МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет)_ Автоматизації і комп'ютерних систем_____
Кафедра _Інформаційних технологій, штучного інтелекту і кібербезпеки_

«До захисту в ЕК»	**«До захисту допущено»**
Директор інституту(декан факультету)	Завідувач кафедри
_____ ___Андрій ФОРСЮК___	_____ _Сергій ГРИБКОВ__
(підпис) (ім'я та прізвище)	(підпис) (ім'я та прізвище)
«07» __червня_____ 2023р.	«07» ____червня_____ 2023р.

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

Зі спеціальності__122 «Комп'ютерні науки»_____
(код та назва спеціальності)
освітньо-професійної програми____Інформаційні системи та штучний інтелект____
_____
на тему: «Розроблення інформаційної системи підтримки управління кав'ярні»

Виконав: здобувач_4_ курсу, групи_КН-4-2_

_____Акулов Ілля Олександрович _____ _____
(прізвище, ім'я, по батькові повністю) (підпис)

Керівник _Самсонов Валерій Васильович_____ _____
(прізвище , ім'я та по батькові повністю) (підпис)

Консультанти_____ _____
(ім'я та прізвище) (підпис)

_____ _____
(ім'я та прізвище) (підпис)

_____ _____
(ім'я та прізвище) (підпис)

Рецензент _____ _____
(ім'я та прізвище) (підпис)

Я як здобувач(ка) Національного університету харчових технологій розумію і підтримую політику університету з академічної доброчесності. Я не надавав(-ла) і не одержував(-ла) недозволеної допомоги під час підготовки цієї роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Здобувач_____
(підпис)

Київ - 2023р.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет) Автоматизації і комп'ютерних систем
Кафедра Інформаційних технологій, штучного інтелекту і кібербезпеки
Освітній ступінь бакалавр
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма Інформаційні системи та штучний інтелект

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інформаційних технологій, штучного інтелекту і кібербезпеки
Сергій ГРИБКОВ

" 04 " квітня 2023 року

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА

Акулов Ілля Олександрович
(прізвище, ім'я, по батькові)

1. Тема роботи «Розроблення інформаційної системи управління кав'ярні»
керівник роботи Самсонов Валерій Васильович, професор, к.т.н
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від 03 квітня 2023 року № 204-кс
2. Строк подання здобувачем роботи 02.06.2023 р.
3. Вихідні дані до роботи
Вимоги користувачів щодо функціональності додатку для використання у кав'ярні.
Технічні вимоги щодо використання додатку у кав'ярні.
Наявність додатку для iOS, написаного на мові програмування Swift з використанням SwiftUI та Core Data Base.

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
1) Актуальність теми
2) Мета та завдання дослідження
3) Вибір технологій та інструментів розробки
4) Розробка функціональності додатку для управління кав'ярні
5) Опис реалізованої функціональності додатку для управління кав'ярні
6) Використання функціональності додатку для управління кав'ярні
7) Висновки про розроблену систему та її перспективи у подальшому використанні та розвитку

5. Перелік графічного матеріалу
   1) Діаграма функціональної моделі бізнес-процесів
   2) Таблиця порівняння існуючих рішень на ринку ІТ
   3) Логічна модель бази даних
   4) Схема БД, структура таблиць
   5) Презентація зі схемою БД і відповідними поясненнями
   6) Діаграма календарного плану виконання роботи

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Самсонов В. В., професор, к.т.н		
2	Самсонов В. В., професор, к.т.н		
3	Самсонов В. В., професор, к.т.н		
4	Самсонов В. В., професор, к.т.н		

7. Дата видачі завдання____04 квітня 2023 року_____

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів виконання кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз потреб та вимог в управлінні кав'ярні	02.05.2023	
2	Вибір технологій та інструментів розробки	09.05.2023	
3	Розробка функціональності додатку	20.05.2023	
4	Тестування та валідація системи	26.05.2023	
5	Написання пояснювальної записки	04.06.2023	

Здобувач _____ Акулов І.О.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Самсонов В.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Метою дипломного проекту є створення додатку для автоматизації процесів управління кав’ярні “Coffeeat33”.

Розробка додатку для автоматизації процесів управління кав’ярні “Coffeeat33” є важливою задачею для підвищення ефективності та якості управлінських процесів.

Об’єктом дослідження є робота кав’ярні, яка займається обслуговуванням клієнтів на ринку спеціалізованих кав’ярень.

Результатом дипломної роботи буде додаток, який забезпечить автоматизацію управлінських процесів у роботі кав‘ярні. Система буде включати функції для створення та редагування розкладу працівників кав’ярні, розрахунку заробітної плати на основі створених графіків та відстежування продуктів та інгредієнтів для повноцінної роботи кав’ярні.

Створення розкладу дозволить зберігати детальну інформацію про кожного працівника, а саме ім’я працівника, дату графіку роботи, початок та кінец робочої зміни. Розрахунок заробітної плати дозволить швидко та точно розраховувати заробітну плату на основі розкладу вибраного працівника.

Система також забезпечить відстежування продуктів та інгредієнтів для контролю наявності усіх потрібних продуктів та інгредієнтів для успішної та повноцінної роботи кав’ярні.

Розроблени додаток має мету поліпшити ефективність та швидкість управління процесами у кав’ярні.

Дипломний проект обсягом у -- сторінок, містить -- сторінок пояснювальної записки, -- сторінок графічних матеріалів, -- літературних джерел.

Ключові слова : СИСТЕМИ АВТОМАТИЗАЦІЇ, ДОДАТОК, КАВ’ЯРНЯ, XCODE, ІНТЕРФЕЙС КОРИСТУВАЧА, ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ, SWIFT, SWIFTUI.

SUMMARY

The aim of the diploma project is to create an application for automating the management processes of the "Coffeeat33" coffee shop.

Developing an application for automating the management processes of the "Coffeeat33" coffee shop is an important task for improving the efficiency and quality of the managerial processes.

The object of research is the operation of a café that serves customers in the specialized café market.

The result of the diploma work will be an application that provides automation of the management processes in the coffee shop. The system will include features for creating and editing the coffee shop staff schedule, calculating wages based on the created schedules, and tracking products and ingredients for the smooth operation of the coffee shop.

Creating a schedule will allow storing detailed information about each employee, including the employee's name, date of the work schedule, and start and end of the shift. Wage calculation will enable quick and accurate calculation of wages based on the schedule of a selected employee.

The system will also track products and ingredients to ensure the availability of all necessary items for the successful and complete operation of the coffee shop.

The developed application aims to improve the efficiency and speed of managing processes in the café.

The diploma project has a total volume of -- pages, including -- pages of explanatory notes, -- pages of graphical materials, and -- literary sources.

Keywords: AUTOMATION SYSTEMS, APPLICATION, CAFÉ, XCODE, USER INTERFACE, INFORMATION TECHNOLOGIES, SWIFT, SWIFTUI.

ЗМІСТ

ВСТУП *1*
РОЗДІЛ 1. СИСТЕМНИЙ АНАЛІЗ КАВ'ЯРНІ "COFFEEAT33" *2*
1.1. Загальна характеристика кав'ярні "Coffeeat33" **2**
1.2. Організаційна структура кав'ярні "Coffeeat33" **2**
1.3. Аналіз нинішнього стану комп'ютеризації "Coffeeat33" **6**
1.4. Розробка функціональної моделі та аналіз існуючих бізнес-процесів **8**
1.5. Обґрунтування доцільності проектування й розроблення "helper." **14**
1.6. Концептуальна модель системи **17**
РОЗДІЛ 2. ТЕХНІЧНЕ ЗАВДАННЯ НА ПРОЄКТУВАННЯ *18*
2.1. Загальні положення. **18**
2.2 Призначення і цілі створення системи. **18**
2.3. Характеристика об'єкта автоматизації. **19**
2.4. Вимоги до системи **19**
2.5. Склад і зміст робіт по створенню системи. **31**
2.6. Порядок контролю і приймання системи. **32**
2.7. Вимоги до додатку і змісту робіт із підготовки до введення системи в дію. **33**
2.8. Вимоги до документації. **34**
2.9. Джерела розробки. **34**
РОЗДІЛ 3. ОПИС КОМПЛЕКСУ ЗАДАЧ АВТОМАТИЗАЦІЇ *35*
3.1. Інформаційне забезпечення системи **35**
3.2. Алгоритмізація та реалізація комплексу задач автоматизації. **39**
3.3. Технічне та системне забезпечення розробки **50**
РОЗДІЛ 4. ОХОРОНА ПРАЦІ *55*
ВИСНОВКИ *56*
БІБЛІОГРАФІЧНИЙ СПИСОК *57*
ДОДАТОК А *60*
ДОДАТОК Б *64*
ДОДАТОК В *66*
ДОДАТОК Г *68*
ДОДАТОК Д *69*

ВСТУП

Сучасні кав’ярні є складною та динамічною галуззю, де швидкість, точність та ефективність грають вирішальну роль у забезпеченні успіху бізнесу. Менеджер кавя’рні, відіграє критичну роль у забезпеченні швидкого та ефективного управління процесами у роботі кав’ярні. Однак, ефективне управління процесами у кав’ярні стає все складнішим завданням у зв'язку зі швидким розвитком сфери обслуговування, а саме кафе та кав’ярень та через плин кадрів.

Одним із ключових аспектів успішного функціонування кав’ярні є належне управління процесами. Потреба у точному та швидкому управлінні виникає на усіх рівнях управління кав’ярні, чи це менеджер, чи власник кав’ярні. Забезпечення точного та ефективного управління процесами кав’ярні є надзвичайно важливим завданням, яке може позитивно вплинути на всю роботу та успішність бізнесу.

У зв'язку з вищезгаданими проблемами та потребами виникає актуальність розроблення додатку для автоматизації деяких процесів в управлінні кав’ярні.

Дипломний проект спрямований на розроблення системи, яка допоможе автоматизувати та спростити управлінські процеси у кав’ярні. Система буде включати функції, які дозволять створювати та редагувати графіки працівників кав’ярні, швидко розраховувати заробітну плату, основуючись на створеному розкладі та розраховувати залишки продуктів та інгредієнтів для повноцінного функціонування кав’ярні.

Розроблений додаток матиме за мету підвищити ефективність в управлінні роботою кав’ярні, поліпшити взаємодію з працівниками кав’ярні та спростити відстежування продуктів та інгредієнтів необхідних для успішного функціонування кав’ярні.

РОЗДІЛ 1. СИСТЕМНИЙ АНАЛІЗ КАВ’ЯРНІ “COFFEEAT33”

1.1. Загальна характеристика кав’ярні “Coffeeat33”

Кав'ярня "Coffeeat33" відкрилася в 2008 році і з того часу стала однією з улюблених місцевих спеціалізованих кав'ярень. Кав'ярня "Coffeeat33" розташована за адресою 33 Trafalgar St, Brighton, Brighton and Hove, Brighton BN1 4ED, Велика Британія. Заснована спільно Тарасом Тунським та Америго Фусі. Тарас виступає ростером: він почав обсмажувати на 2-кілограмовому обсмажувальному обладнанні Novoroaster Air, виготовленому в Німеччині, пізніше перейшов на 10-кілограмову систему Toper у 2017 році і наразі працює над обсмажуванням у компанії Horsham Coffee Roasters на їхньому 35-кілограмовому обладнанні Loring.

1.2. Організаційна структура кав’ярні “Coffeeat33”

Організаційна структура є важливим елементом ефективного управління кав'ярнею "Coffeeat33". Вона визначає ієрархію, ланцюжки командування та відносини між різними ролями та посадами в організації. Кав'ярня "Coffeeat33" має структуру, яка побудована на основі функціональних підрозділів та принципу ієрархії.

Рис.1.1 Ієрархічна система управління в кав'ярні «Coffeeat33»

Власники

Власники, Тарас Тунський та Америго Фусі, відіграють ключову роль у прийнятті стратегічних рішень та керуванні загальною діяльністю кав'ярні. Вони відповідають за визначення мети, виробничої політики та розвитку бізнесу. Власники забезпечують фінансову стабільність, встановлюють бюджет та контролюють фінансові показники кав'ярні "Coffeeat33".

Менеджер

Менеджер відіграє важливу роль у керуванні повсякденною діяльністю кав'ярні. Він забезпечує ефективне функціонування всіх підрозділів, координує роботу персоналу та контролює процеси виконання завдань. Менеджер відповідає за встановлення стандартів обслуговування, забезпечення якості продукції та задоволення потреб клієнтів.

Бариста, пекар, обсмажувач та бухгалтер

Бариста відповідає за приготування кавових напоїв та забезпечення високої якості обслуговування клієнтів. Він володіє навичками професійного кавоваріння та забезпечує свіжість та смакові властивості кави.

Пекар відповідає за приготування свіжої випічки та десертів для клієнтів кав'ярні "Coffeeat33". Він забезпечує високу якість продукції, враховуючи смакові уподобання та вимоги клієнтів.

Обсмажувач має важливу роль у процесі обсмажування кавових зерен. Він знаходиться за постачанням високоякісних кавових зерен та відповідає за створення унікальних смакових профілів кави.

Бухгалтер виконує фінансовий облік, складає звітність та забезпечує ведення бухгалтерського обліку в кав'ярні "Coffeeat33". Він контролює фінансові показники, оптимізує витрати та забезпечує дотримання фінансових процедур.

Завдання і функції кав'ярні "Coffeeat33" наведені в табл. 1.1.

Табл. 1.1. Функції та відповідні задачі управління кав'ярнею "Coffeeat33"

№	Задачі	Функції
1	Приймання замовлень від клієнтів	- реєстрація та обробка замовлень - взаємодія з клієнтами для уточнення деталей замовлення - ведення документації та оформлення замовлень
2	Планування та управління запасами	- аналіз попиту та прогнозування потреб - визначення оптимального рівня запасів - поповнення запасів та контроль за запасами
3	Управління постачанням	- встановлення контактів з постачальниками - переговори про умови постачання та ціни - забезпечення своєчасної доставки товарів
4	Управління персоналом	- планування робочого графіку та розкладу - навчання працівників - оцінка та мотивація працівників
5	Маркетинг та реклама	- розробка та виконання маркетингових стратегій - реклама та просування кав'ярні - взаємодія зі спільнотою та залучення нових клієнтів
6	Фінансове управління	- бюджетування та фінансовий аналіз - контроль витрат та доходів

Організаційна структура кав'ярні "Coffeeat33" передбачає систему комунікації та зв'язків між всіма рівнями та посадами. Це забезпечує ефективний обмін інформацією, сприяє координації роботи та покращує командний дух у колективі.

Організаційна структура кав'ярні "Coffeeat33" побудована на основі ієрархічної моделі, що відображає різні рівні власницького, керівного та виконавчого персоналу. Кожна посада відіграє важливу роль у забезпеченні ефективного функціонування кав'ярні та задоволення потреб клієнтів. Комунікація

та зв'язки між різними рівнями та посадами сприяють гармонійній роботі колективу і досягненню спільних цілей.

1.2.1. Структура менеджменту в “Coffeeat33”

Структура менеджменту визначає організаційну архітектуру і розподіл функцій та відповідальностей у компанії. Вона встановлює ієрархічні зв'язки, комунікаційні ланцюжки та ролі різних підрозділів і посад у компанії. Основна мета структури менеджменту полягає у забезпеченні ефективного управління, координації роботи та досягненні поставлених цілей.

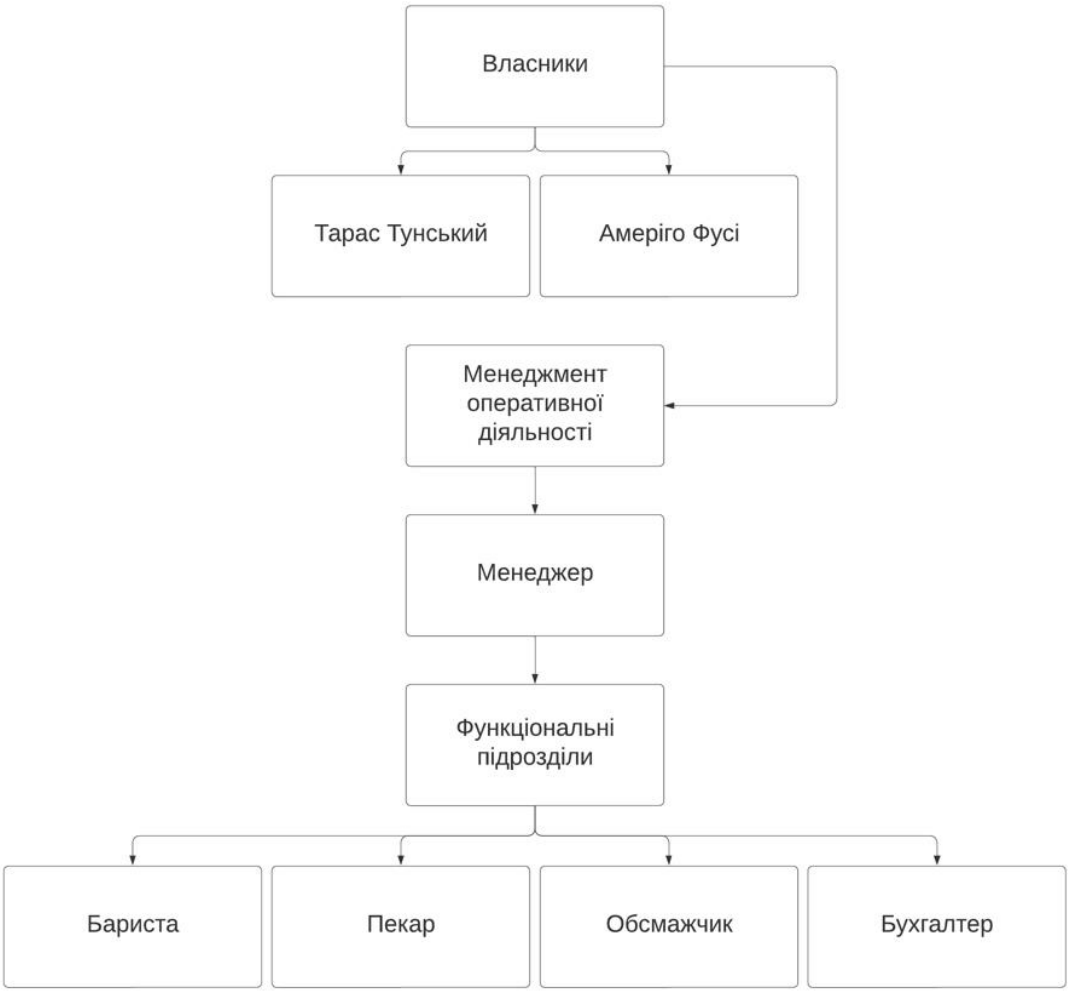
Структура менеджменту включає різні рівні управління, такі як виконавчий рівень, середній рівень керівництва та стратегічний рівень. На виконавчому рівні зазвичай розташовуються оперативні підрозділи, відповідальні за безпосереднє виконання завдань та функцій компанії. На середньому рівні можуть знаходитись підрозділи з управління персоналом, фінансами, маркетингом тощо. Стратегічний рівень включає вище розташоване керівництво компанії, яке визначає стратегію розвитку, приймає стратегічні рішення та встановлює загальні напрямки роботи компанії.

Організаційна структура кав’ярні “Coffeeat33” є функціональною. У функціональній структурі кав’ярні підрозділи організовуються за функціональними областями, такими як власники, менеджмент оперативної діяльності та функціональні підрозділи, що обслуговують клієнтів.

Крім того, структура менеджменту включає посадові обов'язки та відповідальності керівників на різних рівнях управління. Наприклад, виконавчий директор відповідає за загальне керівництво компанією, приймає стратегічні рішення та забезпечує виконання поставлених цілей. На рівні середнього керівництва можуть бути менеджери відділів або підрозділів, відповідальні за

конкретні функціональні області. Кожна посада має свої визначені обов'язки, повноваження та зв'язки з іншими посадами.

Рис. 1.2. Організаційна та функціональна структура менеджменту в "Coffeeat33"



1.3. Аналіз нинішнього стану комп'ютеризації "Coffeeat33"

У сучасному світі комп'ютеризація відіграє важливу роль у багатьох галузях, включаючи сферу ресторанного бізнесу. "Coffeeat33", як один із провідних закладів в галузі кавової індустрії, також звертає увагу на впровадження технологій для поліпшення своєї діяльності. Проаналізуємо нинішній стан комп'ютеризації "Coffeeat33" з метою виявлення потенційних можливостей та проблем, що потребують уваги.

Автоматизація процесів:

На сьогоднішній день "Coffeeat33" вже використовує комп'ютеризовані системи для деяких аспектів своєї діяльності. Наприклад, наявність касової системи дозволяє ефективно обробляти замовлення, відстежувати продажі та контролювати фінансові показники. Однак, є можливість розширення автоматизації і впровадження нових інструментів, таких як система управління графіками роботи персоналу, відстеження наявності продукції на складі та керування інформацією щодо заробітної плати.

Управління персоналом:

З урахуванням розміру і складності "Coffeeat33" важливо мати ефективну систему управління персоналом. Застосування комп'ютеризованої системи для відслідковування графіка роботи, контролю за відсутностями та заробітною платою допоможе менеджерам ефективно планувати ресурси та забезпечувати належний режим роботи закладу.

Менеджмент запасів:

Кавова індустрія потребує точного контролю за наявністю і рухом продукції. Впровадження комп'ютеризованої системи управління запасами дозволить "Coffeeat33" точно відслідковувати кількість сировини, складових компонентів, витратних матеріалів та готової продукції. Це забезпечить ефективне планування постачання, уникнення недостачі або перевищення запасів і зменшення втрат.

Взаємодія з клієнтами:

У сучасному ресторанному бізнесі важливо мати ефективні інструменти взаємодії з клієнтами. Впровадження комп'ютеризованих систем для збору зворотного зв'язку, обробки замовлень та програм лояльності може значно поліпшити якість обслуговування та задоволення клієнтів.

Аналітика та звітність:

Використання комп'ютеризованих інструментів для збору та аналізу даних допоможе "Coffeeat33" управляти своєю діяльністю на основі фактів і покращувати стратегію розвитку. Звіти про продажі, фінансову продуктивність, популярність

конкретних позицій меню та інші аналітичні дані допоможуть в прийнятті обґрунтованих рішень та виявленні нових можливостей для покращення бізнесу.

Враховуючи ці аспекти, виявляється потенціал для розробки програмного додатку, який буде допомагати "Coffeeat33" у вищезазначених сферах. Цей додаток може забезпечити зручний доступ до інформації, автоматизувати рутинні процеси та забезпечити ефективне управління ресурсами. Такий додаток буде сприяти покращенню продуктивності, якості обслуговування та стратегічного розвитку "Coffeeat33".

1.4. Розробка функціональної моделі та аналіз існуючих бізнес-процесів

У рамках розробки програмного додатку для "Coffeeat33" з метою поліпшення управління персоналом, контролю за наявністю продукції та керування інформацією щодо заробітної плати, необхідно провести розроблення функціональної моделі та проаналізувати існуючі бізнес-процеси. Це допоможе зрозуміти потреби та вимоги компанії і визначити функціональні можливості та особливості додатку.

Розроблення функціональної моделі:

Першим кроком у розробці додатку є створення функціональної моделі, яка відображає основні функції та зв'язки між компонентами системи. Ця модель визначає, які можливості будуть вбудовані в додаток і як вони будуть взаємодіяти між собою. Функціональна модель включатиме наступні елементи:

- Менеджерський доступ: Додаток повинен мати окремий доступ для менеджерів, які відповідають за управління персоналом та бізнес-процесами.
- Графік роботи персоналу: Додаток повинен надавати можливість створювати та відслідковувати графік роботи персоналу, включаючи розподіл змін, відпустки та відсутності.
- Наявність продукції: Додаток повинен забезпечувати моніторинг наявності продукції на складі, автоматичне оновлення запасів і сповіщення про нестачу або необхідність поповнення.

- Керування заробітною платою: Додаток повинен дозволяти вести облік робочого часу, розраховувати заробітну плату персоналу та забезпечувати зручний доступ до відповідної інформації.

Аналіз існуючих бізнес-процесів:

Для успішної розробки додатку необхідно аналізувати існуючі бізнес-процеси в "Coffeeat33" та виявити потенційні проблеми, недоліки та можливості для поліпшення. Деякі з ключових бізнес-процесів, які можуть бути включені до аналізу, включають:

- Управління персоналом: Аналізується поточна система управління персоналом, включаючи розподіл робочого часу, контроль за відсутностями та розрахунок заробітної плати.
- Контроль за запасами: Аналізується поточна система контролю за запасами продукції, включаючи складський облік, замовлення та постачання.
- Фінансовий облік: Аналізується поточна система фінансового обліку, включаючи контроль продажів, звіти та обробку платежів.
- Взаємодія з клієнтами: Аналізується поточна система взаємодії з клієнтами, включаючи замовлення, програми лояльності та обробку зворотного зв'язку.

