МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки та програмної інженерії
Кафедра інженерії програмного забезпечення

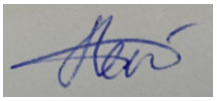
ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

Нестеренко К.С.
“____”_____2023 р.

КВАЛІФІКАЦІЙНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСНИКА ОСВІТНЬОГО СТУПЕНЯ
МАГІСТРА

Тема: «Інтелектуальні технології виявлення шахрайства в інтернеті»

Виконавець: Лепле Владислава Владиславівна 

Керівник: доцент Волкогон В.О. 

Нормоконтролер:

Київ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки та програмної інженерії
Кафедра інженерії програмного забезпечення
Освітній ступінь магістр
Спеціальність 121 «Інженерія програмного забезпечення»
Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Нестеренко К.С.
«___» _____ 2023 р

ЗАВДАННЯ
на виконання кваліфікаційної роботи студентки
Лепле Владислави Владиславівни

1. Тема кваліфікаційної роботи: «Інтелектуальні технології виявлення шахрайства в інтернеті»
затверджена наказом ректора від 04.10.2023 р. № 2034/ст

2. Термін виконання проекту: з 02.10.2023 р. до 31.12.2023 р.

3. Вихідні данні до проекту: програмний застосунок для виявлення шахрайства в інтернеті, наявність модуля авторизації, наявність бази даних, наявність модуля для взаємодії з базою даних, наявність оригінального інформаційного наповнення.

4. Зміст пояснювальної записки:
   1. Аналіз існуючих інформаційних технологій виявлення шахрайства в інтернеті.
   2. Проектування розробки програмного застосунку для виявлення шахрайства в інтернеті
   3. Програмна реалізація тестового програмного застосунку для виявлення фішингу.
   4. Тестування програмного застосунку.

5. Перелік обов’язкових слайдів презентації:
   1. Існуюче програмне забезпечення
   2. Функціональні та нефункціональні вимоги
   3. Структура системи

6. Календарний план-графік

№ пор	Завдання	Термін виконання	Відмітка про виконання
1.	Складання та затвердження графіку роботи дипломного проектування Написання 1 розділу, представлення керівнику	02.10.23 – 10.10.23	
2.	Попередній друк 1 розділу та допоміжних сторінок (чорновик) - титульної, завдання, графіка, реферат, список скорочень, зміст, вступ, список джерел, 1-й нормо-контроль.	11.10.23 – 15.10.23	
3.	Написання 2 розділу, представлення керівнику	16.10.23 – 30.10.23	
4.	Написання 3 розділу, представлення керівнику	01.11.23 – 22.11.23	
5.	Написання 4 розділу, представлення керівнику	23.11.23 – 22.12.23	
6.	Загальне редагування та друк пояснювальної записки, графічного матеріалу	23.12.23 – 24.12.23	
7.	Проходження нормо-контролю, перевірка на антиплагіат, перепліт пояснювальної записки.	25.12.23 – 27.12.23	
8.	Розробка тексту доповіді. Оформлення графічного матеріалу для презентації	28.12.23 – 28.12.23	
9.	Отримання відгуку керівника, рецензії.	29.12.23 – 29.12.23	
10.	Підготовка матеріалів для передачі секретарю ДЕК (ПЗ, CD-R з електронними копіями ПЗ, презентації, відгук керівника, рецензія) в папці	30.12.23 – 31.12.23	

7. Дата видачі завдання 02.10.2023
Керівник: доцент Вікторія ВОЛКОГОН
Завдання прийняв до виконання: Владислава ЛЕПЛЕ

РЕФЕРАТ

Пояснювальна записка до кваліфікаційного проекту «Інтелектуальні технології виявлення шахрайства в інтернеті»: 66 сторінок, 9 рисунків, 3 таблиці, 31 використаних джерел, 1 додаток.

КОРИСТУВАЦЬКИЙ ІНТЕРФЕЙС, ВЕБ-ЗАСТОСУНОК, ПРОГРАМНИЙ ПРОДУКТ, ФІШИНГ, ІНТЕЛЕКТУАЛЬНІ ТЕХНОЛОГІЇ

Об'єкт дослідження – методи та засоби розробки, що допомагають виявити шахрайство в інтернеті.

Мета кваліфікаційної роботи – є дослідження і розробка програмного застосунку для виявлення шахрайства в інтернеті.

Метод дослідження – дослідити предметну область; виконати аналіз існуючих варіантів розв'язання досліджуваної задачі; виконати аналіз методів розв'язання задачі; виконати аналіз технології розробки програмних застосунків для виявлення шахрайства в інтернеті; виконати аналіз мов програмування; виконати аналіз систем управління базами даних; дослідити CRUD-функціональність; розробити базу даних.

Наукова новизна одержаних результатів. Полягає в тому, що було доповнено існуючі уявлення про методи виявлення шахрайства в інтернеті за допомогою інформаційно-комунікаційних технологій, обґрунтувано методичну доцільність використання інформаційних технологій для виявлення шахрайства в інтернеті.

Практичне значення одержаних результатів. Полягає в можливості використання результатів дослідження та розробки застосунку у легшому способі виявлення шахрайства в інтернеті.

Особистий внесок випускника. Розробка алгоритмів для виявлення підозрілих активностей та встановлення ознак шахрайства. Застосування методів статистичного аналізу для ідентифікації нормальної та аномальної поведінки. Реалізація алгоритмів та моделей в програмному забезпеченні для виявлення підозрілих дій.

ABSTRACT

Explanatory note to the diploma project «Intelligent technologies for fraud detection on the Internet»: 66 pages, 9 figures, 3 tables, 31 used sources, 1 appendix.

USER INTERFACE, WEB APPLICATION, SOFTWARE, PHISHING, INTELLIGENT TECHNOLOGIES

The object of research is development methods and tools that help detect fraud on the Internet.

The purpose of the qualification work is to research and develop a software application for detecting fraud on the Internet.

The research method is to investigate the subject area; perform an analysis of existing options for solving the problem under study; perform an analysis of problem solving methods; perform an analysis of software application development technology to detect fraud on the Internet; perform an analysis of programming languages; perform an analysis of database management systems; explore CRUD functionality; develop a database.

Scientific novelty of the obtained results. It consists in the fact that existing ideas about methods of detecting fraud on the Internet with the help of information and communication technologies were supplemented, the methodological expediency of using information technologies for detecting fraud on the Internet was substantiated.

Practical significance of the obtained results. It consists in the possibility of using the results of research and application development in an easier way to detect fraud on the Internet.

Personal contribution of the graduate. Development of algorithms to detect suspicious activities and identify signs of fraud. Application of statistical analysis methods to identify normal and abnormal behavior. Implementation of algorithms and models in software for detecting suspicious activities.

ЗМІСТ

ПЕРЕЛІК ПРИЙНЯТИХ СКОРОЧЕНЬ

БД – Бази Даних

ПП – Програмний Продукт

СКБД – Система Керування Базами Даних

ТЗ – Технічне Завдання

CSS – Cascading Style Sheets

HTML – HyperText Markup Language

IT – Information Technology

JS –JavaScript

MySQL – Structured Query Language

PHP – Hypertext Preprocessor

URL – Uniform Resource Locator

ВСТУП

Актуальність теми. Фішингові атаки – це неправдиві повідомлення з надійних джерел, але вони можуть скомпрометувати всі типи джерел даних. Атаки можуть полегшити доступ до ваших онлайн-облікових записів і особистих даних, отримати доступ для модифікації та порушити підключені системи, такі як термінали торгових точок і системи обробки замовлень, а в деяких випадках захопити цілі комп'ютерні мережі, доки не буде сплачено викуп. Іноді хакери раді отримати вашу особисту інформацію та дані кредитної картки заради фінансової вигоди. В інших випадках фішингові електронні листи надсилаються для збору реєстраційної інформації співробітників або інших даних для використання в більш зловмисних атаках на кількох осіб або конкретну компанію.

Сьогодні статистика показує, що значний відсоток користувачів не розпізнає фішингові листи. Значна кількість організацій стикалися з фішинговими атаками, тому цей проект спрямований на захист користувачів від таких атак. Розроблений механізм виявляє елементи, властиві фішинговим листам, і попереджає користувачів про небезпеку. Крім того, для підвищення ефективності програма виконує машинне навчання раніше виявлених фішингових електронних листів.

Актуальність даної роботи пояснюється тим, що хоча все більше користувачів знайомі з методами фішингу, це не гарантує від обману, оскільки велику роль у цьому відіграє людський фактор.

Метою роботи є дослідження і розробка програмного застосунку для виявлення шахрайства в інтернеті.

Також потрібно реалізувати такі задачі:

− дослідити предметну область;

− виконати аналіз існуючих варіантів розв'язання досліджуваної задачі;

− виконати аналіз методів розв'язання задачі;

− виконати аналіз технології розробки програмних застосунків для виявлення шахрайства в інтернеті;

− виконати аналіз мов програмування;

− виконати аналіз систем управління базами даних;

− дослідити CRUD-функціональність;

− розробити базу даних.

Аналіз вимог: Збір даних про шахрайство та звичайні дзвінки з Інтернету,

Розпізнавання мовлення: переклад голосу в текстові дані.

Обробка природної мови: обробка текстових даних через бібліотеку nltk.

Модель класифікації машинного навчання: модель класифікації для виклику classify.

Програма для Android: модель класифікації машинного навчання з програмою для Android.

Об’єктом дослідження є методи та засоби розробки, що допомагають виявити шахрайство в інтернеті.

Предметом розробки є процес розробки програмного застосунку для виявлення шахрайства в інтернеті.

Наукова новизна одержаних результатів. Полягає в тому, що було доповнено існуючі уявлення про методи виявлення шахрайства в інтернеті за допомогою інформаційно-комунікаційних технологій, обґрунтувано методичну доцільність використання інформаційних технологій для виявлення шахрайства в інтернеті.

Практичне значення одержаних результатів. Полягає в можливості використання результатів дослідження та розробки застосунку у легшому способі виявлення шахрайства в інтернеті.

РОЗДІЛ 1
АНАЛІЗ ІСНУЮЧИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ВИЯВЛЕННЯ ШАХРАЙСТВА В ІНТЕРНЕТІ

1.1 Актуальність розробки програмного забезпечення виявлення шахрайства в інтернеті

До теперішнього часу людство досягло великого прогресу в техніці. Щороку з'являються нові технології, які кардинально змінюють напрямок життя. Інтернет колись став одним із таких важливих відкриттів у житті людини. За допомогою Інтернету люди можуть спілкуватися один з одним на різних континентах, знаходити, зберігати або передавати інформацію, створюючи електронні бібліотеки, до яких можна отримати доступ з будь-якої точки планети в будь-який час. Розвивайте соціальні мережі, створюйте інтернет-банки, інтернет-магазини, інтернет-ресурси тощо. Але є інший бік Інтернету — віруси, шахрайство, фішингові сайти тощо.

Інтернет-магазини є одними з найпопулярніших інтернет-ресурсів. Їх ідея проста. Інтернет-магазин - це дуже простий у використанні веб-ресурс. З цим можуть розібратися навіть люди, які далекі від IT-сфери або рідко користуються Інтернетом. Людина заходить на веб-сайт за допомогою комп'ютера, мобільного телефону чи планшета, вибирає товар, який йому подобається чи потрібен, додає товар у кошик, заповнює адресу доставки, і останній і найважливіший крок: ця покупка вимагає оплати. У деяких інтернет-магазинах це не дуже складно, у них налаштовані платіжні системи, такі як: LiqPay, Google Pay, Apple Pay або інші.

Якщо в інтернет-магазині є така функція, то, швидше за все, достовірність того, що це надійний ресурс, дуже висока. Деякі інтернет-магазини не мають так налаштованої платіжної системи, або використовують менш надійну систему, або практикують переказ коштів з картки покупця на картку продавця. Через це багато алгоритмів було розроблено тими, хто вирішив заробити на інших людях, щоб заробити на життя обманом [4].

В Інтернеті безліч шахрайств, але як їм запобігти? Ніяк, але цього можна уникнути, наприклад, створивши інформаційний ресурс, на якому можна створювати оголошення про те, як їх обдурив той чи інший продавець.

Метою даної роботи є розробка інформаційного ресурсу, який відстежує та зберігає дані про шахраїв та здатний виявляти шахрайську рекламу. Користувачі зможуть подати свої оголошення, в яких буде зазначено: інтернет-посилання на оголошення, назву оголошення, номер телефону шахрая та текст як додаткову інформацію. Заява буде відфільтрована адміністратором порталу та опублікована після перевірки адміністратором порталу. За допомогою поля пошуку можна перевірити, чи публікувалося оголошення раніше.

