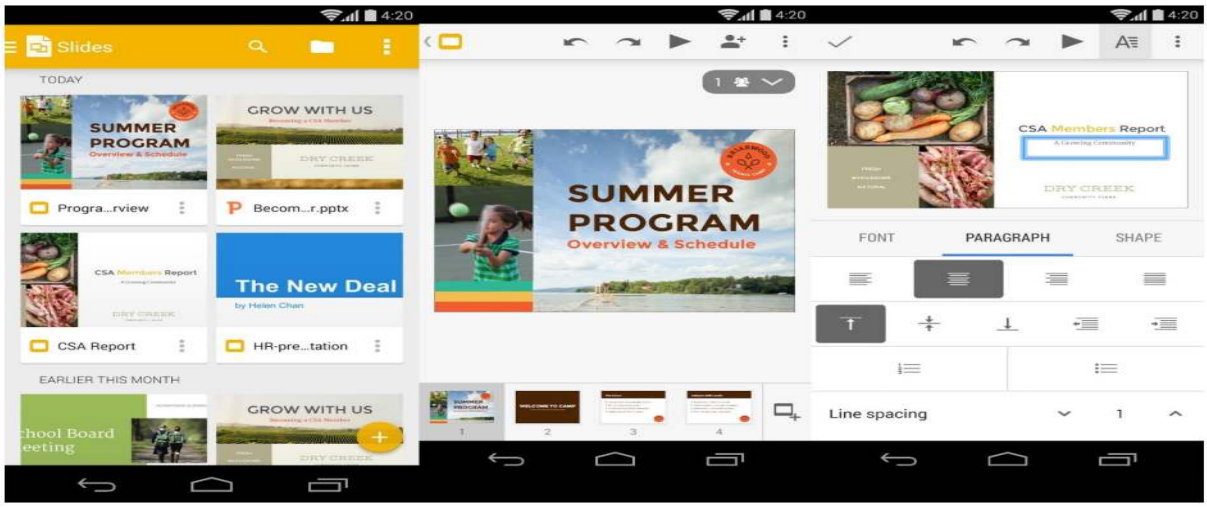

Рисунок 1.1 - Екранні форми додатка Google Презентації

Додаток Presentain. Presentain [2] - додаток, що дозволяє проводити інтерактивні презентації (див рис. 1.2). Відмінними рисами цього додатка є:

- можливість проведення опитувань під час презентації та отримання запитань від глядачів. Додаток є безкоштовним, але має різні платні тарифи, які дозволяють мати безлімітну кількість презентацій, глядачів, опитувань.

Переваги:- запис презентації (аудіо і слайди);- сумісність з файлами PowerPoint;Недоліки:- перегляд презентації з браузера;- немає відео супроводу;- є платний контент;- рідкісні поновлення;

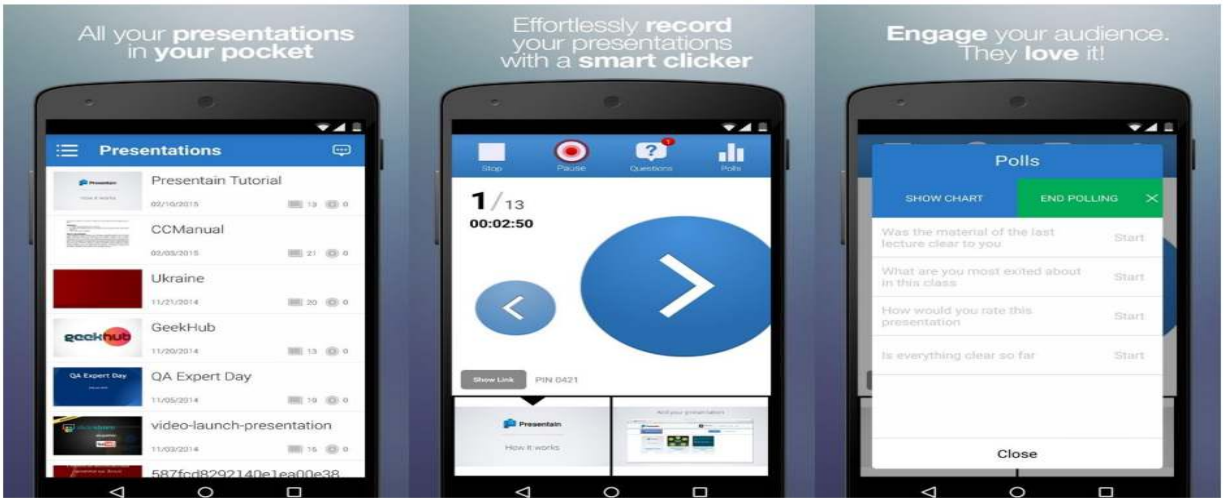

Рисунок 1.2 - Екранні форми додатка Presentain

Дослідивши аналоги, що існують на ринку мобільних додатків було помічено, що існуючі програми не надають можливість одночасної трансляції слайдів і відео з камери. Це функціональне вимога є відмінною рисою розроблюваного додатку. Додаток повинен використовуватися для проведення і перегляду онлайн презентацій, тому функціональність, пов’язана зі створенням презентації відсутня в цьому додатку, користувачі програми повинні заздалегідь експортувати свою презентацію в формат PDF для її використання в додатку. Відео трансляція повинна проводитися в реальному часі (затримка менше 1 секунди). У додатку буде можливість створення позначок на слайдах презентації у час її проведення і отримання доступу до матеріалів після проведення презентації.

1.2. Специфікація вимог до системи

Специфікація вимог до програмної системи – це специфікація окремого програмного продукту, програми або набору програм, які виконують деякі дії в деякому середовищі. Тобто – це повний опис поведінки системи що розробляється [3].

В загальному випадку специфікація включає наступне:

- глосарій проекту;
- опис варіантів використання.

Наведемо список основних термінів та понять в області розробки програмного додатку для проведення презентацій – глосарій. Глосарій – список понять в специфічній області знання з їх визначеннями [3]. Ці поняття та визначення подано у таблиці 1.1.

На рисунку 1.3 представлено діаграму прецендентів взаємодії користувача з програмою. Діаграма варіантів використання дозволяє швидко побачити основні функції, які буде виконувати програма після розробки.

Таблиця 1.1

Глосарій проекту

Термін	Опис терміну
1. Основні поняття та категорії предметної області та проекту	
ОС Android	операційна система для смартфонів, планшетних комп'ютерів, електронних книг, цифрових програвачів, наручних годинників, ігрових приставок, нетбуків, смартбуків
Мобільний інтернет	Сукупна назва бездротових технологій для доступу до мережі Інтернет.
Презентація	інформаційний чи рекламний інструмент, що дозволяє повідомити потрібну інформацію про об'єкт презентації в зручній для одержувача формі.
Бізнес-процес	будь-яка діяльність, що має вхідний продукт, додає вартість до нього, та забезпечує вихідний продукт для внутрішнього або зовнішнього споживача [3].
2. Користувачі системи	
Лектор	Людина, яка здійснює подання лекції, лекцію.
Студент	учень вищого, у деяких країнах і середнього навчального закладу.
3. Вхідні та вихідні документи	
Список презентацій	документ, який містить інформацію про доступні для перегляду презентації

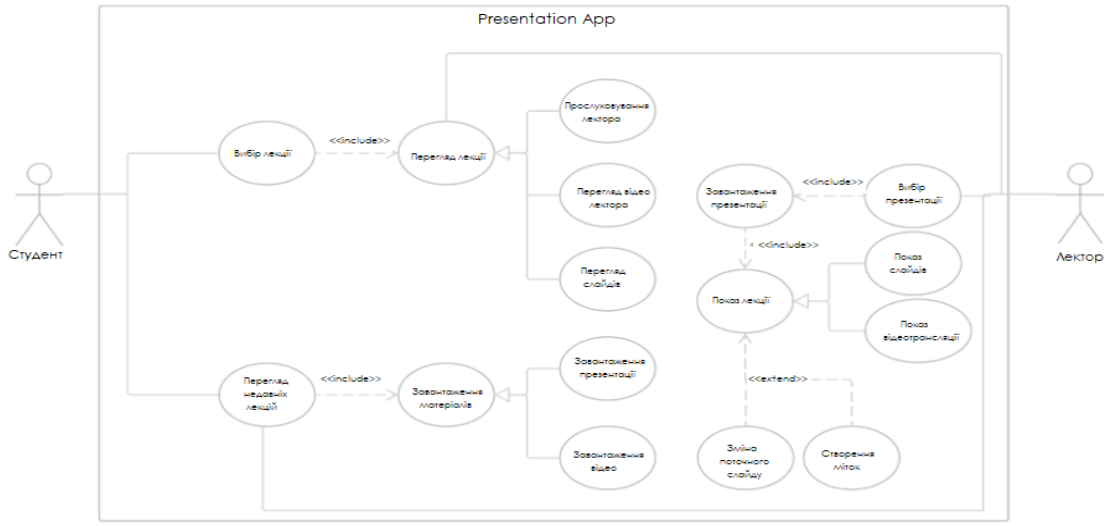

Рисунок 1.3 - Діаграма варіантів використання

На рисунку 1.4 представлено діаграму послідовності використання користувачем мобільного додатку.

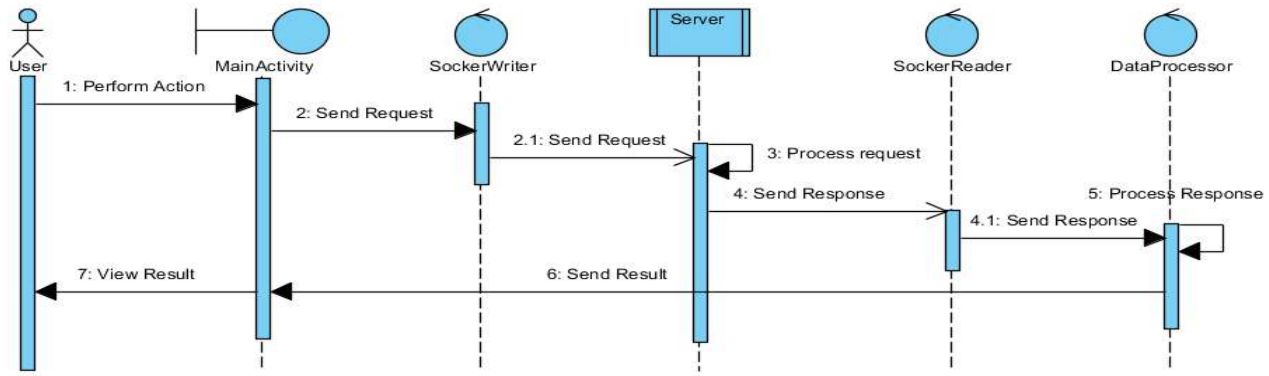

Рисунок 1.4 - Діаграма послідовності

Для створення розкадровки використаємо інструмент Balsamiq Mockups – зручний програмний засіб для створення ескізів екранних форм. На рисунках 1.5 – 1.6 наведено ескізи основних екранних форм відповідно до сценаріїв використання.

Рисунок 1.5 - Ескіз екранної форми авторизації користувача автомобілів

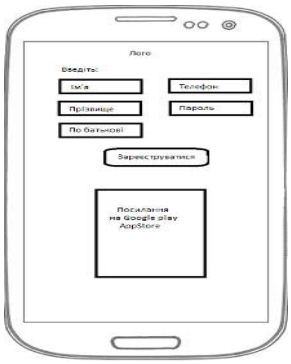

Рисунок 1.6 - Ескіз екранної форми реєстрації лектора

Специфікація функціональних і нефункціональних вимог наведено у таблиці 1.2, 1.3

Таблиця 1.2

Специфікація функціональних вимог

Ідентифікатор вимоги	Назва вимоги (варіанту використання)	Атрибути вимог		
		Пріоритет	Складність	Контакт
1.	реєстрація	Обов’язкова	Низька	Розробник
2.	вхід в систему	Обов’язкова	Низька	Розробник
3.	вибір лекції	Обов’язкова	Середня	Розробник
4.	перегляд лекцій	Обов’язкова	Середня	Розробник
5.	перегляд недавніх лекцій	Обов’язкова	Середня	Розробник
6.	прослуховування лектора	Обов’язкова	Середня	Розробник
7.	перегляд відео лектора	Обов’язкова	Середня	Розробник
8.	перегляд слайдів	Обов’язкова	Середня	Розробник
9.	завантаженняпрезентації	Обов’язкова	Середня	Розробник
10.	завантаження відео	Обов’язкова	Середня	Розробник
11.	вихід з програми	Обов’язкова	Середня	Розробник

Специфікація не функціональних вимог описана у таблиці 1.3.

Таблиця 1.3

Специфікація нефункціональних вимог

Ідентифікатор вимоги	Назва вимоги	Атрибути вимог		
		Пріоритет	Складність	Контакт
1. Застосовність				
1.1	Час, необхідний для навчання звичайних і досвідчених користувачів	Рекомендована	Низька	Користувач
1.2	Основні вимоги застосовності нової системи відносно інших систем, які знають користувачі	Рекомендована	Низька	Користувач

1.3	Вимоги по відповідності загальним стандартам застосовності та стандартам графічного інтерфейсу користувача	Обов’язкова	Низька	Користувач
2. Надійність				
2.1	Доступність	Обов’язкова	Середня	Користувач
2.2	Середній час безвідмовної роботи	Рекомендована	Середня	Користувач
3. Робочі характеристики				
3.1	Використання ресурсів	Рекомендована	Середня	Користувач

Значення нефункціональних вимог: час, необхідний для навчання звичайних користувачів – 30 хвилин; час, необхідний для навчання досвідчених користувачів – 15 хвилин; основні вимоги застосовності нової системи відносно інших систем, які знають користувачі – всі функції системи є легкими у виконанні, а структура програми не відрізняється від існуючих аналогів; вимоги по відповідності загальним стандартам застосовності та стандартам графічного інтерфейсу користувача – програма повинна працювати в операційній системі Android версії вище 5.0; доступність – час, що витрачається на обслуговування системи не повинен перевищувати 3% від загального часу роботи; середній час безвідмовної роботи – 3години; використання ресурсів – мінімальні системні вимоги: 256 Мб пам’яті; 10Мб вільного дискового простору; процесор з тактовою частотою 600МГц; Android 5.0.

1.3. Розроблення архітектури програмної системи

В даному розділі розглянемо основні принципи на яких будувалася архітектура додатку.

У сучасній розробці під Android виникла потреба використання патернів шару представлення. Використовувана API збільшується, збільшується кількість підтримуваного коду і вже буде неможливо використання підходів написання логіки додатка в компонентах, що відповідають за подання призначеного для користувача інтерфейсу. У співтоваристві розробників Android найбільшу популярність отримав патерн MVP через свою простоту реалізації і універсальності застосування. В основі цього патерну лежить принцип розділення відповідальності.

У паттерне MVP (див. рис. 1.7) можна виділити наступні компоненти і їх призначення:

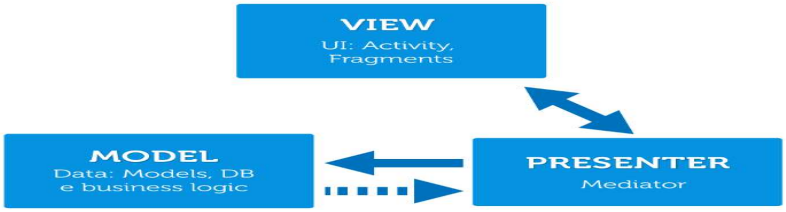

Рисунок 1.7 - Компоненти MVP

- модель - відповідає за дані які виводяться в призначеному для користувача інтерфейсі;

- представлення - інтерфейс через який виводяться дані моделі, а також відбувається передача подій презентеру;

- презентер - посередник між моделлю і представленням, в якому міститься логіка подання;

Android компонентами які повинні реалізовувати інтерфейс View є Activity, Fragment, View (клас з Android SDK). Presenter - компонент без залежностей на класи з Android SDK. Model зазвичай є реалізацією патерну «Сховище». На рисунку 1.8. представлена схема роботи додатку.