Аналіз існуючих бізнес-процесів допоможе виявити потреби та проблеми, які можна вирішити за допомогою розробки програмного додатку. Результати аналізу будуть використані для визначення функціональних вимог і розроблення оптимального рішення, що задовольнятиме потреби "Coffeeat33" і сприятиме покращенню їх бізнес-процесів.

1.4.1 Виявлені проблеми

Управління кав'ярнею включає ряд процесів, які спрямовані на ефективне функціонування закладу, задоволення потреб клієнтів та досягнення успішної фінансової продуктивності. Ці процеси охоплюють різні аспекти управління, включаючи замовлення сировини, приготування напоїв, обслуговування клієнтів,

фінансовий облік, управління персоналом та взаємодію з постачальниками. Розглянемо кожен з цих процесів детальніше та визначимо проблеми, які притаманні кожному з них:

Замовлення сировини:

- Визначення потреби в сировині на основі продажів, прогнозування та інших факторів.
- Створення замовлення відповідно до встановлених стандартів якості та кількості.
- Взаємодія з постачальниками для оформлення та доставки замовлення.
- Контроль якості отриманої сировини.

Проблема: недостатня наявність сировини.

Відсутність або недостатня кількість необхідних інгредієнтів для приготування напоїв може призвести до перебоїв в обслуговуванні клієнтів та втрати потенційних продажів. Це може бути спричинене неправильним прогнозуванням попиту, проблемами з постачальниками або неефективним управлінням запасами.

Обслуговування клієнтів:

- Прийом та обробка замовлень від клієнтів.
- Підготовка та подача напоїв та страв згідно з вимогами клієнтів.
- Надання додаткової інформації про асортимент, акції та інші послуги.
- Взаємодія з клієнтами для збору відгуків та вирішення можливих проблем.

Проблема: задоволення клієнтів.

Центральним елементом кав'ярні є задоволення потреб і очікувань клієнтів. Недостатній рівень обслуговування, якість продукції або недружній підхід до клієнтів можуть призвести до втрати потенційних та постійних клієнтів.

Фінансовий облік:

- Фіксація продажів та відповідних оплат в системі обліку.

- Складання фінансової звітності, включаючи звіти про продажі, витрати та прибуток.
- Аналіз фінансових показників для оцінки ефективності кав'ярні та прийняття управлінських рішень.

Проблема: фінансовий контроль.

Недостатня увага до фінансового обліку та контролю може спричинити проблеми зі збалансованістю бюджету, неправильними розрахунками маржі прибутку та нездатністю вчасно виявляти фінансові витоки або неправомірне використання коштів.

Управління персоналом:

- Розподіл робочих годин та графіків роботи співробітників.
- Навчання та тренінги персоналу щодо приготування напоїв, обслуговування клієнтів та дотримання стандартів якості.
- Здійснення оцінки продуктивності персоналу та надання конструктивного фідбеку.
- Розробка та впровадження політик збереження та розвитку персоналу.

Проблема: нестабільний робочий графік

Управління робочим графіком персоналу може бути складним завданням, особливо в періоди пікового попиту. Недостатня кількість працівників або неправильне розподілення робочих годин можуть призвести до затримок у обслуговуванні, незадоволення клієнтів та навіть втрати покупців.

Цей аналіз процесів управління кав'ярнею "Coffeeat33" дозволяє зрозуміти ключові етапи та функції, які відбуваються у кав'ярні. Він допомагає ідентифікувати можливі проблеми та пропонує основу для розроблення додатку для менеджера, який оптимізує робочий графік, відслідковує наявність використованої продукції та полегшує управління процесами у кав'ярні.

Ці проблеми можуть мати негативний вплив на ефективність та прибутковість кав'ярні "Coffeeat33". Недостатня наявність сировини може призводити до перебоїв у постачанні та втрати потенційних продажів. Нестабільний робочий графік може впливати на якість обслуговування та задоволення клієнтів. Недостатня координація між відділами може призводити до затримок та неправильного використання ресурсів. Недостатній фінансовий контроль може призвести до фінансових проблем та неефективного використання коштів. Важливо розуміти ці проблеми та шукати рішення, що допоможуть покращити управління кав'ярнею, забезпечити якісне обслуговування клієнтів, оптимізувати робочі процеси та досягти більш стабільних фінансових результатів.

1.4.2. Задачі автоматизації

Автоматизація в бізнесі має на меті впровадження технологій та систем, що заміняють або полегшують ручні процеси, щоб досягти ефективності та покращити результативність. Основні задачі автоматизації включають:

Управління графіком роботи персоналу: Розробка функціоналу, що дозволить менеджеру створювати та змінювати графіки роботи працівників, враховуючи розклади, пріоритети та доступність персоналу.

Керування наявністю продукції: Реалізація функцій, що дозволять вести облік запасів сировини та готової продукції, контролювати рівень запасів і спрогнозувати потребу в поповненні. Система може надати можливість нагадування про потребу в поповненні запасів.

Управління інформацією щодо заробітної плати: Розробка функцій для обліку та розрахунку заробітної плати персоналу, враховуючи години роботи, ставки оплати та інші фактори. Система може автоматично, за допомогою влаштованих формул, розраховувати заробітну плату та генерувати звіти.

Забезпечення зручного інтерфейсу для менеджера: Розробка інтуїтивно зрозумілого та зручного інтерфейсу, що дозволяє менеджеру легко взаємодіяти з

додатком, вносити зміни в графіки роботи персоналу, переглядати статус запасів та заробітної плати, отримувати звіти та статистику.

Забезпечення безпеки та захисту даних: Розробка механізмів безпеки для зберігання та обробки конфіденційної інформації, такої як персональні дані працівників і фінансова інформація. Система повинна мати вбудовані механізми автентифікації, резервне копіювання даних та захист від несанкціонованого доступу.

Загалом, задачі автоматизації будуть спрямовані на полегшення та оптимізацію управління графіком роботи, наявністю продукції та інформацією про заробітну плату, забезпечення точності, ефективності та зручності в роботі.

При розгляді існуючих рішень для розв'язання виявлених проблем в компанії "Coffeeat33", варто звернути увагу на наступні можливості:

Підвищення ефективності управління запасами: Використання спеціалізованих програм або систем управління запасами може допомогти відстежувати наявність сировини та готової продукції, прогнозувати потреби в запасах, автоматизувати процес замовлення та контролювати запаси. Приклади таких систем включають Point of Sale (POS) системи, програми обліку запасів та системи управління логістикою.

Впровадження системи управління графіком роботи: Існують спеціалізовані програми для планування та управління робочим графіком персоналу. Ці системи дозволяють створювати графіки роботи, враховуючи доступність персоналу, попереджати про можливі конфлікти в графіку, надавати сповіщення про зміни, а також вести облік робочих годин та заробітної плати. Такі системи можуть значно спростити процес планування та координації роботи персоналу.

Використання програмного забезпечення для фінансового обліку: Додатки для фінансового обліку можуть допомогти в управлінні фінансами компанії, включаючи облік заробітної плати. Вони дозволяють автоматизувати розрахунки, генерувати звіти, відстежувати доходи та витрати, а також контролювати фінансову стабільність

підприємства. Важливо обрати програмне забезпечення, яке відповідає потребам і розміру вашого бізнесу.

Впровадження системи управління відносинами з клієнтами (CRM): CRM системи дозволяють ефективно керувати взаємодією з клієнтами, відстежувати їхні замовлення, передплати та зворотний зв'язок. Вони дозволяють покращити якість обслуговування, персоналізувати пропозиції для клієнтів та збільшити їхню задоволеність. CRM системи можуть бути використані для збільшення лояльності клієнтів та залучення нових.

Використання мобільних додатків для забезпечення доступу до інформації: Розробка мобільного додатка для компанії "Coffeeat33" може дозволити менеджерам та працівникам швидко отримувати доступ до важливої інформації, такої як графік роботи, наявність продукції, заробітна плата та звіти. Мобільний додаток забезпечить зручність та ефективність управління, особливо в умовах мобільності та віддаленого роботи.

Враховуючи виявлені проблеми в компанії "Coffeeat33", розгляд цих існуючих рішень може надати основу для подальшої розробки додатку, який буде спрямований на вирішення конкретних потреб компанії. Важливо аналізувати переваги, недоліки, вартість та сумісність рішень, щоб забезпечити оптимальний вибір та успішну імплементацію.

1.5. Обґрунтування доцільності проектування й розроблення “helper.”

Для досягнення успіху у впровадженні системи необхідно докладно вивчити потреби та вимоги користувачів, а також вибрати відповідний технологічний стек.

Першим кроком є аналіз потреб користувачів. Це включає збір інформації про вимоги різних груп користувачів, таких як менеджери, бариста, пекарі, бухгалтери та інші. Важливо визначити, які функціональні можливості повинна мати система для задоволення потреб кожної групи користувачів. Наприклад, менеджер може потребувати інструментів для управління графіками роботи, замовленням сировини та відслідковуванням фінансових показників, тоді як бариста може потребувати зручного інтерфейсу для приготування напоїв та ведення обліку складових.

Зважаючи на те, що розроблюваний додаток призначений для менеджера кав'ярні "Coffeeat33", варто провести аналіз потреб цільового користувача. Нижче наведена таблиця, яка візуально демонструє основні потреби менеджера та функціональні вимоги до додатку.

Таб. 1.3. Потреби менеджера у кав'ярні та вимоги до додатку.

№	**Потреби менеджера**	**Функціональні вимоги до додатку**
1	Управління графіками роботи	Можливість створення, зміни та перегляду графіків роботи працівників.
2	Управління персоналом	Функціонал для керування персоналом, включаючи розрахування зарплатні на основі робочого графіку тощо.
3	Моніторинг складу	Функціонал для контролю та оновлення запасів на складі, включаючи сповіщення про нестачу або надмір запасів.

Аналізуючи цю таблицю, можна зрозуміти, що менеджеру кав'ярні "Coffeeat33" потрібен додаток, який надасть зручний інструмент для управління різними аспектами їх роботи. Додаток повинен мати інтуїтивно зрозумілий інтерфейс, надійність, а також функціонал, що відповідає вищезазначеним потребам менеджера.

Обґрунтування доцільності проектування й розроблення додатку "helper." для компанії "Coffeeat33" базується на наступних факторах:

Автоматизація та ефективність робочих процесів: Розробка додатку "helper" дозволить автоматизувати багато рутинних завдань, що присутні в роботі кав'ярні.

Наприклад, відслідковування графіка роботи персоналу, контроль за наявністю продукції та керування інформацією щодо заробітної плати можуть бути зручно здійснюватись через додаток. Це дозволить зменшити час, зусилля та можливість помилок, що виникають при виконанні цих завдань вручну.

Покращення обслуговування клієнтів: Додаток “helper." може сприяти покращенню якості обслуговування клієнтів. Швидкий доступ до інформації про наявність продукції, замовлення та історії клієнтів допоможе персоналу бути більш ефективним та забезпечити кращу обробку замовлень. Крім того, можливість персоналізованого підходу до клієнтів, наприклад, шляхом збереження їхніх уподобань та історії замовлень, зробить їх досвід відвідування кав'ярні більш задоволеним.

Покращення управління ресурсами: Додаток "helper." може допомогти у кращому управлінні ресурсами, такими як сировина, працівники та фінанси. Він дозволить вчасно виявляти потреби в сировині, оптимізувати робочий графік персоналу, контролювати заробітну плату та витрати. Це сприятиме зменшенню витрат, уникненню перебоїв та покращенню ефективності використання ресурсів.

Зростання конкурентоспроможності: Впровадження додатку "helper." може покращити конкурентоспроможність компанії "Coffeeat33" на ринку. Завдяки автоматизованому та ефективному управлінню, кав'ярня зможе швидше реагувати на зміни в попиті, забезпечувати високу якість обслуговування та підтримувати задоволеність клієнтів. Це дозволить залучати нових клієнтів та зберігати постійну клієнтську базу.

Отже, розробка додатку "helper." має чіткі переваги для компанії "Coffeeat33". Його впровадження дозволить автоматизувати багато процесів, покращити обслуговування клієнтів, ефективно управляти ресурсами та збільшити конкурентоспроможність компанії на ринку.

1.6. Концептуальна модель системи

Концептуальна модель системи "helper." для компанії "Coffeeat33" включає такі основні компоненти та їх взаємодію:

Користувачі:

- Власник: має повний доступ до всіх функцій та може керувати системою, включаючи управління персоналом, розкладом роботи, товарною наявністю та фінансами.
- Менеджери: відповідають за керування конкретними відділами, включаючи кухню, обслуговування клієнтів та бухгалтерію.
- Бариста та пекарі: виконують безпосередню роботу з приготування напоїв та випічки.
- Бухгалтер: відповідає за фінансовий облік і розрахунки.

Модуль управління ресурсами:

- Сировина: система відстежує наявність необхідних інгредієнтів для приготування напоїв та випічки, сповіщає про нестачу та допомагає замовляти у постачальників.
- Робочий графік: система дозволяє менеджерам створювати графіки роботи для персоналу, враховуючи пікові та звичайні години роботи, та дозволяє персоналу вносити зміни у графік.
- Заробітна плата: система автоматично обраховує заробітну плату на основі робочого часу та ставок, дозволяє вести облік виплат та видачу зарплатних відомостей.

Ця концептуальна модель системи "helper." надає загальний огляд функцій та взаємозв'язків системи. Детальніші технічні характеристики та функціональні можливості можуть бути визначені під час розробки та вибору конкретних інструментів та технологій.

РОЗДІЛ 2. ТЕХНІЧНЕ ЗАВДАННЯ НА ПРОЄКТУВАННЯ

2.1. Загальні положення.

1.1. Найменування системи: Інформаційна система підтримки управління кав'ярні "Coffeeat33". Це технічне завдання (далі – ТЗ) розроблено у відповідності з вимогами ДСТУ 34.602-89 і є основним документом, що визначає вимоги та порядок проектування, розробки та впровадження інформаційної системи кав'ярні" Coffeeat33".

1.2. Результати робіт зі створення системи оформлюються згідно з вимогами ДСТУ на відповідні етапи розробки. Порядок оформлення і передачі результатів у даному випадку визначається змістом і календарним планом виконання розробки.

1.3. У випадку необхідності на наступних стадіях робіт по створенню системи окремі положення можуть уточнюватися і розвиватися.

2.2 Призначення і цілі створення системи.

2.2.1. Призначення системи.

Призначення інформаційної системи підтримки управління кав'ярні" Coffeeat33" полягає у поліпшенні управління менеджерськими обов'язками у закладі та підвищенні ефективності роботи. Застосування інформаційних технологій дозволяє автоматизувати процеси створення графіків для працівників, обробки та аналізу даних про остатки продуктів та інгредієнтів, що знижує ризик помилок та дозволяє швидше виявляти проблемні ситуації.

2.2.2 Цілі створення системи.

Основною метою створення системи є забезпечення оптимізації процесів управління, включаючи створення графіків, розрахунок заробітної платні та розрахунок остатків продуктів та інгредієнтів у кав'ярні.

Це забезпечить створення умов для поліпшення діяльності кав'ярні та підвищить продуктивність.

2.3. Характеристика об’єкта автоматизації.

2.3.1. Короткі відомості про об’єкт автоматизації.

Об’єктом автоматизації є діяльність менеджера у спешалті кав’ярні “Coffeeat33”, а саме процеси створення графіків для працівників, обробки та аналізу даних про остатки продуктів та інгредієнтів, та розрахунок заробітної платні на основі створених графиків.

2.4. Вимоги до системи

2.4.1. Вимоги до системи в цілому.

Система повинна полегшувати процеси менеджера, а саме: передбачати додавання та видалення розкладу змін працівників, розрахунок заробітної платні основуючись на створених розкладах та розраховувати остатки інгредієнтів та продуктів.

2.4.1.1. Вимоги до структури і функціонування системи.

2.4.1.1.1. Система повинна мати клієнт-серверну архітектуру, що використовує єдину базу даних (надалі — БД).

Згідно з функціональною структурою додатку, система повинна бути пов’язана в мережі з:

- Інтерфейсом;
- Базою даних;
- Менеджером кав’ярні;
- Працівниками кав’ярні;

2.4.1.1.2. Діагностування функціонування системи додатку має передбачати виявлення відхилень від нормального процесу розв’язання задач і
порушень у роботі комп’ютерно-технічних засобів, а також програмних помилок, забезпечуючи користувачів відповідними діагностичними повідомленнями.

2.4.1.1.3. Розвиток і модернізація системи повинні проводитися шляхом проведення детального аналізу поточного стану системи та визначення вимог і

потреб щодо керування процесом управління кав’ярні. Це включає збір вимог від працівників, ідентифікацію слабких місць і проблем, а також визначення цілей та очікуваних результатів модернізації.

Розроблення детального плану модернізації, який включає в себе кроки, ресурси, бюджет і графік виконання проекту. План повинен враховувати усі аспекти, такі як апаратне забезпечення, програмне забезпечення, комунікації, безпеку даних і навчання персоналу.

Головні програмно-технічні засоби для функціонування системи керування процесом управління кав’ярні повинні включати надійну та масштабовану інформаційну базу даних та засоби для аналітики та звітності.

2.4.1.1.4. Функціонування системи має забезпечувати діалогову та мережну (розподілену) обробку даних.

2.4.1.2. Вимоги до чисельності і кваліфікації персоналу.

Менеджер повинен мати розуміння усіх процесів управління, вміти використовувати інформаційну систему, бути знайомим з технологіями відстеження, базами даних та аналітичними інструментами. Навички комунікації, аналітичного мислення та проблемного вирішення також є важливими. Досвід у сфері управління проектами є бажаним. Знання про процеси управління, правила і стандарти, а також попередній досвід в використанні інформаційних систем для автоматизування управління кав’ярні буде перевагою.

2.4.1.2.1. Персонал, який використовує автоматизовану систему, повинен дотримуватися наступних вимог:

- пройти навчання і отримати навички роботи на пристрої з iOS або macOS;
- дотримуватись технологічних інструкцій при роботі з додатком;
- дотримуватись умов експлуатації пристрою у відповідності з інструкціями по експлуатації;
- дотримуватись правил зберігання інформації і організації резервних копій БД;

- дотримуватись правил техніки безпеки при роботі з пристроями.

2.4.1.2.2. Користувачами додатку можуть виступати власники або менеджери. Вхід у додаток повинен здійснюватися через пароль, який відображає рівень користувача: із правом створення та коригування БД.

2.4.1.3. Показники призначення

Відповідно до п. 2.1, показники призначення повинні характеризувати ступінь та якість управління кав’ярні “Coffeeat33” для його оптимального функціонування. Перелік і допустимі значення показників, при яких зберігається цільове призначення системи, повинні бути визначені на стадії проектування.

2.4.1.4. Вимоги до надійності.

2.4.1.4.1. Додаток є багатофункціональною і призначена для використання протягом робочого дня. Всі функції додатка виконуються дискретно. У відповідності з ДСТУ 2226-93 оцінка надійності проводиться по кожній функції окремо. Враховуючи особливості функціонування системи, показники її надійності є показниками надійності СУБД, на якій вона реалізована, та технічних засобів, на яких вона експлуатується. Вимоги до надійності додатку управління у кав’ярні включають високу доступність, резервне копіювання, захист від втрати даних, швидке відновлення, масштабованість та систематичне тестування та монітори.

2.4.1.4.2. Комплекс технічних засобів повинен передбачати:

1. Програмування функціоналу: Додаток надає зручний інтерфейс для розробки та налаштування функцій кав'ярні, включаючи взаємодію з користувачами.
2. Захищеність даних: Система забезпечує безпеку та конфіденційність інформації, забезпечуючи безпечне збереження та обробку даних.
3. Інтеграцію з різноманітними платіжними системами, що дозволить користувачам здійснювати оплату товарів і послуг з використанням зручного та безпечного інтерфейсу додатку.

4. Функціонал для моніторингу та аналізу статистики кав’ярні, замовлень та поведінки робітників, отримувати цінну інформацію для вдосконалення своїх послуг та стратегії розвитку.
5. Засоби для автоматизації процесів обробки та створення розкладів, що забезпечать швидке та ефективне виконання операцій у кав’ярні.
6. Систему перевірки наявності продуктів та інгредієнтів у додатку, що дозволить оперативно приймати рішення щодо необхідності тих чи інших поповнень продуктів.

2.4.1.5. Вимоги до безпеки.

- Контроль доступу: Забезпечення обмеження доступу до системи за допомогою паролів, управління правами користувачів та механізмів перевірки автентичності.
- Шифрування даних: Використання шифрування для захисту конфіденційної інформації, яка передається по мережі або зберігається в системі.
- Аудит та моніторинг: Ведення журналів подій та відстеження активності в системі з метою виявлення потенційних загроз безпеці та виявлення вразливостей.
- Захист від зламу: Впровадження заходів для запобігання несанкціонованому доступу, таких як використання файрволів, захисту від вірусів, протидії хакерським атакам тощо.
- Фізична безпека: Забезпечення фізичної безпеки обладнання та інфраструктури, що використовуються в системі, наприклад, контроль доступу до серверних приміщень.
- Безпека даних: Захист даних від втрати, пошкодження або крадіжки шляхом проведення резервного копіювання, використання реплікації та механізмів відновлення.

- Захист від внутрішніх загроз: Реалізація заходів безпеки для запобігання внутрішнім загрозам, таким як несанкціонований доступ співробітників або витік конфіденційної інформації.

2.4.1.6. Вимоги з ергономіки та технічної естетики.

Загальні ергономічні і естетичні вимоги до системи повинні відповідати держстандартам ДСТУ 8604:2015, ДСТУ 7298:2013. Освітленість робочого місця повинна відповідати ДСТУ EN 12464-1:2016, ДБН В.2.5-28-2006.

Засоби відображення повинні розміщуватися таким чином, щоб кут спостереження екрану складав не більше, ніж 45 градусів, мінімальна відстань спостереження екрану — 0,3 м, рекомендована — 0,5 м.

При розробленні ПЗ слід створити зручний інтерфейс для запобігання втомлюваності користувача.

2.4.1.7. Вимоги по експлуатації, технічного обслуговування, ремонту і зберігання компонентів системи.

2.4.1.7.1. Види обслуговування системи визначаються у відповідності з ДСТУ EN 13306:2019. Загальні вимоги по експлуатації, технічному обслуговуванню і ремонту повинні відповідати ДСТУ 3576-97.

2.4.1.7.2. Для розміщення технічних засобів системи необхідні площі, визначені в ДБН В.2.2-9-2009. При цьому слід дотримуватися вимог, зазначених в експлуатаційній документації. Напруга живлення технічних засобів системи 220/380 В змінного струму, частотою (50±1) Гц. Допустиме відхилення напруги від +10 до –15%, тривалість перерв у живленні не повинна перевищувати 0,001 с.

2.4.1.7.3. Кількість, кваліфікація і режими роботи обслуговуючого персоналу повинні відповідати рекомендаціям, зазначеним в технічних умовах і інструкціях з експлуатації окремих ТЗ.

2.4.1.7.4. Склад, розміщення і умови зберігання компонентів технічних засобів системи визначається рекомендаціями, зазначеними в експлуатаційній документації на ці елементи.

2.4.1.7.5. Регламент обслуговування повинен відповідати їх рівню і умовам роботи, щоб у випадку відмови системи забезпечити роботу в аварійному режимі.

2.4.1.8. Вимоги до захисту інформації від несанкціонованого доступу.

Вимоги до захисту інформації від несанкціонованого доступу передбачають ефективну автентифікацію та ідентифікацію користувачів, використання сильних паролів та механізмів керування доступом, шифрування конфіденційної інформації, захист мережевого зв'язку та даних під час передачі, а також фізичний захист серверних приміщень та обладнання.

2.4.1.9. Вимоги щодо збереження інформації при аваріях.

2.4.1.9.1. Необхідно передбачити засоби резервного збереження БД в архіві після коригування і можливість завантажити БД з архіву у випадку її руйнування.

2.4.1.9.2. Резервний архів і БД мають знаходитись на різних машинних носіях чи пристроях.

2.4.1.10. Вимоги по захисту від впливу зовнішніх діянь.

2.4.1.10.1. Електрична складова електромагнітного поля завад в приміщеннях не повинна перевищувати 0,3 В/м2 в діапазоні частот від 0,15 до 300 МГц. Для захисту від впливу електромагнітних полів та індустріальних завад слід передбачити різноманітні екрани та фільтри.

