NIT3003
IT Capstone Project

UrSaviour:
Grocery Discount Assistant – Personalized Savings Application

Group_03 / UrSaviour	Contribution
Justin Lee S8115784	28%
Mio Mizutani S8107356	28%
Austin Le S8106392	22%
Aadarsh Thapa S4679035	22%

Submission date 1st Jun 2025

INDEX

2. Revision History

Version	Date	Description of Changes	Reason
1.0	**26th May 2025**	**Initial Document creation**	**-**
1.1	**06th Jun 2025**	**1. Added 'Database Management' as a centralized admin function**	**Need for integrated system monitoring and management**
		2. Added 'Admin Login' for secure access to admin pages	**To enhance security for administrative functions**
		3. Changed front=end technology from Flutter to HTML/CSS/JS	**Standardization of web technologies and project scope alignment**

2. Introduction

2.1 General Background [Mio]

Australian families face significant expense challenges for groceries because of inflation and housing issues. [1]

Major grocery retailers throughout Australia, including Coles and Woolworths, utilize their online and mobile applications to share discount information. [2][3] Customers have to manually check several sources to find out which stores offer the lowest prices. This process not only takes time but can also lead to inaccurate results due to human error. UrSavour offers a one-stop solution that enables users to manage and view grocery discounts while comparing different options.
This distribution method incorporates an easy-to-use system and sends personalized discount alerts through email while providing an AI chatbot to answer user questions.

The website policies of retailers such as Coles and Woolworths include limitations on automated data collection. While Woolworths provides an API portal that requires an application and approval before access, Coles lacks any form of public API availability. [4]

Instead of relying on web scraping, which is restricted by retailer policies, UrSavour creates simulated sale pamphlets using original data. a manually created base dataset of sample grocery items is prepared(e.g., in an Excel). Python script then will process this foundational dataset, applying randomized discount (such as 30% off, half price, or 10% off) to a selection of items on a weekly basis. This Python-generated weekly offer data is then used to create new, simulated PDF pamphlets which are subsequently processed by the systems' Etl pipeline via email.

The method maintains adherence to service terms while promoting the enduring viability of the solution. Through this approach, UrSavour not only improves user convenience but also fills a critical gap in the current market: UrSavour offers a legitimate discount tracking solution that operates through a central trusted platform.

2.2 Market Analysis [Justin]

Based on research findings, Scale of the grocery store was valued at over AUD 125 billion in 2023, and by significant inflation and rising living cost, it makes consumers price-sensitive [5]

Additionally, following a 2022 consumer survey, it was found that Australians normally tend to shop when groceries are on discount.

international consumers who have recently arrived in Australia, they will often struggle to find deals information and to understand how the grocery pricing and promotion mechanisms. And it led to poor purchasing decisions. [6]

This application will provide easy-to-use tools that assist consumer aware and respond to grocery price changes in regular.

2.3 Competitor Analysis [Austin]

Several alternative platforms in Australia market assist consumers to make better decisions with grocery. The most competitors with UrSaviour are:

Frugl: A comprehensive app helps the user to track and compare prices of an item across major chains such as Woolworth, Coles, Aldi. Its advantages are filters functions, comparisons in real-time, and wide retailers range. In terms of the disadvantages, the syncing data among a large amount of stores lead to less accuracy in data, and its depend on scaping entirely causing the potential of legal and reliability risk.

WhichGrocer: A website-based application shares a similar purpose like Frugl which assist the users to find a cost-effective grocery options among those major chains in Australia. The website offers the price comparison with a specific store, and users can easily navigate the website due to its friendly interface. However, the website has limited features for non-subscription, and the performance could be degraded if using real-time comparison.

Grocerize: A minimal website compares in real time just between Coles and Woolworths. Regarding the pros, with less features the response is provided faster and the backend has processing less than the others. Moreover, building, maintaining, deploying the website is simpler because of using stack like Flask or Node.js. On the other hands, it is way more difficult to add new branches without re-architecting the backend. Only can be maintained manually if its structure changes.

Our UrSaviour web-based application gives users a unique experience with mixed personalization, AI assistant feature to elevate the user's interaction experience, a smart alerting system for watchlist items, by utilizing its own process for generating simulated pamphlet data depend on a controlled, internally managed fake dataset, UrSaviour make sures that adherence to ethical data practices and the sustainability of the application, rather than relying on potentially problematic web scarping or fluctuating external open datasets for its core discount information.

2.4 Project Aims & Unique Value [Aadarsh]

UrSaviour may be a centralized, lawfully compliant stage that makes a difference clients make educated basic supply shopping choices. Not at all like conventional markdown following devices, UrSaviour offers a suite of shrewdly highlights outlined to improve client comfort, straightforwardness, and moral information utilization.

Key Highlights:

1. Personalized Watchlists with Programmed Alarms
Clients can track basic supply things and get moment alarms when rebates or cost drops are accessible.

2. Maintainable, Week by week Information Upgrades
To make sure lawful compliance and provide a consistent source of discount information, Ursaviour's platform relies on an internally managed data generation workflow. This involves: 1) A foundational, manually curated fake dataset of grocery items. 2) Python scripts that periodically apply various simulated discounts to items from this foundational dataset. 3)The generation of new PDF pamphlets based on this discounted item data. This controlled approach guarantees ethical data sourcing and provides reliable, up-to-data information for the system discount tracking features, independent of external data source availability or terms of service

3. Slant Investigation Based on Client Behaviour
UrSavour analyses shopping designs to highlight prevalent or trending things, making a difference clients remain educated approximately showcase patterns.

4. AI-Powered Chatbot Help
A shrewdly chatbot guides clients through the stage, helps with watchlist administration, and answers questions through common dialect intelligent.

Why UrSaviour Stands Out:

1. Moral and Legitimate Information Utilize
All information sources are confirmed to guarantee they meet lawful and moral guidelines, emphasizing client believe and straightforwardness.

2. User-Focused Plan
Built with the client in intellect, UrSaviour prioritizes ease of utilize, robotization, and personalized insights.

3. **All-in-One Quick Save Tool**
By combining keen alarms, slant examination, and AI-driven interaction, UrSaviour conveys a total arrangement for more astute, moral basic supply shopping.

2.5 Objectives

1. Automated gathering the Discount Information: Users, including international students, can seamlessly compare the price across three different grocery stores.

2. For regular discount information simulation, a foundational fake dataset of grocery goods will first be manually created and maintained. Python scripts will then read from this foundational dataset to programmatically apply randomized discounts to a selection of around 10 goods on a weekly basis. This regular discounted data will be used to generate new, simulated PDF pamphlets for fictional stores like Store A, B, and C, with the PDF generation process designed to run regularly.

3. Watchlist management: Users can manage a personalized watchlist of grocery items, and the system will notify them when discounts become available. Additionally, the system analyses aggregated watchlist data to identify popular items and purchasing trends, helping administrators and users understand demand patterns.

4. AI assistant with Knowledge System: The AI assistant retrieves relevant product and discount information from the knowledge database to answer user inquiries through a chatbot. If the assistant cannot answer a question, it will refer the user to an appropriate contact person, such as a system coordinator.

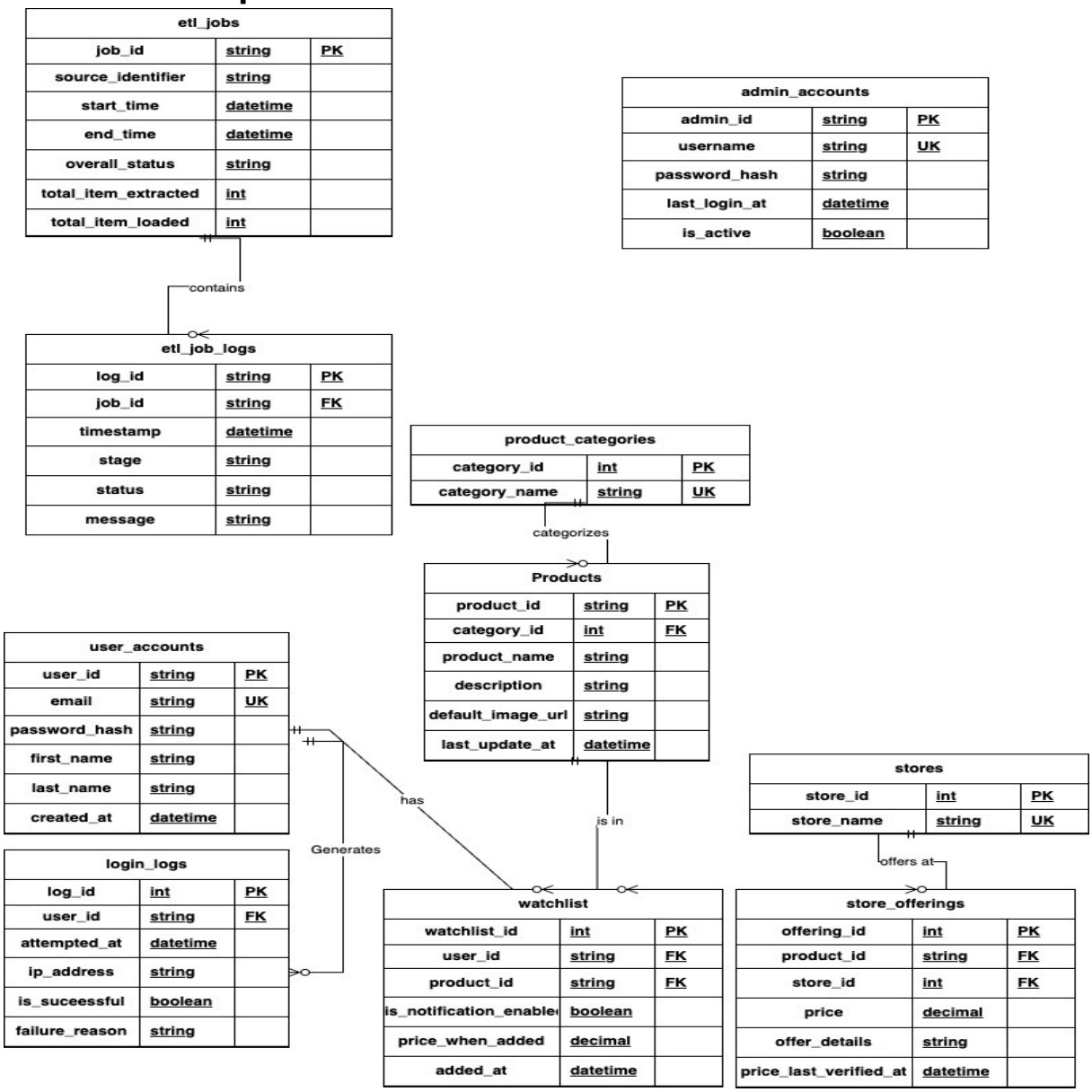
2.6 Scopes

- **Functional Requirements**
  - User Registration (Mio)
  - User Login (Mio)
  - Product Watchlist Management (Aadarsh)
  - PDF scraping and ETL Execution (Fake Dataset) (Justin)
  - Admin Login(Aadarsh)
  - Product Information and Price Display on Website(Justin)
  - AI assistant (Austin)
  - Database Management (Mio)
- **Non-Functional Requirements**
  - Registration (Mio)
  - Login (Mio)
  - Watchlist Submission and Discount Notification (Aadarsh)
  - AI Assistant Response Handling (Austin)
  - Product Information Display (Justin)
  - Important considerations for Dataset Update process (Justin)
  - Security and Privacy (Shared)

2.7 Tools

- WIX: UI/UX Design
- HTML, CSS, JavaScript: Front End
- Python: Back End and Data Engineering
- MySQL: Database
- Rest: API

3. Functional Requirement



To describe the data structures and support these fictional requirements and to provide a broader overview of the database entities involved in the UrSavioour project, example table schemas like ‘userAccounts’ would be shown within relevant subsections.

3.1 User Registration [Mio]

The system should allow users to enter their email address, full name, password, and confirm password to register.

- **Front-end:**
  - Display a sign-up form with fields: Email, Full Name (Last Name and First Name), Password, Confirm Password.
  - Validate inputs:
    - Email format must be correct.
    - Password and Confirm Password must match.
    - Password must meet complexity rules (e.g., ≥8 chars, symbols, numbers).
  - If inputs are valid, send data to the server.
- **Back-end:**
  - Check if the Email already exists in the Users table.
    - If exists: return error message.
    - If not: hash the password using bcrypt and insert the new user data.
  - Record registration date automatically.
- **Database:**

userAccounts		
userId (PK)	VARCHAR(5)	U0001
email	VARCHAR(255)	xxx@xxx.xx
firstName	VARCHAR(50)	Xxxx
lastName	VARCHAR(50)	Xxxx
password	VARCHAR(255)	hashed
createdAt	DATETIME	2025-05-31 16:00:00

3.2 User Login [Mio]

- **Front-end:**
  - Display a login form with fields for email and password.
  - When the user clicks the "Login" button, send the entered credentials to the server.
  - If login fails, show an appropriate error message ("Invalid email or password").
- **Back-end:**
  - Validate the email and password.
  - Check if the email exists in the Users database.
    - If not found, return an error message.
  - If found, verify that the entered password matches the hashed password stored in the database.
    - If matched: allow login, create session/token.
    - If not: return login failure message.
  - Log login attempts for audit/security.
- **Database:**

LoginLogs		
loginId (PK)	VACHER	xxx@xxx.xxx (User's email)
userId (FK)	VARCHAR(5)	U0001
attemptedAt	DATETIME	2025-05-31 16:05:00
ipAddress	VARCHAR(45)	Xxxx
isSuccessful	BOOLEAN	TRUE/FALSE
failureReason	VARCHAR(255)	e.g. User Not Found

3.3 PDF Scraping and ETL Execution (Justin)

- **Front-end (Admin Page):**
  - Automation Configuration and Status Display:
    - Show the present configuration status of the automated email PDF ingestion service (e.g., Specific email account will be monitored, and saved local server PC)
    - Show the present operational status of the email PDF auto-processing script (e.g., Running, Stopped, Error status)

- **Back-end:**
  - Primary ETL Process Triggering Mechanism:
    - ✓ This code ETL process is automatically triggered when a new PDF file updated by email automation script, and it is detected in the designated watch folder (via file system monitoring)
  - ETL Core Logic:
    - ✓ Implements the core ETL (Extract, Transform, Load) functionality
    - ✓ This includes PDF parsing (using defined rules), data validation, transformation, and loading into the database
  - Email Automation Script (External Component/Requirement):
    - ✓ An Independent script (executing on a local server/PC via scheduler) is required.
    - ✓ Its function is to connect to the email account, search for emails, extract PDF attachment, save PDFs to the watch folder (which in turn triggers the Back end ETL), log activities, and mark emails as processed.