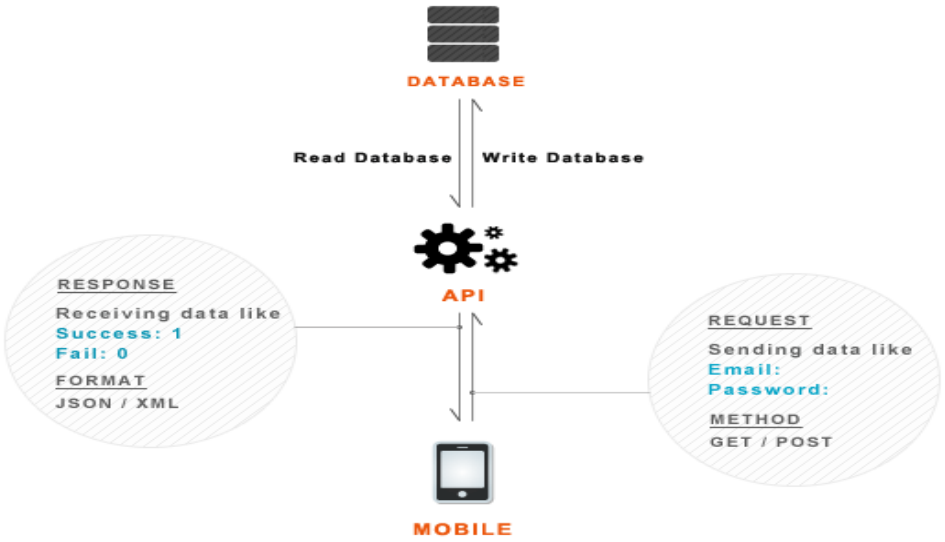

Рисунок 1.8. - Покрокова схема роботи мобільного додатку

Як приклад реалізації цього патерну візьмемо екран «Файли» з розроблюваного додатку. На цьому екрані лектор вибирає презентацію для показу її студентам. На рисунку 1.9. представлена діаграма класів цього екрану.

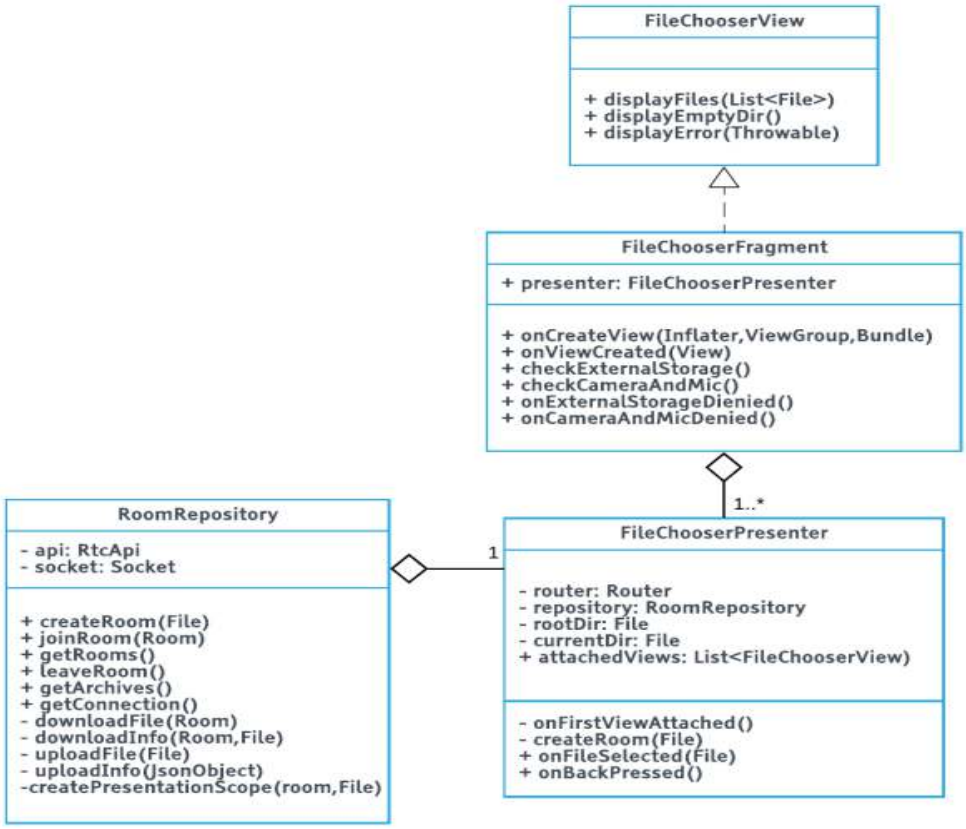

Рисунок 1.9- Діаграма класів екрану «Файли»

Як видно з діаграми класів відображений користувацький інтерфейс (FileChooserFragment) реалізує інтерфейс FileChooserView з наступними методами: показати файли в поточному каталозі, показати порожню директорію, показати помилку. FileChooserFragment має посилання на FileChooserPresenter, для виклик методів: натиснута кнопка «Назад» і файл обраний. FileChooserPresenter містить посилання на RoomRepository дозволяє виконувати операції з кімнатою, а також посилання на FileChooserView. Після виконання операції з кімнатою FileChooserPresenter може викликати методи на FileChooserView для відображення результату операції з кімнатою.

1.4. Реалізація асинхронності роботи додатку

Ще однією з сучасних проблем розробки під Android є забезпечення асинхронної роботи програми. Кожен Android розробник повинен знати, як застосувати багатопоточність в контексті ОС Android, так як за замовчуванням запущений додаток працює в однопоточном режимі.

У головному потоці відбуваються оновлення призначеного для користувача інтерфейсу, обробка користувацьких подій. Коли додаток виконує інтенсивну роботу на головному потоці не відбувається перемальовування екрану і після 5 секунд користувач побачить діалогове вікно «Додаток не відповідає».

Для виконання завдань у фоновому потоці існує багато рішень:- AsyncTask;- Loader;- Handler / Looper;- EventBus;- Java Threads;- RxJava;

Як інструмент для виконання завдань у фоновому потоці була використана бібліотека RxJava. Вона заснована на патерні «Спостерігач» (див. рис. 1.10).

Патерн «Спостерігач» визначає залежності між об’єктами так, що при зміні одного об’єкта (Subject), всі залежні від нього об’єкти (Observer) повідомляються про зміну. Зміни об’єкта, що спостерігається можна

представити у вигляді потоку значень. Для подання цього потоку використовується діаграма Marble. По осі X представлено час, по осі Y представлені потоки значень. Новий елемент в потоці представляється колом зі значенням всередині, вертикальна лінія означає кінець потоку. До цього потоку можна застосовувати оператори, які трансформують його.

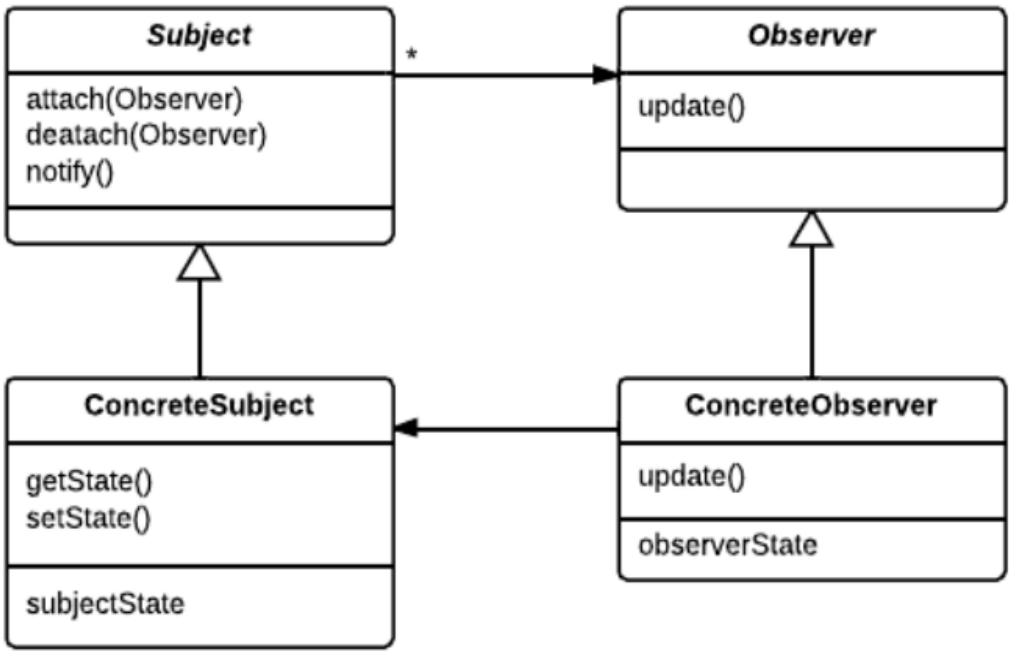

Рисунок 1.10 - Діаграма класів в патерні «Спостерігач»

На рисунку 1.11 представлена робота оператора map, який застосовує задану функцію (множення на 10) до кожного елементу потоку і виходить новий потік. Бібліотека RxJava представлена широким набором операторів, за допомогою яких можна комбінувати події, що виникають в різних потоках.

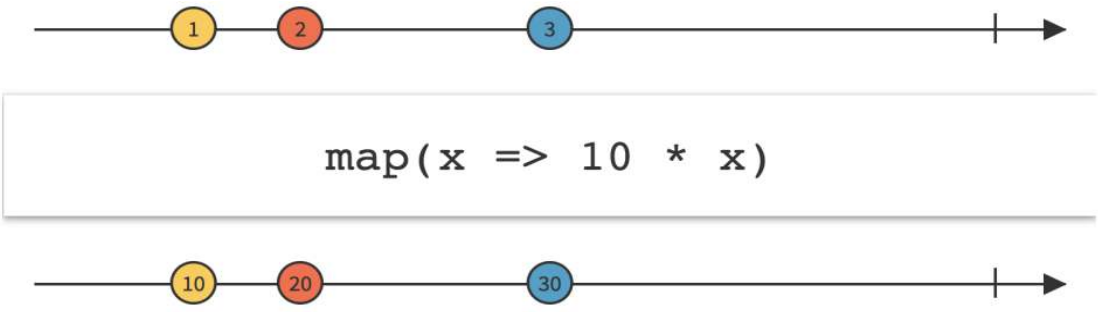

Рисунок 1.11 - Marble діаграма оператора map

До відмінних особливостей RxJava можна віднести: написання паралельного коду в послідовному стилі, широкі можливості по обробці помилок, зміна потоків виконання для кожної операції, комбінування запитів.

Інверсія залежностей ще один принцип на якому будувалася архітектура мобільного додатку. Він застосовується для спрощення зв’язків між модулями програми. Модулі верхніх рівнів не повинні залежати від модулів нижніх рівнів, модулі залежать від абстракції, абстракція не залежить від деталей, а деталі залежать від абстракцій. Це дозволяє підміняти інтерфейс реалізацією, що дуже важливо при тестуванні компонента в ізоляції.

В якості основної моделі даних на сервері виступає клас Room. Кімната - місце проведення лекції, що об’єднує користувачів. Користувачі представлені класом User. У таблиці 1.4 представлено опис класу User, а в таблиці 2.2 класу Room.

Таблиця 1.4

Поля класу User

Поле	Тип	Опис
userId	String	Ідентифікатор користувача
socketId	String	Ідентифікатор сокету користувача

Зв’язок між таблицями User та Room можна реалізувати через вкладеність документів по зв’язку userId та roomId з використанням технології представлення даних JSON.

Жорсткого типу зв’язку по первинних та зовнішніх ключах, які використовуються при реояційному описі моделі даних тут немає, оскільки в якості реалізації системи зберігання та опрацювання даних в додатку використовується MongoDB.

Таблиця 1.5

Поля класу Room

Поле	Тип	Опис
roomId	String	Ідентифікатор кімнати
roomName	String	Назва кімнати
presenter	User	Користувач, який створив кімнату
viewers	Array	Слухачі
currentSlide	Number	Поточний слайд
createdAt	Number	Час створення кімнати

Висновки до розділу 1

У цьому розділі здійснено опис предметної області, напрями діяльності. Визначено склад функцій, що входять до бізнес-процесу на основі яких розроблено схему управління бізнес-процесом. Проведено аналіз відомих програмних систем. Здійснено аналіз вимог до програмної системи.

Розроблено архітектуру мобільного додатку, що дозволить краще зрозуміти функції основних його частин. Створено та описано структурну схему, основними компонентами якої є: рівень клієнта, рівень бізнес-логіки та рівень даних. Описано функціональну структуру системи та її основних елементів – модулів обробки даних.

РОЗДІЛ 2

ПРОГРАМНА РЕАЛІЗАЦІЯ ТЕСТУВАННЯ ТА ДОСЛІДНА ЕКСПЛУАТАЦІЯ ДОДАТКУ ДЛЯ ПРОВЕДЕННЯ ПРЕЗЕНТАЦІЙ

2.1. Обгрунтування вибору засобів реалізації мобільного додатку

В даному розділі описуються обрані технології для розробки мобільного додатку і сервера, наводиться порівняння існуючих протоколів для потокової передачі відео, а також розглядається принцип роботи обраного протоколу.

В даний час розробка під Android ведеться на двох мовах програмування: Java і Kotlin. Більшість компанії переписують свої додатки на нову мову Kotlin, розроблену компанією JetBrains. Творці мови Kotlin також розробили середовище розробки IntelliJ IDEA на основі якої функціонує середовище розробки Android Studio. Тому підтримка в Android Studio мови Kotlin знаходиться на високому рівні.

Мова Java і Kotlin повністю сумісні (див. рис. 2.1). Для розробників, які тільки знайомляться з новою мовою в IDE присутня функція конвертації коду з мови Java в Kotlin. За твердженнями розробників мови [3], використання Kotlin зробить ваш код:

- лаконічним і виразним - багато конструкцій мови дозволяють зменшити написання шаблонного коду і збільшити читабельність коду;

- безпечним - в мову вбудована підтримка типів, які можуть містити порожні значення, а також конструкції, які дозволяють безпечно працювати з цими типами;

Розглянувши всі плюси мови Kotlin було прийнято рішення використовувати його для розробки мобільного додатку під Android.

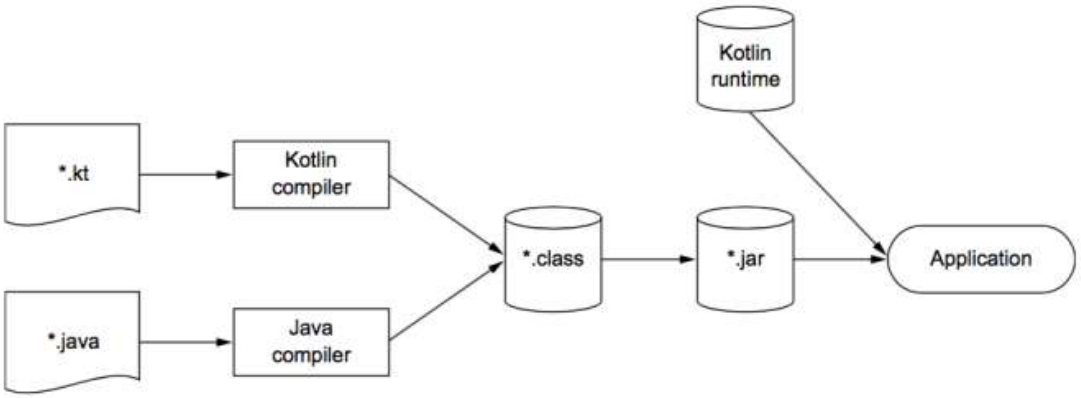

Рисунок 2.1 - Процес компіляції програми на мові Kotlin

На поточний момент існує величезна кількість мов програмування дозволяють написати веб-сервер: PHP, Ruby, Java, Python, Node. Основні функції, що вимагаються від сервера:

- зберігання інформації про минулі лекції в базі даних;

- збереження презентацій і зображень попереднього перегляду для кожної презентації на диску, а також надання API для проведення і перегляду презентацій.