2.4.1.10.2. Засоби, які виключають вплив шкідливих факторів на функціонування комплексу технічних засобів, повинні бути запроектовані згідно з ДБН В.2.2-9-2009. Обчислювальні засоби по стійкості до зовнішніх впливів повинні відповідати ДСТУ 2506-94.

2.4.1.11. Вимоги до патентної чистоти.

При створенні даної системи патентні дослідження не проводяться.

2.4.1.12. Вимоги по стандартизації і уніфікації.

Вимоги щодо стандартизації та уніфікації передбачають встановлення загальноприйнятих норм і стандартів, які регулюють процеси, протоколи та формати обміну даними. Основні мети цих вимог - забезпечити сумісність та інтеграцію різних компонентів системи, а також спростити комунікацію та взаємодію між працівниками кав'ярні.

Вимоги по стандартизації і уніфікації включають наступні аспекти:

- Стандарти обміну даними: Встановлення загальноприйнятих форматів та протоколів для обміну даними між різними системами, що забезпечує сумісність та інтеграцію.
- Стандартизація процесів: Розроблення нормативних документів, які визначають стандартні процедури та методи роботи у кав’ярні, спрощуючи спілкування та забезпечуючи єдність дій.
- Ідентифікаційні стандарти: Встановлення загальноприйнятих методів ідентифікації продуктів, інгредієнтів, товару та інших об'єктів, що дозволяє їх однозначно визначати та відстежувати.
- Безпекові стандарти: Розроблення норм та правил щодо захисту інформації, фізичної безпеки, контролю доступу та інших аспектів безпеки у кав’ярні.

  Стандартизація мови та термінології: Використання загальноприйнятих термінів, визначень та мови комунікації, що дозволяє уникнути непорозумінь та покращує ефективність спілкування.

2.4.2. Вимоги до функцій.

2.4.2.1. Перелік функцій із зазначенням вхідної та вихідної інформації наведено в таблиці 1.

Функції мають забезпечити раціональну організацію роботи користувача на основі безперервної технології: заповнення БД, довідників, формування різнорідних звітів і виконання інших функцій, визначених чинним документом. При цьому пріоритетом є зручність введення та використання інформації користувачем за рахунок формування підказок і меню на екрані монітора.

Таб. 2.1. Перелік функцій, вхідної та вихідної інформації.

№ п/п	Найменування функції	Вхідна інформація	Вихідна інформація
1	Вхід у додаток	Дані для входу (логін та пароль)	Доступ до додатку
2	Створення розкладу робітників	Дані про робітника, дату та час роботи	Виведення створених графіків та їх наявність у файлі БД
3	Видалення створеного розкладу	Створені графіки	Видалений вибраний графік з файлу БД
4	Упорядкування розкладів за ім’ям та датою	Створені графіки	Упорядковані графіки за ім’ям або датою
5	Розразунок заробітної платні на основі створених графіків	Створені графіки та імя робітника	Розрахована заробітня платня для вибраного робітника
6	Розразунок продуктів та інгредієнтів	Дані про залишки продукта та інгредієнта	Повідомлення з інформацією про вистачання продукта та інгредієнта з розрахунком на робочий тиждень

2.4.3. Вимоги до видів забезпечення.

2.4.3.1. У вимогах до математичного забезпечення (МЗ) система не вимагає спеціального математичного забезпечення для реалізації покладених на неї функцій. Достатньо можливостей обраної СУБД.

2.4.3.2. Вимоги до інформаційного забезпечення (ІЗ).

2.4.3.2.1. Інформаційне забезпечення системи повинно містити дані, достатні для виконання всіх покладених на систему функцій. ІЗ повинно гарантувати раціональну організацію зберігання інформації та доступу до неї.

2.4.3.2.2. Слід передбачити захист даних від руйнування при аваріях і порушеннях у енергоживленні системи — використання резервних копій БД.

2.4.3.3. Вимоги до лінгвістичного забезпечення (ЛЗ) включають:

1. Мультиязичність: Додаток підтримує різні мови для взаємодії з користувачами.

2. Точність та зрозумілість: Лінгвістичні елементи системи, такі як повідомлення, інструкції та помилки сформульовані чітко та зрозуміло. Точність і відповідність мовним правилам допомагають уникнути непорозумінь та помилок у взаємодії з користувачами.

3. Гнучкість та розширюваність: Лінгвістичне забезпечення є гнучким і легко розширюваним для введення нових мов або зміни існуючих. Це включає можливість додавання нових перекладів, оновлення локалізації та налаштування мовних параметрів.

4. Локалізація дат, часу та форматів: Додаток підтримує локалізацію дат, часу та форматів відображення числових значень. Це дозволяє адаптувати відображення дати, часу та чисел до мовних та культурних вимог різних користувачів.

2.4.3.3.1. Для розробки програмних засобів, які реалізують виконання функцій і забезпечують сервіс користувачів повинні використовуватися мови високого рівня, які забезпечують створення структурних програм, а також мова обраної СУБД для здійснення доступу та маніпулювання даними.

2.4.3.3.2. Організація діалогу користувача до системи має будуватися на наборах меню, орієнтованих на виконання користувачем функцій. Запити користувача до системи повинні задаватись переважно природною мовою.

2.4.3.4. Вимоги до програмного забезпечення (ПЗ).

2.4.3.4.1. Програмне забезпечення (ПЗ) повинне включати загальне (або системне) ПЗ і спеціалізоване ПЗ.

Загальне ПЗ включає такі програми та інструменти, як:

- Операційна система: Додаток та усі його функції працюють на операційних системах iOS/macOS
- Бази даних: У додатку використовуються Core Data (Swift) у якості СУБД. Вони використовуються для організації та збереження даних, забезпечуючи доступ до них та можливість виконання операцій з даними.
- Розробницькі середовища: Для написання додатка було використано xCode. xCode надає засоби для розробки програм, включаючи редактори коду, відладчики, компілятори та інші інструменти.
- Інструменти для контролю версій: Створення та тестування версій було здійснено за допомогою xCode та iPhone 11 Pro. Вони дозволяють керувати версіями програмного коду, виконувати злиття різних версій, відстежувати зміни тощо.

2.4.3.4.2. Загальні вимоги до системного ПЗ можна сформулювати так:

- мінімальні вимоги до ресурсів технічних засобів (ТЗ);
- максимальна швидкодія;
- повне задоволення потреб функціональних завдань системи.

2.4.3.4.3. Вимоги до ОС:

- мінімальне використання ресурсів для власних потреб, передусім оперативної і дискової пам’яті;
- максимальна швидкодія при управлінні зовнішніми пристроями;
- ОС сервера — iOs, ОС клієнта — iOS .....

2.4.3.4.4. Вимоги до СУБД:

- максимальне задоволення потреб функціональних задач;
- надійність;
- ефективне управління потрібного обсягу і структури;
- швидкість виконання запитів користувачів;
- мінімальні вимоги до ТЗ.

2.4.3.4.5. Програмні засоби введення та виведення даних і ведення діалогу повинні забезпечувати:

- Програма повинна мати зрозумілий та легкий у використанні інтерфейс для зручного введення та виведення даних.
- Програма повинна мати механізми для перевірки правильності та валідації введених даних, щоб уникнути помилок та забезпечити цілісність даних.
- Програма повинна мати можливість ведення діалогу з користувачем, щоб отримати необхідну інформацію та надати відповіді на запитання.
- Програма повинна бути здатна обробляти різні формати даних, такі як текстові файли, бази даних, електронні таблиці тощо.
- Програма повинна мати механізми захисту введених даних від несанкціонованого доступу та зберігання їх у безпечному форматі.

2.4.3.4.6. При розробленні спеціального ПЗ слід виконати наступні вимоги:

- використані програми мають бути сумісні між собою та із загальносистемним ПЗ;
- ПЗ має розроблятися засобами об’єктно-орієнтованого програмування;
- забезпечити відповідність інтерфейсу користувача стандартам Windows;
- необхідна модульна структура програм;
- повинна бути передбачена можливість розширення складу задач у відповідності з новими функціональними потребами;
- ПЗ не повинно залежати від типу зовнішніх пристроїв (принтерів, дисків, сканерів тощо);
- діалог із користувачем повинен проводитись за допомогою клавіатури або миші з поясненням виконання дій і можливістю отримання підказки.

2.4.3.5. Вимоги до технічного забезпечення.

2.4.3.5.1. Технічні засоби системи (табл. 2) повинні забезпечувати виконання функцій, перерахованих в таблиці 1.

2.4.3.5.2. Засоби обчислювальної техніки повинні забезпечувати обмін інформації в об’ємах, приведених в п. 4.3.2.

Таб. 2.2. Вимоги до технічного забезпечення системи.

№ п/п	**Основні характеристики комп’ютера**
Технічне забезпечення для сервера	
1	2x Intel Xeon E5-2670 v3 Cores: 2x 12x 2.30 GHz (Dual 12 Core) RAM: 192 GB DDR4 ECC reg. HDDs: 16x 4 TB SATA 7.2k RPM HW Raid IPMI/KVM
Технічне забезпечення для клієнта	
1	Клієнтам може знадобитися мобільний пристрій (смартфон, планшет), залежно. Цей пристрій повинен мати відповідну обчислювальну потужність та вільну для використання пам’ять.
2	Операційна система: Клієнтам потрібно мати операційну систему, яка підтримується додатком (iOS/macOS).
3	Інтернет-з'єднання: Для доступу до додатку доступ до інтернет-з’єднання не є необхідним.

2.4.3.6. Вимоги до метрологічного забезпечення.

Система не має вимірювальних каналів, вимірювального обладнання і приладів, тому вимоги до даного виду забезпечення не висуваються.

2.4.3.7. Вимоги до організаційного забезпечення.

2.4.3.7.1. Організаційне забезпечення системи розробляється в відповідності з вимогами державного стандарту по АСУП.

2.4.3.7.2. При впровадженні системи не передбачається збільшення штатної чисельності підприємства. Територіальне розміщення робочих місць, на яких буде встановлена система, визначається підприємством.

2.4.3.7.3. До функціонування системи висуваються наступні вимоги:

- Надійність: Система повинна працювати безперебійно і надійно, уникати відмов і збоїв, а також забезпечувати збереження та відновлення даних при несприятливих умовах.
- Швидкодія: Система повинна забезпечувати швидку обробку даних та ефективну реакцію на користувацькі запити, забезпечуючи задоволення вимог до продуктивності.
- Масштабованість: Система повинна бути здатною розширюватися та масштабуватися, щоб впоратися з зростаючим обсягом даних, навантаженням і кількістю користувачів.
- Безпека даних: Система повинна забезпечувати захист даних від несанкціонованого доступу, зберігання даних в захищеному стані, шифрування комунікацій та інші механізми безпеки.
- Гнучкість: Система повинна бути гнучкою та здатною адаптуватися до змінних потреб бізнесу, забезпечуючи можливість легко налаштовувати і змінювати функціональність.
- Легкість використання: Система повинна мати зрозумілий та зручний інтерфейс користувача, який дозволяє легко виконувати операції, навіть без попередньої підготовки або досвіду використання системи.
- Документація та підтримка: Система повинна мати належну документацію, яка охоплює опис функціональності системи, інструкції з використання, конфігурацію та іншу необхідну інформацію. Крім того, система повинна мати надійну технічну підтримку, яка забезпечує реагування на запити користувачів, вирішення проблем та надання консультацій.

2.5. Склад і зміст робіт по створенню системи.

2.5.1. Стадії створення системи і терміни виконання робіт наведені в таблиці 3.

Таб. 2.3. Найменування робіт при створенні системи.

№ п/п	Найменування робіт	Строки виконання робіт
1	Передпроектне дослідження об’єкта автоматизації	10.03.2023
2	Технічне завдання	15.04.2023
3	Технічний проект	20.05.2023
4	Оформлення документації	10.06.2023

2.6. Порядок контролю і приймання системи.

2.6.1. Перед прийманням системи здійснюється комплексний контроль її відповідності вимогам і технічним характеристикам, встановленим у технічному завданні та документації проекту.

2.6.2. Контроль і приймання системи проводяться за наступним порядком:

2.6.2.1. Перевірка виконання технічних вимог: Перевіряється, чи система відповідає вимогам, зазначеним у технічному завданні. Це включає перевірку функціональності, продуктивності, надійності та інших технічних параметрів.

2.6.2.2. Тестування системи: Здійснюється комплексне тестування системи з метою перевірки її працездатності, відповідності вимогам та виявлення можливих помилок або недоліків. Тестування може включати функціональне тестування, тестування продуктивності, безпеки та інші види тестування, залежно від потреб проекту.

2.6.2.3. Перевірка документації: Перевіряється наявність та відповідність документації, пов'язаної з системою, такої як технічна документація, інструкції з експлуатації, керівництва для користувача та інші документи. Вони повинні бути повними, зрозумілими і відповідати вимогам стандартів та рекомендацій.

2.6.2.4. Приймання системи: Після успішного проходження контролю і тестування система приймається замовником або відповідними структурними підрозділами компанії. У разі виявлення недоліків або відхилень вимагається їх усунення перед прийманням.

2.6.2.5. Документування результатів: Результати контролю і приймання системи фіксуються відповідною документацією, яка включає протоколи тестування, акти приймання, відомості про виявлені недоліки та іншу необхідну інформацію.

2.6.3. Контроль і приймання системи проводяться відповідно до узгодженого графіка і залежно від потреб проекту. Після приймання система готова до подальшої експлуатації та використання замовником.

2.7. Вимоги до додатку і змісту робіт із підготовки до введення системи в дію.

2.7.1. Вимоги до додатку:

- Додаток повинен бути розроблений з використанням сучасних технологій та забезпечувати надійне та безперебійне функціонування.
- Додаток повинен мати інтуїтивно зрозумілий інтерфейс, що дозволяє користувачам легко взаємодіяти з ним та здійснювати необхідні операції.
- Додаток повинен забезпечувати безпеку і захист інформації користувачів, включаючи механізми автентифікації, шифрування та контролю доступу.

2.7.2. Зміст робіт із підготовки до введення системи в дію:

- Налаштування додатку: Включає налаштування параметрів додатку, перевірка створення усіх необхідних Data Base файлів.
- Тестування та відладка: Проводиться комплексне тестування додатку та його функцій для перевірки його працездатності, виявлення та виправлення можливих багів або недоліків.
- Навчання персоналу: Проводиться навчання менеджеру та інших відповідальних осіб, які будуть користуватися додатком.
- Підготовка документації: Розробляється необхідна документація, включаючи інструкції з використання, керівництва для користувачів та інші відповідні матеріали.

- Планування впровадження: Розробляється план впровадження системи в дію, включаючи часові рамки, розподіл ролей та завдань, комунікацію зі зацікавленими сторонами та інші необхідні етапи.

2.8. Вимоги до документації.

2.8.1. Загальні вимоги до документації:

- Документація повинна бути чітко структурованою, зрозумілою та доступною для користувачів.
- Документація повинна містити повну та точну інформацію про систему, її функціональні можливості, установку, налаштування та використання.
- Документація повинна бути збережена у відповідному форматі, який дозволяє зручне зчитування та пошук необхідної інформації.

2.9. Джерела розробки.

2.9.1. При розробленні технічного завдання на систему використано наступні документи:

- ДСТУ Б В.2.5-64: Норми пожежної безпеки будівель. Цей документ містить вимоги до системи пожежної безпеки, включаючи системи пожежного сповіщення, евакуації, гасіння пожеж і т.д.
- ДСТУ Б В.1.1-7: Норми з енергозбереження будинків та споруд. Цей документ встановлює вимоги до енергоефективності будівель та рекомендації щодо використання енергозберігаючих систем.
- ДСТУ Б Д.1.1-1: Будівельні конструкції. Основні положення. Цей документ містить загальні вимоги до будівельних конструкцій, їх надійності, міцності та інших параметрів.
- ДСТУ 3008-2015. Документація. Звіти у сфері науки і техніки. Структура та правила оформлювання;

РОЗДІЛ 3. ОПИС КОМПЛЕКСУ ЗАДАЧ АВТОМАТИЗАЦІЇ

3.1. Інформаційне забезпечення системи

Для створення розробленого додатку “helper.” для управління кав’ярнею було використано такі програмні засоби:

1. xCode.
2. iPhone 11 Pro.
3. Figma
4. Canva.

Спочатку нам потрібно створити проект у xCode та вибрати один з представлених шаблонів (iOS – App).

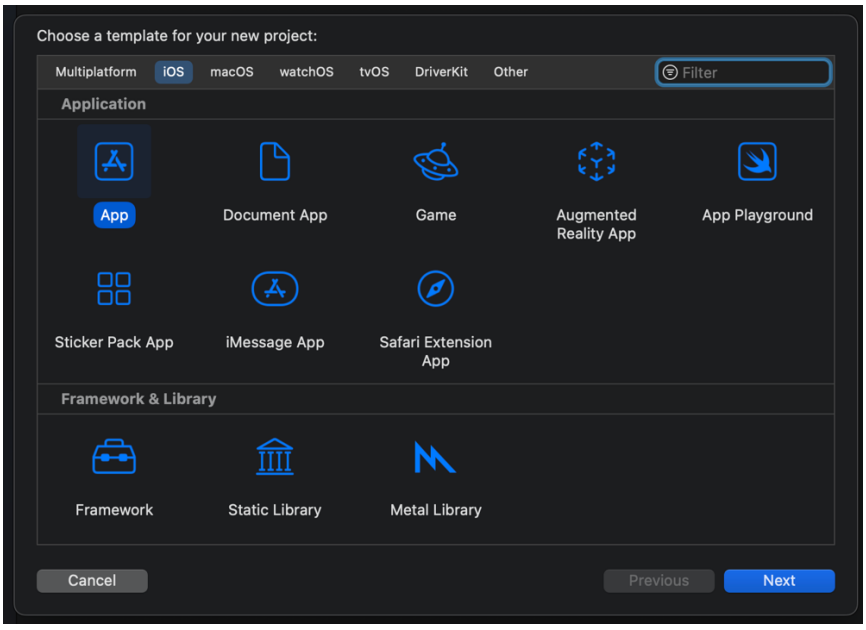


Рисунок 1.1. Створення проекту та вибір шаблонів у xCode.

Буде створено папку з назвою проекту та основними файлами програми.

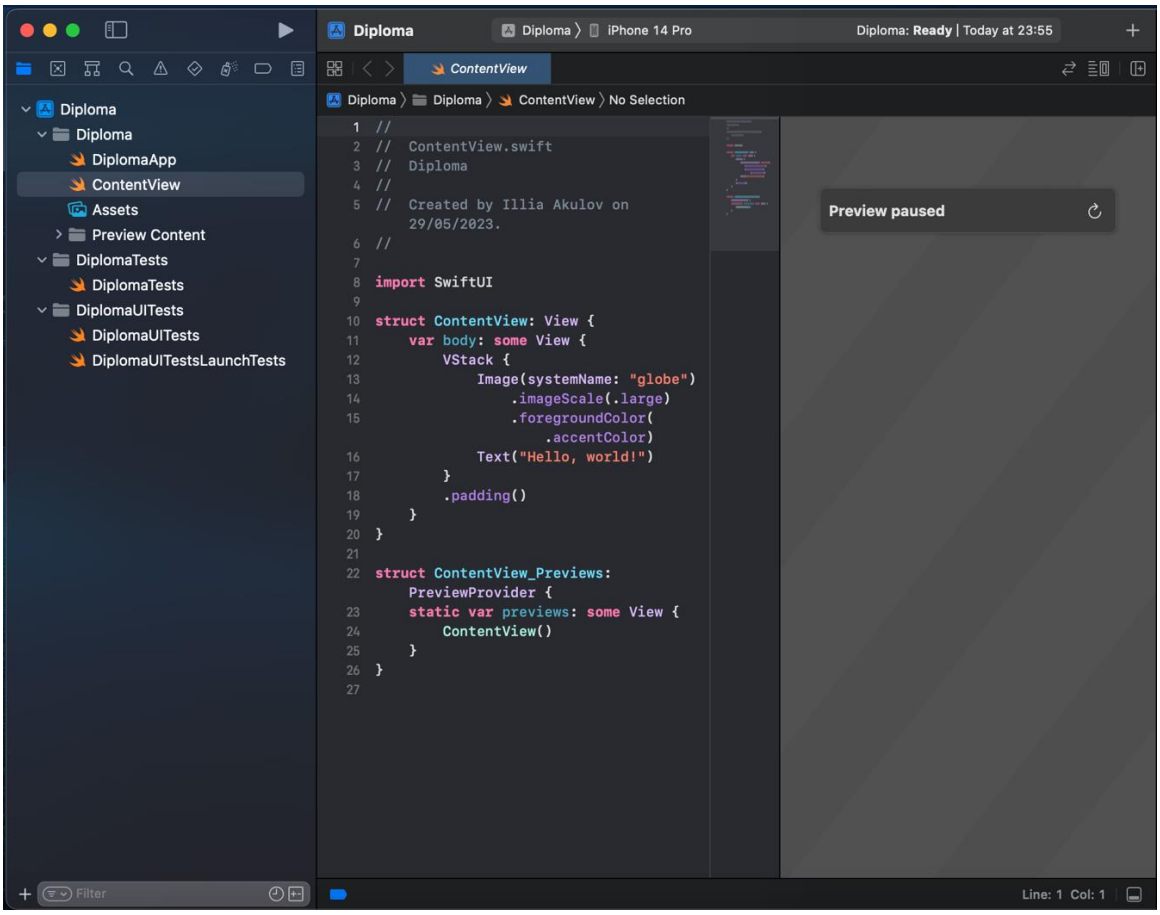


Рисунок 1.2. Приклад створеного проекту за шаблоном.

Після треба створити основні файли програми (.swift/.xcdatamodel) такі як:

- ‘HelperApp’ – файл, який відповідає за запуск програми та запуск вікон для входу у додаток та головного меню додатку.
- ‘HelperLoginView’ – файл, який відповідає за інтерфейс та функції входу у додаток. Код файлу продемонстрований у Додатку А.
- ‘HelperMainMenu’ - файл, який відповідає за інтерфейс та функції головного меню. Код файлу продемонстрований у Додатку А.
- ‘CoreDataManager’ – файл, який відповідає за роботу з Data Base файлом. Код файлу продемонстрований у Додатку А.
- ‘HelperDataBase’ – основний Data Base файл додатку. Приклад налаштування баз даних продемонстрований у Додатку А.

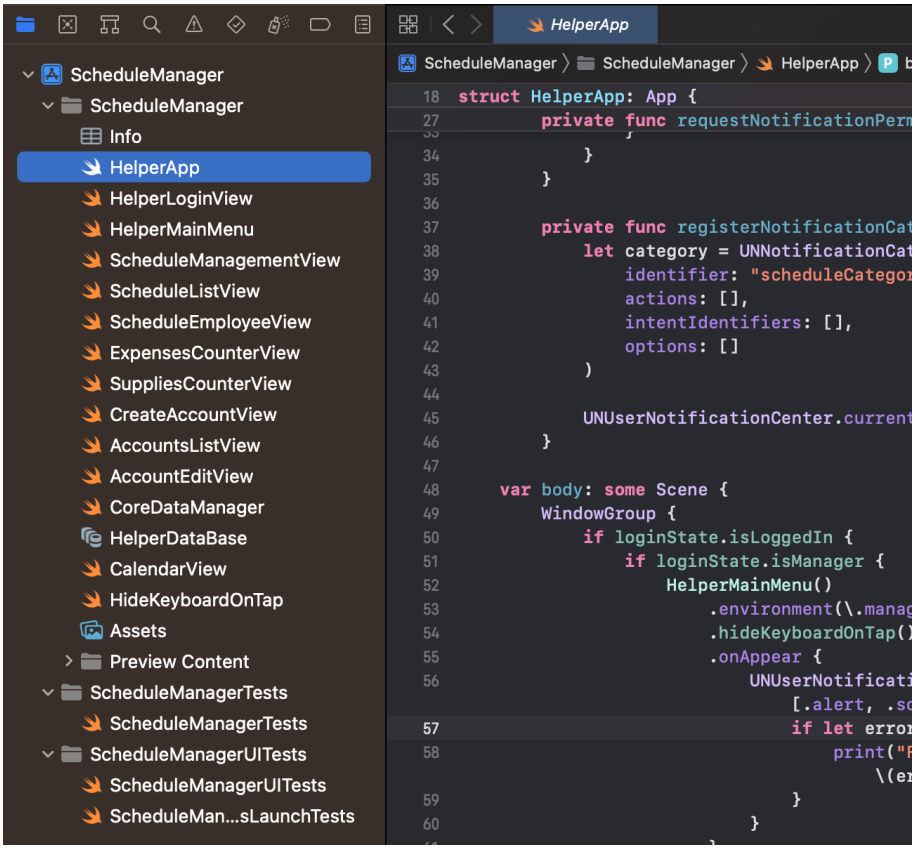


Рисунок 2.1. Ієрархія файлів проекту.