- **Database:**
  - **For PDF scarping and ETL process management**

etlJobs		
jobId	VARCHAR(255)	etl_run_20250531_abc1
sourceIdentifier	VARCHAR(255)	weekly_special_A.pdf
startTime	DATETIME	2025-05-31 16:00:00
endTime	DATETIME	2025-05-31 16:05:30
overallStatus	VARCHAR(20)	Completed
totalItemsExtracted	INT	150
totalItemsLoaded	INT	148

  - **Stores detailed logs for each step/stage within an ETL job**

etlJobLogs		
logID	INT	1
jobID	VARCHAR(255)	etl_run_20250531_abc1
timestamp	DATETIME	2025-05-31 16:00:01
stage	VARCHAR(50)	EmailIngestion
status	VARCHAR(20)	Success
message	TEXT	PDF weekly_special_A.pdf saved to watch folder

3.4 Product Information and Price Display on Website (Justin)

- **Front-end(User Website):**
  - Shows product listings with images, names and prices from different groceries
  - Helping users to explore product details, including a comparison of prices from all available stores.
  - Provides filtering, search (e.g., by store, price range), and sorting (e.g., by price, name) functionalities
  - Fetches product data from the back-end API

- **Back-end:**
  - Provides API endpoints for the front-end to query product information
  - Retrieves processed product data from the database
  - Implements search, filtering, and sorting logic
  - Optimizes API responses for fast loading

- **Database:**
  - **Stores general information about unique products**

products		
productId	VARCHAR(5)	P0001
productName	VARCHAR(255)	Organic Bananas
description	TEXT	Bundle of 5 organic bananas
defaultImageUrl	VARCHAR(1024)	/images/products/banana1001.jpg
lastUpdatedAt	DATETIME	2025-05-31 16:05:30

  - **Stores detailed logs for each step/stage within an ETL job**

storeOffering		
offeringId	INT	1
productId	VARCHAR(5)	P0001
storeName	VARCHAR(20)	Justin Groceries
price	DECIMAL(10,2)	3.50
offerDetails	TEXT	{"special": "2 for $6.00"}
priceLastVerifiedAt	DATETIME	2025-05-31 16:05:30

3.5 Watchlist Management (Aadarsh)

- **Front end:**
  - The user interface is built by HTML/CSS/JavaScript

Main features:

- ✓ Add products to the monitoring list.
- ✓ View and manage the products of the monitoring list.
- ✓ Erases the factors from the monitoring list.
- ✓ Received an intuitive warning for items at a decrease price.

UI Components:

- ✓ Lists displayed elements listed by items.
- ✓ Interactive buttons to “Add a monitoring lust” and “delete”.
- ✓ Integrated with local notification systems (e.g. pushing notice or toast messages)

- **Back end:**
  - Back end is developed in Python with FastAPI or Flask to manage API requirements, users sessions and background warning processes.
    API Endpoints (Example Calls):

```
POST /watchlist/add

        Body: { "user_id": 101, "item_id": 25 }

        GET /watchlist/101

        DELETE /watchlist/remove Body: { "user_id": 101, "item_id": 25 }
```

Background Process:

A planned task (using APSChedul or Celery) periodically runs to:

- ✓ To retrieve the latest price of products in watchlists form the systems' internal database (i.e., the StoreOfferings table, which is updated by the ETL process)
- ✓ Compares prices with stored values.
- ✓ If discounted for an article in the list of monitoring, activating notifications or emails.

Email Notification Example:

Subject: Your watchlisted item is now discounted! 🎉
Hi Aadarsh,

The item "Coca Cola 1L" you're watching is now on discount!
Current Price: $1.99 (Previously: $2.50)

Visit UrSavour to grab the deal now.

Authentication:

- Request users to verified by JWT or the notification code to manage the security session

- **Database:**
  - Storage relations with structured by MySQL to manage users, elements, monitoring lists and warnings.

watchlist		
watchlistId (PK)	INT	1
userId (FK)	VARCHAR(5)	U0001
productId (FK)	VARCHAR(5)	P0001
notificationEnable	BOOLEAN	T
addedAt	DATETIME	2025-05-31 16:05:30
added_price	DECIMA(10, 2)	10.40

3.6 AI Assistant (Austin)

- **Front-end:**
  - Provides users with a well-response and friendly interface.
    - On the main page, it appears to be a small chatbot logo.
    - After clicking to it, it will open new page with the appearance being similar to ChatGPT.
  - Captures the queries from the user and display them as well as the response from the chatbot.
    - When the user enters their message, the message will be added to the list
    - Then an HTTP POST request to backend, and the response from AI will be added back to list on return.
  - Refers to human coordinator support for unfound questions.
    - If the chatbot is unable to answer a question from the users, it will suggest and provide the human support information.

- **Back-end:**
  - Serves as a core engine for back-end.
  - Integrates with OpenAI GPT to give a more natural response.
    - To achieve that natural replies, gpt-3.5 will be used via REST API
    - The process contains: Submitted question from user then the back-end will retrieve the data from PDF Scaping, and finally prepare a prompt which enhances the outcome of AI response.
  - Retrieve the data from PDF Scraping and ETL Execution Management.
    - Weekly the PDF of product data will be updated.

- **Database:**
  - Manage the product's data by using automatic ETL pipeline.
  - Become an AI's knowledge base to retrieve the information.
  - Similarly, the database includes **PDF Scraping and ETL Execution Management, Stores detailed logs, Stores general information of unique products, Store offerings.**

ETL_Jobs		
jobId	VARCHAR(255)	etl_run_20250531_abc1
sourceIdentifier	VARCHAR(255)	weekly_special_A.pdf
dtartTime	DATETIME	2025-05-31 16:00:00
endTime	DATETIME	2025-05-31 16:05:30
overallStatus	VARCHAR(20)	Completed

3.7 Admin Login (Aadarsh)

- **Front-end:**
  - A basic, responsive login page open as it were to admin clients.
  - Areas: Email/Username, Password
- **UI Highlights:**
  - Login Button
  - “Forget password” link
  - Mistake message show for invalid accreditaions.
  - On sucessful login, it directly takes to the Admin Dashboard(e.g. to scteen ETL, arrange robotization).
- **Back-end :**
  - Created utilizing Python (FastAPI or Jar).
  - Verifies admin qualifications utilizing secure hashing (e.g., bcrypt).
  - Issues JWT tokens upon effective login for session administration.

API Endpoint:

```
POST /admin/login

Body: { "username": "admin@example.com", "password": "*****" }
```

- ✓ Approves JWT tokens for admin-only courses (e.g., /admin/dashboard).
- ✓ Handles rate-limiting and brute-force assault security (discretionary but perfect).

- **Database:**

Column Name	Type	Example
adminId	VARCHAR(5)	A0001
username	VARCHAR(100)	admin@example.com
passwordHash	VARCHAR(255)	$2b$12$... (bcrypt hash)
lastLoginAt	DATETIME	2025-06-06 10:30:00
isActive	BOOLEAN	TRUE

3.8 Database management (Mio)

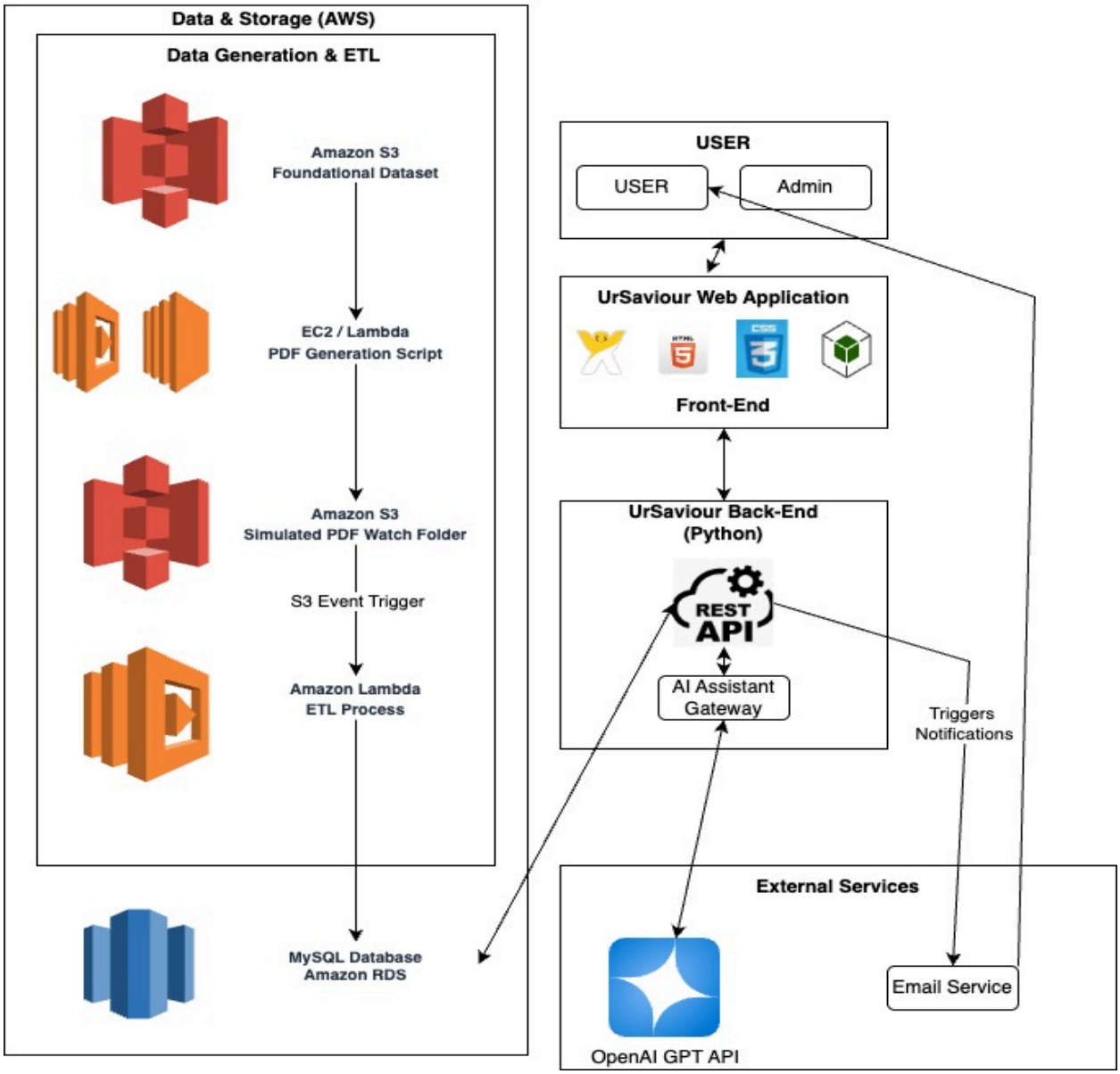
- **Front-end (Admin Dashboard):**
  - **ETL Monitoring**
    - Display a list of recent ETL jobs with status, duration and timestamp.
    - Allow the admin to select a job to view its details.
    - All data is retrieved from the ELTJobs table (no separate) log table.
  - **User Account Management**
    - Display a full list of users from the Users table
    - Provide search and filter options (e.g., by email, registration date, isActive status)
    - Allow the admin to lock or unlock user accounts
  - **Login Logs Monitoring**
    - Display a list of login attempts from the LoginLogs Table.
    - Show details including success/failure, IP address, and timestamp.
    - Enable conditional filtering (e.g., failed attempts in the past 30 minuets).
- **Back-end:**
  - **ETL Monitoring**
    - Get job data from the ETLJobs table.
    - Return list of ETL jobs and if required, return full details of selected job (no separate log table).
  - **User Account Management**
    - Get user list from the Users table.
    - Process account status updates (lock/unlock)
  - **Login Logs Monitoring**
    - Query LoginLogs table and return the result.
    - Support dynamic filtering through query parameters.
- **Database:**

etlJobs		
jobId (PK)	VACHER (10)	JOB001
startedAt (FK)	DATETIME	2025-06-07 10:00:00
endedAt	DATETIME	2025-06-31 16:05:00
Status	VARCHAR(20)	Success/Failed
Duration	FLOAT	3.7 (in mutes)
Message	TEXT	“Discounts loaded”

userAccounts		
userId (PK)	VARCHAR(5)	U0001
email	VARCHAR(255)	xxx@xxx.xx
firstName	VARCHAR(50)	Xxxx
lastName	VARCHAR(50)	Xxxx
password	VARCHAR(255)	hashed
createdAt	DATETIME	2025-05-31 16:00:00

loginlogs Monitoring		
loginId (PK)	VARCHAR(10)	LOG001
userId (FK)	VARCHAR(5)	U001
attemptedAt	DATETIME	2025-05-31 16:00:00
ipAddress	VARCHAR(45)	192.168.1.12
isSuccessful	BOOLEAN	FALSE
failureReason	VARCHAR(255)	Incorrect password

4. Data Pipeline Architecture (Justin)



4.1 Overview

This chapter will elaborate on UrSavior system's data processing architecture. Ensure compliance with the law and copyright and sustainability. This project will not scrap web which occur retailer service policies. Instead, UrSavior will be controlled by internal data-creating and processing workflow.