З кожним роком покупки в Інтернеті стають все більш популярними. Ви можете порівняти ціни з різних інтернет-джерел, вибрати найдешевший варіант і замовити доставку до дверей. Але є один важливий недолік – шахраї. Вони особливо люблять Інтернет через його анонімність і розробили безліч схем, за допомогою яких вони заробляють на людях. Мережа переповнена фальшивими інтернет-магазинами та відсутніми товарами.

До найбільш відомих видів шахрайства відносяться:

• Фішинг. Це викрадення персональних даних, таких як логіни та паролі. Дані викрадають через масові електронні листи, які нібито представляють банки та бренди, або просто через месенджери в соціальних мережах. У таких листах буде посилання на веб-сайт, відвідавши який жертва буде обманом ввести свої конфіденційні дані. Посилання на веб-сайт, надане потенційним жертвам, буде схоже на посилання на офіційний веб-сайт банку або платформи онлайн-продажів, але з незначними змінами, наприклад додаванням додаткової літери до адреси веб-сайту. Жертва переходить за посиланням, залишаючи свої дані - і шахрай отримує доступ до його коштів

• Підроблені інтернет-магазини. Шахраї створюють підроблені (не справжні) інтернет-магазини або інтернет-магазини популярних брендів і продають «повітря». Покупці платять за товари, яких не існує.

• Покупці – шахраї. Ви не тільки можете стати жертвою шахраїв, коли купуєте товар, але також можете стати жертвою шахраїв, коли продаєте товар. Адже часто покупцями можуть виступати шахраї. Вони в основному торгують на платформі продажів і мають багато планів. Практично кожен готовий відправити товар в інше місто і отримати передоплату на банківську картку. У цьому і полягає хитрість шахраїв: вони вибирають щось дороге, дзвонять запитують, можуть поторгуватися, щоб зняти настороженість, а потім погоджуються надіслати гроші.

• Картка банку «Липова». Потенційні жертви отримували нібито вигідну пропозицію: відкрити банківську картку з великим кредитним лімітом і низькою кредитною ставкою. Все, що вам потрібно, це невелика абонентська плата. [5]

За перше півріччя 2023 року в Україні зафіксовано понад 47 тис. фактів шахрайства з банківськими картками із загальними збитками понад 86 млн грн.

1.2 Існуючі продукти на ринку інформаційних технологій

1. Cyberpolice

Це офіційний сайт української кіберполіції. Оскільки це офіційний веб-ресурс, він містить багато інформації, яка не має відношення до основної мети роботи. Тим не менш, на цьому ресурсі стільки інформації, що знайти функцію, відповідальну за пошук шахраїв, буде складно.

На порталі є розділ «Stop Fraud», який спочатку важко зрозуміти людям, які не розуміють іноземних мов, адже Stop Fraud в перекладі з англійської означає Stop Fraud. На вкладці бачимо поле пошуку і напис «Перевірити інформацію можна за такими параметрами: номер банківської карти, номер телефону або посилання на сайт».

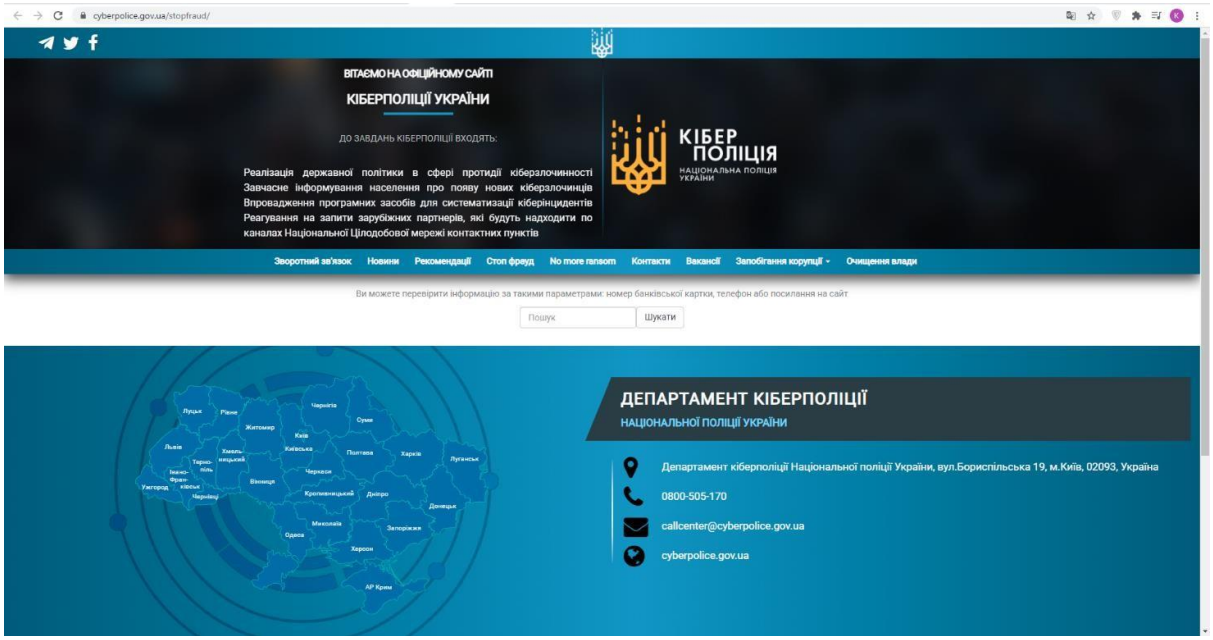


Рис. 1.1. Зовнішній вигляд веб-сайту cyberpolice.gov.ua

Ви не можете переглянути дані на сайті, якщо у вас немає необхідних критеріїв пошуку.

2. Foe

Зайшовши на ресурс, відразу можна помітити, що карта сайту непогана. Основні функціональні блоки сайту розташовані в самому верху, тому відразу можна зрозуміти, що і до чого. Основні модулі: оголошення, доповнення, коди та пошук. Зрозуміти, який блок за яку функцію відповідає, неважко.

Однією цікавою особливістю, яка безперечно дає йому перевагу над іншими порталами, є система рейтингу оголошень. Тобто є реклама, яку люди можуть оцінити: чи вони вважають її корисною.

Крім того, вибравши будь-яке оголошення, ви зможете дізнатися про нього більш детально.

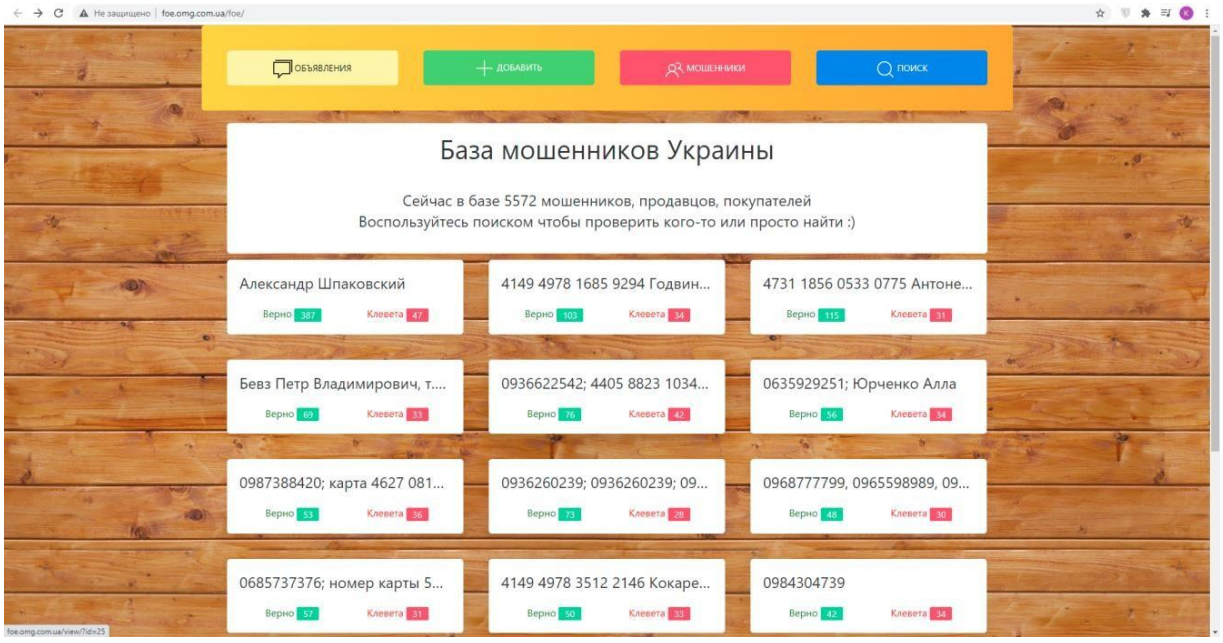


Рис. 1.2. Зовнішний вигляд веб-сайту foe.omg.com.ua/foe

3. Privatbank

Цей ресурс, як і перший, також є офіційним. Інтернет-ресурси належать офіційному банку України - Приват Банку. Після натискання посилання користувач відразу потрапляє на сторінку, де можна вибрати країну та регіон.Якщо користувач вибирає регіон, йому показуються всі шахраї в цьому регіоні.

Для перегляду інформації необхідно завантажити файл. Функція пошуку на порталі відсутня.

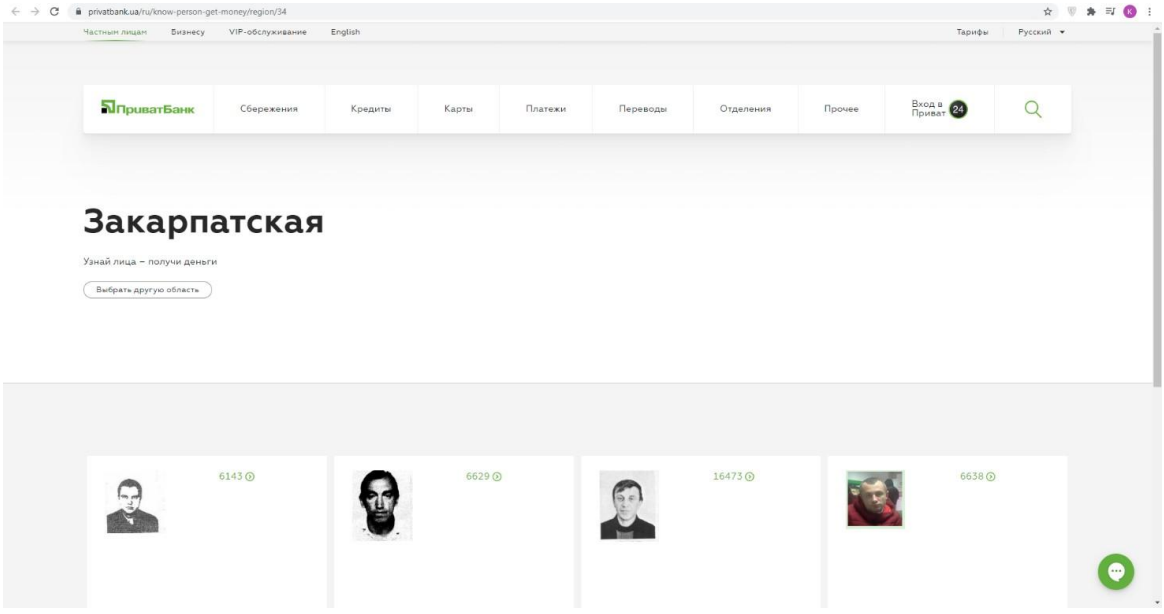


Рис. 1.3. Зовнішній вигляд веб-сайту privatbank.ua

Після огляду інформаційних порталів, можна виділити як переваги так й недоліки. Аналізуючи результат, можна виділити такі переваги й недоліки.

Таблиця 1.1.

Інформаційний портал	Переваги	Недоліки
Cyberpolice	1) Офіційний портал країни; 2) Присутній функціонал пошуку; 3) Приємний дизайн;	1) Не має можливості переглянути оголошення, якщо критерії пошуку відсутні; 2) Складно знайти потрібний функціонал; 3) Інформаційний портал спеціалізуються на різноманітному фунціоналі; 4) Відсутня можливість створення оголошень
Foe	1) Присутня можливість створення оголошень; 2) Присутній функціонал пошуку; 3) Система оцінювання оголошень; 3) Можливість перегляду оголошень без критерію пошуку; 4) Можливість більш детальніше ознайомитись з оголошенням	1) Створення оголошень відбувається на сторонньому порталі; 2) Не ергономічний дизайн;

Privatbank	1) Офіційний портал країни; 2) Пошук шахраїв по регіону; 3) Зображення шахрая;	1) Відсутній функціонал пошуку; 2) Перегляд оголошень тільки доступний при завантаженні файлу 3) Відсутня можливість публікації оголошень

1.3 Визначення проблематики кваліфікаційної роботи

Проблематика дослідження інформаційних технологій для виявлення шахрайства в Інтернеті є актуальною та складною завданням, оскільки інтернет надає безмежні можливості для шахраїв і злочинців здійснювати різноманітні види атак та обману. Основні аспекти проблематики цього дослідження включають:

Різноманітність шахрайств: Шахраї можуть використовувати різні технології та методи обману, включаючи фішинг, соціальний інженерінг, шкідливе програмне забезпечення, обман через соціальні мережі, а також криптовалютні та фінансові махінації.

