SENG306 - Database Modelling and Design

Software Requirements Specification

Version 1.3

19.05.2025

Project: AllInBee

Group No: 12

Team Members

Barış Cem Bayburtlu 202228009
Batuhan Bayazıt 202228008
Burak Aydoğmuş 202128028
Efe Çelik 202128016

Instructor: Prof. Dr. Nergiz ÇAĞILTAY

Spring 2024/2025

Contents

1. Introduction **3**
1.1 Purpose 3
1.2 Scope 4
1.3 Overview 4
2. Overall Description **5**
2.1 Product Perspective 5
2.2 Product Features 5
2.3 Assumptions and Dependencies 6
3. Specific Requirements **7**
3.1 System Interfaces 7
3.2 Functional Requirements 7
3.2.1 Data Model Overview 7
3.2.2 User Management 8
3.2.3 Cafeteria System 9
3.2.4 Ring Tracking System 10
3.2.5 Appointment System 10
4. Non-Functional Requirements **12**
4.1 Performance Requirements 12
4.2 Security Requirements 12
4.3 Accessibility Requirements 13
4.4 Usability Requirements 13
4.5 Reliability Requirements 14
4.6 Data Requirements 14
Conceptual Model 17
Logical Model (Mapping 3NF) 18
Physical Model (SQL) 19
UI **33**
References 36
Source Code (Github Link) https://github.com/byigitt/seng306-allinbee **37**
Appendices **37**

Revision History

Date	Description	Author	Comments
24.03.2025	Version 1.0	Team AllInBee	SRS created
29.03.2025	Version 1.1	Team AllInBee	SRS reviewed
23.04.2025	Version 1.2	Team AllInBee	Problems are fixed, new model with new requirements
19.05.2025	Version 1.3	Team AllInBee	Finalized

Document Approval

The following Software Requirements Specification has been accepted and approved by the following:

Signature	Printed Name	Title	Date

1. Introduction

1.1 Purpose

This document specifies the software requirements for the AllInBee application, designed for Çankaya University students and staff. It details the functional and non-functional requirements, with a particular emphasis on the **data requirements** necessary to support the application's features. The AllInBee app aims to consolidate essential campus services – including cafeteria information and payments, real-time ring bus tracking, and appointment booking – into a single, user-friendly mobile platform.

1.2 Scope

The AllInBee project encompasses the development of a mobile application with the following core functionalities:

- **User Account Management:** Secure registration, login, and profile management.
- **Cafeteria System:** Displaying daily menus, nutritional information, prices, managing a digital wallet, facilitating QR code payments, and providing transaction history.
- **Ring Bus Tracking:** Real-time GPS tracking of university ring buses, displaying routes, stops, and estimated arrival times (ETAs).
- **Appointment Booking:** Scheduling appointments with various university services (sports center, psychologist, library rooms).

This document outlines the specific requirements for each feature, focusing on the data entities, attributes, relationships, and constraints needed for their implementation. It targets iOS and Android platforms and considers integration with existing university systems where applicable.

1.3 Overview

This document outlines the system requirements for the AllInBee project.

- **Section 1:** Provides the introduction, purpose, scope, definitions, and overview.
- **Section 2:** Gives a general description of the product perspective, key functions/features from a data viewpoint (aligned with the new EER model), user characteristics related to data interaction, operating environment, constraints, and dependencies influencing data management.
- **Section 3:** Details the specific requirements, focusing heavily on functional requirements and the underlying data model (EER v1.2), entities, attributes, relationships, and processing for each core feature.
- **Section 4:** Specifies the non-functional requirements, emphasizing aspects relevant to data handling like performance, security, and reliability.

2. Overall Description

2.1 Product Perspective

AllInBee is envisioned as a client-server application. The mobile app (client) will interact with a backend server that manages the database (structured according to the EER v1.2 model) and business logic. Crucially, the backend may need to interface with existing Çankaya University systems (e.g., student information system for authentication/validation, potential payment gateways for wallet top-up) via APIs. Secure and reliable data exchange with these external systems is paramount. The system must store and manage data related to Users (`Student`, `Staff`, `Admin`), cafeteria operations (`Dish`, `Menu`, `DigitalCard`, `QR`), bus logistics (`Bus`, `Route`, `Station`), and appointments (`BookAppointment`, `SportAppointment`, `HealthAppointment`, `Book`).

2.2 Product Features

The main features, driven by specific data needs defined in the EER v1.2 model, are:

1. User Management: Storing and managing `User` credentials (`UserID`, `E-Mail`, `Password`) and profile data (`Name`, `Phone_Number`), including subtypes `Student`, `Admin`, `Staff`.
2. Cafeteria Services: Managing `Dish` information, daily `Menu` offerings, `Student DigitalCard` balances, `QR` code generation and payment processing, and tracking sales (`Sales`).
3. Ring Bus Tracking: Storing `Route`/`Station` data and processing real-time `Bus` location (`Live_Location`) data to provide ETAs. Managing User `Favorite_Routes`.
4. Appointment Scheduling: Managing availability (implicitly), booking `BookAppointment`, `SportAppointment`, `HealthAppointment`, managing library `Book` borrowing, and tracking appointment details.

2.3 Assumptions and Dependencies

- **External System Integration**: Assumes availability and accessibility of necessary Çankaya University systems/APIs for data validation (e.g., initial student email verification). Successful data exchange depends on these external systems functioning correctly.
- **Payment System**: Assumes a reliable external payment gateway handles actual credit card processing for DigitalCard top-ups (deposit money relationship). AllInBee's DB manages the internal DigitalCard.Balance and transaction context. QR payments are internal balance deductions.
- **GPS Data Feed**: Assumes a reliable source provides real-time GPS data for Bus entities, which the AllInBee backend can ingest and store/update (Live_Location). Assumes necessary timestamp information is available with GPS data.
- **Data Provisioning**: Assumes authorized Staff will provide and maintain accurate data (Dish, Menu).
- **Appointment Availability**: The EER model stores booked appointments. The system must calculate availability by checking for time conflicts based on existing appointments for a given resource (e.g., specific sports facility, specific health service slot). The logic for defining resources and checking conflicts needs implementation.

3. Specific Requirements

3.1 System Interfaces

- **Mobile App <-> Backend API:** The primary interface for data exchange. Mobile clients send requests (e.g., fetch Menu, create QR, book SportAppointment, get Bus location) and receive data responses via secure RESTful APIs designed around the EER model entities.

- **Backend <-> Database:** The backend system performs CRUD (Create, Read, Update, Delete) operations on the application database, adhering to the EER v1.2 structure and constraints.
- **Backend <-> External Systems (Potential):** Interfaces via APIs for user validation, payment gateway interaction (for deposit money), or GPS data ingestion (Bus.Live_Location), requiring secure data handling.

3.2 Functional Requirements

3.2.1 Data Model Overview

A relational database model implementing the provided EER diagram (v1.2) is required. The core is the User entity, identified by UserID and having a unique E-Mail. User undergoes overlapping ('O') specialization into Student, Admin, and Staff subtypes, linked via UserID.

- Students have a mandatory 1:1 relationship with DigitalCard (for balance) and relationships for creating QR codes, depositing money, taking appointments, and borrowing books.
- Staff interacts with Menu (writes) and Appointments (manages), implying role-based access.
- Admin manages Student and Staff records.
- The Cafeteria system involves Dish (items), Menu (daily offerings with price/date), QR (payment tokens linked to Menu), DigitalCard (student balance), and Sales (daily revenue tracking linked to Menu).
- Ring Tracking uses Bus (with Live_Location), Route, and Station (with Location), linked via M:N relationships. Users can have Favorite_Routes.
- Appointments are modeled using union ('O') specialization into BookAppointment, SportAppointment, and HealthAppointment. Each subtype stores relevant details (e.g., Sport_Type, Health_Type). BookAppointment links to the Book entity (M:N borrow relationship between Student and Book). Appointments are linked to Student (takes) and Staff (manages).
- Data integrity must be enforced through Primary Keys (PKs), Foreign Keys (FKs - e.g., UserID in subtypes), relationship cardinalities (1:1, 1:N, M:N),

participation constraints (total/partial), unique constraints (e.g., User.E-Mail), and appropriate data types.

3.2.2 User Management

- **Functional Requirements for User Management:**
  - **REQ-UM-1**: The system must store base user account information in the **User** table (PK: **UserID**, attributes: **E-Mail**, **Password**, **Name**, **Phone_Number**). Specific user types (**Student**, **Admin**, **Staff**) must be represented by corresponding records in subtype tables, linked via UserID (FK). Specialization is overlapping ('O').
  - **REQ-UM-2**: The system must enforce uniqueness constraints on **User.E-Mail**.
  - **REQ-UM-3**: The system must allow users to update their profile information (**Phone_Number**) and securely update their **Password** (hashed).
  - **REQ-UM-4**: The system must differentiate user roles and permissions based on the existence of records in the **Student**, **Admin**, **Staff** subtype tables and their relationships (e.g., only **Staff** connected to **writes** can modify **Menu**, only **Admin** can use **manage** relationships).
  - **REQ-UM-5**: All sensitive user data, especially **Password**, must be stored securely (e.g., using strong hashing algorithms).
  - **REQ-UM-6**: The system must manage student-specific interactions via the Student entity, primarily its mandatory 1:1 link to **DigitalCard** and relationships for **QR** creation, **Appointment** taking, and **Book** borrowing.
  - **REQ-UM-7**: The system must manage staff-specific interactions via the **Staff** entity and its relationships, such as **writes Menu** and **manages Appointment**, implying role-based capabilities.

3.2.3 Cafeteria System

- **Functional Requirements for Cafeteria System:**
  - **REQ-CS-1**: The system must store data for cafeteria food items in the **Dish** table (PK: **Dish_ID**, attributes: **Dish_Name**, **Calories**).

- **REQ-CS-2**: The system must store daily menu offerings in the Menu table (PK: **Menu_ID**, attributes: **Menu_Name**, **Price**, **Date**). (Note: Assumes Menu represents a specific offering, potentially linking implicitly or explicitly to **Dish**). Must link **Menu** to **Sales** (1:1).
- **REQ-CS-3**: The system must allow authorized **Staff** users (via **writes** relationship) to create, update, and delete **Dish** and **Menu** data.
- **REQ-CS-4**: The system must maintain a **DigitalCard** entity for each **Student** user (linked 1:1, mandatory), storing their current **Balance** (PK: **Card_NO**).
- **REQ-CS-5**: The system must provide a QR code payment mechanism: **Student** creates **QR** (PK: **QR_ID**, attributes: **Expired_Date**, **RemainingTime**). The **QR** must be usable via the **pays for** relationship (1:N with Menu) to securely identify the student, verify sufficient **DigitalCard.Balance**, deduct the **Menu.Price**, and link the payment to the specific **Menu** item purchased. Database updates (balance deduction, linking QR to Menu) must be atomic.
- **REQ-CS-6**: The system must allow **Student** users to add funds to their **DigitalCard.Balance** via the **deposit money** relationship (attributes **Date**, **Amount**), triggered by successful external payment confirmation.
- **REQ-CS-7**: The system must allow students to retrieve their deposit history (querying **deposit money** relationship data linked to their **DigitalCard**).
- **REQ-CS-8**: The system must allow students to retrieve their purchase history (querying **Menu** items linked via **pays for to QR** codes they **created**).
- **REQ-CS-9**: The system must allow authorized **Staff** to query aggregate sales data from the **Sales** table (attributes: **Date**, **Daily_Revenue**, **Num_Sold**), which is linked 1:1 to **Menu**.
- **REQ-CS-10**: The system must be able to track or report the number of meals purchased per user per day by analyzing the **QR pays for Menu** relationship, linking back to the **Student** who **created** the **QR**, and grouping by **Menu.Date**.

3.2.4 Ring Tracking System

- **Functional Requirements for Ring Tracking System:**
    - **REQ-RT-1**: The system must store definitions of ring bus routes in the **Route** table (PK: **Route_ID**, attributes: **Name**, **Departure_Times**) and associated stops in the **Station** table (PK: **Station_ID**, attributes: **Name**, composite **Location** (Latitude, Longitude)). Routes and Stations are linked via a M:N relationship.
    - **REQ-RT-2**: The system must store and update the real-time geographic coordinates (**Live_Location** composite attribute) and an associated timestamp (assumed required) for active ring buses in the **Bus** table (PK: **Vehicle_ID**).
    - **REQ-RT-3**: The system must retrieve and display current **Bus.Live_Location** data on a map interface.
    - **REQ-RT-4**: The system must retrieve and display **Route** paths (derived from **Station** order) and **Station.Location** markers on a map interface.
    - **REQ-RT-5**: The system must calculate Estimated Times of Arrival (ETAs) for **Stations** based on **Bus.Live_Location**, **Route/Station** data (including the **drive in** M:N relationship between **Bus** and **Route**), and potentially external traffic information. ETAs are calculated on demand.
    - **REQ-RT-6**: The system must allow users to find the nearest **Station** based on their current device location and stored **Station.Location** data.
    - **REQ-RT-7**: The system must store user preferences for favorite routes using the **User.Favorite_Routes** multivalued attribute.
    - **REQ-RT-8**: The system must allow authorized personnel to create, update, and delete **Route** and **Station** data.