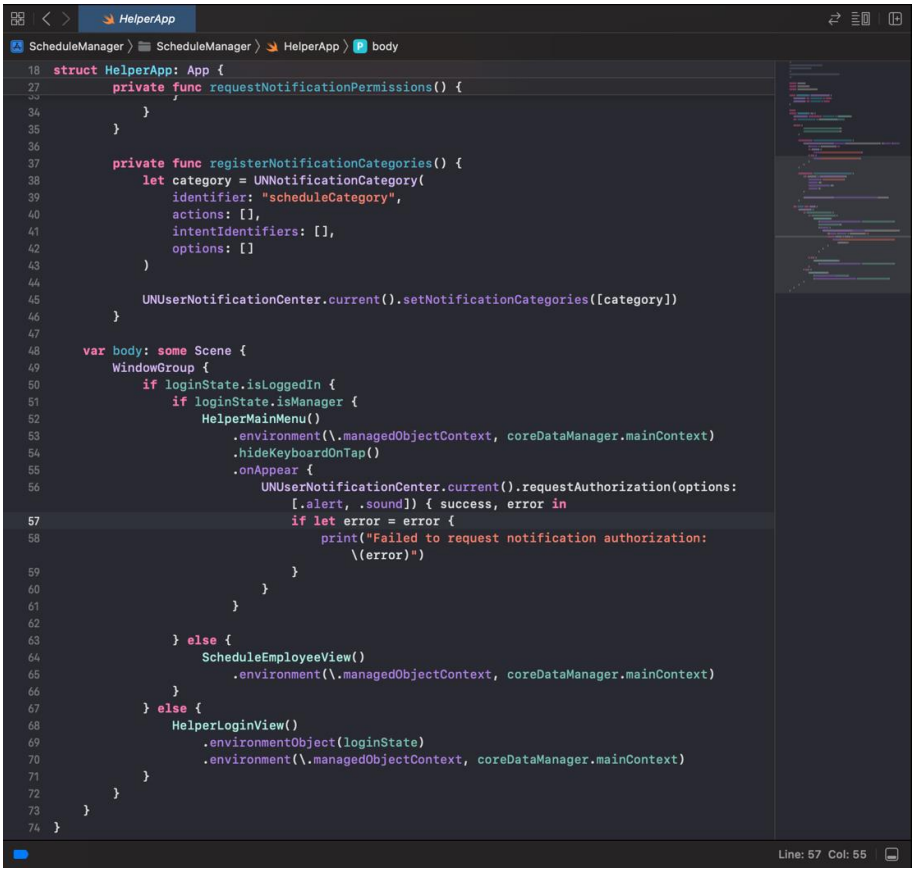


Рисунок 2.2. Код файлу ‘HelperApp.swift’.

Один з основних файлів - ‘CoreDataManager.swift’ використовує основний створений клас ‘CoreDataManager’ робота якого можлива за умови імпорту бібліотеки ‘CoreData’.

Головні функції файлу ‘CoreDataManager.swift’:

1. **‘NSPersistentContainer’** є класом мови програмування Swift, який відповідає за керування основною функціональністю збереження даних в Core Data. Він надає зручний спосіб створення та налаштування Core Data stack, включаючи NSManagedObjectModel, NSPersistentStoreCoordinator і NSManagedObjectContext. NSPersistentContainer також забезпечує автоматичну обробку міграції даних та резервне копіювання.
2. **‘NSManagedObjectContext’** є класом мови програмування Swift, який відповідає за керування контекстом роботи з об'єктами Core Data. Цей контекст використовується для зберігання, зчитування та зміни об'єктів у базі даних. NSManagedObjectContext забезпечує взаємодію з базою даних Core Data, включаючи створення та видалення об'єктів, виконання запитів і збереження змін.
3. **‘NSFetchRequest’** є класом мови програмування Swift, який відповідає за визначення запитів до бази даних Core Data для отримання певних об'єктів. Він дозволяє встановлювати різні умови, сортування та обмеження для пошуку конкретних об'єктів або наборів даних. NSFetchRequest може бути використаний разом з NSManagedObjectContext для виконання запитів та отримання результатів з бази даних Core Data.

Всі ці функції є частинами фреймворка Core Data у мові програмування Swift. Вони надають потужні можливості для роботи з базою даних, зокрема збереження, зчитування та вибірку даних, а також керування контекстами роботи з об'єктами.

3.2. Алгоритмізація та реалізація комплексу задач автоматизації.

1. Система працює шляхом навігації за допомогою відповідних кнопок меню та переходу до різних вкладок, де відображаються різноманітні форми.

Процес авторизації та аутентифікації у додатку включає запит користувача на введення облікових даних, їх перевірку з внутрішньою базою даних, а також використання додаткових методів перевірки, які можуть бути налаштовані залежно від вимог системи та рівня безпеки. За замовчуванням стоїть логін “Manager” та пароль “manager1’. Також є можливість увійти до додатку працівникам кав’ярні (за створеними менеджером даними)

```
59      private func login() {
60          if username == "Manager" && password == "manager1" {
61              // Manager login successful
62              loginState.isLoggedIn = true
63          } else {
64              showAlert = true
65          }
66      }
67  }
68
69  class LoginState: ObservableObject {
70      @Published var isLoggedIn = false
71  }
72
73
```

Рисунок 3.1.Частина коду, яка відповідає за вхід у додаток.

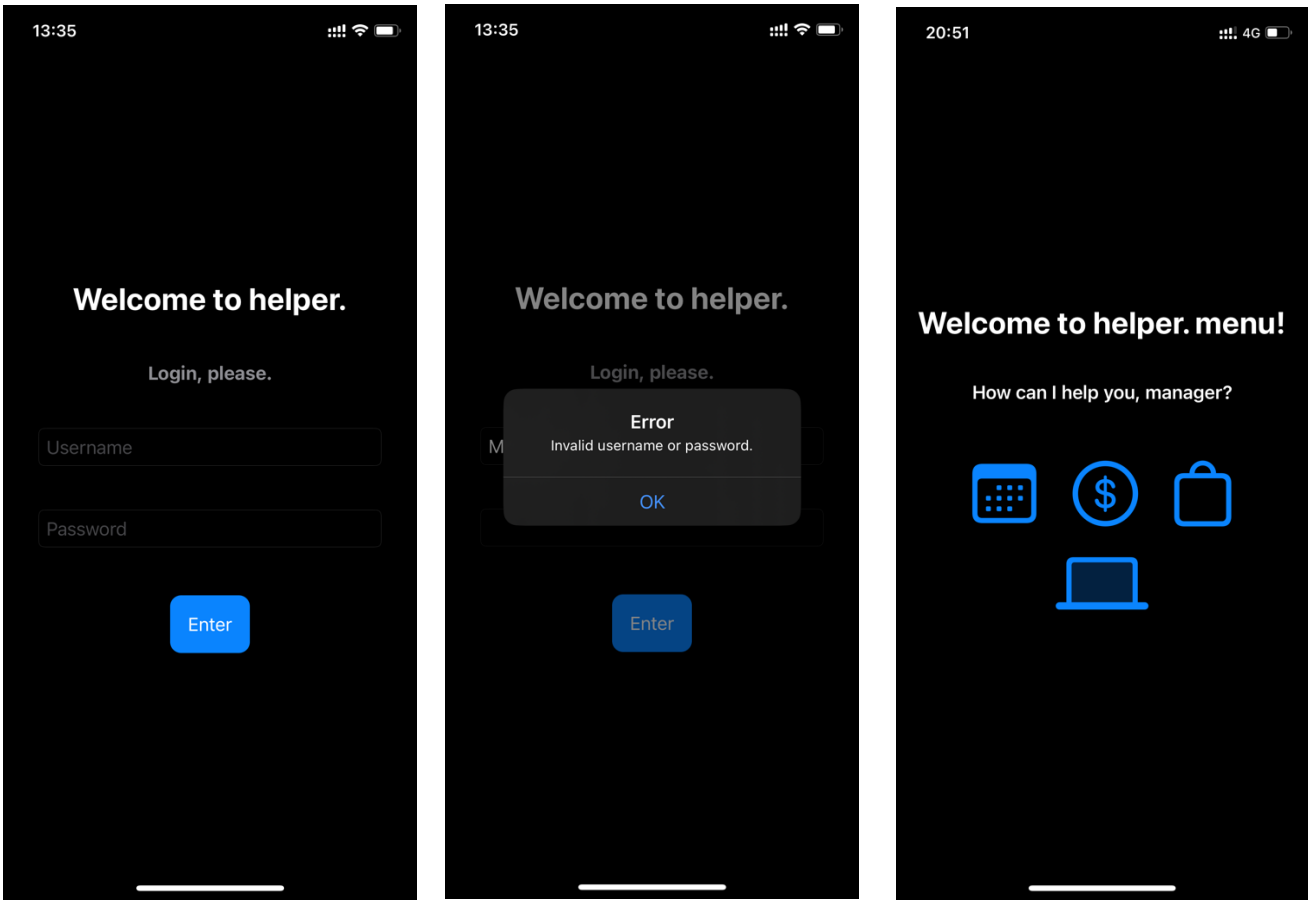


Рисунок 3.2-4.Вікно для входу, помилка при введені неправильних даних, основне меню додатку.

Якщо менеджер успішно авторизувався, його допускає до основного меню додатку та є 4 кнопки на вибір.

Рисунок 3.5. Кнопки “Календар”, “Монета”, “Сумка” та “Ноутбук”.

Функція ‘NavigationLink’ створює 3 навігаційні кнопки, які перенаправляють користувача до наступних вікон та функцій додатку на вибір:

- ‘ScheduleManagementView’ – вікно для роботи з графіками.
- ‘ExpensesCounterView’ – вікно для роботи з розрахунками заробітної платні.
- ‘SuppliesCounterView’ – вікно для роботи з остатками продуктів та інгредієнтів.
- ‘CreateAccountView’ – вікно для роботи з аккаунтами працівників.

- **Файл ‘ScheduleManagementView.swift’** – створює вікно для роботи з графіками та відповідає за функції створення графіків. Працює з допоміжним файлом **‘ScheduleListView.swift’**, який відповідає за відображення усіх створених графіків, їх сортування (за ім’ям робітника чи датою) та видалення. Код файлів ‘ScheduleManagementView.swift’ та ‘ScheduleListView’ продемонстрований у Додатку Б.

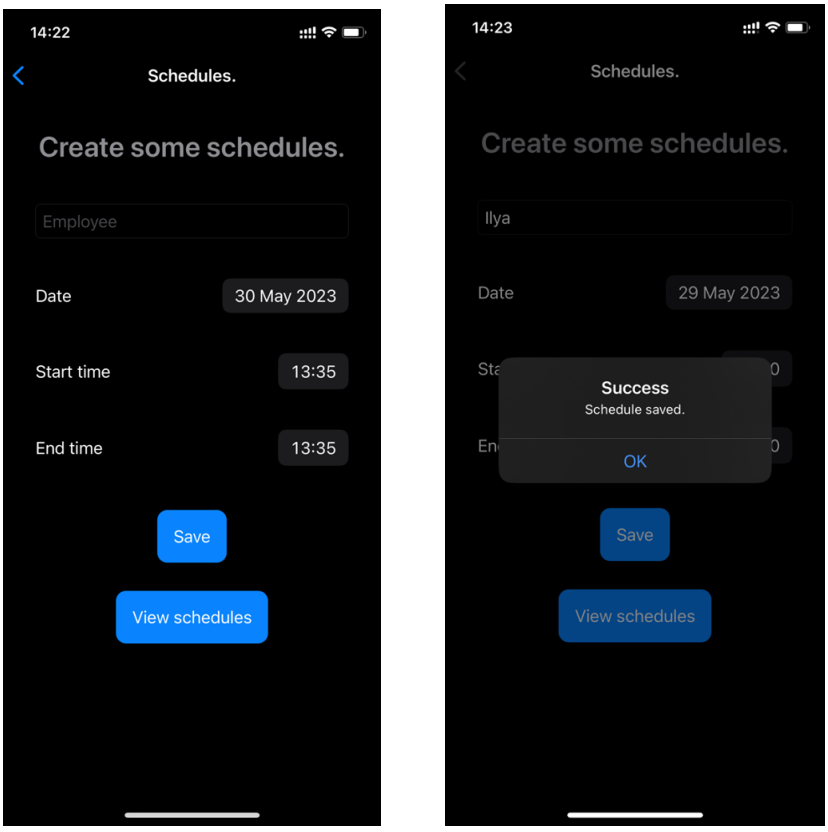


Рисунок 3.12-13. Основне вікно для роботи з графіками та приклад успішного створення графіку.

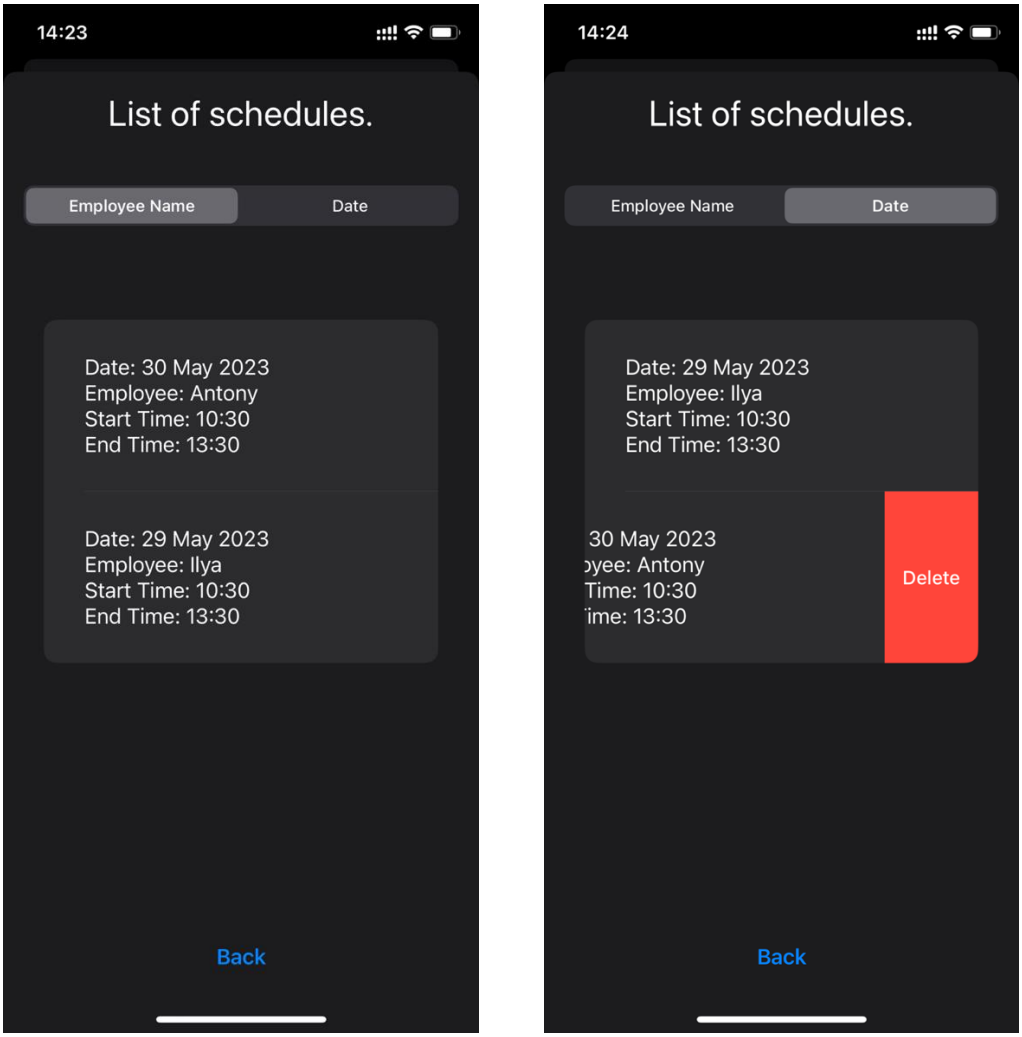


Рисунок 3.14-15. Вікно відображення створених графіків, приклад сортування графіків за ім'ям або датою та приклад видалення графіку.

Також у додатку передбачено вікно відображення розкладу до створення графіків. Якщо користувач намагається створити вже існуючий графік, то програма виведе вікно з попередженням, що даний графік вже існує.

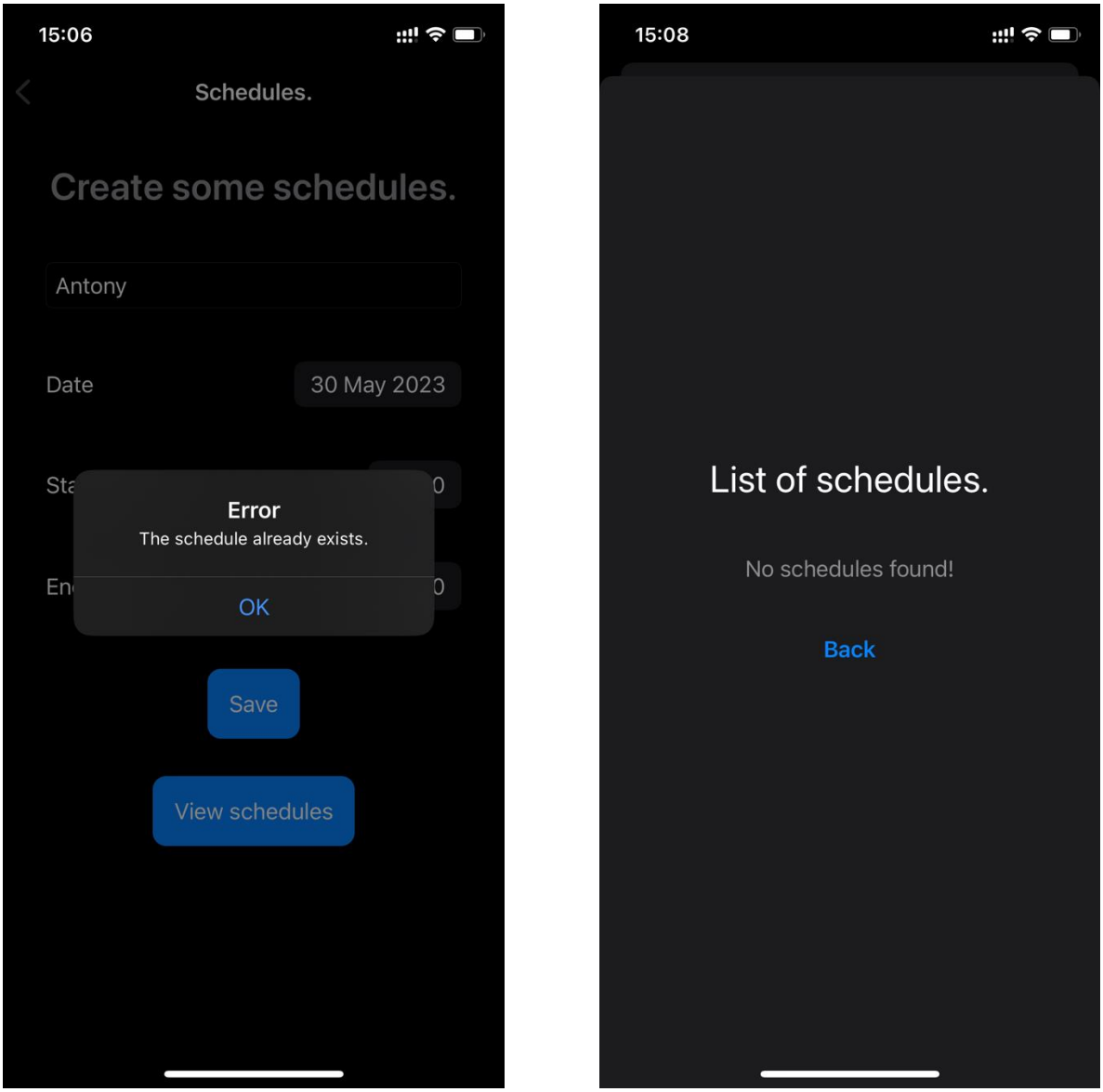


Рисунок 3.16-17. Вікно з повідомленням про помилку у створенні графіку, приклад вікна з розкладами за відсутності графіків.

- **Файл ‘CounterExpensesView.swift’** - створює вікно для роботи з розрахунками заробітної платні. Для розрахунку потрібно обрати ім’я працівника та ввести його погодинну оплату праці. Працює з допоміжними класами **‘ExpensesCounter’ та ‘ShiftView’**. Перший відповідає за розрахунки заробітної платні за формулою (робочі години з розкладу обраного працівника*введену погодинну оплату), другий відповідає за відображення розкладу, за яким виконуються розрахунки. Код файлу ‘CounterExpensesView.swift’ продемонстрований у Додатку В.

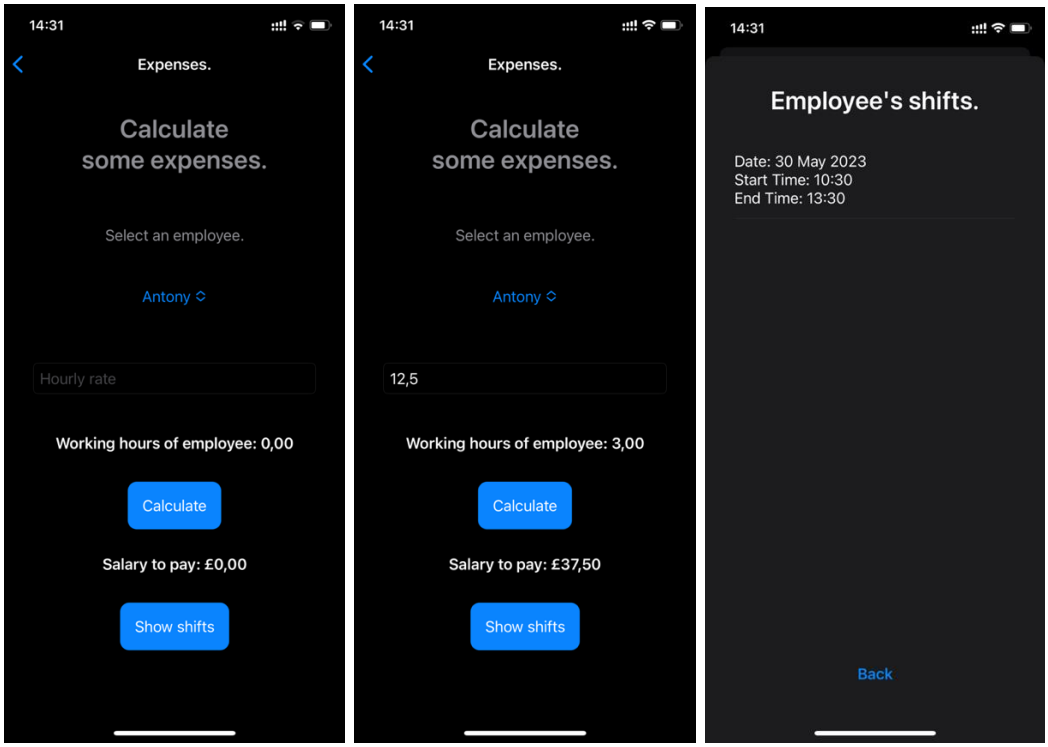


Рисунок 3.24-26. Основне вікно для розрахунку заробітної платні, приклад успішного розрахунку для обраного робітника та вікно відображення усіх графіків за якими було виконано розрахунки.