1. **Foundational Dataset Management:** SSOT(single source of truth) will role of master dataset management in All basic details of products.
2. **Simulated PDF pamphlet Generation:** Base on basic datasets will create automatically weekly discount pamphlets
3. **AWS-Based ETL Workflow**: Pipeline base on AWS event and Extract, Transfer and Load from simulated pamphlet automatically

This architecture approaching will not only make sure ethical data sourcing, but also it will make Whole UrSaviour system test and manage constantly and substantial

4.2 Foundation Dataset Management

The foundational will roll as SSOS for all base product information in UrSaviour ecosystem. This maser is essential for simulated discount pamphlets in later steps, and through it, Ensures data integrity and consistency.

Storage: To ensure durability, version control, and secure access, The foundational dataset will be maintained as a .csv or .xlsx file in a dedicated Amazon S3 Bucket

Schema:

Column	Description	Example
productId	A unique identifier for each product	P0001
productName	The name of the product	Organic Bananas
description	A brief description of the product	Bundle of 5
storeName	The fictional store selling the product	Justin Groceries
basePrice	The standard, non-discounted price	3.50
defaultImageUrl	A link to the product's default image.	/images/p/P0001.jpg

4.3 Simulated PDF Pamphlet Generation

This step of the pipeline is a role for creating realistic, weekly discount pamphlets that serve as the major input for the ETL process. This generation is fully automated for simulate real-world data.

This process is executed by a scheduled Python script running from AWS EC2 or an AWS Lambda function. This script use the ReportLab library, a useful open-source tool, to create PDF documents. ReportLab will provide granular control over the document's structure, To allow the script make draw text, tables, and graphics to construct a visually organized and realistic pamphlet.

This workflow is:

1. The script accesses the foundational dataset from the S3 Bucket.
2. It randomly chooses a subset of products and applies discount logic
3. Using ReportLab, the script generates a new PDF document, creating elements
4. It iterates via the discounted product data, adding each item as a row in the PDF table
5. The final step, complete PDF file(ex, weekly_speical_20200607.pdf) is saved to a separate S3 bucket designated as the ETL "watch folder"

4.4 AWS-based ETL Workflow

The last step of the pipeline is the ETL workflow, that is totally automated and built on a serverless system, event-driven architecture within AWS. This design ensures scalability, and resilience, as the process only runs once new data is present.

The workflow will be triggered by the arrival of a new PDF pamphlet in the designated S3 "watch folder" and proceed as below steps:

1. Trigger (S3 Event Notification): An S3 event Notification is configured on the "watch folder" S3 bucket. The bucket will receive a newly created PDF file, and An event will be created and triggering the next stage in the pipeline.

2. Execution (AWS Lambda): The S3 event triggers a committed AWS Lambda feature. This function includes the Python code reliable for the overall ETL logic, eliminating the need for a continuously running server.

3. Extract: The Lambda function is gathered with information about the new PDF file. It accesses the file straightly from S3 and uses parsing libraries to extract raw data such as product names, and prices, and offer details

4. Transform: The extracted raw data is cleaned, validated, and transformed into a structured format suitable for the database. This includes sanitizing text, converting currency strings to decimal numbers, and standardizing data types.

5. load: Transformed Data will load on Amazon RDS(MySQL). The script will update to table of product and storeOfferings through Insert or Update working then will reflect updated discount information

6. Logging: Through overall processing, the Lambda function will record processing status, status, and errors within the same RDS in etlJobs and etlJobLogs tables. It will provide full monitoring and tracking

5. Non-Functional Requirements

5.1 Registration [Mio]

- Users must create a strong password that meets complexity requirements (e.g., minimum 8 characters, includes numbers and symbols).
- The system must provide a password strength indicator to guide users during registration.
- Passwords must be securely hashed (e.g., using bcrypt or Firebase Auth) and stored in the database.
- The registration process should complete within 2 seconds to ensure good performance.
- All data transmissions must be encrypted using HTTPS (TLS).

5.2 Login [Mio]

- Password verification must use secure hashing (e.g., bcrypt).
- CAPTCHA or a similar "I am human" test should be used to prevent bots.
- The login process should respond within 2 seconds to maintain good usability.
- If users fail to log in multiple times (e.g., 5 attempts), the account should be temporarily locked for security.
- Login sessions must time out after a set period of inactivity to prevent unauthorised access.
- All login attempts, including timestamp, user ID, IP address, and success/failure status, must be logged to support security audits and detect suspicious activities.

5.3 Watchlist Submission and Discount Notification (Aadarsh)

- **Avoiding Duplicates in the Watchlist**
  To keep the watchlist clean and organized, each client will as it were be permitted to include a particular basic supply thing from a specific store once. This makes a difference anticipate disarray and dodges copy passages for the same item-store combination.
- **Smart and Relevant Notifications**
  Clients will get a markdown notice fair once per markdown cycle. This guarantees they are kept educated without being overpowered by rehashed cautions for the same offer.
- **Flexible Notification Settings**
  For way better control, clients can turn notices on or off separately for each thing in their watchlist. This makes the encounter more personalized and helpful.

5.4 AI Assistant Response Handling (Austin)

- User must receive a response from the AI assistant within 2 seconds, and smooth interaction with the chatbot.
- An automatic suggestion to contact human coordinator in case AI is unable to give a solution.
- The assistant should maintain the availability at 99.9 percent uptime.
- If the OpenAI API is down, the user should be informed by AI assistant.

5.5 Product information Display (Justin)

- Goods listings, search results, and detail pages have to load quickly (e.g., within 3 seconds)
- Users has to easily access and find desired product and price information, with a consistent user experience across various devices.
- Goods information and prices showed on the website must accurately reflect the latest processed through ETL.

5.6 Important considerations for Dataset Update Process (Justin)

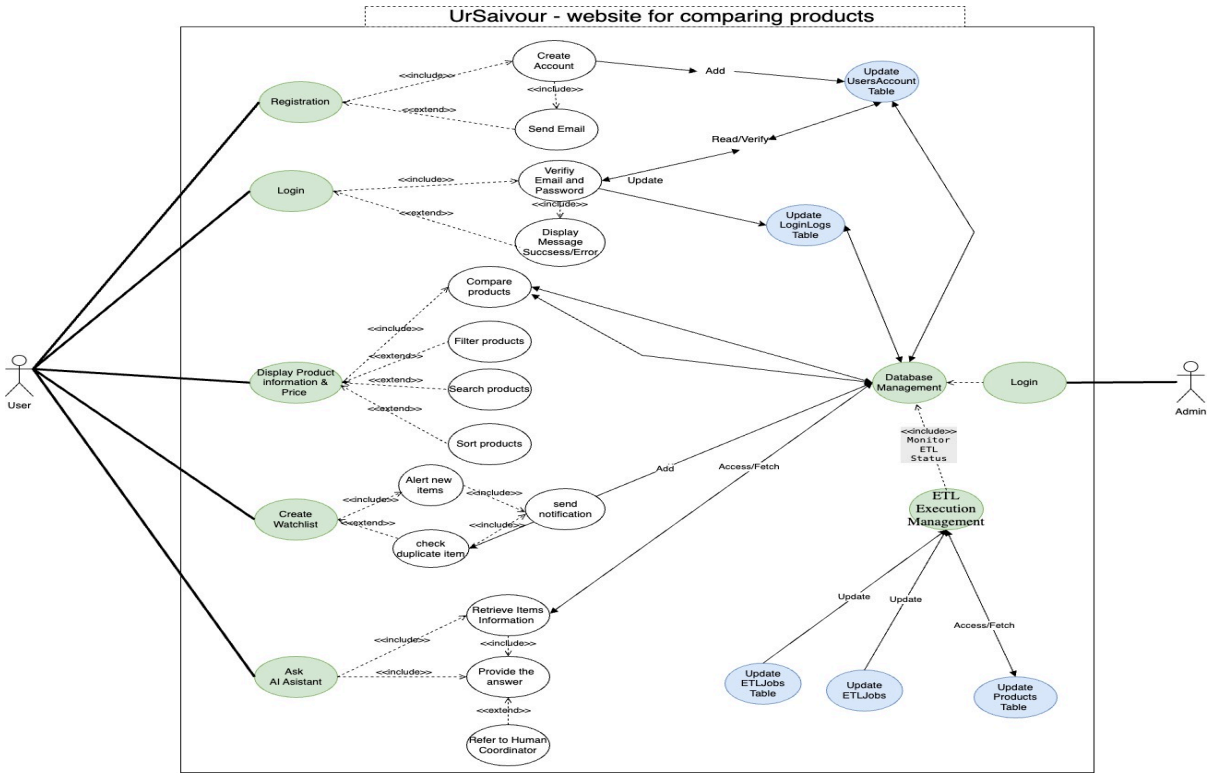
- Generating fake data: ensure realism and diversity
    - Creating fake data with complexity and diversity, it will be similar with real data to cover various test scenarios.
- Guarente consistency: Synchronies between datasets and Pdfs
    - Once foundational fake datasets are updated, ensure corresponding virtual PDF content also matches to prevent distorted test results.
- Maintenance: Regular updates versioning and well documentation
    - Regular updates of fake data, version control, and well documentation of generation rules and assumptions are needed.
- Aware testing limitations: Understanding differences from reality and accuracy benchmarks
    - Accepting that fake data could not replace all real-world variables, and ETL accuracy will be measured against the generated fake data.
- Ethical and Copyright compliance: Strictly Prohibit use of real data
    - Strictly manage the creation of fake data and virtual PDFs to ensure real data (store names, product info, etc.) is prohibited.

6. Use Case Diagram:

The description of UrSaviour and System configuration (Austin)

In our Use Case Diagram, it indicates 6 primary functionalities of the website including Registration, Login, Display Product Information and Price, Create Watchlist, Ask AI Assistant, and Database Management. Following that two types of users can access to the website being User (Customer) and Administrator.

User	Administrator
All users can access our UrSaviour web-based application to search for grocery discounts and make a price comparison between two franchises or chains, focusing more on international students.	The admin is responsible for monitoring and managing backend processes and data within the UrSaviour system.
Key Functions	
Registration: Create a new account in UrSaviour. **Login:** Log in to a registered account. **Create Watchlist:** Add, view and manage favorite items, and receive an alert about discounts. **Ask AI Assistant:** Ask questions about products or discounts via a chatbot and receive answers.	**Database Management:** Monitor, manage other databases, and show the status of automated PDF scraping and ETL (Extract, Transform, Load) processes. An exclusive function is accessed by administrators.



Use case Name:	**Registration (Mio)**
Participating Actor:	User (e.g., international students)
Goal:	To successfully create a new UrSavour account.
Brief Condition:	This use case describes how a new user creates an account in the UrSaviour system by entering their email, full name(first name and lastname), and password. The system verifies the input, checks for duplicates, and stores the user's information securely.
Entry Condition:	Visits through a web browser
Flow of Events: (Basic Flow / Main scenario)	1. The user enters their emall, first Name, last Name, password, and confirmPassword. 2. The UrSavior system checks email format, password meets requirements, and match password and confirmPassword. 3. The user clicks "Register" button after filling the fields. 4. If step 2 confirmations are invalid, the system display an error message. If valid, go to step 5. 5. The UrSavior system sends the registration user data to backend. 6. The UrSavior system checks if the Email already exist in User table. 7. If the Email does not exist, the system hashes the password using bcrypt. 8. The UrSavior system inserts the new user data (userId, email, firstName, lastName, hashedPassword, createdAt) into the Users table. The registration date (createdAt) creates automatically. 9. The system displays a success message (e.g., "Registration successful!") and a new account created.
Alternative Flows:	**AF1: If the email is already registered:** 1. The UrSavior system displays an error message: "This email is already registered." 2. The user is asked to log in instead or use a different email address. **AF2 If password and confirmation do not match:** 1. The UrSavior system displays an error message: "Passwords do not match." 2. The usr re-enters password and tries again. **AF3: If any required field is empty:** 1. The UrSavior system highlights the missing field(s) in red. 2. The user must fill in all required fields before proceeding. **AF4: If the password does not meet complexity requirements (e.g., fewer than 8 characters, no symbol, etc.)** 1. The UrSavior system displays an error message: "Password must be at least 8 characters long and include a number and a symbol." 2. Password field is highlighted. 3. The user edits the password and tries again.
Exit Condition:	After registration successfully, redirect to Login page. OR Receive an error massage if there are any problem with the registration process.
Exceptions:	**E1: Database connection failure:** If the UrSavior system cannot connect to the database, registration cannot proceed. **E2: Unexpected system error:** If an unexpected back-end error occurs (e.g., server timeout), the UrSavior system displays an error message.
Notes and Issues:	**Secure Data Transmission:** All data transmissions during the registration process must be encrypted using HTTPS (TLS) to ensure data privacy and protection. **Password strength indicator:** A password strength bar improves user understanding of password quality.