3.2.5 Appointment System

- **Functional Requirements for Appointment System:**

- **REQ-AS-1**: The system must store information about different university services implicitly through the specialized appointment types: **BookAppointment**, **SportAppointment**, **HealthAppointment**. It must store details about rentable library items in the Book table (PK: **ISBN**, attributes: **Title**, **Author**, **Quantity**).
- **REQ-AS-2**: The system must allow querying for available appointment times by checking for non-conflicting time intervals (**StartTime**, **EndTime**, **Time_Period**) based on existing records in the relevant appointment subtype tables for the implicitly defined resource. Availability is determined by the absence of conflicts.
- **REQ-AS-3**: The system must allow **Student** users (via **takes** M:N relationship) to book an available time slot. This action must create a record in the appropriate subtype table (**BookAppointment**, **SportAppointment**, **HealthAppointment**) with a unique Appointment_ID (PK), common details (linking to **User**, **Staff** via takes/manages), time details (**StartTime**, **EndTime**, **Time_Period**), and service-specific details (**Sport_Type**, **Health_Type**, or linking to **Book** via borrow M:N relationship for **BookAppointment**).
- **REQ-AS-4**: The system must include a **Status** attribute (assumed) on appointment records (e.g., 'Scheduled', 'Completed', 'Cancelled'). Cancellation must update this status.
- **REQ-AS-4b**: The system must prevent double booking by ensuring no conflicting appointment (same resource, overlapping time) exists before creating a new appointment record (atomic check/transaction).
- **REQ-AS-5**: The system must store user-provided or service-specific information required for booking (e.g., **Sport_Type**, **Health_Type**) within the respective appointment subtype record. For BookAppointment, it must store BorrowDate and ReturnDate.
- **REQ-AS-6**: The system must send a confirmation (email) upon successful booking, using stored **User.E-Mail** and appointment details.
- **REQ-AS-7**: The system must allow users (**Students**) to view their scheduled appointments by querying the appointment subtype tables linked to their **UserID** via the **takes** relationship.

- **REQ-AS-8**: The system must allow users (**Students**) to change or cancel their appointments (updating the assumed **Status** attribute), subject to predefined rules (e.g., cancellation deadlines).
- **REQ-AS-9**: The system must support sending reminder notifications based on stored appointment **StartTime** data (requires an assumed mechanism like a reminder flag/timestamp).
- **REQ-AS-10**: The system must allow authorized **Staff** users (via **manages** M:N relationship) to manage (view, potentially cancel/confirm by updating status) appointment records relevant to their specific service type/resource.
- **REQ-AS-11**: For **BookAppointments** linked to **Book** via the **borrow** relationship, the system must manage **Book.Quantity**, decrementing upon borrowing (**BorrowDate**) and potentially incrementing upon return (**ReturnDate**). The BookAppointment should store **BorrowDate** and **ReturnDate**.

4. Non-Functional Requirements

4.1 Performance Requirements

- **NFR-PERF-1**: Data retrieval operations (viewing **Menu**, **Bus.Live_Location**, **DigitalCard.Balance**, **Appointment** schedules) must be responsive, completing quickly under normal load.
- **NFR-PERF-2**: The database and backend system must handle concurrent data requests from many users effectively. Database queries, especially those involving joins (e.g., user subtypes, appointments, menu/dish), location lookups (**Station**, **Bus**), and balance checks (**DigitalCard**), should be optimized (e.g., proper indexing on PKs, FKs, frequently queried attributes like **Date**, **StartTime**, location coordinates).

4.2 Security Requirements

- **NFR-SEC-1**: User registration should ideally be restricted or validated against valid Çankaya University email domains (**User.E-Mail**).

- **NFR-SEC-2**: User passwords (**User.Password**) must be stored using strong, one-way hashing algorithms in the database. Passwords must never be stored in plain text.
- **NFR-SEC-3**: All data transmitted between the client app and the backend server must be encrypted using HTTPS.
- **NFR-SEC-4**: Access to sensitive user information (e.g., viewing other users' details beyond basic directory info, financial transaction history via deposit money or QR usage, managing **Student**/**Staff** via Admin) must be strictly limited to authorized personnel based on their specific User subtype (**Admin**, **Staff** with specific roles) as defined by system authorization logic and EER relationships. Regular **Users** (**Student** or general **User**) must only access their own data or publicly available information (e.g., general **Menu**, **Route** info).
- **NFR-SEC-5**: The data flow supporting QR code payments (**QR** creation by **Student**, **pays for** link to **Menu**, deduction from **DigitalCard.Balance**) must be secure against common vulnerabilities (e.g., replay attacks, unauthorized balance modification). Database transactions for payments (**Balance** update) must be atomic and consistent.

4.3 Accessibility Requirements

- **NFR-ACC-1**: Data presented in the UI must be compatible with assistive technologies like screen readers.
- **NFR-ACC-2**: The application storing and retrieving data must function correctly on both supported iOS and Android platforms.

4.4 Usability Requirements

- **NFR-USAB-1**: Data retrieved from the database (e.g., **Menu** items/prices, **Bus** ETAs, **Appointment** details, **DigitalCard.Balance**) must be presented to the user in a clear, understandable format.
- **NFR-USAB-2**: Users must be able to interact with data features (booking appointments, paying via **QR**, viewing history, checking **Bus** locations) easily and quickly.

4.5 Reliability Requirements

- **NFR-REL-1**: The system must store user information (**User**, subtypes), financial data (**DigitalCard.Balance**, transaction context), appointment data, and other critical entities safely and persistently.
- **NFR-REL-2**: The database must ensure data integrity through appropriate constraints derived from the EER model: PKs, FKs (e.g., **UserID** linking **User** to subtypes, **Card_NO** linking **DigitalCard** to **Student**), data types, non-null constraints where applicable, uniqueness constraints (**User.E-Mail**), and enforcing cardinalities/participation (e.g., the mandatory 1:1 between Student and **DigitalCard**).
- **NFR-REL-3**: Regular backups of the application database must be performed to prevent data loss.

4.6 Data Requirements

- **NFR-DATA-1:** Relationship: **drive in** (between **Bus** and **Route**)
    - A **Bus** can drive in one or many **Routes** (M:N).
    - A **Route** can be driven by one or many **Buses** (M:N).
    - Participation of **Bus** in the 'drive in' relationship is **optional** (Partial).
    - Participation of **Route** in the 'drive in' relationship is **optional** (Partial).
- **NFR-DATA-2:** Relationship: **has** (between **Route** and **Station**)
    - A **Route** can have one or many **Stations** (M:N).
    - A **Station** can be part of one or many **Routes** (M:N).
    - Participation of **Route** in the 'has' relationship is **optional** (Partial)
    - Participation of **Station** in the 'has' relationship is **mandatory** (Total).
- **NFR-DATA-3:** Relationship: **manages** (between **Staff** and **Route**)
    - A **Staff** can manage at one or many **Routes** (M:N).
    - A **Route** can be managed at by one or many **Staff** (M:N).
    - Participation of **Staff** in the 'manages' relationship is **optional** (Partial).
    - Participation of **Route** in the 'manages' relationship is **optional** (Partial).

- **NFR-DATA-4:** Relationship: **looks** (between **User** and **Route**)
    - A **User** can look at one or many **Routes** (M:N).
    - A **Route** can be looked at by one or many **Users** (M:N).
    - Participation of **User** in the 'looks' relationship is **optional** (Partial).
    - Participation of **Route** in the 'looks' relationship is **optional** (Partial).
- **NFR-DATA-5:** Relationship: **has** (between **Student** and **DigitalCard**)
    - A **Student** must have exactly one **DigitalCard** (1:1).
    - A **DigitalCard** must belong to exactly one **Student** (1:1).
    - Participation of **Student** in the 'has' relationship is **mandatory** (Total).
    - Participation of **DigitalCard** in the 'has' relationship is **mandatory** (Total).

- **NFR-DATA-6:** Relationship: **deposit money** (between **Student** and **DigitalCard**)
    - A **Student** can make one or many 'deposit money' transactions related to their **DigitalCard** (1:N).
    - A **DigitalCard** can receive zero or many 'deposit money' transactions from its associated **Student** (1:N).
    - Participation of **Student** in 'deposit money' is **optional** (Partial).
    - Participation of **DigitalCard** in 'deposit money' is **optional** (Partial).
- **NFR-DATA-7:** Relationship: **create** (between **DigitalCard** and **QR**)
    - A **DigitalCard** can create one or many **QR** codes (1:N).
    - A **QR** code must be created by exactly one **DigitalCard** (1:N).
    - Participation of **DigitalCard** in the 'create' relationship is **mandatory** (Total).
    - Participation of **QR** in the 'create' relationship is **optional** (Partial).
- **NFR-DATA-8:** Relationship: **manage** (between **Admin** and **Student**)
    - An **Admin** can manage one or many **Students** (1:N).
    - A **Student** can be managed by exactly one **Admin** (1:1).
    - Participation of **Admin** in this 'manage' relationship is **mandatory** (Total).

    - Participation of **Student** in this 'manage' relationship is **optional** (Partial).
- **NFR-DATA-9:** Relationship: **manage** (between **Admin** and **Staff**)
    - An **Admin** can manage one or many **Staff** members (1:N).
    - A **Staff** member can be managed by exactly one **Admin** member (1:1).
    - Participation of **Admin** in this 'manage' relationship is **mandatory** (Total).
    - Participation of **Staff** in this 'manage' relationship is **optional** (Partial).
- **NFR-DATA-10:** Relationship: **manage** (between **Staff** and **Menu**)
    - A **Staff** member can manage one or many **Menu** entries (1:N).
    - A **Menu** entry must be managed by exactly one **Staff** member (1:1).
    - Participation of **Staff** in the ‘manage’ relationship is **mandatory** (Total).
    - Participation of **Menu** in the ‘manage’ relationship is **optional** (Partial).

- **NFR-DATA-11:** Relationship: **part of** (between **Dish** and **Menu**)
    - A **Dish** can be part of one or many **Menu** entries (1:N).
    - A **Menu** entry can contain one or many **Dishes** (1:N).
    - Participation of **Dish** in the 'part of' relationship is **mandatory** (Total).
    - Participation of **Menu** in the 'part of' relationship is **optional** (Partial).
- **NFR-DATA-12:** Relationship: **results in** (between **Menu** and **Sales**)
    - A **Menu** can be part of one or many **Sales** entries (1:N).
    - A **Sales** record must correspond to exactly one **Menu** entry (1:1).
    - Participation of **Menu** in the 'results in' relationship is **mandatory** (Total).
    - Participation of **Sales** in the 'results in' relationship is **optional** (Partial).
- **NFR-DATA-13:** Relationship: **pays for** (between **QR** and **Menu**)
    - A **QR** code can pay for zero or one **Menu** item in a transaction (1:1).

    - A **Menu** item/entry can be paid by one or many **QR** codes over time (1:N).
    - Participation of **QR** in the 'pays for' relationship is **optional** (Partial).
    - Participation of **Menu** in the 'pays for' relationship is **optional** (Partial).
- **NFR-DATA-14:** Relationship: **takes** (between **Student** and **Appointment**)
    - A **Student** can take one or many **Appointments** (M:N).
    - An **Appointment** must be taken by exactly one **Student** (1:1).
    - Participation of **Student** in the 'takes' relationship is **mandatory** (Total).
    - Participation of **Appointment** in the 'takes' relationship is **optional** (Partial).
- **NFR-DATA-15:** Relationship: **manages** (between **Staff** and **Appointment**)
    - A **Staff** member can manage one or many **Appointments** (M:N).
    - An **Appointment** must be managed by one **Staff** member (1:1).
    - Participation of **Staff** in this 'manages' relationship is **mandatory** (Total).
    - Participation of **Appointment** in this 'manages' relationship is **optional** (Partial).
- **NFR-DATA-16:** Relationship: **borrow** (between **Book** and **BookAppointment**)
    - A **BookAppointment** can borrow one, or many **Books** (1:N).
    - A **Book** can be borrowed by one or many **BookAppointment** over time (M:N).
    - Participation of **BookAppointment** in the 'borrow' relationship is **optional** (Partial).
    - Participation of **Book** in the 'borrow' relationship is **mandatory** (Total).