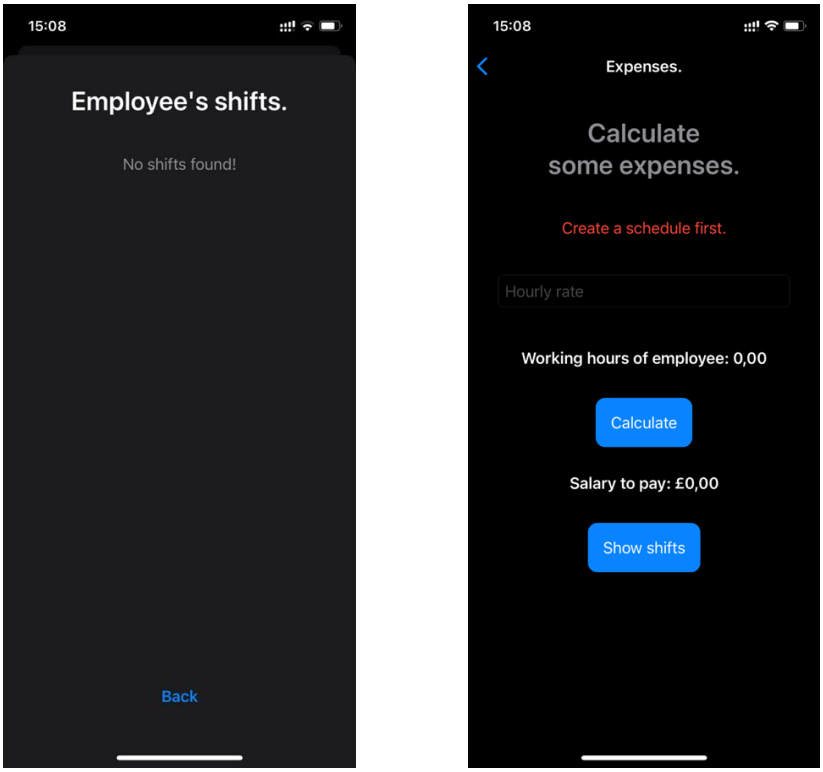


Рисунок 3.27-28. Вікно відображення відсутніх графіків працівника та вікно з нагадуванням про створення розкладу перед розрахунком.

Також у додатку передбачено вікно відображення відсутніх графіків працівника та нагадування у текстовому форматі, у головному меню розрахунку, про створення розкладу перед проведенням розрахунку заробітної платні.

- **Файл ‘SuppliesCounterView.swift’** - створює вікно для роботи з розрахунками залишків основних продуктів та інгредієнтів для роботи кав’ярні. Для розрахунку потрібно ввести дані про наявність продуктів та інгредієнтів, після програма проведе розрахунок та перевірку чи вистачає наявних продуктів чи інгредієнтів на робочий тиждень. Проводить розрахунки за статистикою середнього споживання продуктів та інгредієнтів за день роботи кав’ярні у звичайному режимі (5 кілограмів кави, 500 грамів чаю, 200 грамів матча, 10 літрів молока та 10 літрів вівсяного молока). Код файлу ‘SuppliesCounterView.swift’ продемонстрований у Додатку Г.

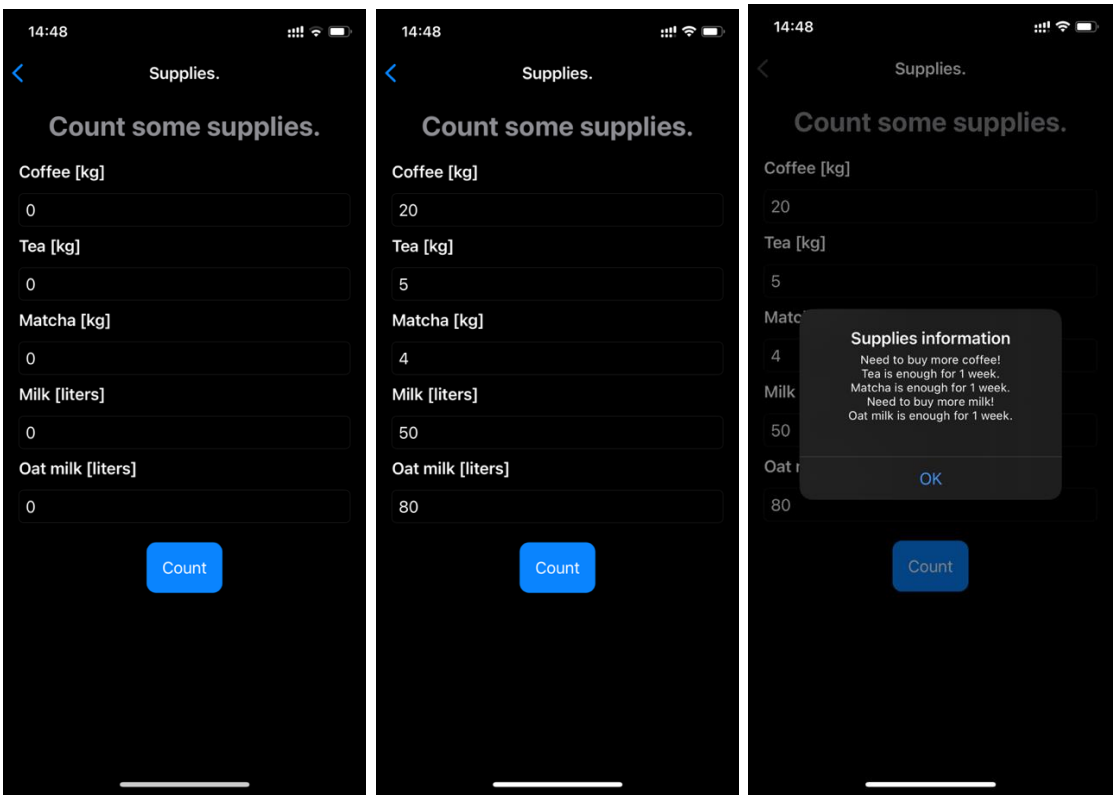
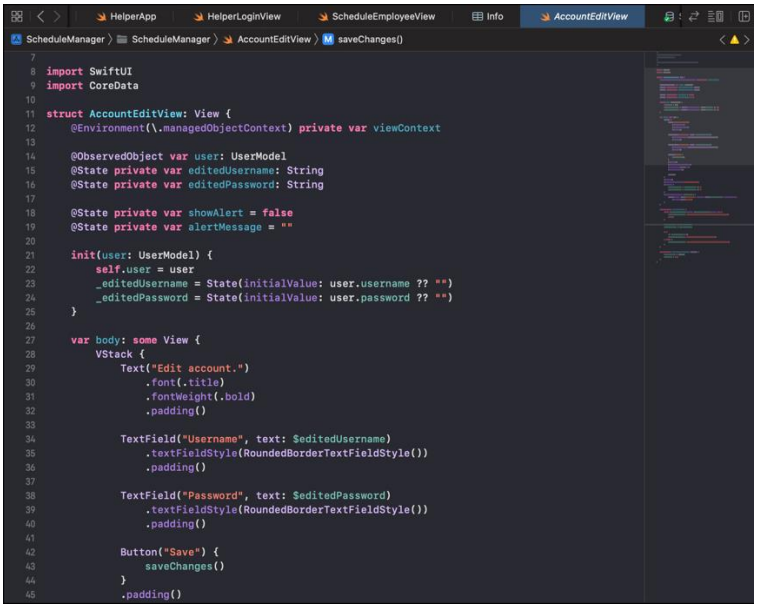
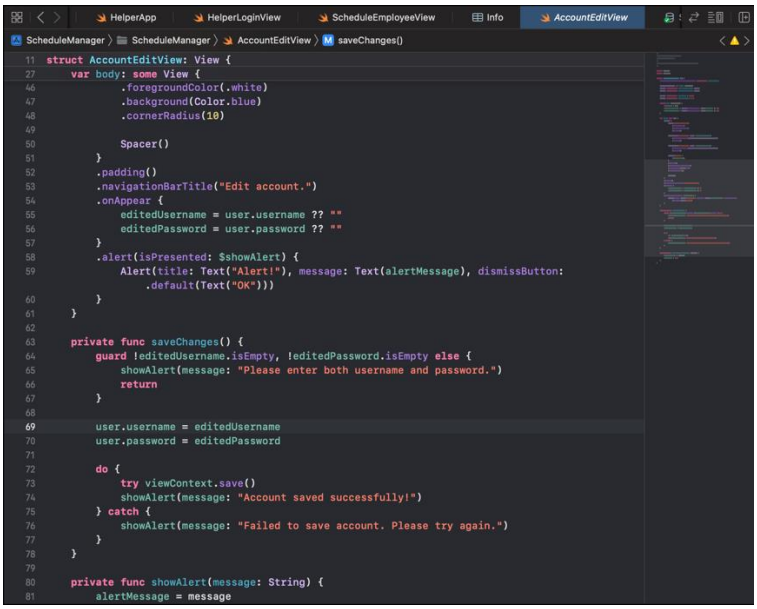


Рисунок 3.31-32. Основне вікно для розрахунку залишків, приклад даних для розрахунку залишків та вікно з повідомленням про вистачу залишків на робочий тиждень кав’ярні.

- **Файл ‘CreateAccountView.swift’** – створює вікно для роботи з аккаунтами працівників, а саме їх створення та редагування. Після створення аккаунту працівника (логін та пароль), інформація надається працівнику та він має доступ до додатку, а саме може передивлятися графік та ставити нагадування про робочу зміну. Працює з файлами ‘AccountListView.swift’ та ‘AccountEditView’, які відповідають за перегляд та редагування аккаунтів. Основний файл працює з БД ‘UserModel’ з файлу ‘HelperDataBase’.



```swift
import SwiftUI
import CoreData

struct AccountEditView: View {
    @Environment(\.managedObjectContext) private var viewContext

    @ObservedObject var user: UserModel
    @State private var editedUsername: String
    @State private var editedPassword: String

    @State private var showAlert = false
    @State private var alertMessage = ""

    init(user: UserModel) {
        self.user = user
        _editedUsername = State(initialValue: user.username ?? "")
        _editedPassword = State(initialValue: user.password ?? "")
    }

    var body: some View {
        VStack {
            Text("Edit account.")
                .font(.title)
                .fontWeight(.bold)
                .padding()

            TextField("Username", text: $editedUsername)
                .textFieldStyle(RoundedBorderTextFieldStyle())
                .padding()

            TextField("Password", text: $editedPassword)
                .textFieldStyle(RoundedBorderTextFieldStyle())
                .padding()

            Button("Save") {
                saveChanges()
            }
            .padding()
```



```swift
struct AccountEditView: View {
    var body: some View {
            .foregroundColor(.white)
            .background(Color.blue)
            .cornerRadius(10)

            Spacer()
        }
        .padding()
        .navigationBarTitle("Edit account.")
        .onAppear {
            editedUsername = user.username ?? ""
            editedPassword = user.password ?? ""
        }
        .alert(isPresented: $showAlert) {
            Alert(title: Text("Alert!"), message: Text(alertMessage), dismissButton:
                .default(Text("OK")))
        }
    }

    private func saveChanges() {
        guard !editedUsername.isEmpty, !editedPassword.isEmpty else {
            showAlert(message: "Please enter both username and password.")
            return
        }

        user.username = editedUsername
        user.password = editedPassword

        do {
            try viewContext.save()
            showAlert(message: "Account saved successfully!")
        } catch {
            showAlert(message: "Failed to save account. Please try again.")
        }
    }

    private func showAlert(message: String) {
        alertMessage = message
```

Рисунок 3.33. Код файлу ‘CreateAccountView.swift’

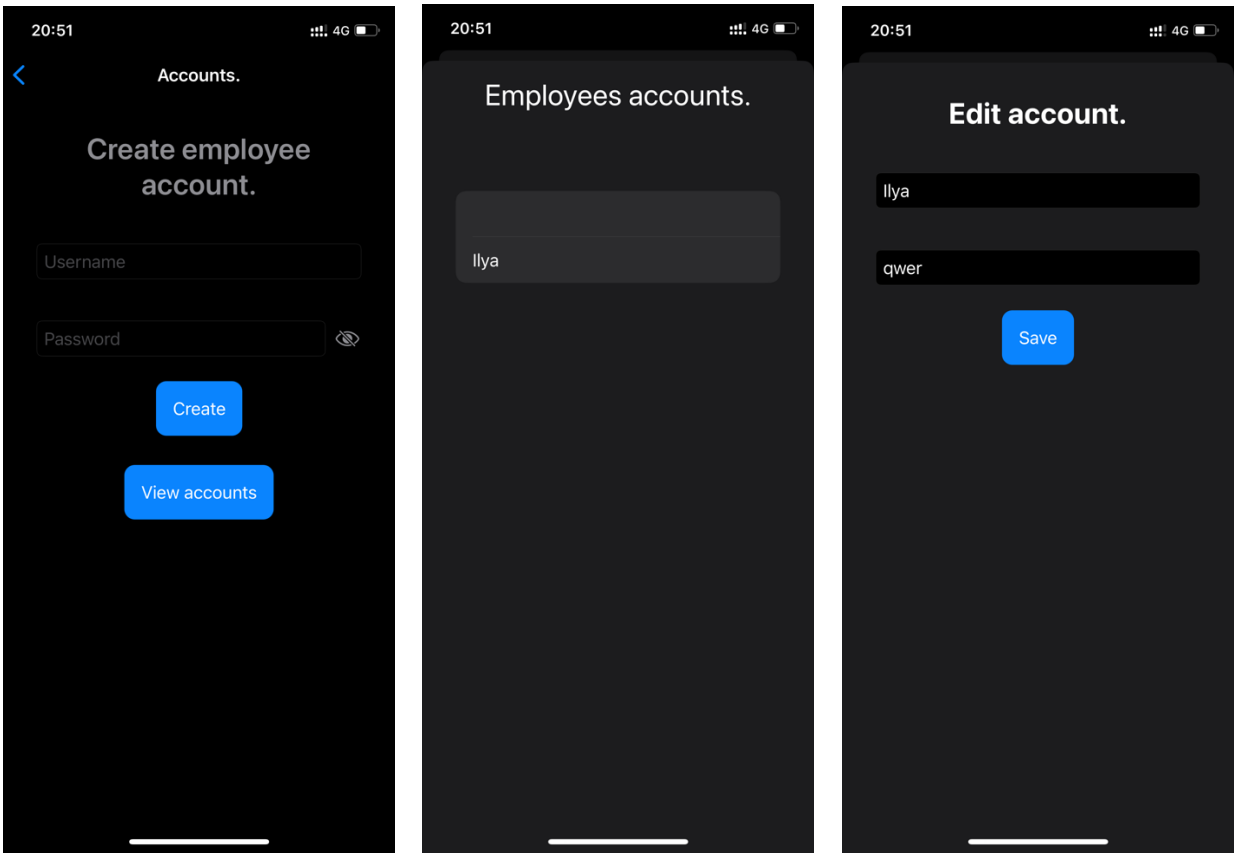


Рисунок 3.34-36. Вікно для створення аккаунта працівника, вікно для перегляду усіх аккаунтів, вікно для редагування аккаунта.

- **Файл ‘ScheduleEmployeeView.swift’ –** створює вікно для перегляду розкладів, сортування та встановлення нагадувань після входу до додатку з аккаунту працівника. Код файлу ‘ScheduleEmployeeView.swift’ продемонстровано у Додатку Д.

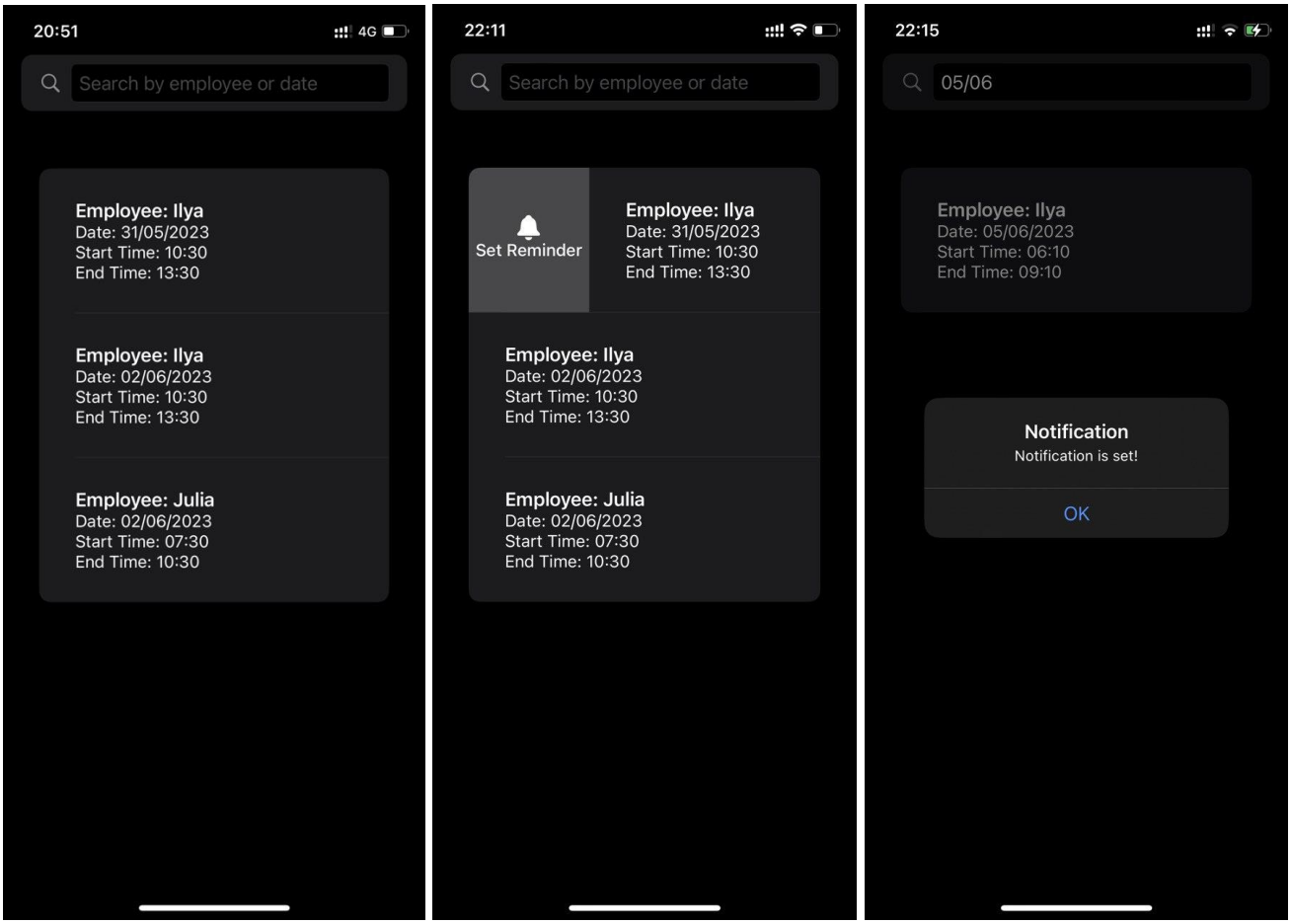


Рисунок 3.41-43. Вікно розкладу для аккаунту персонала, кнопка встановлення нагадування про обраний графік та повідомлення успішно встановленого нагадування.

- **Допоміжний файл ‘CalendarView.swift’** – відповідає за вигляд календаря, який відображується при виборі дати для створення розкладів у файлі ‘ScheduleManagementView.swift’.

```swift
//

import SwiftUI

struct CalendarView: View {
    @State private var selectedDate: Date = Date()

    var body: some View {
        VStack {
            Text("Selected Date:")
            Text(dateFormatter.string(from: selectedDate))
                .font(.title)
                .fontWeight(.bold)

            DatePicker("", selection: $selectedDate, displayedComponents: .date)
                .datePickerStyle(GraphicalDatePickerStyle())
                .labelsHidden()
        }
        .padding()
        .navigationBarTitle("Calendar")
    }

    private let dateFormatter: DateFormatter = {
        let formatter = DateFormatter()
        formatter.dateFormat = "dd.MM.yyyy"
        formatter.locale = Locale(identifier: "en_US")
        return formatter
    }()
}
```

Рисунок 3.44. Код файлу ‘CalendarView.swift’

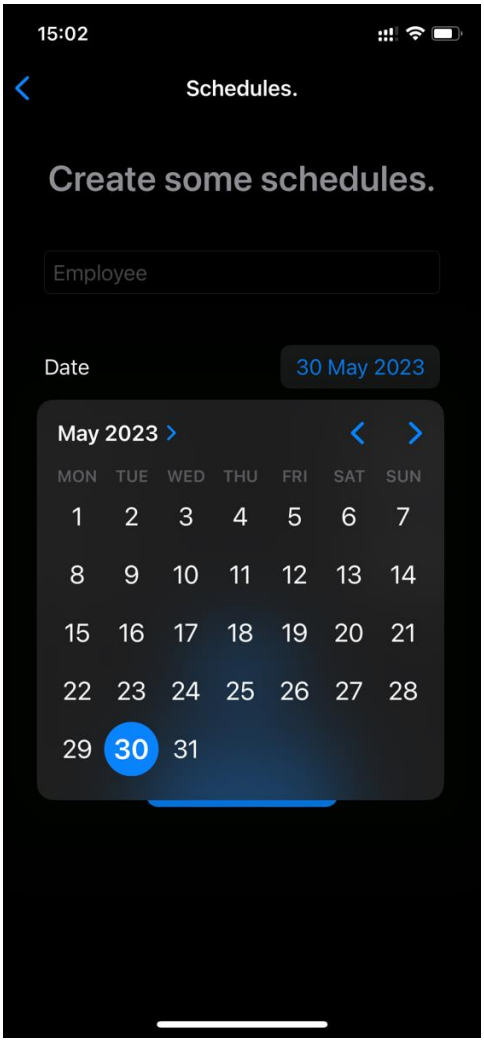
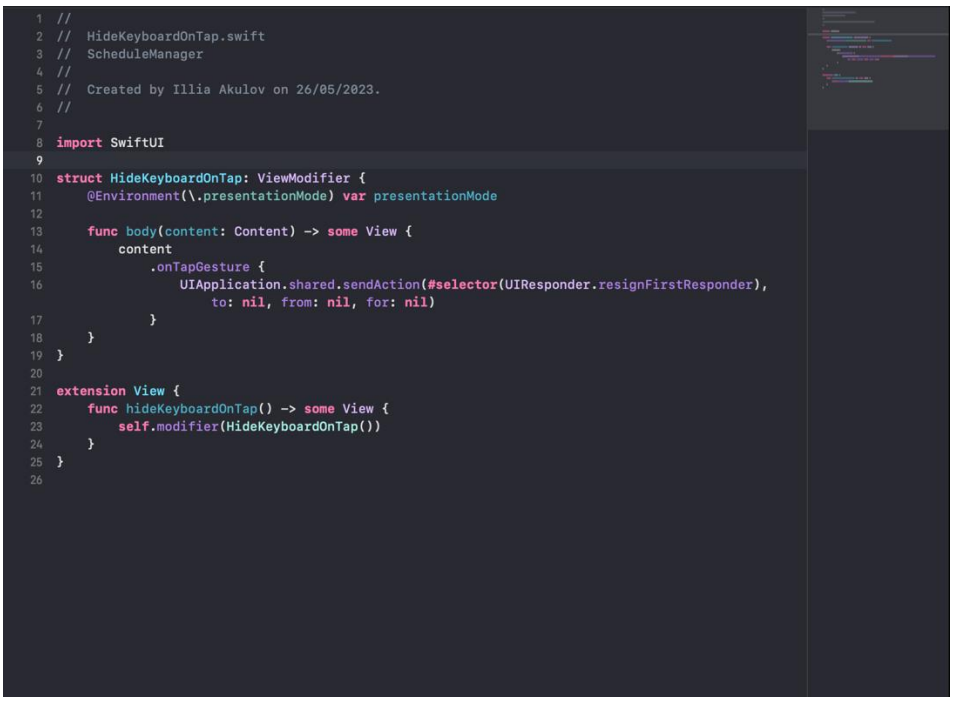


Рисунок 3.45. Приклад роботи файлу ‘CalendarView.swift’.

- **Допоміжний файл ‘HideKeyboardOnTap.swift’** – відповідає за згортання клавіатури при жесті натисканні на фон у додатку. Використовується у основному файлі ‘HelperApp.swift’.



```swift
//
//  HideKeyboardOnTap.swift
//  ScheduleManager
//
//  Created by Illia Akulov on 26/05/2023.
//

import SwiftUI

struct HideKeyboardOnTap: ViewModifier {
    @Environment(\.presentationMode) var presentationMode

    func body(content: Content) -> some View {
        content
            .onTapGesture {
                UIApplication.shared.sendAction(#selector(UIResponder.resignFirstResponder),
                    to: nil, from: nil, for: nil)
            }
    }
}

extension View {
    func hideKeyboardOnTap() -> some View {
        self.modifier(HideKeyboardOnTap())
    }
}
```

Рисунок 3.46. Код файлу ‘HideKeyboardOnTap.swift’

3.3. Технічне та системне забезпечення розробки

3.3.1. Обґрунтування вибору технічних засобів

№ п/п	**Основні характеристики комп’ютера**
Технічне забезпечення для сервера	
1	2x Intel Xeon E5-2670 v3 Cores: 2x 12x 2.30 GHz (Dual 12 Core) RAM: 192 GB DDR4 ECC reg. HDDs: 16x 4 TB SATA 7.2k RPM HW Raid IPMI/KVM
Технічне забезпечення для клієнта	

1	Клієнтам може знадобитися мобільний пристрій (смартфон, планшет), залежно. Цей пристрій повинен мати відповідну обчислювальну потужність та вільну для використання пам’ять.
2	Операційна система: Клієнтам потрібно мати операційну систему, яка підтримується додатком (iOS/macOS).
3	Інтернет-з'єднання: Для доступу до додатку доступ до інтернет-з’єднання не є необхідним.