Use case Name:	**Login (Mio)**
Participating Actor:	User (e.g., international students)
Goal:	Access the system with your Email and Password
Brief Condition:	This use case explains how a registered user logs into UrSaviour using valid credentials. The system checks the entered email and password, verifies them, and grants access to the home page if successful.
Entry Condition:	Visits thorough a web browser
Flow of Events (Basic Flow / Main scenario):	1. The user enters email and password. 2. The user clicks the "Login" button after filling the fields. 3. The system sends the entered Email and Password to the backend. 4. The system checks if the Email exits. 5. If the email does not exist, the system display an error message. (e.g., “This email does not exist.”) If the email exits, go to step 6. 6. The UrSavior system verifies the entered password matches the hashed password stored in the Users database using secure hashing (e.g., bcrypt). 7. The UrSavior system allows login and creates a session/token for the user. 8. The UrSavior system logs the successful login attempt in the LoginLogs table, including loginID, userID, attempedAt, ipAddress, and IsSuccessfull. 9. The system displays the user's personalized UrSavior Home page.
Alternative Flows:	**AF1: If the email does not exist:** 1. The UrSavior system displays an error message: “This email does not exists.” 2. The user is prompted to register or check the email address. **AF2: If the password is incorrect:** 1. The UrSavior system highlights the missing fields and display a message: ”Please fill in all fields.” **AF4: If the account is temporarily locked due to multiple failed login attempts:** 1. If the user fails to login more than 5 times within30 minutes, the UrSavior system locks the account. 2. A message is shown: “Your account is temporarily locked due to multiple failed attempts. Please try again later. 3. The user must wait for a cooldown period (e.g., 1 hour).
Exit Condition:	After login successful, redirect to Home page OR Receive an error message if there are any problem with the login process
Exceptions:	**E1: Database connection error:** If the UrSavior system fails to connect to the database, the login request cannot be processed. The UrSavior system displays: “Login is temporarily unavailable. Please try again later.” **E2: Server timeout or internal server error** If the server takes too long to respond or encounters an unexcepted error, the UrSavior system shows an error message. **E3: Invalid session/token creation** If the UrSavior system fails to generate a valid session or token after login, the user will be redirected. The UrSavior system displays: “Login failed due to technical error. Please try again later.”
Notes and Issues:	**Secure Data Transmission:** All data transmissions during the registration process must be encrypted using HTTPS (TLS) to ensure data privacy and protection. **Password Security:** All passwords must be securely hashed using industry standards (e.g., bycrypt). Plain text passwords should never be stored. **Session Management:** Login sessions must automatically expire after a set period (e.g., 30 minutes) of user inactivity to prevent unauthorised access.

Use case Name:	**User Views Product Information and Prices (Justin)**
Participating Actor:	User
Goal:	To enable the user to find and compare product details and prices from various grocery stores easily and accurately.
Brief Condition:	This use case describes how a user browses, searches, and views detailed information, including prices from different stores, for grocery products on the UrSavior website.
Entry Condition:	• The UrSavior website is online and accessible. • The user has navigated to a product listing or search page. • The database (Products, StoreOfferings) is updated with product and pricing data via the ETL process.
Flow of Events (Basic Flow / Main scenario):	1. The User navigates to the UrSavior product listing page. 2. The UrSavior System fetches product data from the back-end API. 3. The UrSavior System retrieves processed product data from the Products and StoreOfferings databases. 4. The UrSavior System displays product listings with images, names, and prices from different groceries. 5. The User can use filtering (e.g., by store, price range), search (e.g., by product name), and sorting (e.g., by price, name) functionalities. 6. The UrSavior System implements search, filtering, and sorting logic based on user input. 7. The User selects a specific product to view its details. 8. The UrSavior System fetches detailed product information and price comparisons from all available stores via the back-end API. 9. The UrSavior System displays the product details page, including product name, description, image, and price comparison across different stores.
Alternative Flows:	**AF1: No Products Found for Search/Filter:** 1. (After step 4 or 9) If a search or filter yields no results, the UrSavior System displays a message indicating "No products found." 2. The User can modify their search/filter criteria. **AF2: Slow Loading of Product Information:** 1. (During step 4 or 9) If product listings or detail pages load slowly (exceeding 3 seconds), the User may experience frustration. **2.** The UrSavior System's back-end should optimize API responses for faster loading.
Exit Condition:	• The User has successfully viewed product information and prices. • Goods listings, search results, and detail pages load quickly (within 3 seconds). • The displayed product information and prices accurately reflect the latest data processed through ETL. • The user experience is consistent across various devices.
Exceptions:	**E1: API/Backend Error:** If the backend API or server encounters an error while fetching data, the UrSavior System will display an error message to the user, and **product information may not be displayed correctly.** **E2:** Database Unavailability: If the database (Products, StoreOfferings) is unreachable, no product information can be retrieved, leading to an error display.
Notes and Issues:	The backend provides API endpoints for the front-end to query product information

Use case Name:	**User Manages Watchlist and Receives Discount Notifications (Aadarsh)**
Participating Actor:	User
Goal:	To allow users to track specific grocery items and receive timely, personalized alerts when discounts become available.
Brief Condition:	This use case describes how a user can add, view, and manage items in their personalized watchlist and receive automated notifications when discounts become available for those items.
Entry Condition:	• The user is logged into their UrSavior account. • Product information is available in the UrSavior database (StoreOfferings). • The UrSavior System's background task for watchlist monitoring is running.
Flow of Events (Basic Flow / Main scenario):	**Part A: Adding/Managing Watchlist Items** 1. The User browses product listings or searches for an item. 2. The User selects a product they wish to add to their watchlist. 3. The User clicks the "Add to Watchlist" button. 4. The UrSavior System sends a request to the backend API (POST /watchlist/add) with the user_id and item_id. 5. The UrSavior System checks for duplicates in the Watchlist table to ensure the user adds a specific grocery item from a specific store only once. 6. If no duplicate, the UrSavior System inserts the UserID, ProductID, NotificationEnable (defaulted to True), and AddedAt into the Watchlist table. 7. The UrSavior System displays a confirmation message (e.g., "Item added to watchlist"). 8. The User can view and manage their watchlist by navigating to the "Watchlist" section. 9. The UrSavior System sends a request to the backend API (GET /watchlist/{user_id}) to retrieve the user's watchlist items. 10. The UrSavior System displays a list of elements listed by items with interactive buttons. 11. The User can select an item from their watchlist and click the "Delete" button. 12. The UrSavior System sends a request to the backend API (DELETE /watchlist/remove) with user_id and item_id. 13. The UrSavior System removes the item from the Watchlist table. 14. The UrSavior System displays a confirmation message (e.g., "Item removed from watchlist"). 15. The User can toggle notification settings on or off individually for each item in their watchlist.
	Part B: Receiving Discount Notifications 8. A planned background task (using APSChedul or Celery) periodically runs. 9. The background task retrieves the latest price of products in watchlists from the system's internal database (i.e., the StoreOfferings table, which is updated by the ETL process). 10. The background task compares the latest prices with previously stored values for watchlisted items. 11. If a discount is detected for an article in the watchlist, and notifications are enabled for that item, the UrSavior System activates a notification or email. 12. The UrSavior System sends a personalized email notification to the user (e.g., "Your watchlisted item is now discounted!"). The notification is sent once per discount cycle. 13. The UrSavior System can also deliver local push notifications or toast messages.
Alternative Flows:	**AF1: Item Already in Watchlist (Duplicate Attempt):** (After step 5 in Part A) If the user attempts to add an item that is already in their watchlist, the UrSavior System displays a message (e.g., "Item already in watchlist") and prevents adding the duplicate. **AF2: No Discount Detected:** (During Part B, after step 3) If no discount is detected for any watchlisted item, the background task completes without sending notifications. **AF3: Notification Disabled by User:** (During Part B, after step 4) If a discount is detected but the user has turned off notifications for that specific item, no notification is sent for that item.
Exit Condition:	• The User's watchlist is accurately managed in the Watchlist table. • Users receive smart and relevant discount notifications once per discount cycle for enabled items. • Aggregated watchlist data can be analyzed to identify popular items and purchasing trends.
Exceptions:	**E1: Database Write/Read Error:** If there is an issue with writing to or reading from the Watchlist table, the user's action (add/delete) may fail, or notifications may not be sent correctly. **E2: Email Service Outage:** If the email notification service is down, emails will not be sent, though local notifications may still function.
Notes and Issues:	• User authentication for API calls is handled via JWT or notification code to manage security sessions. • The backend is developed in Python with FastAPI or Flask to manage API requirements, user sessions, and background warning processes.

Use case Name:	**Interaction with AI Assistant (Austin)**
Participating Actor:	User
Goal:	To boost the user's experience to another level with AI-powered chatbot answering grocery questions from user quickly, accurately, human-liked answer, and help user to make more right decision in purchasing any item.
Bried Condition:	The Use Case Diagram shows all the way AI-powered chatbot functioning when user give any question of grocery items or discount items and receive precisely answer in short time.
Entry Condition:	• The UrSaviour website-based application is running successfully. • On the main page, the chatbot logo is displayed on the bottom right conner. • The backend of our application is connected to OpenAI GPT API. • The Products and StoreOfferings databases are integrated with up-to-date product and discount information from the ETL pipeline which plays a role as a knowledge's foundation for AI. • The uptime of AI assistant maintains 99.9% consistently.
Flow of Events (Basic Flow / Main scenario):	1. On the main page of our UrSaviour application, the user clicks on the chatbot logo on the bottom right corner. 2. The application's system brings the user to a new page with the typing bar in the middle, similar to ChatGPT. 3. The user enters their question to the input typing bar. 4. The user submits the question by clicking to the send button. 5. The application's system captures and displays the query from user on the interface. 6. The application system sends an HTTP POST request with the user's query to the backend. 7. The backend fetches the query items information from our databases such as Products database and StoreOfferings database. 8. The backend prepares a prompt to enhance the better AI response for the OpenAI GPT. 9. OpenAI GPT uses gpt-3.5 via REST API for huma-liked response. 10. The backend sends the prepared prompt to the OpenAI GPT. 11. The AI sends back with the final response to the backend. 12. The backend sends the response to the interface. 13. The application system displays the response from AI in the chat windows. 14. The user receives the final responses from the AI assistant within 2 seconds.
Alternative Flows:	**AF1: AI Assistant Unable to Answer (Referral to Human Coordinator):** 1. After step 9, If the AI assistant is unable to provide the answer, the backend will identify this issue. 2. The backend is responsible for sending a message to contact the human coordinator for further support on the interface. 3. The application system displays the message with the contact information on the interface. **AF2: OpenAI API Downtime:** 1. During step 8, If the OpenAI API is down, the backend fails to receive a response. 2. The backend sends a message to the interface. 3. The interface shows the message "AI is unavailable".
Exit Condition:	• The user has received an answer for the question about products and discounts information. • The AI assistant responses within 2 seconds with smooth interaction. • The AI assistant is ensured to be available with 99.9% uptime.
Exceptions:	**E1: Network Connection Loss:** An message is showed to the user for these related issue like No internet connection, Fail communication with the backend, and Fail to response from AI. **E2: Backend Server Error:** Unhandling error relates to the backend server which happens during the process of AI, then an error massage is displayed to the user.
Notes and Issues:	• The backend acts as the core engine for the AI's integration. • ETL pipeline manages the AI's foundation knowledge automatically.

