МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ПРИРОДОКОРИСТУВАННЯ
ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

першого (бакалаврського) рівня вищої освіти

на тему:

«ПРОЕКТУВАННЯ ПІДСИСТЕМИ ІНФОРМАЦІЙНОЇ БЕЗПЕКИ СИСТЕМИ УПРАВЛІННЯ РОЗУМНИМ БУДИНКОМ»

Виконав: здобувач 4-ого курсу
спеціальності 126 «Інформаційні системи та технології»

_____ Сітніков М. В.
(прізвище та ініціали)

Керівник: _____ Пташник В. В.
(прізвище та ініціали)

Рецензент: _____ Сиротюк С. В.
(прізвище та ініціали)

ДУБЛЯНИ-2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ПРИРОДОКОРИСТУВАННЯ
ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Перший (бакалаврський) рівень вищої освіти
Спеціальність 126 «Інформаційні системи та технології»

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____
(підпис)
д.т.н., професор, Тригуба А. М.
(вч. звання, прізвище, ініціали)
“____” _____202 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Сітніков Максим Віталійович
(прізвище, ім’я, по батькові)

1. Тема роботи *«Проектування підсистеми інформаційної безпеки системи управління розумним будинком»*
керівник роботи *к. т. н., доцент., Пташник В. В.*
(наук.ступінь, вч. звання, прізвище, ініціали)
затверджені наказом Львівського НУП *від 30.12.2022 року №453/к-с*
2. Строк подання студентом роботи *15 червня 2023 року*
3. Вихідні дані до роботи: *характеристика комерційних систем безпеки та управління «розумним будинком»; паспорти та технічна документація до комерційних мікропроцесорних систем домашньої автоматизації Ajax, Xiaomi, Fibaro, Easy Smart Box, Inwion, SenseHome тощо; науково-технічна і довідкова література.*
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
Вступ
1. Аналіз загроз інформаційної безпеки розумного будинку
2. Засоби розробки елементів інформаційної безпеки розумного будинку
3. Реалізація підсистеми безпеки розумного будинку
4. Охорона праці та безпека в надзвичайних ситуаціях
Висновки
Список використаних джерел
5. Перелік графічного матеріалу
Графічний матеріал подається у вигляді презентації

6. Консультанти розділів

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата		Відмітка про виконання
		завдання видав	завдання прийняв	
1, 2, 3	*Пташник В. В., к.т.н., доцент*			
4	*Городецький І. М., к.т.н., доцент*			

7. Дата видачі завдання *30 грудня 2022 р.*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Відмітка про виконання
1	*Складання інженерної характеристики об'єкту проектування*	*01.01.2023 – 31.01.2023*	
2	*Аналіз промислових стандартів захисту інформації та інформаційної безпеки розумного будинку*	*01.02.2023 – 28.02.2023*	
3	*Проектування інформаційної системи розумний будинок з розробленням системи інформаційної безпеки*	*01.03.2023 – 15.04.2023*	
4	*Розгляд питань з охорони праці та безпеки у надзвичайних ситуаціях*	*16.04.2023 – 30.04.2023*	
5	*Завершення оформлення розрахунково-пояснювальної записки та презентаційного матеріалу*	*01.05.2023 – 31.05.2023*	
6	*Завершення роботи в цілому. Підготовка до захисту кваліфікаційної роботи*	*01.06.2023 – 10.06.2023*	

Здобувач _____ *Сітніков М. В.*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Пташник В. В.*
(підпис) (прізвище та ініціали)

УДК 681.521 / 681.518

Проектування підсистеми інформаційної безпеки системи управління розумним будинком. Сітніков М. І. Кафедра інформаційних технологій – Дубляни, Львівський національний університет природокористування, 2023.

Кваліфікаційна робота: 67 сторінок текстової частини, 10 рисунків, 19 джерел літератури, 8 додатків.

Мета кваліфікаційної роботи полягає у визначенні основних загроз інформаційній безпеці розумного будинку та розробенні підсистеми інформаційної безпеки системи управління розумним будинком.

Об'єктом дослідження є програмні алгоритми та технічні засоби, необхідні для функціонування підсистеми інформаційної безпеки системи управління розумним будинком.

Предмет дослідження вивчає особливості інформаційної безпеки системи управління розумним будинком.

У роботі проаналізовано предметну область, визначено основні загрози інфомаційній безпеці розумного будинку, розглянуто існуючі аналоги та визначено їх функціональні можливості. Обґрунтовано особливості реалізації та практичного використання елементів системи безпеки розумного будинку та керування розумними приладами. Запропоновано такі елементи системи безпеки розумного будинку як двофакторна автентифікація та рівні доступу. Здійснено проектування системи безпеки розумного будинку у відповідності до існуючих методик. Здійснено аналіз травматичних ситуацій при виконанні різних робіт у сфері використанням комп'ютерної техніки, викладено питання охорони праці.

Ключові слова: розумний будинок, інформаційна безпека, двофакторна автентифікація, інтернет речей, мобільний додаток, Telegrame Bot.

ЗМІСТ

ВСТУП

Концепція систем безпеки розумних будинків зазнала істотних змін та покращень за останні роки. Порівняно із традиційними системами автоматики з обмеженою сферою відповідальності, системи безпеки розумного будинку розумніші ніж будь-коли раніше, стежать за вашим будинком коли ви спите, коли не вдома та коли не підозрюєте про загрозу. Окрім кращого захисту у надзвичайних ситуаціях, вони підтримують нові функції, засоби та розумні технології, які старі системи не можуть запропонувати.

Метою дипломної роботи є проектування та розробка елементів системи безпеки розумного будинку, таких як, двофакторна автентифікація та рівні доступу.

Тема дипломної роботи є актуальною, оскільки системи розумних будинків зараз є популярними та розвиваються, проте їх безпека ще не доведена до ідеалу і має чимало недоліків.

Для виконання кваліфікаційної роботи поставлено наступні завдання:

1. Дослідити інформаційну складову розумних будинків та систем безпеки розумних будинків.
2. Провести порівняльний аналіз існуючих систем керування розумним будинком.
3. Дослідити методи двофакторної автентифікації та обрати один з них для реалізації.
4. Спроектувати мобільний додаток та реалізувати в ньому обраний метод двофакторної автентифікації.
5. Описати програмну реалізацію додатку.
6. Протестувати розроблений додаток.

РОЗДІЛ 1
АНАЛІЗ ЗАГРОЗ ІНФОРМАЦІЙНОЇ БЕЗПЕКИ РОЗУМНОГО БУДИНКУ

1.1 Розумний будинок як інформаційна система

Технологія розумного будинку є частиною технологічного руху, який називається «Інтернет речей» (IoT, з англ. – Internet of Things). IoT базується на ідеї, що пристрої, машини і будівлі коли-небудь будуть об'єднані у бездротову мережу і будуть збирати та передавати дані про навколишнє середовище. Ця концепція спирається на повсюдні обчислення, практику додавання мікропроцесорів та підключення до Інтернету основних пристроїв та приладів. Повсюдні обчислення спрямовані та те, щоб зробити обробку даних постійною фоновою діяльністю, що вбудовується у повсякденне життя.

Технологія розумного будинку, перебуваючи ще в зародковому стані, має на меті підвищити безпеку та зручність для власників будинків. Наприклад, будинок можна налаштувати таким чином, щоб датчики руху виявляючи активність вранці відчиняли віконні штори, вмикали кавоварку, і включали опалення. Також автоматизація будинків починає допомагати інвалідам і людям похилого віку. Такі функції, як активація голосом, мобільні додатки і сенсорні дисплеї полегшують власникам процес виконання буденних справ та самостійного життя.

Споживачі не так часто застосовують технології розумного будинку з кількох причин. Перша – це страх, що виробники будуть використовувати такі прилади щоб збирати і продавати їх особисті дані. Інша – це страх, що безпека таких систем може бути зламана і призведе до таких проблем безпеки, як крадіжка зі зломом. Вподобання багатьох користувачів, динаміка енергоспоживання та швидкі зміни в графіку що є в сімейному житті,

створюють додаткові проблеми для систем розумного будинку. З дебютом переносних цифрових пристроїв, таких як розумні годинники, деякі компанії експерементують з підключенням їх до розумного будинку. Переносні пристрої які можуть зчитувати біометричні дані можуть бути використані щоб підтвердити особу власника будинку для підвищення безпеки і персоналізації.

Технологія розумних будинків охоплює широкий спектр сучасних домашніх пристроїв які можуть взаємодіяти між собою та з мережею Інтернет. Це надає власнику можливість планувати прості щоденні завдання та здійснювати віддалений контроль за їх виконанням. Створені для зручності, розумні будинки також дотримуються ідеї покращувати самостійне життя людей похилого віку та людей з обмеженими можливостями.

Розумні будинки, або автоматизовані будинки, є будинками в яких домашнє обладнання, елементи захисту будинку, та інші прилади з'єднані в єдину мережу. Попит на автоматизацію помешкання постійно зростає у всьому світі, адже це дозволяє власникам слідкувати за своїм помешканням віддалено, автоматизовувати базові домашні процеси, і заощаджувати гроші в довгостроковій перспективі на оплаті комунальних послуг та інших витратах. Проектування розумного дому, як правило включає придбання концентратора, і різних розумних пристроїв, побутових приладів, штекерів та датчиків, які можна підключити через цей концентратор до домашньої мережі. Налаштування розумного будинку вимагає залучення кваліфікованих спеціалістів і коштує досить дорого. Адля його ефективного функціонування необхідна потужна мережева інфраструктура.

Різні елементи розумного будинку використовують різні засоби підключення до загальної мережі. Пристрої з підтримкою Wi-Fi використовують радіосигнали надвисокої частоти для зв'язку пристроїв з інтернетом через маршрутизатори, модеми та подовжувачі діапазону. Інші розумні пристрої використовують Bluetooth з'єднання, він також використовує надвисокочастотний радіосигнали але працює на значно меншій відстані, що дозволяє суттєво знизити рівень енергоспоживання таких

пристроїв. Відзначимо, що різні пристрої з Bluetooth керуванням можна підключати до мережі розумного будинку для віддаленого керування, але вони не можуть бути керовані поза примфщенням. А Wi-Fi мережа надає власнику можливість керувати усією системою через інтернет.

Розширені варіанти мереж розумного будинку включають так звані сітчасті мережі. У таких мережах, кожен вузол може отримувати, повторювати та передавати сигнали усім іншим вузлам у мережі. Сітчаста мережа підтримується як у бездротовому, так і у дротовому виконанні. Частіше зустрічається бездротове виконання мережі, яке використовує Bluetooth або Wi-Fi для обміну високочастотними радіо сигналами між окремими вузлами. Існують мережі адаптовані для роботи з двосмуговим з'єднанням, як з дротовим, так і з бездротовим зв'язком між вузлами.

Мережі з подвійною сіткою спроектовані, щоб уникнути електронних перешкод від зовнішніх сигналів, що надходять від таких джерел, як мікрохвильові печі та телевізори. У мережі з подвійною сіткою перешкоди від ліній електропередач, що потрапляють додому, або до домашньої бездротової мережі не заважатимуть сигналу надходити до мережевого пристрою. Сітчасті мережі також працюють з пристроями різних виробників, тоді як інші типи мереж можуть бути несумісними з деякими пристроями власника. Приклад розгортання розумного будинку наведено на рис. 1.1.

До мережі розумного будинку можуть бути під'єднані різноманітні типи пристроїв. Розумні перемикачі світла та регулятори використовують для контролю існуючих елеменів освітлення, увімкнення та вимкнення світла або під'єднання їх до системи планування для активації за розкладом. Власник розумного будинку може програмувати роботу системи освітлення, наприклад на цикл сходу/заходу сонця. Використання розумних розеток також дозволяє контролювати наявні електропристрої.

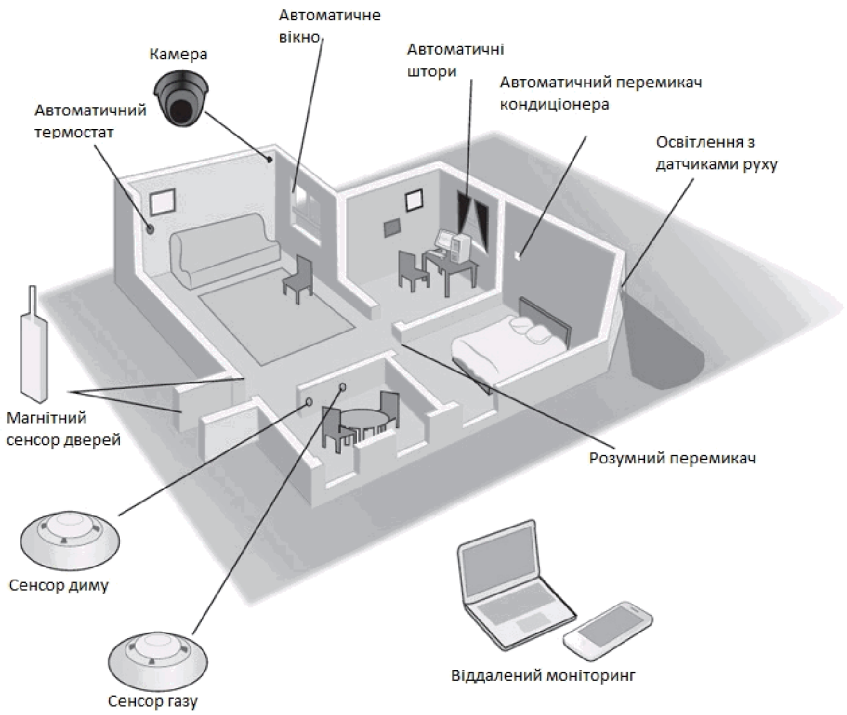


Рисунок 1.1 – Функціональні елементи розумного будинку

Існує ряд пристроїв та технологій, які дозволяють покращити ефективність використання можливостей розумного будинку, дозволяючи людям налаштовувати своє середовище в будь-який час та в будь-якому місці. Деякі пристрої, наприклад, програмовані термостати, можна запрограмувати так, щоб вони запускалися лише в визначений час, оптимальний для зменшення використання ресурсів. Також використовують пристрої, які синхронізуються зі смартфоном та дозволяють власнику будинку контролювати охорону, освітлення та багато інших пристроїв віддалено.

1.2 Системи безпеки розумного будинку

У еру розвитку технології та підвищення інтересу до Інтернету мережева безпека стала основною проблемою для компаній у всьому світі. Широка доступність інформації та інструментів, необхідних для несанкціононваного

проникнення в корпоративні мережі посилює цю стурбованість. Адміністратори мережі часто витрачають більше зусиль на захист своїх мереж, ніж на її налаштування та адміністрування. Інструменти, які перевіряють системні вразливості, такі як інструмент адміністратора безпеки для аналізу мереж та деякі пакети для виявлення вторгнення, допомагають у цих зусиллях, але ці інструменти лише вказують на слабкі місця та можуть не надати засобів для захисту мереж від усіх можливих атак. Таким чином адміністратор мережі повинен намагатися постійно бути в курсі актуальних проблем безпеки.

Конфіденційна інформація може знаходитись у двох станах в мережі. Вона може знаходитись на фізичних носіях інформації, таких як жорсткий диск або пам'ять, або може знаходитися «в дорозі» крізь фізичну мережеву лінію зв'язку у вигляді пакетів. Кожен з можливих станів надає безліч можливостей для атак як з боку користувачів у внутрішній мережі, так і користувачів в Інтернеті. З точки зору проблем безпеки розумного будинку вирішальне значення відіграють проблеми безпеки мережі. Нижче наведено п'ять найпоширеніших методів атаки, які надають можливість скомпрометувати інформацію у мережі розумного будинку:

- сніффери мережевих пакетів;
- атаки на паролі;
- підробка IP-адрес;
- транскордонні атаки;
- поширення конфіденційної внутрішньої інформації до зовнішніх джерел.

Захищаючи інформацію від цих атак забезпечується захист від крадіжки, знищення, спотворення та введення хибної інформації, яка може завдати непоправної шкоди чутливим та конфіденційним даним.

Звичайні системи безпеки забезпечують захист власників будинків та їх власності від зловмисників. Однак розумна система безпеки будинку пропонує значно більше переваг. Технологія домашньої автоматизації повідомляє власників будинків про будь-які проблеми, щоб вони могли провести

розслідування. Програми штучного інтелекту відстежують звички домовласника та іншу важливу інформацію, і за необхідності повідомляють обслуговуючий персонал.

Сигналізація, детектори руху та камери надають інформацію розумній домашній системі безпеки, дозволяючи їй визначити, чи є особа резидентом, відвідувачем або тим, кому вхід заборонено. Детектори руху викликають сповіщення, даючи програмі штучного інтелекту зрозуміти, що є хтось чи щось для оцінки. Програмне забезпечення для розпізнавання обличчя та коди безпеки дозволяють системі безпеки пропускати мешканців додому, тоді як на основі заздалегідь запрограмованої інформації обмежують доступ інших осіб.