Conceptual Model

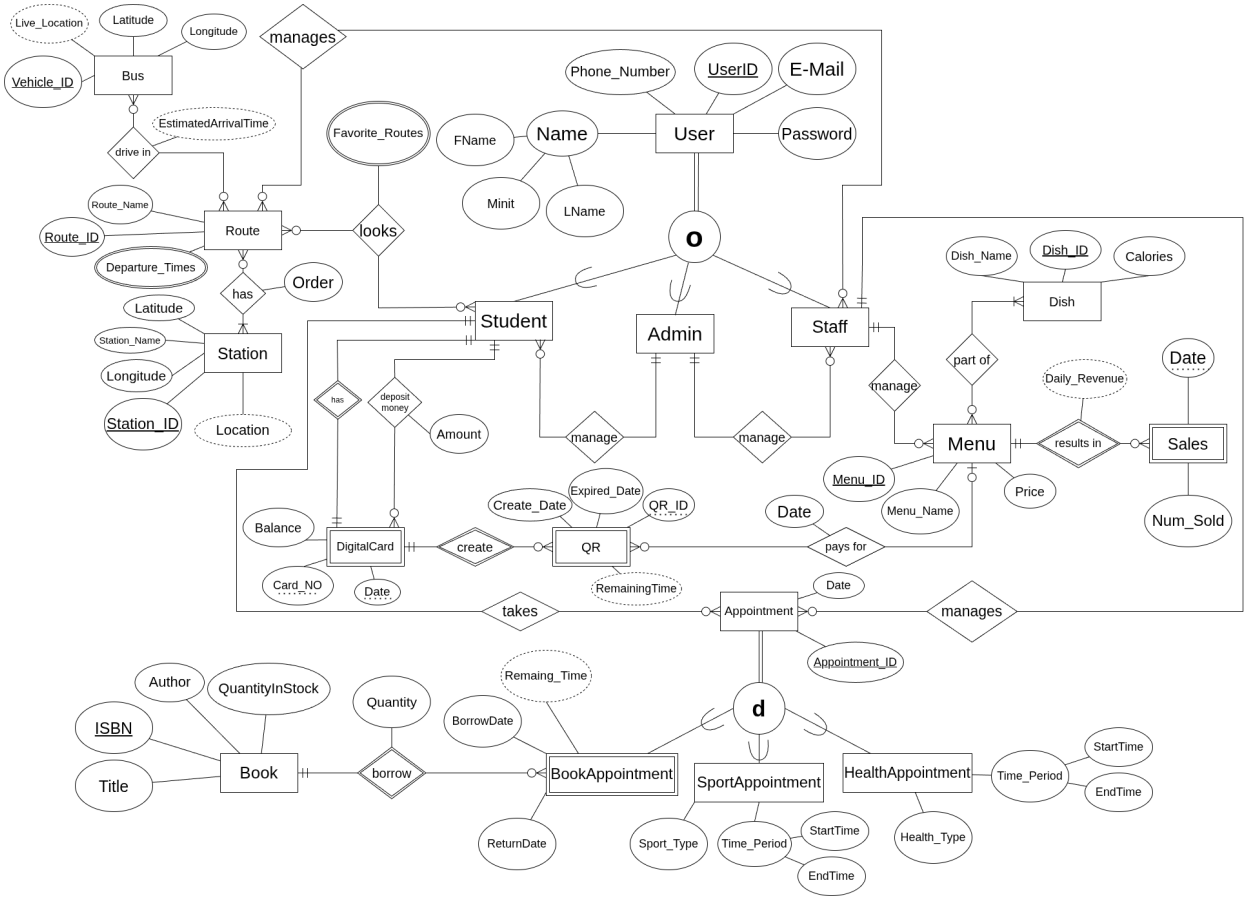


Image Conceptual Model

Logical Model (Mapping 3NF)

USER[UserID, FName, Minit, LName, Phone_Number, E-Mail, Password]

STUDENT[UserID (FK refers to USER), MNGUserID]

ADMIN[UserID (FK refers to USER)]

STAFF[UserID (FK refers to USER), MNGUserID]

MENU[Menu_ID, Menu_Name, Price MngUserID]

DISH[Dish_ID, Dish_Name, Calories]

ROUTE[Route_ID, RouteName]

STATION[Station_ID, Station_Name, Station_Latitude, Station_Longitude]

BUS[Vehicle_ID, Live_Latitude, Live_Longitude]

APPOINTMENT[Appointment_ID, SUserID, MNGStaffID, AppointmentDate]

SPORTAPPOINTMENT[Appointment_ID (FK refers to APPOINTMENT), Sport_Type, StartTime, EndTime]

HEALTHAPPOINTMENT[Appointment_ID (FK refers to APPOINTMENT), Health_Type, StartTime, EndTime]

BOOK[ISBN, Title, Author, QuantityInStock, CurrentQuantity]

DIGITALCARD[SUserID, Card_NO, HASUserID, Balance]

DIGITALCARD_DEPOSIT[Card_NO, Date, DepositMoneyAmount]

QR[Card_NO,QR_ID, Date, DCard_NO,PaysFor_Date]

QR_PAYMENT[QR_ID, Create_Date, Expired_Date,MMenu_ID]

BOOKAPPOINTMENT[BISBN, Appointment_ID, BorrowDate, ReturnDate, Borrow_Quantity]

SALES[MMenu_ID, Date, Num_Sold]

PART_OF[Menu_ID, Dish_ID]

MANAGES[UserID, Route_ID]

LOOKS[UserID, Route_ID]

DRIVE_IN[Vehicle_ID, Route_ID]

HAS[Route_ID, Station_ID, Order]

ROUTE_DEPARTURE_TIMES[Route_ID , DepartureTime]

FAVORITE_ROUTES[UserID, Route_ID,Favorite_Route]

Physical Model (SQL)

```
-----------------------------------------------------------------
-- A. Core Tables
-----------------------------------------------------------------
CREATE TABLE "USER" (
  "UserID"       varchar(25) PRIMARY KEY,
  "FName"        varchar(100) NOT NULL,
  "Minit"        char(1),
  "LName"        varchar(100) NOT NULL,
  "Phone_Number"  varchar(30),
  "E-Mail"       varchar(255) UNIQUE,
  "Password"     text NOT NULL
);

CREATE TABLE "ADMIN" (
  "UserID" varchar(25) PRIMARY KEY
        REFERENCES "USER"("UserID") ON DELETE CASCADE
);

CREATE TABLE "STUDENT" (
```

```
  "UserID"   varchar(25) PRIMARY KEY
          REFERENCES "USER"("UserID") ON DELETE CASCADE,
  "MNGUserID" varchar(25)
          REFERENCES "USER"("UserID") ON DELETE SET NULL
);

CREATE TABLE "STAFF" (
  "UserID"   varchar(25) PRIMARY KEY
          REFERENCES "USER"("UserID") ON DELETE CASCADE,
  "MNGUserID" varchar(25)
          REFERENCES "USER"("UserID") ON DELETE SET NULL
);
```

-- B. Food Module

```
CREATE TABLE "MENU" (
  "Menu_ID"   uuid PRIMARY KEY DEFAULT uuid_generate_v4(),
  "Menu_Name" varchar(120) NOT NULL,
  "Price"     numeric(10,2) NOT NULL CHECK ("Price" >= 0),
  "MngUserID" varchar(25)
           REFERENCES "USER"("UserID") ON DELETE RESTRICT
);

CREATE TABLE "DISH" (
  "Dish_ID"   uuid PRIMARY KEY DEFAULT uuid_generate_v4(),
  "Dish_Name" varchar(120) UNIQUE NOT NULL,
  "Calories"  int CHECK ("Calories" >= 0)
);

CREATE TABLE "PART_OF" (
  "Menu_ID" uuid REFERENCES "MENU"("Menu_ID")  ON DELETE CASCADE,
  "Dish_ID" uuid REFERENCES "DISH"("Dish_ID")  ON DELETE CASCADE,
  PRIMARY KEY ("Menu_ID","Dish_ID")
);

CREATE TABLE "SALES" (
  "MMenu_ID" uuid REFERENCES "MENU"("Menu_ID") ON DELETE CASCADE,
```

```
  "Date"     date NOT NULL,
  "Num_Sold" int  NOT NULL DEFAULT 0 CHECK ("Num_Sold" >= 0),
  PRIMARY KEY ("MMenu_ID","Date")
);
-----------------------------------------------------------------
-- C. Transport Module
-----------------------------------------------------------------
CREATE TABLE "ROUTE" (
  "Route_ID"  uuid PRIMARY KEY DEFAULT uuid_generate_v4(),
  "RouteName" varchar(120) UNIQUE NOT NULL
);




CREATE TABLE "STATION" (
  "Station_ID" uuid PRIMARY KEY DEFAULT uuid_generate_v4(),
  "Station_Name"       varchar(120) NOT NULL,
  "Station_Latitude"   numeric(9,6) NOT NULL,
  "Station_Longitude"  numeric(9,6) NOT NULL,
  UNIQUE ("Station_Latitude","Station_Longitude")
);

CREATE TABLE "BUS" (
  "Vehicle_ID"    varchar(40) PRIMARY KEY,
  "Live_Latitude" numeric(9,6),
  "Live_Longitude" numeric(9,6)
);

CREATE TABLE "DRIVE_IN" (
  "Vehicle_ID" varchar(40)
       REFERENCES "BUS"("Vehicle_ID") ON DELETE CASCADE,
  "Route_ID"   uuid
       REFERENCES "ROUTE"("Route_ID") ON DELETE CASCADE,
  PRIMARY KEY ("Vehicle_ID","Route_ID")
);

CREATE TABLE "HAS" (
  "Route_ID"   uuid REFERENCES "ROUTE"("Route_ID") ON DELETE CASCADE,
  "Station_ID" uuid REFERENCES "STATION"("Station_ID") ON DELETE
CASCADE,
  "Order"      int NOT NULL CHECK ("Order" > 0),
  PRIMARY KEY ("Route_ID","Station_ID"),
```

```
  UNIQUE ("Route_ID","Order")
);

CREATE TABLE "ROUTE_DEPARTURE_TIMES" (
  "Route_ID" uuid REFERENCES "ROUTE"("Route_ID") ON DELETE CASCADE,
  "DepartureTime" time NOT NULL,
  PRIMARY KEY ("Route_ID","DepartureTime")
);
```

```
-----------------------------------------------------------------
-- D. Appointment Module
-----------------------------------------------------------------
CREATE TABLE "APPOINTMENT" (
  "Appointment_ID" uuid PRIMARY KEY DEFAULT uuid_generate_v4(),
  "SUserID"   varchar(25)
          REFERENCES "STUDENT"("UserID") ON DELETE CASCADE,
  "MNGStaffID" varchar(25)
          REFERENCES "STAFF"("UserID")   ON DELETE RESTRICT,
  "AppointmentDate" date NOT NULL
);

CREATE TABLE "SPORTAPPOINTMENT" (
  "Appointment_ID" uuid PRIMARY KEY
      REFERENCES "APPOINTMENT"("Appointment_ID") ON DELETE CASCADE,
  "Sport_Type" varchar(80) NOT NULL,
  "StartTime"  time NOT NULL,
  "EndTime"    time NOT NULL,
  CHECK ("EndTime" > "StartTime")
);

CREATE TABLE "HEALTHAPPOINTMENT" (
  "Appointment_ID" uuid PRIMARY KEY
      REFERENCES "APPOINTMENT"("Appointment_ID") ON DELETE CASCADE,
  "Health_Type" varchar(80) NOT NULL,
  "StartTime"   time NOT NULL,
  "EndTime"     time NOT NULL,
  CHECK ("EndTime" > "StartTime")
);
```

```
-----------------------------------------------------------------
-- E. Library Module
-----------------------------------------------------------------
CREATE TABLE "BOOK" (
  "ISBN"            varchar(20) PRIMARY KEY,
  "Title"           varchar(255) NOT NULL,
  "Author"          varchar(255),
  "QuantityInStock"   int NOT NULL CHECK ("QuantityInStock" >= 0),
  "CurrentQuantity"   int NOT NULL CHECK ("CurrentQuantity" BETWEEN 0 AND
"QuantityInStock")
);

CREATE TABLE "BOOKAPPOINTMENT" (
  "BISBN"        varchar(20)
         REFERENCES "BOOK"("ISBN") ON DELETE RESTRICT,
  "Appointment_ID" uuid
         REFERENCES "APPOINTMENT"("Appointment_ID") ON DELETE
CASCADE,
  "BorrowDate"  date NOT NULL,
  "ReturnDate"  date,
  "Borrow_Quantity" int NOT NULL DEFAULT 1,
  PRIMARY KEY ("BISBN","Appointment_ID")
);

-----------------------------------------------------------------
-- F. Digital Card and Payment Module
-----------------------------------------------------------------
CREATE TABLE "DIGITALCARD" (
  "SUserID"  varchar(25) PRIMARY KEY
           REFERENCES "STUDENT"("UserID") ON DELETE CASCADE,
  "Card_NO"  varchar(40) UNIQUE NOT NULL,
  "Date"     timestamptz NOT NULL DEFAULT now(),
  "HASUserID" varchar(25)  -- kartı veren personel
           REFERENCES "STAFF"("UserID") ON DELETE SET NULL,
  "Balance"  numeric(10,2) NOT NULL DEFAULT 0 CHECK ("Balance" >= 0),
  "DepositMoneyAmount" numeric(10,2) NOT NULL DEFAULT 0
);

CREATE TABLE "QR" (
  "SUserID"      varchar(25)
       REFERENCES "STUDENT"("UserID") ON DELETE CASCADE,
  "Card_NO"      varchar(40),
  "Date"         timestamptz NOT NULL DEFAULT now(),
  "QR_ID"        uuid PRIMARY KEY DEFAULT uuid_generate_v4(),
```

```
  "Create_Date"  timestamptz NOT NULL DEFAULT now(),
  "Expired_Date" timestamptz NOT NULL,
  "DCard_NO"     varchar(40),
  "MMenu_ID"     uuid REFERENCES "MENU"("Menu_ID") ON DELETE SET
NULL,
  "PaysFor_Date" timestamptz
);



----------------------------------------------------------------
-- G.  Other Tables
----------------------------------------------------------------
CREATE TABLE "MANAGES" (
  "UserID"   varchar(25) REFERENCES "USER"("UserID") ON DELETE CASCADE,
  "Route_ID" uuid        REFERENCES "ROUTE"("Route_ID") ON DELETE
CASCADE,
  PRIMARY KEY ("UserID","Route_ID")
);

CREATE TABLE "LOOKS" (
  "UserID"   varchar(25) REFERENCES "USER"("UserID") ON DELETE CASCADE,
  "Route_ID" uuid        REFERENCES "ROUTE"("Route_ID") ON DELETE
CASCADE,
  PRIMARY KEY ("UserID","Route_ID")
);

CREATE TABLE "FAVORITE_ROUTES" (
  "UserID"   varchar(25) REFERENCES "USER"("UserID") ON DELETE CASCADE,
  "Route_ID" uuid        REFERENCES "ROUTE"("Route_ID") ON DELETE
CASCADE,
  "Favorite_Route" boolean NOT NULL DEFAULT true,
  PRIMARY KEY ("UserID","Route_ID")
);
================================================================
  2)  10 Data Loading Examples
================================================================

1) User + Roles
INSERT INTO "USER"("UserID", "E-Mail", "FName","LName","Password")
VALUES
('USR_ADMIN', 'admin@example.com', 'Super','Admin','phash'),
```

```
('USR_STAFF1', 'staff1@example.com', 'John','Staff','phash'),
('USR_STAFF2', 'staff2@example.com', 'Jane','Driver','phash'),
('USR_STD1', 'student1@example.com', 'Alice','Wonder','phash'),
('USR_STD2', 'student2@example.com', 'Bob','Builder','phash');

INSERT INTO "ADMIN"  VALUES ('USR_ADMIN');
INSERT INTO "STAFF"  VALUES ('USR_STAFF1','USR_ADMIN'),
                    ('USR_STAFF2','USR_ADMIN');
INSERT INTO "STUDENT"VALUES ('USR_STD1','USR_ADMIN'),
                    ('USR_STD2','USR_ADMIN');
```