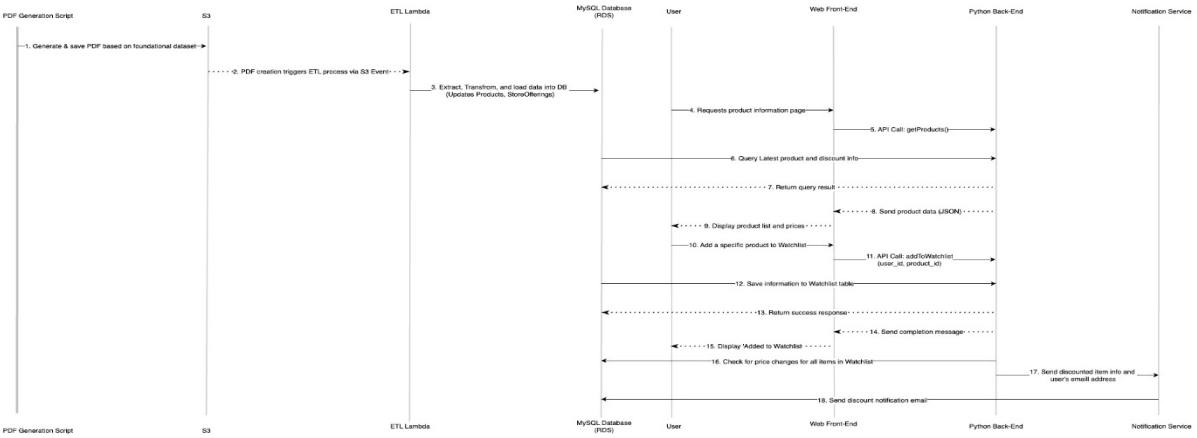
Use case Name:	**PDF Scraping and ETL Execution Management (Justin)**
Participating Actor:	System
Goal:	To automatically process simulated grocery discount data from PDFs and load it into the UrSavior database, ensuring accurate and up-to-date product and pricing information.
Brief Condition:	This use case describes how the UrSaviour System's backend ETL pipeline automatically processes data.
Entry Condition:	• An external email automation script is running and monitoring a designated email account for new PDF attachments. • A "watch folder" on the local server is configured to receive saved PDFs. • The foundational fake dataset of grocery items is manually created and maintained. • Python scripts are configured to apply randomized discounts and generate new simulated PDF pamphlets weekly.
Flow of Events (Basic Flow / Main scenario):	1. Weekly, Python scripts process the foundational fake dataset, applying randomized discounts to selected items. 2. Based on this discounted data, new simulated PDF pamphlets for fictional stores (Store A, B, and C) are generated. 3. The external email automation script connects to the designated email account. 4. The email automation script searches for emails containing the new PDF attachments. 5. The email automation script extracts the PDF attachments. 6. The email automation script saves the PDFs to the designated watch folder on the local server. 7. The email automation script logs its activities and marks the emails as processed. 8. The UrSavior System's primary ETL process is automatically triggered upon detection of a new PDF file in the watch folder (via file system monitoring). 9. The ETL process logs the start of the job in the ETLJobs table (JobID, SourceIdentifier, StartTime). 10. The ETL process performs PDF parsing based on defined rules to extract product information and prices. 11. The ETL process performs data validation and transformation on the extracted data. 12. The ETL process loads the validated and transformed data into the Products and StoreOfferings tables in the database. 13. The ETL process logs detailed steps/stages within the ETL job in the ETLJobLogs table (LogID, JobID, Timestamp, Stage, Status, Message). 14. The ETL process updates the ETLJobs table with the EndTime, OverallStatus (e.g., "Completed"), TotalItemsExtracted, and TotalItemsLoaded).
Alternative Flows:	**AF1: PDF Parsing Error:** 1. (During step 10) If the PDF parsing encounters errors (e.g., malformed PDF), the ETL process logs the error in ETLJobLogs and attempts to skip problematic entries or abort the job. 2. The OverallStatus in ETLJobs might be set to "Completed with Warnings" or "Failed." 3. The Admin is alerted to review the logs. **AF2: Data Validation Failure:** 1. **(**During step 11) If extracted data fails validation (e.g., missing price, invalid product name), the ETL process logs the specific validation failure in ETLJobLogs. 2. The invalid data is either rejected, corrected if possible, or loaded with flags for review. 3. TotalItemsLoaded will be less than TotalItemsExtracted. **AF3: Database Load Failure:** 1. (During step 12) If there are issues loading data into the database, the ETL process logs the error and might attempt retries. 2. The OverallStatus might be "Failed," and the Admin is alerted. **AF4: Email Automation Script Error:** 5. (During steps 3-7) If the external email automation script encounters an error (e.g., cannot connect to email, file system error), it logs its own error and potentially retries. The ETL process will not be triggered. **6.** The Admin may need to manually intervene to resolve the email script issue.
Exit Condition:	• The UrSavior database is updated with the latest product and discount information from the simulated PDFs. • ETLJobs and ETLJobLogs tables contain detailed records of the ETL process. • The Admin page reflects the current status of the automation. • The dataset update process ensures realism and diversity in fake data, consistency between datasets and PDFs, and is regularly updated with versioning and documentation.
Exceptions:	**E1: Watch Folder Unreachable:** If the watch folder becomes inaccessible, the ETL process will fail to trigger, and an error will be logged by the monitoring system. **E2: Database Connection Loss:** If the database connection is lost during ETL, the process will halt, and errors will be logged, requiring manual intervention.

Notes and Issues:	• Strictly managed creation of fake data and virtual PDFs to ensure real data (store names, product info, etc.) is prohibited. • ETL accuracy will be measured against the generated fake data.
Use case Name:	**Database Management (MIO)**
Participating Actor:	Admin
Goal:	To securely monitor and manage UrSavior system's data operations, including ETL processes, user accounts, and login activities through a centralised dashboard.
Brief Condition:	This use case describes how an Admin uses the database management interface to monitor ELT job history, user account status, and login activity logs, ensuring to keep data accurate and to show how the system is working in the UrSavior system.
Entry Condition:	The Admin is successfully logged into the Admin page. The UrSavior system is connected to its core databases (e.g., Users, LoginLogs, ETLJobs)
Flow of Events (Basic Flow / Main scenario):	1. The Admin logs in and navigates to the Database Management Dashboard. 2. The system displays a summary of key metrics (e.g., Total registered users, Summary of recent ETL job status, Number of failed/successful logins in the past 24 hours) 3. ETL Process Monitoring a. The Admin clicks the ETL Monitoring tab b. The system queries the ETLJobs table and displays recent job runs, with status, duration, and timestamp. c. The Admin selects a job to view detailed logs from the ETLJobLogs table including any warnings or errors. 4. User Account Management a. The Admin accesses the User Management tab. b. The system fetches a list of user accounts from the Users table. c. The Admin can: ▪ Search/filter users (e.g., by email, registration date, or isActive) ▪ Lock or unlock accounts (e.g., after repeated failed login attemps) ▪ View user roles or account creation time 5. Login Logs Monitoring a. The Admin accesses the Login Logs tab. b. The system queries the LoginLogs table to display all attempts including userId, loginTimestamp, ipAddress, isSuccessful c. The Admin can filter logs (e.g., failed logins within last 30 minutes) to detect suspicious activities. 6. Product Table Management a. The Admin access the Product Management tab in the dashboard. b. The UrSavio system gets product data from Products table. c. The admin can: ▪ Search/filter items ▪ Edit product details ▪ Delete outdate or incorrect entires ▪ Manually add new items if needed d. The system updates the Products table confirms the change with a success message.
Alternative Flows:	**AF1: ETL Script Error Handling** If the UrSavior system detects an ETL failure (e.g., though scheduled background task monitoring), an alert is sent to the Admin. The Admin reviews ETL job for error traces. The Admin restarts or patches the ETL job from the dashboard. **AF2: Unauthorized Access Atempt** If an abnormal number of failed logins is detected from a single IP, the Admin is alerted. The Admin may choose to lock the account or blacklist the IP temporarily thorough the dashboard.
Exit Condition:	The Admin has successfully reviewed ETL job statuses, user account data, and login logs using the database management dashboard. Any necessary manual interventions (e.g., resolving script errors or user issues) have been logged or completed. The system remains stable, with data integrity and visibility of system activities ensured.
Exceptions:	**E1: Database Connectivity Failure:** If the system fails to connect to the database (e.g, MySQL server is down) the dash board will display an error message: "Database connection failed. Please try again later." **E2: API Request Timeout:** If the dashboard fails to fetch ETL logs or data due to a slow or timed-out API response, an error message is displayed: "Data could not be loaded. Please refresh the page or check your connection." **E3: Unexpected Script Errors:** If the dashboard encounters an internal script error (e.g., logic bug or rendering issue), an error message is displayed: "An unexpected error occurred." **E4: Product Table Write Failure:** If the UrSavior system encounters a database error while updating the Products table (e.g., due to invalid input or connection issue), an error massage is displayed: "Failed to update product information. Please try it again."
Notes and Issues:	• **Role-Based Access Control:** Only authorized adnin users can access the database management dashboard. • **Frontend and backend integration thorough REST API:** The dashboard fetches metrics and logs by calling backed API endpoints. These APIs must be optimized for performance and security.

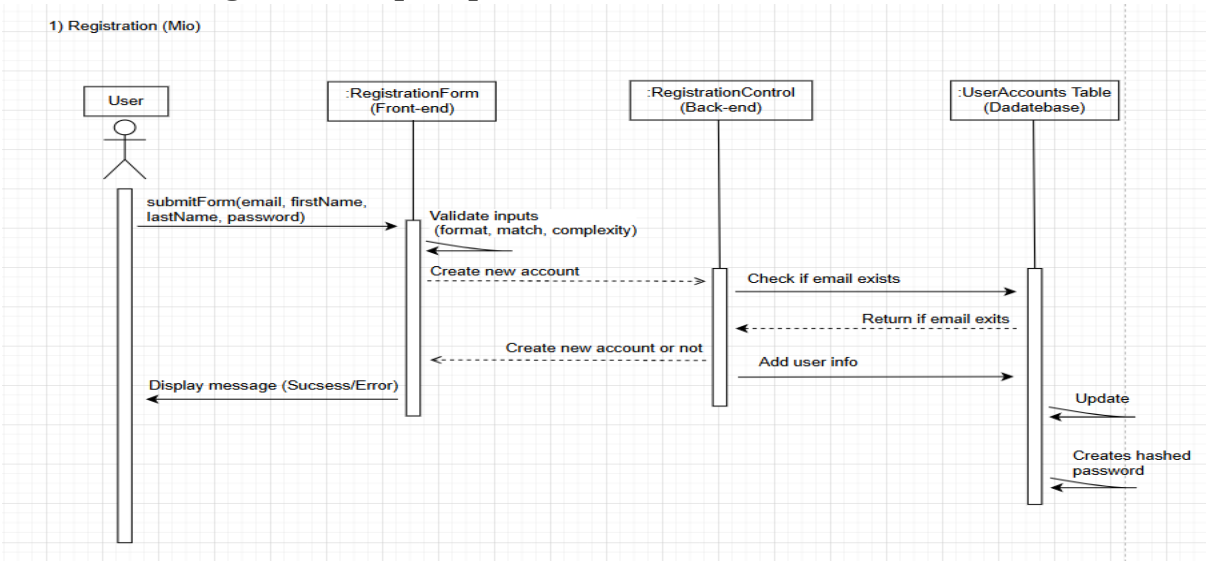
	• **Real-time log updates are not implemented:** Log data is retrieved only when the Admin accesses the page or refreshes it. • **Manual product editing by admins:** should be logged for audit purposes. • **To prevent invalid entries:** The UrSavior system should prevent invalid entries (e.g., negative number for price)
Use case Name:	**Admin Login (Aadarsh)**
Participating Actor:	Admin user
Goal:	To confirm the admin client safely and give get to to the admin dashboard and functionalities.
Brief Condition:	This utilize case depicts the method where an admin logs into the UrSaviour stage utilizing substantial qualifications. Upon fruitful login, a session is made utilizing JWT (JSON Web Token), empowering get to to backend administrations and the admin interface.
Entry Condition:	• The admin client includes a substantial account made within the framework. • The login page is available by means of the front-end interface. • The backend confirmation benefit is running and associated to the database.
Flow of Events (Basic Flow / Main scenario):	1. The framework shows the login shape asking email/username and password. 2. The admin inputs substantial qualifications and clicks “Login”. 3. The frontend sends a POST ask to the verification API endpoint(/api/admin/login). 4. The backend confirms the accreditations against put away hashed passwords with in the database. 5. In the event that substantial, the framework: 6. Produces a JWT token. 7. Logs the login timestamp and IP. 8. Sends the token and victory reaction back to the frontend. 9. The frontend stores the token(e.g. in session/local capacity). 10. The admin is diverted to the dashboard. 11. Admin can presently get to the secured routes/features until session lapes or logout to the database.
Alternative Flows:	• A1- Invalid Credentials: In case the admin enters off base qualifications, the framework shows an blunder message “Invalid username or password” and does not create a token • A2- Server Error: On the off chance that the backend is down or the database association falls flat, the framework shows a message expressing, “Server is incidentally inaccessible. It would be ideal if you attempt once more later.” • A3- Account Locked: If there is multiple failed found then the accounts get locked temporarily and admin gets notified and it need a manual unlock.
Exit Condition:	Upon effective login, the admin is diverted to the dashboard with a secure session started through a substantial JWT token for future activities.
Exceptions:	E1: Database Connection Failure: If the system cannot connect to the database during the authentication process, the login cannot proceed, and a server error is shown to the user> E2: Unexpected System Error: If an unexpected back-end error occurs (e.g., server timeout, JWT generation failure), the system displays a generic error message.
Notes and Issues:	Password Security: All admin passwords must be securely hashed using industry-standard algorithms like bcrypt. Plain text passwords must never be stored. Secure Data Transmission: All data transmitted during the login process must be encrypted using HTTPS (TLS) to ensure data privacy and protection from man-in-the-middle attacks. Session Management: The JWT token must have a reasonably short expiration time to enhance security. The system should also support a secure logout process that invalidates the token if possible. Rate Limiting: To further mitigate brute-force attacks, the login endpoint should have rate-limiting implemented.