Кожен раз, коли розумна система охорони будинку виявляє когось невідомого, вона може надати відео про відвідувача власнику будинку. Запрошеним відвідувачам можна дати доступ і пустити в будинок віддалено. Небажаних відвідувачів можна ігнорувати, а у разі, якщо певні особи намагаються увірватися, система викличе поліцію.

Розумна система охорони будинку пропонує набагато більше захисту, ніж типова пожежна сигналізація. Цей тип системи перевіряє рівень окису вуглецю, а також відстежує наявність ознак пожежі та контролює всі ділянки будинку. У разі пожежі розумна система безпеки будинку може попередити власника будинку та повідомити аварійні служби. Програми штучного інтелекту навіть можуть точно визначити місце пожежі та надати цю інформацію працівникам пожежної служби, коли вони реагують на виклик системи.

Зловмисники та пожежі – не єдина небезпека в домі. Розумна система безпеки будинку також захищає мешканців від несподіваних проблем зі здоров'ям. Використовуючи ті самі камери та детектори руху, які захищають зовнішню частину будинку, розумні будинки можуть дізнатися про звички та звичні рухи мешканців. Коли мешканець робить щось несподіване і не відновлює звичайну діяльність, розумний будинок може попередити членів

сім'ї або аварійні служби. Цей аспект розумного будинку особливо корисний для людей похилого віку або тих, хто має слабке здоров'я.

Багато розумних домашніх пристроїв забезпечують технологію домашньої автоматизації, але розумна система безпеки будинку пропонує багато переваг, які можуть забезпечити безпеку власника будинку. Далі ми розглянемо інструменти, пов'язані із системою Smart Home Security.

Ця схема автентифікації має дві фази: фазу реєстрації користувача та фазу автентифікації користувача. По-перше, уповноважені користувачі повинні зареєструватися в системі автентифікації, вказавши свій логін та пароль. На другому етапі, тобто на етапі автентифікації користувача, система перевіряє легітимність користувачів.

На етапі реєстрації користувача системний адміністратор отримує «тренувальні» схеми з іменами користувачів та паролями для навчання нейронної мережі. Процес реєстрації відбувається таким чином:

- кожен користувач вибирає належне ім'я користувача та пароль і передає їх системному адміністратору.

- система застосовує односторонню хеш-функцію до імені користувача та пароля, і результат використовується як шаблон навчання. Отже, шаблон навчання складається з хешованого імені користувача як входу нейронної мережі та відповідного хешованого пароля як бажаного виводу нейронної мережі.

- перед тим, як тренувати нейронну мережу, системі потрібно нормалізувати коди ASCII символів навчальних моделей.

- системний адміністратор використовує ці схеми навчання для навчання мережі RBF. Після навчального процесу системний адміністратор зберігає ваги мережі RBF у системі.

На етапі автентифікації користувача система автентифікації використовує навчену мережу RBF і застосовує ту саму односторонню хеш-функцію для автентифікації законності користувачів. Процес автентифікації описується наступним чином:

- система застосовує ту саму хеш-функцію до введеного імені користувача та пароля.

- система витягує результат через навчену нейронну мережу.

- система порівнює вихідні дані мережі RBF з хешованим паролем. Якщо результати рівні, користувач визнається авторизованим користувачем. В іншому випадку користувач відхиляється як неавторизований користувач.

Середовища розумного будинку, як правило, оснащені різними типами датчиків і пристроїв відстеження для контекстного надання послуг. З одного боку, люди хочуть скористатися комфортом та доданою вартістю персоналізованих контекстних служб, але конфіденційність та відстежуваність стає серйозною проблемою, з іншого боку. Виникає запитання, як отримати довіру до невід'ємних служб у потенційно ворожому середовищі? Як можна гарантувати, що врешті-решт усі конфіденційні дані будуть видалені або безпечно збережені? Концепція Sentry @ HOME, як частина орієнтованої на користувача системи конфіденційності, вирішує ці проблеми.

Sentry @ HOME створений, щоб стати невід’ємною частиною домашнього середовища користувача; безпроблемно вбудований в інфраструктуру програмного забезпечення розумного будинку.

Захисний механізм Sentry @ HOME є багатошаровим підходом до захисту DDoS-атаки, спричиненої спамом. Цей підхід був впроваджений у поштовій системі та контролював результати. Результат показує, що даний підхід є дуже ефективним. Підхід має шість шарів. Цей підхід є поєднанням тонкої настройки вихідних фільтрів, фільтрів вмісту, політики моніторингу мережі, загальної політики електронної пошти, навчання користувачів та своєчасних логічних рішень адміністратора мережі. Точне налаштування вихідних фільтрів відхиляє вхідні з'єднання перед доставкою спаму. Фільтри вмісту аналізують вміст листів та блокують вхідні небажані повідомлення. Підхід мережевого моніторингу забезпечує загальне рішення для ідентифікації атак до атаки, а також під час атаки. Розумні будинки повинні проінформувати

користувача про можливі сценарії атак та способи реагування на них. Логічні рішення власника відіграють важливу роль протягом періоду атаки і навіть після атаки. Поєднання цих шарів забезпечує найкращу методологію для зупинки DDoS-атак, реалізованих за допомогою спаму.

1.3 Існуючі системи «Розумний будинок»

1.3.1 Ajax

Продукція компанії Ajax розробляється та виробляється в Україні з 2011 року, відповідно, за завмовчуванням підтримується українська мова інтерфейсу. На сьогоднішній день продукція компанії представлена на ринку понад 120 країн світу.

Дана система автоматизації будинку одночасно вирішує два важливі завдання:

- забезпечує комфорт і зручність в керуванні життєзабезпеченням приміщення;
- гарантує безпеку житла в повній мірі, контролюючи межі об'єкта на предмет взлому, а також електричну, пожежну, газову та інші можливі загрози для дому.

Обладнання Ajax працює з використанням надійно зашифрованого і захищеного радіозв'язку Jeweller власної розробки, має повну автономність від електромережі завдяки резервному джерелу живлення – хабу, характеризується стильним дизайном усіх своїх пристроїв.

1.3.2 BroadLink

Системи BroadLink виробляються в Китай, офіційно імпортуються до України і підтримують україномовний інтерфейс.

Обладнання BroadLink є комплектом сучасних цифрових пристроїв, створених для раціонального керування побутовою технікою, а також освітлювальною, енергетичною, охоронною та іншими системами в будинку. Кожен елемент такого комплексу може працювати як самостійно, так і взаємодіяти з іншими системами.

1.3.3 Fibaro

Країна виробник – Польща (розробка та реєстрація бренду – США). Важко знайти український інтерфейс.

Розумний дім Fibaro відноситься до професійного обладнання для автоматизації та забезпечення безпеки дому з широким функціоналом. Однак, на відміну від багатьох аналогічних систем, потребує встановлення та налаштування кваліфікованими спеціалістами.

1.3.4 Orvibo

Продукція Orvibo виробляється в Китаї. Відсутність української мови інтерфейсу за замовчуванням (проте є англійська) дещо ускладнює встановлення та налаштування обладнання для пересічного користувача.

Orvibo – це недорогий комплект простого в експлуатації обладнання, головне завдання якого полягає у забезпеченні безпеки дому. І лише опосередковано установка може слугувати базою для організації повноцінної системи Розумний будинок.

1.3.5 Xiaomi

Виробничі потужності компанії Xiaomi розміщено у Китаї. Відсутність української мови інтерфейсу за замовчуванням, тільки англійська та китайська, що додає складності процесу налаштування та встановлення.

Розумний дім від компанії Xiaomi належить до середнього цінового сегменту та дозволяє зробити керування різноманітними пристроями і побутовою технікою вдома максимально простим та зручним. Дана компанія займає значний сегмент даного ринку завдяки широкому спектру надавних послуг. Водночас питання безпеки залишаються одними з пріоритетних, враховуючи значну кількість підробок продукції даного бренду.

1.4 Двофакторна автентифікація

Двофакторна автентифікація забезпечує значне покращення безпеки порівняно зі звичайною комбінацією імені користувача та паролю. Використання двох факторів автентифікації передбачає використання чогось, що знає користувач та чогось, що перебуває в його розпорядженні. В світі однофакторної автентифікації пароль був «тим, що ви знаєте». Додатковий фактор – «те чим володіє користувач», є ключовим компонентом двофакторної автентифікації.

Другим фактором може бути маркер, смарт-карта, PIN/TAN, sms-повідомлення, повідомлення на електронну пошту або біометрія.

Маркер на маленькому дисплеї відображає набір цифр. Зазвичай, набір цифр змінюється кожної хвилини. Цей номер з'єднується з користувацьким паролем або пін кодом, щоб створити код доступу. Відразу можна помітити підвищення рівня безпеки за рахунок впровадження цієї форми двофакторного захисту. Оскільки частина коду доступу постійно змінюється то загроза прослуховування або перехоплення паролю різко падає. Фізична особа може записати свій пін-код, щоб запам'ятати його, однак без комбінованого номера з брелока, пароль стає непотрібним.

Смарт-карти використовуються у комбінації зі зчитувачем смарт-карт. Користувач вставляє карту і карта відправляє закодоване повідомлення на сайт, або зчитувач відображає унікальний код, який буде вводити користучач.

PIN/TAN автентифікація означає персональний ідентифікаційний номер. Користувач отримує аркуш з таблицею яка містить багато різних номерів. Кожен номер використовується лише один раз, для підтвердження ідентифікації. Метод PIN/TAN став популярним у Європі.

Біометрична ідентифікація використовує біологічні аспекти користувача, такі як відбитки пальців або скан сітківки шоб пройти автентифікацію. Інші методи біометричного розпізнання включають електронні підписи та динаміку натискання клавіш, що не лише фіксує остаточний підпис або слово, але і те, як підпис було набрано.

Ще однією перевагою використання двофактроної автентифікації є зменшення навантаження на внутрішню або контрактну службу підтримки. За даними інституту довідкової служби США, сімнадцять відсотків усіх дзвінків надходять від осіб, які втратили або загубили свої паролі. Люди будуть рідше запам’ятовувати статичний контактний номер, і такі типи дзвінків до довідкової служби впадуть.

Використання маркерів, смарт-карт і брелоків є основним фактором в двофакторній автентифікації. Проте, із розвитком технологій біометрія відіграє все більшу роль у забезпеченні перевірки особистості, яка намагається отримати доступ до ресурсів.

Біометрична ідентифікація – це перевірка особистості користувача за допомогою фізичної риси або поведінкових характеристик, які неможливо легко змінити, таких як відбитки пальців.

Одним з найбільш популярних методів біометричної автентифікації є автентифікація за динамікою підпису. Такий метод робить не лише запис остаточного зображення підпису, ця технологія також записує як зображення було створено, яка різниця в тиску та швидкість з якою записано підпис. Через збільшену кількість змінних, електронні підписи вважаються непідробними.

Професійні шахраї здатні відтворити підпис іншої особи, однак, відтворити точно, як було створено підпис практично неможливо.

Шаблон набору тексту схожий до динаміки підпису. Так само, як реєструється електронний підпис, біометрія шаблону введення записує інтервали між символами та загальну швидкість і зразок набору тексту.

Скануванню очей, або скануванню райдужної оболонки приділяють чимало уваги у фільмах та літературі, але це біометрія, про яку можна говорити вже сьогодні. Багатьом незручно користуватися сканером райдужної оболонки. Лазерні сканери любої частини вашого тіла, а тим більше, і очного яблука викликають багато проблем із здоров'ям і безпекою. Проте, ці занепокоєння є необгрунтованими, і здебільшого базуються на міфах.

Сканер райдужної оболонки ока не використовує лазери для сканування ваших очей. Розпізнавальна камера робить чорні та білі фото і використовує неінвазивне ближнє інфрачервоне світло, що є непомітим для ока і дуже безпечним.

Відбитки пальців – це унікальна біометрія, що використовується десятиліттями для ідентифікації особи. Технологія розпізнавання відбитку пальця також займає відносно мале місце на носіях, для даних які вона фіксує. Використання сучасних алгоритмів машинного навчання дозволяє легко доопрацьовувати отримані відбитки, щоб мінімізувати необхідність у повторному зчитуванні біометрії.

Пристрої геометрії рук або долонь подібні до пристроїв розпізнавання відбитків пальців але є значно досконалішими. Ці пристрої вимірюють усю руку і вимірюють довжину та кут пальців. Цей метод є більш зручним для користувачів ніж пристрої сканування сітківки, проте апаратне забезпечення яке підтримує такі пристрої є громіздким і не дуже легко транспортується.

Розпізнавання голосу співставляє голос людини, що говорить, шаблонам мови, а розпізнавання обличчя вимірює контури обличчя. Такі особливості, як обриси очниць, вилиць, боків рота, а також точне розташування носа та очей.

Особливості лінії волосся зазвичай не вивчаються через тенденції зміни в області волосся.

Хоча двофакторна автентифікація може здатися ідеальним засобом для захисту мереж та ресурсів, існує безліч дір у безпеці, від яких цей тип автентифікації не захистить.

Фальшиві веб-сайти пропонують потенційним хакерам спосіб отримання персональної інформації користувачів. Сайт виглядає справжнім, і користувач вводить на ньому свою інформацію: номер кредитної карти, номер соціального страхування і дані для входу в обліковий запис банку, після чого ця інформація потрапляє до рук зловмисників. Двофакторна автентифікація може не захистити користувача від цих типів атак.

Останнім фактором, який може перешкоджати впровадженню двофакторної автентифікації, є вартість. У двофакторній схемі автентифікації, де другим фактором є використання брелоків або жетонів, витрати лише на ці пристрої можуть становити від 75 до 100 доларів за токен. Для компанії, в якій працює сотня чоловік персоналу, ці початкові витрати можуть бути досить високими. Для компанії, яка має тисячі клієнтів в інтернеті, вартість лише жетонів може сягати мільйонів доларів. Також необхідно врахувати вартість інфраструктури, потрібної для підтримки системи. Сервери, ліцензування, адміністратори та допоміжний персонал повинні отримати компенсацію, апаратне забезпечення сервера повинно бути оновленим, а також витрати на підтримку та ліцензування продукції. На перший погляд лише компанії зі значним капіталом можуть собі дозволити використовувати цю функцію. Однак кожна система автентифікації несе витрати, навіть при використанні можливостей мережевої операційної системи або корпоративної програми. І навіть з найпростішими схемами автентифікації, що передбачають розробку бази даних користувачів та визначення прав доступу слід враховувати витрати на навчання та підтримку користувачів.

РОЗДІЛ 2
ЗАСОБИ РОЗРОБКИ ЕЛЕМЕНТІВ ІНФОРМАЦІЙНОЇ БЕЗПЕКИ РОЗУМНОГО БУДИНКУ

2.1 Опис функціональних вимог

Об’єктом проектування є програмний засіб – мобільний додаток, за допомогою якого здійснюється керування приладами, під’єднаними до мережі розумного будинку. Розроблюваним елементом системи безпеки є двофакторна автентифікація та рівні доступу.

Створюваний програмний засіб повинен виконувати наступні функції:

1) Реєстрація користувачів в системі.
2) Авторизація користувачів.
3) Можливість відновлення забутого паролю.
4) Можливість керування обліковими записами користувачів:
    a. зміна рівня доступу;
    b. видалення облікового запису;
    c. пошук користувачів в базі даних.
5) Керування розумними приладами, під’єднаними до мережі розумного будинку.
6) Обмеження доступу до функцій в залежності від рівня доступу користувача.
7) Функція внесення користувачів у базу даних.
8) Двофакторна автентифікація за допомогою додатку Telegram Messenger.
9) Функція «Запам’ятати мене» при авторизації.
10) Вихід із системи для зміни облікового запису.
11) Зручний графічний інтерфейс.
12) Оптимізація роботи на непродуктивних пристроях.

2.2 Обгрунтування вибору засобів розробки

2.2.1 Мова програмування

Однією з задач розробки мобільного додатку для операційної системи Android було створення системи яку можна було б запустити на переважній більшості мобільних пристроїв. Тому мовою програмування на проекті було обрано Java Android SDK. Мова Java вже адаптова для роботи з переважною більшістю гаджетів. Крім того додатки Android працюють на спеціальній віртуальній машині під назвою Dalvik VM, яка є подібною до віртуальної машини Java під назвою JVM. Додаток Android може працювати на будь-якому пристрої, де реалізована спеціальна віртуальна машина Dalvik VM. Таким чином програми для Android збираються та працюють в оптимальному середовищі з функцією незалежності від платформи.

Хорошим підходом до розробки програмного забезпечення є об'єктно-орієнтований підхід. Java заснована на концепції ООП. Android в значній мірі покладається на основи Java, такі як класи, об'єкти та інші його корисні функції.