Складність виявлення: Шахраї постійно адаптуються до захисних заходів, що призводить до складності виявлення шахрайської діяльності. Вони використовують нові технології, таємність і анонімність для уникнення виявлення.

Великий обсяг даних: В Інтернеті щодня генерується величезна кількість інформації, і важко визначити легітимну діяльність від шахрайства. Обробка великих обсягів даних в реальному часі є викликом для систем виявлення шахрайства.

Приватність і безпека даних: При розробці інформаційних технологій для виявлення шахрайства необхідно забезпечувати захист особистих даних користувачів та дотримання приватності, щоб уникнути порушення законів і прав користувачів.

Надійність та точність: Системи виявлення шахрайства повинні бути надійними і точними, оскільки помилки можуть призвести до невинних людей отримання хибної позитивної або негативної реакції.

Легіслятивна база: Легісляція та правові аспекти, пов’язані з виявленням шахрайства в Інтернеті, можуть варіюватися в різних країнах. Для ефективного боротьби з цією проблемою потрібна адекватна легіслятивна база.

Підготовка персоналу: Інформаційні технології для виявлення шахрайства вимагають висококваліфікованого персоналу, знань у сфері кібербезпеки та аналітичних навичок.

Дослідження інформаційних технологій для виявлення шахрайства в Інтернеті має на меті розробку та вдосконалення систем та алгоритмів, які допомагатимуть виявляти та запобігати шахрайству в цифровому просторі, забезпечуючи безпеку та довіру користувачів Інтернету.

1.4 Аналіз методів розпізнавання фішингу

Існує два основних методи виявлення фішингу: навчання користувачів і класифікація програмного забезпечення [4].

1) Навчання користувачів: користувачів можна навчити краще розуміти природу фішингових атак і таким чином правильно ідентифікувати фішингові та справжні веб-сайти. Це різко контрастує з класифікацією в [7], де навчання користувачів вважається профілактичним заходом. Але метою навчання користувачів є остаточне виявлення фішингових атак від цих користувачів, тому це вважається методом ідентифікації.

2) Використовуйте програмне забезпечення: цей метод спрямований на більш точну класифікацію фішингових і справжніх веб-сайтів і зменшення розриву, спричиненого людською помилкою чи незнанням. Це важливий недолік, який необхідно усунути, оскільки навчання користувачів коштує дорожче, ніж автоматична класифікація, і не завжди можливо. Наприклад, коли база користувачів дуже велика (PayPal, eBay, Amazon тощо).

Точність розпізнавання можна підвищити в процесі навчання класифікатора (людини або програмного забезпечення). Під час ідентифікації користувачів якість можна покращити шляхом накопичення кінцевими користувачами знань, навчання через онлайн-досвід або зовнішні навчальні програми. У випадку класифікації

програмного забезпечення покращень можна досягти шляхом навчання класифікаторів на основі алгоритмів машинного навчання або вдосконалення правил виявлення в системах на основі правил.

Технологія ідентифікації не тільки безпосередньо захищає користувачів від потрапляння в мережу зловмисників, вона також може допомогти посилити фішингові спокуси та відрізнити фішинговий спам від нефішингового.

Метод запрограмованої ідентифікації:

1. Чорний список

Чорний список — це регулярно оновлюваний список, який містить інформацію про раніше виявлені фішингові URL-адреси та IP-адреси. Недоліком таких списків є те, що вони не захищають від новостворених фішингових сайтів, оскільки останні мають бути виявлені та внесені до списку першими. Однак рівень хибного розпізнавання чорного списку нижчий, ніж евристичний [8]. Так само в роботі [8] чорні списки виявилися неефективними для виявлення нових сайтів, виявивши лише 20% із них. Дослідження показали, що від 47% до 83% фішингових URL потрапляють у чорний список протягом 12 годин. Ця затримка є серйозною проблемою, оскільки 63% фішингових кампаній зазнають невдачі протягом перших 2 годин.

Крім чорного списку існує також білий список. На відміну від чорних списків, ці списки містять інформацію про надійні та перевірені URL-адреси. Їх можна використовувати для зниження показників FP.

2. Евристика

Евристика фішингу – це набір характеристик, які присутні в реальних фішингових атаках, але не завжди гарантовано, що вони будуть відбуватися під час кожної атаки. Якщо визначено загальний набір евристичних тестів, його можна застосовувати для виявлення новостворених сайтів. Це перевага евристики над чорними списками. Однак у цьому випадку існує ризик неправильної класифікації реальних сайтів.

Програмне забезпечення можна встановити на стороні клієнта або сервера для перевірки корисних даних різних протоколів за допомогою різних алгоритмів. Це

може бути HTTP, SMTP або будь-який інший протокол. Алгоритмом може бути будь-який механізм, що використовується для виявлення або запобігання фішинговим атакам.

Сучасні веб-браузери та клієнти електронної пошти оснащено механізмами захисту від фішингу, такими як евристичні тести для виявлення фішингових атак. Подібним чином, евристика для виявлення фішингу також може бути включена в антивірусне програмне забезпечення.

3. Візуальна схожість

Метод ідентифікації базується на зовнішньому вигляді фішингового веб-сайту, а не на аналізі вихідного коду та інформації мережевого рівня. Фішингові веб-сайти дуже схожі на оригінальні веб-сайти та призначені для введення користувачів в оману. У цьому підході для прийняття рішень використовується набір функцій, таких як текстовий вміст, формат тексту, теги HTML, CSS, зображення тощо. Підозрілі веб-сайти порівнюються з відповідними перевіреними веб-сайтами з використанням різних характеристик, і якщо схожість перевищує попередньо встановлений поріг, веб-сайт оголошується фішинговим.

Щоб уникнути виявлення фішингу, зловмисники часто вставляють зображення, Flash, ActiveX і аплети Java замість тексту HTML. Методи на основі візуальної схожості можуть швидко виявити наявність таких вбудованих об’єктів у фішингових веб-сторінках. Подібним чином методи на основі візуальної подібності використовують підписи для ідентифікації фішингових веб-сторінок. Підписи створюються з використанням загальних функцій усього веб-сайту, а не окремих веб-сторінок. Таким чином, одного підпису достатньо, щоб виявити різні цільові сторінки одного веб-сайту або різних версій веб-сайту [9].

4. Машинне навчання

Методи машинного навчання розглядають виявлення фішингових атак як проблему класифікації документів або кластеризації, де переваги машинного навчання та алгоритмів класифікації використовуються для побудови моделей, таких як k-найближчий сусід (kNN), J48, опорна векторна машина (SVM) , нейронна мережа, наївні методи Байєса та логістична регресія.

Висновок

В цьому розділі досліджувались особливості шахрайства в інтернеті, актуальне програмне забезпечення та методи розпізнавання шахрайства в інтернеті.

Інтелектуальні технології грають важливу роль у виявленні шахрайства в Інтернеті, оскільки дозволяють автоматизувати і покращити процес виявлення, а також реагування на потенційні загрози.

Машинне навчання дозволяє аналізувати великі обсяги даних для виявлення аномалій та патернів, характерних для шахраїв. Алгоритми класифікації, кластеризації та регресії використовуються для ідентифікації підозрілих взаємодій.

Глибокі нейронні мережі, зокрема з використанням навчання з підкріпленням, допомагають виявляти складні патерни та атаки, які можуть бути важко виявити за допомогою традиційних методів.

NLP дозволяє аналізувати текстовий контент, такий як електронні листи, повідомлення, коментарі тощо, для виявлення спроб фішингу, обману або інших маніпуляцій з текстом.

Великі обсяги даних зберігаються та аналізуються для виявлення аномалій, відхилень та схожих злочинів у великих масштабах.

IDS використовуються для моніторингу мережі та виявлення незвичних чи підозрілих мережових активностей.

Інтелектуальні системи можуть створювати профілі користувачів та аналізувати їх поведінку для виявлення незвичних дій, які можуть свідчити про шахрайську активність.

Використання аналітики даних та алгоритмів машинного навчання для виявлення нових шахрайських схем на основі аналізу великих обсягів історичних даних.

Застосування штучного інтелекту для миттєвого виявлення та блокування шахрайської активності в реальному часі.

Ці інтелектуальні технології спільно з великими обсягами даних та аналізом допомагають підвищити ефективність виявлення шахрайства в Інтернеті та зменшити його вплив на користувачів та організації.

РОЗДІЛ 2
ПРОЕКТУВАННЯ РОЗРОБКИ ПРОГРАМНОГО ЗАСТОСУНКУ ДЛЯ ВИЯВЛЕННЯ ШАХРАЙСТВА В ІНТЕРНЕТІ

2.1 Постановка задачі розробки

У ході виконання роботи необхідно:

− виконати аналіз технології розробки програмних застосунків для виявлення шахрайства в інтернеті;

− виконати аналіз мов програмування;

− виконати аналіз систем управління базами даних;

− дослідити CRUD-функціональність;

− розробити базу даних.

Опираючись на аналіз предметної області поставимо наступне завдання:

2. Дослідити й проаналізувати бази даних, за допомогою яких можна буде реалізувати програмне забезпечення;

3. Проаналізувати сучасні технології та обрати такий стек технологій, щоб використовуючи його можна було перевершувати конкурентів;

4. Розробити програмний застосунок з функціоналом:

- шифрування дзвінків;

- перевірка ініціатора виклику по базі;

- публікація шахраїв (зі сторони адміністратора);

- функцію пошуку користувача по базі;

- блокування виклику від незахищеного користувача;

- детальний перегляд списку шахраїв.

5. Спроєктувати користувацький інтерфейс;

6. Провести тестування створеного програмного застосунку.

5. Нефункціональні вимоги:

- використання шифрування для захисту вмісту дзвінків від несанкціонованого доступу;

- контроль цілісності дзвінків, щоб визначити, чи були вони змінені або пошкоджені під час передачі;

- можливість системного моніторингу для виявлення незвичайної чи підозрілої активності;

- механізми відновлення для відновлення роботи системи після інцидентів безпеки;

антивірусне програмне забезпечення для виявлення та усунення шкідливих програм.

2.2 Розробка засобів реалізації програмного продукту

2.2.1 Вибір мови програмування

Java - широко використовується мова програмування для написання інтернет-додатків. Мова Java широко використовувалася протягом більш як двох десятиліть. Мільйони програм Java використовуються і сьогодні. Java - це багатоплатформна, об’єктно-орієнтована і сетецентрична мова, яка сама по собі може використовуватися як платформа. Це швидка, безпечна та надійна мова програмування для всього: від мобільних додатків та корпоративного ПЗ до додатків для роботи з великими даними та серверних технологій.

Оскільки Java є безкоштовною та універсальною мовою, на ній створюються локалізовані та розповсюджувані програми.

1. Розробка ігор

Багато популярних мобільних, комп’ютерних і відеоігор створено на Java. Навіть сучасні ігри, в яких використовуються передові технології, такі як машинне навчання або віртуальна реальність, створюються за допомогою технології Java.

2. Хмарні обчислення

Мова Java часто називають WORA (Write Once and Run Anywhere – «Напиши один раз, запускай будь-де»), що робить його ідеальним для децентралізованих

хмарних додатків. Постачальники хмарних послуг вибирають мову Java для запуску програм на широкому спектрі базових платформ.

3. Великі дані

Мова Java використовується для механізмів обробки даних, які можуть працювати зі складними наборами даних та великими обсягами даних у режимі реального часу.

4. Штучний інтелект

Java – це джерело бібліотек машинного навчання. Завдяки своїй стабільності та швидкості мова стала вибором № 1 для розробки додатків штучного інтелекту, таких як обробка природної мови та глибоке навчання.

5. Інтернет речей

Мова Java використовується для програмування датчиків та апаратного забезпечення периферійних пристроїв, які можуть незалежно підключатися до Інтернету.

Секрет популярності Java полягає у простоті його використання. Деякі причини, через які розробники віддають перевагу Java іншим мовам програмування/

Високоякісні навчальні ресурси

Оскільки Java існує вже давно, для нових програмістів є безліч навчальних ресурсів. Детальна документація, вичерпні друковані матеріали та курси допомагають розробникам протягом усього навчання. Крім того, новачки можуть почати писати код Core Java, перш ніж переходити на Advanced Java.