3.3.2. Обґрунтування вибору ОС та протоколу обміну даними

При виборі операційної системи (ОС) та протоколу обміну даними для розробки додатку, слід враховувати деякі фактори:

1. **Підтримка мови програмування**: Перевірте, яку мову програмування ви плануєте використовувати для розробки додатку. Переконайтеся, що ОС підтримує дану мову програмування та її необхідні бібліотеки та фреймворки.

2. **Надійність та безпека**: ОС повинна бути надійною та забезпечувати високий рівень безпеки, оскільки розробка додатку і пов'язана з обробкою фінансових даних та конфіденційною інформацією.

3. **Сумісність з інструментами**: Впевніться, що ОС підтримує необхідні інструменти та сервіси, які ви плануєте використовувати у роботі додатка.

4. **Масштабованість**: Розгляньте можливість масштабування та можливість додавання нових функцій, оскільки додаток може знадобитися для використання у інших функціях автоматизації процесів кав’ярні.

5. **Спільнота розробників**: Необхідно враховувати наявність та активність розробницької спільноти, яка підтримує обрану ОС та протокол обміну даними.

Це сприятиме отриманню швидкої та ефективної підтримки, порад та розв'язанню проблем.

У випадку розробки додатків для управління кав’ярнею, популярним вибором ОС може бути iOS, macOS, або Windows. iOS здебільшого використовується в розробці додатків для iPhone та iPad, та має широку підтримку для багатьох мов програмування та інструментів. Windows також є популярним вибором для розробки та має хорошу сумісність з багатьма сервісами та інструментами.

У будь-якому випадку, вибір ОС та протоколу обміну даними повинен відповідати вимогам вашого проекту та забезпечувати надійну та безпечну роботу додатку.

3.3.4. Адміністрування системою

Адміністрування системою в контексті розробки додатку включає керування та підтримку інфраструктури, бази даних, безпеки та інших аспектів системи.

Основні завдання адміністрування системи включають:

- Керування інфраструктурою: Це означає налаштування та керування серверами, мережами, хостингом та іншими інфраструктурними компонентами, необхідними для безперебійної роботи додатка для управління кав'ярнею. Важливо забезпечити оптимальну масштабованість, доступність та продуктивність системи.
- Безпека: Забезпечення безпеки системи є важливим аспектом, на який потрібно звернути особливу увагу. Це включає різні заходи для захисту від зламів, зловживань та інших загроз. Використання відповідних заходів безпеки, таких як шифрування даних, захист від вразливостей та системний моніторинг, допоможе гарантувати безпеку даних у вашому додатку.
- База даних: Адміністрування бази даних включає створення, оновлення та управління базою даних, яка зберігає дані про користувачів, замовлення,

товари та інші відомості. Важливо забезпечити резервне копіювання даних, моніторинг продуктивності бази даних та оптимізацію запитів для швидкого доступу до інформації.

- Моніторинг та аналіз: Моніторинг системи допоможе виявляти проблеми, відстежувати продуктивність та забезпечувати належну реагування на випадки неполадок. Аналіз даних може надати цінну інформацію про використання системи, поведінку користувачів та ефективність функцій.
- Підтримка та усунення несправностей: Адміністратор повинен бути готовим надавати підтримку користувачам, відповідати на їх запити та вирішувати проблеми, що виникають під час використання додатку. Це може включати відповіді на запитання, вирішення технічних проблем та надання консультацій.

3.3.5. Заходи захисту від несанкціонованого доступу до системи

Для забезпечення захисту від несанкціонованого доступу до системи можна використовувати наступні заходи безпеки:

- Вимога до сильних паролів: Рекомендується встановлювати вимоги до складних паролів, які містять комбінацію великих і малих літер, цифр та спеціальних символів. Користувачам слід регулярно змінювати свої паролі.
- Використання двофакторної аутентифікації: Рекомендується ввімкнути двофакторну аутентифікацію, яка вимагає додаткового коду або використання аутентифікаційного пристрою після введення пароля.
- Обмеження доступу: Налаштування прав доступу до системи для авторизованих користувачів і обмеження їхніх прав відповідно до їхніх ролей.
- Моніторинг активності: Встановлення системи моніторингу, яка виявляє незвичну або підозрілу активність, наприклад, невдалі спроби входу або зміни конфіденційної інформації.

Ці заходи безпеки сприяють забезпеченню захищеності системи від потенційних загроз і недозволеного доступу.

РОЗДІЛ 4. ОХОРОНА ПРАЦІ

Охорона праці в контексті компанії, що займається розробкою додатків для управління кав'ярнями, повинна включати наступні аспекти:

1. Оцінка ризиків: Проведення оцінки ризиків, пов'язаних з усіма етапами розробки, тестування та експлуатації додатків. Виявлення потенційних небезпек та прийняття заходів для їх усунення або зниження ризику.
2. Захист інформації: Забезпечення конфіденційності та захисту інформації, що обробляється системою, включаючи дані компанії та особисті дані користувачів. Застосування шифрування, мережевих заходів захисту та систем виявлення вторгнень для забезпечення безпеки даних.
3. Безпека програмного забезпечення: Використання найкращих практик безпеки під час розробки та тестування додатків. Перевірка на наявність вразливостей, встановлення патчів та оновлень для запобігання вторгнень та зловживань.
4. Безпека робочого середовища: Забезпечення безпечних умов праці для співробітників, зайнятих розробкою та підтримкою додатків. Дотримання норм охорони праці, надання необхідного обладнання та захисних засобів.
5. Навчання та свідомість: Проведення навчання співробітників щодо охорони праці, безпеки інформації, захисту даних та кібербезпеки. Популяризація кращих практик та політик безпеки серед персоналу.
6. Аварійна готовність: Розробка планів та процедур реагування на аварійні ситуації, включаючи відновлення системи після проблем або збоїв. Проведення регулярних практичних навчань та тестування системи в аварійних ситуаціях.
7. Медичне обслуговування: Забезпечення доступу до медичного обслуговування та надання першої медичної допомоги співробітникам у разі потреби.

ВИСНОВКИ

У рамках цієї дипломної роботи успішно розроблено та реалізовано додаток для управління кав'ярнею. Для досягнення поставлених завдань був проведений аналіз вимог користувачів, вибір необхідних технологій та розробка функціоналу, що дозволяє створювати розклади, розраховувати заробітну плату та контролювати залишок продуктів та інгредієнтів для ефективного функціонування кав'ярні.

Розроблений додаток надає користувачам зручний інструмент для створення розкладів, розрахунку заробітної плати та контролю залишку продуктів та інгредієнтів. Перспективи розробленої системи полягають у подальшому розширенні функціоналу, покращенні інтерфейсу та введенні нових можливостей для користувачів. Застосування цього додатку може бути корисним для різних галузей бізнесу, що працюють у сфері надання послуг, таких як кав'ярні, кафе, ресторани та інші.

Очікується, що впровадження розробленої системи сприятиме спрощенню процесу управління, підвищенню ефективності бізнесу та розширенню можливостей. Рекомендації щодо практичного використання результатів дипломної роботи включають подальшу оптимізацію та розширення функціоналу додатку залежно від конкретних потреб бізнес-сценаріїв. Також рекомендується провести тестування та впровадження розробленої системи в реальних умовах, залучивши партнерів та клієнтів для отримання об'єктивної оцінки її ефективності та придатності.

У процесі виконання дипломної роботи були використані наукові статті, публікації та документація щодо управління кав'ярнями та розробки додатків на мові Swift. Всі використані джерела детально зазначені в списку використаних джерел, що подані в кінці роботи.

До додатків дипломної роботи включено протоколи тестування, демонстраційні зображення та звіти про апробацію розробленої системи в реальних умовах, а також опис алгоритмів та програмний код для розробленого додатку.

Загалом, результати дипломної роботи підтверджують досягнення поставленої мети щодо розробки та впровадження додатку для управління кав'ярнею. Розроблена система має потенціал для практичного використання в різних галузях бізнесу.

БІБЛІОГРАФІЧНИЙ СПИСОК

1. ДСТУ 3008:2015 — Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання. – К.: ДП «УкрНДНЦ», 2015. – 32 с.
2. ДСТУ 3918:1999 (ISO/IEC 12207:2008). Інформаційні технології. Процеси життєвого циклу програмного забезпечення. – 57 с.
3. ДСТУ ISO/IEC TR 15504. Інформаційні технології. Оцінювання процесів життєвого циклу програмних засобів. – 315 с.
4. ДСТУ 2226:1993. Автоматизовані системи.
5. ДСТУ ISO/IEC 27000:2015. Інформаційні технології. Методи захисту. Система управління інформаційною безпекою. Огляд і словник.
6. ДСТУ ISO/IEC 12207:2016 (ISO/IEC 12207:2008, IDT). Інженерія систем і програмного забезпечення. Процеси життєвого циклу програмного забезпечення.
7. ДСТУ ISO/IEC/IEEE 29119-1:2017. Інженерія систем і програмних засобів. Тестування програмних засобів. Частина 1. Поняття та визначення (ISO/IEC/IEEE 29119-1:2013, IDT).
8. ДСТУ 2941:1994. Системи оброблення інформації. Розроблення систем. Терміни та визначення.
9. ДСТУ 1.0:2003. СТУ 1.0:2003. Національна стандартизація. Основні положення.
10. ДСТУ 3321:2003. Системи конструкторської документації. Терміни та визначення основних понять.

11. ДСТУ ISO 6309:2007. Пожежна безпека. Загальні вимоги.
12. ДСТУ ISO/IEC 29155-1:2015. Розроблення систем і програмного забезпечення. Платформи для тестування проектів з розроблення інформаційних систем. Частина 1. Концепції та визначення.
13. ДСТУ ISO/IEC 12207:2014. Інженерія систем і програмного забезпечення. Процеси життєвого циклу програмного забезпечення.
14. ДСТУ ISO/IEC 15910:2012. Інформаційні технології. Документування програм. Документація користувача.
15. Гайна Г. А. Основи проектування баз даних : навч. посіб. – К. : КНУБА, 2005. – 204 с.
16. Пасічник В. В., Резніченко В. А. Організація баз даних та знань. – К. : BHW, 2006. – 384 с.
17. М'якшило О.М. CASE-технології у проектуванні інформаційних систем: електронний навчальний посібник для студ. вищих навч. закладів / О.М. М'якшило, Л.Г. Загоровська,– К.: НУХТ, 2017. – 190 с.
18. Кафедра інформаційних технологій, штучного інтелекту і кібербезпеки НУХТ [Електронний ресурс]. Режим доступу до ресурсу - https://nuft.edu.ua/fakultet-aks/kafedra-nformaczjnix-sistem/
19. Методичні вказівки до виконання лабораторної роботи [Електронний ресурс]. Режим доступу до ресурсу - https://cde.nuft.edu.ua/pluginfile.php?file=%2F469947%2Fmod_resource%2Fcontent%2F1%2F%D0%9B%D0%B0%D0%B1_%E2%84%966_1%D1%81%D0%B5%D0%BC.pdf
20. Методичні вказівки до виконання лабораторної роботи [Електронний ресурс]. Режим доступу до ресурсу - https://cde.nuft.edu.ua/mod/resource/view.php?id=623391
21. Кафедра інформаційних технологій, штучного інтелекту і кібербезпеки НУХТ [Електронний ресурс]. Режим доступу до ресурсу - https://nuft.edu.ua/fakultet-aks/kafedra-nformaczjnix-sistem/

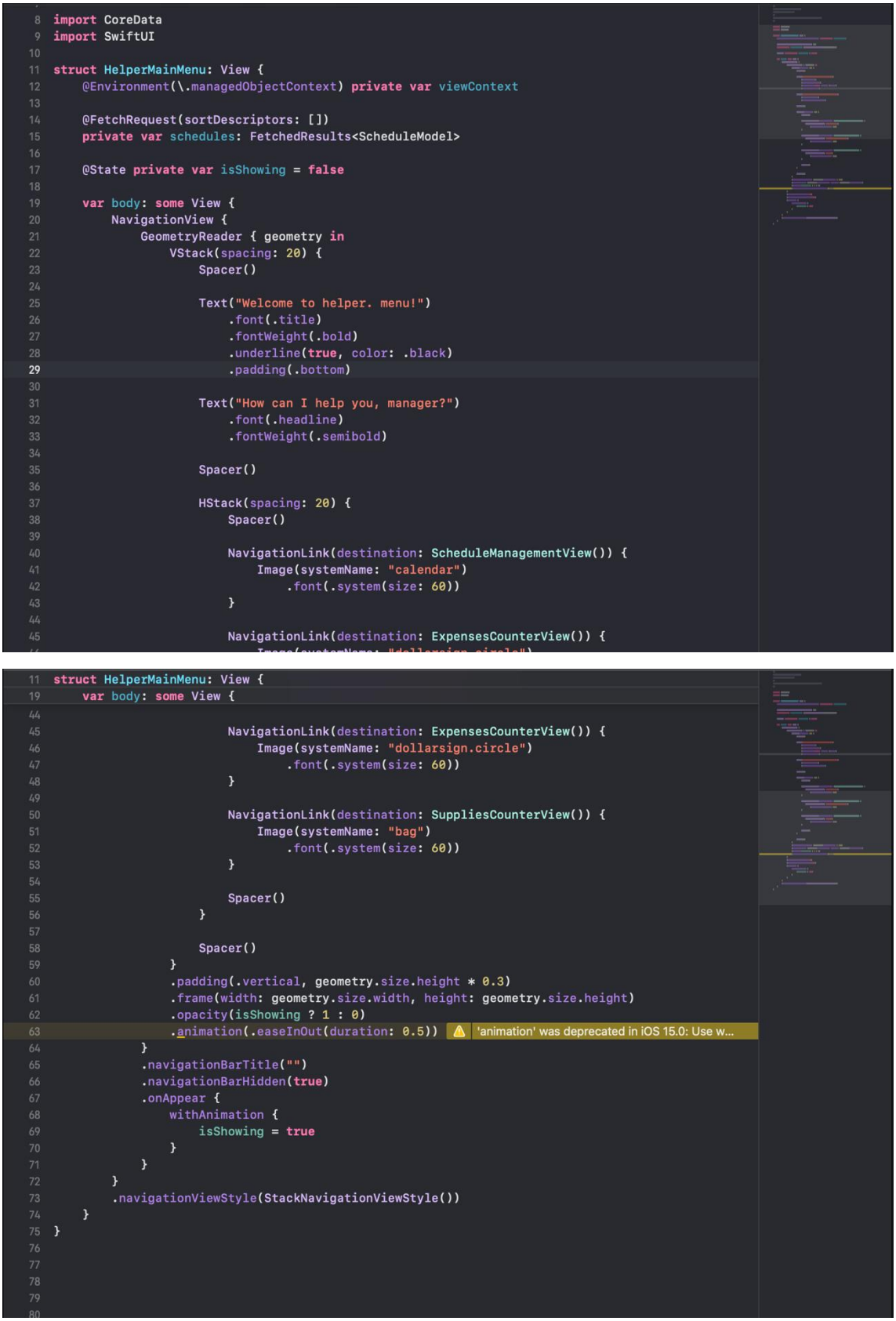
22. Порівняння систем управління проектами - https://uk.myservername.com/top-10-best-warehouse-management-software-systems-2021#WMS_Comparison_Table
23. Вказівки до роботи з середовищем розробки Apple xCode - https://developer.apple.com/documentation/xcode/creating-an-xcode-project-for-an-app
24. Вказівки до роботи з Core Data бібліотекою. - https://developer.apple.com/documentation/coredata
25. Вказівки до роботи з SwiftUI - https://developer.apple.com/tutorials/swiftui
26. Вказівки до роботи з повідомленнями (User Notification) - https://developer.apple.com/documentation/usernotifications

ДОДАТОК А

```swift
import Foundation
import SwiftUI

struct LoginView: View {
    @State private var username = ""
    @State private var password = ""
    @ObservedObject var loginState: LoginState
    @State private var showAlert = false

    var body: some View {
        NavigationView {
            VStack {
                Text("Welcome to helper.")
                    .font(.title)
                    .fontWeight(.bold)
                    .padding()

                Text("Login, please.")
                    .font(.headline)
                    .fontWeight(.bold)
                    .foregroundColor(.gray)
                    .padding()

                TextField("Username", text: $username)
                    .textFieldStyle(RoundedBorderTextFieldStyle())
                    .padding()

                SecureField("Password", text: $password)
                    .textFieldStyle(RoundedBorderTextFieldStyle())
                    .padding()

                Button("Enter") {
                    login()
                }
                .padding()
                .foregroundColor(.white)
                .background(Color.blue)
                .cornerRadius(10)
```

```swift
struct LoginView: View {
    var body: some View {
                }
                .padding()
                .foregroundColor(.white)
                .background(Color.blue)
                .cornerRadius(10)
                .padding(.top, 20)
            }
            .padding()
            .alert(isPresented: $showAlert) {
                Alert(
                    title: Text("Error"),
                    message: Text("Invalid username or password."),
                    dismissButton: .default(Text("OK"))
                )
            }
        }
    }

    private func login() {
        if username == "Manager" && password == "manager1" {
            // Manager login successful
            loginState.isLoggedIn = true
        } else if username == "Employee" && password == "employee1" {
            // Employee login successful
            loginState.isLoggedIn = true
        } else {
            showAlert = true
        }
    }
}

class LoginState: ObservableObject {
    @Published var isLoggedIn = false
}
```

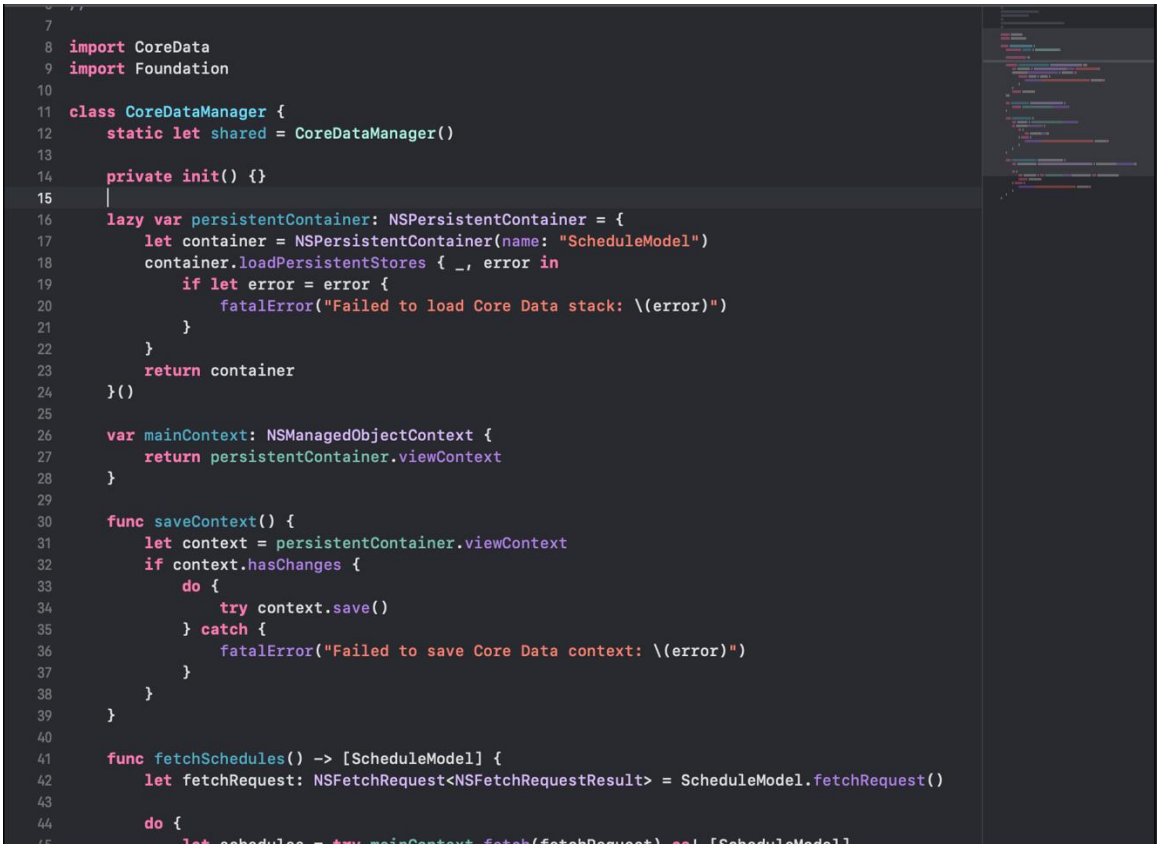
Додаток А. Код файлу ‘HelperLoginView.swift’.



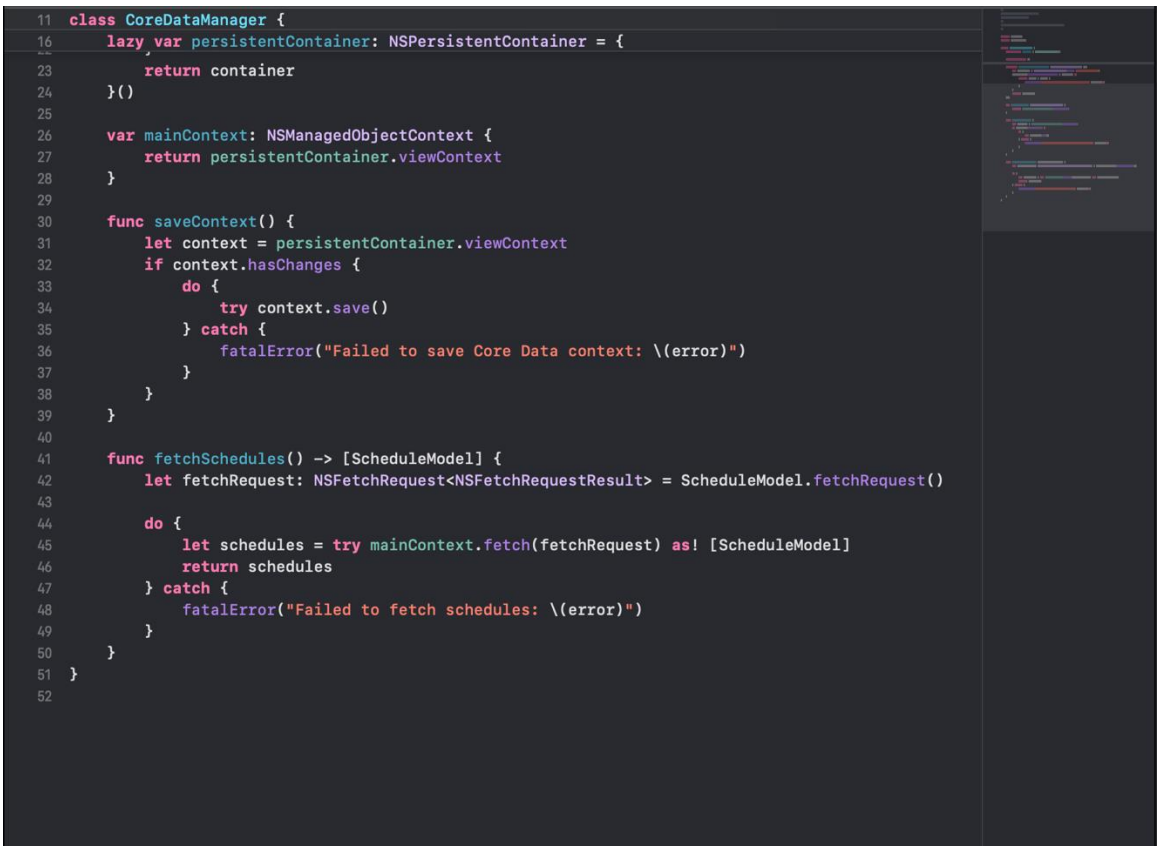
```swift
import CoreData
import SwiftUI

struct HelperMainMenu: View {
    @Environment(\.managedObjectContext) private var viewContext

    @FetchRequest(sortDescriptors: [])
    private var schedules: FetchedResults<ScheduleModel>

    @State private var isShowing = false

    var body: some View {
        NavigationView {
            GeometryReader { geometry in
                VStack(spacing: 20) {
                    Spacer()

                    Text("Welcome to helper. menu!")
                        .font(.title)
                        .fontWeight(.bold)
                        .underline(true, color: .black)
                        .padding(.bottom)

                    Text("How can I help you, manager?")
                        .font(.headline)
                        .fontWeight(.semibold)

                    Spacer()

                    HStack(spacing: 20) {
                        Spacer()

                        NavigationLink(destination: ScheduleManagementView()) {
                            Image(systemName: "calendar")
                                .font(.system(size: 60))
                        }

                        NavigationLink(destination: ExpensesCounterView()) {
```

```swift
struct HelperMainMenu: View {
    var body: some View {

                        NavigationLink(destination: ExpensesCounterView()) {
                            Image(systemName: "dollarsign.circle")
                                .font(.system(size: 60))
                        }

                        NavigationLink(destination: SuppliesCounterView()) {
                            Image(systemName: "bag")
                                .font(.system(size: 60))
                        }

                        Spacer()
                    }

                    Spacer()
                }
                .padding(.vertical, geometry.size.height * 0.3)
                .frame(width: geometry.size.width, height: geometry.size.height)
                .opacity(isShowing ? 1 : 0)
                .animation(.easeInOut(duration: 0.5)) 'animation' was deprecated in iOS 15.0: Use w...
            }
            .navigationBarTitle("")
            .navigationBarHidden(true)
            .onAppear {
                withAnimation {
                    isShowing = true
                }
            }
        }
        .navigationViewStyle(StackNavigationViewStyle())
    }
}
```

Додаток А. Код файлу ‘HelperMainMenu.swift’.