7. Sequence Diagram

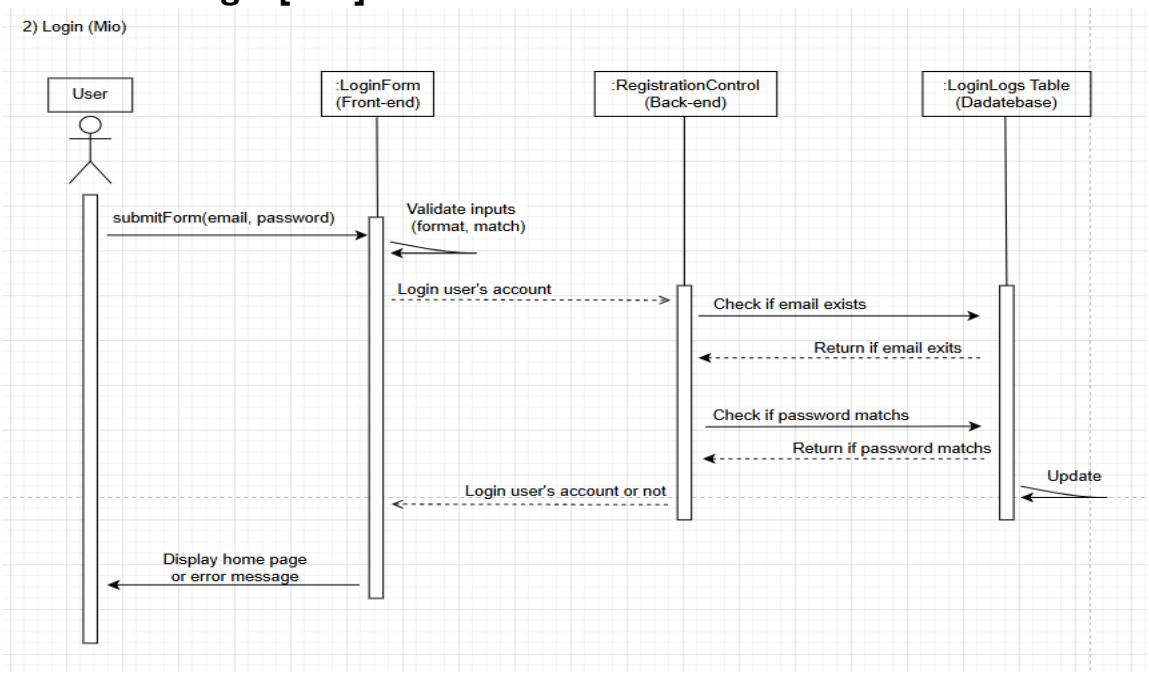
7.1 Overall



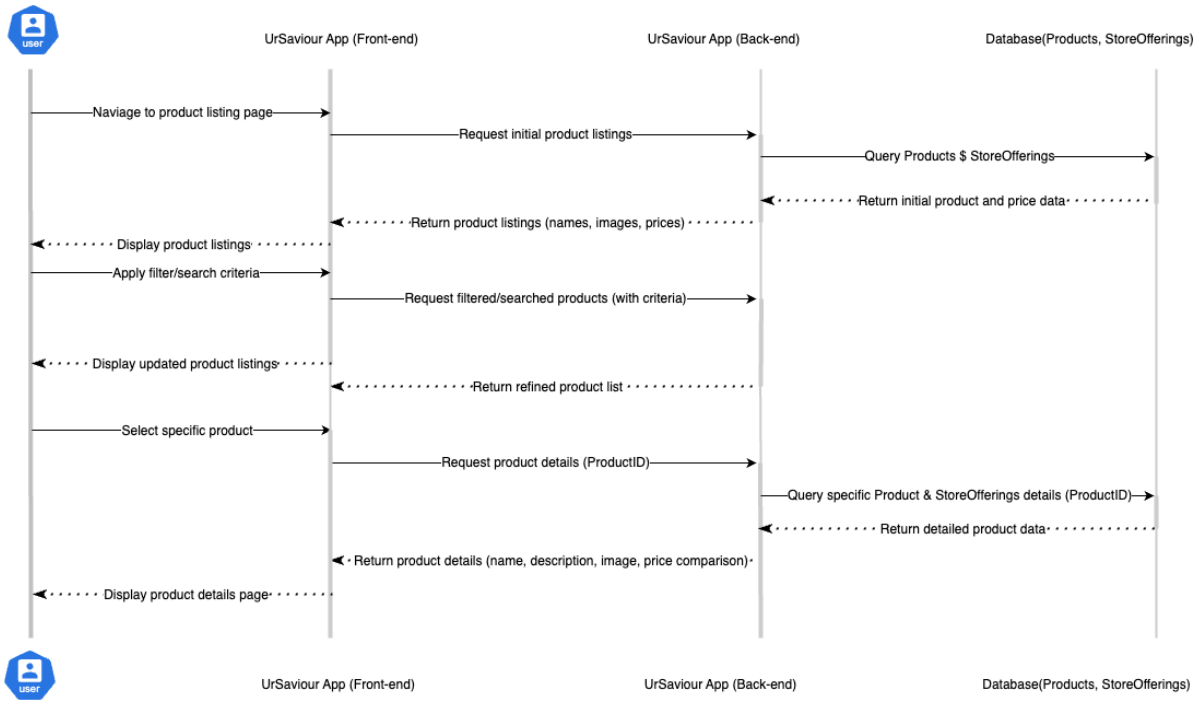
7.2 User Registration [Mio]



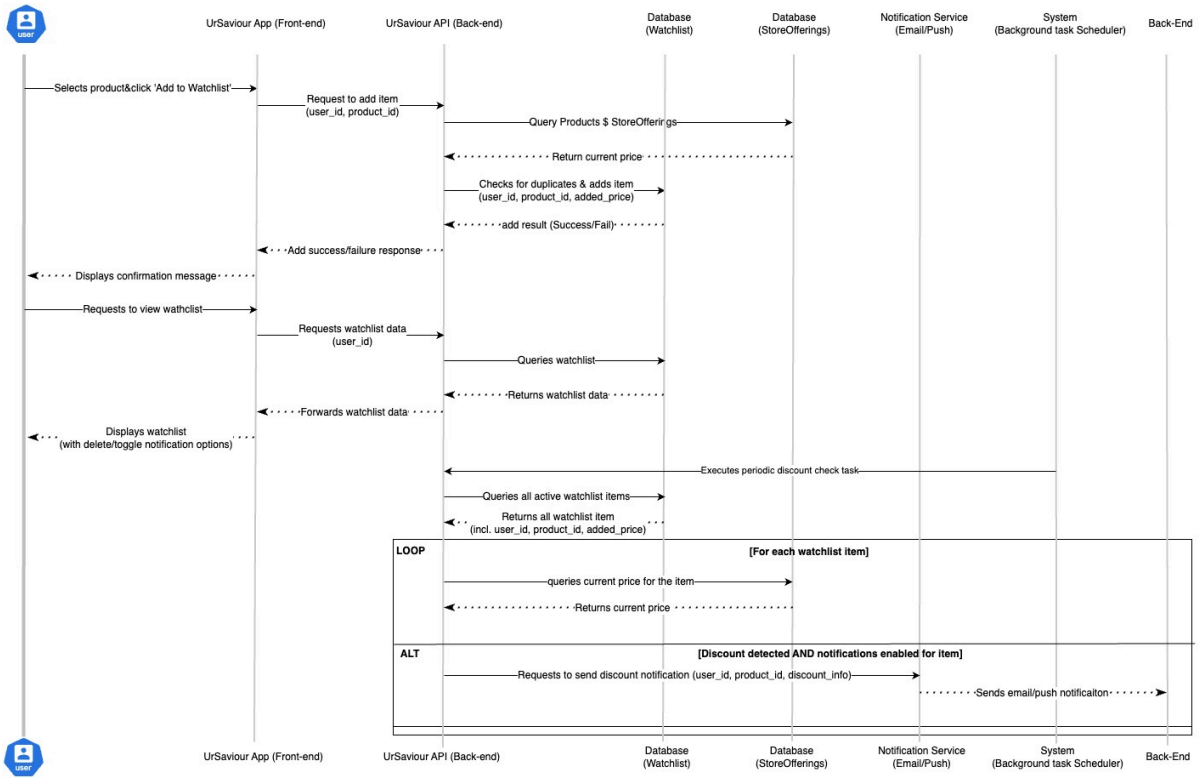
7.3 User Login [Mio]



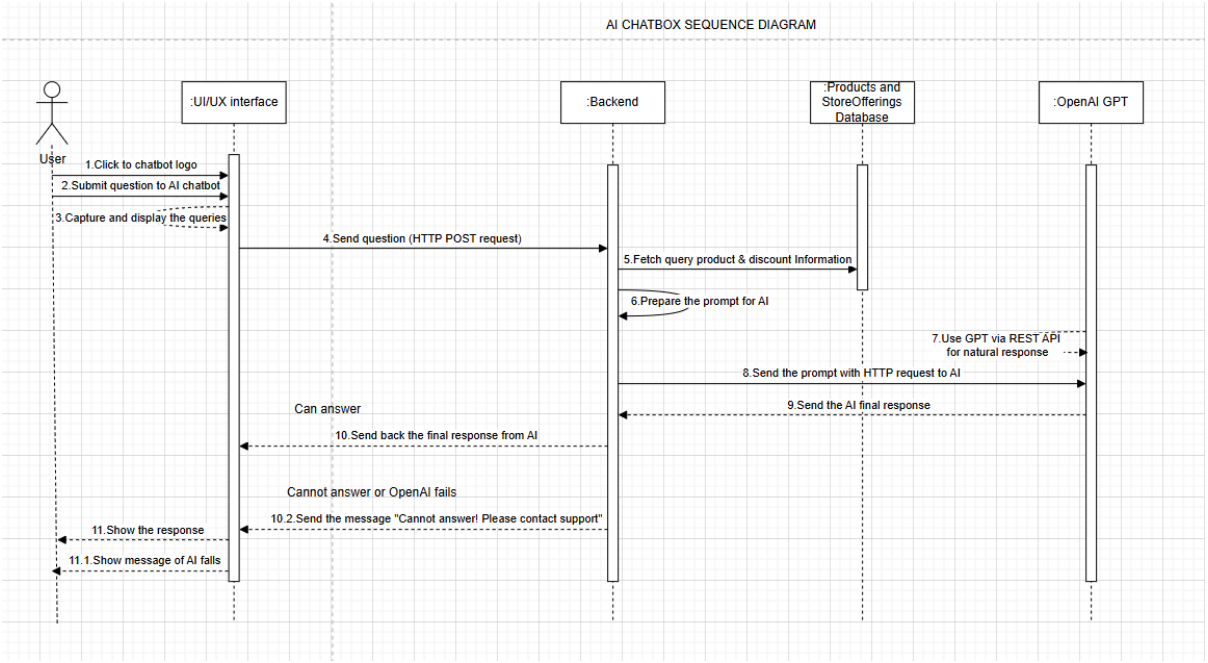
7.4 User views Product Information and prices (Justin)



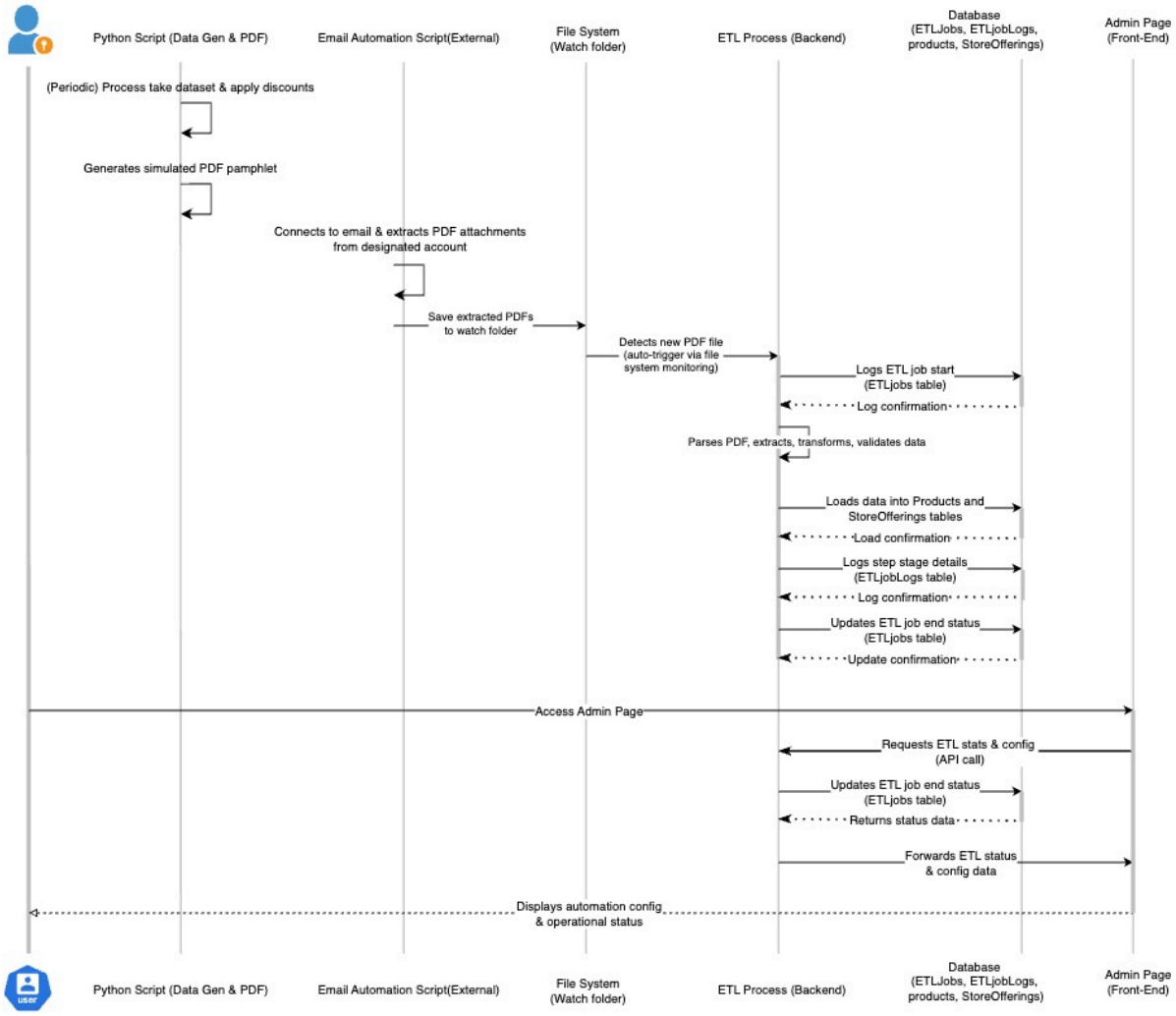
7.5 Watchlist Management (Aadarsh)