Для реалізації цієї функціональності був обраний Node. Node – середовище виконання JavaScript, в основі якої лежить подієво-керована модель з неблокуючим введенням. Серед основних особливостей Node можна виділити: простоту освоєння, високу продуктивність, велика кількість вільно розповсюджуваних бібліотек.

В якості СУБД обрана MongoDB. MongoDB - NoSQL база даних, яка використовує для зберігання JSON документи. Використання MongoDB разом з Node збільшує продуктивність роботи, так як для тестування роботи бази даних застосовується синтаксис JavaScript.

Для роботи з зображеннями обрана консольна утиліта ImageMagick. Її основні особливості: підтримка великого числа графічних форматів, конвертація з одного графічного формату в інший, перетворення зображення, застосування спецефектів, малювання тексту, ліній, еліпсів.

HLS - протокол, розроблений компанією Apple для потокової передачі медіа по протоколу HTTP [4]. В основі роботи протоколу лежить розбиття медіа файлу на фрагменти (зазвичай тривалістю 10 секунд) і створення індексного файлу - файлу з розширенням m3u8. В індексному файлі міститься інформація про окремі фрагментах і часу їх програвання (див. рис. 2.2). Під час трансляції можлива зміна якості відтворення, для цього на сервері створюються фрагменти для кожного підтримуваного розширення і в підсумковому індексному файлі містяться посилання на індексні файли з заданими дозволами. За протоколом HLS можна роздавати як відео за запитом, так і трансляції в реальному часі. Мінусом HLS протоколу є висока затримка для проведення трансляцій в реальному часі, HLS був розроблений для збільшення якості трансляцій, а не для зменшення затримок.

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-TARGETDURATION:18
#EXT-X-MEDIA-SEQUENCE:0
#EXTINF:10.416667,
index0.ts
#EXTINF:12.625000,
index1.ts
#EXTINF:10.416667,
index2.ts
#EXTINF:10.416667,
index3.ts
#EXTINF:12.208333,
index4.ts
#EXTINF:4.583333,
index5.ts
#EXTINF:10.625000,
```

Рисунок 2.2 - Приклад m3u8 файла

MPEG-DASH - стандарт для адаптивного потокового мовлення, переданого по протоколу HTTP [5]. Як і в протоколі HLS в DASH медіа файл розбивається на сегменти з різним розширенням і бітрейтом. На відміну від HLS DASH не вимагає використання певного кодека, відео може бути закодовано з використанням кодеків VP9, H264 або інших. Стандартна тривалість сегмента 2-4 секунди. Файлом, в якому міститься інформація про сегменти є MPD файл (див. рис. 2.3).

```
<MPD xmlns="urn:mpeg:DASH:schema:MPD:2011" xmlns:ytdrm="http://youtube.com/ytdrm"
mediaPresentationDuration="PT0H3M1.63S" minBufferTime="PT1.5S"
profiles="urn:mpeg:dash:profile:isoff-on-demand:2011"
type="static">
  <Period duration="PT0H3M1.63S" start="PT0S">
    <AdaptationSet>
      <ContentComponent contentType="video" id="1" />
      <ContentProtection schemeIdUri="com.youtube.clearkey">
        <ytdrm:License keyid="60061e017e477e877e57d00d1ed00d1e"
key="1a8a2095e4deb2d29ec816ac7bae2082"/>
      </ContentProtection>
      <Representation bandwidth="4190760" codecs="avc1.640028" height="1080" id="1"
mimeType="video/mp4" width="1920">
        <BaseURL>car_cenc-20120827-89.mp4</BaseURL>
        <SegmentBase indexRange="2755-3230">
          <Initialization range="0-2754" />
        </SegmentBase>
      </Representation>
      <Representation bandwidth="2073921" codecs="avc1.4d401f" height="720" id="2"
mimeType="video/mp4" width="1280">
        <BaseURL>car_cenc-20120827-88.mp4</BaseURL>
        <SegmentBase indexRange="2789-3264">
          <Initialization range="0-2788" />
        </SegmentBase>
      </Representation>
      <Representation bandwidth="869460" codecs="avc1.4d401e" height="480" id="3"
mimeType="video/mp4" width="854">
        <BaseURL>car_cenc-20120827-87.mp4</BaseURL>
        <SegmentBase indexRange="2789-3264">
          <Initialization range="0-2788" />
```

Рисунок 2.3 - Приклад MPD файла

RTMP - протокол, розроблений компанією Adobe для передачі потокового відео. RTMP підтримує постійне з'єднання між медіа сервером і відео плеєром, через це проведення трансляції здійснюється з низькими затримками. Для роботи необхідний Adobe Media Server, з яким Flash Player відкриває з'єднання по протоколу RTMP (рисунок 2.4).

WebRTC - технологія розроблена компанією Google, що дозволяє з'єднати двох клієнтів для передачі потокового відео [6]. Для з'єднання клієнтів не потрібна установка додаткового ПЗ. Медіа дані передаються безпосередньо між клієнтами, після установки з'єднання всі дані шифруються. Якість відео трансляції залежить від пропускної здатності.

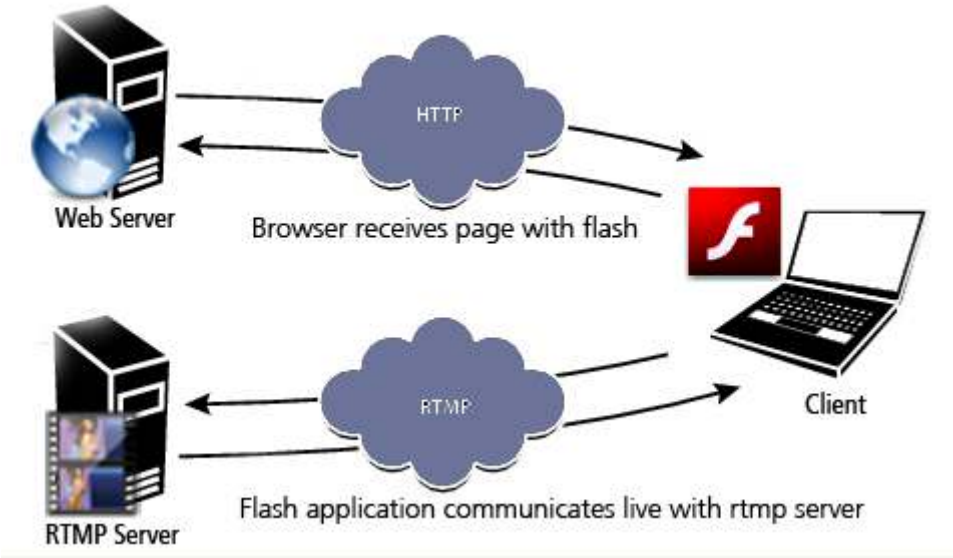

Рисунок 2.4 - Зв’язок між медіа сервером і Flash плеєром

Мінусом WebRTC є важке масштабування, так як веб-сервер відповідає тільки за передачу метаінформації перед встановленням з’єднання P2P.

Протоколи HLS і MPEG-DASH підходять для проведення трансляцій в яких якість важливіше затримки. Для розроблюваного додатку потрібно відтворювати відео в реальному часі для виконання цієї вимоги підходять RTMP і WebRTC.

Причини, за якими була обрана технологія WebRTC:

- не потрібна установка додаткового ПЗ на сервер і клієнтів;

- Adobe припинить підтримку Flash Player до 2020 року, використання якого є обов’язковим для перегляду відео трансляцій.

MediaStream - представляє певне джерело медіа з одним або декількома медіа доріжками.

PeerConnection - API для встановлення аудіо, відео дзвінків. PeerConnection [7] відповідає за наступні функції:

- з’єднання P2P;

- годування, декодування медіа;

- шифрування;

- спостереження за пропускною спроможністю;

Для встановлення сесії клієнту необхідно мати:

- локальну SDP (описує медіа настройки клієнта);

- вилучену SDP (описує медіа настройки опонента);

- видалених ICE кандидатів (описують як підключитися до опонента);

Є два типи користувачів, хто телефонує і очікує дзвінка. Користувач першим відправив SDP є дзвоном. SDP відправлена дзвоном називається Offer, відправлена у відповідь SDP називається Answer.

Для встановлення з'єднання той, хто телефонує створює MediaStream (наприклад, з камери телефону), PeerConnection і додає MediaStream в PeerConnection. Далі той, хто телефонує створює і посилає Offer очікує дзвінка, зберігаючи його як локальну SDP. Якщо очікувач дзвінка приймає Offer, то він зберігає цей Offer як віддалену SDP, створює Answer, зберігає його як локальну SDP і посилає його тому, хто дзвонить. Після отримання Answer той, хто телефонує зберігає його як віддалену SDP. Починаючи з цього моменту у обох учасників генеруються ICE кандидати, обидва учасники повинні переслати один одному своїх кандидатів і зберегти чужі. Після цього процес сигналінгу закінчений і очікувач дзвінка отримує відео від абонента, яке передається P2P (див. рис. 2.5).

Розглянутий варіант працює якщо обидва учасники мають відкриті IP адреса, якщо один з учасників знаходиться за NAT, то необхідно використовувати STUN сервер. Запит на STUN сервер створює запис в таблиці NAT для вузла, який знаходиться за NAT. STUN сервер повертає підмінений роутером IP адреса по якому доступний вузол. Для цього вузла створюється додатковий ICE кандидат. При використанні симетричного NAT в таблиці NAT додаються ще 2 поля, що відповідають за зовнішній IP адреса і порт з якого пакет може бути прийнятий, тому використання STUN сервера не допоможе. Для вирішення цієї проблеми використовують TURN сервер, який є посередником між двома вузлами, всі дані передаються через нього, а не P2P.

WebRTC не має протоколу для передачі інформації про сесію, тому для реалізації передачі можуть використовуватися як протокол Websocket так і HTTP або SMTP.

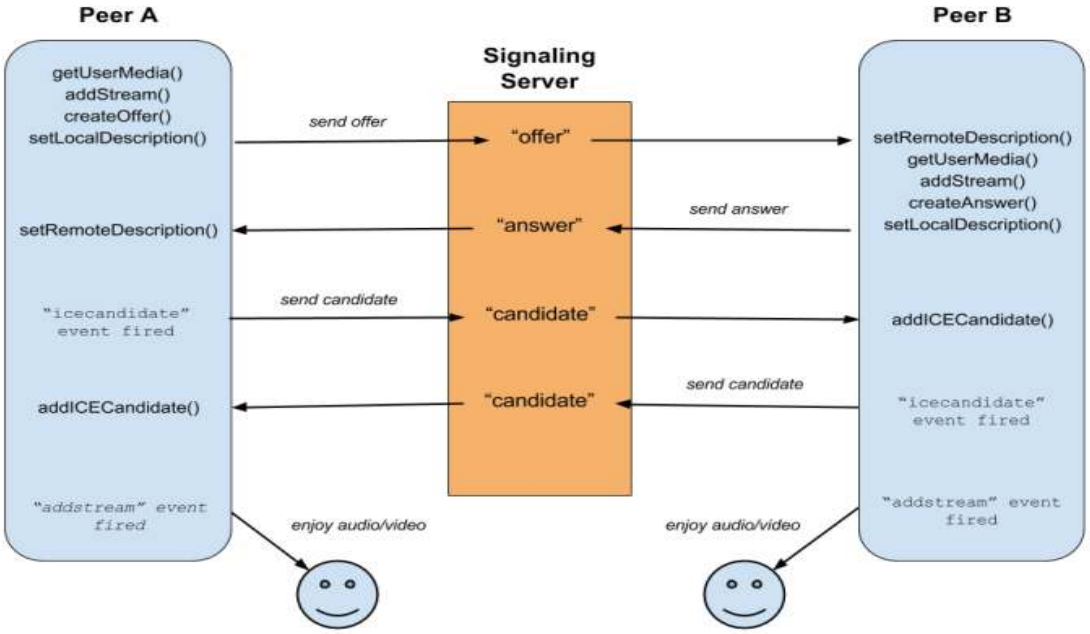

Рисунок 2.5 - Процес сигналінгу

2.2. Програмна реалізація серверної частини додатку

Для потокової передачі відео від лектора до студентів необхідно реалізувати механізм сигналінгу WebRTC, а також надати STUN і TURN сервера. Платформи надають відкрите API побудоване на технологіях WebRTC, що дозволяє спростити процес розробки, однією з таких платформ є OpenTok. OpenTok надає свої STUN і TURN сервера, а також SDK для різних мов програмування.

OpenTok є платною платформою (вартість тарифу для невеликих команд 10$ в місяць), проте для етапу розробки вистачить безкоштовного пробного періоду. Зареєструвавшись на сайті OpenTok [8], ми потрапляємо в панель управління своїм акаунтом (див. рис. 2.6). Вибравши пункт меню Projects створимо новий проект, після створення проекту нас перенаправить на сторінку проекту, на якій нам необхідно отримати 2 ключа PROJECT API

KEY і PROJECT SECRET (див. рис. 2.7), які стануть в нагоді при використанні OpenTok SDK як на сервері, так і в мобільному додатку.

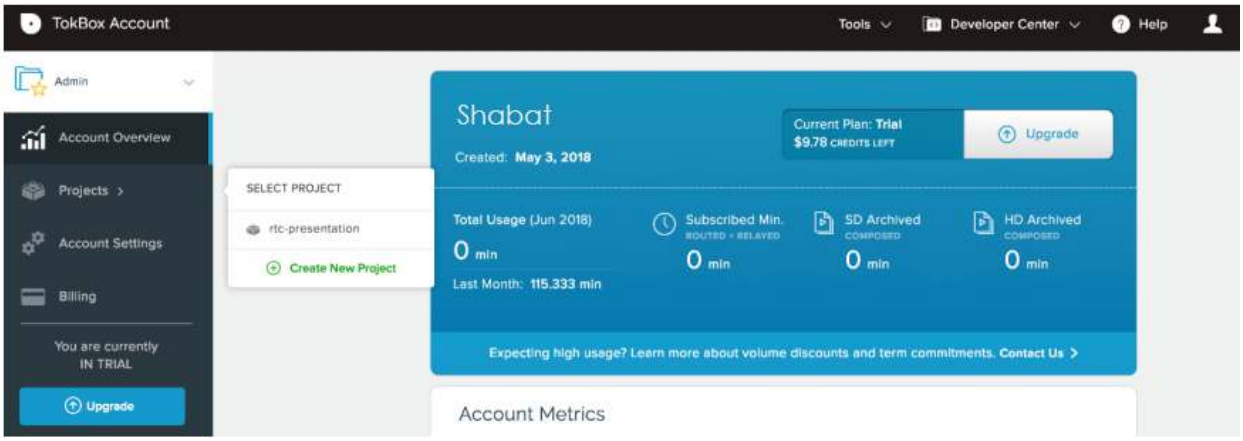

Рисунок 2.6 - Платформа OpenTok

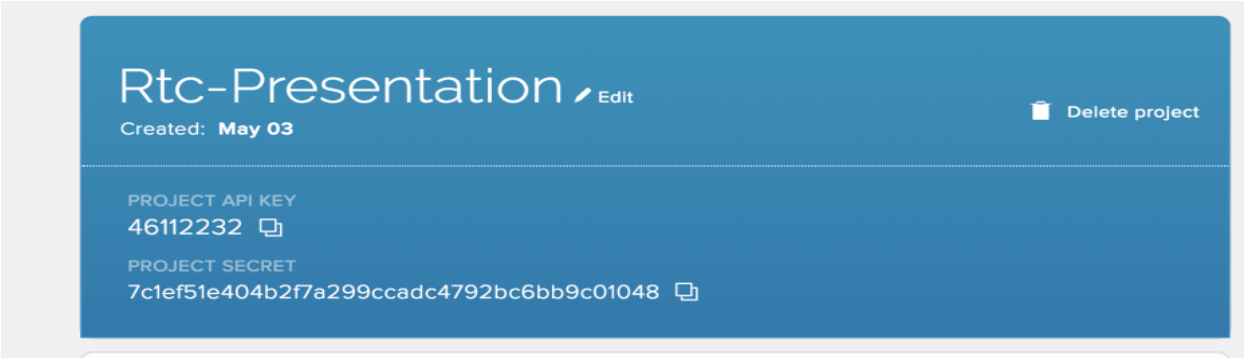

Рисунок 2.7 - Інформація про проект

На рисунку 2.8. представлена архітектура додатків, побудованих на платформі OpenTok.

- Session - «кімната» в якій користувачі можуть взаємодіяти в реальному часі;

- Token - унікальний ключ авторизації, що дозволяє клієнту підключитися до сесії. Токену можна призначити ролі: publisher, subscriber, moderator які визначають дозволу;

- Client - веб або мобільний додаток, що використовує OpenTok SDK, відповідальний за підключення до сесії, публікацію і підписку на відео потік, відправку подію в сесію;

- Server - відповідальний за створення сесії і токенів, які відправляються клієнту;

- OpenTok Cloud - управляє сесіями користувачів, займається виконанням запитів до API;

На рисунку 2.8 представлені основні дії необхідні для проведення трансляції. Розроблений сервер створює сесію посилаючи запит на сервера OpenTok, а також маркери для кожного нового клієнта.

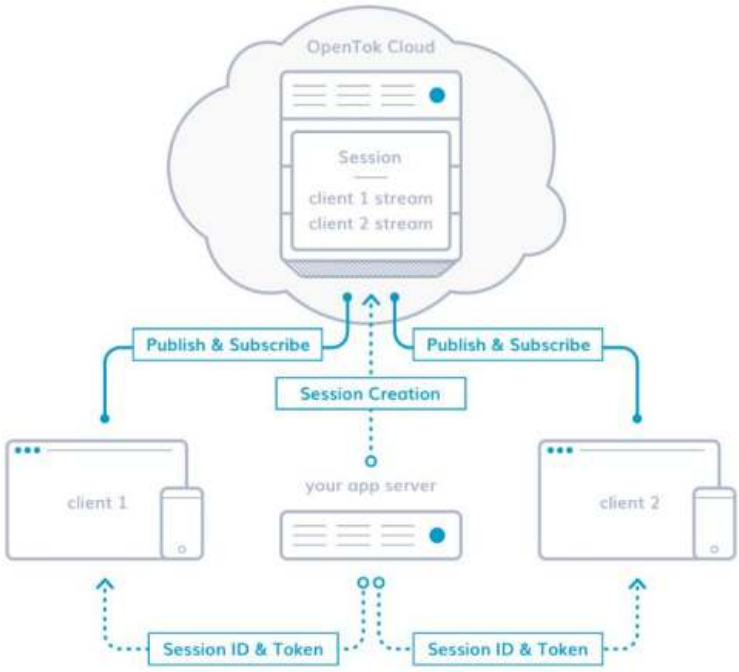

Рисунок 2.8 - Архітектура додатку, побудованого на платформі OpenTok

Клієнт, який бажає провести трансляцію публікує свій потік на сервера OpenTok, а глядачі підписуються на цей потік. Для створення сесії на сервері необхідно ініціалізувати об'єкт OpenTok за допомогою ключів, отриманих в попередньому розділі. Серверне API надається платформою OpenTok і дозволяє автоматично записувати трансляції, займатися адмініструванням поточних сесії.

Для надання доступу до файлів на сервері, а також архівів проведених презентацій було розроблено API з використанням фреймворку Express. Для створення кінцевих точок в Express використовується маршрутизація.

Кожен маршрут характеризується унікальним URI і HTTP методом. До кожного маршруту можна додати функцію зворотного виклику, яка спрацює, коли користувач пошле HTTP запит по заданому URI на хост де запушено наш додаток. У функції зворотного виклику можна отримати доступ до об’єкта запиту і сформувати відповідь. Для створення відповідей використовуються представлення, які описано в таблиці 2.1. У таблиці 4 представлені всі кінцеві точки, які описані на сервері.

Таблиця 2.1

Методи для формування відповіді сервера

Метод	Опис
res.sendFile()	Відправлення файлу
res.sendStatus()	Установка коду стану відповіді і відправка його представлення у вигляді рядка
res.json()	Відправлення відповіді в форматі JSON

Таблиця 2.2

Кінцеві маршрути сервера

URI	Метод	Параметри	Призначення
/upload	POST	filename – презентація для завантаження	Завантаження презентації на сервер
/download/:filename	GET	filename – назва файлу на сервері	Отримання файлу на сервері
/archives	GET	-	Отримання інформації по переглянутих лекціях

Як приклад маршруту розглянемо завантаження презентації на сервер. Як видно з представленого коду в додаток був доданий маршрут по URI / upload і HTTP методу POST. Усередині функції зворотного виклику відбувається розбір відправленого клієнтом файлу, створення зображення

попереднього перегляду 1 слайда. В результаті успішної обробки запиту створюється відповідь, в тілі якого знаходиться JSON об’єкт з посиланнями на завантажені файли. В результаті невдалої обробки у відповіді сервера поле Status-Code буде встановлено в 500.

Socket.IO - бібліотека для обміну даними в реальному часі. В основному застосовується як обгортка на WebSocket, розширюючи API WebSocket в якому всього 4 події (Open, Message, Close, Error). Користувач може відправити сокету створену подію разом з даними, а сервер може прослуховувати цю подію. Також Socket.IO надає API, за допомогою якого можна відправляти події всім підключеним до сервера сокетам або тільки сокетам, що знаходяться в одній кімнаті.

Для додання додатком інтерактивності були розроблені події, представлені в таблиці 2.3

Таблиця 2.3

Події опрацьовувані сервером

Події	Праметри	Призначення
createRoom	roomName - назва кімнати, attachment -url презентації на сервері, preview - url превью зображення на сервері	Створення кімнати
joinRoom	roomId – ідентифікатор кімнати	Вхід в кімнату
getRooms	-	Отримання списку активних кімнат
leftRoom	-	Покинути комнату
updateSlide	slide – номер слайду	Оновити поточний номер слайда в кімнаті
forceDisconnect	-	Сигнал на вихід всім користувачам з кімнати

2.3. Програмна реалізація бази даних

У MongoDB база даних складається з колекцій. Як правило, колекція відображає сутність предметної області. Колекція складається з

"документів" (далі запис). Запис є JSON-об’єкт з обов’язковою наявністю ідентифікатора ObjectId, який MongoDB задає автоматично і контролює його унікальність, аж до унікальності в усьому світі [2]. ObjectId є типом даних. У порівнянні з реляційними СУБД, колекція є таблицею, а запис є рядком в таблиці.

Тип збережених в запису даних - довільний, з перерахованих нижче:

- String;
- Array (структура даних з доступом по цілочисельному індексу);
- Boolean;
- Date;
- Integer;
- Null.
- Object (структура даних, що надає доступ до елементів по ключу).

Цілісність зв'язків в MongoDB не підтримується. Це завдання вирішується на рівні коду сервера додатків, що і було реалізовано за допомогою інструменту ODM, - Mongoose.

Для того щоб мати представлення відношення між сутностями, була розроблена схема БД, на якій відображені псевдо-зв’язки, рисунок 2.8, нотація Гордона Евересту [21].

Доповнимо до схеми, що найменування колекцій і псевдо-зв'язки можуть представляти дещо інший сенс, ніж таблиці в реляційних БД, так як існують вкладені об’єкти. Прикладом є колекція з найменуванням product_type.

Типи псевдо-відношень документа з іншими сутностями, враховуючи не структурованість даних:

- HasOne: має один тип документа.
- HasOne: має одну мову.
- ManyToMany: належить до одного або множини продуктів (BelongsTo), так як є загальним. З іншого боку один товар має множину документів на різних мовах (HasMany).

● HasMany: має багато змін.

● BelongsTo: належить до одного скачування, відправлення по email або замовлення.

Для опису особливостей проектування схем в MongoDB, були використані такі поняття:

● Вкладення - повне утримання об’єкта (ів) однієї сутності в іншій.

● Посилання - унікальний ідентифікатор ObjectId зовнішньої сутності.

Використання інструменту Mongoose дозволяє автоматично генерувати ObjectId при вставці вкладеного об'єкта. Наприклад, при додаванні продукту в тип продукту. При проектуванні були визначені деякі особливості проектування схеми БД в MongoDB, які полягають в тому, що:

● При складових псевдо-ключах різко втрачається гнучкість роботи з даними.

● Зберігання тільки ідентифікатора об’єкта зовнішньої сутності зажадає виконати додатковий запит для отримання всіх даних зовнішньої сутності.

● Вкладення об’єкта сутності має сенс тоді, коли вкладений об'єкт буде зустрічатися разом з батьківським.

● Вкладення об'єкта сутності відображає в реляційній схемі зв’язок один до багатьох.

● Посилання забезпечують більшу гнучкість при частих змінах вкладеної сутності.

● Масив посилань об'єктів сутності відображає в реляційній схемі зв'язок багато до багатьох.

Розглянемо приклади основних запитів до спроектованої моделі даних, яка використовується розробленим мобільним додатком.

В даних запитах використовуються наступні оператори MongoDB [9]:

- $ne - вибирає документи, в яких значення поля не дорівнює зазначеному;

- $or - виконує операцію логічного АБО над масивом виразів і вибирає документи, що задовольняють хоча б одній умові;

- $push - додає значення в зазначений масив;

- $elemMatch - вибирає документи, що містить елемент масиву, який відповідає всім зазначеним критеріям запиту;

2.4. Тестування додатку

У процесі розробки програми проводилося поетапне тестування з метою виявлення програмних помилок і невідповідностей ТЗ (з технічним завданням). Для цього нами були створені емулятори смартфона і планшета з різними діагоналями екрана для різних версій Android.

Тестований програмний продукт послідовно запускався на цих емуляторах, його поведінка аналізувалася, і при необхідності по результатах аналізу вносилися зміни в код [12].

Для тестування окремих модулів роботи з базою даних в текст програми були внесені спеціальними функціями, які аналізувати базу даних і, при підозрі на помилку, виводилися повідомлення в системний журнал. Вони також відомі як юніт-тести. Наприклад, при змінах в базі даних проводилася перевірка цілісності бази даних (перевірка на відповідність ключів - індексам), після чого при необхідності виводилося повідомлення в системний log.

Були проведені наведені нижче тести.

1. Кожна активність була піддана юніт-тестування з метою виявлення помилок, викликаних невідповідністю очікуваних і отриманих параметрів. Для цього для кожної активності був створений спеціальний юніт-клас, який посилає в активність різні вірні і неправильні запити. При аномальній поведінці активності або її збої, мною аналізувалася поведінка і помилка виправлялася.

2. У базу даних навмисно вносилися неприпустимі дані в відповідні поля, які могли бути невірно інтерпретовані програмою. Потім мною аналізувалася поведінка активності у час обробки неприпустимих даних.

3. Додаток було запущено на пристроях, що працюють під керуванням різних версій Android з метою виявлення особливостей роботи програми, запущеного в різних операційних системах.

4. Після завершення циклу розробки, програмний продукт тестувався на реальних пристроях. За результатами тестування була додана віртуальна кнопка «Меню» для пристроїв, які не мають апаратних кнопок.

Тестування є важливою частиною розробки програмного забезпечення. Воно дозволяє виявити помилка та дефекти в роботі програм. Існує безліч видів тестувань, кожен з яких дозволяє перевірити на працездатність програму з тієї чи іншої сторони[14].

Враховуючи специфіку поставленого завдання, для того щоб максимально точно перевірити функціонування розробленого завдання в різних умовах було вирішено використати наступні види тестування:

- Модульне тестування.
- Функціональне.
- Тестування продуктивності.

Модульне тестування. В першу чергу було проведено модульне тестування. Воно дозволяє протестувати найменші частини додатку, які виконують якусь конкретну функцію. Модульне тестування для кожного модуля проводиться в ізоляції ввід інших частин системи [15].

Для виконання модульного тестування було обрано Java JUnit. JUnit - бібліотека для модульного тестування програмного забезпечення на мові Java. JUnit - простий і в той же час дуже потужний інструмент для написання unit тестів. У даному розділі ми будемо використовувати останню на даний момент версію JUnit, а саме 4.1.0.

Спочатку було протестовано конструктор класу Main_activity. Це проводиться для того, щоб визначити чи встановлюються іконки обраної

категорії та чи створюється об’єкт, при вказанні великих розмірів цієї ж іконки.