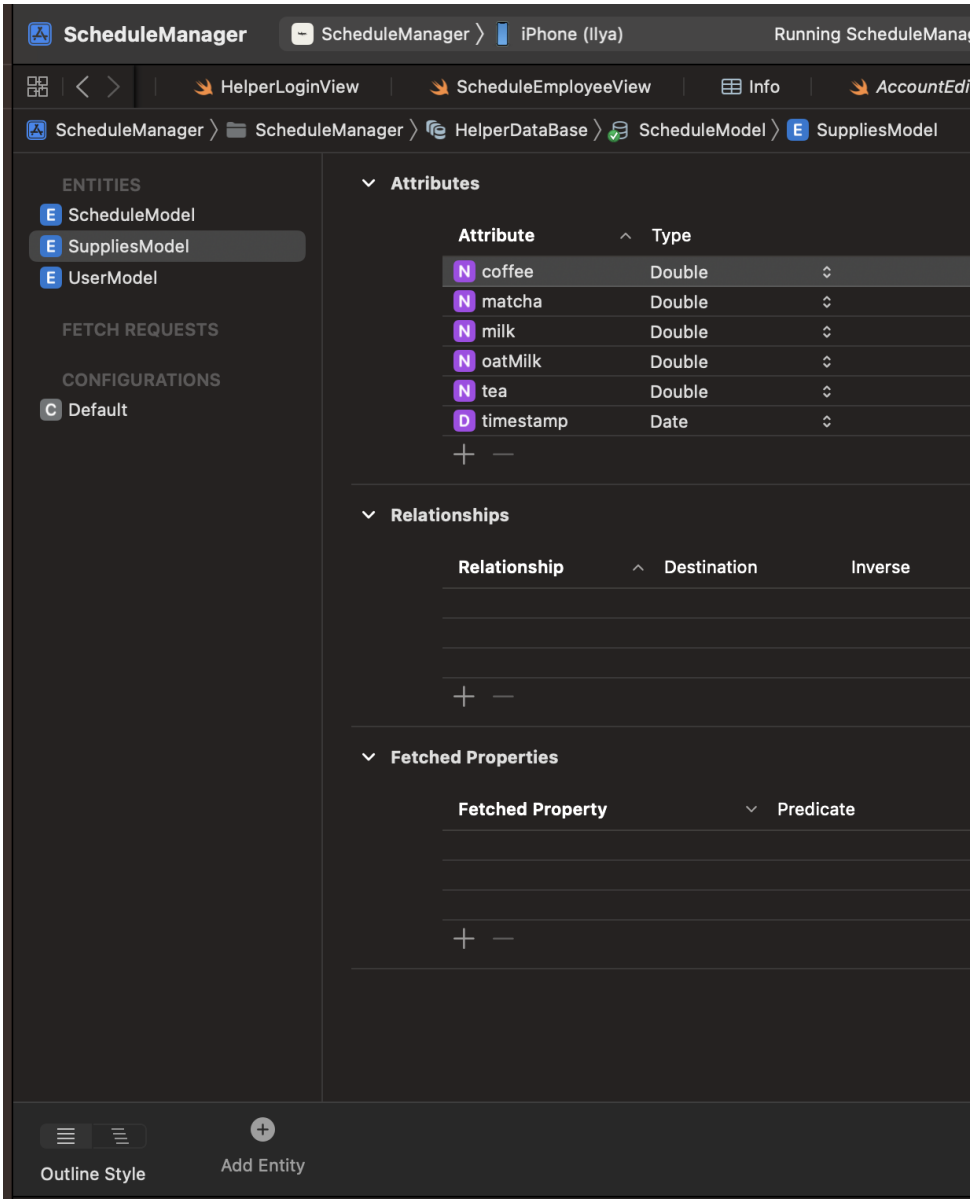


```swift
import CoreData
import Foundation

class CoreDataManager {
    static let shared = CoreDataManager()

    private init() {}

    lazy var persistentContainer: NSPersistentContainer = {
        let container = NSPersistentContainer(name: "ScheduleModel")
        container.loadPersistentStores { _, error in
            if let error = error {
                fatalError("Failed to load Core Data stack: \(error)")
            }
        }
        return container
    }()

    var mainContext: NSManagedObjectContext {
        return persistentContainer.viewContext
    }

    func saveContext() {
        let context = persistentContainer.viewContext
        if context.hasChanges {
            do {
                try context.save()
            } catch {
                fatalError("Failed to save Core Data context: \(error)")
            }
        }
    }

    func fetchSchedules() -> [ScheduleModel] {
        let fetchRequest: NSFetchRequest<NSFetchRequestResult> = ScheduleModel.fetchRequest()

        do {
```



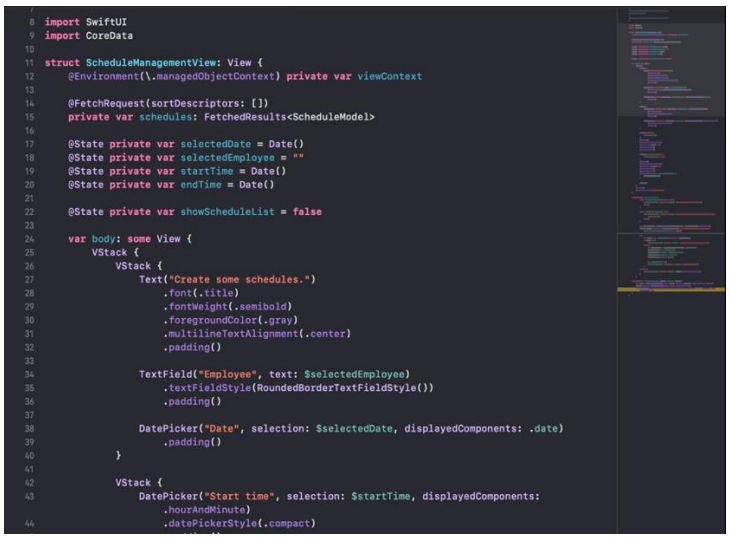
```swift
class CoreDataManager {
    lazy var persistentContainer: NSPersistentContainer = {
        return container
    }()

    var mainContext: NSManagedObjectContext {
        return persistentContainer.viewContext
    }

    func saveContext() {
        let context = persistentContainer.viewContext
        if context.hasChanges {
            do {
                try context.save()
            } catch {
                fatalError("Failed to save Core Data context: \(error)")
            }
        }
    }

    func fetchSchedules() -> [ScheduleModel] {
        let fetchRequest: NSFetchRequest<NSFetchRequestResult> = ScheduleModel.fetchRequest()

        do {
            let schedules = try mainContext.fetch(fetchRequest) as! [ScheduleModel]
            return schedules
        } catch {
            fatalError("Failed to fetch schedules: \(error)")
        }
    }
}
```

Додаток А. Код файлу ‘CoreDataManager.swift’.

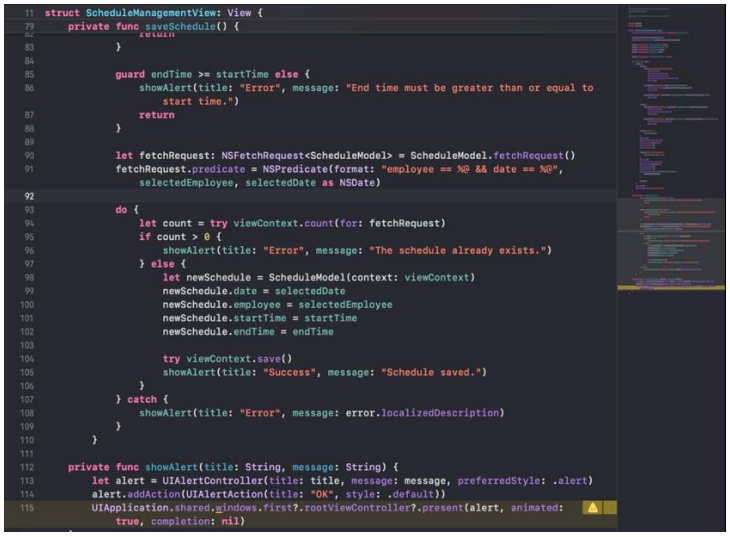


Додаток А. *Приклад налаштування бази даних ‘SuppliesModel’ у файлі ‘HelperDataBase’.*

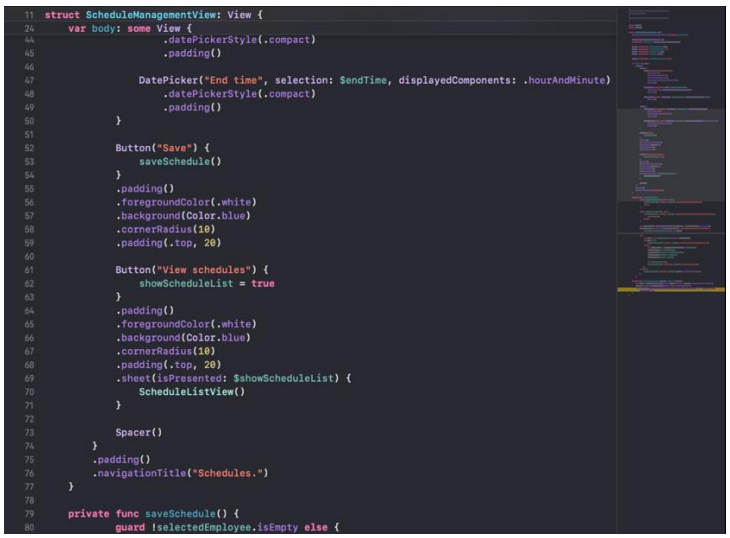
ДОДАТОК Б



```swift
import SwiftUI
import CoreData

struct ScheduleManagementView: View {
    @Environment(\.managedObjectContext) private var viewContext

    @FetchRequest(sortDescriptors: [])
    private var schedules: FetchedResults<ScheduleModel>

    @State private var selectedDate = Date()
    @State private var selectedEmployee = ""
    @State private var startTime = Date()
    @State private var endTime = Date()

    @State private var showScheduleList = false

    var body: some View {
        VStack {
            VStack {
                Text("Create some schedules.")
                    .font(.title)
                    .fontWeight(.semibold)
                    .foregroundColor(.gray)
                    .multilineTextAlignment(.center)
                    .padding()

                TextField("Employee", text: $selectedEmployee)
                    .textFieldStyle(RoundedBorderTextFieldStyle())
                    .padding()

                DatePicker("Date", selection: $selectedDate, displayedComponents: .date)
                    .padding()
            }

            VStack {
                DatePicker("Start time", selection: $startTime, displayedComponents:
                    .hourAndMinute)
                    .datePickerStyle(.compact)
```

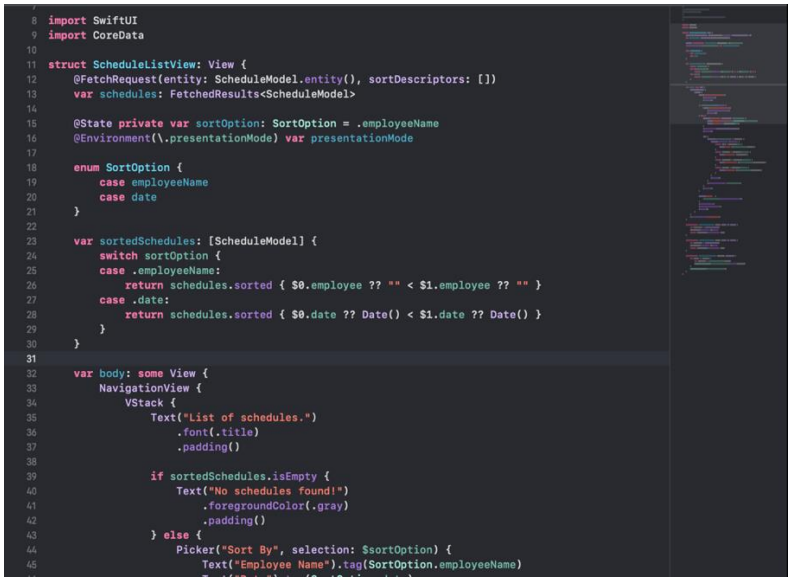


```swift
struct ScheduleManagementView: View {
    private func saveSchedule() {
            }

            guard endTime >= startTime else {
                showAlert(title: "Error", message: "End time must be greater than or equal to
                    start time.")
                return
            }

            let fetchRequest: NSFetchRequest<ScheduleModel> = ScheduleModel.fetchRequest()
            fetchRequest.predicate = NSPredicate(format: "employee == %@ && date == %@",
                selectedEmployee, selectedDate as NSDate)

            do {
                let count = try viewContext.count(for: fetchRequest)
                if count > 0 {
                    showAlert(title: "Error", message: "The schedule already exists.")
                } else {
                    let newSchedule = ScheduleModel(context: viewContext)
                    newSchedule.date = selectedDate
                    newSchedule.employee = selectedEmployee
                    newSchedule.startTime = startTime
                    newSchedule.endTime = endTime

                    try viewContext.save()
                    showAlert(title: "Success", message: "Schedule saved.")
                }
            } catch {
                showAlert(title: "Error", message: error.localizedDescription)
            }
        }

    private func showAlert(title: String, message: String) {
        let alert = UIAlertController(title: title, message: message, preferredStyle: .alert)
        alert.addAction(UIAlertAction(title: "OK", style: .default))
        UIApplication.shared.windows.first?.rootViewController?.present(alert, animated:
            true, completion: nil)
```

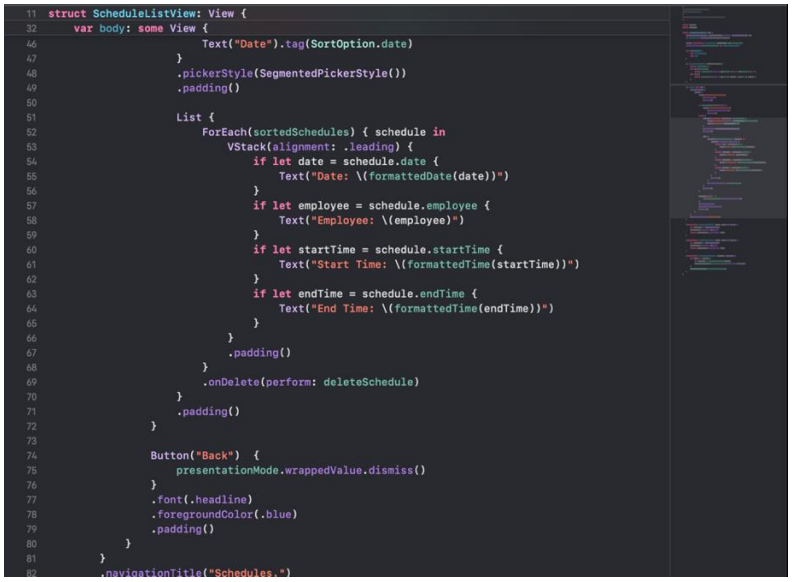


```swift
struct ScheduleManagementView: View {
    var body: some View {
                    .datePickerStyle(.compact)
                    .padding()

                DatePicker("End time", selection: $endTime, displayedComponents: .hourAndMinute)
                    .datePickerStyle(.compact)
                    .padding()
            }

            Button("Save") {
                saveSchedule()
            }
            .padding()
            .foregroundColor(.white)
            .background(Color.blue)
            .cornerRadius(10)
            .padding(.top, 20)

            Button("View schedules") {
                showScheduleList = true
            }
            .padding()
            .foregroundColor(.white)
            .background(Color.blue)
            .cornerRadius(10)
            .padding(.top, 20)
            .sheet(isPresented: $showScheduleList) {
                ScheduleListView()
            }

            Spacer()
        }
        .padding()
        .navigationTitle("Schedules.")
    }

    private func saveSchedule() {
            guard !selectedEmployee.isEmpty else {
```

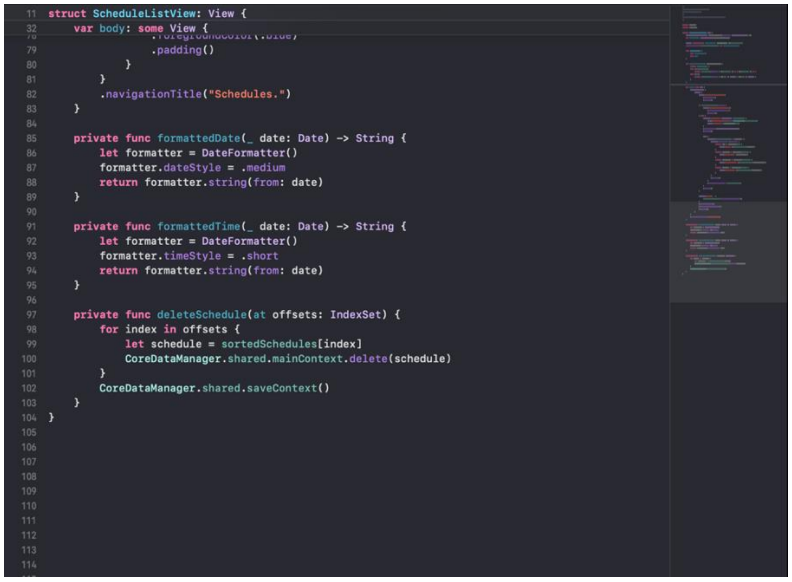
Додаток Б. Код файлу ‘ScheduleManagementView.swift’



```swift
import SwiftUI
import CoreData

struct ScheduleListView: View {
    @FetchRequest(entity: ScheduleModel.entity(), sortDescriptors: [])
    var schedules: FetchedResults<ScheduleModel>

    @State private var sortOption: SortOption = .employeeName
    @Environment(\.presentationMode) var presentationMode

    enum SortOption {
        case employeeName
        case date
    }

    var sortedSchedules: [ScheduleModel] {
        switch sortOption {
        case .employeeName:
            return schedules.sorted { $0.employee ?? "" < $1.employee ?? "" }
        case .date:
            return schedules.sorted { $0.date ?? Date() < $1.date ?? Date() }
        }
    }

    var body: some View {
        NavigationView {
            VStack {
                Text("List of schedules.")
                    .font(.title)
                    .padding()

                if sortedSchedules.isEmpty {
                    Text("No schedules found!")
                        .foregroundColor(.gray)
                        .padding()
                } else {
                    Picker("Sort By", selection: $sortOption) {
                        Text("Employee Name").tag(SortOption.employeeName)
```



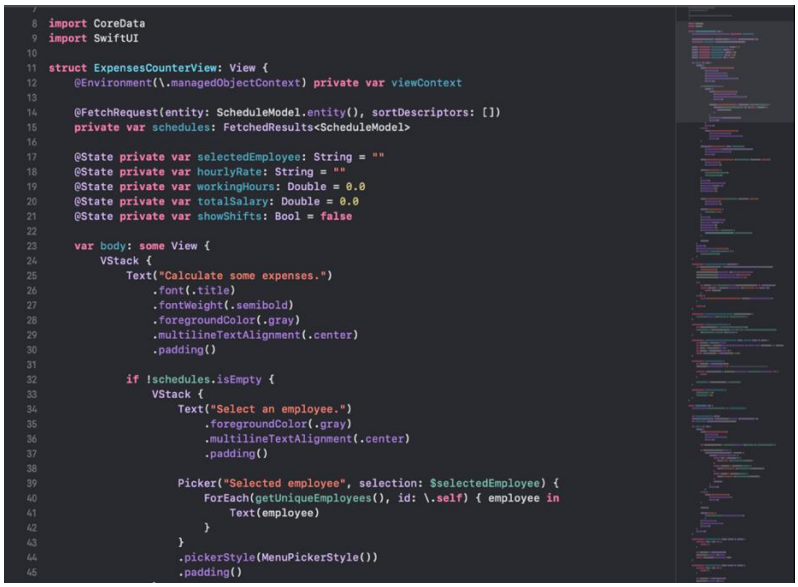
```swift
struct ScheduleListView: View {
    var body: some View {
                        Text("Date").tag(SortOption.date)
                    }
                    .pickerStyle(SegmentedPickerStyle())
                    .padding()

                    List {
                        ForEach(sortedSchedules) { schedule in
                            VStack(alignment: .leading) {
                                if let date = schedule.date {
                                    Text("Date: \(formattedDate(date))")
                                }
                                if let employee = schedule.employee {
                                    Text("Employee: \(employee)")
                                }
                                if let startTime = schedule.startTime {
                                    Text("Start Time: \(formattedTime(startTime))")
                                }
                                if let endTime = schedule.endTime {
                                    Text("End Time: \(formattedTime(endTime))")
                                }
                            }
                            .padding()
                        }
                        .onDelete(perform: deleteSchedule)
                    }
                    .padding()
                }

                Button("Back")  {
                    presentationMode.wrappedValue.dismiss()
                }
                .font(.headline)
                .foregroundColor(.blue)
                .padding()
            }
        }
        .navigationTitle("Schedules.")
```



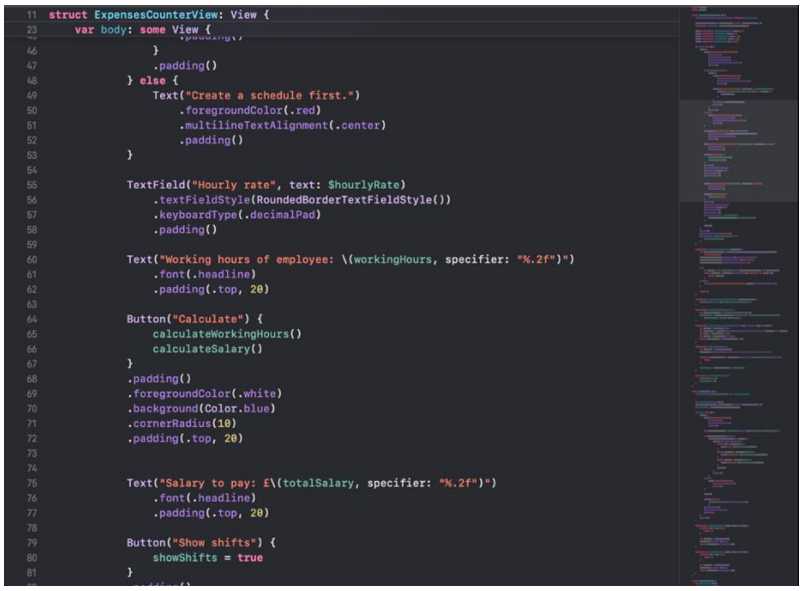
```swift
struct ScheduleListView: View {
    var body: some View {
                .padding()
            }
        }
        .navigationTitle("Schedules.")
    }

    private func formattedDate(_ date: Date) -> String {
        let formatter = DateFormatter()
        formatter.dateStyle = .medium
        return formatter.string(from: date)
    }

    private func formattedTime(_ date: Date) -> String {
        let formatter = DateFormatter()
        formatter.timeStyle = .short
        return formatter.string(from: date)
    }

    private func deleteSchedule(at offsets: IndexSet) {
        for index in offsets {
            let schedule = sortedSchedules[index]
            CoreDataManager.shared.mainContext.delete(schedule)
        }
        CoreDataManager.shared.saveContext()
    }
}
```

Додаток Б. Код файлу 'ScheduleListView.swift'