2) Digital Card

```
INSERT INTO "DIGITALCARD"
 ("SUserID","Card_NO","Balance")
VALUES ('USR_STD1','CARD-ALICE-001',50),
     ('USR_STD2','CARD-BOB-002',25);
```

3) Route

```
INSERT INTO "ROUTE"("Route_ID","RouteName") VALUES
 ('ROUTE_AAAAAAAA-1111-1111-1111-111111111111','Campus Loop');

INSERT INTO
"STATION"("Station_ID","Station_Name","Station_Latitude","Station_Longitude")
VALUES
 ('STAT_A','Library',34.0522,-118.2437),
 ('STAT_B','Admin Bldg',34.0530,-118.2445);

INSERT INTO "HAS" VALUES
 ('ROUTE_AAAAAAAA-1111-1111-1111-111111111111','STAT_A',1),
 ('ROUTE_AAAAAAAA-1111-1111-1111-111111111111','STAT_B',2);
```

4) Food Data

```
INSERT INTO "DISH"("Dish_ID","Dish_Name","Calories") VALUES
 ('DISH_VEG','Vegan Burger',450);

INSERT INTO "MENU"("Menu_ID","Menu_Name","Price","MngUserID") VALUES
 ('MENU_001','Lunch Combo',12.99,'USR_STAFF1');

INSERT INTO "PART_OF" VALUES ('MENU_001','DISH_VEG');
```

5) Sales

```
INSERT INTO "SALES" VALUES ('MENU_001',CURRENT_DATE,5);
```

6) QR

```
INSERT INTO "QR"
  ("SUserID","Card_NO","QR_ID","Expired_Date","MMenu_ID")
VALUES ('USR_STD1','CARD-ALICE-001',uuid_generate_v4(),now()+interval '1
day','MENU_001');
```

7) Bus & Drive_in

```
INSERT INTO "BUS"("Vehicle_ID") VALUES ('BUS_01');
INSERT INTO "DRIVE_IN" VALUES ('BUS_01','ROUTE_AAAAAAAA-1111-1111-
1111-111111111111');
```

8) Appointment

```
INSERT INTO "APPOINTMENT"
 ("Appointment_ID","SUserID","MNGStaffID","AppointmentDate")
VALUES ('APPT_01','USR_STD2','USR_STAFF1',current_date+2);

INSERT INTO "SPORTAPPOINTMENT" VALUES
 ('APPT_01','Basketball','14:00','15:30');
```

9) Book

```
INSERT INTO "BOOK"
  VALUES ('ISBN123','SQL Basics','J.Doe',10,8);
```

10) BookAppointment

```
INSERT INTO "BOOKAPPOINTMENT"
  VALUES ('ISBN123','APPT_01',current_date+2,NULL,1);
```

==========

3) 10 Samples queries

==========

1) Students managed by the Admin (USR_ADMIN)

```
SELECT u."FName",u."LName"
FROM   "STUDENT" st
JOIN   "USER" u ON u."UserID"=st."UserID"
WHERE  st."MNGUserID"='USR_ADMIN';
```

2) Today's sales quantity and total revenue of a menu

```
SELECT m."Menu_Name",
    s."Num_Sold",
    s."Num_Sold"*m."Price" AS revenue
FROM   "SALES" s
JOIN   "MENU"  m ON m."Menu_ID"=s."MMenu_ID"
WHERE  s."Date" = CURRENT_DATE;
```

3) Stop names and their order on the Campus Loop route

```
SELECT st."Station_Name", h."Order"
FROM   "HAS" h
JOIN   "STATION" st ON st."Station_ID"=h."Station_ID"
WHERE  h."Route_ID"='ROUTE_AAAAAAAA-1111-1111-1111-111111111111'
ORDER  BY h."Order";
```

4) Books with stock below 20%

```
SELECT "Title","CurrentQuantity","QuantityInStock"
FROM   "BOOK"
WHERE  "CurrentQuantity" < 0.2*"QuantityInStock";
```

5) Active (non-expired) QR codes

```
SELECT "QR_ID","Expired_Date"
FROM   "QR"
WHERE  "Expired_Date" > now();
```

6) All upcoming appointments of the student

```
SELECT a."Appointment_ID",a."AppointmentDate"
FROM   "APPOINTMENT" a
WHERE  a."SUserID" = 'USR_STD1'
  AND  a."AppointmentDate" >= CURRENT_DATE;
```

7) Menus managed by a staff member

```
SELECT m."Menu_Name"
FROM   "MENU" m
WHERE  m."MngUserID" = 'USR_STAFF1';
```

8) Number of users who looked at the routes

```
SELECT r."RouteName", COUNT(l."UserID") AS viewer_count
FROM   "ROUTE" r
LEFT JOIN "LOOKS" l ON l."Route_ID" = r."Route_ID"
GROUP  BY r."Route_ID",r."RouteName";
```

9) Users with more than one favorite route

```
SELECT u."FName",u."LName", COUNT(*) AS fav_cnt
FROM   "FAVORITE_ROUTES" f
JOIN   "USER" u ON u."UserID" = f."UserID"
GROUP  BY u."UserID",u."FName",u."LName"
HAVING COUNT(*) > 1;
```

10) Today's departure times (ROUTE_DEPARTURE_TIMES)

```
SELECT r."RouteName",rdt."DepartureTime"
FROM   "ROUTE_DEPARTURE_TIMES" rdt
JOIN   "ROUTE" r ON r."Route_ID"=rdt."Route_ID"
WHERE rdt."DepartureTime"::date = CURRENT_DATE AND
rdt."DepartureTime"::time > CURRENT_TIME::time
ORDER  BY rdt."DepartureTime";
```

==========

Sample Queries Results

==========

1) Students managed by the Admin (USR_ADMIN):

Executing SQL: SELECT u."FName",u."LName" FROM "STUDENT" st JOIN "USER" u ON u."UserID"=st."UserID" WHERE st."MNGUserID"='USR_ADMIN';

```
Response: [
  { FName: 'Alice', LName: 'Wonder' },
  { FName: 'Bob', LName: 'Builder' },
  { FName: 'Sophia', LName: 'Taylor' },
  { FName: 'Lucas', LName: 'Wilson' },
]
```

2) Today's sales quantity and total revenue of a menu:

Executing SQL: SELECT m."Menu_Name", s."Num_Sold", s."Num_Sold"*m."Price" AS revenue FROM "SALES" s JOIN "MENU" m ON m."Menu_ID"=s."MMenu_ID" WHERE s."Date" = CURRENT_DATE;

```
Response: [
{ Menu_Name: 'Lunch Combo', Num_Sold: 5, revenue: 64.95 }
 ]
```

3) Stop names and their order on the Campus Loop route:

Executing SQL: SELECT st."Station_Name", h."Order" FROM "HAS" h JOIN "STATION" st ON st."Station_ID"=h."Station_ID" WHERE h."Route_ID"='ROUTE_AAAAAAAA-1111-1111-1111-111111111111' ORDER BY h."Order";

Response: [
 { Station_Name: 'Library', Order: 1 },
 { Station_Name: 'Admin Bldg', Order: 2 }
]

4) Books with stock below 20%:
Executing SQL: SELECT "Title","CurrentQuantity","QuantityInStock" FROM "BOOK" WHERE "CurrentQuantity" < 0.2*"QuantityInStock";

Response: [
 {
  Title: "Harry Potter and the Sorcerer's Stone",
  CurrentQuantity: 0,
  QuantityInStock: 40
 },
 { Title: 'Brave New World', CurrentQuantity: 0, QuantityInStock: 17 },
 {
  Title: 'The Catcher in the Rye',
  CurrentQuantity: 0,
  QuantityInStock: 40
 },
 {
  Title: 'The Lord of the Rings',
  CurrentQuantity: 2,
  QuantityInStock: 11
 }
]
5) Active (non-expired) QR codes:
Executing SQL: SELECT "QR_ID","Expired_Date" FROM "QR" WHERE "Expired_Date" > now();

Response: [
 {
  QR_ID: '78817611-f29e-483d-8878-15c9a77523f0',
  Expired_Date: 2025-05-21T06:44:58.317Z
 },
 {
  QR_ID: '6b1e1d95-626b-4d24-a7de-7162a39d4fe5',
  Expired_Date: 2025-06-08T06:44:58.317Z
 },
 {
  QR_ID: 'eaf18eee-deb8-4321-a27a-eb0564af2e15',
  Expired_Date: 2025-05-23T06:44:58.317Z
 },

```
 {
   QR_ID: 'e6688e9c-371b-4d86-8a99-3bc612def52b',
   Expired_Date: 2025-05-28T06:44:58.317Z
 },
 {
   QR_ID: '0acfc654-7a71-41a1-820b-39cfc3061364',
   Expired_Date: 2025-06-10T06:44:58.317Z
 },
 {
   QR_ID: '95239991-2738-4dff-aae9-525b2ecf87ee',
   Expired_Date: 2025-06-15T06:44:58.317Z
 },
]
```