```
    @Test
    public void testDrawConstructorS() {
   DrawablePlaceIcon d1 = new DrawablePlaceIcon (“sto”, “TestObject”, 1000, 50, 500, getResources(), 100);
    DrawablePlaceIcon d2 = new DrawablePlaceIcon (“sto”, “TestObject2”, 1000, 50, 500, getResources(), 500);
    DrawablePlaceIcon d3 = new DrawablePlaceIcon (“sto”, “TestObject2”, 1000, 50, 500, getResources(), 1000);
                assertNotNull(d1);
                assertNotNull(d2);
                assertNotNull(d3);

    }
```

Виконання данного тесту показало різні результати на різних пристроях. На пристроях, з малим об'ємом оперативної пам'яті він був провалений, оскільки не вистарчало пам'яті для відмалювання піктограми. Але на пристроях, де її об'єм був більший 256 Мб, тест виконався успішно.

Наступним тестом буде перевірка конструктора на приймання невалідної інформації, такої як невірний формат картинки, великий розмір файлу та пуста назва об’єкту.

```
@Test(expected = Exception.class)
    public void testDrawConstructorEr() {
        DrawablePlaceIcon d1 = new DrawablePlaceIcon (“sto”, “TestObject”, 1000, 50, 500, getResources(), 100);
       DrawablePlaceIcon d2 = new DrawablePlaceIcon (“”, “TestObject2”, 1000, 50, 500, getResources(), 500);
    DrawablePlaceIcon d3 = new DrawablePlaceIcon (“sto”, “TestObject2”, -50, 50, 500, getResources(), 1000);
    DrawablePlaceIcon d4 = new DrawablePlaceIcon (“sto”, “TestObject2”, 1000, 50, 500, getResources(), -100);
    }
```

Цей тест був провалений, оскільки в класі не було проведено необхідних перевірок на коректність параметрів, і видавалися необроблені виключні ситуації. Для вирішення цієї проблеми, у разі отримання некоректних параметрів, повертається пустий об’єкт.

Також необхідно провести тест методів класу Lection, щоб дізнатися, чи не повертає він невалідної інформації у разі передачі йому різної якості параметрів.

Для цього також був створений тестовий метод, у якому по черзі були передані некоректні параметри. Код методу представлений нижче.