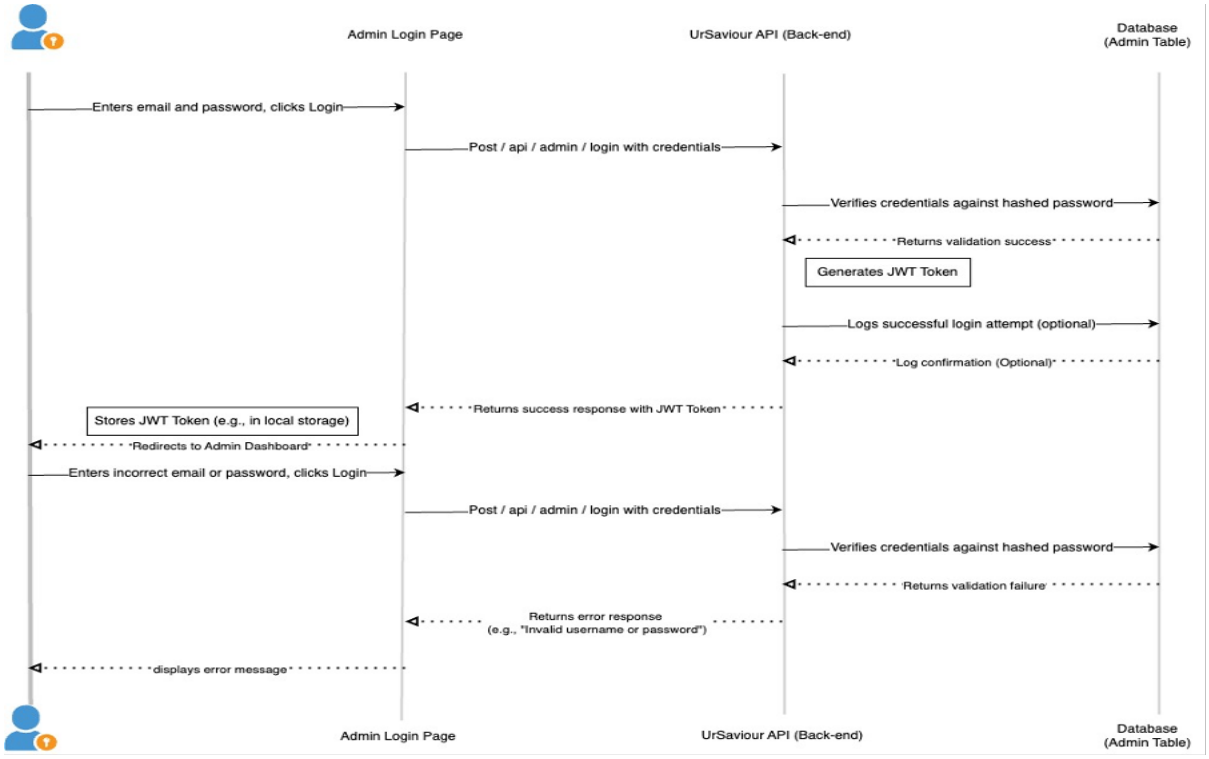
7.6 AI Assistant (Austin)



7.7 PDF Scraping and ETL Execution Management (Justin)

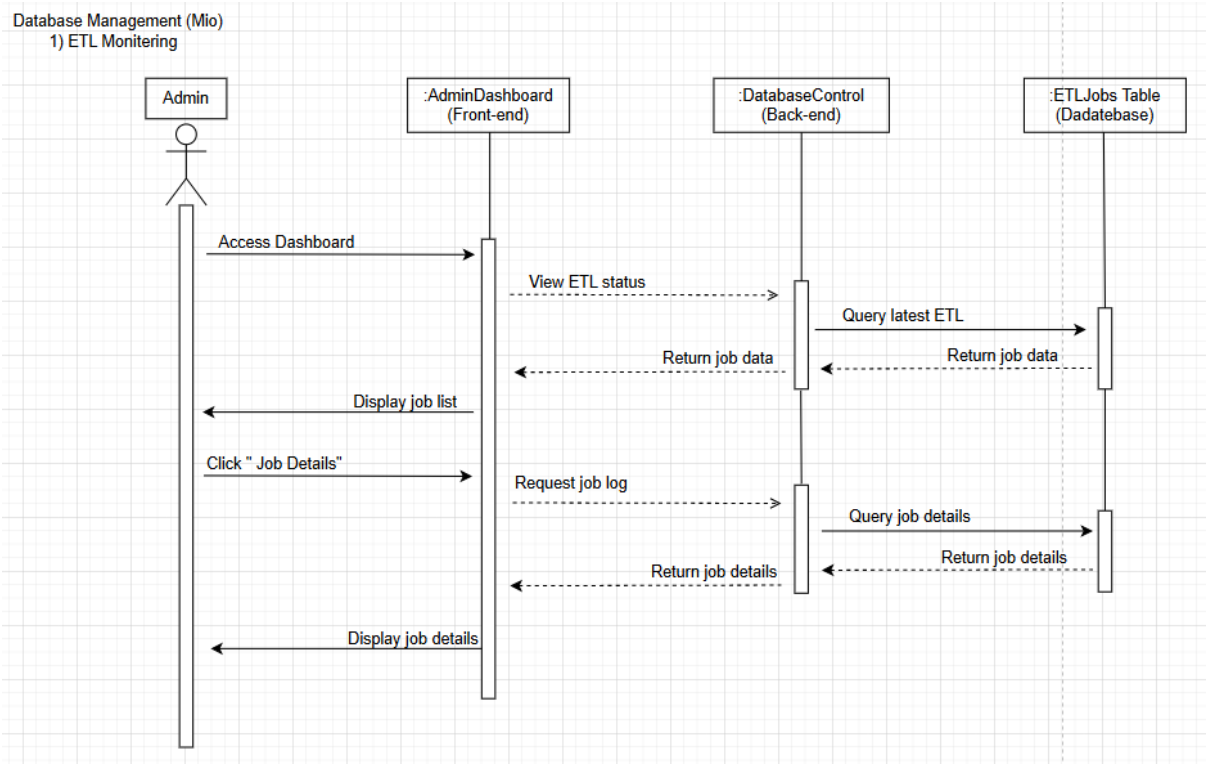


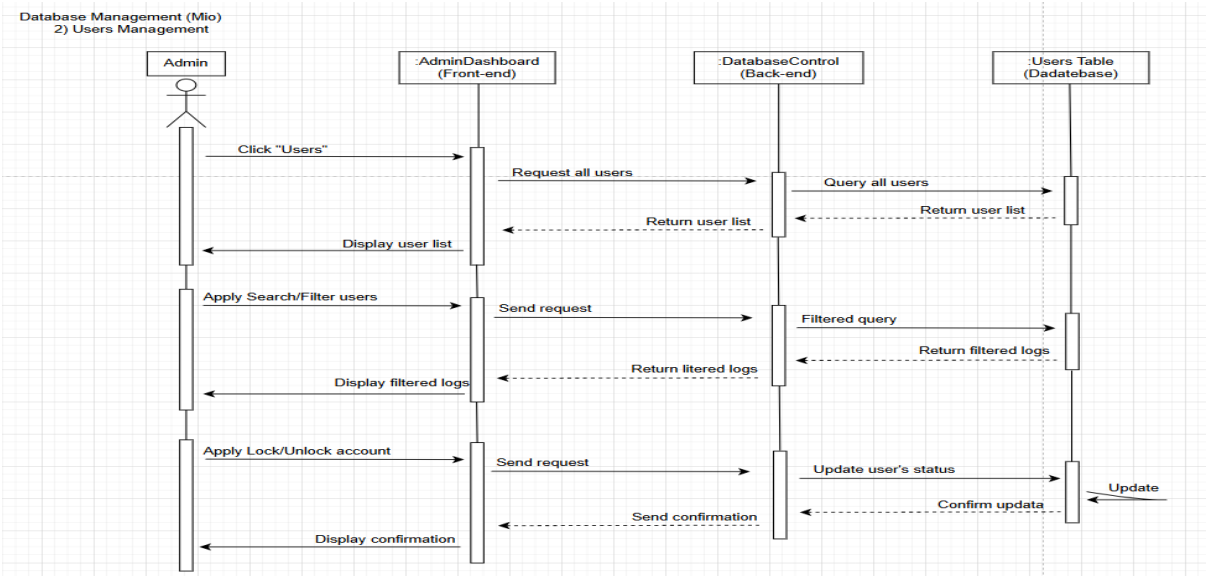
7.8 Admin

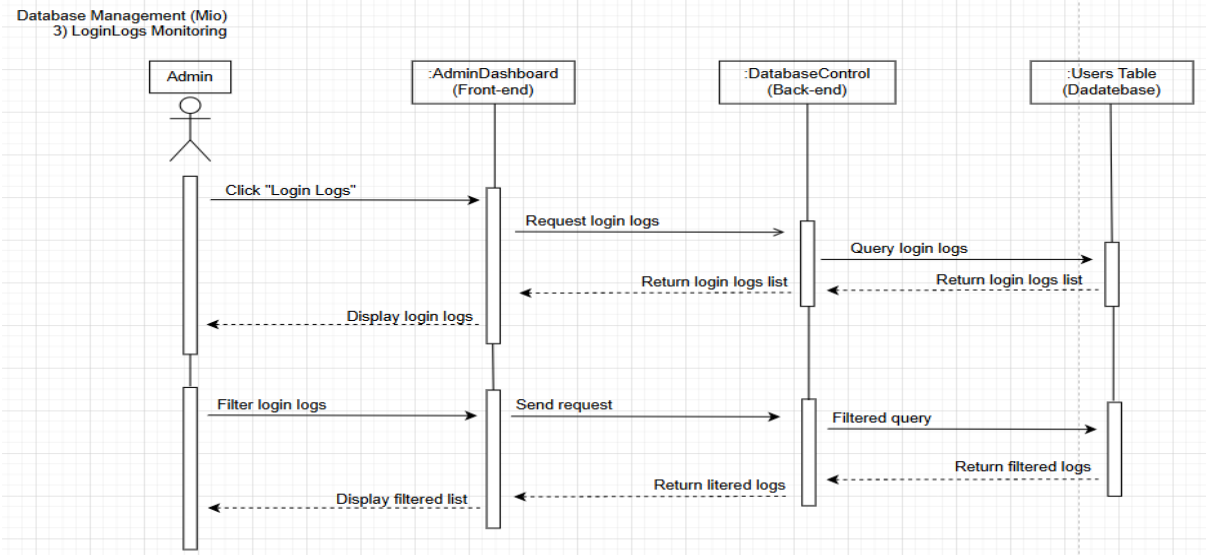


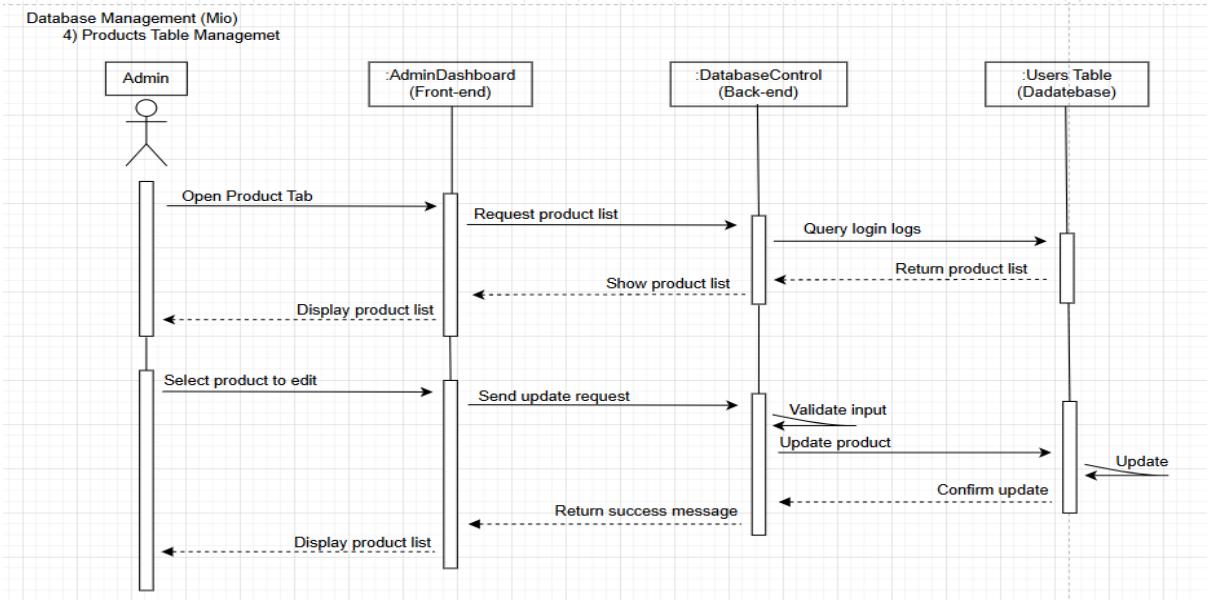
Login (Aadarsh)

7.9 Database Management (Mio)




Database Management (Mio)
2) Users Management
Admin
:AdminDashboard (Front-end)
:DatabaseControl (Back-end)
:Users Table (Dadatebase)
Click "Users"
Request all users
Query all users
Return user list
Return user list
Display user list
Apply Search/Filter users
Send request
Filtered query
Return filtered logs
Return litered logs
Display filtered logs
Apply Lock/Unlock account
Send request
Update user's status
Update
Confirm updata
Send confirmation
Display confirmation


Database Management (Mio)
3) LoginLogs Monitoring
Admin
:AdminDashboard (Front-end)
:DatabaseControl (Back-end)
:Users Table (Dadatebase)
Click "Login Logs"
Request login logs
Query login logs
Return login logs list
Return login logs list
Display login logs
Filter login logs
Send request
Filtered query
Return filtered logs
Return litered logs
Display filtered list