Крім того Java має великий набір бібліотек. Скористатися цими бібліотеками легко. Android SDK містить багато стандартних бібліотек Java. Вони надають функціональні можливості для структури даних, математичних функцій, імплантації графіки, багато функцій мережі та багато іншого. Ці бібліотеки Java допомагають реалізовувати майже будь-які функції. Таким чином Java дозволяє швидко та ефективно розробляти програми для Android.

Android створений для роботи на різних апаратних платформах. Таким чином архітектурна нейтральність є бажаною та необхідною. Код Android пишеться один раз, і для його виконання потрібно скомпілювати та оптимізувати власний код для кращої роботи на різних пристроях. Java є незалежною від платформи, тому вона використовується для розробки на Android.

Java є дуже популярною мовою програмування завдяки своїм чудовим можливостям та продуктивності. Спільнота розробників, які володіють знанням Java насправді велика. Таким чином, розробники Android можуть обрати Java, оскільки вже існує хороша база Java програмістів, які можуть допомогти у створенні, вдосконаленні програм для Android, а також завдяки багатьом бібліотекам та інструментам Java полегшує життя розробників. Велика база розробників Java дозволяє швидко розробити безліч програм для Android, тому вона базується на Java.

Розробникам, які не використовують Java, доводиться стикатися з такими серйозними проблемами, як відведення пам'яті та не коректне відображення графічного інтерфейсу. Іноді ці проблеми завдають шкоди на найвищому рівні, такі як збій програми або збій ОС. Android легко реалізує та виправляє загальні проблеми з іншими мовами програмування за допомогою Java. Java не залежить від машини і працює лише в просторі JVM, тому вона захищає вас від цих проблем.

Java – це насправді єдиний варіант для нативних додатків, а нативні додатки – це серце Android.

2.2.2 Мова розмітки

Мовою розмітки сторінок інтерфейсу було обрано розширювану мову розмітки XML. Подібно до HTML (або мови розмітки гіпертексту), XML також є мовою розмітки. Вона була створена як стандартний спосіб кодування даних в інтернет-додатках. Однак на відміну від HTML, XML чутливий до регістру, вимагає, щоб кожен тег був закритий незалежним чином і зберігає пробіли. Теги XML не визначені в XML. Ми повинні визначити власні теги. XML так само добре читається як людиною так і машиною. Крім того, він масштабований і простий у розробці. В Android ми використовуємо XML для розробки макетів, оскільки XML є легкою мовою, він не обтяжує макет.

Android оптимізований для пристроїв з обмеженою пам'яттю та потужністю, тому виглядає дивним, що він використовує XML настільки широко. Зрештою, XML – це багатослівний, зручний для читання формат, не відомий своєю стислістю чи ефективністю.

Під час написання програми відбувається безпосереднє написання XML коду. Після цього плагін Eclipse викликає компілятор ресурсів Android, для попередньої обробки у стислий двійковий формат. Саме цей формат, а не оригінальний текст XML, зберігається на пристрої.

Не менш важливою причиною того, що було обрано XML, є значна кількість інструментів в IDE, які його підтримують. Можна було вибрати, наприклад JSON і всеодно скомпілювати все в двійковий файл. Автоматично сформований файл R.java є помічником для IDE, який забезпечує автозавершення для доступу до ресурсу.

XML легко проаналізувати та обробити програмно, в основному це деревоподібна структура і більшість інструментів створення інтерфейсу вже використовують його. Крім того, щоб побачити візуальний макет інтерфейсу, над яким відбувається робота не потрібно мати жодного робочого Java класу. Елемент/сторінка XML – це, по суті, документ, який можна проаналізувати та відобразити. Якби це був вихідний файл, довелося б або ретельно проаналізувати його, або скомпілювати (все це складніше, ніж аналіз коду у XML).

Макети Android – це деревоподібні структури з деякими вимушеними правилами. XML ідеально підходять для цієї мети. JSON також має деревоподібну структуру, але він орієнтований на дані, тоді як XML орієнтований на документи.

Значення XML базується на ідеї, що документи мають структурні та інші семантичні елементи, які можна описати без посилання на те, як такі елементи повинні відображатися. Фактичне відображення такого документу може змінюватися залежно від носія та обраного стилю.

Використання XML для обміну інформацією також надає багато переваг, зокрема:

XML використовує людську, а не комп'ютерну мову. XML є читабельним і зрозумілим навіть для новачків і на ньому не складніше кодувати, ніж на HTML.

XML повністю сумісний з Java і на 100% портативний. Будь-яка програма, яка може обробляти XML, може використовувати вашу інформацію, незалежно від платформи.

XML можна розширити. Можна створити власні теги або використовувати теги, створені іншими, які використовують нативну мову домену, мають потрібні атрибути, які мають сенс для розробника та кінцевих користувачів.

Приклад у Додатку А ілюструє читабельність і розширюваність мови XML.

Назви тегів HTML нічого не розкривають про значення їх змісту. У наведеному вище прикладі використовується список визначень HTML, але проблеми, властиві використанню HTML, виникають, якщо дані містяться в таблиці або іншому виді HTML-тегів. Приклади:

Багато HTML-тегів є абревіатурами, тому вони не такі читабельні, як загальноприйнята мова. Теги HTML представляють дані як елементи для відображення, наприклад, як визначення у списку або клітинках таблиці. Це ускладнює маніпулювання даними або обмін ними між програмами.

Назви тегів XML читаються і передають значення даних. Інформаційну структуру легко розпізнають як люди, так і комп’ютери, оскільки кожен тег XML безпосередньо передує відповідним даним. Структура даних дотримується помітного та корисного зразка, що полегшує маніпулювання та обмін даними.

2.2.3 База даних

Для створення бази даних було обрано SQLite. Для розробки мобільного додатку було обрано базу даних, яка буде швидкою, масштабованою та безпечною. Маючи в розпорядженні понад сотню доступних баз даних, вибирати потрібну було не дуже просто. Розробники дотримуються набору правил під час вибору бази даних мобільних додатків. Наприклад, очевидно, але важливо знати, з яким типом даних переважно працюватиме програма, як вона буде структурована та який запити використовуватиме.

SQLite – це реляційна система управління базами даних для зберігання великих записів з будь-яким адмініструванням. Він автономний і зберігає об’єкти як файл.

Організації використовують SQLite у своєму додатку за його дивовижний набір інструментів для обробки всіляких даних без будь-яких обмежень сервера.

База даних SQLite містить лише один файл на диску, що робить її портативнішою порівняно з будь-якою іншою базою даних. Деякі організації використовують більше ніж одну базу даних у своїх додатках, тому дуже важливо, щоб база даних була портативною. SQLite також може використовуватися для власних і крос-платформних додатків. Отже, для розробки додатку на React Native або Java можна використовувати SQLite.

Нижче наведено ще кілька аргументів на користь використання SQLite:

SQLite використовує SQL, тому він має всі функції стандартної бази даних SQL.

Деякі розробники потребують баз даних, які можуть масштабувати та забезпечувати підтримку паралельності. SQLite, з його багатим функціоналом, може бути пов'язаний з будь-яким виробничим додатком.

Часто розробникам важко проводити тестування, коли в базі даних додатків є помилки. SQLite дуже зручний для тестування.

Нульова конфігурація: SQLite не потребує жодного комплексу для зберігання даних. Коли ви створюєте власні програми на Java, вони інтегруються з платформою.

Розробники називають SQLite, безсерверною базою даних, і вона справді виправдовує очікування. Вам не потрібно налаштовувати будь-який API або встановлювати будь-яку бібліотеку для доступу до даних із SQLite.

SQLite є кроссплатформенним, що означає, що його можна використовувати в додатку Android, побудованому на Java, а також у кроссплатформенному додатку, побудованому на React Native.

Дуже важливо знати, які типи даних потрібні для проекту і чи підтримує їх база даних чи ні. SQLite підтримує значення NULL. Отже, при збереженні об'єктів, які можуть не мати ідентичності, SQLite не видасть помилки. Цілі числа та текстові рядки також підтримуються SQLite. Крім того SQLite також підтримує дані JSON BLOB.

У багатьох випадках програми, яким потрібно безпосередньо читати / писати файли на сервер, можуть скористатися переходом на SQLite для отримання додаткової функціональності та простоти, яка виникає при використанні мови структурованих запитів.

Для великої частини додатків надмірно використовувати додатковий процес для перевірки бізнес-логіки та функціональності.

2.2.4 Мобільна опереаційна система

Android – це операційна система з відкритим кодом на базі Linux. Вперше вона була представлена 5 листопада 2007 року. В основному Android розглядається як мобільна операційна система. Але вона не обмежується лише мобільними пристроями. В даний час Android використовується в різних пристроях, таких як мобільні телефони, планшети, телевізори, приставки тощо.

Android забезпечує багату структуру програм, що дозволяє створювати інноваційні програми та ігри для мобільних пристроїв у мовному середовищі Java.

Програмний стек Android з відкритим кодом складається з додатків Java, що працюють на об'єктно-орієнтованій структурі додатків на основі Java, поверх основних бібліотек Java, що працюють на віртуальній машині Dalvik із компіляцією JIT.

Android підтримує безліч технологій підключення, включаючи GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE, NFC, WiMAX.

Полегшена реляційна база даних SQLite використовується для зберігання даних.

Android підтримує різні типи аудіо/відео форматів медіа, таких як H.263, H.264, MPEG-4 SP, AMR, AMR-WB, AAC, HE-AAC, AAC 5.1, MP3, MIDI, Ogg Vorbis, WAV , JPEG, PNG, GIF, BMP та WebP.

Веб браузер доступний в Android базується на механізмі розмітки з відкритим вихідним кодом Blink у поєднанні з механізмом JavaScript V8 Chrome що підтримує HTML5 і CSS3.

Також доступні такі форми обміну повідомленнями як CMC та MMC. Вони також включають текстові повідомлення з різьбовим надсиланням повідомлень та обмін повідомленнями Android Cloud To Device (C2DM). Також підтримується вдосконалена версія C2DM. Android Google Cloud Messaging (GCM) також є частиною служб Android Push Messaging.

Доступна багатозадачність додатків з унікальною обробкою розподілу пам'яті, за допомогою чого користувач може переходити від одного завдання до іншого. Також забезпечено підтримку одночасної роботи різних програм.

Операційна система підтримує змінювані віджети. Віджети можна змінити, тому користувачі можуть розширювати їх, щоб показувати більше вмісту, або зменшувати, щоб заощадити місце.

Wi-Fi технологія дозволяє програмам встановлювати безпосередній зв'язок через однорангове з'єднання з високою пропускною здатністю. Також починаючи з версії Android 4.0 забезпечено підтримку знімку екрана, натискаючи кнопки живлення та головний екран одночасно. Популярна технологія Android-Ray на основі NFC дозволяє користувачам миттєво ділитися, просто торкаючись двох телефонів із підтримкою NFC. Android підтримує багато різних мов, також підтримує односпрямований і двонаправлений текст та має вбудовану підтримку мультитач.

2.2.5 Telegram Messenger та Telegram Bot

У якості методу двофакторної автентифікації, що реалізується у додатку було обрано підтвердження за номером телефону через додаток Telegram Messenger.

Розглянемо основні переваги і недоліки такого способу автентифікації.

Переваги:

- генерація нових кодів при кожному вході у систему. Якщо зловмисники перехоплять логін та пароль користувача, вони нічого не зможуть зробити без коду;
- прив'язка до номеру телефону. Без фізичного доступу до телефону неможливо здійснити вхід;
- Telegram відомий, як один з найбезпечніших месенджерів, тож отримати доступ до коду в теорії можливо, проте дуже важко.

Недоліки:

- потрібно встановлювати додатковий програмний засіб, але якщо замовник всеодно ним користується, то цей недолік нівелюється;
- неможливо увійти у систему за відсутності інтернет-з'єднання;
- якщо користувач авторизується і отримує коди на одному і тому ж пристрої, наприклад на смартфоні, то захист перестає бути двофакторним. Проте в додатку Телеграм існує можливість

зашифрувати чат з ботом, встановити пароль, приховати повідомлення, так щоб тільки користувач, знаючи пароль міг увійти в переписку і прочитати код доступу.

Бот, що надсилає коди доступу було розроблено за допомогою Telegram API. Telegram пропонує два типи API для розробників. API Bot дозволяє легко створювати програми, які використовують повідомлення з Telegram для інтерфейсу. API Telegram і TDLib дозволяють створювати власні клієнти Telegram. Можна безкоштовно використовувати обидва API.

Bot API дозволяє підключати ботів до системи Telegram. Боти Telegram – це спеціальні облікові записи, для налаштування яких не потрібен додатковий номер телефону. Ці облікові записи служать інтерфейсом для коду, який працює на сервері. Для використання цих можливостей не потрібно детально розбиратись у протоколі шифрування MTProto – посередницький сервер Telegram Bot буде обробляти все шифрування та підтримувати зв'язок з API Telegram. Достатньо налагодити зв'язок із цим сервером через простий HTTPS-інтерфейс, який пропонує спрощену версію API Telegram.

Розробник ботів також може використовувати API для платежів, щоб приймати платежі від користувачів Телеграм по всьому світу.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ПІДСИСТЕМИ БЕЗПЕКИ РОЗУМНОГО БУДИНКУ

3.1 Функціональні можливості системи реєстрації та автентифікації користувача

Програмна реалізація проекту виконана у середовищі Android Studio. Функціональна частина розроблена мовою програмування Java Android SDK з залученням бібліотек SQLiteDBHelper та Telegram API. Графічний інтерфейс розроблено у середовищі Android Studio за допомогою мови розмітки XML.

База даних також розроблена у середовищі Android Studio з використанням технології SQLite та плагіну SQLiteScout, що використовується для керування базою даних.

Для двофакторної реалізації обрано метод підтвердження особистості через відправлення коду підтвердження користувачу в дотатку Telegram Messenger за допомогою бота, з використанням Telegram API.

Реєстрація нового користувачів відбувається у такій послідовності:

- користувач натискає кнопку «Немає облікового запису» та скеровується у вікно реєстрації (рис. 3.1), де зазначає свої особисті дані, такі як: ПІБ, номер телефону, пароль, підтвердження паролю. Після заповнення необхідних полів він натискає кнопку «Підтвердити»;

```
case R.id.btnSignUp:
    Intent intent = new Intent( packageContext: this, SignUpActivity.class);
    startActivity(intent);
    break;
```

Рисунок 3.1 – Фрагмент коду «Перехід у вікно реєстрації»

- додаток перевіряє чи заповнені обов’язкові поля, чи збігаються між собою два введені паролі та чи немає користувача з таким номером телефону у

загальній базі даних. Якщо усі перевірки завершились успішно то відбувається перехід у наступне вікно, якщо ні – користувач отримує повідомлення з подальшими інструкціями (рис. 3.2);

```
case R.id.btnSubmit:

    if (checkFields()) {
        if (checkEmail()) {
            if (checkPasswords()) {
                Intent intent1 = new Intent( packageContext: this, TelegramActivity.class);
                intent1.putExtra( name: "name", name );
                intent1.putExtra( name: "surname", surname);
                intent1.putExtra( name: "phone", phone);
                intent1.putExtra( name: "password", password);
                startActivity(intent1);
                break;
            } else {
                Toast.makeText( context: this,  text: "Паролі не збігаються!", Toast.LENGTH_SHORT).show();
            }
        } else {
            Toast.makeText( context: this,  text: "Користувач з таким номером телефону вже зареєстрований!", Toast.LENGTH_SHORT).show();
        }
    } else {
            Toast.makeText( context: this,  text: "Заповніть усі поля!", Toast.LENGTH_SHORT).show();
        }

    break;
default:
    break;
```

Рисунок 3.2 – Фрагмент коду «Підтвердження реєстрації»

- потім додаток перенаправляє користувача у вікно підтвердження облікового запису. На цьому етапі користувач повинен встановити додаток Telegram Messenger, здійснити пошук бота SSHSAuth_bot, та розпочати діалог, натиснувши кнопку «Старт»;
- бот здійснює перевірку chat_id користувача, підтверджує його обліковий запис і заносить відповідний особистий ідентифікаційний код в базу даних. Авторизація користувача у системі відбувається за таким алгоритмом:
- у головному вікні програми автентифікації користувач вводить персональний номер та пароль;
- система перевіряє чи зареєстровано користувача з такими даними у базі та чи співпадає ідентифікатор користувача та пароль. Якщо все вірно, користувач потрапляє на вікно підтвердження особи. Якщо ні – користувач отримує повідомлення про помилку і запит на повторне введення даних (рис. 3.3);

```
case R.id.btnLogin:

    Cursor c = db.query( table: "Accounts",  columns: null,  selection: null,  selectionArgs: null,  groupBy: null,  having: null,  orderBy: null);

    if (c.moveToFirst()) {
        int phoneColIndex = c.getColumnIndex( columnName: "phone");
        int passwordColIndex = c.getColumnIndex( columnName: "password");
        int statusColIndex = c.getColumnIndex( columnName: "status");
        int chatIDColIndex = c.getColumnIndex( columnName: "chatID");

        do {
            if (login.equals(c.getString(phoneColIndex)) & password.equals(c.getString(passwordColIndex))) {
                correctData = true;
                if (chbRememberUser.isChecked()){
                    saveText();
                }
                chatID = c.getString(chatIDColIndex);
                webViewMain.loadUrl("https://api.telegram.org/bot1854828309:AAEewx6CkPnmOofk5KbnfyF8X3RNvdU5JZY/sendMessage?chat_id=" + chatID + "?&text=" + valcodeString);
                valcodeString = String.valueOf(valcode);
                Intent intent1 = new Intent( packageContext: this, LoginConfirmActivity.class);
                intent1.putExtra( name: "status", c.getString(statusColIndex));
                intent1.putExtra( name: "phone", c.getString(phoneColIndex));
                intent1.putExtra( name: "valcode", valcodeString);

                c.moveToFirst();
                c.close();
                startActivity(intent1);
                break;
            }
            else {
                correctData = false;
            }
        } while (c.moveToNext() & !correctData);
        if (!correctData) {
            Toast.makeText( context: this,  text: "Невірна комбанація логіну та паролю!", Toast.LENGTH_SHORT).show();
            break;
        }
    } else {
        c.close();
        break;
    }
    break;
```

Рисунок 3.3 – Фрагмент коду «Процес авторизації користувача»

- у наступному вікні користувач повинен ввести варифікаційний код, отриманий у вигляді текстового повідомлення від боту SSHSAuth_bot у додатку Telegram Messenger (рис. 3.4);

```
if (correctData) {
            webView.loadUrl("https://api.telegram.org/bot1854828309:AAEewx6CkPnmOofk5KbnfyF8X3RNvdU5JZY/sendMessage?chat_id=" + chatID + "?&text=" + valcodeString);
            Log.d(TAG,  msg: "Message sent: https://api.telegram.org/bot1854828309:AAEewx6CkPnmOofk5KbnfyF8X3RNvdU5JZY/sendMessage?chat_id= " + chatID + "?&text=" + valcodeString);

    try {
        Log.d(TAG,  msg: "Start waiting");
        TimeUnit.SECONDS.sleep( timeout: 2);
        Log.d(TAG,  msg: "Stop waiting");
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
```

Рисунок 3.4 – Фрагмент коду «Відправлення коду підтвердження користувачу через Telegram-бота»

- якщо користувач вводить правильний код автентифікації, то потрапляє у головне меню керування приладами та переходить до безпосереднього управління елементами розумного будинку. Якщо ні – отримує сервісне повідомлення про помилку.