```
@Test(expected = Exception.class)
    public void testDrawConstructorEr() {
                if(position <-1)
                        fail("Negative value");
                int position2 = Stos.getOnScreenLocation(null, new Lection(500 Mb);
                if(position <-1)
                        fail("Negative value");
```

Цей тест був виконаний успішно, оскільки в класі передбачена можливість вводу некоректних параметрів. В такому разі метод повертає результат -1, який і використаний для перевірки в даному тесті.

Функціональне тестування. Наступним було проведено функціональне тестування. Функціональне тестування дозволяє перевірити чи відповідають функції розробленого ПЗ вимогам, що були зазначені в специфікації вимог [17]. Для цього було створено тестові випадки для кожного з варіантів використання.

За результатами функціонального тестування 3 з 3-х тестових випадків пройшли успішно, отже – тестування можна вважати успішним – 100% тестових випадків пройшли. Це означає, що програма повністю реалізовує всі функції, що зазначені в вимогах до програмного продукту.

Тестування продуктивності. Тестування продуктивності проводиться з метою визначення, як швидко працює програма або її частина під деяким навантаженням та в певних умовах. Тестування продуктивності намагається враховувати продуктивність на стадії.

Важливим показником для розроблюваного додатку є показник використання інтернет трафіку. Адже, чим більше трафіку використає додаток – тим більше потрібно буде заплатити за нього, і тим більше часу буде очікувати користувач на отримання відповіді. Для цього проаналізуємо інтернет з’єднання пристрою під час виконання додатку. Для цього використаємо інструмент «NetworkStatisticTool», що входить до набору

інструментів DDMS (DalvikDebugMonitor Service), що входять в пакет ADT (AndroidDevelopmentTools) для Eclipse [19]. Даний інструмент відображає обсяги обміну даних через мережу Інтернет в кожен момент часу. «NetworkStatisticTool» був запущений під час виконання розроблюваного додатка протягом 30 секунд, результати його виконання зображені на рисунку 2.9.

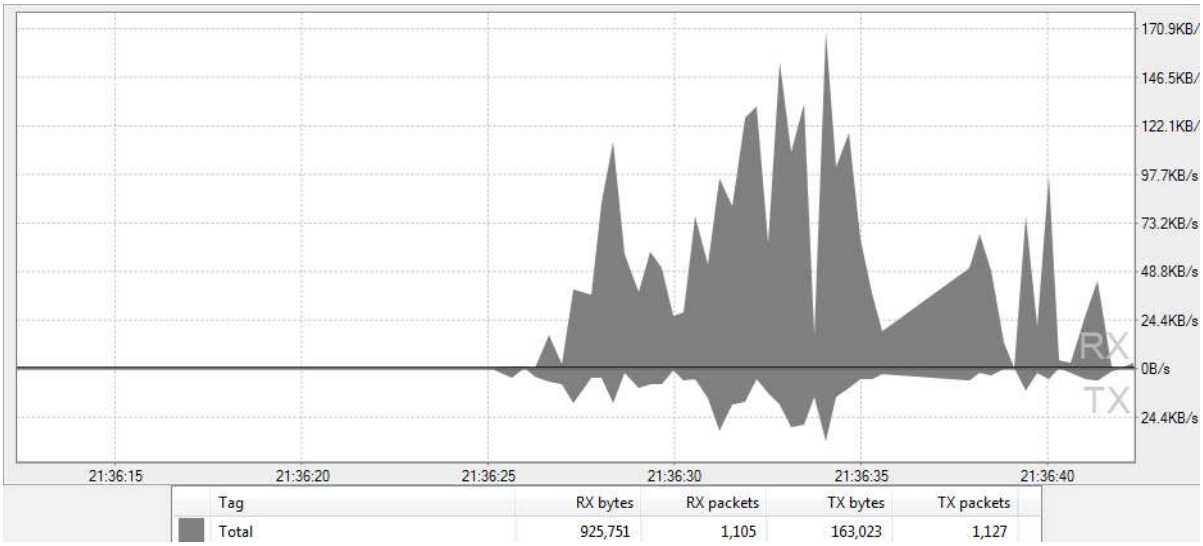

Рисунок 2.9 - Статистика використання інтернет з’єднання

Як бачимо, максимальна кількість використаного трафіка досягає 170 КБ/с. Такий результат тест показує під час отримання інформації про всі об’єкти однієї категорії. Проте в період, коли додаток отримує інформацію про конкретну презентацію, обсяг отриманих даних становить менше 5 кб/с. Враховуючи те, що на даний момент, середня швидкість мобільного інтернет з’єднання дорівнює приблизно 250-300 КБ/с, для даного додатку цього повністю хватає, і не завдаватиме проблем в його роботі.

Android Studio будує проект автоматично в процесі його зміни, а не по команді. Під час побудови інструментарій Android бере ваші ресурси, код і файл AndroidManifest.xml (що містить метадані додатку) і перетворює їх в файл .apk. Отриманий файл підписується налагоджувальний ключем, що дозволяє запускати його в емуляторі (див. рис. 2.10).

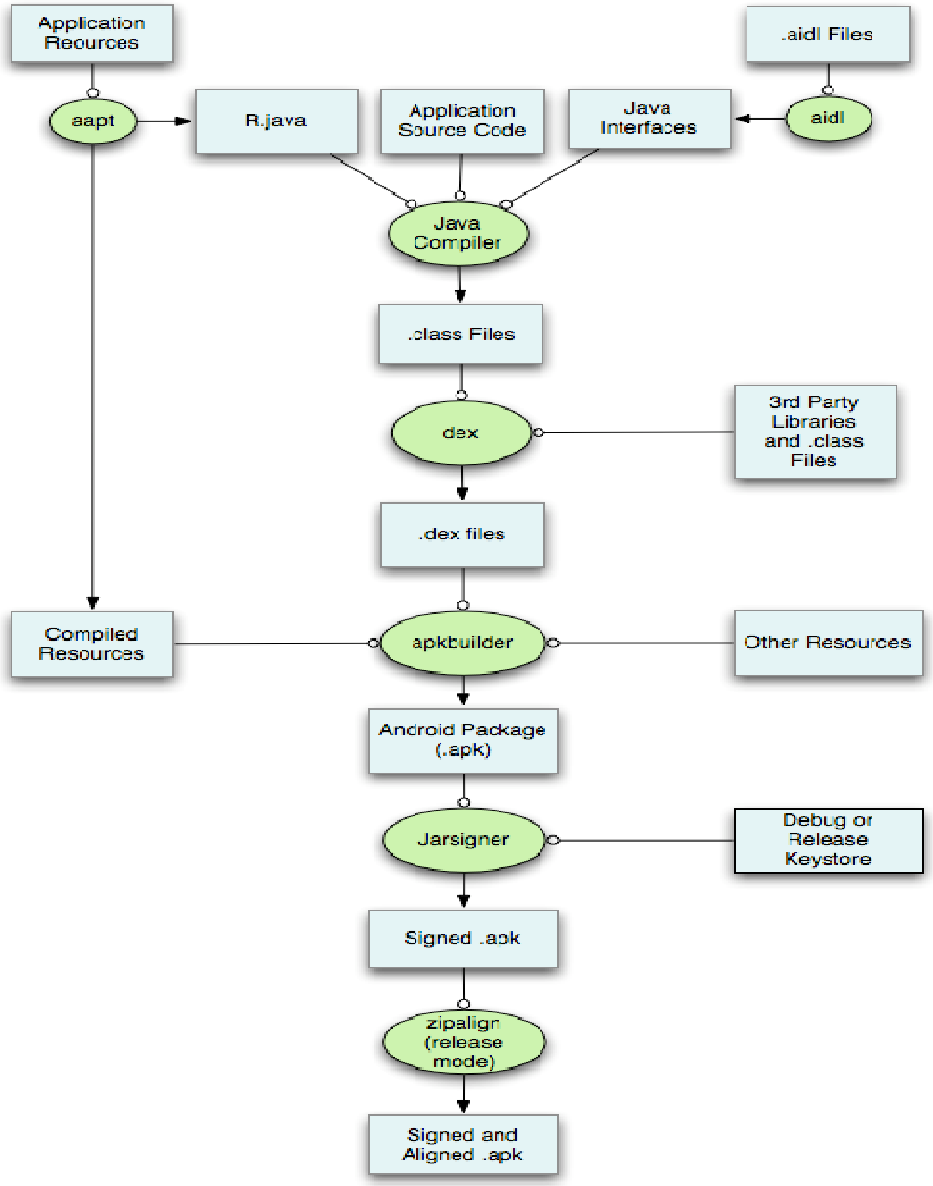

Рисунок 2.10 - Процес побудови додатку

Щоб поширювати файл .apk серед користувачів, необхідно підписати його ключем публікації. В процесі побудови утиліта aapt (Android Asset Packaging Tool) компілює ресурси файлів макетів в більш компактний формат. Відкомпілювані ресурси упаковуються в файл .apk. Потім, коли метод setContentView (...) викликається в методі onCreate (...) класу QuizActivity. QuizActivity використовує клас Layoutlnflater для створення примірників усіх об'єктів View, визначених у файлі макета. Компіляція вихідного коду в Android Studio проводиться з допомогою aapt (Android Asset Packaging Tool) - утиліти, яка шукає в проекті компільовані ресурси,

такі як AndroidManifest.xml і xml файли з res / і компілює їх в бінарне представлення, а спочатку бінарні ресурси не компілюються.

Далі, ця утиліта генерує найважливіший клас R.java для вашого додатку, завдяки якому ви можете звертатися до ресурсів з вашого коду без будь-яких труднощів з читанням файлів, як скажімо с / assets. До речі, крім компіляції xml-ресурсів, aapt створює файл resources.arsc, який представляє собою таблицю для маппінга ресурсів у час виконання додатку, туди входять всі ресурси з / res /, в тому числі і / Res / raw /, вміст / assets не включається в таблицю. Утиліта aapt знаходиться в папці platform-tools Android SDK. У файл classes.dex компілятор записує весь виконуваний код проекту. У підсумку ми отримуємо скомпільовані ресурси - xml-ресурси додатку, скомпільовані в бінарне представлення [8].

Збірка ресурсів робиться для того щоб отримати .apk файл. Цей файл ми зможемо запустити в емуляторі або на пристрої. Для збірки пакетів в файл послідовно використовуються кілька утиліт:

- apkbuilder - утиліта, якій на вхід подають скомпільовані ресурси, classes.dex і інші ресурси, а вона збирає з цього наш жаданий .apk файл. Утиліта лежить в папці tools Android SDK.

- jarsigner - це утиліта Oracle для підписання .jar архівів. Він підписує ваш .apk обраним вами ключем.

- Zipalign - утиліта, яка оптимізує .apk для більш швидкого запуску і меншого споживання ОЗП при роботі програми. Вона вирівнює вміст .apk для більш ефективної розархівації.

Зібраний додаток в файл .apk - це архів, що містить скомпільовані і нескомпільовані ресурси, classes.dex, resources.arsc, META-INF, AndroidManifest.xml і т.д. Формат .apk це надбудова над .jar, а .jar - надбудова над zip, так що, .apk ви можете відкрити zip архіватором.

В Android унікальним ідентифікатором додатка є ім’я пакету додатку. Але щоб зловмисник не зміг підмінити ваш встановлений додаток на свій з

таким же пакетом, Android виконує перевірку, на те щоб новий .apk був підписаний тим же сертифікатом, що і вже встановлений.

Крім того, якщо є викладений додаток в Google Play, не можна оновити його, якщо нова версія підписана іншим сертифікатом. Так що потрібно підписати сертифікат, а так само не забути пароль до нього [15].

Debug або Release сховище ключів - сховище, з якого jarsigner візьме ключі для підпису додатка. Якщо ви збираєте Debug версію (для запуску на емуляторі або підключеному пристрої), то .apk підписується debug ключем, в Windows він знаходиться в папці користувача / .android /. Тепер запустимо додаток в емуляторі. Для цього в AVD Manager додаємо пристрій (див. рис. 2.11). Налаштовуємо пристрій вказуючи розширення екрана, версію Android, розмір пам'яті на диску.

Рисунок 2.11 - Вікно AVD Manager

Після того, як додали пристрій, запускаємо віртуальний пристрій (див. рис. 2.12).

Рисунок 2.12 - Віртуальне пристрій AVD Manager

Далі запускаємо Android Studio: Tools → Android і ставимо галочку навпроти рядка «Enable ADB Integration» (ADB - Android Debug Bridge).

Після цього потрібно налаштувати Android Studio так, щоб при натисканні на зелену кнопочку «Run» додаток відразу встановлювався і запускався на підключеному смартфоні. Натискаємо Run → Edit Configurations. З'являється наступне вікно (див. рис. 2.13):

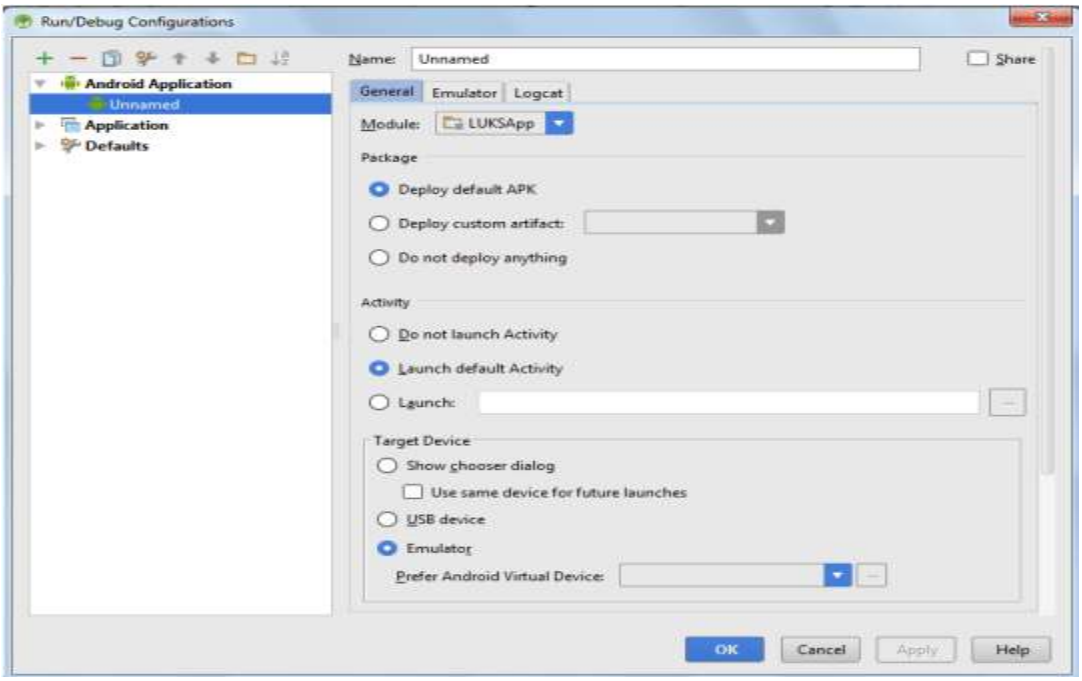

Рисунок 2.13 - Вікно Run / Debug Configuration

У блоці «Target Device» ставимо галочку на пункт «USB Device» і натискаємо OK. Після цього при натисканні на кнопку запуску програми (див. рис. 4.6), додаток встановиться і запуститься на підключеному віртуальному пристрої (див. рис. 4.7).

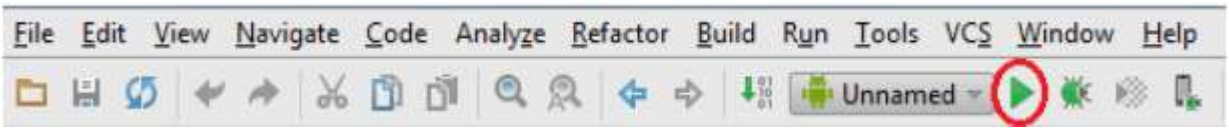

Рисунок 2.14 - ToolBar в Android Studio

Опис процесу запуску налагодження під Windows 7 в Android Studio, по кроках:

а) чи потрібно увімкнути режим налагодження USB. Для цього

відкриваємо Налаштування → Параметри розробника → Ставимо галочку «Налагодження USB». Висвітиться попередження, підтверджуючи позитивно - «Так».

Рисунок 2.15 - Запуск додатку в емуляторі

б) треба на комп’ютері встановити драйвер Android ADB Driver. Це можна зробити, якщо завантажити і запустити програму UsbDriverTool-sfx.exe. Після запуску вказуємо папку, куди потрібно розпакувати утиліту, наприклад c: \ temp, утиліта розпакується в папку C: \ temp \ UsbDriverTool \. В папці C: \ temp \ UsbDriverTool \ AndroidUsb \ знаходитиметься драйвер Android ADB Driver, який нам потрібен.

в) підключаємо смартфон з Android через USB до комп’ютера. На смартфоні повинен визначитися режим «Підключений як камера (PTP)».

Комп'ютер виявить новий пристрій, запуститься майстер установки драйвера.

г) Далі запускаємо Android Studio. Натискаємо Tools → Android і ставимо галочку навпроти рядка «Enable ADB Integration» (ADB - Android Debug Bridge). Після цього потрібно налаштувати Android Studio так, щоб при натисканні на зелену кнопочку «Run» додаток відразу встановлювався і запускався на підключеному смартфоні. Натискаємо Run → Edit

Configurations. З’являється наступне вікно (див. рис. 2.16):

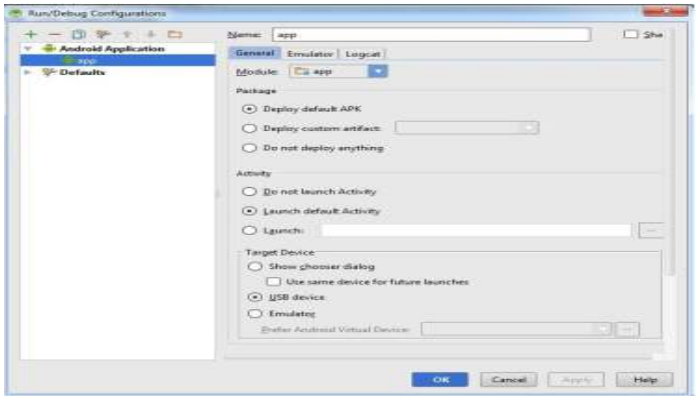

Рисунок 2.16 - Вікно Run / Debug Configuration

У блоці «Target Device» ставимо галочку на пункт «USB Device» і натискаємо OK. Після цього драйвер визначає пристрій, і при натисканні на кнопку запуску програми (див. рис. 2.17), додаток встановиться і запуститься на підключеному пристрої.

Рисунок 2.17. ToolBar в Android Studio

Після цього залишається тільки взяти в руки підключений апарат і тестувати додаток. Після запуску програми на пристрої, відкриється головне меню додатку.

2.5 Інструкція користувача

При запуску програми користувач потрапляє на головний екран з обраною вкладкою «Файли» (див. рис. 2.18). Для виведення списку файлів у форматі PDF і папок користувач повинен дати дозвіл на отримання цієї інформації, для цього потрібно натиснути кнопку «Дозволити». При натисканні на папку користувач побачить список вкладених папок і файлів. Натискання кнопки «Назад» повертає користувача до головної папки. При

натисканні на файл користувач перейде на екран управління презентацією, для цього потрібно дати дозвіл на запис аудіо і відео з камери.

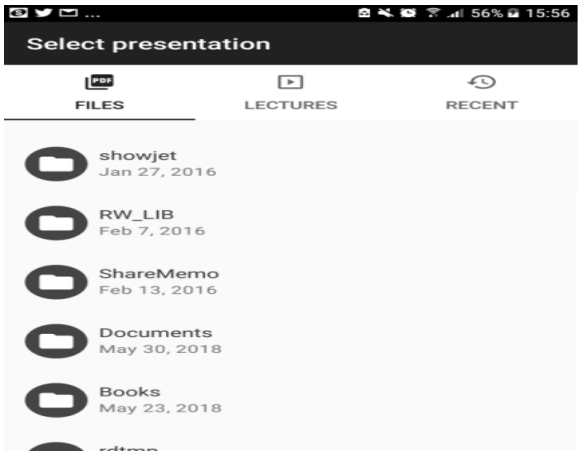

Рисунок 2.18 - Вкладка «Файли»

На вкладці «Лекції» (див. рис. 2.19) представлені лекції, що проходять в поточний момент, для кожної лекції виводиться інформація про кількість глядачів, назву лекції, відносному часу початку, а також виведено зображення першого слайда лекції. При натисканні на елемент списку користувач перейде на екран перегляду лекції.

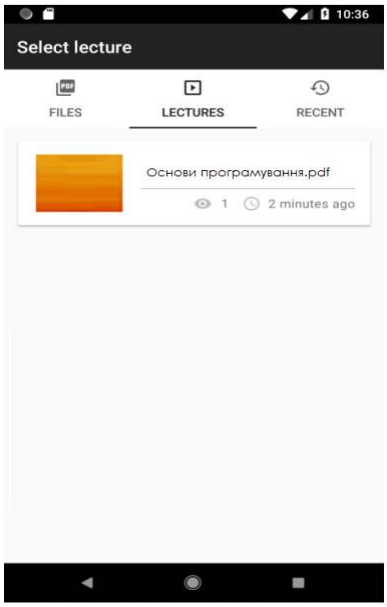

Рисунок 2.19 - Вкладка «Лекції»

На вкладці «Недавні» (див. рис. 2.20) користувач може побачити історію переглянутих лекцій. Для кожного елемента списку виводиться

назва лекції і зображення першого слайда. Натискання на елемент списку почне завантаження відео лекції та презентації.

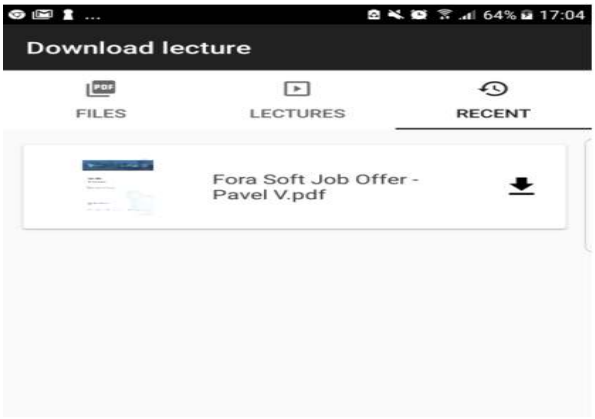

Рисунок 2.20 - Вкладка «Недавні»

На екрані «Управління презентацією» (див. рис. 2.21) користувач побачить поточний слайд презентації, а також горизонтальний список всіх слайдів.