Вбудовані функції та бібліотеки

При використанні Java розробникам не потрібно щоразу писати нову функцію з нуля. Як альтернатива Java надає багату екосистему вбудованих функцій та бібліотек для розробки низки додатків.

Активна підтримка спільноти

Java має багато активних користувачів і спільноту, яка може підтримати розробників, коли вони стикаються з труднощами при написанні коду. ПЗ Java також регулярно підтримується та оновлюється.

Високоякісні інструменти розробки

Мова Java пропонує різні інструменти для підтримки автоматизованого редагування, налагодження, тестування, розгортання та керування змінами. Ці інструменти роблять програмування на Java економічним та швидким.

Незалежність від платформи

Код Java може працювати на будь-якій базовій платформі, такій як Windows, Linux, iOS або Android без перезапису. Таким чином, мова особливо ефективна в сучасному середовищі, де програми запускаються на декількох пристроях.

Безпека

Користувачі можуть завантажувати ненадійний код Java через мережу і запускати його в безпечному середовищі, в якому він не може завдати жодної шкоди. Ненадійний код не може заразити вірус хост-системи, а також не може читати або записувати файли з жорсткого диска. Рівні безпеки та обмеження у Java також легко налаштовуються.

Усі мови програмування є засобом спілкування із машинами. Апаратне забезпечення машини реагує лише на електронний зв’язок. Мови програмування високого рівня, такі як Java, відіграють роль моста між людською та апаратною мовою. Для використання Java розробники повинні розуміти дві речі.

1. Мова Java та API

Це зовнішній інтерфейс між розробником та платформою Java.

2. Віртуальна машина Java

Це внутрішній зв’язок між платформою Java та базовим апаратним обладнанням.

2.2.2 Архітектура програмного застосунку

Концептуальна модель системи представлена діаграмами випадків за допомогою UML (Unified Modeling Language).

В UML діаграми випадків моделюють поведінку системи та допомагають визначити системні вимоги.

Діаграми випадків описують функціональні можливості високого рівня та обсяг системи. Ці діаграми також визначають взаємодію між системою та її

акторами. Варіанти використання та актори на діаграмі варіантів використання описують, що робить система та як актори її використовують, але не те, як система працює всередині.

Діаграма випадку ілюструє та визначає контекст і вимоги всієї системи або значної частини системи. Ви можете використовувати діаграму одного випадку для моделювання складної системи або створити багато діаграм для моделювання компонентів системи. Діаграми прецедентів зазвичай розробляються на початку проекту та посилаються на них протягом усього процесу розробки.

Діаграми варіантів використання корисні, коли:

- перед початком проекту необхідно створити діаграму випадку для бізнес-моделювання, щоб усі учасники проекту мали розуміння співробітників, клієнтів і корпоративної діяльності;

- збираючи вимоги, ви можете створювати діаграми випадків, щоб відобразити системні вимоги та показати іншим, що повинна робити система;

- на етапах аналізу та проектування варіанти використання та актори на діаграмі випадку можуть бути використані для визначення класів, необхідних системі;

- під час фази тестування діаграми варіантів використання можуть бути використані для визначення системних тестів [13].

За допомогою розробленої діаграми прецедентів (див. рис. 2.1) можна визначити певну послідовність дій, результатом яких буде виконання заданої функції.

Учасники системи користувач мають можливість генерувати секретні ключі зв’язку, підключатися до своїх співрозмовників і обмінюватися повідомленнями в зашифрованому вигляді.

Роль системного адміністратора може оновлювати програмне забезпечення, налаштовувати канали зв’язку та створювати звіти.

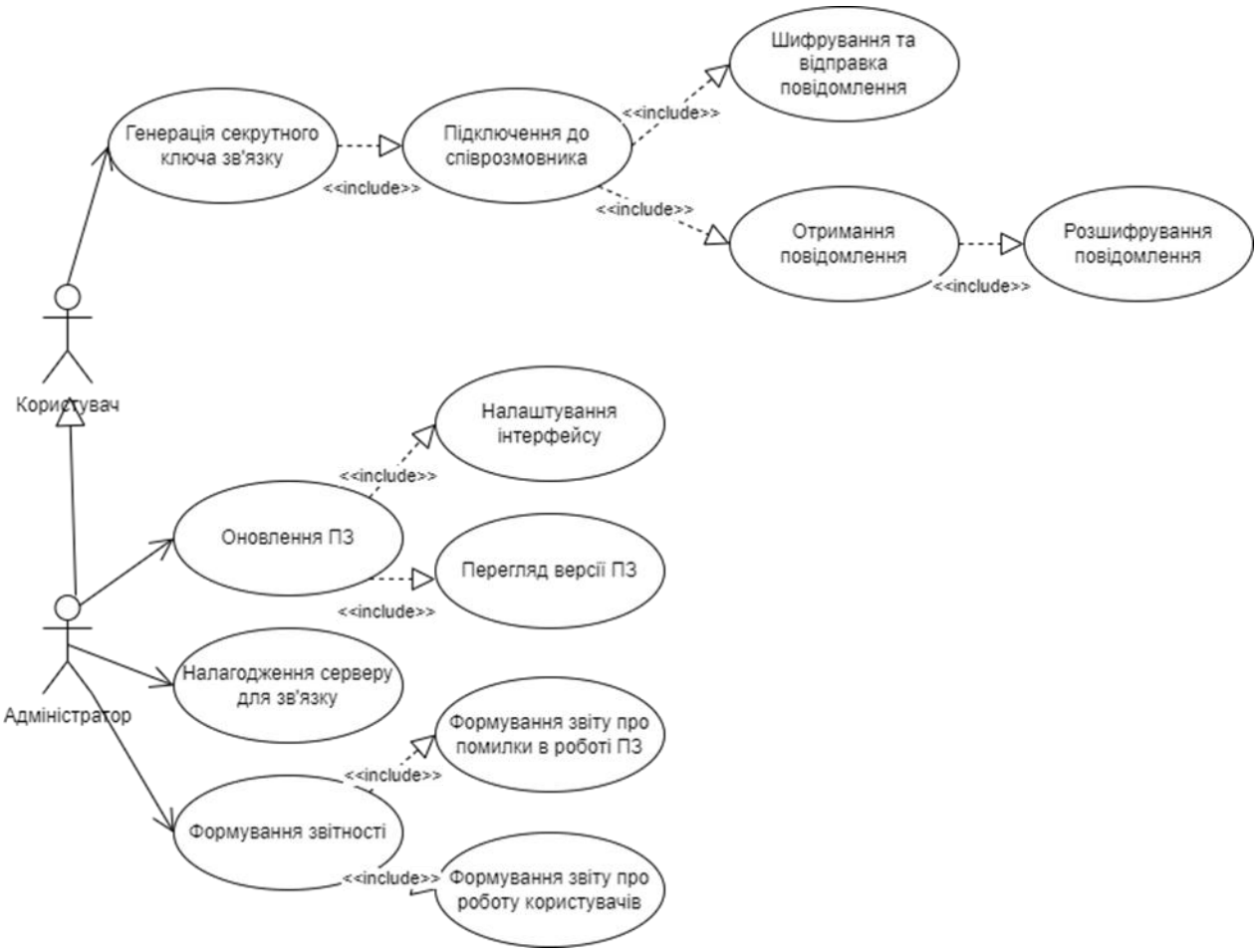


Рис. 2.1. Діаграма прецедентів проектованої системи

Очікувана поведінка системи, що проектується, може бути представлена за допомогою діаграми стану.

Діаграма стану — це графічне зображення стану системи. Діаграма станів показує модель поведінки, що складається зі станів, переходів станів і операцій.

Діаграми станів UML засновані на концепції діаграми станів Девіда Харела. Діаграма станів описує дозволені стани та переходи, а також події, які впливають на ці переходи.

Діаграми станів зазвичай використовуються у сфері вбудованих систем. Діаграми станів допомагають візуалізувати весь життєвий цикл об’єкта, тим самим допомагаючи краще зрозуміти системи на основі стану [24].

Діаграма стану представлена на рис. 2.2.

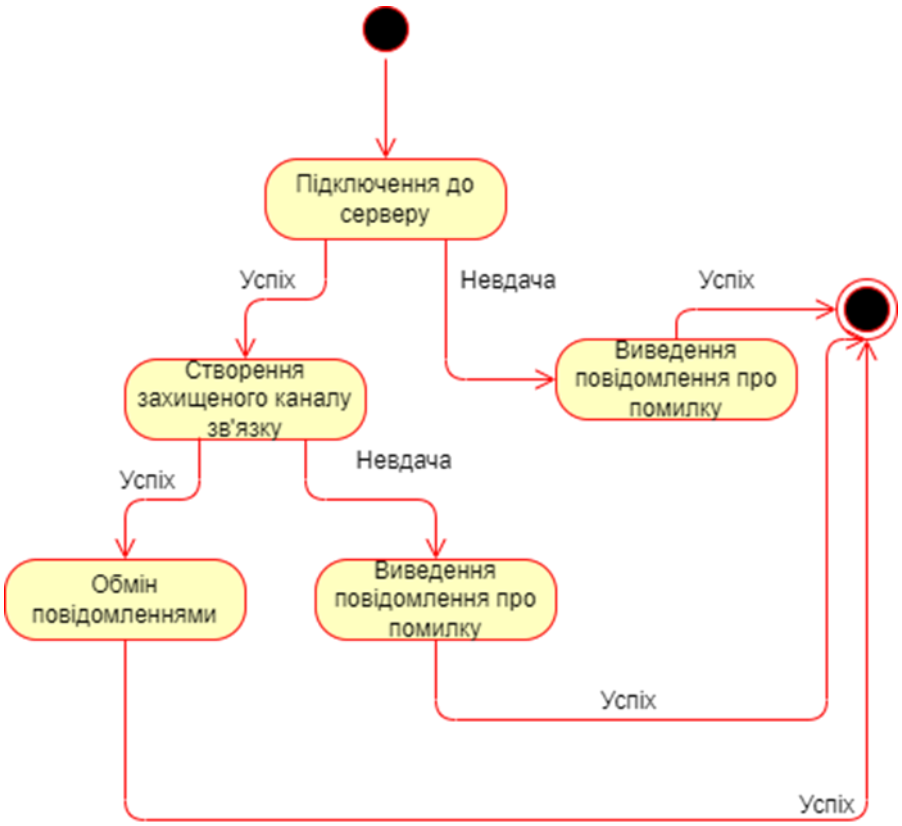


Рис. 2.2. Діаграма стану

Спочатку спробуйте підключитися до сервера, якщо з’єднання вдалось, програма запускається успішно, інакше виводиться повідомлення про помилку.

Потім спробуйте створити захищений канал зв’язку, згенерувавши ключ.Після успішної синхронізації вам буде надана можливість обмінюватися зашифрованими повідомленнями, інакше з’явиться повідомлення про помилку.

Тому визначення архітектури спроектованої системи на ранній стадії розробки полегшить подальше впровадження програмного забезпечення.

Діаграми компонентів описують організацію та зв’язки фізичних компонентів у системі.

Діаграми компонентів зазвичай малюються, щоб допомогти зрозуміти деталі реалізації моделі та перевірити, чи запланована конструкція охоплює всі аспекти необхідної функціональності системи.

У перших версіях UML компоненти, що містилися в цих діаграмах, були фізичними: документи, таблиці бази даних, файли та виконувані файли, і всі фізичні елементи мали місце розташування.

В UML 2 ці компоненти є менш фізичними, більш концептуальними, незалежними елементами дизайну (такими як бізнес-процеси), які забезпечують або вимагають інтерфейсів для взаємодії з іншими конструкціями в системі. Фізичні елементи, описані в UML 1, такі як файли та документи, тепер називаються артефактами.

Компонент UML 2 може містити кілька фізичних артефактів, якщо вони разом [15].

Діаграма компонентів представлена на рис. 2.3.

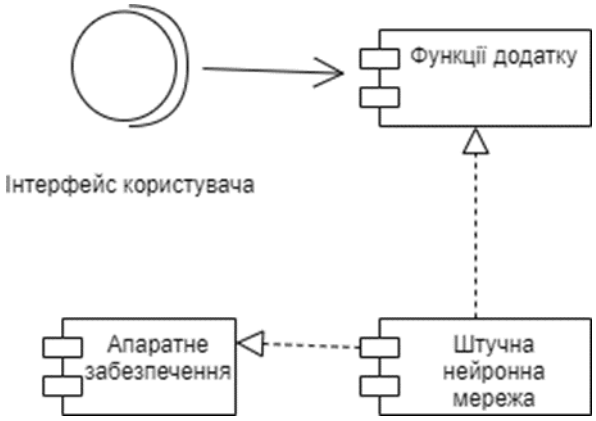


Рис. 2.3. Діаграма компонентів системи

Інтерфейс користувача потрібен для взаємодії між користувачем та ПЗ.