6) All upcoming appointments of the student (USR_STD1):

```
Executing SQL: SELECT a."Appointment_ID",a."AppointmentDate" FROM
"APPOINTMENT" a WHERE a."SUserID" = 'USR_STD1' AND a."AppointmentDate"
>= CURRENT_DATE;

Response: [
 {
   Appointment_ID: '70fcc743-a733-435b-bdaa-d69941fa5355',
   AppointmentDate: 2025-06-10T00:00:00.000Z
 }
]
```

7) Menus managed by a staff member (USR_STAFF1):

```
Executing SQL: SELECT m."Menu_Name" FROM "MENU" m WHERE
m."MngUserID" = 'USR_STAFF1';

Response: [
 { Menu_Name: 'Lunch Combo' },
 { Menu_Name: 'Special Menu 4' },
 { Menu_Name: 'Special Menu 5' },
 { Menu_Name: 'Weekly Menu 7' },
 { Menu_Name: 'Weekly Menu 10' }
]
```

8) Number of users who looked at the routes:

Executing SQL: SELECT r."RouteName", COUNT(l."UserID") AS viewer_count FROM "ROUTE" r LEFT JOIN "LOOKS" l ON l."Route_ID" = r."Route_ID" GROUP BY r."Route_ID",r."RouteName";

Response: [
 { RouteName: 'Beachcomber', viewer_count: 6n },
 { RouteName: 'Campus Loop', viewer_count: 6n },
 { RouteName: 'Riverside Runner', viewer_count: 7n },
 { RouteName: 'Crosstown Shuttle', viewer_count: 9n },
 { RouteName: 'Mountain View', viewer_count: 6n },
 { RouteName: 'Tech Park Circulator', viewer_count: 7n }
]

9) Users with more than one favorite route:
Executing SQL: SELECT u."FName",u."LName", COUNT(*) AS fav_cnt FROM "FAVORITE_ROUTES" f JOIN "USER" u ON u."UserID" = f."UserID" GROUP BY u."UserID",u."FName",u."LName" HAVING COUNT(*) > 1;

Response: [
 { FName: 'Isabella', LName: 'Johnson', fav_cnt: 2n },
 { FName: 'Bob', LName: 'Builder', fav_cnt: 2n },
 { FName: 'Harper', LName: 'Davis', fav_cnt: 3n },
 { FName: 'Sophia', LName: 'Taylor', fav_cnt: 3n },
 { FName: 'Liam', LName: 'Gonzalez', fav_cnt: 3n },
 { FName: 'Mia', LName: 'Brown', fav_cnt: 3n }
]

10) Today's upcoming departure times:
Executing SQL: SELECT r."RouteName",rdt."DepartureTime" FROM "ROUTE_DEPARTURE_TIMES" rdt JOIN "ROUTE" r ON r."Route_ID"=rdt."Route_ID" WHERE rdt."DepartureTime"::date = CURRENT_DATE AND rdt."DepartureTime"::time > CURRENT_TIME::time ORDER BY rdt."DepartureTime";

Response: [
 {
  RouteName: 'Riverside Runner',
  DepartureTime: 2025-05-20T09:50:39.000Z
 },
 {
  RouteName: 'Tech Park Circulator',
  DepartureTime: 2025-05-20T10:12:45.000Z
 },

```
  {
    RouteName: 'Tech Park Circulator',
    DepartureTime: 2025-05-20T10:34:56.000Z
  },
  {
    RouteName: 'Mountain View',
    DepartureTime: 2025-05-20T10:46:23.000Z
  },
  { RouteName: 'Campus Loop', DepartureTime: 2025-05-20T13:12:00.000Z },
  {
    RouteName: 'Crosstown Shuttle',
    DepartureTime: 2025-05-20T15:08:26.000Z
  },
  {
    RouteName: 'Crosstown Shuttle',
    DepartureTime: 2025-05-20T16:43:05.000Z
  },
]
```

UI

Login/Signup Page

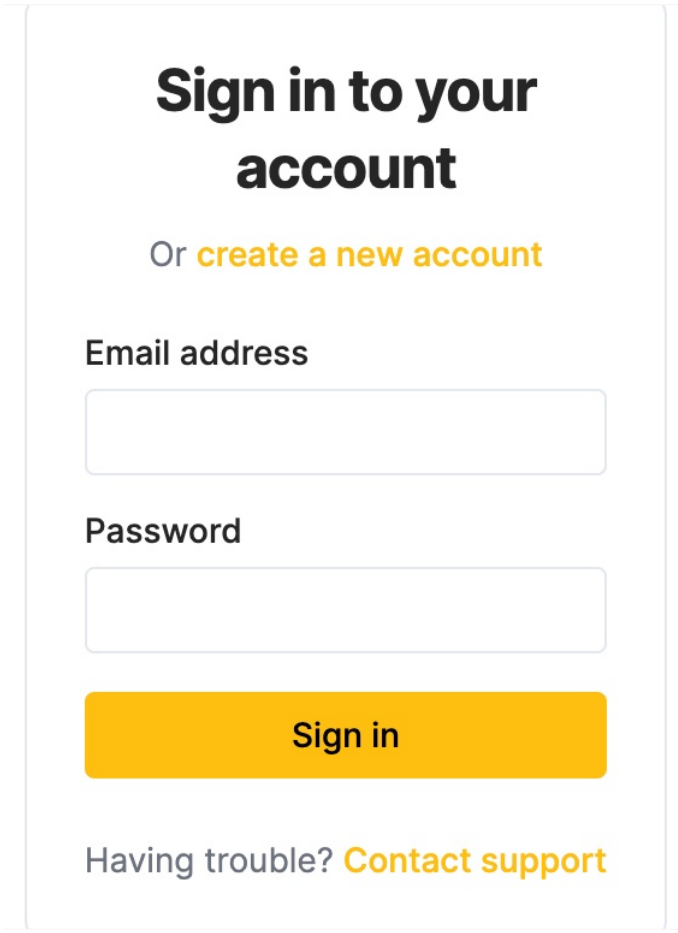


Image 1 - Sign In

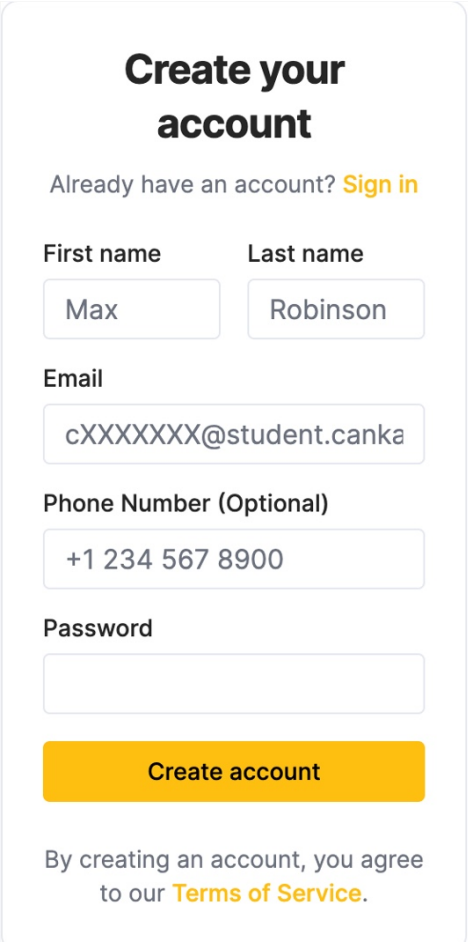


Image 2 - Register

Main Page

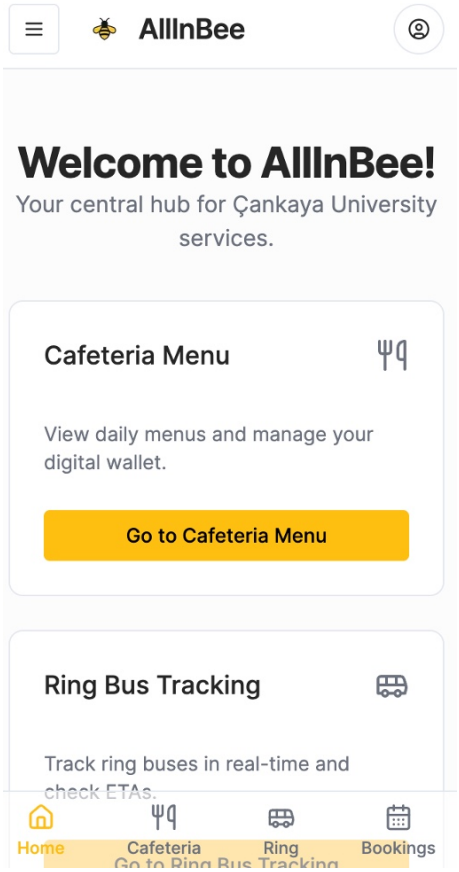


Image 3 - Cafeteria

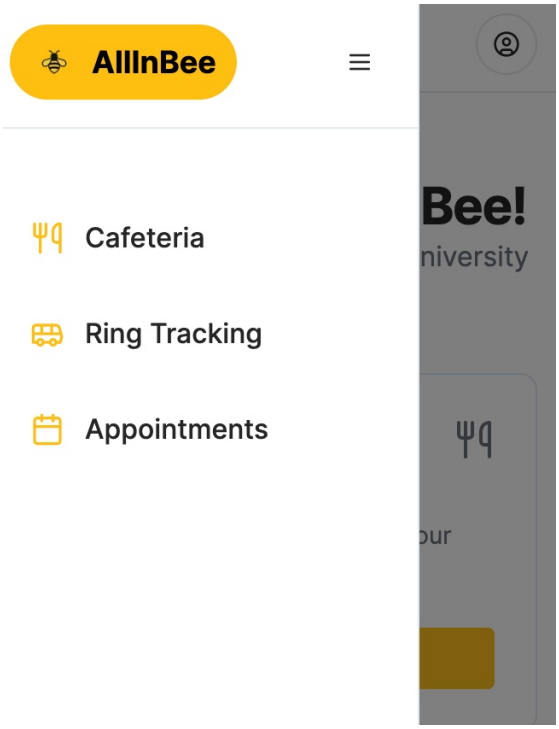


Image 4 - Navbar

Cafeteria

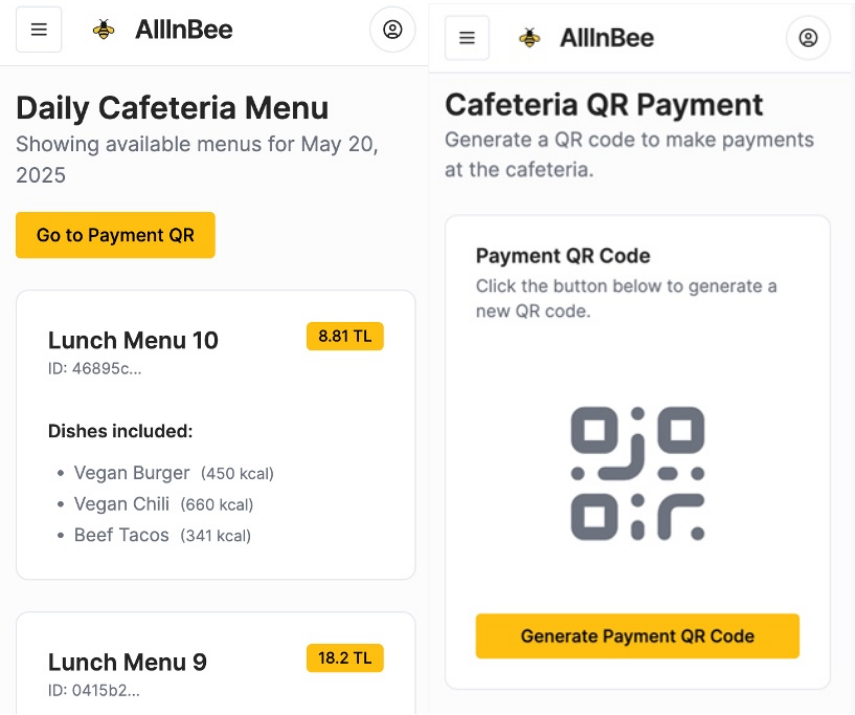


Image 5 - Cafeteria Menu Image 6 - Create QR

Digital Wallet (inside cafeteria)