Натискання на іконку змінити на панелі дій включає режим редагування слайду, на слайді можна робити позначки, і вони будуть відображені у всіх глядачів. Також можна вимкнути режим редагування натиснувши скасування на панелі дій (всі позначки очистяться на поточному слайді) або підтвердження змін (див. рис. 2.22).

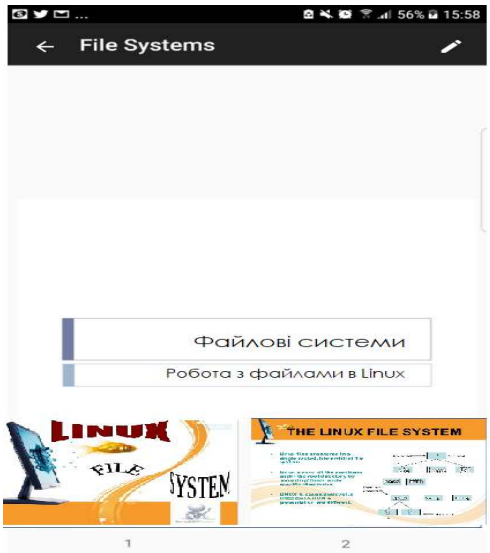

Рисунок 2.21 - Екран «Управління презентацією»

Рисунок 2.22 - Панель управління в режимі змін

Під час перегляду презентації відображається точно такий же інтерфейс, як і на екрані «Управління презентацією» крім іконки редагування на панелі дій і списку всіх слайдів. У горизонтальному режимі роботи програми на екрані видно відео з камери лектора (див. рис. 2.23). Під всіма режимами роботи користувач чує лектора.

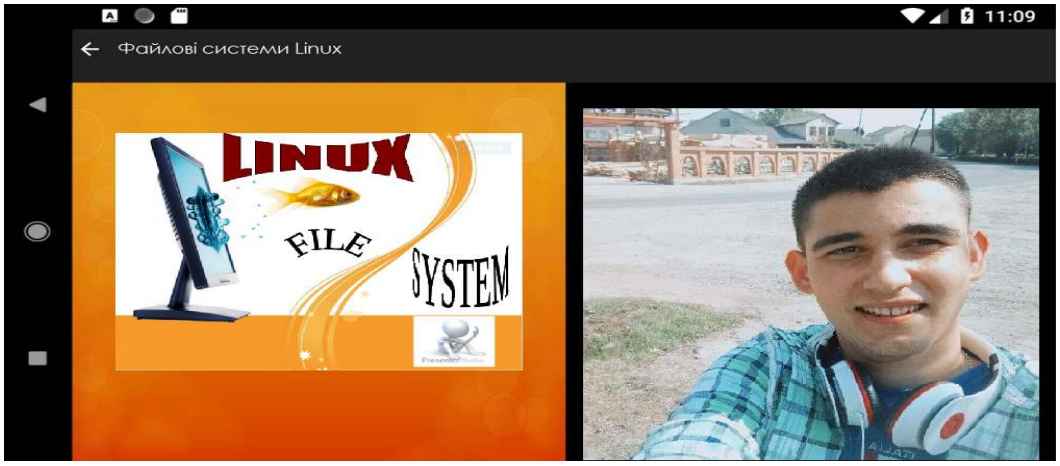

Рисунок 2.23 - Екран «Перегляд презентації»

Даний додаток досить зручний у користуванні, не вимагає додаткових знань і допомагає як лекторам так і слухачам у представленні лекційних матеріалів.

В рамках кваліфікаційної роботи розроблявся мобільний додаток під ОС Android для проведення онлайн презентацій. Споживачів цієї програми можна розділити на дві групи: особи, які бажають продемонструвати презентацію і їх глядачів, наприклад, лектор і студенти. Кожна з цих груп очікує високоякісний програмний продукт. Затребуваність і рейтинг додатку безпосередньо залежать від його якості.

Поняття якість кожен розуміє по-своєму, для одних - це чуйність системи, для інших стабільність роботи програми. Для вирішення цієї

неоднозначності міжнародною організацією по стандартам були сформульовані терміни в області якості (стандарт ISO 9000). В цьому стандарті термін «якість» визначено наступним чином - «ступінь відповідності сукупності властивих характеристик вимогам»

Якість програмного продукту може бути оцінена за наступними характеристикам: - функціональні можливості - виконання поставлених вимог; - надійність - відмовостійкість, здатність виконувати функціональні можливості в заданих умовах; - практичність - простота і зручність використання; - ефективність - забезпечення очікуваного рівня продуктивності; - супровід - процес підтримки, оптимізації і зміни програмного забезпечення; - переносимість - здатність ПЗ бути перенесеним в інше оточення.

В рамках виконання роботи були проведено тестування на відповідність якості розробленого додатку.

Таблиця 2.4

Стабільність використання	
Елементи	Коментарі
Критерій: стабільність використання	Чи не відбувається аварійних виходів, несподіваних зависань на будь-яких підтримуваних версіях ОС
Тест: виконання програми з різними призначеними для користувача взаємодіями	Випадковим чином генерується послідовність користувацьких і системних подій (натискання на екран, прокрутка списку і т. д.) в випадково вибраних областях екрану
Рішення: якщо за час тесту не відбулося аварійного виходу, то додаток відповідає стандарту	

Для виконання тесту був використаний сервіс Firebase Test Lab. Він надає доступ до реальних мобільних пристроїв на ОС Android. Було вибрано пристрій для тесту, на якому були згенеровані випадкові призначені для

користувача події. Отримана карта дій представлена на рисунку 2.24. Результати проведення тесту представлені на рисунку 2.25.

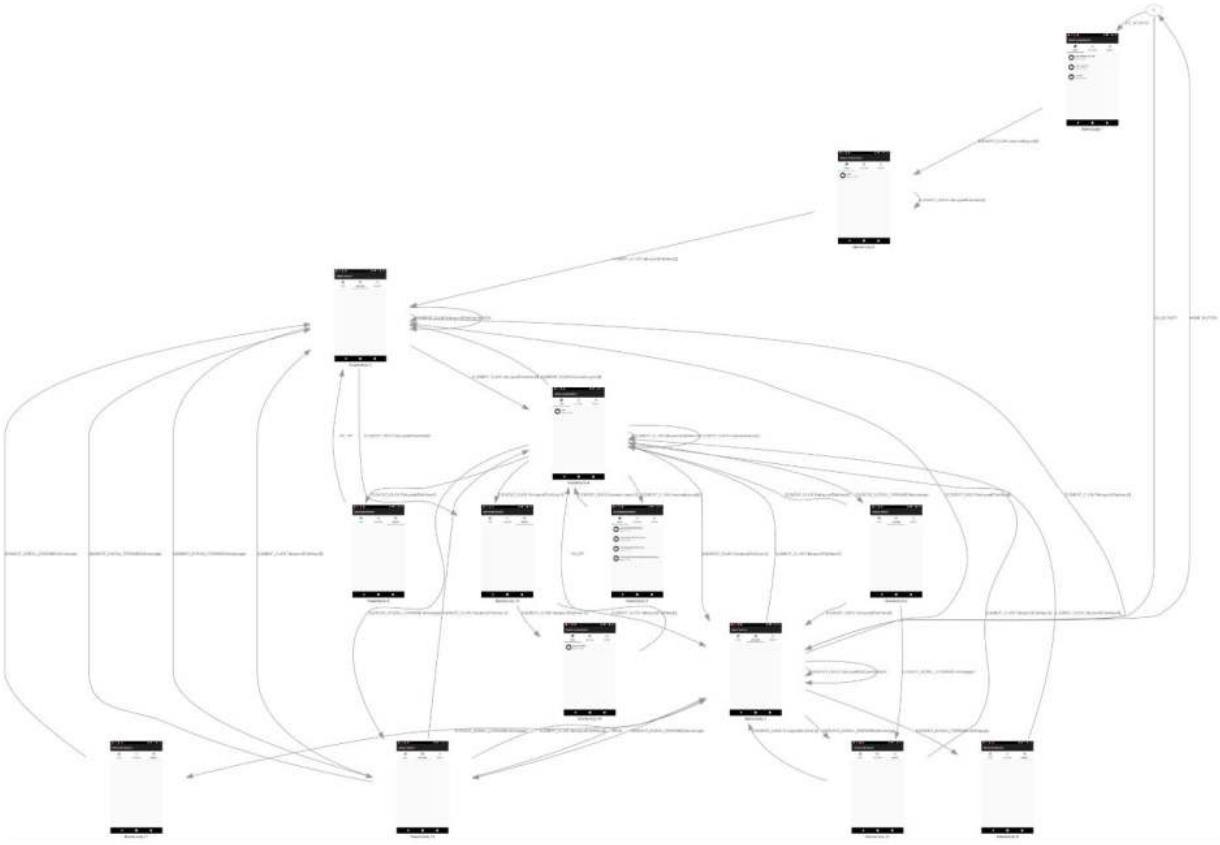

Рисунок 2.24 - Карта дій

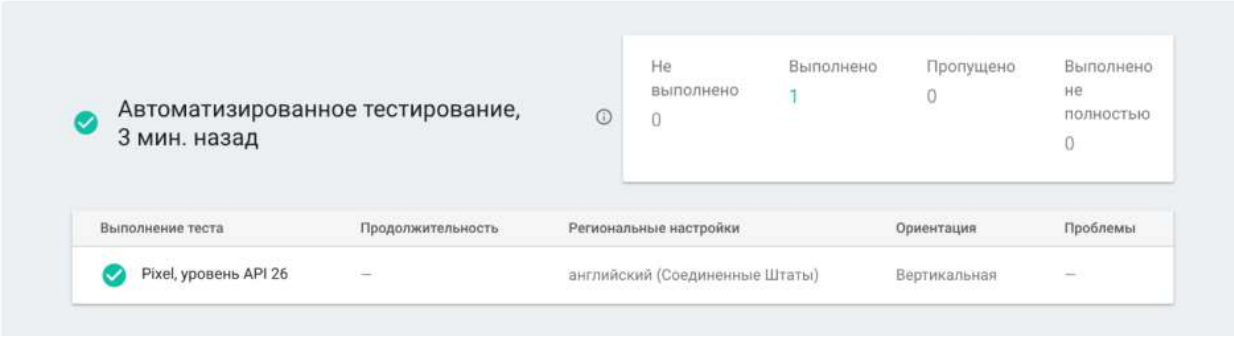

Рисунок 2.25 - Результати тестування

Для зменшення часу на впровадження змін в програмний продукт необхідно заздалегідь розробити архітектуру. Архітектура - організація системи, сукупність компонентів, а також залежностей між ними і

середовищем. Саме архітектура визначає переносимість системи, надійність і супровід.

Одним з методів підвищення надійності програмного продукту є тестування. Тестування - перевірка відповідності між реальним поведінкою продукту і очікуваним. Тестування має такі переваги:

- зменшення помилок;
- позбавлення від помилок регресії;
- збереження сумісності;

Тести діляться на наступні типи: юніт (тестування ізольованих компонентів), інтеграційні (тестування взаємодії компонентів), UI (тестування з точки зору користувача основних варіантів використання програми). На рисунку 2.26 представлена залежність кількості тестів від їх типу.

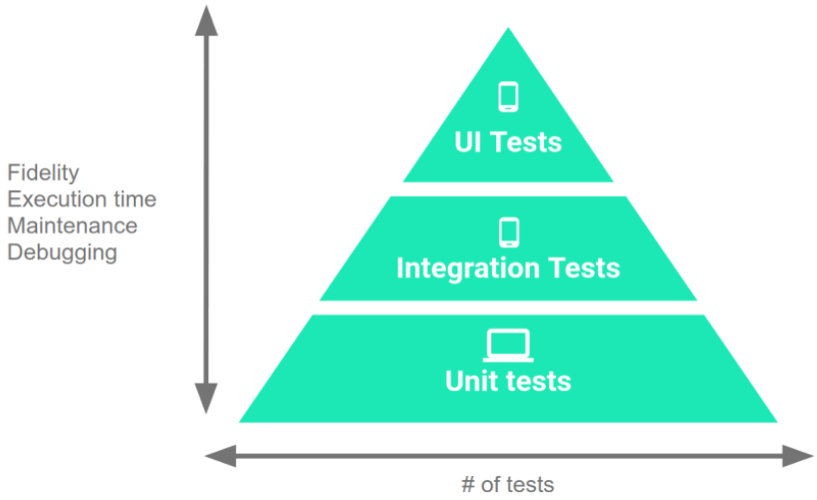

Рисунок 2.26 - Піраміда тестування

Іншим методом є застосування практики Continuous Integration (Безперервна інтеграція). Безперервна інтеграція - практика розробки ПЗ при, якій в певний проміжок часу відбувається збірка продукту. Застосування Continuous Integration дозволяє знизити кількість помилок, які виникають при інтеграції змін.

Висновки до розділу 2

У даному пункті пояснювальної записки здійснено опис процедур тестування та їхніх результатів, описані тест-вимоги до програмного забезпечення, а також виявлені дефекти. По результатам тестування сформовано підсумок тестування. Також в даному розділі було розкрито питання встановлення та налаштування програмного забезпечення, а також вказані вимоги, дотримання яких необхідно для користування програмою. У даному розділі також описана інструкція користувача для роботи із додатком.

РОЗДІЛ 3
БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

3.1. Класифікація небезпек.

Перелік назв, термінів можливих небезпек, тобто номенклатура небезпек, нараховує понад 150 найменувань і при цьому не вважається за повну. В окремих випадках складається номенклатура небезпек для окремих об’єктів (підприємств, цехів, професій, місць праці та інше).

Джерелами (носіями небезпек) є природні процеси і явища, техногенне середовище та людські дії. Небезпеки існують у просторі й часі та реалізуються у вигляді потоків енергії, речовини та інформації.

При ідентифікації небезпек, тобто при знаходженні типу небезпеки та встановленні її характеристик, необхідно виходити з принципу “все впливає на все”, тобто джерелом небезпеки може бути все живе й неживе, а підлягати небезпеці також може все живе й неживе.

Ідентифікація необхідна для розробки заходів щодо запобігання небезпекам або вже ліквідації наслідків. Класифікація та систематизація явищ, процесів, об’єктів, які здатні завдати шкоду людині (таксономія небезпек), повністю не розроблена.

Прикладом таксономії небезпек може бути такий поділ: – за походженням (природна, техногенна, екологічна); – за локалізацією (космос, атмосфера, літосфера, гідросфера); – за наслідками (захворювання, травми, загибель, пожежі); – за шкодою (соціальна, технічна, екологічна); – за сферою прояву (побутова, виробнича, спортивна, дорожньо-транспортна).

Найбільш вдалою класифікацією небезпек є класифікація за джерелами походження, згідно з якою всі небезпеки поділяються на 4 групи: природні, техногенні, соціально-політичні та комбіновані.

Подібна класифікація прийнята і в державних стандартах при визначенні надзвичайних ситуацій. Перші три класифікації належать до

елементів життєвого середовища, яке оточує людину, – природного, техногенного та соціального.

До четвертої групи належать природнотехногенні, природно-соціальні та соціально-техногенні небезпеки, джерелами яких є комбінація різних елементів життєвого середовища.

Природні джерела небезпеки – це природні об'єкти, явища природи та стихійні лиха, які можуть спричинити шкоду людині або ж становлять загрозу для життя чи здоров'я людини (землетруси, зсуви, селі, вулкани, повені, снігові лавини, шторми, урагани, зливи, град, тумани, ожеледі, блискавки, астероїди, сонячне та космічне випромінювання, небезпечні тварини, рослини, риби, комахи, гриби, бактерії, віруси, заразні хвороби).

Техногенні небезпеки – це небезпеки, пов'язані з використанням транспортних засобів, з експлуатацією підіймально-транспортного обладнання, з використанням горючих легкозаймистих і вибухонебезпечних речовин та матеріалів, процесів, що відбуваються при підвищених температурі й тиску, електричної енергії, хімічних речовин, різних видів випромінювання (іонізуючого, електромагнітного, віброакустичного).

Джерелами техногенних небезпек є відповідні об'єкти, що породжують їх. Соціальні небезпеки – це небезпеки, викликані низьким духовним та культурним рівнем (бродяжництво, проституція, пияцтво, алкоголізм, тютюнопаління). Джерелами цих небезпек є незадовільний матеріальний стан, погані умови проживання, страйки, повстання, конфліктні ситуації на міжнаціональному, етнічному, расовому чи релігійному ґрунті.

Джерелами політичних небезпек є конфлікти на міжнаціональному та міждержавному рівні, духовне гноблення, політичний тероризм, ідеологічні, міжпартійні та збройні конфлікти, війни. Найбільшу кількість становлять комбіновані небезпеки – природно-техногенні, природно-соціальні та соціально-техногенні.

Природно-техногенні небезпеки: смог, кислотні дощі, пилові бурі, ерозія ґрунтів, зменшення родючості ґрунтів, виникнення пустель, зсуви, селі, землетруси та інші тектонічні явища, які спонукала людська діяльність.

Природно-соціальні небезпеки: наркоманія, епідемія інфекційних захворювань, венеричні захворювання, СНІД. Соціально-техногенні небезпеки: професійна захворюваність, професійний травматизм, психічні відхилення та захворювання, викликані виробничою діяльністю, масові психічні відхилення та захворювання, викликані впливом на свідомість і підсвідомість засобами масової інформації та спеціальними технічними засобами, токсикоманія.

Можливість реалізації небезпеки і ступінь несприятливого впливу її на людину залежить від відповідних факторів. Фактор (лат. factor – діючий, що вчиняє) – причина, рушійна сила будь-якого процесу, яка визначає його характер або окремі риси.

У виробничій сфері фактори поділяються на вражаючі, небезпечні та шкідливі. Вражаючі фактори можуть призвести до загибелі людини. Небезпечні фактори викликають в окремих випадках травми чи раптове погіршення здоров’я (головний біль, погіршення зору, слуху, зміни психологічного та фізичного стану).

Шкідливі фактори можуть спричиняти захворювання чи зниження працездатності людини як у явній, так і прихованій формах. Розподіл факторів на вражаючі, небезпечні та шкідливі – досить умовний.

3.2. Заходи та засоби захисту людини від дії електричного струму

Правила електробезпеки визначають два види заходів, що забезпечують безпеку робіт в електроустановках:

1) організаційні заходи;

2) технічні заходи і засоби захисту.

До організаційних заходів належать:

а) вимоги до електротехнічного персоналу:

– вік персоналу для самостійної роботи повинен бути не менше 18 років;

- персонал повинен бути здоровий, не мати хвороб і каліцтв, що перешкоджають роботі в електроустановках (медичні установи мають перелік хвороб, при яких не можна працювати в електроустановках);
- – персонал повинен бути навчений, мати кваліфікаційну групу, що свідчить про рівень знань у галузі правил експлуатації електроустаткування і техніки безпеки;