До функцій додатку відносяться процес шифрування, розшифрування та надсилання і отримання повідомлень.

Апаратне забезпечення складається з комп’ютерних комплектуючих системи користувача.

ШНМ являє собою головну функцію системи, а саме генерація ключа шифрування.

2.2.3 Логічна та фізична моделі бази даних

База даних - це впорядкований набір структурованої інформації або даних, які зазвичай зберігаються в електронному вигляді в комп’ютерній системі. База даних зазвичай управляється системою управління базами даних (СКБД). Дані разом з СУБД, а також додатки, які з ними пов’язані, називаються системою баз даних, або

для стислості, просто базою даних. Дані в найбільш поширених типах сучасних баз даних зазвичай формуються у вигляді рядків і стовпців в ряді таблиць, щоб забезпечити ефективність обробки і запитів даних. Потім можна легко отримувати доступ до даних, управляти ними, змінювати, оновлювати, контролювати та організовувати. У більшості баз даних для запису і запитів даних використовується мова структурованих запитів (SQL).

База даних SQL (Structured Query Language) - це система управління базами даних, яка використовує мову запитів SQL для взаємодії з даними. Ось деякі особливості баз даних SQL:

Типи даних: SQL підтримує різноманітні типи даних, такі як цілі числа, рядки, дати, час, булеві значення та інші. Це дозволяє ефективно зберігати та обробляти різноманітні дані.

Транзакції: SQL забезпечує підтримку транзакцій, що дозволяє виконувати групу операцій як єдину атомарну одиницю. Це гарантує консистентність та надійність даних.

Мова запитів: Однією з ключових особливостей SQL є можливість використання мови запитів для взаємодії з даними. SQL дозволяє виконувати операції вставки, оновлення, видалення та вибору даних з бази даних.

Нормалізація: SQL підтримує принципи нормалізації даних, що дозволяє ефективно організовувати дані для уникнення аномалій та забезпечення їх консистентності.

Обмеження цілісності: SQL дозволяє встановлювати різноманітні обмеження цілісності для забезпечення правильності та консистентності даних. Наприклад, обмеження унікальності, зовнішні ключі, обмеження CHECK тощо.

Індексація: SQL дозволяє створювати індекси для поліпшення продуктивності запитів. Індексація дозволяє швидше здійснювати пошук та сортування даних.

В’язкість даних: SQL забезпечує в’язкість даних, що означає, що дані завжди залишаються в консистентному стані під час виконання транзакцій та інших операцій.

Безпека: SQL має механізми безпеки, такі як рівні доступу, автентифікація користувачів та авторизація для забезпечення захисту від несанкціонованого доступу до даних.

Резервне копіювання та відновлення: SQL бази даних дозволяють робити резервні копії та відновлювати дані, що є важливим для захисту від втрати даних через випадкове видалення або інші проблеми.

Ці особливості роблять SQL популярним та потужним інструментом для управління даними в різноманітних застосунках та сценаріях.

Базова модель клієнт/сервер нічого не говорить про розташування різних компонентів. Проте, оскільки компоненти є різними, їх можна знайти на різних комп'ютерах. Модель розподіленої клієнт-серверної системи, в якій клієнт знаходиться на одному комп'ютері, а сервер та база даних - на іншому, настільки популярна, що її зазвичай називають моделлю клієнт-сервер (рис. 2.3).

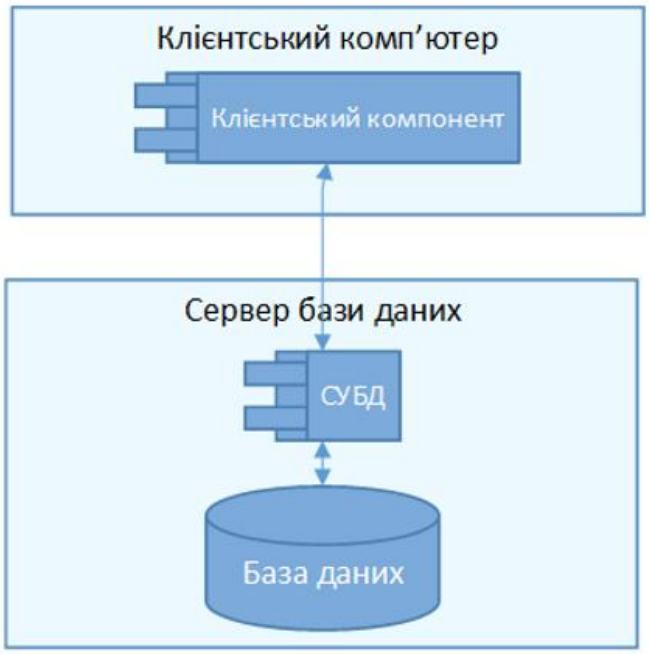


Рис. 2.3. Розподілена клієнт-серверна система

Для формування глобальної логічної моделі даних необхідно кожному виділеного об'єкта логічної моделі даних визначити набір атрибутів із зазначенням джерела даних їх заповнення. Опис сутностей та його атрибутів представлено у таблиці 2.1.

Таблиця 2.1.

Характеристика вхідних даних

Дані	Реквізити	Джерело
	ПІБ	
	дата народження	
Відомості про клієнтів	Серія та номер паспорта дата видачі паспорта Ким виданий паспорт	Паспорт клієнта
	Адреса	
	Телефон Електронна пошта	За словами клієнта
Відомості про відвідування занять клієнтом	Вид послуги дата відвідування Час відвідування Інструктор Коментарі	Реєстр відвідувань
Відомості про транзакції	Клієнт Дата транзакції Опис транзакції Сума	Транзакція клієнта
Відомості про послуги	Послуга Опис послуги Вартість	Транзакція клієнта
Відомості про адміністратора	ПІБ Телефон Електронна пошта	Дані компанії

Проведемо опис глобальної логічної моделі даних. Опис сутностей глобальної логічної моделі даних та атрибутів сутностей представлено у таблиці 2.2.

Таблиця 2.2.

Опис глобальної логічної моделі даних

Сутність	Атрибут	Опис
Клієнт	кодКлієнта	Унікальний ідентифікатор
	ПІБКлієнта	Прізвище, ім’я та по-батькові клієнта
	Дата народження	Дата народження клієнта
	Паспорт	Серія та номер паспорта клієнта
	Дата видачі	Дата видачі паспорта клієнта
	Ким виданий	Ким виданий паспорт
	Адреса	Адреса проживання клієнта
	Телефон	Контактний телефон клієнта
	Email	Електронна адреса клієнта
Послуга	кодПослуги	Унікальний ідентифікатор послуги
	Назва Послуги	Назва послуги
	ОписПослуги	Опис послуги
	Вартість послуги	Вартість послуги
Договір	Номер договору	Унікальний ідентифікатор договору
	кодЗаявки	Код заявки
	кодАдміністратора	Код Адміністратора
	ДатаДоговору	Дата оформлення договору
Адміністратор	кодАдміністратора	Унікальний ідентифікатор
	ПІБАдміністратора	ПІБ адміністратора
	ТелАдмін	Телефон адміністратора
	emailАдмін	Електронна адреса адміністратора

Висновок

В даному розділі аналізуються технології розробки програмних застосунків для виявлення шахрайства в інтернеті. Було проаналізовано такі мови веб-програмування зі сторони клієнта як HTML, XML, JavaScript. Розглянуто технології створення веб-застосунків і веб-сервісів Microsoft Silverlight, AJAX, ASP.NET. Також було проведено аналіз мов програмування і систем управління базами даних.

На основі отриманих даних мовою веб-програмування зі сторони клієнта було обрано Java, технологією створення веб-застосунків і веб-сервісів - ASP.NET з архітектурним шаблоном MVC, мовою реалізації програмного застосунку - Java, системою управління базами даних Python.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ ТЕСТОВОГО ПРОГРАМНОГО ЗАСТОСУНКУ ДЛЯ ВИЯВЛЕННЯ ФІШИНГУ

3.1 Опис програмної реалізації серверної частини та реалізація алгоритму синхронізації двох ШНМ

Використовуйте мову програмування Java для написання клієнт-серверної частини програмного коду. Програмну реалізацію серверної та клієнтської частин можна розділити на кілька етапів.

Перший етап повинен включати впровадження серверної частини. Спочатку налаштуйте можливість підключення клієнтів до сервера, відкривши порт підключення. Далі реалізується функціонал, який дозволяє отримувати і відправляти повідомлення між сервером і клієнтом.

Наступним кроком є розробка функції отримання ідентифікатора користувача, який намагається підключитися до сервера, що дозволить нам виключити спроби сторонніх підключень. Крім того, реалізована функція отримання ваг користувачів, щоб можна було контролювати процес синхронізації двох штучних нейронних мереж. Однією з найважливіших функцій серверної частини є підтвердження синхронізації штучної нейронної мережі клієнтів, тобто підтвердження того, що вихідні нейрони кожного клієнта мають однакове значення.

Другий етап розробки – впровадження клієнтської частини. Ця частина відповідає за реалізацію алгоритму синхронізації двох ДМП, тобто синхронізацію штучної нейронної мережі двох користувачів і отримання ключа шифрування. Першим кроком є впровадження функціональних можливостей, які забезпечують підключення до сервера.

Далі відправляємо ідентифікатор користувача на сервер і отримуємо ідентифікатор співрозмовника. Після цього була реалізована функція, що відповідає за синхронізацію штучної нейронної мережі користувача та штучної нейронної мережі співрозмовника, тобто реалізація алгоритму синхронізації ДМП. Після успішної синхронізації штучної нейронної мережі двох користувачів ключ

шифрування генерується та зберігається локально на кожній станції, що забезпечує додатковий захист від можливого отримання ключів третіми особами. Після отримання ключа шифрування користувач відкриває веб-додаток, який дозволяє йому обмінюватися повідомленнями зі своїм співрозмовником у зашифрованому вигляді.

3.2 Опис програмної реалізації веб-додатку

Процес реалізації веб-додатку можна розділити на два етапи: налаштування інтерфейсу користувача за допомогою HTML і реалізація функціональності системи шифрування.

Перший етап включає процес налаштування інтерфейсу користувача. HTML дозволяє налаштовувати елементи інтерфейсу відповідно до побажань розробника. Позиція історії обміну повідомленнями, позиція елемента, відповідального за введення повідомлення користувачем, позиція елемента, який виконує функцію надсилання повідомлення на сервер, і позиція елемента, який отримує секретний ключ шифрування, визначає подальше шифрування. і розшифрування повідомлення.

Другий етап — використання Java для реалізації функції. Ці функції включають: процес підключення до сервера; функцію надсилання повідомлення на сервер, а повідомлення надсилається з сервера іншому користувачеві; функцію шифрування та дешифрування повідомлень; функцію завантаження ключів шифрування на веб-додаток.

3.3 Розробка інтерфейсу видів тестів програмного застосунку

На рис. 3.1. відображено головний екран застосунку і його меню

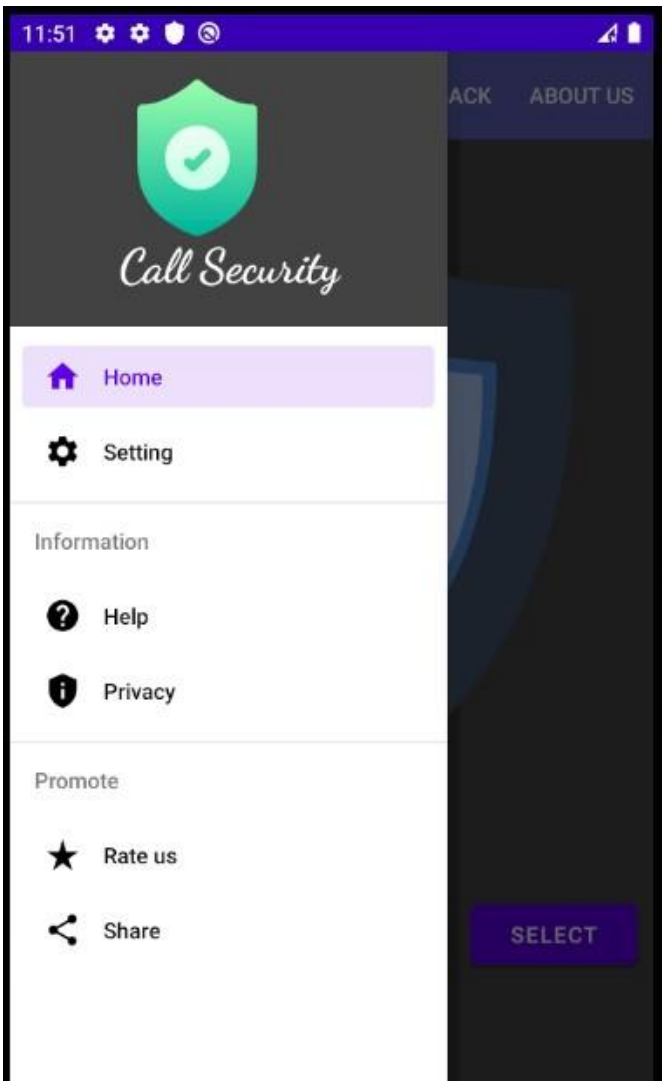


Рис. 3.1. Головний екран та меню програмного застосунку

Для того, щоб всі дзвінки користувача були зашифовані, необхідно зайти у додаток та натиснути на старт, тоді програма вмикається та всі дзвінки стають захищеними (рис. 3.2).