Відновлення паролю:

- для відновлення паролю, користувач вводить свій номер телефону у відповідному полі;
- система перевіряє, чи є у базі даних користувач з таким номером. Якщо такого користувача зареєстровано, то він отримує на свій номер телефону код в SSHSAuth_bot. Якщо ні – отримує повідомлення, про те, що такий користувач відсутній у базі даних;
- користувач вводить отриманий код у відповідне поле, натискає кнопку «Продовжити» та перенаправляється у вікно для створення нового паролю. Вводить новий пароль, підтверджує та може авторизуватися з новим паролем.

Керування приладами:

- користувачі з різними рівнями доступу мають доступ до різних приладів;
- передбачено рівні доступу: «Адміністратор», «Персонал», «Гість» та «Незнайомець»;
- користувач з рівнем доступу «Адміністратор» має доступ до усіх приладів, і має доступ до вікна керування обліковими записами;
- користувач з рівнем доступу «Персонал» має доступ до усіх необхіжних приладів, за виключенням сенсору газу;
- користувач з рівнем доступу «Гість» може віддалено керувати лише освітлювальним обладнанням (рис. 3.5);
- користувач з рівнем доступу «Незнайомець» немає прав доступу до керування жодним з пристроїв;
- усі нові зареєстровані користувачі отримують рівень доступу «Незнайомець». Адміністратор у вікні керування обліковими записами встановлює інший рівень доступу користувачу. Таким чином зловмисники, які зареєструвалися у системі не отримують жодних прав доступу до керування приладами, поки Адміністратор не видасть їм ці права;
- користувач може вийти з системи та повернутися у вікно авторизації.

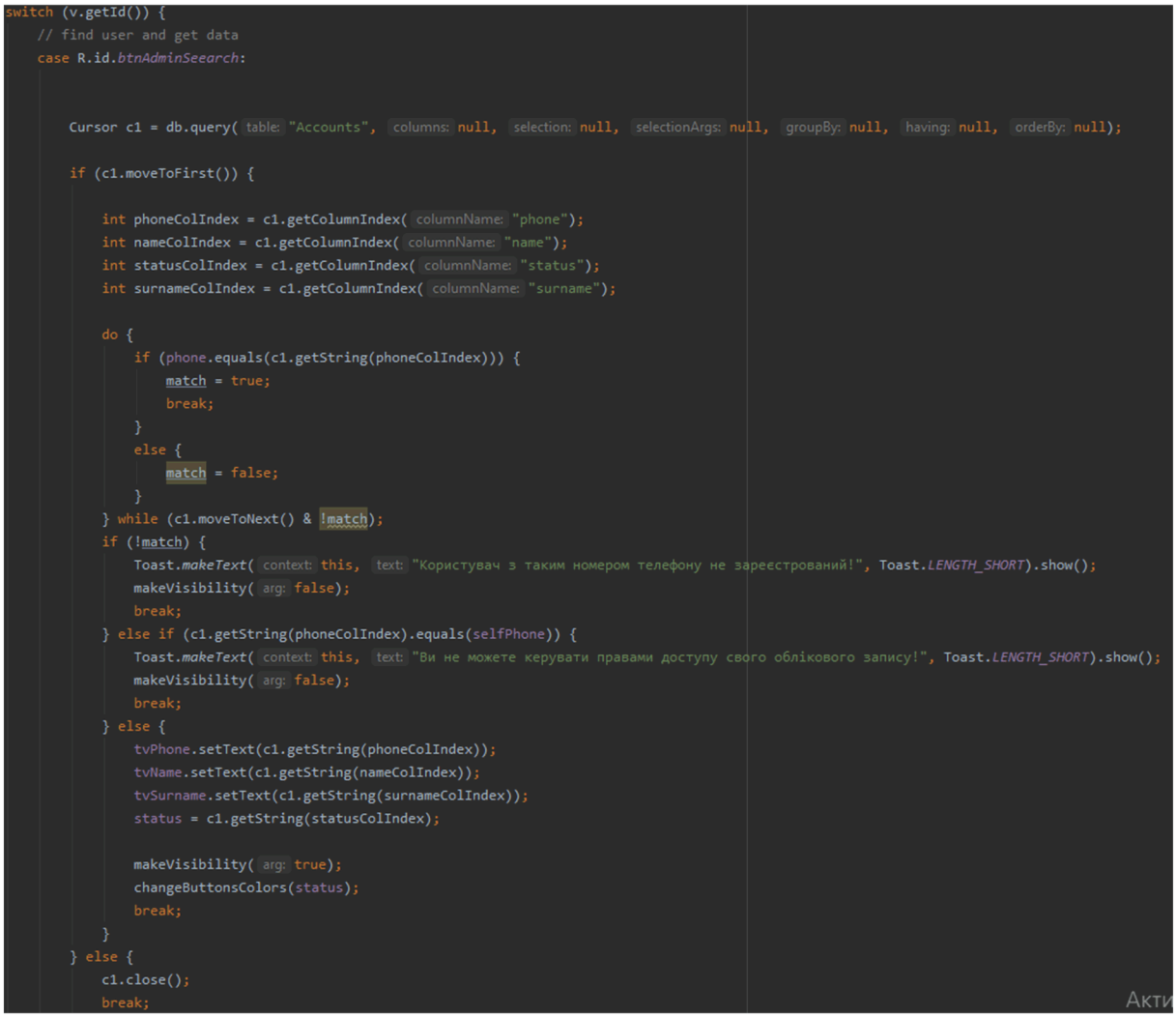
```
case R.id.btnLight:
    if (lightBtnAccess) {
        if(!light) {
            btnLight.setText("Вимкнути світло");
            Toast.makeText( context: this,  text: "Світло увімкнуте", Toast.LENGTH_SHORT).show();
            light = true;
            break;
        } else {
            btnLight.setText("Увімкнути світло");
            Toast.makeText( context: this,  text: "Світло вимкнуте", Toast.LENGTH_SHORT).show();
        } light = false;
    } else {
        Toast.makeText( context: this,  text: "Немає доступу!", Toast.LENGTH_SHORT).show();
    }
    break;
```

Рисунок 3.5 – Фрагмент коду «Керування приладами розумного будинку»

Керування користувачами:

- користувачі з рівнем доступу «Адміністратор», з вікна керування приладами можуть переміститись у вікно керування обліковими записами натиснувши на відповідний елемент керування;
- у вікні керування обліковими записами «Адміністратор» може провести пошук користувача (рис. 3.6) за його номером телефону і отримати додаткову інформацію про нього, наприклад ім’я чи рівень доступу;
- «Адміністратор» не може бачити пароль користувача. Однак доступ до паролів має працівник, що здійснює безпосереднє обслуговування бази даних;
- «Адміністратор» може змінювати рівень доступу будь-якого іншого користувача з правами «Персонал», «Гість» або «Незнайомець». Для зміни прав доступу іншого «Адміністратора» потрібна відповідна згода;
- «Адміністратор» може видаляти або тимчасово блокувати облікові записи інших користувачів;
- Якщо систему обслуговує лише один «Адміністратор» він не може змінювати власний рівень доступу та видаляти власний обліковий запис.

Детальний програмний код наведено у додатках Б-Д.



```
switch (v.getId()) {
    // find user and get data
    case R.id.btnAdminSeearch:


        Cursor c1 = db.query( table: "Accounts",  columns: null,  selection: null,  selectionArgs: null,  groupBy: null,  having: null,  orderBy: null);

        if (c1.moveToFirst()) {

            int phoneColIndex = c1.getColumnIndex( columnName: "phone");
            int nameColIndex = c1.getColumnIndex( columnName: "name");
            int statusColIndex = c1.getColumnIndex( columnName: "status");
            int surnameColIndex = c1.getColumnIndex( columnName: "surname");

            do {
                if (phone.equals(c1.getString(phoneColIndex))) {
                    match = true;
                    break;
                }
                else {
                    match = false;
                }
            } while (c1.moveToNext() & !match);
            if (!match) {
                Toast.makeText( context: this,  text: "Користувач з таким номером телефону не зареєстрований!", Toast.LENGTH_SHORT).show();
                makeVisibility( arg: false);
                break;
            } else if (c1.getString(phoneColIndex).equals(selfPhone)) {
                Toast.makeText( context: this,  text: "Ви не можете керувати правами доступу свого облікового запису!", Toast.LENGTH_SHORT).show();
                makeVisibility( arg: false);
                break;
            } else {
                tvPhone.setText(c1.getString(phoneColIndex));
                tvName.setText(c1.getString(nameColIndex));
                tvSurname.setText(c1.getString(surnameColIndex));
                status = c1.getString(statusColIndex);

                makeVisibility( arg: true);
                changeButtonsColors(status);
                break;
            }
        } else {
            c1.close();
            break;
```

Рисунок 3.6 – Фрагмент коду «Пошук користувачів у меню адміністратора»

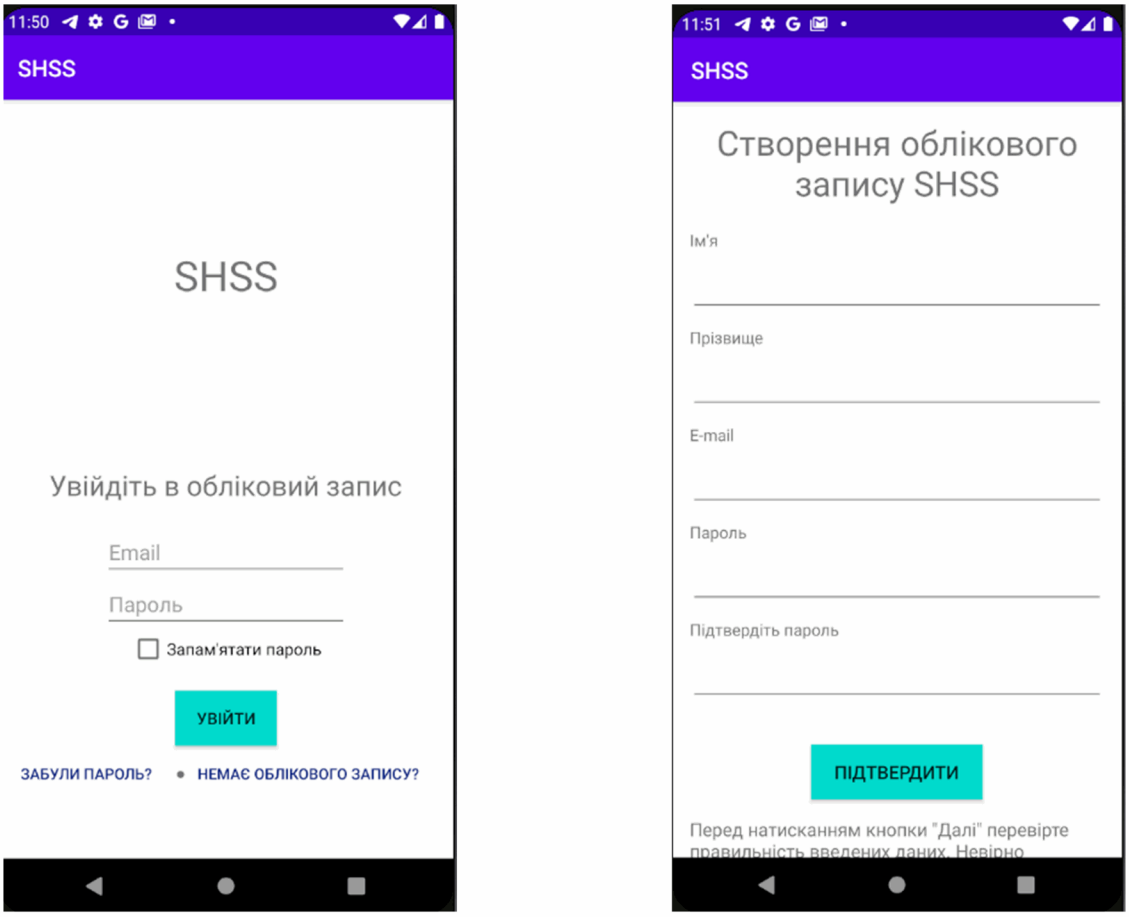
Для збереження даних облікових записів користувачів спроектовано та запущено базу даних за технологією SQLite. Вона містить 1 таблицю з даними про облікові записи. Таблиця містить такі стовпці як id – номер запису в таблиці (integer), name – ім’я (string), surname – прізвище (string), number – номер телефону (string), password – пароль (string), status – рівень доступу (string), chat_id – особистий ідентифікатор (string).

Додаток звертається до бази даних за допомогою класу SQLiteOpenHelper, який надсилає до бази даних запити мовою SQL за допомогою медодів DBHelper, що є конструктором суперкласів onCreate (він викликається якщо база даних ще не створена і лише створює таблицю) та onUpgrade, за допомогою якого виконуються усі зміни в базі даних.