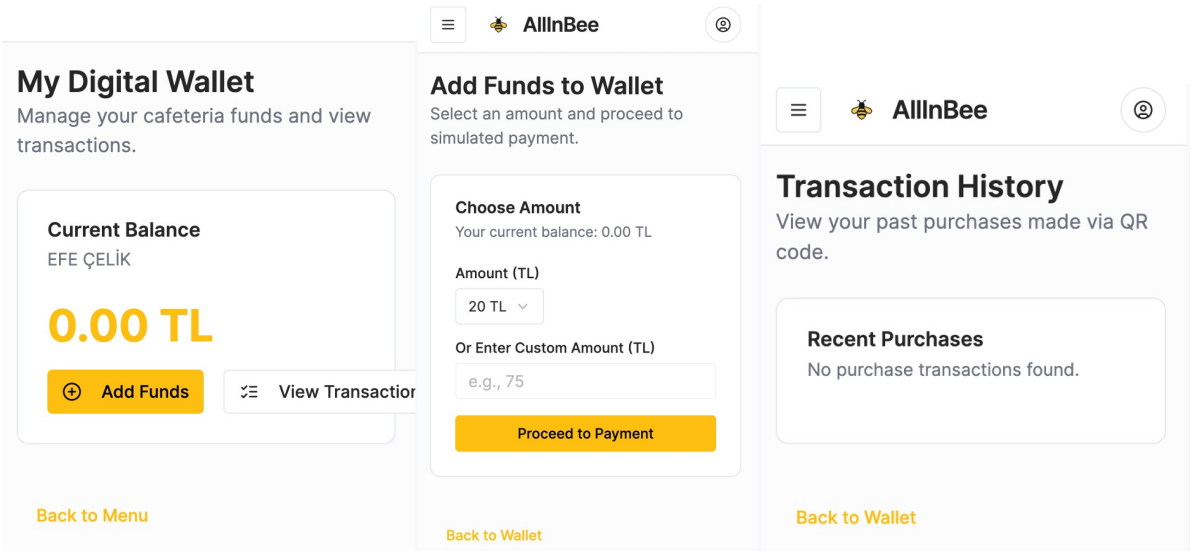
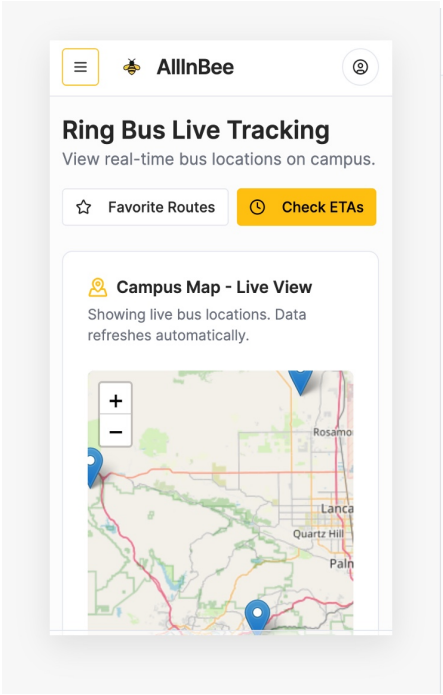