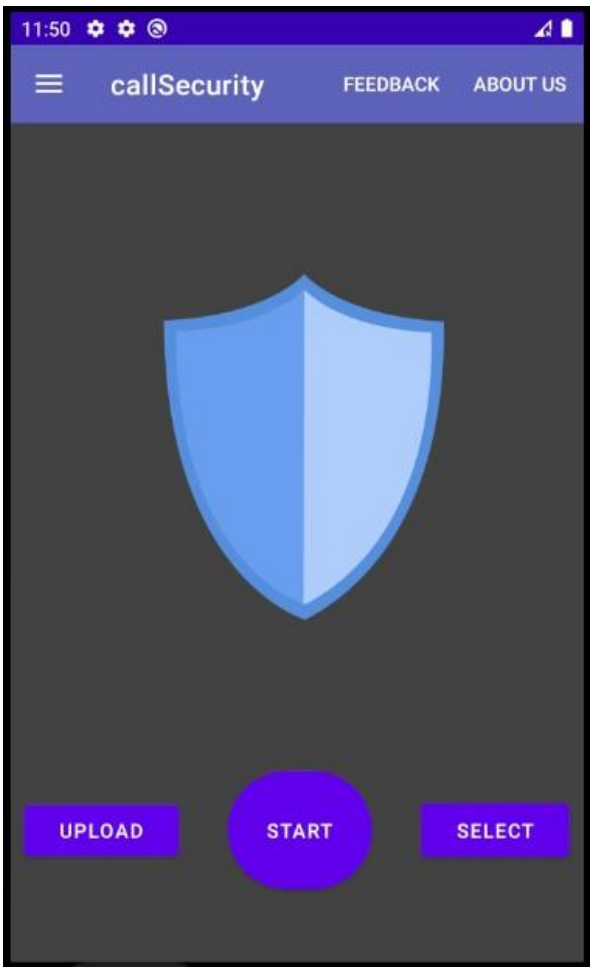


Рис. 3.2. Увімкнення шифрування дзвінку

Коли шифрування дзвінків користувачу не потрібне, необхідно повторно зайти у застосунок, натиснути «Стоп» та відключити шифрування (рис. 3.3).

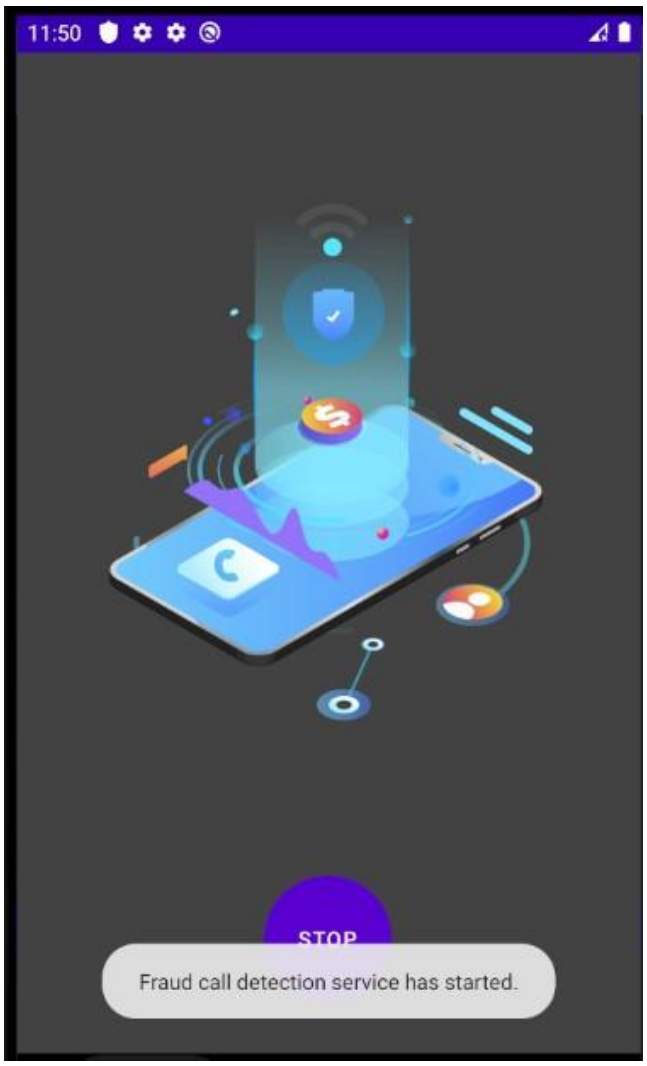


Рис. 3.3. Відключення шифрування дзвінків

3.4 Особливості створення бази даних програмного застосунку для виявлення шахрайства в інтернеті за допомогою SQL Server

База даних (скорочено – БД) – впорядкований набір логічно взаємопов’язаних даних, що використовуються спільно та призначені для задоволення інформаційних потреб користувачів.

Головне завдання БД – гарантоване збереження значних обсягів інформації (так звані записи даних) та надання доступу до неї користувачеві або ж прикладній програмі. Таким чином, БД складається з двох частин: збереженої інформації та системи керування нею.

Об’єкти бази даних:

− таблиці;

− запити;

− форми;

− звіти.

Таблиці містять дані у вигляді двовимірних таблиць. Таблиці є основою бази даних, і всі інші об’єкти бази даних залежать від них. Кожна таблиця складається із записів (рядків) і полів (стовпців). Робота з таблицями виконується в двох основних режимах: режим конструктора і режим таблиці.

Запити використовуються для отримання даних з однієї або кількох таблиць. Відбір необхідної інформації здійснюється на основі визначених користувачем критеріїв.

Форми використовуються для легкого введення даних. Форма — це діалогове вікно, яке дозволяє вводити, переглядати та друкувати дані. Форма може відображати дані з кількох таблиць або запитів.

Звіти призначені для відображення даних у зручному для користувача вигляді. На основі звіту може бути створений документ, який модна роздрукувати на принтері чи додати до іншої програми.

При побудові програмного застосунку для виявлення шахрайства в інтернеті необхідно зберігати інформацію, що міститься в програмі і програмно реалізується. Для цих цілей використовуються бази даних SQL Server 2012.

База даних у SQL Server складається з набору таблиць, у яких зберігається певний набір структурованих даних. Таблиця містить набір рядків, які також називають записами або кортежами, і стовпців, які також називають атрибутами. Кожен стовпець таблиці призначений для зберігання певного типу інформації, наприклад, дати, імена, суми в доларах і цифри.

На комп’ютері може бути встановлений один або кілька екземплярів SQL Server. Кожен екземпляр SQL Server може містити одну або кілька баз даних. У базі даних існує одна або кілька груп власності на об’єкт, які називаються схемами. У кожній схемі є такі об’єкти бази даних, як таблиці, подання та збережені процедури. Деякі об’єкти, такі як сертифікати та асиметричні ключі, містяться в базі даних, але не містяться в схемі. Додаткову інформацію про створення таблиць див.

Бази даних SQL Server зберігаються у файловій системі у файлах. Файли можна згрупувати в файлові групи. Додаткову інформацію про файли та файлові групи див. у розділі Файли та файлові групи бази даних.

Коли люди отримують доступ до екземпляра SQL Server, вони ідентифікуються як логін. Коли люди отримують доступ до бази даних, вони ідентифікуються як користувач бази даних. Користувач бази даних може бути заснований на логіні. Якщо ввімкнено містяться бази даних, можна створити користувача бази даних, який не базується на логіні.

Користувачеві, який має доступ до бази даних, можна надати дозвіл на доступ до об’єктів у базі даних. Хоча дозволи можна надавати окремим користувачам, ми рекомендуємо створити ролі бази даних, додати користувачів бази даних до ролей, а потім надати їм дозвіл на доступ. Надання дозволів ролям замість користувачів полегшує збереження дозволів узгодженими та зрозумілими, оскільки кількість користувачів зростає та постійно змінюється.

Висновок

В даному розділі було проаналізовано програмну реалізацію застосунку для виявлення шахрайства в інтернеті і розроблено його інтерфейс. Також були визначені особливості створення бази даних для програмного застосунку.

Програмний застосунок для виявлення шахрайства в інтернеті був розроблений таким чином, щоб забезпечити максимальну наочність і зручність у користуванні.

Розроблений застосунок підтримує можливості вдосконалення та покращення, що дуже важливо в умовах сучасного розвитку Internet технологій.

РОЗДІЛ 4
ТЕСТУВАННЯ ПРОГРАМНОГО ЗАСТОСУНКУ

Тестування програмного забезпечення – це процес технічного дослідження, призначений для виявлення інформації про якість продукту відносно контексту, в якому він має використовуватись. Техніка тестування також включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою оцінки. Може оцінюватись:

− відповідність вимогам, якими керувалися проектувальники та розробники;

− правильна відповідь для усіх можливих вхідних даних;

− виконання функцій за прийнятний час;

− практичність;

− сумісність з програмним забезпеченням та операційними системами;

− відповідність задачам замовника.

Оскільки число можливих тестів навіть для нескладних програмних компонент практично нескінченне, тому стратегія тестування полягає в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів. Як результат програмне забезпечення (ПЗ) тестується стандартним виконанням програми з метою виявлення баґів (помилок або інших дефектів).

Тестування ПЗ може надавати об’єктивну, незалежну інформацію про якість ПЗ, ризики відмови, як для користувачів так і для замовників.

Тестування може проводитись, як тільки створено виконуваний код (навіть частково завершено). Процес розробки зазвичай передбачає коли та як буде відбуватися тестування. Наприклад, при поетапному процесі, більшість тестів відбувається після визначення системних вимог і тоді вони реалізуються в тестових програмах. На противагу цьому, відповідно до вимог гнучкої розробки ПЗ, програмування і тестування часто відбувається одночасно [15].

Під час проведення тестування програмного застосунку для виявлення шахрайства в інтернеті виявлено, що застосунок відображається однаково в найбільш популярних браузерах, таких як Internet Explorer, Opera, Mozilla Firefox, Chrome.

Перевірка програми на юзабіліті показала, що програмний застосунок є зручним для користувача, має зручну навігаційну панель та інтуїтивно зрозумілу логічну структуру.

Під час перевірки не виявлено помилок коду.

Висновки

В даному розділі розглянута інструкція користувача. Також було проведено тестування програмного застосунку для виявлення шахрайства в інтернеті. В ході тестування жодних помилок не виявлено, програмний застосунок працює коректно.

ВИСНОВКИ

Метою даної кваліфікаційної роботи є розробка програмного застосунку для виявлення шахрайства в інтернеті

В першому розділі вивчався розвиток інформаційних технологій відображення створеної реклами, створення реклами та виявлення шахрайської реклами.

Інтелектуальні технології грають важливу роль у виявленні шахрайства в Інтернеті, оскільки дозволяють автоматизувати і покращити процес виявлення, а також реагування на потенційні загрози. Ось деякі з основних інтелектуальних технологій, які використовуються для цієї мети:

Машинне навчання та аналіз даних: Машинне навчання дозволяє аналізувати великі обсяги даних для виявлення аномалій та патернів, характерних для шахраїв. Алгоритми класифікації, кластеризації та регресії використовуються для ідентифікації підозрілих взаємодій.

Нейронні мережі: Глибокі нейронні мережі, зокрема з використанням навчання з підкріпленням, допомагають виявляти складні патерни та атаки, які можуть бути важко виявити за допомогою традиційних методів.

Обробка природної мови (NLP): NLP дозволяє аналізувати текстовий контент, такий як електронні листи, повідомлення, коментарі тощо, для виявлення спроб фішингу, обману або інших маніпуляцій з текстом.

Аналітика великих даних (Big Data): Великі обсяги даних зберігаються та аналізуються для виявлення аномалій, відхилень та схожих злочинів у великих масштабах.

Системи виявлення інтранет-загроз (IDS): IDS використовуються для моніторингу мережі та виявлення незвичних чи підозрілих мережових активностей.