3.2 Інформаційне наповнення та дизайн мобільного додатку

Запроектований програмний засіб для двофакторної автентифікації у системі управління розумним будинком складається з 8 вікон (activity), що розрізняються за своїм функціональним призначенням:

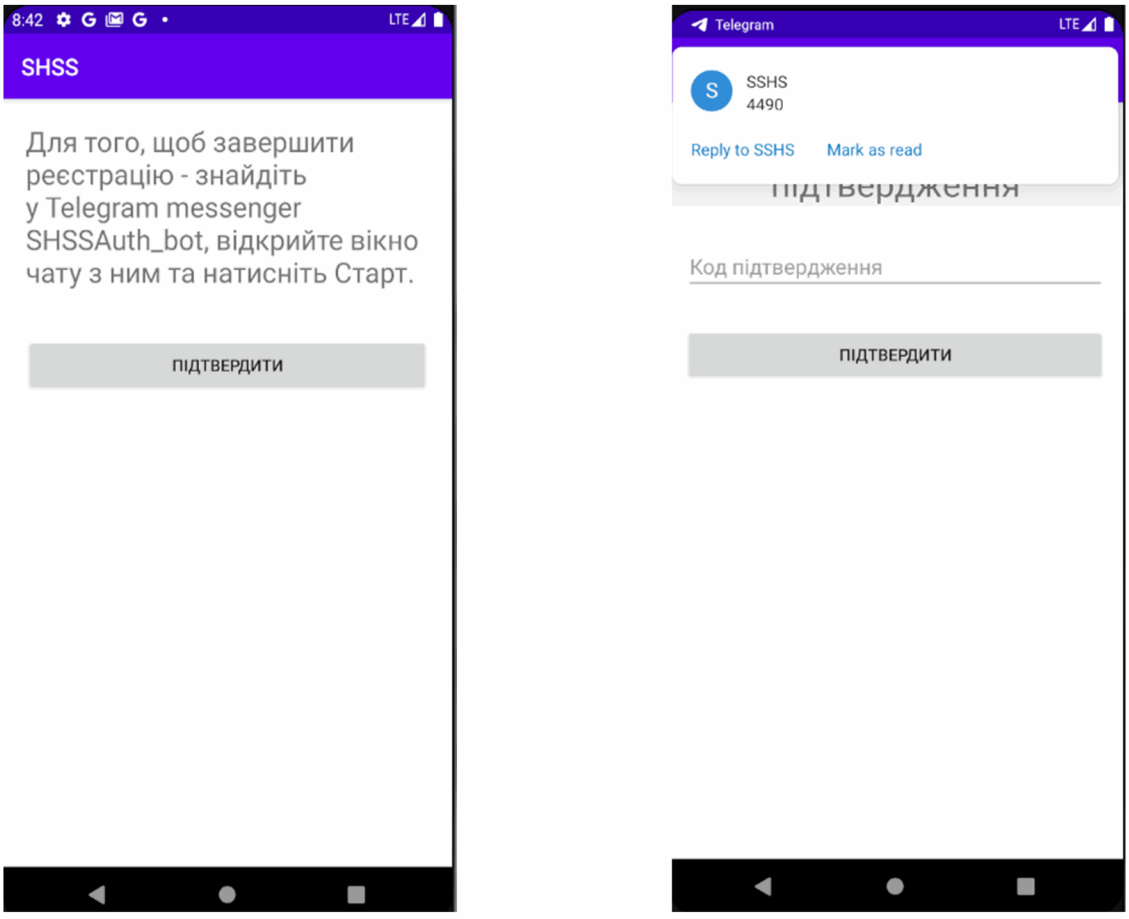
1) Стартове вікно (рис. 3.7, а) або вікно авторизації складається з двох частин. Верхня частина містить 2 текстові блоки, де вказано назву програми та підказки користувачу. У нижній частині вікна розіщено поля для введення номеру телефону та паролю користувача, перемикач «Запам’ятати мене», кнопку «Увійти», кнопку відновлення паролю та кнопку створення нового облікового запису.



а) б)

Рисунок 3.7 – Інтерфейс вікон авторизації (а) та реєстрації (б)

2) Вікно реєстрації (рис. 3.7, б) містить 5 полів для введення текстових даних: імені, прізвища, номеру телефону, паролю та підтвердження пароль. До кожного текстового поля прикріплено відповідні підказки. Внизу вікна є кнопка «Підтвердження реєстрації» та підказка, щодо введення коректних даних.
3) Вікно підтвердження авторизації у Телеграм-боті (рис. 3.8). Містить текстовий блок з підказкою щодо активації бота та кнопку «Підтвердити».



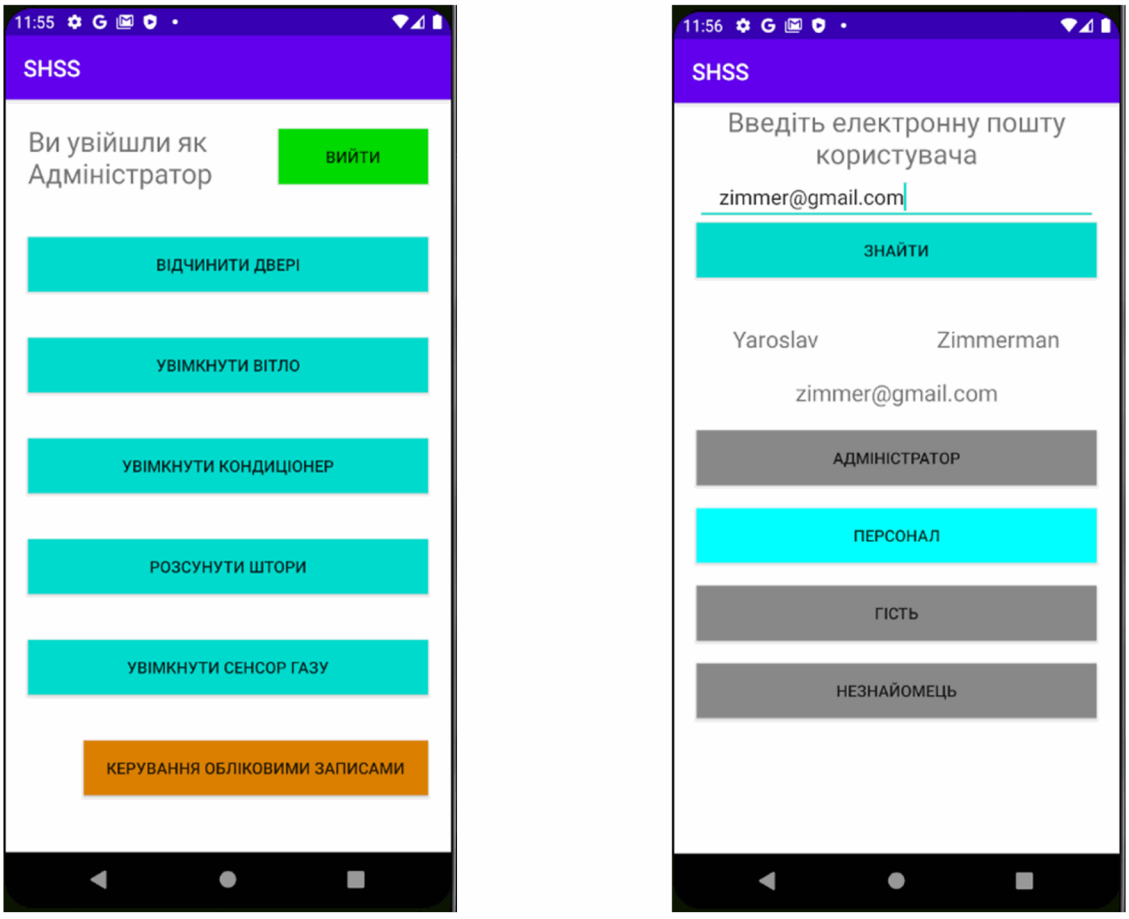
а) б)

Рисунок 3.8 – Вікна активації телеграм бота (а) і підтвердження авторизації (б)

4) Головне меню програми (рис. 3.9, а). Містить кнопки керування функціональними системами розумного будинку, що змінюють колір залежно від рівня доступу користувача. Також кнопки виходу із системи, текстовий блок, де відображається рівень доступу користувача

та кнопка керування обліковими записами (тільки для користувачів з рівнем доступу «Адміністратор»).

5) Вікно керування обліковими записами (рис. 3.9, б). Містить текстовий блок з підказкою, поле для введення номеру телефону користувача, кнопку «Знайти» та 3 текстових блоки для відображення ім'ені, прізвищем та номером телефону знайденого користувача. Нижче розташовано кнопки для зміни рівня доступу користувача або видалення його облікового запису.



а) б)

Рисунок 3.9 – Інтерфейс вікон меню (а) та керування обліковими записами (б)

6) Вікно відновлення паролю. Містить текстовий блок з підказкою та поле для введення номеру телефону і кнопку «Підтвердити». Кнопка відправляє користувача до наступного вікна, де знаходиться поле для

введення коду з телеграм-бота та 2 поля для створення і реєстрації нового паролю.

Для зручного використання додатку на пристроях із різним розміром екрану усі вікна підтримують прогортування якщо розміру екрану недостатньо для відображення усіх елементів вікна. Детальний XLM код основних вікон даного проекту наведено у додатках Е-Ж.

3.3 Тестування та експлуатація

Програмний засіб тестувався за допомогою вбудованого емулятора Android Pixel 3 API 30 на смартфоні Xiaomi Redmi Note 12 під управлінням операційної системи Android 13. Додаток не виявив помилок та багів.

Проводилося тестування на помилки при користування базою даних, відправлення пустого поля в базу даних унеможливлено. Отримання з бази даних порожніх полів також унеможливлено програмними засобами.

В інтерфейсі додатку помилок не виявлено. Додаток тестувався на двох пристроях з різною роздільною здатністю екрану та різним розміром екрану. На обидвох пристроях інтерфейс відображався нормально, елементи інтерфейсу не накладалися, не виходили за край екрану і не зникали.

Усі функції закладені в програму виконуються правильно.

Тест під високими навантаженнями не виконувався, через недоцільність його проведення. Додаток може одночасно використовувати лише один користувач і у нього не закладено функції, які можуть суттєво навантажити ресурси пристрою, тому навантажити додаток недоцільно на даному етапі розробки.

Бот в додатку Telegram, що розроблений для реалізації двофакторної автентифікації працює правильно. Отримує ідентифікатор користувача, його номер мобільного телефону та відправляє код підтвердження.

3.4 Ефективність прийнятих рішень

Вибір мови програмування Java дозволив досягнути коректої роботи розробленого програмного засобу на пристроях Android а також можливість його перенесення на інші операційні платформи (Windows, Linux).

Використання XML дозволило оптимізувати процес створення графічного інтерфейсу для мобільного додаткуоскільки усі необхідні інструменти для роботи з XML вже вбудовано в середовище IDE.

Для підвищення безпеки систем розумного будинку було обрано двофакторну ідентифікацію користувачів та розподіл рівнів доступу користувачів перш за все через низьку собіватрість таких рішень. Двофакторна автентифікація в реалізації за допомогою Telegram-бота не потребує додаткових витрат адже API Telegram є безкоштовним ресурсом і в даному випадку не потребує видатків на хостинг, як наприклад поштовий сервер, що відправляє електронну пошту з кодами доступу або сервіс для розсилки sms-повідомлень. У разі вибору іншого способу двофакторної автентифікації потрібно було б придбати апаратні токени або ж встановлювати камери для розпізнавання обличчя, сканери відбитків пальця або мікрофони для голосового підтвердження.

Функціонування система розподілу рівнів доступу також не потребує додаткових витрат для розгортання, а її обслуговування може здійснювати сам користувач (власник розумного будинку). Вона реалізується програмно як і двофакторна автентифікація за допомогою тих самих безкоштовних засобів розробки.

РОЗДІЛ 4
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Аналіз травмонебезпечних ситуацій під час роботи з комп'ютером

Комп'ютер є життєво важливим інструментом у багатьох різних робочих місцях для дорослих та дітей. Але тривалі періоди використання комп'ютера збільшують шанси отримати травму. Неправильне використання комп'ютера може спричинити біль у м'язах та суглобах, надмірне пошкодження плеча, руки, зап'ястя чи кисті та перенапруження очей.

Можна зменшити або уникнути цих ризиків за допомогою правильних меблів, кращої постави та хороших звичок, таких як: перерва на відпочинок та обмеження часу, проведеного за комп'ютером.

Біль у спині та шиї, головні болі та біль у плечах та руках – це поширені комп'ютерні травми. Такі проблеми з м'язами та суглобами можуть бути спричинені або погіршені через поганий дизайн робочого місця, погану поставу та тривале сидіння.

Хоча сидіння вимагає менших зусиль ніж стояння, воно все одно викликає фізичну втому, і людині потрібно тримати частини тіла рівномірно протягом тривалого періоду часу. Це зменшує кровообіг м'язів, кісток, сухожиль та зв'язок, іноді призводячи до скутості та болю. Якщо робоче місце не встановлене належним чином, ці стійкі положення можуть спричинити ще більше навантаження на м'язи та суглоби.

М'язи та сухожилля можуть почати боліти при повторюваних рухах і незручних позах. Це називається «травмою від надмірного використання», яка зазвичай виникає в лікті, зап'ясті або кисті користувачів комп'ютерів.

Симптомами цих травм, пов'язаних із надмірним навантаженням, є біль, набряк, скутість суглобів, слабкість і оніміння.

Фокусування очей на одній точці протягом тривалого періоду часу викликає втому. Людське око структурно воліє дивитись на предмети, що знаходяться на відстані більше 6 метрів, тому будь яка робота, що виконується поблизу спричиняє додаткове навантаження очних м'язів.

Екран комп'ютера з підсвічуванням також може спричинити втому очей. Хоча немає жодних доказів того, що втома очей пошкоджує зір, користувачі комп'ютерів можуть отримати такі симптоми, як затуманення зору, тимчасову неможливість зосередитися на віддалених предметах та головний біль.

Зростаюче використання портативних комп'ютерів спричиняє біль, напруження та травми серед користувачів комп'ютерів.

Ноутбуки були розроблені для використання протягом коротких періодів часу, коли людина не могла отримати доступ до настільного комп'ютера. Але в наш час багато людей постійно користуються ноутбуком.

Проблема в тому, що монітор і клавіатура ноутбука знаходяться дуже близько один до одного. Розташування монітора на правильній висоті для спини та шиї змушує вас занадто високо підіймати руки та плечі. Але щоб розташувати клавіатуру на найкращій висоті для ваших рук і плечей, ви повинні згорнути плечі та шию, щоб побачити монітор. Носіння ноутбука також може напружувати м'язи та суглоби.

Обмежена рухливість сприяє травмуванню частин тіла, що відповідають за рух: м'язів, кісток, сухожиль та зв'язок. Іншим фактором є стійке, локалізоване напруження на певних ділянках тіла. Шия і поперек – це регіони, які зазвичай найбільше страждають.

Тривале сидіння зменшує рух тіла, роблячи м'язи в'ялими, що тягнуться, стискаються або напружуються при раптовому розтягуванні, викликає втому в м'язах шиї та спини, уповільнюючи кровопостачання та створює сильну напругу у хребті, особливо в попереку або шиї, і спричиняє стійке стиснення дисків хребта.

4.2 Інструкція з охорони праці під час роботи з комп'ютером

1.1. Типова інструкція з охорони праці під час роботи з комп'ютером розробляється відповідно до Положення про розробку інструкцій з охорони праці, затвердженого наказом Держнаглядохоронпраці від 29.01.1998 № 9, Типового положення про порядок проведення навчання і перевірки знань з питань охорони праці, затвердженого наказом Держнаглядохоронпраці від 26.01.2005 № 15, Правил охорони праці під час експлуатації електронно-обчислювальних машин, затверджених наказом Державного комітету України з промислової безпеки, охорони праці та гірничого нагляду від 26.03.2010 № 65, Державних санітарних правил і норм роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПіН 3.3.2.007-98, затверджених постановою Головного державного санітарного лікаря України від 10.12.1998 № 7, Загальних вимог стосовно забезпечення роботодавцями охорони праці працівників, затверджених наказом Міністерства надзвичайних ситуацій України від 25.01.2012 № 67 (НПАОП 0.00-7.11-12).

1.2. Працівника, який використовує персональний комп'ютер (далі — користувач), інструктують перед початком роботи (первинний інструктаж), а потім через кожні 6 місяців (повторний інструктаж). Результати інструктажу заносять до Журналу реєстрації інструктажів з питань охорони праці на робочому місці (у журналі має бути підпис особи, яка інструктує, та користувача).

1.3. Користувач зобов'язаний дбати про особисту безпеку і здоров`я, а також про безпеку і здоров'я довколишніх при виконанні будь-яких робіт, а також під час перебування на території підприємства.

1.4. До роботи на персональному комп'ютері допускають осіб, які пройшли інструктажі з питань охорони праці та пожежної безпеки.

1.5. Користувач зобов'язаний: виконувати правила внутрішнього трудового розпорядку; не допускати за своє робоче місце сторонніх осіб; не

виконувати вказівок, які суперечать правилам охорони праці та пожежної безпеки; знати правила надання домедичної допомоги; знати розташування та вміти користуватись первинними засобами пожежогасіння; вміти працювати з комп'ютером.

1.6. Основні небезпечні та шкідливі виробничі фактори, що можуть впливати на користувача: підвищений рівень статичної електрики; нерівномірність розподілу яскравості в полі зору; підвищена яскравість світлового зображення; ураження електричним струмом; перенапруга зору та уваги; тривалі статичні навантаження.

1.7. У приміщеннях із комп'ютером має бути природне і штучне освітлення.

1.8. При розміщенні робочих місць необхідно унеможливити пряме засвічування екрана природним освітленням.

1.9. При природному освітленні слід передбачити наявність сонцезахисних засобів (плівка, жалюзі, штори тощо).

1.10. Світлові відблиски із клавіатури, екрана та інших частин ПК у напрямку очей користувача неприпустимі.

1.11. Основним обладнанням робочого місця є ПК або ноутбук, монітор, клавіатура, маніпулятор, робочий стіл, стілець (крісло). При розміщенні елементів робочого місця слід враховувати: робочу позу користувача; простір для розміщення користувача; можливість огляду елементів робочого місця; можливість огляду простору поза межами робочого місця; можливість робити записи, розміщувати на робочому столі документацію та матеріали, які використовує користувач.

1.12. Розміщення елементів робочого місця не має заважати рухам та переміщенню для експлуатування ПК.