б) усі роботи в електроустановках виконуються, як правило, за нарядом, і тільки для оперативно-чергового персоналу припускається виконання робіт з усного розпорядження із записом в оперативному журналі.

До технічних заходів належать:

- відключення місця роботи, тобто струмопровідних частин або устаткування, на яких будуть виконуватися ремонтні роботи або роботи з налагодження;
- – встановлення попереджувальних, забороняючих плакатів і огороджень місця роботи; – перевірка відсутності напруги;
- – накладення переносних захисних заземлень на відключені струмопровідні частини з усіх боків, звідки може надходити напруга.

Більшість нещасних випадків на виробництві виникають через невиконання організаційних заходів, при цьому розподіл нещасних випадків за причинами виникнення наступний: - технічні 33,8 % (відсутність або несправність засобів безпеки 12,4 %; конструктивні недоліки устаткування, недоліки в проектах або відступ від проектів 10,0%; несправність устаткування 7,6 %; інші 3,8 %) - організаційні 66,2 % (порушення вимог

нормативних документів безпеки 32,1 %; відсутність, неповнота, неправильне оформлення документів з безпеки праці 10,3 %; погана організація забезпечення безпеки робіт з боку ІТП (інженерно-технічних працівників) 4,4 %; порушення організації робіт за нарядом-допуском 3,8 %; необережність, неуважність постраждалого 2,8 %; інші 5,2 %)

Захисними засобами в електроустановках називаються прилади, апарати, переносні пристосування й пристрої, що служать для захисту персоналу, який працює в електроустановках, від ураження електричним струмом, а також від впливу електричної дуги, продуктів її горіння і т. д.

Вони поділяються на ізолюючі, огороджувальні та допоміжні. Ізолюючі захисні засоби – це засоби, виготовлені з ізоляційного матеріалу (бакеліт, текстоліт, фарфор, гума, пластмаса та ін.). Вони, у свою чергу, поділяться на основні й додаткові.

Основні захисні засоби – це такі, ізоляція яких надійно витримує робочу напругу електричної установки. За допомогою основних засобів можна торкатися струмопровідних частин, що знаходяться під напругою.

До них відносяться: оперативні і вимірювальні штанги; ізолюючі і струмовимірні кліщі; покажчики напруги; спеціальні пристрої для ремонтних робіт (ізолювання майданчика, східці, ланки телескопічних вишок і т. д.).

В установках до 1 000 В основними захисними засобами є: діелектричні рукавички; інструмент з ізольованими ручками (заводського виготовлення); індикатори. Додаткові захисні засоби – це такі засоби, що не гарантують надійну ізоляцію від робочої напруги і є додатковим заходом.

До них належать: – в установках вище 1000 В – діелектричні рукавички, діелектричні боти, діелектричні килими; ізолюючі підставки на порцелянових ізоляторах; – в установках до 1000 В – діелектричні калоші, діалектричні коврики, ізолюючі підставки.

Огороджуючі захисні засоби. До них належать: ширма, бар'єри, щити, сітки і т. д.; ізолюючі накладки і ковпаки (з ізоляційного матеріалу); переносні захисні заземлення; попереджувальні, забороняючі плакати.

Допоміжні захисні засоби. Застосовуються для захисту від падіння з висоти (захисні пояси, страхувальні канати), підйому на висоту (пазурі, східці, драбини), захисту від світлових, теплових, механічних, хімічних впливів електричного струму (захисні окуляри, протигази, рукавиці, фартухи, костюми, спецвзуття та ін.).

ВИСНОВКИ

Метою даної роботи була розробка мобільного додатку для проведення онлайн презентацій.

Були виконані наступні завдання:

- проведено аналіз існуючих аналогів, виявлені їх сильні і слабкі сторони;

- описано вимоги до додатка;

- досліджено технології для потокової передачі медіа і обрана технологія для вирішення поставленого завдання;

- розроблено архітектуру та екранні форми мобільного додатку;

- розроблено архітектуру і спосіб зберігання даних на сервері;

- реалізована клієнтська частина на мові Kotlin (3000 рядків коду) і серверна частина на мові Node (230 рядків коду);

Розроблений додаток може застосовуватися в дистанційному навчанні, в аудиторіях без проектора або використовуватися студентами для перегляду матеріалів після проведення лекції.

В подальшому напрямку розвитку роботи можна вибрати підвищення стабільності роботи, перехід на використання чистого WebRTC API замість платформи OpenTok, підтримка інших форматів презентацій.

ПЕРЕЛІК ДЖЕРЕЛ

1. Руснак І.С., Бабюк А.В., Федорова О.Є. Охорона праці при роботі з комп'ютером. – Чернівці: Рута, 2005. – 128 с.

2. Електронний ресурс – http://developer.android.com.

3. Інформаційні системи. Визначення інформаційної системи. Створення моделі інформаційних систем. [Електронний ресурс] / Морозов Олександр – 2002 – Режим доступу: http://www.unicyb.kiev.ua/~boiko/it/morozoff.htm

4. Програмна інженерія.-/Лавріщева К. М.-Підручник.-К.: Академперіодика, 2008.-319 с.

5. Основы инженерии качества программных систем.-Андон Ф. И., Коваль Г. И., Коротун Т. М., Лаврищева Е. М., Суслов В. Ю.-Киев: Академпериодика, 2007.-680 с.-(рус.).

6. The Java Language Specification. Java SE 8 Edition. [Електронний ресурс]/ James Gosling, Bill Joy, Guy Steele, Gilad Bracha, Alex Buckley – Режим доступу: http://docs.oracle.com/javase/specs/jls/se8/html/index.html

7. Канер Кем, Фолк Джек, Нгуен Енг Кек - Тестирование программного обеспечения. Фундаментальные концепции менеджмента бизнес-приложений. — Киев: ДиаСофт, 2001. — 544 с. — ISBN 9667393879

8. Software Perfomance Testing. [Електронний ресурс] Вікіпедія. Режим доступу: http://en.wikipedia.org/wiki/Software_performance_testing

9. Android Debug Bridge [Електронний ресурс] Android Developers. Режим доступу: http://developer.android.com/tools/help/adb.html

10. YouTube / [Електронний ресурс] // Режим доступу: http://www.youtube.com.

11. Philips. Audio Fingerprinting technology / [Електронний ресурс] // Режим доступу: http://www.research.philips.com/initiatives/contentid/audiofp.html. 92

12.Microsoft. Communication, Collaboration, and Signal Processing / [Електронний ресурс] // Режим доступу: http://research.microsoft.com/research/ccsp/.

13.Google / [Електронний ресурс] // Режим доступу: http://www.google.com.

14.Secure Digital Music Initiative / [Електронний ресурс] // Режим доступу: http://www.sdmi.org.

15. Battle E. Scalability issues in an HMM-based audio fingerprinting / Battle E., Masip J., Guaus E., Cano P. // International Conference on Multimedia Computing and Systems.- 2004.- Vol. 1. – P. 735-738. \

16.Lourens J. Detection and Logging Advertisements Using Its Sound // Proc. COMSIG.- Johannesburg (SAR).- 1990.

17.Kurth F. Identification if Highly Distorted Audio Material for Quering Large Scale Databases / Kurth F., Ribbrock A., Clausen M. // Proc. AES 112th Int. – Munich (Germany). – 2002.

18.Batlle E. Feature Decorrelation Methods in Speech Recognition. A Comparative Study / Batlle E., Nadeu C., Fonollosa J. // Proc. of International Conference on Speech and Language Processing. – 1998. - Sydney (Australia).- P. 951-954.

19.Lu L. A robust audio classification and segmentation method / Lu L., Jiang H., Zhang H., Proc. ACM Multimedia (MM'01).– Ottawa (Canada). - 2001. P. 203– 211.

20.Batlle E. Automatic song identification in noisy broadcast audio / Batlle E., Masip J., Guaus E. // Proc. of the SIP. - 2002.

21.Morishima M. Phonetically adaptive cepstrum mean normalization for acoustic mismatch compensation / Morishima M., Isobe T., Takahashi J. // Proc. Automatic Speech Recognition and Understanding. – Santa Barbara (USA). - 1997.- P. 436-441.

22.Haitsma J. Robust audio hashing for content identification / Haitsma J., Kalker T., Oostveen J. // Proc. of the Content-Based Multimedia Indexing -

Firenze (Italy) .- 2001. 93

23.Cano P. Obust sound modeling for song detection in broadcast audio / Cano P., Batlle E., Mayer H., Neuschmied H. //Proc. AES 112th Int. Conv.- Munich (Germany). – 2002.

24.Zhang T., Hierarchical classification of audio data for archiving and retrieving // Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing.– 1999. – Phoenix (USA). - Vol. 6, pp. 3001–3004.

25. Жирков А.О. Графический метод представления и нейросетевое распознавание частотно-временных векторов речевой информации / Жирков А.О., Корчагин Д.Н., Лукин А.С., Крылов А.С., Баяковский Ю.М. // Программирование.- 2003.- №4.- С.41-52.

26.Allamanche E. Content-Based Identification of Audio Material Using MPEG-7 Low Level Description / Allamanche E., Herre J., Helmuth O., Fröba B., Kasten T. // Proc. International Symposium of Music Information Retrieval. - Indiana (USA). - 2002.

27.Cano P. A review of algorithms for audio fingerprinting. / Cano P., Batlle E., T. Kalker, Haitsma J. //Proc. IEEE Workshop on Multimedia Signal Processing. - 2002. - P. 169-173.

28.Neuschmied H. Content-based identification of audio titles on the internet / Neuschmied H., Mayer H., Batlle E. // Proc. International Conference on Web Delivering of Music.- 2001.

29.Haitsma J. Highly Robust Audio Fingerprinting System// Proc. of the 3rd Int. Symposium on Music Information Retrieval. – 2002. - P. 144-148.

30.Sd Jin Soo Seo. Linear speed-change resilient audio fingerprinting // Sd Jin Soo Seo, Haitsma J., Kalker T., Proc. IEEE Benelux Workshop on Model based Processing and Coding of Audio. – Leuven (Belgium). – 2002.

31.Zwicker E., Fastl H. Psychoacoustics: Facts and Models / Springer Verlag, 2nd ed. - 1999.

32.Plomp, R., & Levelt, W. J. M. Tonal consonance and critical bandwidth // Journal of the Acoustical Society of America. – 1965. - Vol. 37, P.548-560.

33.Dudley H. Remaking speech // Journal of the Acoustical Society of America. – 1939. – Vol. 11. No 2. – P.169-177. 94

34.Drullman R. Effect of temporal envelope smearing on speech reception / Drullman R., Festen J.M., Plomp R. // JASA. – 1994. – Vol. 95. No.2. - P.1053-1064.

35.H. J. M. Steeneken. A physical method for measuring speech transmission quality. H. J. M. Steeneken and T. Houtgast. Journal of the Acoustical Society of America, 67(1):318-326, January 1980.

36.Rhebergen, K. S., ”A Speech Intelligibility Index-based approach to predict the speech reception threshold for sentences in fluctuating noise for normal-hearing listeners,” J. Acoust. Soc. Am. 117, - 2005, P. 2181-2192.

37.T. Chi. “Spectrotemporal modulation transfer functions and speech intelligibility,” / T. Chi, Y. Gao, M. Guyton, P. Ru, and S. Shamma // Journal of Acoustical Society of America, vol. 106, no. 5, pp. 2719–2732, 1999.

38.S. Sheft “Temporal integration in amplitude modulation detection,”/ S. Sheft and W. Yost // Journal of Acoustical Society of America, vol. 88, - 1990. pp. 796– 805.

39.Marrakchi-Mezghani I. Robustness of audio fingerprinting systems for connected audio applications / Marrakchi-Mezghani I., Turki-Hadj Alouane M., Jaidane-Saidane M. // Proc. Second International Symposium on Communications, Control and Signal Processing.- 2006.

ДОДАТКИ

ДОДАТОК А

ЛІСТИНГ ОСНОВНИХ МОДУЛІВ СИСТЕМИ

app.js