Аналіз поведінки користувачів: Інтелектуальні системи можуть створювати профілі користувачів та аналізувати їх поведінку для виявлення незвичних дій, які можуть свідчити про шахрайську активність.

Роботи з використанням великих даних (Big Data Mining): Використання аналітики даних та алгоритмів машинного навчання для виявлення нових шахрайських схем на основі аналізу великих обсягів історичних даних.

Штучний інтелект (AI) в реальному часі: Застосування штучного інтелекту для миттєвого виявлення та блокування шахрайської активності в реальному часі.

Системи виявлення аномалій: Використання алгоритмів для виявлення аномалій в поведінці системи або користувачів, що може свідчити про шахрайську діяльність.

Ці інтелектуальні технології спільно з великими обсягами даних та аналізом допомагають підвищити ефективність виявлення шахрайства в Інтернеті та зменшити його вплив на користувачів та організації.

В другому розділі аналізуються технології розробки програмних застосунків для виявлення шахрайства в інтернеті. Було проаналізовано такі мови веб-програмування зі сторони клієнта як HTML, XML, JavaScript. Розглянуто технології створення веб-застосунків і веб-сервісів Microsoft Silverlight, AJAX, ASP.NET. Також було проведено аналіз мов програмування і систем управління базами даних.

На основі отриманих даних мовою веб-програмування зі сторони клієнта було обрано Java, технологією створення веб-застосунків і веб-сервісів - ASP.NET з архітектурним шаблоном MVC, мовою реалізації програмного застосунку - Java, системою управління базами даних Python.

В третьому розділі було проаналізовано програмну реалізацію застосунку для виявлення шахрайства в інтернеті і розроблено його інтерфейс. Також були визначені особливості створення бази даних для програмного застосунку.

Програмний застосунок для виявлення шахрайства в інтернеті був розроблений таким чином, щоб забезпечити максимальну наочність і зручність у користуванні.

Розроблений застосунок підтримує можливості вдосконалення та покращення, що дуже важливо в умовах сучасного розвитку Internet технологій.

В четвертому розділі розглянута інструкція користувача. Також було проведено тестування програмного застосунку для виявлення шахрайства в

інтернеті. В ході тестування жодних помилок не виявлено, програмний застосунок працює коректно.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Azzief Kh. 10 Free Mobile Apps To Help You Learn English Faster / Kh. Azzief. [Електронний ресурс]. Режим доступу: http://www.hongkiat.com/ blog/mobile-apps-learn-english/
2. C# Guide [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/dotnet/csharp/.
3. Chinnery G. M. Going to the MALL: Mobile Assisted Language Learning /G. M. Chinnery// Language Learning & Technology. 2006, № 10. – P. 9-16.
4. Developer Survey. [Електронний ресурс] — Режим доступу: https://insights.stackoverflow.com/survey/2020#most-popular-technologies
5. Digital in 2020. – Електрон. дан. – Режим доступу: https://wearesocial.com/digital-2020
6. Ionic - Cross-Platform Mobile App Development – [Електронний ресурс]. https://ionicframework.com/
7. JQuery API Documentacion [Електронний ресурс]. – Режим доступу: https://api.jqueryui.com/
8. Kukulska-Hulme A. Mobile language learning now and in the future. / A. Kukulska-Hulme– London: Swedish Net University, 2006. – P. 295-310.
9. Mobile Technology. [Електронний ресурс] – Режим доступу: http://en.wikipedia.org/wiki/Mobile_technology
10. Mobile Technology. [Електронний ресурс] – Режим доступу: http://www.itbusinessedge.com/topics/show.aspx?t=738
11. MySQL [електронний ресурс] // Режим доступу: https://www.mysql.com/ - Назва з екрану
12. Privatbank.ua/ru/know-person-get-money [Електронний ресурс] – Режим доступу до ресурсу: https://privatbank.ua/ru/know-person-get-money/region.

13. Web-технології та Web-дизайн" [Електронний ресурс] – 2015 – Режим доступу до ресурсу:https://dl.sumdu.edu.ua/textbooks/86975/413008/index.html
14. What is Mobile Technology? [Електронний ресурс] – Режим доступу: http://www.strategicgrowthconcepts.com/growth/mobile-technology-facts.html
15. What is Use Case Diagram?- [Електронний ресурс]. – Режим доступу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-usecase-diagram/
16. Yogesh R. Python: Simple though an Important Programming language [Електронний ресурс] / R. Yogesh // International Research Journal of Engineering and Technology, 2019. — Vol. 06. — P. 1856-1858. — Режим доступу: https://www.irjet.net/archives/V6/i2/IRJET-V6I2367.pdf
17. База шахраїв України [Електронний ресурс] – Режим доступу до ресурсу: http://foe.omg.com.ua/foe/
18. Введение в C# [Електронний ресурс] – Режим доступу до ресурсу: https://metanit.com/sharp/tutorial/1.1.php.
19. Види шахрайств [Електронний ресурс] – Режим доступу до ресурсу: https://thepage.ua/finance/moshennichestvo-v-internete-kakim-byvaet-i-kak-sebya-obezopasit.
20. Гнатієнко Г.М., Снитюк В.Є. Експертні технології прийняття рішень: Монографія / К.: ТОВ «Маклаут», 2008. 444 с.
21. Ілляшенко С.М. Сучасні тенденції застосування інтернет-технологій. /С.М. Ілляшенко// [Електронний ресурс] – Режим доступу: http://www.nbuv.gov.ua/portal/Soc_Gum/Mimi/2011_4_2/2_1.pdf
22. Начало работы. Visual Studio [Електронний ресурс] – Режим доступу до ресурсу: https://metanit.com/sharp/tutorial/1.2.php.
23. Організаційна структура проекту (OBS) – [Електронний ресурс]. – Режим доступу: https://studopedia.com.ua/1_243503_organizatsiyna-struktura-proektuoBS.html
24. Попова Ю. В. Сутність і технічні інструменти інтернет-маркетингу. /Ю.В. Попова// [Електронний ресурс] – Режим доступу: http://www.nbuv.gov.ua/portal/soc_gum/Uproz/2011_6/u1106pop.pdf

25. Принципи побудови та етапи проектування баз даних − [Електронний ресурс], — Режим доступу: http://stud.com.ua/35671/informatika/kontseptsiya_baz_danih

26. Cyberpolice [Електронний ресурс] – Режим доступу до ресурсу: https://cyberpolice.gov.ua/.

27. Система керування базами даних [електронний ресурс] // Режим доступу: http://uk.wikipedia.org/wiki/Система_керування_базами_даних – Назва з екрану.

28. Статистика шахрайств в Україні [Електронний ресурс] – Режим доступу до ресурсу: https://finclub.net/analytics/moshenniki-lishayut-deneg-doverchivykh-ukraintsev.html.

29. Типи мобільних додатків. [Електронний ресурс] — Режим доступу: https://smile-ukraine.com/ua/mobile-apps/mobile-apps-types

30. Шахрайство в інтернеті [Електронний ресурс] – Режим доступу до ресурсу: https://minjust.gov.ua/m/yak-ne-stati-jertvoyu-shahraiv-v-interneti-ta-scho-robiti-yakscho-vi-potrapili-u-pastku.

31. Що таке JQuery? [Електронний ресурс]. – Режим доступу: https://phpacademy.kiev.ua/uk/blog/what-is-jquery

ДОДАТКИ

Додаток А. Частина коду програми

AndroidManifest.xml

```
<?xml version=«1.0» encoding=«utf-8»?>
<manifest xmlns:android=«http://schemas.android.com/apk/res/android»
    xmlns:tools=«http://schemas.android.com/tools»
    package=«com.aryan.callsecurity»>

    <uses-permission android:name=«android.permission.PROCESS_OUTGOING_CALLS» />
    <uses-permission android:name=«android.permission.READ_PHONE_STATE» />
    <uses-permission android:name=«android.permission.RECORD_AUDIO» />
    <uses-permission android:name=«android.permission.WRITE_EXTERNAL_STORAGE» />
    <uses-permission android:name=«android.permission.READ_EXTERNAL_STORAGE» />
    <uses-permission android:name=«android.permission.INTERNET» />

    <application
        android:allowBackup=«true»
        android:icon=«@mipmap/ic_launcher»
        android:label=«@string/app_name»
        android:roundIcon=«@mipmap/ic_launcher_round»
        android:supportsRtl=«true»
        android:theme=«@style/Theme.CallSecurity»>
        <activity android:name=«.ImportanceActivity»></activity>
        <activity android:name=«.warning» />
        <activity android:name=«.privacy» />
        <activity android:name=«.help» />
        <activity android:name=«.setting» />
        <activity android:name=«.main_running» />
        <activity android:name=«.feedback» />
```

```
    <activity android:name=«.MainActivity»>
        <intent-filter>
            <action android:name=«android.intent.action.MAIN» />
            <!-- <action android:name=«android.intent.action.NEW_OUTGOING_CALL» /> -->
            <!-- <action android:name=«android.intent.action.PHONE_STATE» /> -->

            <category android:name=«android.intent.category.LAUNCHER» />
        </intent-filter>
    </activity>

    <receiver
        android:name=«.DeviceAdminDemo»
        android:description=«@string/device_description»
        android:label=«@string/device_admin_label»
        android:permission=«android.permission.BIND_DEVICE_ADMIN»>
        <meta-data
            android:name=«android.app.device_admin»
            android:resource=«@xml/my_admin» />

        <intent-filter>
            <action android:name=«android.app.action.DEVICE_ADMIN_ENABLE» />
            <action android:name=«android.app.action.DEVICE_ADMIN_DISABLED» />
            <action android:name=«android.app.action.DEVICE_ADMIN_DISABLE_REQUESTED» />
        </intent-filter>
    </receiver>


    <service android:name=«.Recording_service» />
    <service android:name=«.FileSendtoServer» />
</application>
```

```
</manifest>
```

menu.xml

```
<?xml version=«1.0» encoding=«utf-8»?>
<menu xmlns:android=«http://schemas.android.com/apk/res/android»>
    <group android:checkableBehavior=«single»>
        <item
            android:id=«@+id/nav_home»
            android:icon=«@drawable/home»
            android:title=«Home»>

        </item>
        <item
            android:id=«@+id/nav_setting»
            android:icon=«@drawable/setting»
            android:title=«Setting»>

        </item>

    </group>
    <item android:title=«Information»>
        <menu>
            <group android:checkableBehavior=«single»>
                <item
                    android:id=«@+id/nav_help»
                    android:icon=«@drawable/ic_help_24px»
                    android:title=«Help»>

                </item>
                <item
                    android:id=«@+id/privacy»
```

```
                android:icon=«@drawable/privacy»
                android:title=«@string/privacy_feedback»>

            </item>

        </group>
    </menu>
</item>
<item android:title=«Promote»>
    <menu>
    <group android:checkableBehavior=«single»>
        <item
            android:id=«@+id/rate_us»
            android:icon=«@drawable/star_rate_24px»
            android:title=«Rate us»

            />

        <item
            android:id=«@+id/share»
            android:icon=«@drawable/share»
            android:title=«Share» />
    </group>


</menu>
</item>

</menu>
```