1.13. Монітор встановлюють так, щоб відстань від поверхні екрана до очей користувача була 600-700 мм залежно від розміру екрану.

1.14. Клавіатуру розміщують на робочому або окремому столі на відстані 100-300 мм від краю з боку користувача. Положення клавіатури та кут

її нахилу залежить від побажання користувача (як правило, в межах 5-15°). Не допускати хитання клавіатури.

1.15. Конструкція робочого столу має бути такою, щоб оптимально розмістити на робочій поверхні обладнання, що використовують, з урахуванням кількості, розмірів, конструктивних особливостей і характеру його роботи.

1.16. Крісло має забезпечувати підтримку раціональної робочої пози під час виконання основних виробничих операцій та можливість зміни пози. Тип робочого крісла обирають залежно від характеру та тривалості роботи.

1.17. Раціональна поза користувача: ступні розташовані на підлозі або на підставці для ніг; стегна зорієнтовані у горизонтальній площині; верхні ділянки рук вертикальні; кут ліктьового суглоба у межах 70-90°; зап'ястя зігнуті під кутом не більше ніж 20°; нахил голови у межах 15-20°, а часті її повороти виключені.

1.18. Для забезпечення оптимальної робочої пози користувача необхідно: засоби праці, з якими користувач має тривалий або найбільш частий зоровий контакт, розмістити у центрі зони зорового спостереження та моніторного поля; забезпечити відстань близько 500 мм між найважливішими засобами праці, з якими користувач працює найчастіше.

1.19. ПК встановлювати на рівній твердій поверхні (столі). Не дозволено встановлювати ПК та оргтехніку на хитких підставках чи на похилій поверхні.

1.20. ПК не встановлювати впритул до стіни, перегородки тощо. Не допускати загородження вентиляційних отворів ПК сторонніми предметами.

1.21. Розетка біля ПК має бути в доступному місці, щоб в аварійних випадках можна було своєчасно відімкнути обладнання. Не рекомендовано використовувати подовжувачі.

1.22. Під час переміщення ПК чи периферійних пристроїв слід витягти вилку живлення з розетки. Не допускати ушкодження чи модифікування шнура живлення. Заборонено ставити важкі речі на шнур живлення, тягнути

чи надмірно перегинати його, скручувати та перев'язувати шнур живлення вузлом.

1.23. ПК під'єднувати до електромережі лише за допомогою справних штепсельних з'єднань та електророзеток заводського виробництва.

1.24. Заборонено під'єднувати електрообладнання до звичайної двошнурової електромережі.

1.25. Штепсельні з'єднання та електророзетки мають бути зі спеціальними контактами для під'єднання нульового захисного провідника. Їхня конструкція має забезпечувати з'єднання нульового захисного провідника раніше, ніж з'єднання фазового та нульового робочого провідників. Порядок роз'єднань при вимкненні має бути зворотнім.

4.3 Схема освітлення і вентиляції робочого місця

Як відомо, тривала робота за комп'ютером та з документами при недостатньому рівні освітленості може призвести до значного перенапруження зору, тому вимоги до освітлення є досить важливими.

Додатково, окрім вже перелічених документів, вимоги до освітлення встановлено ДБН В.2.5-28-2006 «Природне і штучне освітлення», затвердженими наказом Мінрегіону від 15.05.2006 р. № 168.

Робоче місце необхідно організовувати так, щоб природне світло було з лівого боку (п. 4.3 ДСанПіН 3.3.2.007-98). Робоче місце необхідно розміщувати таким чином, щоб уникнути попадання прямого світла в очі. Для нормалізації рівня світлового випромінювання комп'ютерного монітору необхідно застосовувати приекранні фільтри, локальні світлофільтри (засоби індивідуального захисту очей) та інші засоби захисту, що пройшли випробування в акредитованих лабораторіях і мають щорічний гігієнічний сертифікат (п. 4.19 ДСанПіН 3.3.2.007-98).

Штучне освітлення приміщення має здійснюватись системою загального рівномірного освітлення (п. 3.2.2 ДСанПіН 3.3.2.007-98). У приміщеннях при переважній роботі з документами допускається використання системи комбінованого освітлення, тобто встановлення світильників місцевого освітлення додатково до загального.

Як джерела штучного освітлення необхідно використовувати люмінесцентні лампи. Згідно з п. 3.2.5 ДСанПіН 3.3.2.007-98 система загального освітлення має бути у вигляді суцільних або переривчатих ліній світильників, що розташовані збоку від робочих місць (зазвичай ліворуч) паралельно лінії зору працівників.

Допускається застосування ламп розжарювання у світильниках місцевого освітлення та, у разі влаштування відбитого освітлення у виробничих чи адміністративно-громадських приміщеннях, металогалогенних ламп потужністю 250 Вт.

Коефіцієнт пульсації не повинен перевищувати 5% (п. 3.2.14 ДСанПіН 3.3.2.007-98). Рівень освітленості на робочому столі в зоні розташування документів має бути в межах 300–500 лк. Світильники місцевого освітлення слід встановлювати таким чином, щоб не створювати відблисків на поверхні екрана, а освітленість екрана має не перевищувати 300 лк.

Для забезпечення нормованих значень освітленості у приміщеннях відповідно до п. 3.2.15 ДСанПіН 3.3.2.007-98 необхідно мити вікна і світильники не рідше 2 разів на рік, а також своєчасно замінювати лампи, що перегоріли.

Приміщення для роботи з персональними комп'ютерами мають бути обладнані системами опалення, кондиціонування повітря, або припливно-витяжною вентиляцією. У приміщеннях на робочих місцях мають забезпечуватись оптимальні значення параметрів мікроклімату: температури, відносної вологості та рухливості повітря відповідно до норм та правил, а також ДБН В.2.5-67:2013 «Опалення, вентиляція та кондиціонування», затверджених наказом Мінрегіону від 25.01.2013 р. № 24.

Відповідно до санітарних норм мікроклімату виробничих приміщень ДСН 3.3.6.042-99 в офісних приміщеннях температура повітря повинна становити 16–25°С, відносна вологість повітря — 40–60%, швидкість руху повітря — не більше 0,1 м/с.

Під час перевищення припустимих значень робочий день співробітників повинен бути скорочений мінімум на 10%.

Для підтримки допустимих значень мікроклімату та концентрації позитивних і негативних іонів необхідно передбачати установки або прилади зволоження та/або штучної іонізації, кондиціювання повітря. В Україні відсутні затверджені на законодавчому рівні гранично допустимі норми вмісту вуглекислого газу в повітрі для житлових, офісних та громадських споруд. Проте, враховуючи його вплив на працівників, а саме суттєве зниження їх працездатності, роботодавцям варто приділяти цьому питанню увагу та вживати заходи профілактики.

Окрім цього, наслідком сучасного технічного прогресу є зростання з кожним роком енергоспоживання та збільшення навантаження на кабелі, що в свою чергу призводить до збільшення напруги електромагнітних полів, несприятлива дія яких може призвести до погіршення стану здоров’я працівників. Таким чином, варто пам’ятати, що причиною зниження працездатності офісних працівників дуже часто є саме незадовільні параметри мікроклімату.

ВИСНОВКИ

Сучасні інформаційні технології активно впроваджуються в домашні повсякденні справи, створюються нові програмні застосунки для вирішення простих щоденних питань, або керування глобальними пректами. Загроза втрати інформації або її несанкціонованого поширення досить гостро постає для різноманітних систем Інтернету речей, зокрема для розумного будинку.

Аналіз промислових аналогів показав, що єдиний підхід до формування концепції безпеки у IoT відсутній, тому більшість розробників запозичують ідеї та підходи у технологіях кіберзахисту. Порівняльний аналіз систем керування розумним будинком від Xiaomi, Orvibo, Fibaro, Ajax показав, що основними методами автентифікації користувачів є зв'язка логін-пароль-електронна пошта, а до додаткових функцій безпеки можна віднести зашифровані канали передачі даних та апаратний захист (камери, сенсори).

Проведено аналіз існуючих варіантів двофакторної автентифікації, а саме: підтвердження через СМС; підтвердження через електронну пошту; підтвердження через месенджери; сканування відбитків пальців; сканування райдужної оболонки ока; розпізнавання голосу; розпізнавання обличчя; аналіз динаміки підпису і друку. Вибрано і реалізовано такий варіант двофакторної автентифікації як підтвердження через месенджер, а саме Телеграм. API, оскільки він не потребує розробки окремого серверного обладнання для іншої платформи, як у випадку з електронною поштою, та не потребує встановлення додаткового обладнання, як у випадку із біометрією.

Розроблено демонстраційну систему керування розумними приладами для демонстрації можливостей двофакторної автентифікації і рівнів доступу, що також були включені у систему. Основна мова розробки – Java, інтерфейс створено за допомогою мови розмітки XML, а бази даних адмініструються за допомогою SQLite.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Lea P. Internet of Things for Architects: Architecting IoT solutions by implementing sensors, communication infrastructure, edge computing, analytics, and security. Birmingham: Packt Publishing, 2018. 524 p.

2. Інтернет речей для індустріальних і гуманітарних застосунків. У трьох томах. Том 1. Основи і технології / За ред. В. С. Харченка. Харків: Національний аерокосмічний університет ХАІ, 2019. 547 с.

3. Інтернет речей для індустріальних і гуманітарних застосунків. У трьох томах. Том 2. Моделювання і розробка / За ред. В. С. Харченка. Харків: Національний аерокосмічний університет ХАІ, 2019. 574 с.

4. Інтернет речей для індустріальних і гуманітарних застосунків. У трьох томах. Том 3. Оцінювання та впровадження / За ред. В. С. Харченка. Харків: Національний аерокосмічний університет ХАІ, 2019. 921 с.

5. Sklyar V.V., Yatskiv V.V., Yatskiv N.G. Dependability and Security of IoT: Practicum / Kharchenko V.S. and Sklyar V.V. (Eds.). Ministry of Education and Science of Ukraine, National Aerospace University “KhAI”, Ternopil National Economic University, 2019. 98 p.

6. Norris D. The Internet of Things: Do-It-Yourself at Home Projects for Arduino, Raspberry Pi and BeagleBone Black. McGraw Hill TAB, 2015. 352 p.

7. Singh R., Gehlot A., Gupta L., Singh B., Swain M. Internet of Things with Raspberry Pi and Arduino. CRC Press, 2019. 190 p.

8. Жураковський Б.Ю., Зенів І.О. Технології Інтернету речей: навчальний посібник, Київ: КПІ ім. Ігоря Сікорського, 2021. 271 с.

9. Локотков А. Інтерфейси послідовної передачі даних. Стандарти RS-422/RS-485. СТА. 1997. № 3. С. 110-119.

10. Chumakevych V., Puleko I., Ptashnyk V., Sokulskyi O. Development of an algorithm for increasing the image contrast of objects in an urban agglomeration

with high-rise buildings. 15th International Conference Monitoring of Geological Processes and Ecological Condition of the Environment, Kyiv, 2021, p. 1-5.

11. Лапоніна О. Основи мережевої безпеки: криптографічні алгоритми і протоколи взаємодії, Київ: ІНТУІТ, 2010. 608 с.

12. Бурячок В., Аносов А., Семко В., Соколов В., Складанний П. Технології забезпечення безпеки мережевої інфраструктури, Київ: КУБГ, 2019. 218 с.

13. Ярочкін В. Інформаційна безпека, Харків: Академічний проект, 2004. 640 с.

14. Gasson M., Meints M., Warwick K. A study on PKI and biometrics, Warszawa: OLO, 2005. 138 p.

15. Гультяєв А. Проектування користувацького інтерфейсу, Львів: Solution, 2012. 352 с.

16. Домарев В. Безпека ІТ: методологія створення систем безпеки, Миколаїв: Адміралтейство, 2007. 688 с.

17. СНіП РК 2.04.-05.2002. Природне та штучне освітлення. Київ, 2002. 21 с.

18. Гандзюк М. П., Желібо Є. П., Халімовський М. О. Основи охорони праці: підручник. Київ: Каравела, 2004. 408 с.

19. Апостолюк В. С., Джигирей А. В. та інші. Безпека праці: ергономічні та естетичні основи: навч. пос. Київ: Знання, 2006. 215 с.

ДОДАТКИ

ДОДАТОК А

Порівняння XML та HTML коду

Мова розмітки	HTML	XML
Код	<HTML> <H1 ID="MN">State</H1> <H2 ID="12">City</H2> <DL> <DT>Name</DT> <DD>Johnson</DD> <DT>Population</DT> <DD>5000</DD> </DL> <H2 ID="15">City</H2> <DL> <DT>Name</DT> <DD>Pineville</DD> <DT>Population</DT> <DD>60000</DD> </DL> <H2 ID="20">City</H2> <DL> <DT>Name</DT> <DD>Lake Bell</DD> <DT>Population</DT> <DD>20</DD> </DL> </HTML>	<?XML VERSION="1.0" STANDALONE="yes" ?> <STATE STATEID="MN"> <CITY CITYID="12"> <NAME>Johnson</NAME> <POPULATION>5000</POPULATION> </CITY> <CITY CITYID="15"> <NAME>Pineville</NAME> <POPULATION>60000</POPULATION> </CITY> <CITY CITYID="20"> <NAME>Lake Bell</NAME> <POPULATION>20</POPULATION> </CITY> </STATE>
Результат	**State** **City** Name Johnson Population 5000 **City** Name Pineville Population 60000 **City** Name Lake Bell Population 20	**State** **City** Name Johnson Population 5000 **City** Name Pineville Population 60000 **City** Name Lake Bell Population 20