Image 7 - Digital Wallet Image 8 - Add funds Image 9 - Transaction History

Bus Tracking



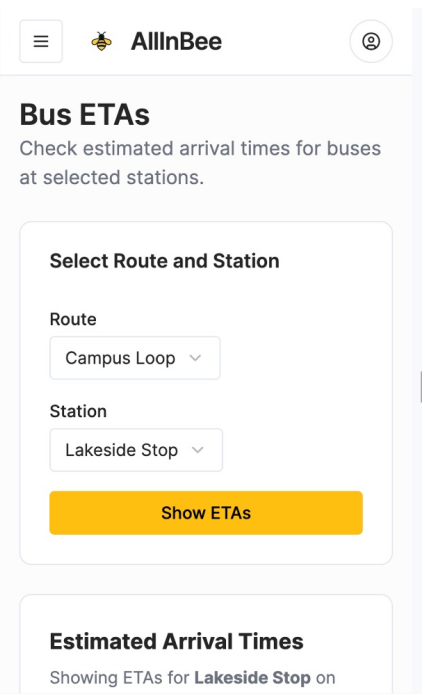


Image 10 - Live Location

Image 11 - ETA

Appointments

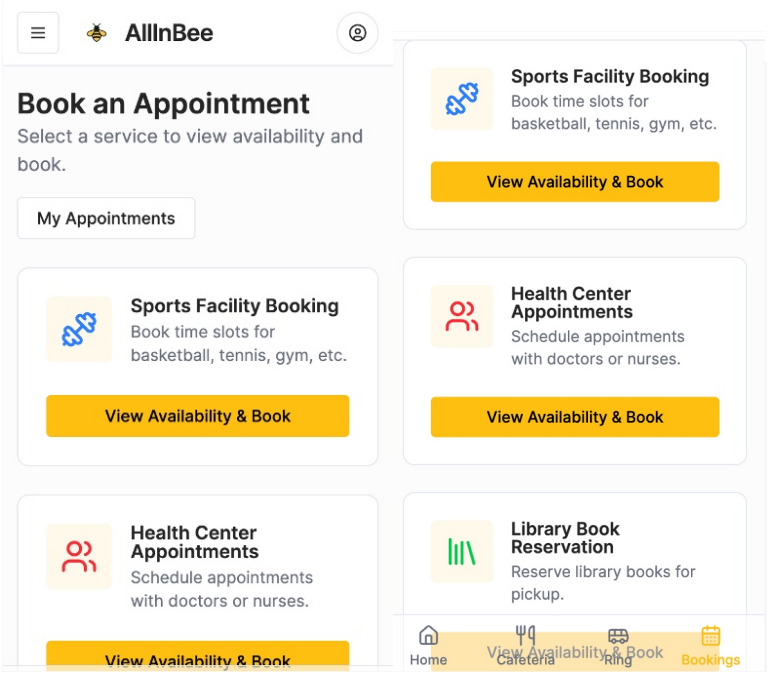


Image 12 - Book Appointment

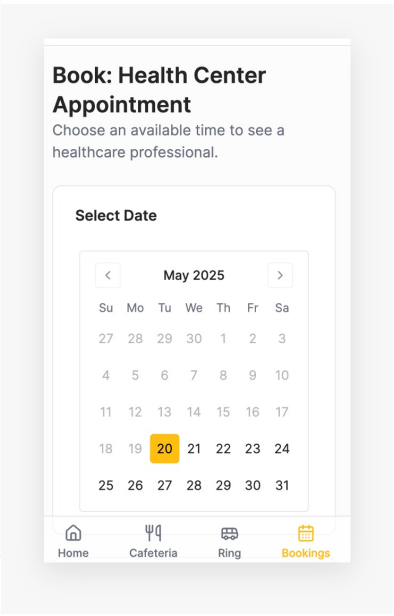


Image 13 - Book Appointment 2

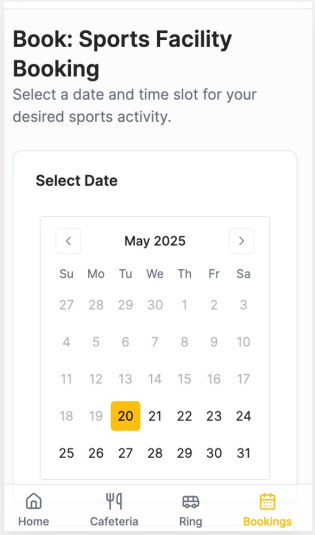


Image 14 - Date select

Image 15 - Sport Book

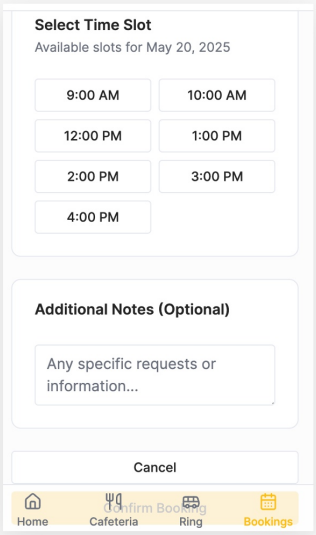


Image 16 - Select Time Slot

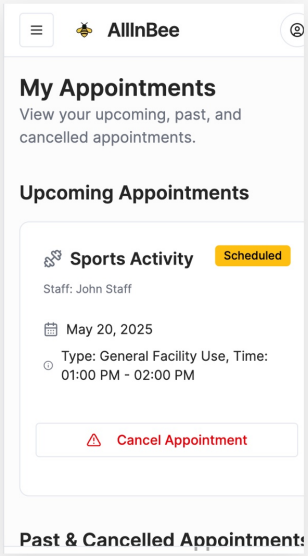


Image 17- My Appointments

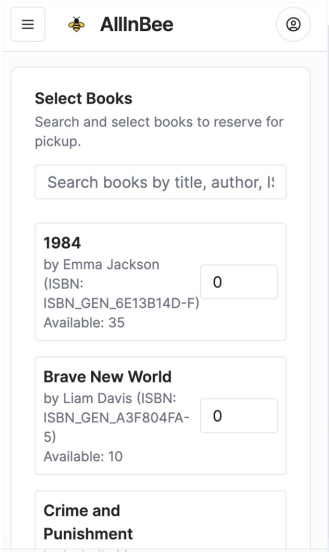


Image 18- Select Books

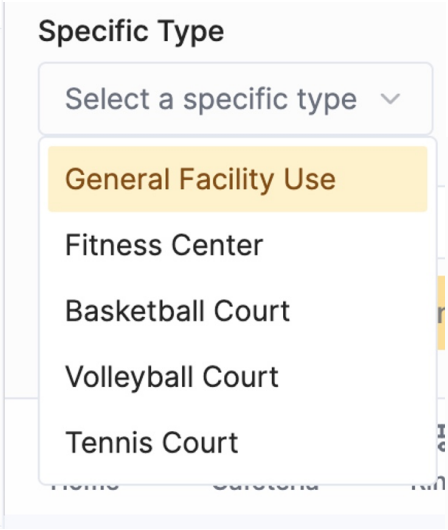


Image 19- Select Type

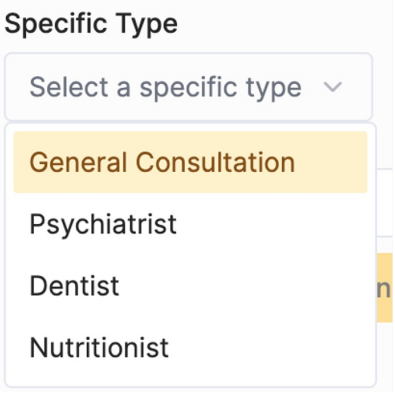


Image 20- Specific Type

Staff Page

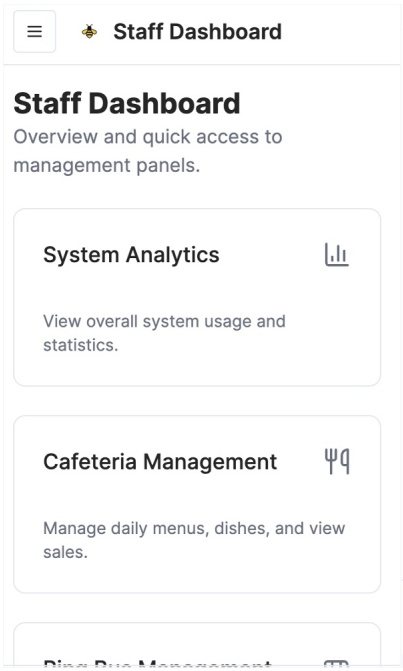


Image 21- Staff Dashboard

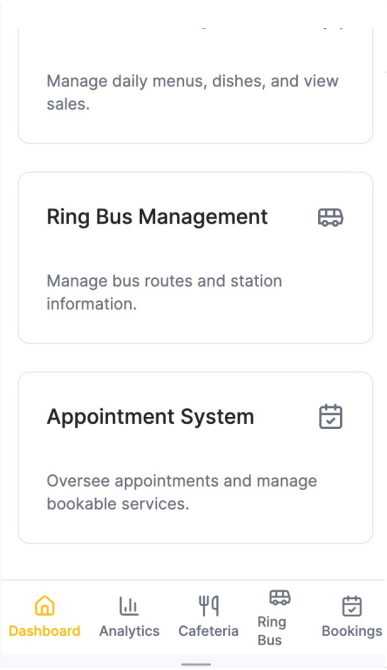


Image 22- Staff Dashboard 2

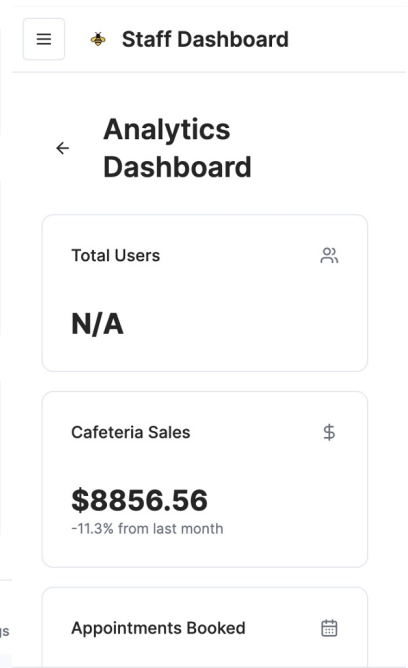


Image 23- Staff Analytics

Staff Manages Menu

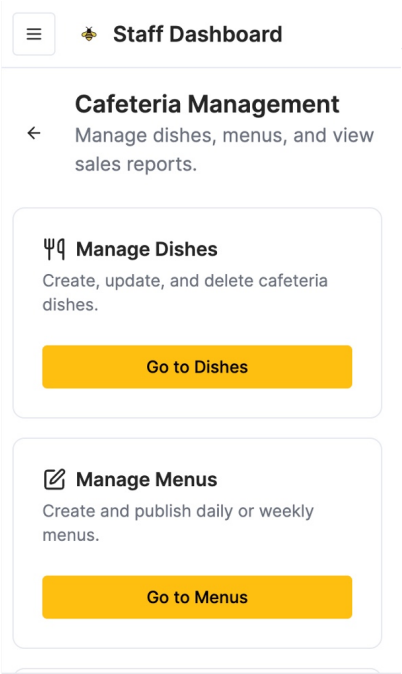


Image 24- Cafeteria Management

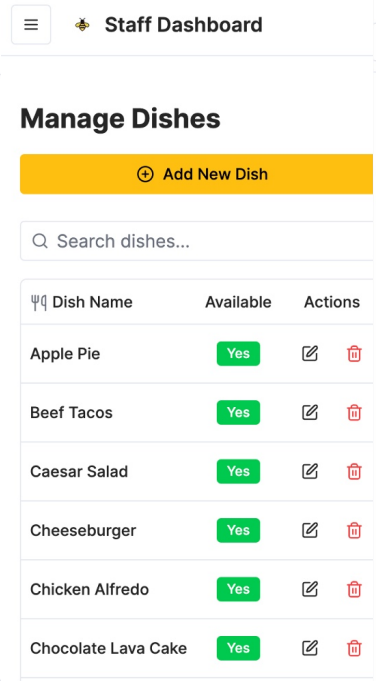


Image 25- Manage Dishes

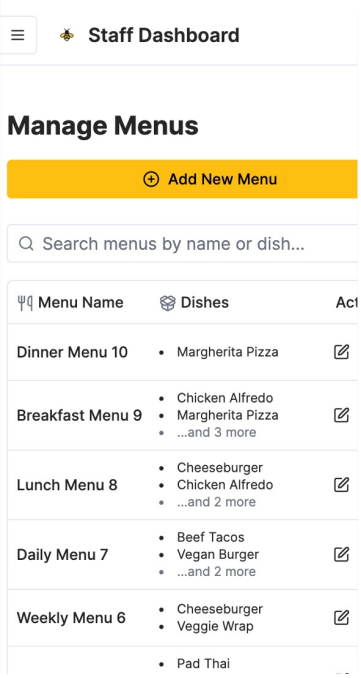
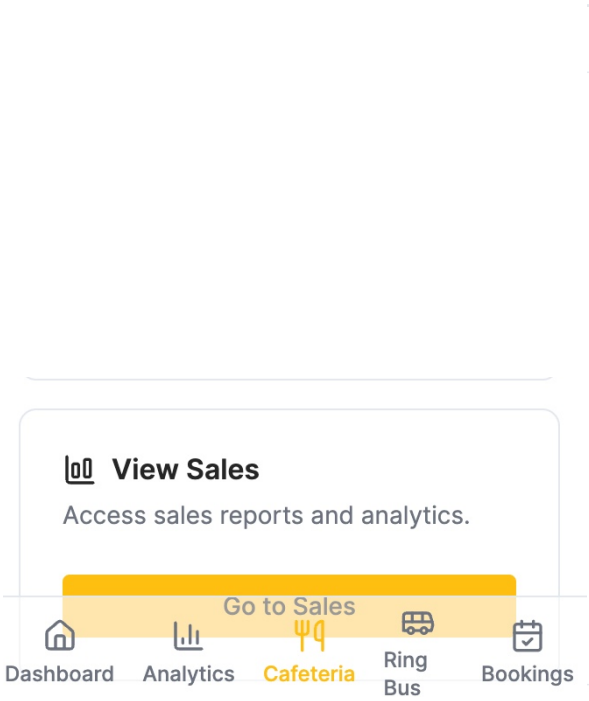


Image 26- Manage Menus

Staff Views Sales



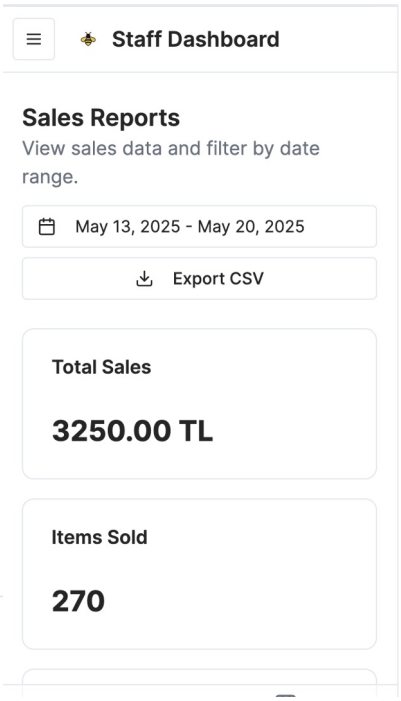
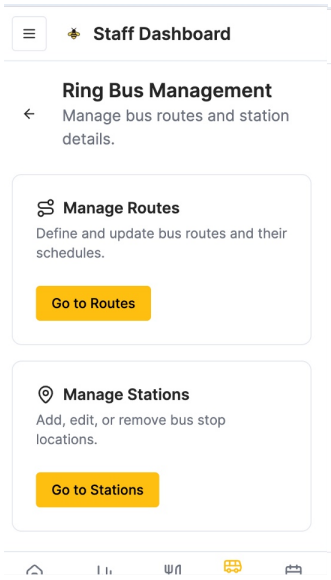
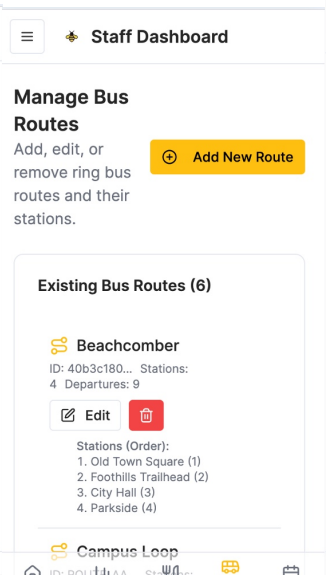


Image 27- View Sales

Image 28- Sales Report

Staff Manages Routes





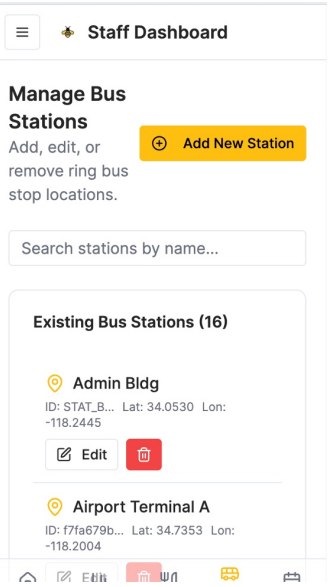


Image 29- Ring Bus

Image 30 - Manage bus Routes

Image 31 - Manage bus Routes 2

Staff Manages Appointments

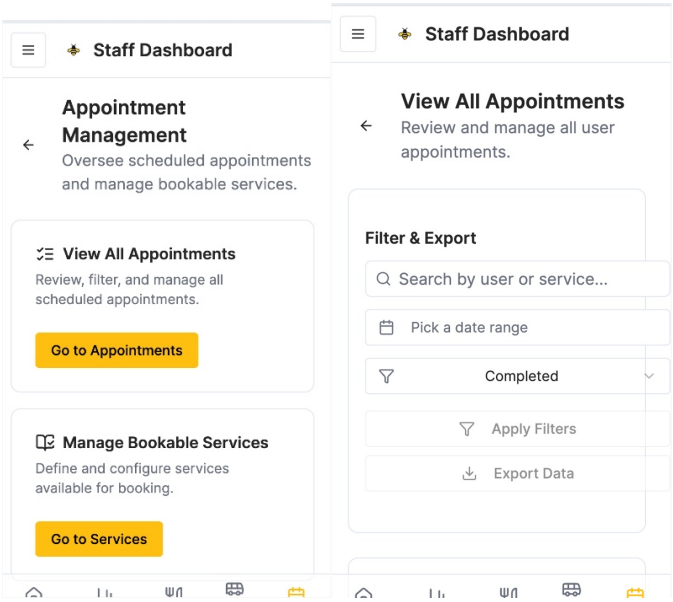


Image 32 -Appointment Management Image 33 - View all Appointments

Staff Manages Books

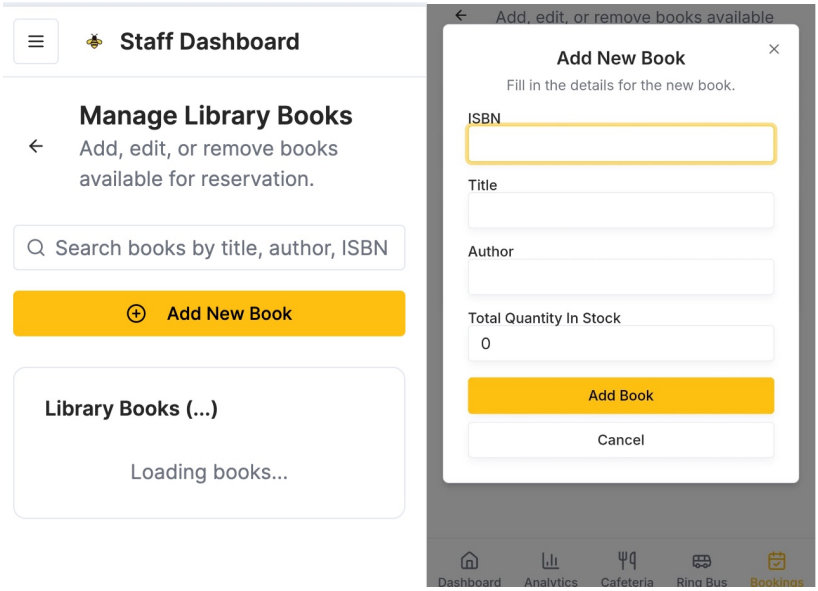


Image 34 - Manage Library Books Image 35 - Add new books

Admin Panel

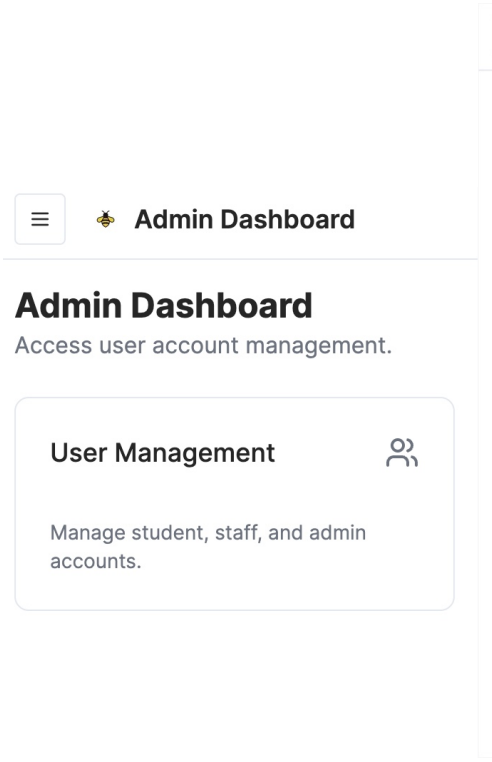


Image 36 - Admin Dashboard

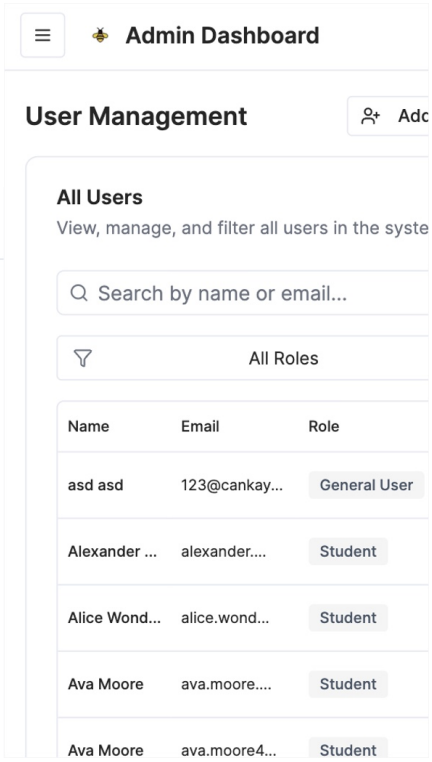


Image 37 - User Management

References

1) ÇAĞILTAY, N. E. (2024-2025). SENG306 - Database Modelling and Design: **introduction [Lecture Notes]**. Çankaya University, Department of Computer Engineering.