Database Management (Mio)
4) Products Table Managemet
Admin
:AdminDashboard (Front-end)
:DatabaseControl (Back-end)
:Users Table (Dadatebase)
Open Product Tab
Request product list
Query login logs
Return product list
Show product list
Display product list
Select product to edit
Send update request
Validate input
Update product
Update
Confirm update
Return success message
Display product list

8. Resource Management

8.1 Introduction to Resource Management (Austin)

Resource management dignifies the transparency during the project is implemented. Due to its benefits, it helps to better minimize both idle time and overutilization of resources to attain all the requirements. By utilizing the workforce in our group to assign the project's tasks, our workload can be performed smoothly to achieve our objectives. Regarding to financial management, it mitigates the financial risk and stays in budget. The available allocated resources with a careful plan is used effectively to ensure the accomplishment of the outcome. In terms of technological, material, supplementing resources, all sources are mainly chosen carefully to enhance the consistency in the project. The timeline for our project is expected to be completed before the NIT3004, but before that the final project proposal and presentation by June 17th 2025. [9] [10]

8.2 Identify Required Resources (Justin)

- **Human Resources:**
  - **Team members:** Team UrSaviour project is consists of four members, our responsible for function and non-functional aspects of the "UrSaviour" web-based application:
  - **Justin Lee (S8115784)**

Responsibility
✓ PDF Scraping and ETL Execution
✓ Product Information and Price Display on Website

  - **Mio Mizutani (S8107356)**

Responsibility
✓ User Registers for UrSaviour System
✓ User Login
✓ Database management

  - **Austin Le (S8106392)**

Responsibility
✓ The AI assistant
✓ UI Desgin

  - **Aadarsh Thapa (S4679035)**

Responsibility
✓ Watchlist Management
✓ Admin Login

- **Financial Resources:**
  - **Budget:** Because of a university capstone project, our direct financial requirements will expect to be minimal
  - Our team will consider software and tools what it are available through Victoria University licenses or education license for being minimize personal expense.
  - Potential cost may arise from OpenAI API for the AI assistant
  - Our team would discuss any other minor expenses if university resources are unavailable.

- **Materials and Supplies:**
  - **Software License:** The project fundamentally focus to utilize free and open-source, or Licensed from University
  - **Data Storage:**
    - ✓ MySQL database
    - ✓ Will be stored for simulated PDF pamphlets in the specific watch folder, such as a local server or PC
- **Technology:**
  - **Hardware:** All development, testing and documentation activities will be implementation through personal computer
  - **Software Development Tools & Platforms:**
    - ✓ **UI/UX design Tool:** Designing the user interface will use **WIX** including user experience mockups/prototypes
    - ✓ **Front-end Development: HTML/CCS/JavaScript** Will be used for our web-application front-end development
    - ✓ **Back-end Development and Data Engineering: Python** will be adapted for this development, including the PDF scraping and ETL execution management, and the automated generation of simulated PDF pamphlets from the foundational fake dataset.
    - ✓ **Database: MySQL** will be adapted our server as the project's database management system
    - ✓ **API: REST API** will be adapted for communication between front-end and back-end components.
    - ✓ **Version Control System: Git/GitHub**
    - ✓ **Integrated Development Environments (IDEs):** It will be depending on team member's preferences (e.g., VS code, PyCharm)
    - ✓ **Collaboration Tools: GitHub Project**
    - ✓ **Deployment Platform: AWS** will be used for hosting and deploying the UrSavior web application in a scalable and secure environment. Services like EC2, RDS, and S3 will be utilized for backend hosting, database management, and storing simulated PDF pamphlets.
- **Timeline:**
  - **Overall Project Duration:**
    - ✓ The UrSavior Project will be undertaken between NIT3003 and before commencing the 'NIT3004 IT Capstone Project 2' course
    - ✓ The final project proposal and presentation will be completed by June 17th, 2025
    - ✓ All development and implementation of the project must be completed before commencing of NIT3004 course (25th August 2025)
  - **Phase-based Time Estimation:** Specific time allocations for each project phase (from initial setup, foundational dataset development, via to module implementation – user registration, PDF & ETL processing, product information display, watchlist, AI assistant, integration, testing, and final report generation) will be detailed in the "Resource Scheduling" (Timeline/Gantt chart) section.

8.3 Allocation of Resources (Mio)

To ensure efficient use of resources throughout UrSavior project, responsibilities and allocations are clearly defined based on team member expertise, project requirements, and time constrains. Each member of team is assigned roles that align with their strengths, enabling focused and progress.

- **Human Resource Allocation**
  - Each team member is responsible for specific system functions as listed in section 7.2
  - Workload is distributed evenly with weekly meetings to monitor progress
  - GitHub Projects is used to assign and tasks transparently.
- **Financial Resource Management**
  - We use open-source and university-licensed tools to minimize expenses.
  - Costs for API are limited and must be approved by the team
  - A shared spreadsheet is used for tracking all expenses.
- **Materials and Supplies Allocation**
  - Software: Python, MySQL, IDEs like VS Code – all open-source or university-provided.
  - Storage: Simulated PDF are stored locally during development, then integrated.
- **Technology Resource Allocation**
  - Development Tools: VS Code, GitHub, MySQL Workbench, WIX (for UI mockups)
  - Cloud Platform:
    AWS (Amazon Web Services) may be used for:
    - Scalable data starage
    - Hosting the ETL pipeline
    - Backend deployment and testing
- **Time Resource Allocation**
  - Key Milestones:
    - Week 1: Introduction, Functional requirements, Non-functional requirements
    - Week 2: Use Case Diagram, Sequence Diagram
    - Week 3:
    - Week 4: Presentation
    - Week 5 ~ 13+: Implementation
    - Week 14~: Project 2
  - Weekly meetings and task reviews ensure we stay on schedule.

8.4 Resource Scheduling (Mio)

To ensure timely and efficient progress of UrSavior project, a structured resource scheduling plan will be implemented. A Gantt chart will be created to clearly visualise the project timeline, task dependencies, and resource allocations. This chart will help identify milestones, manage tasks, and monitor deadlines for each development phase.

The Gantt chart will include:

- **Tasks:** A comprehensive list of all project tasks, broken down from the major from the major features and phases:
  - Fake Dataset Creation
  - PDF Generation Script Development
  - ELT Pipeline Implementation
  - User Registration and Login Module Development
  - Admin Login Module Development
  - Product Display UI & API
  - Watchlist Feature Implementation
  - AI Assistant Integration Testing
  - System Integration Testing
  - Final Report Writing
- **Assignees:** Each task will be assigned to the primary responsible team member as outlined in the section 7.2
- **Duration:** Estimated time required to complete each task.
- **Start and End Dates:** Specific planned dates for each task.
- **Dependencies:** Clear identification of tasks that must be completed before other tasks.
- **Milestones:** Key project milestones will be defined:
  - ETL Pipeline Fully Function
  - Core User Features Implemented (Registration, Loin, Display, Watchlist)
  - Core Admin Features Implemented (Login, Display, Database management)
  - AI Assistant Integrated
  - All Modules Integrated and Tested
  - Final Project Report Draft Complete

The Gantt chart will be regularly updated during weekly team meetings to ensure the project remains on track and can adapt unexpected delays or resource changes.

8.5 Contingency Planning (Austin)

To minimise the potential risks and challenges and lower the unfortunate impact on the outcome of the project, these back-up plans will be executed:

- **Unavailable team member:** the core foundation such as code, documents, and the process of the project must be shared and updated constantly in the central GitHub repository. For short-term absences of a member, the other member can cover temporarily using the knowledge, code reviews, and collaborative development as a source to keep up with the continue parts to ensure the workflow. On the other hand, in case of one member having an extended absence, the project's scope and work's timeline will be re-discussed an alternative plan in the group as well as the team leader to meet the deadlines on time.
- **Slow task progress:** a task on the verge of delay will be discussed and addressed in weekly team meetings. Re-evaluation the situation and the capability of each member is in need to find a suitable solution. Buffer time is added to the new schedule for the critical path tasks, if possible, to prevent the impact of the delayed tasks to the entire project. To update any adjustments to the schedule, the Grantt Chart will be used with its.
- **Technical issues / Software bugs:** a dedicated time is allocated for debugging and testing reasons in the final stage of the project. Any roadblocks with technical appears relatively to the chosen technology or library, then our team will collaboratively research alternative solutions. Moreover, if necessary, simplifying the affected features might be considered, following the advice of the supervisor.
- **OpenAI API service disruptions:** The functionalities of the AI assistant are operated depending on the OpenAI API service. In case of downtime or any significant changes directly impacting its functionality, the system notifies the user about the unavailability of AI assistant. The other features including discount tracking and watchlist management can continue to function independently from the AI. To demonstrate AI responses might be simulated when the live service is unavailable.
- **Exceeding the Budget:** Under this circumstance, the risk to exceed the budget is foreseeable to be low. However, for any unexpected essential expenses, our team will have a discussion together found the right solution.

8.6 Monitoring and Evaluation (Aadarsh)

Continuous monitoring and evaluation of resource utilization will be implemented to ensure the project progresses efficiently and effectively:

- **Regular Team Meetings**: Week by week group gatherings will be the essential gathering for checking on advance against the Gantt chart, talking about asset utilization (entering on group part time and errand completion), distinguishing any current or potential barricades, and arranging exercises for the ensuing week.
- **Progress Updates & Reporting:** Group individuals will give upgrades on their doled out errands amid these gatherings. The GitHub store, counting commit history and GitHub Ventures sheets, will serve as a straightforward apparatus for following code commitments, assignment status, and in general extend force.
- **Code Reviews (Peer Review):** Informal peer surveys will be conducted for key modules or complex pieces of code. This hone points to move forward code quality, encourage information sharing inside the group, and recognize potential bugs or integration issues at an early organize.
- **Testing and Feedback Cycles:** As person modules are created, they will experience unit and integration testing. Input from these inside testing cycles will be utilized to refine functionalities and client encounter.
- **Final Evaluation:** Upon completion of the "UrSaviour" venture, a comprehensive last assessment will be conducted by the group. This audit will survey the viability of the asset administration methodologies utilized, compare real advance and asset utilization against introductory plans, and analyze the taking care of of any possibilities. Lessons learned from this prepare will be recorded and can be joined into the ultimate venture report to educate future endeavors.

8. Conclusion (Austin)

In summary, the importance of Resource Management is undeniable throughout our project, as it forms a strong layer for ensuring the effectiveness and efficiency in the implementation stage. By using its principles wisely, we strongly believe that all the project goals and objectives can be achieved successfully.

13. References

[1]news.com.au. (2025). *‘Major, unavoidable’ reason Aussies aren’t eating healthily in 2025*
https://www.news.com.au/lifestyle/health/diet/major-unavoidable-reason-aussies-arent-eating-healthily-in-2025/news-story/8ecc27b39408f7ca0dd1ddc0932e0875

[2] Coles. (2025). *This Week's Catalogue*
https://d3vvi2v9oj75wh.cloudfront.net/uploads/pdf/COLNSWMETRO_210525_3TMK4D4A.pdf
[3] Woolworth. (2025). *All Specials and Offers*
https://www.woolworths.com.au/shop/browse/specials?icmpid=sm-prnav-sc-specialshub
[4] Woolworths Group. (2025). *Woolworths Group API Catalogue*
https://apiportal.woolworths.com.au/
[5] IBISWorld, 2023
https://www.ibisworld.com/australia/industry/supermarkets-and-grocery-stores/1834/
[6] Australian Bureau of Statistics. (2022). How do consumers react to the price of food? Evidence from supermarket micro data. Retrieved from
https://www.abs.gov.au/system/files/documents/3c5d0beaf792cdb58fdaa07d1015e22d/How%20do%20consumers%20react%20to%20the%20price%20of%20food%20and%20evidence%20from%20supermarket%20micro%20data.pdf

[7] Hasan, R., Rifat, S. M., & Rahman, M. A. (2020). *Smart Grocery System using Real-time Price Comparison and Personalized Notification*. Proceedings of the 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), IEEE.
https://doi.org/10.1109/ICCCNT49239.2020.9225429

[8] Shaikh, M. (2021). *Real-Time Push Notifications in Flutter Using Firebase Cloud Messaging (FCM)*. Medium.
https://medium.com/flutterdevs/flutter-firebase-push-notification-2f4680d6c5d3

[9] Townsend, S. (n.d.). What Is Resource Management and Why Is It Important? Planview.

https://www.planview.com/resources/guide/resource-management-software/resource-management-leverage-people-budgets/

[10] Gupta, O. (2025, April 25). What is Resource Management? A Comprehensive Guide. Saviom.

https://www.saviom.com/blog/what-is-resource-management/

14. Contribution

Justin Lee(28%)	Market Analysis (1.2)
	PDF Scraping and ETL Execution Management (2.3)
	Product Information and Price Display on Website (2.4)
	Important Considerations for Dataset Update Process (Non-Functional) (3.6)
	Product Information Display (Non-Functional) (3.5)
Mio Mizutani(28%)	General Background (1.1)

	User Registration (2.1)
	User Login (2.2)
	Registration (Non-Functional) (3.1)
	Login (Non-Functional) (3.2)
Austin Le(22%)	Competitor Analysis (1.3)
	AI Assistant (2.6)
	AI Assistant Response Handling (Non-Functional) (3.4)
Adarsh Thapa (22%)	Project Aims & Unique Value (1.4)
	Watchlist Management (2.5)
	Watchlist Submission and Discount Notification (Non-Functional) (3.3)