ДОДАТОК Б

Опис класу activity_Menu.java

```
package com.example.securitysystem;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.graphics.Color;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;
public class activity_Menu extends
AppCompatActivity implements
View.OnClickListener {
   // interface elements initialization, declaring
variables
   Button btnDoor;
   Button btnLight;
   Button btnCondition;
   Button btnCurtains;
   Button btnGas;
   Button btnAdmin;
   Button btnLogOut;
   TextView tvStatus;
   SharedPreferences loginData;
   String status, selfPhone;
   String adminStatus = "Ви увійшли як
Адміністратор";
   String staffStatus = "Ви увійшли як Персонал";
   String guestStatus = "Ви увійшли як Гість";
   String nonameStatus = "Ви увійшли як
Незнайомець";
   boolean doors = false;
   boolean light = false;
   boolean condition = false;
   boolean curtains = false;
   boolean gas = false;
   boolean doorsBtnAccess;
   boolean lightBtnAccess;
   boolean conditionBtnAccess;
   boolean curtainsBtnAccess;
   boolean gasBtnAccess;
   private static final String TAG = "myLogs";
   @Override
   protected void onCreate(Bundle
savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity__menu);
      Intent statusIntent = getIntent();
      status = statusIntent.getStringExtra("status");
      selfPhone =
statusIntent.getStringExtra("phone");
      Log.d(TAG, "Received status: " + status);
      Log.d(TAG, "Received phone: " + selfPhone);
      // find interface elements by id
      btnAdmin = (Button)
findViewById(R.id.btnAdmin);
      btnDoor = (Button)
findViewById(R.id.btnDoor);
      btnLight = (Button)
findViewById(R.id.btnLight);
      btnCondition = (Button)
findViewById(R.id.btnCondition);
      btnCurtains = (Button)
findViewById(R.id.btnCurtains);
      btnGas = (Button) findViewById(R.id.btnGas);
      btnLogOut = (Button)
findViewById(R.id.btnLogOut);
      tvStatus = (TextView)
findViewById(R.id.tvStatus);
      btnAdmin.setOnClickListener(this);
      btnDoor.setOnClickListener(this);
      btnLight.setOnClickListener(this);
      btnCondition.setOnClickListener(this);
      btnCurtains.setOnClickListener(this);
      btnGas.setOnClickListener(this);
      btnLogOut.setOnClickListener(this);
      checkStatus();  }
   // button actions
   @Override
   public void onClick(View v) {switch (v.getId()) {
         case R.id.btnAdmin:
            if (status.equals("admin")) {
               Intent intent1 = new Intent(this,
AdminActivity.class);
               intent1.putExtra("phone", selfPhone);
               startActivity(intent1); }
            break;
         case R.id.btnLogOut:
            clearLoginData();
            Intent intent1 = new Intent(this,
MainActivity.class);
            startActivity(intent1);
            break;
         case R.id.btnDoor:
            if (doorsBtnAccess) {
               if(!doors) { btnDoor.setText("Зачинити
двері");
                  Toast.makeText(this, "Двері
відчинено", Toast.LENGTH_SHORT).show();
                  doors = true;
                  break;
               } else { btnDoor.setText("Відчинити
двері");
                  Toast.makeText(this, "Двері
зачинено", Toast.LENGTH_SHORT).show();
               } doors = false;
            } else {Toast.makeText(this, "Немає
доступу!", Toast.LENGTH_SHORT).show();}
            break;
         case R.id.btnLight:
            if (lightBtnAccess) {
               if(!light) {
                  btnLight.setText("Вимкнути світло");
                  Toast.makeText(this, "Світло
увімкнуте", Toast.LENGTH_SHORT).show();
```

```
                light = true;
                break;
            } else {btnLight.setText("Увімкнути
світло");
                Toast.makeText(this, "Світло
вимкнуте", Toast.LENGTH_SHORT).show();
            } light = false;
        } else { Toast.makeText(this, "Немає
доступу!", Toast.LENGTH_SHORT).show();}
        break;
      case R.id.btnCondition:
        if (conditionBtnAccess) {
          if(!condition) {
            btnCondition.setText("Вимкнути
кондиціонер");
            Toast.makeText(this, "Кондиціонер
увімкнено", Toast.LENGTH_SHORT).show();
            condition = true;
            break;
          } else {
btnCondition.setText("Увімкнути кондиціонер");
            Toast.makeText(this, "Кондиціонер
вимкнено", Toast.LENGTH_SHORT).show();
          } condition = false;
          break;
        } else {Toast.makeText(this, "Немає
доступу!", Toast.LENGTH_SHORT).show();}
        break;
      case R.id.btnCurtains:
        if (curtainsBtnAccess) {if(!curtains) {
btnCurtains.setText("Зсунути штори");
            Toast.makeText(this, "Штори
розсунуті", Toast.LENGTH_SHORT).show();
            curtains = true;
            break;
          } else { btnCurtains.setText("Розсунути
штори");
            Toast.makeText(this, "Штори
зсунуті", Toast.LENGTH_SHORT).show();
          } curtains = false; break;
        } else {Toast.makeText(this, "Немає
доступу!", Toast.LENGTH_SHORT).show();}
        break;
      case R.id.btnGas:
        if (gasBtnAccess) {
          if(!gas) {
 btnGas.setText("Вимкнути сенсор газу");
            Toast.makeText(this, "Сенсор газу
увімкнено", Toast.LENGTH_SHORT).show();
            gas = true;
            break;
          } else { btnGas.setText("Увімкнути
сенсор газу");
            Toast.makeText(this, "Сенсор газу
вимкнено", Toast.LENGTH_SHORT).show();
          } gas = false;
          break;
        } else {
          Toast.makeText(this, "Немає доступу!",
Toast.LENGTH_SHORT).show();}
        break;
      default:
        break; } }
  // check user status
  void checkStatus(){switch (status) {case "admin":
        tvStatus.setText(adminStatus);
        doorsBtnAccess = true;
        conditionBtnAccess = true;
        curtainsBtnAccess = true;
        gasBtnAccess = true;
        lightBtnAccess = true;
        break;
      case "guest":
        tvStatus.setText(guestStatus);
        doorsBtnAccess = false;
btnDoor.setBackgroundColor(Color.GRAY);
        conditionBtnAccess = false;
btnCondition.setBackgroundColor(Color.GRAY);
        curtainsBtnAccess = false;
btnCurtains.setBackgroundColor(Color.GRAY);
        gasBtnAccess = false;
        btnGas.setBackgroundColor(Color.GRAY);
        lightBtnAccess = true;
        btnAdmin.setClickable(false);
        btnAdmin.setVisibility(View.INVISIBLE);
        break;
      case "staff":
        tvStatus.setText(staffStatus);
        doorsBtnAccess = false;
btnDoor.setBackgroundColor(Color.GRAY);
        conditionBtnAccess = true;
        curtainsBtnAccess = true;
        gasBtnAccess = false;
        btnGas.setBackgroundColor(Color.GRAY);
        lightBtnAccess = true;
        btnAdmin.setClickable(false);
        btnAdmin.setVisibility(View.INVISIBLE);
        break;
      case "noname":
        tvStatus.setText(nonameStatus);
        doorsBtnAccess = false;
btnDoor.setBackgroundColor(Color.GRAY);
        conditionBtnAccess = false;
btnCondition.setBackgroundColor(Color.GRAY);
        curtainsBtnAccess = false;
btnCurtains.setBackgroundColor(Color.GRAY);
        gasBtnAccess = false;
        btnGas.setBackgroundColor(Color.GRAY);
        lightBtnAccess = false;
btnLight.setBackgroundColor(Color.GRAY);
        btnAdmin.setClickable(false);
        btnAdmin.setVisibility(View.INVISIBLE);
        break;
      default:
        break; } }
  void clearLoginData () {
    loginData = getSharedPreferences("Login data",
MODE_PRIVATE);
    SharedPreferences.Editor ed = loginData.edit();
    ed.clear();
    ed.apply();}}
```

ДОДАТОК В

Опис класу AdminActivity.java

```
package com.example.securitysystem;

import androidx.appcompat.app.AppCompatActivity;

import android.content.ContentValues;
import android.content.Context;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.graphics.Color;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
public class AdminActivity extends
AppCompatActivity implements
View.OnClickListener {
    Button btnAdminSearch, btnAdminStatus,
btnStaffStatus, btnGuestStatus, btnNonameStatus;
    TextView tvAdminHint, tvName, tvSurname,
tvPhone;
    EditText etAdminSearch;
    String phone, status, selfPhone;
    DBHelper dbHelper;
    private static final String TAG = "myLogs";
    // when activity opens
    @Override
    protected void onCreate(Bundle
savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_admin);
        Intent statusIntent = getIntent();
        selfPhone =
statusIntent.getStringExtra("phone");
        btnAdminSearch = (Button)
findViewById(R.id.btnAdminSeearch);
        btnAdminStatus = (Button)
findViewById(R.id.btnAdminStatus);
        btnStaffStatus = (Button)
findViewById(R.id.btnStaffStatus);
        btnGuestStatus = (Button)
findViewById(R.id.btnGuestStatus);
        btnNonameStatus = (Button)
findViewById(R.id.btnNonameStatus);
        btnAdminSearch.setOnClickListener(this);
        btnAdminStatus.setOnClickListener(this);
        btnStaffStatus.setOnClickListener(this);
        btnGuestStatus.setOnClickListener(this);
        btnNonameStatus.setOnClickListener(this);
        tvPhone = (TextView)
findViewById(R.id.tvPhone);
        tvSurname = (TextView)
findViewById(R.id.tvSurname);
        tvName = (TextView)
findViewById(R.id.tvName);
        tvAdminHint = (TextView)
findViewById(R.id.tvAdminHint);
        etAdminSearch = (EditText)
findViewById(R.id.etAdminSearch);
        dbHelper = new DBHelper(this);
        makeVisibility(false);
        Log.d(TAG, "Self phone: " + selfPhone);
    }
    public void onClick(View v) {
        phone = etAdminSearch.getText().toString();
        boolean match = false;
        Log.d(TAG, "Phone: " + phone);
        SQLiteDatabase db =
dbHelper.getWritableDatabase();
        switch (v.getId()) {
            // find user and get data
            case R.id.btnAdminSeearch:
                Cursor c1 = db.query("Accounts", null,
null, null, null, null, null);
                if (c1.moveToFirst()) {
                    int phoneColIndex =
c1.getColumnIndex("phone");
                    int nameColIndex =
c1.getColumnIndex("name");
                    int statusColIndex =
c1.getColumnIndex("status");
                    int surnameColIndex =
c1.getColumnIndex("surname");
                    do {
                        if
(phone.equals(c1.getString(phoneColIndex))) {
                            match = true;
                            break;
                        }
                        else {
                            match = false;
                        }
                    } while (c1.moveToNext() & !match);
                    if (!match) {
                        Toast.makeText(this, "Користувач з
таким номером телефону не зареєстрований!",
Toast.LENGTH_SHORT).show();
                        makeVisibility(false);
                        break;
                    } else if
(c1.getString(phoneColIndex).equals(selfPhone)) {
                        Toast.makeText(this, "Ви не можете
керувати правами доступу свого облікового
запису!", Toast.LENGTH_SHORT).show();
                        makeVisibility(false);
                        break;
                    } else {
tvPhone.setText(c1.getString(phoneColIndex));
    tvName.setText(c1.getString(nameColIndex));
 tvSurname.setText(c1.getString(surnameColIndex));
                        status = c1.getString(statusColIndex);
                        makeVisibility(true);
```

```
                changeButtonsColors(status);
                break;
            }
          } else {
            c1.close();
            break;
          }
        case R.id.btnAdminStatus:
          changeButtonsColors("admin");
          Toast.makeText(this, "Рівень доступу
змінено на" + " " +
btnAdminStatus.getText().toString(),
Toast.LENGTH_SHORT).show();
          changeStatus("admin");
          break;
        case R.id.btnStaffStatus:
          changeButtonsColors("staff");
          Toast.makeText(this, "Рівень доступу
змінено на" + " " +
btnStaffStatus.getText().toString(),
Toast.LENGTH_SHORT).show();
          changeStatus("staff");
          break;
        case R.id.btnGuestStatus:
          changeButtonsColors("guest");
          Toast.makeText(this, "Рівень доступу
змінено на" + " " +
btnGuestStatus.getText().toString(),
Toast.LENGTH_SHORT).show();
          changeStatus("guest");
          break;
        case R.id.btnNonameStatus:
          changeButtonsColors("noname");
          Toast.makeText(this, "Рівень доступу
змінено на" + " " +
btnNonameStatus.getText().toString(),
Toast.LENGTH_SHORT).show();
          changeStatus("noname");
          break;
        default:
          break;
      }
    }
    static class DBHelper extends SQLiteOpenHelper
{
      public DBHelper(Context context) {
        // superclass constructor
        super(context, "SHSS", null, 1);
      }
      @Override
      public void onCreate(SQLiteDatabase db) {
        // create table with fields
        db.execSQL("create table Accounts (id
integer primary key autoincrement, name text,
surname text, phone text, password text, status
text);");
      }
      @Override
      public void onUpgrade(SQLiteDatabase db, int
oldVersion, int newVersion) {
      }
    }
    public void makeVisibility (boolean arg) {
      if (arg) {
        btnAdminStatus.setVisibility(View.VISIBLE);
        btnStaffStatus.setVisibility(View.VISIBLE);
        btnGuestStatus.setVisibility(View.VISIBLE);
btnNonameStatus.setVisibility(View.VISIBLE);
        tvPhone.setVisibility(View.VISIBLE);
        tvName.setVisibility(View.VISIBLE);
        tvSurname.setVisibility(View.VISIBLE);
      } else {
        tvPhone.setVisibility(View.INVISIBLE);
        tvName.setVisibility(View.INVISIBLE);
        tvSurname.setVisibility(View.INVISIBLE);
btnAdminStatus.setVisibility(View.INVISIBLE);
btnStaffStatus.setVisibility(View.INVISIBLE);
  btnGuestStatus.setVisibility(View.INVISIBLE);
   btnNonameStatus.setVisibility(View.INVISIBLE);
      }
    }
    public void changeButtonsColors (String status) {
      switch(status) {
        case "admin":
btnAdminStatus.setBackgroundColor(Color.CYAN);
btnStaffStatus.setBackgroundColor(Color.GRAY);
btnGuestStatus.setBackgroundColor(Color.GRAY);
btnNonameStatus.setBackgroundColor(Color.GRAY);
          break;
        case "staff":
btnAdminStatus.setBackgroundColor(Color.GRAY);
 btnStaffStatus.setBackgroundColor(Color.CYAN);
btnGuestStatus.setBackgroundColor(Color.GRAY);
btnNonameStatus.setBackgroundColor(Color.GRAY);
          break;
        case "guest":
btnAdminStatus.setBackgroundColor(Color.GRAY);
btnStaffStatus.setBackgroundColor(Color.GRAY);
btnGuestStatus.setBackgroundColor(Color.CYAN);
btnNonameStatus.setBackgroundColor(Color.GRAY);
          break;
        case "noname":
btnAdminStatus.setBackgroundColor(Color.GRAY);
btnStaffStatus.setBackgroundColor(Color.GRAY);

btnGuestStatus.setBackgroundColor(Color.GRAY);
btnNonameStatus.setBackgroundColor(Color.CYAN);
          break;
        default:
          break;
      }   }
    public void changeStatus (String status) {
      ContentValues cv = new ContentValues();
      String phoneToChange =
tvPhone.getText().toString();
      SQLiteDatabase db =
dbHelper.getWritableDatabase();
      cv.put("status", status);
      int updCount = db.update("Accounts", cv,
"phone = ?",
          new String[] { phoneToChange });
      dbHelper.close();
    }
}
```

ДОДАТОК Г

Опис класу MainActivity.java

```
package com.example.securitysystem;

import androidx.appcompat.app.AppCompatActivity;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.webkit.WebView;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.Toast;
import java.util.Random;
public class MainActivity extends
AppCompatActivity implements
View.OnClickListener {
    EditText etPhone;
    EditText etPassword;
    LinearLayout loginLayout;
    CheckBox chbRememberUser;
    Button btnLogin;
    Button btnSignUp;
    Button btnRestorePass;
    SharedPreferences loginData;
    WebView webViewMain;
    boolean correctData;
    String login, password;
    String valcodeString;
    int valcode;
    String chatID;
    DBHelper dbHelper;
    ValidationActivity validator;
    final String LOGIN_TEXT = "Login";
    final String PASSWORD_TEXT = "Password";
    @Override
    protected void onCreate(Bundle
savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        etPhone = findViewById(R.id.etPhone);
        etPassword = findViewById(R.id.etPassword);
        loginLayout = findViewById(R.id.loginLayout);
        webViewMain =
findViewById(R.id.webViewMain);

        chbRememberUser =
findViewById(R.id.chbRememberUser);

        btnLogin = findViewById(R.id.btnLogin);
        btnSignUp = findViewById(R.id.btnSignUp);
        btnRestorePass =
findViewById(R.id.btnRestorePass);
        btnSignUp.setOnClickListener(this);
        btnLogin.setOnClickListener(this);
        btnRestorePass.setOnClickListener(this);

        dbHelper = new DBHelper(this);

        loadText();
        autoLogin();

        valcode = randomize();
        valcodeString = String.valueOf(valcode);
    }

    @Override
    public void onClick(View v) {
        login = etPhone.getText().toString();
        password = etPassword.getText().toString();
        correctData = false;
        SQLiteDatabase db =
dbHelper.getWritableDatabase();
        switch (v.getId()) {
            case R.id.btnSignUp:
                Intent intent = new Intent(this,
SignUpActivity.class);
                startActivity(intent);
                break;
            case R.id.btnLogin:
                Cursor c = db.query("Accounts", null, null,
null, null, null, null);
                if (c.moveToFirst()) {
                    int phoneColIndex =
c.getColumnIndex("phone");
                    int passwordColIndex =
c.getColumnIndex("password");
                    int statusColIndex =
c.getColumnIndex("status");
                    int chatIDColIndex =
c.getColumnIndex("chatID");
                    do {
                        if
(login.equals(c.getString(phoneColIndex)) &
password.equals(c.getString(passwordColIndex))) {
                            correctData = true;
                            if
(chbRememberUser.isChecked()){
                                saveText();
                            }
                            chatID =
c.getString(chatIDColIndex);
webViewMain.loadUrl("https://api.telegram.org/bot1
854828309:AAEewx6CkPnmOofk5KbnfyF8X3RNv
dU5JZY/sendMessage?chat_id=" + chatID +
"?&text=" + valcodeString);
                            valcodeString =
String.valueOf(valcode);
                            Intent intent1 = new Intent(this,
LoginConfirmActivity.class);
```

```
                    intent1.putExtra("status",
c.getString(statusColIndex));
                    intent1.putExtra("phone",
c.getString(phoneColIndex));
                    intent1.putExtra("valcode",
valcodeString);
                    c.moveToFirst();
                    c.close();
                    startActivity(intent1);
                    break;
                }
                else {
                    correctData = false;
                }
            } while (c.moveToNext() &
!correctData);
            if (!correctData) {
                Toast.makeText(this, "Невірна
комбанація логіну та паролю!",
Toast.LENGTH_SHORT).show();
                break;
            }
        } else {
            c.close();
            break;
        }
        break;
    case R.id.btnRestorePass:
        Intent intent2 = new Intent(this,
ValidationActivity.class);
        startActivity(intent2);
        break;
    default:
        break;
  }
 }
 static class DBHelper extends SQLiteOpenHelper
{
    public DBHelper (Context context) {
        // superclass constructor
        super(context, "SHSS", null, 1);
    }
    @Override
    public void onCreate(SQLiteDatabase db) {
        // create table with fields
        db.execSQL("create table Accounts (id
integer primary key autoincrement, name text,
surname text, phone text, password text, status text,
chatID text);");
    }
  @Override
  public void onUpgrade(SQLiteDatabase db, int
oldVersion, int newVersion) {
  }
 }
 void saveText() {
  loginData = getSharedPreferences("Login data",
MODE_PRIVATE);
  SharedPreferences.Editor ed = loginData.edit();
  ed.putString(LOGIN_TEXT,
etPhone.getText().toString());
  ed.putString(PASSWORD_TEXT,
etPassword.getText().toString());
  ed.apply();
  /*
  If I want to create file with custom name (It will
be MyPref in this case)
  Instead of getPreferences(MODE_PRIVATE)
method we use getSharedPreferences("My pref",
MODE_PRIVATE) method
   */
 }
 /*
 - Use getPreferences if we work with current
Activity data and don't want to choose filename
 - Use getSharedPreferences if we save data which
is common to multiple activities and choose filename
for yourself
 */
 void loadText() {
  loginData = getSharedPreferences("Login data",
MODE_PRIVATE);
  String savedLogin =
loginData.getString(LOGIN_TEXT, "");
  String savedPassword =
loginData.getString(PASSWORD_TEXT, "");
  etPhone.setText(savedLogin);
  etPassword.setText(savedPassword);
  /*
  If I want to load file with custom name (It will
be MyPref in this case)
  Instead of getPreferences(MODE_PRIVATE)
method we use getSharedPreferences("My pref",
MODE_PRIVATE) method
   */
 }
 void autoLogin () {
  login = etPhone.getText().toString();
  password = etPassword.getText().toString();
  if (!login.isEmpty() & !password.isEmpty()) {
   btnLogin.performClick();
  }
 }
 public int randomize() {
  int min = 1000;
  int max = 9999;
  int diff = max - min;
  Random random = new Random();
  int i = random.nextInt(diff + 1);
  i += min;
  return i;
 }
}
```