2) ÇAĞILTAY, N. E. (2024-2025). SENG306 - Database Modelling and Design: **database_DMS_DS [Lecture Notes]**. Çankaya University, Department of Computer Engineering.

3) ÇAĞILTAY, N. E. (2024-2025). SENG306 - Database Modelling and Design: Three Schema **Architecture [Lecture Notes]**. Çankaya University, Department of Computer Engineering.

4) ÇAĞILTAY, N. E. (2024-2025). SENG306 - Database Modelling and Design: **ERD [Lecture Notes].** Çankaya University, Department of Computer Engineering.

5) ÇAĞILTAY, N. E. (2024-2025). SENG306 - Database Modelling and Design: **weak entities [Lecture Notes].** Çankaya University, Department of Computer Engineering.

6) ÇAĞILTAY, N. E. (2024-2025). SENG306 - Database Modelling and Design: **EERD [Lecture Notes].** Çankaya University, Department of Computer Engineering.

7) ÇAĞILTAY, N. E. (2024-2025). SENG306 - Database Modelling and Design: **recursive relations [Lecture Notes].** Çankaya University, Department of Computer Engineering.

8) ÇAĞILTAY, N. E. (2024-2025). SENG306 - Database Modelling and Design: **logical model [Lecture Notes].** Çankaya University, Department of Computer Engineering.

9) ÇAĞILTAY, N. E. (2024-2025). SENG306 - Database Modelling and Design: **Physical Model [Lecture Notes]**. Çankaya University, Department of Computer Engineering.

10) ÇAĞILTAY, N. E. (2024-2025). SENG306 - Database Modelling and Design: **functional dependencies [Lecture Notes]**. Çankaya University, Department of Computer Engineering.

11) ÇAĞILTAY, N. E. (2024-2025). SENG306 - Database Modelling and Design: **Normalization_1 [Lecture Notes]**. Çankaya University, Department of Computer Engineering.

12) ÇAĞILTAY, N. E. (2024-2025). SENG306 - Database Modelling and Design: **SQL_1 [Lecture Notes].** Çankaya University, Department of Computer Engineering.

13) ÇAĞILTAY, N. E. (2024-2025). SENG306 - Database Modelling and Design: **2_joins[Lecture Notes]**. Çankaya University, Department of Computer Engineering.

14) ÇAĞILTAY, N. E. (2024-2025). SENG306 - Database Modelling and Design: 3**_subqueries [Lecture Notes].** Çankaya University, Department of Computer Engineering.

15) IEEE. (1998). **IEEE Std 830-1998**: IEEE Recommended Practice for Software Requirements Specifications.

Appendices

This section provides supplementary materials that support the main body of this Software Requirements Specification.

Appendix A: Conceptual Model

The AllInBee EER diagram is a conceptual data model detailing the relationships between user roles—student, staff, admin—and systems like cafeteria, bus tracking, appointments, and digital cards.

Appendix B: User Interface (UI) Images

The following images, presented as numbered images corresponding to their appearance in this document (pages 35-44), provide a visual representation of the AllInBee application's key screens and intended user flow.

- **Image 1 & 2 (from document page 35): Login/Signup Page**
    - **Image 1 - Sign In:** Allows existing users to log in using their email address and password. Includes links to "create a new account" and "Contact support."
    - **Image 2 - Register:** Enables new users to register by providing First Name, Last Name, Email (likely university-specific), and Password. An optional Phone Number field is included. A link to "Sign in" and a Terms of Service agreement notice are present.
- **Image 3 & 4 (from document page 36): Main Page**
    - **Image 3 - Welcome Screen:** Displays a welcome message and features primary service access cards for "Cafeteria Menu" and "Ring Bus Tracking," each with a "Go to..." button. A bottom navigation bar provides access to Home, Cafeteria, Ring (Bus), and Bookings.
    - **Image 4 - Navbar/Alternative Main View:** Shows a compact list of main services (Cafeteria, Ring Tracking, Appointments) accessible via a hamburger menu icon, representing an alternative navigation or a collapsed menu view.

- **Image 5 & 6 (from document page 37, top): Cafeteria**
    - **Image 5 - Cafeteria Menu:** Displays available menus for a specific date (e.g., May 20, 2025), showing menu names (e.g., Lunch Menu 10, Lunch Menu 9), prices, and included dishes with calorie information. A "Go to Payment QR" button is provided.
    - **Image 6 - Create QR (Cafeteria QR Payment):** Allows users to generate a QR code for cafeteria payments via a "Click the button below to generate a new QR code" prompt and displays the generated QR code upon clicking "Generate Payment QR Code."
- **Image 7, 8 & 9 (from document page 37, bottom): Digital Wallet (inside cafeteria)**
    - **Image 7 - Digital Wallet History:** Shows the user's current balance (e.g., EFE ÇELİK, 0.00 TL), with "Add Funds" and "View Transactions" options, and a "Back to Menu" link.
    - **Image 8 - Add funds:** Displays current balance and allows users to select a predefined (e.g., 20 TL) or enter a custom amount to add, with a "Proceed to Payment" button and "Back to Wallet" link.
    - **Image 9 - Transaction History:** Intended to list past QR code purchases (image shows "No purchase transactions found"), with a "Back to Wallet" link.
- **Image 10 & 11 (from document page 38): Bus Tracking**
    - **Image 10 - Live Location:** Features "Favorite Routes" and "Check ETAs" buttons, and a "Campus Map - Live View" showing real-time bus locations with a note that data refreshes automatically.
    - **Image 11 - ETA:** Allows users to select a Route (e.g., Campus Loop) and Station (e.g., Lakeside Stop) to view "Estimated Arrival Times" via a "Show ETAs" button.
- **Image 12, 13 & 14 (from document page 39, top): Appointments (Booking Overview)**
    - **Image 12 - Book Appointment (Main Selection):** Provides access to "My Appointments" and service selection cards for "Sports Facility Booking," "Health Center Appointments," and "Library Book Reservation," each with a "View Availability & Book" button.
    - **Image 13 - Book Appointment 2 (Alternative Service Selection):** Shows a similar service selection screen, possibly a different layout or entry point for booking Sports Facility, Health Center, or Library Book services.
    - **Image 14 - Date select (Health Center):** Presents a calendar interface to "Select Date" for a Health Center appointment.
- **Image 15, 16 & 17 (from document page 39, bottom): Appointments (Specific Booking & Viewing)**
    - **Image 15 - Sport Book (Sports Facility Date Selection):** Shows a calendar to "Select Date" for a sports facility booking.

    - **Image 16 - Select Time Slot:** After date selection, allows users to pick an "Available slot" (e.g., 9:00 AM, 10:00 AM) and add "Additional Notes (Optional)."
    - **Image 17 - My Appointments:** Displays "Upcoming Appointments" with details (e.g., service, staff, date, time) and a "Cancel Appointment" option, along with a section for "Past & Cancelled Appointments."
- **Image 18, 19 & 20 (from document page 40, top): Select Books & Specific Type Selection**
    - **Image 18 - Select Books (Library Reservation):** Features a search bar for books and lists available books with details (title, author, ISBN, availability) and a quantity input field.
    - **Image 19 - Select Type (General Facility):** Shows a dropdown to select a specific type for "General Facility Use" (e.g., Fitness Center, Basketball Court, Volleyball Court, Tennis Court).
    - **Image 20 - Specific Type (Health Consultation):** Shows a dropdown to select a specific type for "General Consultation" (e.g., Psychiatrist, Dentist, Nutritionist).
- **Image 21, 22 & 23 (from document page 40, bottom): Staff Page (Dashboard & Analytics)**
    - **Image 21 - Staff Dashboard (Overview):** Main staff dashboard providing quick access to management panels like System Analytics, Cafeteria Management, etc.
    - **Image 22 - Staff Dashboard 2 (Alternative Overview):** Another view of the staff dashboard, possibly with different emphasis, showing access to Ring Bus Management, Appointment System, etc., and a bottom navigation for Dashboard, Analytics, Cafeteria, Bus, Bookings.
    - **Image 23 - Staff Analytics:** Displays key metrics like "Total Users" (N/A in image), "Cafeteria Sales" (e.g., $8856.56), and "Appointments Booked."
- **Image 24, 25 & 26 (from document page 41): Staff Manages Menu**
    - **Image 24 - Cafeteria Management (Navigation):** Staff dashboard section for cafeteria, with options to "Manage Dishes" and "Manage Menus."
    - **Image 25 - Manage Dishes:** Interface for staff to "Add New Dish," search, and view/edit/delete existing dishes (e.g., Apple Pie, Beef Tacos), showing availability.
    - **Image 26 - Manage Menus:** Interface for staff to "Add New Menu," search, and view/edit/delete existing menus (e.g., Dinner Menu 10, Breakfast Menu 9), showing constituent dishes.
- **Image 27 & 28 (from document page 42, top): Staff Views Sales**
    - **Image 27 - View Sales (Navigation):** Staff dashboard entry point for accessing sales reports and analytics.

    - **Image 28 - Sales Report:** Displays "Sales Reports" with options to filter by date range, "Export CSV," and shows "Total Sales" (e.g., 3250.00 TL) and "Items Sold" (e.g., 270).

- **Image 29, 30 & 31 (from document page 42, bottom): Staff Manages Routes**
    - **Image 29 - Ring Bus Management (Navigation):** Staff dashboard section for ring bus system, with options to "Manage Routes" and "Manage Stations."
    - **Image 30 - Manage bus Routes:** Interface for staff to "Add New Route" and view/edit existing routes (e.g., Beachcomber, Campus Loop), showing route ID and associated stations.
    - **Image 31 - Manage bus Routes 2 (Manage Bus Stations):** Interface for staff to "Add New Station" and view/edit existing bus stations (e.g., Admin Bldg, Airport Terminal A), showing station ID and location details.
- **Image 32 & 33 (from document page 43, top): Staff Manages Appointments**
    - **Image 32 - Appointment Management (Navigation):** Staff dashboard section for appointments, with options to "View All Appointments" and "Manage Bookable Services."
    - **Image 33 - View all Appointments:** Interface for staff to review and manage all user appointments, with filter and export options (e.g., search by user/service, pick date range, filter by status like "Completed").
- **Image 34 & 35 (from document page 43, bottom): Staff Manages Books**
    - **Image 34 - Manage Library Books:** Staff interface to "Add New Book," search books by title/author/ISBN, and view a list of library books (image shows "Loading books...").
    - **Image 35 - Add new books (Modal):** A form for staff to input ISBN, Title, Author, and "Total Quantity In Stock" to add a new book to the library.
- **Image 36 & 37 (from document page 44): Admin Panel**
    - **Image 36 - Admin dashboard (Overview):** Main admin dashboard providing access to user account management.
    - **Image 37 - User Management:** Admin interface to view, manage, and filter all users in the system. Features include searching by name/email, filtering by roles, and a list of users with their name, email, and role (e.g., General User, Student).

Appendix C: Live Demo and Source Code Links

- **Live Demo:**
  A functional demonstration of the AllInBee application can be accessed at:
  https://allinbee.bayburt.lu/
- **Source Code (GitHub):**
  The complete source code for the project, including backend logic and database schema, is publicly available at:
  https://github.com/byigitt/seng306-allinbee