```
const http = require('http');
const path = require('path');
const express = require('express');
const mongoose = require('mongoose');
const socketIO = require('socket.io');
const config = require('./config.json');
const router = require('./domain/router');
const {createRoom, joinRoom, leftRoom, getRooms, updateSlide} =
require('./domain/events');
const port = process.env.PORT || config.port;
50
const app = express();
const server = http.createServer(app);
const io = socketIO(server);
app.use(express.static(path.join(__dirname, 'public')));
// підключенння маршрутів
app.use('/', router);
// підключенння обобника подій

io.on('connection', (socket) => {
socket.on('createRoom', createRoom(socket));
socket.on('joinRoom', joinRoom(socket));
socket.on('getRooms', getRooms);
socket.on('updateSlide', updateSlide(socket));
socket.on('leftRoom', leftRoom(socket));
socket.on('disconnect', leftRoom(socket));
});
// старт сервера
server.listen(port, () => {
mongoose.connect('mongodb://localhost:27017/test');
console.log(`Server is up on port ${port}`)
});
router.js
const path = require('path');
const router = require('express').Router();
const axios = require('axios');
const formidable = require('formidable');
const PDFImage = require('pdf-image').PDFImage;
const token = require('../helpers.js');
```

```
const Room = require('../models/room.js');
// обробник маршруту завантаження презентації

router.post('/upload', function (req, res) {
const form = new formidable.IncomingForm();
form.uploadDir = path.join(__dirname, 'uploads');
form.parse(req, (err, fields, files) => {
if (err) return res.sendStatus(500);
const fieldName = Object.keys(files)[0];
const filePath = files[fieldName].path;
const pdfImage = new PDFImage(filePath);
pdfImage.convertPage(0).then((imagePath) => {
res.json({
attachment: filePath,
preview: imagePath
});
});
});
});
// обробник маршруту скачування презентаціями
router.get('/download/:filename', function (req, res) {
const filename = req.params.filename;
res.sendFile(path.join(__dirname, 'uploads', filename))
});
// обробник маршруту отримання архівів
router.get('/archives', function (req, res) {
const url =
`https://api.opentok.com/v2/project/${config.api_key}/archive`;
axios.get(url, {headers: {'X-OPENTOK-AUTH': token()}})
.then(response => {
const obj = {};
51
response.data.items
.filter(archive => archive.url !== null)
.forEach(archive => obj[archive.sessionId] = archive);
Room.find().then(rooms => {
rooms.forEach(room => {
if (obj[room.roomId]) {
obj[room.roomId].name = room.roomName;
}
});
res.json(Object.values(obj));
}
);
})
```

```
.catch(error => res.sendStatus(500));
});
module.exports = router;
events.js
const fs = require('fs');
const path = require('path');
const Room = require('../models/room');
const openTok = require('./opentok');
// функція обробки події створення кімнати

const createRoom = socket => {

// roomInfo - об'єкт з полями roomName - ім'я кімнати,
attachment - url

// презентації на сервері, preview - url зображення 1 слайда на
сервері
return (roomInfo, callback) => {
openTok.createSession({mediaMode: 'routed', archiveMode:
'always'}, (err, session) => {
if (err) return callback(err);
const {roomName, attachment, preview} = JSON.parse(roomInfo);
fs.renameSync(attachment, path.join(__dirname, 'uploads',
`${session.sessionId}.pdf`));
fs.renameSync(preview, path.join(__dirname, 'uploads',
`${session.sessionId}.png`));
const room = new Room({
roomId: session.sessionId,
roomName: roomName,
presenter: {
userId: openTok.generateToken(session.sessionId),
socketId: socket.id
},
viewers: [],
createdAt: new Date().getTime()
});
room.save().then(room => {
socket.join(room.roomId);
callback(room);
});
});
};
};
// функція обробки входу в кімнату
const joinRoom = socket => {
// roomId – ідентифікатор кімнати
```

```
return (roomId, callback) => {
Room.findOneAndUpdate({roomId: roomId}, {
$push: {
'viewers': {
userId: openTok.generateToken(roomId),
socketId: socket.id
}
}
52
}).then(room => {
socket.join(roomId);
callback(room);
});
}
};
// функція обробки виходу з кімнати
const leftRoom = socket => {
return () => {
Room.findOne({
$or: [
{
'presenter.socketId': socket.id
}, {
viewers: {
$elemMatch: {
socketId: socket.id
}
}
}
]
}).then(room => {
if (room) {
if (room.presenter.socketId === socket.id) {
room.presenter = {};
room.save();
socket.broadcast.to(room.roomId).emit('forceDisconnect');
} else {
room.viewers = room.viewers.filter(viewer => viewer.socketId !==
socket.id);
}
socket.leave(room.roomId, null);
}
});
};
```

```
};
// функція обробки отримання кімнат

const getRooms = callback => Room.find ({presenter: {$ ne:
{}}}). then (rooms => callback (rooms));

// функція обробки зміни поточного слайда

const updateSlide = socket => {

// slide - номер слайда
return slide => {
Room.findOne({
'presenter.socketId': socket.id
}).then(room =>
socket.broadcast.to(room.roomId).emit('updateSlide', slide));
}
};
module.exports = {
createRoom, joinRoom, leftRoom, getRooms, updateSlide
};
config.json
{
«api_key»: «46112232»,
«api_secret»: «7c1ef51e404b2f7a299ccadc4792bc6bb9c01048»,
«port»: 4567
}
build.gradle
53
apply plugin: 'com.android.application'
apply plugin: 'kotlin-android'
apply plugin: 'kotlin-android-extensions'
apply plugin: 'kotlin-kapt'
android {
compileSdkVersion 'android-P'
defaultConfig {
applicationId «com.pvasiliev.rtcpresentation»
minSdkVersion 21
targetSdkVersion 26
versionCode 1
versionName «1.0»
testInstrumentationRunner
«android.support.test.runner.AndroidJUnitRunner»
}
buildTypes {
debug {
buildConfigField «String», «API_KEY», «\»46112232\»»
```

```
buildConfigField «String», «NODE_ADDRESS»,
«\»http://192.168.43.196:4567\»»
}
release {
buildConfigField «String», «NODE_ADDRESS», «\»https://evening-
tor-33683.herokuapp.com\»»
minifyEnabled false
proguardFiles getDefaultProguardFile('proguard-android.txt'),
'proguard-rules.pro'
}
}
compileOptions {
sourceCompatibility 1.8
targetCompatibility 1.8
}
}
dependencies {
implementation fileTree(dir: 'libs', include: ['*.jar'])
implementation «org.jetbrains.kotlin:kotlin-stdlib-
jre7:$kotlin_version»
implementation 'com.android.support:appcompat-v7:28.0.0-alpha1'
implementation 'com.android.support:design:28.0.0-alpha1'
implementation 'com.android.support.constraint:constraint-
layout:1.1.0'
implementation 'com.squareup.retrofit2:retrofit:2.4.0'
implementation 'com.squareup.retrofit2:adapter-rxjava2:2.4.0'
implementation 'com.squareup.retrofit2:converter-gson:2.4.0'
implementation('io.socket:socket.io-client:1.0.0') {
exclude group: 'org.json', module: 'json'
}
implementation 'com.opentok.android:opentok-android-sdk:2.13.0'
implementation 'com.github.stephanenicolas.toothpick:toothpick-
runtime:1.1.3'
kapt 'com.github.stephanenicolas.toothpick:toothpick-
compiler:1.1.3'
implementation 'com.arello-mobile:moxy:1.5.3'
implementation 'com.arello-mobile:moxy-app-compat:1.5.3'
kapt 'com.arello-mobile:moxy-compiler:1.5.3'
implementation 'ru.terrakok.cicerone:cicerone:3.0.0'
implementation 'io.reactivex.rxjava2:rxandroid:2.0.2'
implementation «io.reactivex.rxjava2:rxjava:2.1.13»
implementation 'io.reactivex.rxjava2:rxkotlin:2.2.0'
implementation 'com.github.bumptech.glide:glide:4.7.1'
implementation 'com.afollestad.material-dialogs:core:0.9.6.0'
implementation 'com.afollestad.material-dialogs:commons:0.9.6.0'
```

```
implementation 'com.github.marlonlom:timeago:3.0.2'
54
implementation 'com.github.chrisbanes:PhotoView:2.1.3'
implementation 'com.jakewharton.timber:timber:4.7.0'
testImplementation 'junit:junit:4.12'
androidTestImplementation
'com.android.support.test:runner:1.0.2'
androidTestImplementation
'com.android.support.test.espresso:espresso-core:3.0.2'
}
RtcApi.kt
interface RtcApi {
@POST(«/upload»)
@Multipart
fun upload(@Part presentation: MultipartBody.Part):
Single<JsonObject>
@GET(«/download/{filename}»)
fun download(@Path(«filename») filename: String): Single<File>
@GET(«/archives»)
fun listArchives(): Single<List<Archive>>
}
FileConverterFactory.kt
// клас для десеріалізації файлу
class FileConverterFactory : Converter.Factory() {
override fun responseBodyConverter(type: Type, annotations:
Array<out Annotation>?, retrofit:
Retrofit?): Converter<ResponseBody, *>? {
return if (type == File::class.java) {
FileConverterFactory.FileConverter
} else {
null
}
}
companion object FileConverter : Converter<ResponseBody, File> {
override fun convert(responseBody: ResponseBody): File {
val inputStream = responseBody.byteStream()
val file = File.createTempFile("tmp", FORMAT_PDF)
val outputStream = FileOutputStream(file)
inputStream.copyTo(outputStream)
return file
}
}
}
RoomRepository.kt
class RoomRepository @Inject constructor(private val socket:
```

```
Socket, private val api: RtcApi) {
companion object {
private const val KEY_FILENAME = "filename"
private const val KEY_ROOM_NAME = "roomName"
}
// функція для створення кімнати

// file - файл презентації

fun createRoom (file: File): Single <Room> =

uploadFile (file) .flatMap {uploadInfo (it)} .doOnSuccess
{createPresentationScope (it, file)}

// функція для входу в кімнату

// room - інформацією про кімнату
fun joinRoom(room: Room): Single<Pair<Room, File>> =
downloadFile(room).flatMap { downloadInfo(room, it) }.doOnSuccess {
createPresentationScope(it.first, it.second) }
// функція для отримання списку кімнат
fun getRooms(): Single<List<Room>> =
55
Single.create { emitter ->
socket.emit(GET_ROOMS, Ack {
val rooms: List<Room> = Gson().fromJson(it.joinToString(), object :
TypeToken<List<Room>>() {}.type)
emitter.onSuccess(rooms)
})
}
// функція для виходу з кімнати
fun leaveRoom() = socket.emit(LEFT_ROOM)
fun getRoomConnection(): Single<Boolean> =
Single.create { emitter ->
socket.on(EVENT_DISCONNECT, {
emitter.onSuccess(true)
socket.off(EVENT_DISCONNECT)
socket.off(FORCE_DISCONNECT)
})
socket.on(FORCE_DISCONNECT, {
emitter.onSuccess(true)
socket.off(EVENT_DISCONNECT)
socket.off(FORCE_DISCONNECT)
})
```