ДОДАТОК Д

Опис класу SignUpActivity.java

```
package com.example.securitysystem;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Context;
import android.content.Intent;
import android.database.Cursor;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.widget.Toast;
public class SignUpActivity extends
AppCompatActivity implements
View.OnClickListener {
    Button btnSubmit;
    EditText etName;
    EditText etSurname;
    EditText etPhone;
    EditText etPassword;
    EditText etConfirmPassword;
    String name, surname, phone, password, conPass,
status;
    DBHelper dbHelper;
    @Override
    protected void onCreate(Bundle
savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_sign_up);

        btnSubmit = (Button)
findViewById(R.id.btnSubmit);
        btnSubmit.setOnClickListener(this);

        etName = (EditText)
findViewById(R.id.etName);
        etSurname = (EditText)
findViewById(R.id.etSurname);
        etPhone = (EditText)
findViewById(R.id.etPhone);
        etPassword = (EditText)
findViewById(R.id.etPassword);
        etConfirmPassword = (EditText)
findViewById(R.id.etConfirmPassword);
        // creates object to manage DB versions
        dbHelper = new DBHelper(this);
    }
    @Override
    public void onClick(View v) {
        //get data from fields
        name = etName.getText().toString();
        surname = etSurname.getText().toString();
        phone = etPhone.getText().toString();
        password = etPassword.getText().toString();
        conPass =
etConfirmPassword.getText().toString();
        switch (v.getId()) {
            case R.id.btnSubmit:
                if (checkFields()) {
                    if (checkEmail()) {
                        if (checkPasswords()) {
                            Intent intent1 = new Intent(this,
TelegramActivity.class);
                            intent1.putExtra("name", name );
                            intent1.putExtra("surname",
surname);
                            intent1.putExtra("phone", phone);
                            intent1.putExtra("password",
password);
                            startActivity(intent1);
                            break;
                        } else {
                            Toast.makeText(this, "Паролі не
збігаються!", Toast.LENGTH_SHORT).show();
                        }
                    } else {
                        Toast.makeText(this, "Користувач з
таким номером телефону вже зареєстрований!",
Toast.LENGTH_SHORT).show();
                    }
                } else {
                    Toast.makeText(this, "Заповніть усі
поля!", Toast.LENGTH_SHORT).show();
                }
                break;
            default:
                break;
        }
    }
    public boolean checkFields () {
        if (name.isEmpty() || surname.isEmpty() ||
phone.isEmpty() || password.isEmpty() ||
conPass.isEmpty()) {
            return false;
        } else {
            return true;
        }
    }
    public boolean checkPasswords() {
        if (password.equals(conPass)) {
            return true;
        } else {
            return false;
        }
    }
    public boolean checkEmail() {
        boolean match;
        SQLiteDatabase db =
dbHelper.getWritableDatabase();
        Cursor c = db.query("Accounts", null, null, null,
null, null, null);
        if (c.moveToFirst()) {
            int phoneColIndex =
c.getColumnIndex("phone");
            do {
```

```
            if
(phone.equals(c.getString(phoneColIndex))) {
                match = true;
                } else {
                match = false;
            }
        } while (c.moveToNext() & !match);
        if (match) {
            return false;
        } else {
            return true;
        }
    } else { c.close(); }
    return true;
  }
  class DBHelper extends SQLiteOpenHelper {

    public DBHelper (Context context) {
      // superclass constructor
      super(context, "SHSS", null, 1);
    }
    @Override
    public void onCreate(SQLiteDatabase db) {
      // create table with fields
      db.execSQL("create table Accounts (id
integer primary key autoincrement, name text,
surname text, phone text, password text, status text,
chatID text);");
    }
    @Override
    public void onUpgrade(SQLiteDatabase db, int
oldVersion, int newVersion) {
    }
  }
}
                    e.printStackTrace();
                }
            }
        });
            try {
                Log.d(TAG, "Start waiting");
                TimeUnit.SECONDS.sleep(2);
                Log.d(TAG, "Stop waiting");
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            Log.d(TAG, "Telegram id after .run(): " +
telegramID);
        if (checkId(telegramID)) {
            // create object for data
            ContentValues cv = new ContentValues();
            // connect to DB
            SQLiteDatabase db =
dbHelper.getWritableDatabase();
            cv.put("name", name);
            cv.put("surname", surname);
            cv.put("phone", phone);
            cv.put("password", password);
            cv.put("chatID", telegramID);
            cv.put("status", status);
            //inject the record and get it's ID
            long rowID = db.insert("Accounts", null,
cv);
            // close DB connection
            dbHelper.close();
            Intent intent = new Intent(this,
MainActivity.class);
            startActivity(intent);
          } else {
            Toast.makeText(this, "Ви не
авторизувались у SSHSAuth_bot",
Toast.LENGTH_LONG).show();
          }
          break;
          default:
            break;
      }    }
    class DBHelper extends SQLiteOpenHelper {

      public DBHelper (Context context) {
        // superclass constructor
        super(context, "SHSS", null, 1);
      }
      @Override
      public void onCreate(SQLiteDatabase db) {
        // create table with fields
        db.execSQL("create table Accounts (id
integer primary key autoincrement, name text,
surname text, phone text, password text, status text,
chatID text);");
      }
      @Override
      public void onUpgrade(SQLiteDatabase db, int
oldVersion, int newVersion) {
      }
    }
    public boolean checkId(String infoToCheck) {
      boolean match;
      SQLiteDatabase db =
dbHelper.getWritableDatabase();
      Cursor c = db.query("Accounts", null, null, null,
null, null, null);
      if (c.moveToFirst()) {
        int idColIndex =
c.getColumnIndex("chatID");
        do {
          if
(infoToCheck.equals(c.getString(idColIndex))) {
            match = true;
          } else {
            match = false;
          }
        } while (c.moveToNext() & !match);
        if (match) {
          return false;
        } else {
          return true;
        }
      } else {
        c.close();
      }
      c.close();
      return true;
    }    }
```

ДОДАТОК Е

Конструювання вікна activity_menu.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/
android"
   tools:context=".activity_Menu">
   <ScrollView
      android:layout_width="match_parent"
      android:layout_height="match_parent">
      <LinearLayout
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:orientation="vertical">
      <LinearLayout
         android:layout_width="match_parent"
         android:layout_height="wrap_content"
         android:orientation="horizontal">
         <TextView
            android:id="@+id/tvStatus"
            android:layout_width="137dp"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_marginLeft="20sp"
            android:layout_marginTop="20sp"
            android:layout_marginRight="20sp"
            android:layout_marginBottom="20sp"
            android:layout_weight="1"
            android:gravity="left"
            android:text="Ви увійшли, як Адмін"
            android:textSize="24sp"
            android:visibility="visible" />
         <Button
            android:id="@+id/btnLogOut"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_marginLeft="20sp"
            android:layout_marginTop="20sp"
            android:layout_marginRight="20sp"
            android:layout_marginBottom="20sp"
            android:layout_weight="1"
            android:background="#0fda08"
            android:paddingLeft="20sp"
            android:paddingRight="20sp"
            android:text="Вийти" />
      </LinearLayout>
         <Button
            android:id="@+id/btnDoor"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_marginTop="20sp"
            android:layout_marginLeft="20sp"
            android:layout_marginRight="20sp"
            android:layout_marginBottom="20sp"
            android:background="#08dacc"
            android:text="Відчинити двері" />
         <Button
            android:id="@+id/btnLight"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_marginTop="20sp"
            android:layout_marginLeft="20sp"
            android:layout_marginRight="20sp"
            android:layout_marginBottom="20sp"
            android:background="#08dacc"
            android:text="Увімкнути вітло" />
         <Button
            android:id="@+id/btnCondition"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_marginTop="20sp"
            android:layout_marginLeft="20sp"
            android:layout_marginRight="20sp"
            android:layout_marginBottom="20sp"
            android:background="#08dacc"
            android:text="Увімкнути кондиціонер" />
         <Button
            android:id="@+id/btnCurtains"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_marginTop="20sp"
            android:layout_marginLeft="20sp"
            android:layout_marginRight="20sp"
            android:layout_marginBottom="20sp"
            android:background="#08dacc"
            android:text="Розсунути штори" />
         <Button
            android:id="@+id/btnGas"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_marginTop="20sp"
            android:layout_marginLeft="20sp"
            android:layout_marginRight="20sp"
            android:layout_marginBottom="20sp"
            android:background="#08dacc"
            android:text="Увімкнути сенсор газу" />
         <Button
            android:id="@+id/btnAdmin"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="end|bottom"
            android:layout_marginLeft="20sp"
            android:layout_marginTop="20sp"
            android:layout_marginRight="20sp"
            android:layout_marginBottom="20sp"
            android:background="#da7f08"
            android:paddingRight="20sp"
            android:text="Керування" />
      </LinearLayout>
   </ScrollView>
</androidx.constraintlayout.widget.ConstraintLayout
>
```

ДОДАТОК Є

Конструювання вікна activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical"
tools:context="com.example.securitysystem.MainAct
ivity"
   tools:layout_editor_absoluteX="134dp"
   tools:layout_editor_absoluteY="243dp">
   <LinearLayout
      android:id="@+id/logoLayout"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:layout_weight="70"
      android:background="#00C41414"
      android:orientation="vertical">
      <TextView
         android:id="@+id/tvLabel"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_gravity="center"
         android:layout_weight="20"
         android:text="SHSS"
         android:gravity="center"
         android:textSize="36sp"
         android:visibility="visible" />
   </LinearLayout>
   <LinearLayout
      android:id="@+id/loginLayout"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:layout_weight="60"
      android:orientation="vertical">
      <TextView
         android:id="@+id/tvHint1"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_gravity="center"
         android:layout_marginTop="14sp"
         android:layout_marginBottom="20sp"
         android:text="Увійдіть в обліковий запис"
         android:textSize="24sp" />
      <EditText
         android:id="@+id/etPhone"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_gravity="center"
         android:ems="10"
         android:hint="Номер телефону"
         android:inputType="phone" />
      <EditText
         android:id="@+id/etPassword"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_gravity="center"
         android:ems="10"
         android:gravity="left"
         android:hint="Пароль"
         android:inputType="textPassword" />
         <Button
         android:id="@+id/btnLogin"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_gravity="center"
         android:layout_marginTop="20sp"
         android:background="#08dacc"
         android:text="Увійти" />
      <LinearLayout
         android:layout_width="wrap_content"
         android:layout_height="match_parent"
         android:layout_gravity="center"
         android:orientation="horizontal">
         <Button android:id="@+id/btnRestorePass"
style="@style/Widget.AppCompat.Button.Borderless
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="top"
            android:layout_marginRight="10sp"
            android:layout_marginBottom="5dp"
            android:background="#00FCF8F8"
            android:text="Забули пароль?"
            android:textColor="#081C7E"
            android:textSize="13sp" />
         <TextView
            android:id="@+id/tvDot"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="top|center"
            android:layout_marginLeft="10sp"
            android:layout_marginRight="10sp"
android:layout_marginBottom="5dp"
            android:layout_weight="1"
            android:gravity="center"
            android:text="●" />
         <Button
            android:id="@+id/btnSignUp"
style="@style/Widget.AppCompat.Button.Borderless
"           android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="top"
            android:layout_marginEnd="10sp"
            android:layout_marginRight="10sp"
            android:layout_marginBottom="5dp"
            android:background="#00FCF8F8"
            android:text="Немає облікового запису?"
            android:textColor="#081C7E"
            android:textSize="13sp" />
      </LinearLayout>
      <WebView
         android:id="@+id/webViewMain"
         android:layout_width="match_parent"
         android:layout_height="match_parent"
         android:visibility="invisible"
         tools:visibility="invisible" />
   </LinearLayout>
</LinearLayout>
```

ДОДАТОК Ж

Конструювання вікна activity_sign.up.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/
android"
    xmlns:app="http://schemas.android.com/apk/res-
auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".SignUpActivity">
    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:layout_editor_absoluteY="731dp">
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">
            <TextView
                android:id="@+id/tvHint5"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                android:layout_marginLeft="15sp"
                android:layout_marginTop="15sp"
                android:layout_marginRight="15sp"
                android:layout_marginBottom="15sp"
                android:gravity="center"
                android:text="Створення облікового
запису SHSS"
                android:textSize="30sp" />
            <TextView
                android:id="@+id/tvUserName"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginLeft="15sp"
                android:layout_marginTop="5sp"
                android:layout_marginRight="15sp"
                android:layout_marginBottom="5sp"
                android:text="Ім'я" />
            <EditText
                android:id="@+id/etName"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginLeft="15sp"
                android:layout_marginTop="5sp"
                android:layout_marginRight="15sp"
                android:layout_marginBottom="5sp"
                android:ems="10"
                android:inputType="textPersonName" />
            <TextView
                android:id="@+id/tvSurname"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginLeft="15sp"
                android:layout_marginTop="5sp"
                android:layout_marginRight="15sp"
                android:layout_marginBottom="5sp"
                android:text="Прізвище" />
            <EditText
                android:id="@+id/etSurname"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginLeft="15sp"
                android:layout_marginTop="5sp"
                android:layout_marginRight="15sp"
                android:layout_marginBottom="5sp"
                android:ems="10"
                android:inputType="textPersonName" />
            <TextView
                android:id="@+id/tvEmail"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginLeft="15sp"
                android:layout_marginTop="5sp"
                android:layout_marginRight="15sp"
                android:layout_marginBottom="5sp"
                android:text="Номер телефону" />
            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:orientation="horizontal">
                <TextView
                    android:id="@+id/tvPhone"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_marginLeft="15sp"
                    android:layout_marginTop="5sp"
                    android:layout_marginRight="5sp"
                    android:layout_marginBottom="5sp"
                    android:layout_weight="1"
                    android:text="+38" />
                <EditText
                    android:id="@+id/etPhone"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_marginLeft="5sp"
                    android:layout_marginTop="5sp"
                    android:layout_marginRight="15sp"
                    android:layout_marginBottom="5sp"
                    android:layout_weight="100"
                    android:ems="10"
                    android:inputType="phone" />
            </LinearLayout>
            <TextView
                android:id="@+id/tvPassword"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginLeft="15sp"
                android:layout_marginTop="5sp"
                android:layout_marginRight="15sp"
                android:layout_marginBottom="5sp"
                android:text="Пароль" />
            <EditText
                android:id="@+id/etPassword"
                android:layout_width="match_parent"
```

```
            android:layout_height="wrap_content"
            android:layout_marginLeft="15sp"

            android:layout_marginTop="5sp"
            android:layout_marginRight="15sp"
            android:layout_marginBottom="5sp"
            android:ems="10"
            android:inputType="textPassword" />
        <TextView
            android:id="@+id/tvConfirmPassword"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginLeft="15sp"
            android:layout_marginTop="5sp"
            android:layout_marginRight="15sp"
            android:layout_marginBottom="5sp"
            android:text="Підтвердіть пароль" />
        <EditText
            android:id="@+id/etConfirmPassword"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginLeft="15sp"
            android:layout_marginTop="5sp"
            android:layout_marginRight="15sp"
            android:layout_marginBottom="5sp"
            android:ems="10"
            android:inputType="textPassword" />
        <Button
            android:id="@+id/btnSubmit"
            style="@style/Widget.AppCompat.Button"
            android:layout_width="150dp"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_marginTop="30sp"
            android:background="#08dacc"
            android:text="Підтвердити"
            android:textSize="16sp" />
    </LinearLayout>
  </ScrollView>
</androidx.constraintlayout.widget.ConstraintLayout>
```