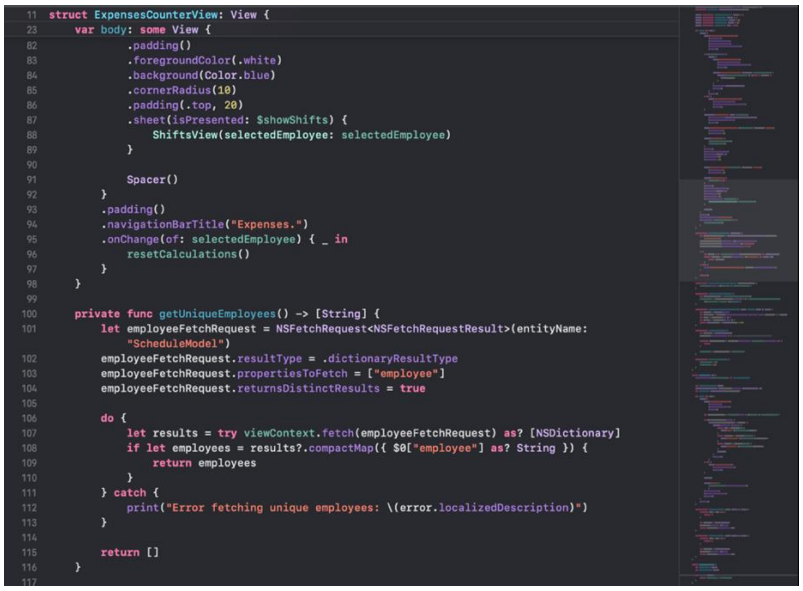
ДОДАТОК В



```swift
import CoreData
import SwiftUI

struct ExpensesCounterView: View {
    @Environment(\.managedObjectContext) private var viewContext

    @FetchRequest(entity: ScheduleModel.entity(), sortDescriptors: [])
    private var schedules: FetchedResults<ScheduleModel>

    @State private var selectedEmployee: String = ""
    @State private var hourlyRate: String = ""
    @State private var workingHours: Double = 0.0
    @State private var totalSalary: Double = 0.0
    @State private var showShifts: Bool = false

    var body: some View {
        VStack {
            Text("Calculate some expenses.")
                .font(.title)
                .fontWeight(.semibold)
                .foregroundColor(.gray)
                .multilineTextAlignment(.center)
                .padding()

            if !schedules.isEmpty {
                VStack {
                    Text("Select an employee.")
                        .foregroundColor(.gray)
                        .multilineTextAlignment(.center)
                        .padding()

                    Picker("Selected employee", selection: $selectedEmployee) {
                        ForEach(getUniqueEmployees(), id: \.self) { employee in
                            Text(employee)
                        }
                    }
                    .pickerStyle(MenuPickerStyle())
                    .padding()
```



```swift
struct ExpensesCounterView: View {
    var body: some View {
                }
                .padding()
            } else {
                Text("Create a schedule first.")
                    .foregroundColor(.red)
                    .multilineTextAlignment(.center)
                    .padding()
            }

            TextField("Hourly rate", text: $hourlyRate)
                .textFieldStyle(RoundedBorderTextFieldStyle())
                .keyboardType(.decimalPad)
                .padding()

            Text("Working hours of employee: \(workingHours, specifier: "%.2f")")
                .font(.headline)
                .padding(.top, 20)

            Button("Calculate") {
                calculateWorkingHours()
                calculateSalary()
            }
            .padding()
            .foregroundColor(.white)
            .background(Color.blue)
            .cornerRadius(10)
            .padding(.top, 20)


            Text("Salary to pay: £\(totalSalary, specifier: "%.2f")")
                .font(.headline)
                .padding(.top, 20)

            Button("Show shifts") {
                showShifts = true
            }
```



```swift
struct ExpensesCounterView: View {
    var body: some View {
            .padding()
            .foregroundColor(.white)
            .background(Color.blue)
            .cornerRadius(10)
            .padding(.top, 20)
            .sheet(isPresented: $showShifts) {
                ShiftsView(selectedEmployee: selectedEmployee)
            }

            Spacer()
        }
        .padding()
        .navigationBarTitle("Expenses.")
        .onChange(of: selectedEmployee) { _ in
            resetCalculations()
        }
    }

    private func getUniqueEmployees() -> [String] {
        let employeeFetchRequest = NSFetchRequest<NSFetchRequestResult>(entityName:
            "ScheduleModel")
        employeeFetchRequest.resultType = .dictionaryResultType
        employeeFetchRequest.propertiesToFetch = ["employee"]
        employeeFetchRequest.returnsDistinctResults = true

        do {
            let results = try viewContext.fetch(employeeFetchRequest) as? [NSDictionary]
            if let employees = results?.compactMap({ $0["employee"] as? String }) {
                return employees
            }
        } catch {
            print("Error fetching unique employees: \(error.localizedDescription)")
        }

        return []
    }
```

```swift
struct ExpensesCounterView: View {
    private func selectedEmployeeSchedules() -> [ScheduleModel] {
        schedules.filter { $0.employee == selectedEmployee }
    }

    private func calculateWorkingHours() {
        let selectedSchedules = selectedEmployeeSchedules()
        workingHours = selectedSchedules.reduce(0) { $0 + workingHoursBetween(startTime:
            $1.startTime!, endTime: $1.endTime!) }
    }

    private func workingHoursBetween(startTime: Date, endTime: Date) -> Double {
        let calendar = Calendar.current
        let components = calendar.dateComponents([.hour, .minute], from: startTime, to: endTime)
        let hours = components.hour ?? 0
        let minutes = components.minute ?? 0
        return Double(hours) + Double(minutes) / 60.0
    }

    private func calculateSalary() {
        let formatter = NumberFormatter()
        formatter.decimalSeparator = "," // Set the decimal separator to comma

        guard let hourlyRateValue = formatter.number(from: hourlyRate)?.doubleValue else {
            return
        }

        totalSalary = hourlyRateValue * workingHours
    }

    private func resetCalculations() {
        workingHours = 0.0
        totalSalary = 0.0
    }
}

struct ShiftsView: View {
    @Environment(\.presentationMode) var presentationMode
```



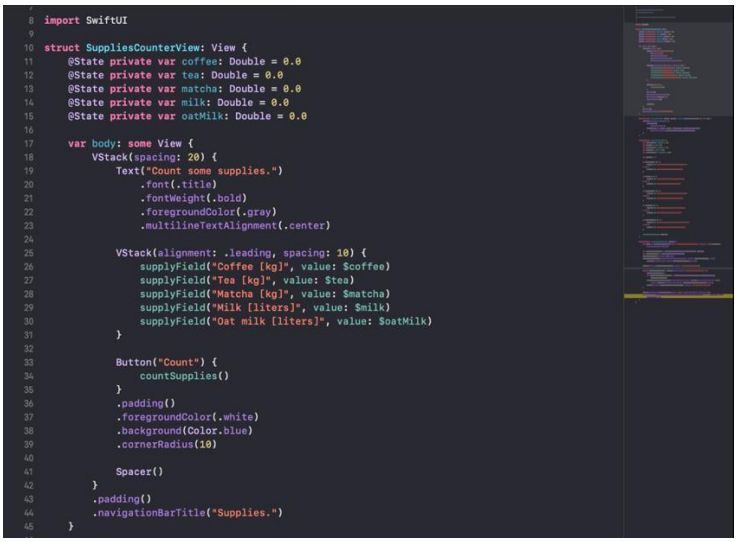
```swift
struct ShiftsView: View {

    var selectedEmployee: String
    @FetchRequest(entity: ScheduleModel.entity(), sortDescriptors: [])
    var schedules: FetchedResults<ScheduleModel>

    var body: some View {
        VStack {
            Text("Employee's shifts.")
                .font(.title)
                .fontWeight(.semibold)
                .padding()

            let employeeSchedules = schedules.filter { $0.employee == selectedEmployee }

            if !employeeSchedules.isEmpty {
                ForEach(employeeSchedules) { schedule in
                    VStack(alignment: .leading) {
                        if let date = schedule.date {
                            Text("Date: \(formattedDate(date))")
                        }
                        if let startTime = schedule.startTime {
                            Text("Start Time: \(formattedTime(startTime))")
                        }
                        if let endTime = schedule.endTime {
                            Text("End Time: \(formattedTime(endTime))")
                        }
                        Divider()
                    }
                    .padding()
                }
            } else {
                Text("No shifts found!")
                    .foregroundColor(.gray)
                    .padding()
            }

            Spacer()
```



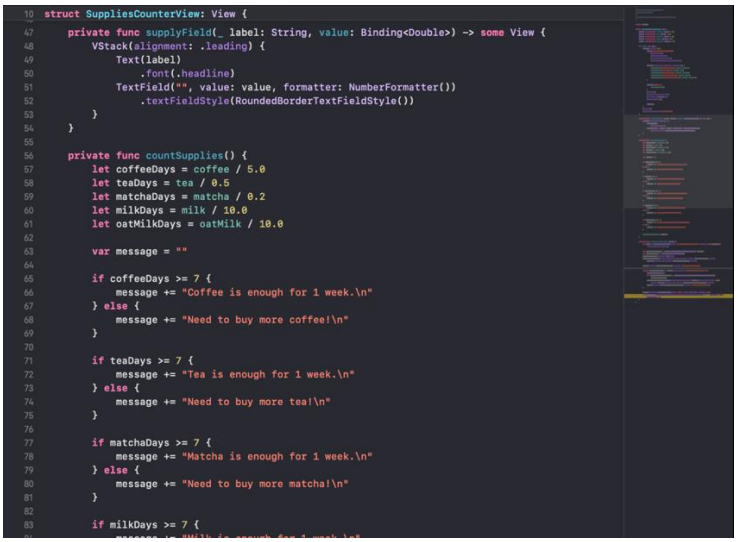
```swift
struct ShiftsView: View {
    var body: some View {
            Button("Back") {
                presentationMode.wrappedValue.dismiss()
                }
            .font(.headline)
            .foregroundColor(.blue)
            .padding()
        }
        .padding()
    }

    private func formattedDate(_ date: Date?) -> String {
        guard let date = date else {
            return ""
        }

        let formatter = DateFormatter()
        formatter.dateStyle = .medium
        return formatter.string(from: date)
    }

    private func formattedTime(_ date: Date?) -> String {
        guard let date = date else {
            return ""
        }

        let formatter = DateFormatter()
        formatter.timeStyle = .short
        return formatter.string(from: date)
    }
}

struct ExpensesCounter {
    let hourlyRate: Double
    let workingHours: Double

    var salary: Double {
        return hourlyRate * workingHours
```

Додаток В. Код файлу 'CounterExpensesView.swift'

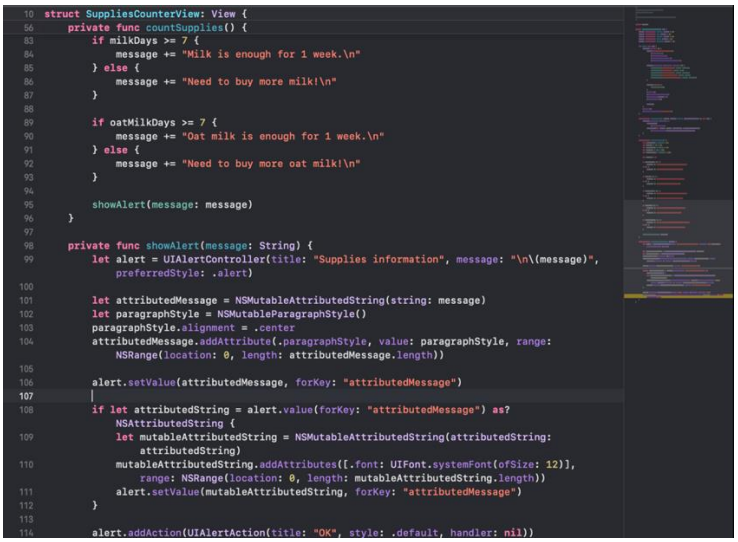
ДОДАТОК Г



```swift
import SwiftUI

struct SuppliesCounterView: View {
    @State private var coffee: Double = 0.0
    @State private var tea: Double = 0.0
    @State private var matcha: Double = 0.0
    @State private var milk: Double = 0.0
    @State private var oatMilk: Double = 0.0

    var body: some View {
        VStack(spacing: 20) {
            Text("Count some supplies.")
                .font(.title)
                .fontWeight(.bold)
                .foregroundColor(.gray)
                .multilineTextAlignment(.center)

            VStack(alignment: .leading, spacing: 10) {
                supplyField("Coffee [kg]", value: $coffee)
                supplyField("Tea [kg]", value: $tea)
                supplyField("Matcha [kg]", value: $matcha)
                supplyField("Milk [liters]", value: $milk)
                supplyField("Oat milk [liters]", value: $oatMilk)
            }

            Button("Count") {
                countSupplies()
            }
            .padding()
            .foregroundColor(.white)
            .background(Color.blue)
            .cornerRadius(10)

            Spacer()
        }
        .padding()
        .navigationBarTitle("Supplies.")
    }
```



```swift
struct SuppliesCounterView: View {
    private func supplyField(_ label: String, value: Binding<Double>) -> some View {
        VStack(alignment: .leading) {
            Text(label)
                .font(.headline)
            TextField("", value: value, formatter: NumberFormatter())
                .textFieldStyle(RoundedBorderTextFieldStyle())
        }
    }

    private func countSupplies() {
        let coffeeDays = coffee / 5.0
        let teaDays = tea / 0.5
        let matchaDays = matcha / 0.2
        let milkDays = milk / 10.0
        let oatMilkDays = oatMilk / 10.0

        var message = ""

        if coffeeDays >= 7 {
            message += "Coffee is enough for 1 week.\n"
        } else {
            message += "Need to buy more coffee!\n"
        }

        if teaDays >= 7 {
            message += "Tea is enough for 1 week.\n"
        } else {
            message += "Need to buy more tea!\n"
        }

        if matchaDays >= 7 {
            message += "Matcha is enough for 1 week.\n"
        } else {
            message += "Need to buy more matcha!\n"
        }

        if milkDays >= 7 {
```



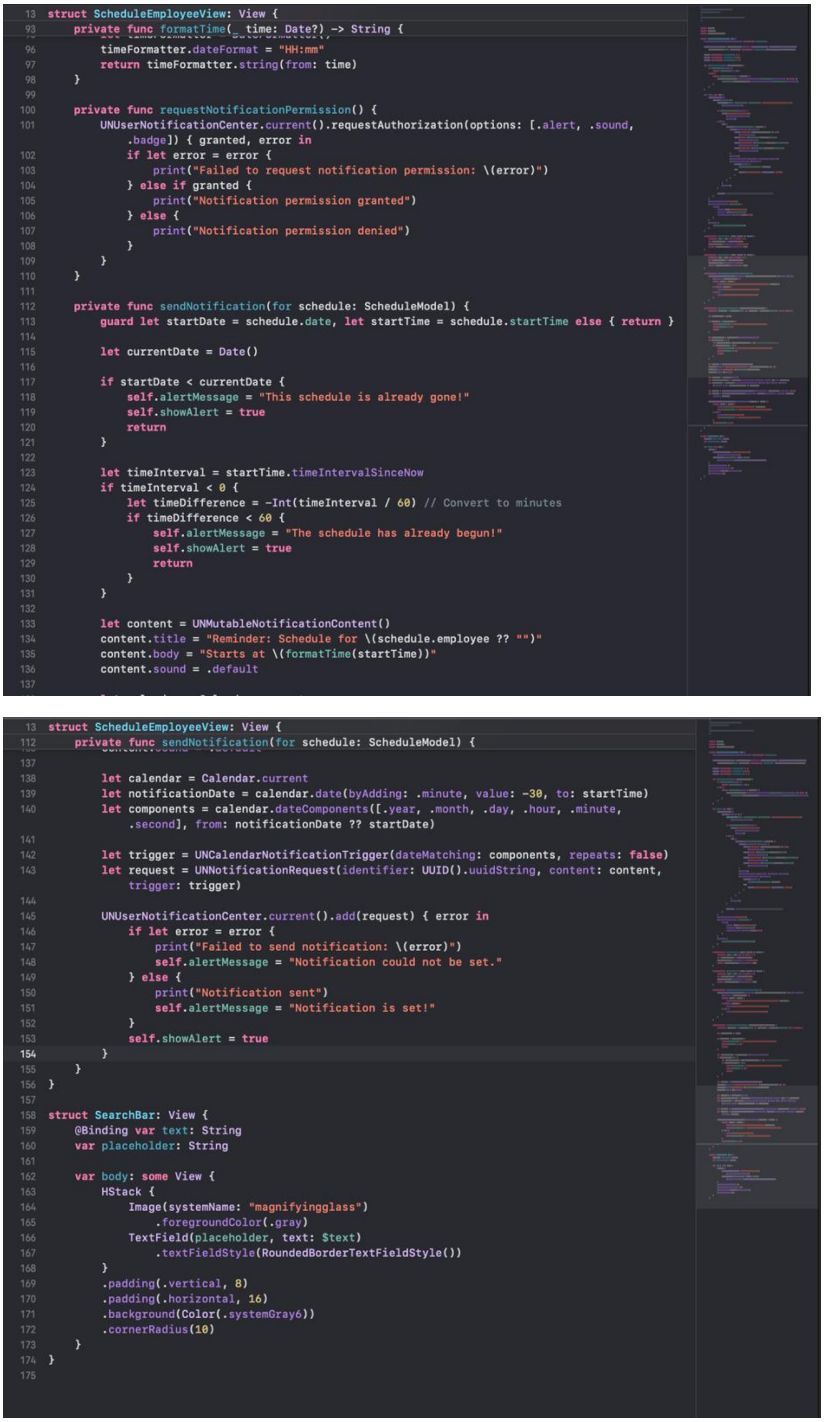
```swift
struct SuppliesCounterView: View {
    private func countSupplies() {
        if milkDays >= 7 {
            message += "Milk is enough for 1 week.\n"
        } else {
            message += "Need to buy more milk!\n"
        }

        if oatMilkDays >= 7 {
            message += "Oat milk is enough for 1 week.\n"
        } else {
            message += "Need to buy more oat milk!\n"
        }

        showAlert(message: message)
    }

    private func showAlert(message: String) {
        let alert = UIAlertController(title: "Supplies information", message: "\n\(message)",
            preferredStyle: .alert)

        let attributedMessage = NSMutableAttributedString(string: message)
        let paragraphStyle = NSMutableParagraphStyle()
        paragraphStyle.alignment = .center
        attributedMessage.addAttribute(.paragraphStyle, value: paragraphStyle, range:
            NSRange(location: 0, length: attributedMessage.length))

        alert.setValue(attributedMessage, forKey: "attributedMessage")

        if let attributedString = alert.value(forKey: "attributedMessage") as?
            NSAttributedString {
            let mutableAttributedString = NSMutableAttributedString(attributedString:
                attributedString)
            mutableAttributedString.addAttributes([.font: UIFont.systemFont(ofSize: 12)],
                range: NSRange(location: 0, length: mutableAttributedString.length))
            alert.setValue(mutableAttributedString, forKey: "attributedMessage")
        }

        alert.addAction(UIAlertAction(title: "OK", style: .default, handler: nil))
```

Додаток Г. Код файлу ‘SuppliesCounterView.swift’

ДОДАТОК Д

```swift
import SwiftUI
import CoreData
import UserNotifications

struct ScheduleEmployeeView: View {
    @Environment(\.managedObjectContext) private var viewContext

    @FetchRequest(entity: ScheduleModel.entity(), sortDescriptors: [NSSortDescriptor(keyPath:
        \ScheduleModel.date, ascending: true)]) var schedules: FetchedResults<ScheduleModel>

    @State private var searchQuery = ""
    @State private var showAlert = false
    @State private var alertMessage = ""

    var filteredSchedules: [ScheduleModel] {
        if searchQuery.isEmpty {
            return schedules.map { $0 }
        } else {
            return schedules.filter { schedule in
                schedule.employee?.lowercased().contains(searchQuery.lowercased()) ?? false ||
                    formatDate(schedule.date).lowercased().contains(searchQuery.lowercased())
            }
        }
    }

    var body: some View {
        NavigationView {
            VStack(spacing: 0) {
                SearchBar(text: $searchQuery, placeholder: "Search by employee or date")
                    .padding(.horizontal)

                if filteredSchedules.isEmpty {
                    Text("No schedules found!")
                        .foregroundColor(.gray)
                        .padding()
                } else {
                    List {
                        ForEach(filteredSchedules) { schedule in
                            VStack(alignment: .leading) {
                                Text("Employee: \(schedule.employee ?? "")")
                                    .font(.headline)
                                Text("Date: \(formatDate(schedule.date))")
                                    .font(.subheadline)
                                Text("Start Time: \(formatTime(schedule.startTime))")
                                    .font(.subheadline)
```

```swift
struct ScheduleEmployeeView: View {
    var body: some View {
                                Text("End Time: \(formatTime(schedule.endTime))")
                                    .font(.subheadline)
                            }
                            .padding()
                            .frame(maxWidth: .infinity, alignment: .leading)
                            .swipeActions(edge: .leading) {
                                Button(action: {
                                    sendNotification(for: schedule)
                                }) {
                                    Label("Set Reminder", systemImage: "bell")
                                }
                            }
                        }
                    }
                    .padding()
                }

                Spacer() // Add spacer to push the content to the top
            }
        }
        .navigationTitle("Schedule")
        .alert(isPresented: $showAlert) {
            Alert(
                title: Text("Notification"),
                message: Text(alertMessage),
                dismissButton: .default(Text("OK"))
            )
        }
        .onAppear {
            requestNotificationPermission()
        }
    }

    private func formatDate(_ date: Date?) -> String {
        guard let date = date else { return "" }
        let dateFormatter = DateFormatter()
        dateFormatter.dateFormat = "dd/MM/yyyy"
        return dateFormatter.string(from: date)
    }

    private func formatTime(_ time: Date?) -> String {
        guard let time = time else { return "" }
        let timeFormatter = DateFormatter()
```



```
struct ScheduleEmployeeView: View {
    private func formatTime(_ time: Date?) -> String {
        timeFormatter.dateFormat = "HH:mm"
        return timeFormatter.string(from: time)
    }

    private func requestNotificationPermission() {
        UNUserNotificationCenter.current().requestAuthorization(options: [.alert, .sound,
            .badge]) { granted, error in
            if let error = error {
                print("Failed to request notification permission: \(error)")
            } else if granted {
                print("Notification permission granted")
            } else {
                print("Notification permission denied")
            }
        }
    }

    private func sendNotification(for schedule: ScheduleModel) {
        guard let startDate = schedule.date, let startTime = schedule.startTime else { return }

        let currentDate = Date()

        if startDate < currentDate {
            self.alertMessage = "This schedule is already gone!"
            self.showAlert = true
            return
        }

        let timeInterval = startTime.timeIntervalSinceNow
        if timeInterval < 0 {
            let timeDifference = -Int(timeInterval / 60) // Convert to minutes
            if timeDifference < 60 {
                self.alertMessage = "The schedule has already begun!"
                self.showAlert = true
                return
            }
        }

        let content = UNMutableNotificationContent()
        content.title = "Reminder: Schedule for \(schedule.employee ?? "")"
        content.body = "Starts at \(formatTime(startTime))"
        content.sound = .default
```



```
struct ScheduleEmployeeView: View {
    private func sendNotification(for schedule: ScheduleModel) {

        let calendar = Calendar.current
        let notificationDate = calendar.date(byAdding: .minute, value: -30, to: startTime)
        let components = calendar.dateComponents([.year, .month, .day, .hour, .minute,
            .second], from: notificationDate ?? startDate)

        let trigger = UNCalendarNotificationTrigger(dateMatching: components, repeats: false)
        let request = UNNotificationRequest(identifier: UUID().uuidString, content: content,
            trigger: trigger)

        UNUserNotificationCenter.current().add(request) { error in
            if let error = error {
                print("Failed to send notification: \(error)")
                self.alertMessage = "Notification could not be set."
            } else {
                print("Notification sent")
                self.alertMessage = "Notification is set!"
            }
            self.showAlert = true
        }
    }
}

struct SearchBar: View {
    @Binding var text: String
    var placeholder: String

    var body: some View {
        HStack {
            Image(systemName: "magnifyingglass")
                .foregroundColor(.gray)
            TextField(placeholder, text: $text)
                .textFieldStyle(RoundedBorderTextFieldStyle())
        }
        .padding(.vertical, 8)
        .padding(.horizontal, 16)
        .background(Color(.systemGray6))
        .cornerRadius(10)
    }
}
```

Додаток Д. Код файлу ‘ScheduleEmployeeView.swift’.