MainActivity.java

```
package com.aryan.callsecurity;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.ActionBarDrawerToggle;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.core.app.ActivityCompat;
import androidx.core.app.NotificationCompat;
import androidx.core.app.NotificationManagerCompat;
import androidx.core.content.ContextCompat;
import androidx.core.view.GravityCompat;
import androidx.drawerlayout.widget.DrawerLayout;

import android.Manifest;
import android.app.Activity;
import android.app.NotificationChannel;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.app.ProgressDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.view.Menu;
```

```
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.Toast;

import com.aryan.callsecurity.Retrofit.IUploadAPI;
import com.aryan.callsecurity.Retrofit.RetrofitClient;
import com.aryan.callsecurity.Utils.Common;
import com.aryan.callsecurity.Utils.IUploadcallback;
import com.aryan.callsecurity.Utils.ProgressRequestBody;
import com.google.android.material.navigation.NavigationView;
import com.aryan.callsecurity.Recording_service;


import java.io.File;
import java.net.URISyntaxException;

import okhttp3.MultipartBody;
import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;


public class MainActivity extends AppCompatActivity implements
NavigationView.OnNavigationItemSelectedListener, IUploadcallback {
    // initialize variable
    private static final int PICK_FILE_REQUEST = 1000;
    private static final String CHANNEL_ID = «callnotification100»;
    IUploadAPI mService;
    Button btnupload, btnsend;
    ImageView imageView;
```

```
Uri selectedFileUri;
ProgressDialog dialog;



private IUploadAPI getAPIUpload() {
    return RetrofitClient.getClient().create(IUploadAPI.class);

}

static final int REQUEST_CODE = 123;
DrawerLayout drawerLayout;
NavigationView navigationView;
Toolbar toolbar;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    // notification channel
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        CharSequence name = getString(R.string.channel_name);
        String description = getString(R.string.channel_description);
        int importance = NotificationManager.IMPORTANCE_DEFAULT;
        NotificationChannel channel = new NotificationChannel(CHANNEL_ID, name, importance);
        channel.setDescription(description);
        NotificationManager notificationManager = getSystemService(NotificationManager.class);
        notificationManager.createNotificationChannel(channel);
    }
    //.....phone service
    /*-------Button----------------*/
    Button button = findViewById(R.id.start_operation);
```

```
button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        if (ContextCompat.checkSelfPermission(MainActivity.this,
Manifest.permission.PROCESS_OUTGOING_CALLS)
                + ContextCompat.checkSelfPermission(MainActivity.this,
Manifest.permission.RECORD_AUDIO)
                + ContextCompat.checkSelfPermission(MainActivity.this,
Manifest.permission.READ_EXTERNAL_STORAGE)
                + ContextCompat.checkSelfPermission(MainActivity.this,
Manifest.permission.READ_PHONE_STATE)
                + ContextCompat.checkSelfPermission(MainActivity.this,
Manifest.permission.WRITE_EXTERNAL_STORAGE)

                != PackageManager.PERMISSION_GRANTED) {
            // when permission is not granted.
            if (ActivityCompat.shouldShowRequestPermissionRationale(MainActivity.this,
Manifest.permission.PROCESS_OUTGOING_CALLS)
                    || ActivityCompat.shouldShowRequestPermissionRationale(MainActivity.this,
Manifest.permission.RECORD_AUDIO)
                    || ActivityCompat.shouldShowRequestPermissionRationale(MainActivity.this,
Manifest.permission.WRITE_EXTERNAL_STORAGE)
                    || ActivityCompat.shouldShowRequestPermissionRationale(MainActivity.this,
Manifest.permission.READ_EXTERNAL_STORAGE)
                    || ActivityCompat.shouldShowRequestPermissionRationale(MainActivity.this,
Manifest.permission.READ_PHONE_STATE)) {
                // create alert dialog
                AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
                builder.setTitle(«Please grant these permission»);
                builder.setMessage(«All the above permission is mandatory for this application.»);
                builder.setPositiveButton(«OK», new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialogInterface, int i) {
                        ActivityCompat.requestPermissions(MainActivity.this, new String[]{
```

```
                        Manifest.permission.PROCESS_OUTGOING_CALLS,
                        Manifest.permission.READ_EXTERNAL_STORAGE,
                        Manifest.permission.READ_PHONE_STATE,
                        Manifest.permission.WRITE_EXTERNAL_STORAGE,
                        Manifest.permission.RECORD_AUDIO
                    }, REQUEST_CODE);
                }
            });
            builder.setNegativeButton(«Cancel», null);
            AlertDialog alertDialog = builder.create();
            alertDialog.show();

        } else {
            ActivityCompat.requestPermissions(MainActivity.this, new String[]{
                    Manifest.permission.PROCESS_OUTGOING_CALLS,
                    Manifest.permission.READ_EXTERNAL_STORAGE,
                    Manifest.permission.READ_PHONE_STATE,
                    Manifest.permission.WRITE_EXTERNAL_STORAGE,
                    Manifest.permission.RECORD_AUDIO
            }, REQUEST_CODE);
        }
    } else {
        // when permission are already granted.
        operation_on();
        //Toast.makeText(MainActivity.this, «permission is granted, operation will be perform.»,
Toast.LENGTH_SHORT).show();
    }
  }
});
// for uploading file.
    mService = getAPIUpload();
    btnupload = findViewById(R.id.upload);
```

```
        btnsend = findViewById(R.id.sendtoserv);
        //***automatics loading..

        btnupload.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                chooseFile();
//                uploadFile();
//                Toast.makeText(MainActivity.this, «error...», Toast.LENGTH_SHORT).show();
            }
        });

        btnsend.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                uploadFile();
//                Toast.makeText(MainActivity.this, «error...», Toast.LENGTH_SHORT).show();
            }
        });

        // navigation bar
        /*------------- hook----------------------*/
        drawerLayout = findViewById(R.id.drawer_layout);
        navigationView = findViewById(R.id.nav_view);
        toolbar = findViewById(R.id.toolbar);

        /*-------tool bar-------*/
        setSupportActionBar(toolbar);
```

```
        /*-----------navigation driver menu------*/
        navigationView.bringToFront();
        ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(this, drawerLayout, toolbar,
R.string.navigation_drawer_open, R.string.navigation_drawer_close);
        drawerLayout.addDrawerListener(toggle);
        toggle.syncState();


        navigationView.setNavigationItemSelectedListener(this);
        navigationView.setCheckedItem(R.id.nav_home);
    }



    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull
int[] grantResults) {
        if (requestCode == REQUEST_CODE) {
            if ((grantResults.length > 0) && (grantResults[0] + grantResults[1] + grantResults[2] +
grantResults[3] + grantResults[4] == PackageManager.PERMISSION_GRANTED)) {
                // permission is granted
                Toast.makeText(getApplicationContext(), «permission Granted..»,
Toast.LENGTH_SHORT).show();
            } else {
                Toast.makeText(getApplicationContext(), «permission not granted..»,
Toast.LENGTH_SHORT).show();
            }
        }
    }


    @Override
    public void onBackPressed() {
        if (drawerLayout.isDrawerOpen(GravityCompat.START)) {
            drawerLayout.closeDrawer(GravityCompat.START);
        } else {
```

```
        super.onBackPressed();
    }

}

@Override
public boolean onNavigationItemSelected(@NonNull MenuItem menuItem) {
    switch (menuItem.getItemId()) {
        case R.id.nav_home:
            break;
        case R.id.nav_setting:
            Intent intent = new Intent(MainActivity.this, setting.class);
            startActivity(intent);
            break;

        case R.id.nav_help:
            Intent intent1 = new Intent(MainActivity.this, help.class);
            startActivity(intent1);
            break;
        case R.id.privacy:
            Intent intent2 = new Intent(MainActivity.this, privacy.class);
            startActivity(intent2);
            break;
        case R.id.rate_us:
            Toast.makeText(this, «please rate-us, to appreciate.», Toast.LENGTH_SHORT).show();
            break;
        case R.id.share:
            Toast.makeText(this, «share for other security.», Toast.LENGTH_SHORT).show();
            Intent sharingIntent = new Intent(Intent.ACTION_SEND);
            sharingIntent.setType(«text/plain»);
            String shareBody = «here is the shere content body.»;
```

```
            sharingIntent.putExtra(Intent.EXTRA_SUBJECT, «subject here»);
            sharingIntent.putExtra(Intent.EXTRA_TEXT, shareBody);
            startActivity(Intent.createChooser(sharingIntent, «share via»));
            break;
        }
        drawerLayout.closeDrawer(GravityCompat.START);
        return true;
    }


    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    // navigation bar
    // start button
    public void operation_on() {
        // start of service
        Intent intent4 = new Intent(this, main_running.class);
        startActivity(intent4);
        // notification
        Intent intenttabactionfornotification = new Intent(this, main_running.class);
        intenttabactionfornotification.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
        PendingIntent pendingIntent = PendingIntent.getActivity(this, 0, intenttabactionfornotification, 0);

        NotificationCompat.Builder builder = new NotificationCompat.Builder(this, CHANNEL_ID)
            .setSmallIcon(R.drawable.insurance2)
            .setContentTitle(«CallSecurity»)
            .setContentText(«Don’t worry we are with you.»)
```

```
            .setPriority(NotificationCompat.PRIORITY_DEFAULT)
            .setContentIntent(pendingIntent)
            .setAutoCancel(true);
    NotificationManagerCompat notificationmanager = NotificationManagerCompat.from(this);
    notificationmanager.notify(1, builder.build());
    //end notification

    Intent intent3 = new Intent(MainActivity.this, Recording_service.class);
    startService(intent3);
    Toast.makeText(this, «Fraud call detection service has started.», Toast.LENGTH_LONG).show();
}


// menu bar
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();
    if (id == R.id.feedback) {
        //action
        Toast.makeText(this, «this feedback is opened.», Toast.LENGTH_SHORT).show();
        return true;
    }
    if (id == R.id.about_us) {
        // action
        Toast.makeText(this, «this is about us opened», Toast.LENGTH_SHORT).show();
        return true;
    }
    return true;
}  //menu
//file upload
```

```
    @Override
    protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (resultCode == Activity.RESULT_OK) {
            if (requestCode == PICK_FILE_REQUEST) {
                if (data != null) {
                    selectedFileUri = data.getData();
                    if (selectedFileUri != null && !selectedFileUri.getPath().isEmpty())
//                      imageView.setImageURI(selectedFileUri);
                        Toast.makeText(this, selectedFileUri.toString(), Toast.LENGTH_SHORT).show();
//                      Toast.makeText(this, «file selected.», Toast.LENGTH_SHORT).show();
                    else
                        Toast.makeText(this, «file not found.», Toast.LENGTH_SHORT).show();
                }
            }
        }

    private void uploadFile() {

        if (selectedFileUri != null) {
            dialog = new ProgressDialog(this);
            dialog.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
            dialog.setMessage(«uploading.....»);
            dialog.setIndeterminate(false);
            dialog.setMax(100);
            dialog.setCancelable(false);
            dialog.show();
            File file = null;
            try {
                file = new File(Common.getFilePath(this, selectedFileUri));
```

```
        } catch (URISyntaxException e) {
            e.printStackTrace();
        }
        if (file != null) {
//              final ProgressRequestBody requestBody = new ProgressRequestBody(file,this);
            final ProgressRequestBody requestBody = new ProgressRequestBody(file, this);
            final MultipartBody.Part body = MultipartBody.Part.createFormData(«data», file.getName(), requestBody);

            new Thread(new Runnable() {
                @Override
                public void run() {
                    mService.UploadFile(body).enqueue(new Callback<String>() {
                        @Override
                        public void onResponse(Call<String> call, Response<String> response) {
                            dialog.dismiss();
                            if((response.body()).equals(«fraud»)) {
//
                                Intent fullScreenIntent = new Intent(MainActivity.this,ImportanceActivity.class);
                                PendingIntent fullscreenPending = PendingIntent.getActivity(MainActivity.this,0,fullScreenIntent,PendingIntent.FLAG_UPDATE_CURRENT);

                                NotificationCompat.Builder builder= new NotificationCompat.Builder(MainActivity.this,CHANNEL_ID)
                                .setSmallIcon(R.drawable.insurance2)
                                .setContentTitle(«alert»)
                                .setContentText(«be careful this is fraud call.»)
                                .setPriority(NotificationCompat.PRIORITY_DEFAULT)
                                .setFullScreenIntent(fullscreenPending,true);

                                NotificationManagerCompat notificationmanager = NotificationManagerCompat.from(MainActivity.this);
```

```
                    notificationmanager.notify(1, builder.build());
                    Toast.makeText(MainActivity.this, response.body().toString(),
Toast.LENGTH_LONG).show();
                    }
                    else if(response.body().equals(«normal»))
                    {
                       Intent fullScreenIntent = new Intent(MainActivity.this,ImportanceActivity.class);
                       PendingIntent fullscreenPending =
PendingIntent.getActivity(MainActivity.this,0,fullScreenIntent,PendingIntent.FLAG_UPDATE_CURRE
NT);

                       NotificationCompat.Builder builder= new
NotificationCompat.Builder(MainActivity.this,CHANNEL_ID)
                            .setSmallIcon(R.drawable.insurance2)
                            .setContentTitle(«alert»)
                            .setContentText(«be continue, this is normal call.»)
                            .setPriority(NotificationCompat.PRIORITY_DEFAULT)
                            .setFullScreenIntent(fullscreenPending,true);

                       NotificationManagerCompat notificationmanager =
NotificationManagerCompat.from(MainActivity.this);
                       notificationmanager.notify(1, builder.build());
                       Toast.makeText(MainActivity.this, «normal», Toast.LENGTH_SHORT).show();
                    }
                 }
                 @Override
                 public void onFailure(Call<String> call, Throwable t) {
                    dialog.dismiss();
                    Toast.makeText(MainActivity.this, t.getMessage(),
Toast.LENGTH_SHORT).show();
                 }
              });
           }
        }).start();
```

```
        } else {
            Toast.makeText(this, «Can’t upload», Toast.LENGTH_SHORT).show();
        }

    }
}

private void chooseFile() {
    Intent intent = new Intent(Intent.ACTION_GET_CONTENT);
    intent.addCategory(Intent.CATEGORY_OPENABLE);
    intent.setType(«*/*»);
    startActivityForResult(intent, PICK_FILE_REQUEST);
}

@Override
public void onProgressUpdate(int percent) {

}


}
```