SENG306 - Database Modelling and Design

Software Requirements Specification

Version 1.3

27.04.2025

Project: AllInBee

Team Members

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

Instructor: Prof. Dr. Nergiz ÇAĞILTAY

Spring 2024/2025

Contents

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
28.04.2025	Version 1.3	Team AllInBee	Added data requirements

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.
- **Appendices:** Includes a glossary reflecting the new data model and the EER Diagram (v1.2).

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

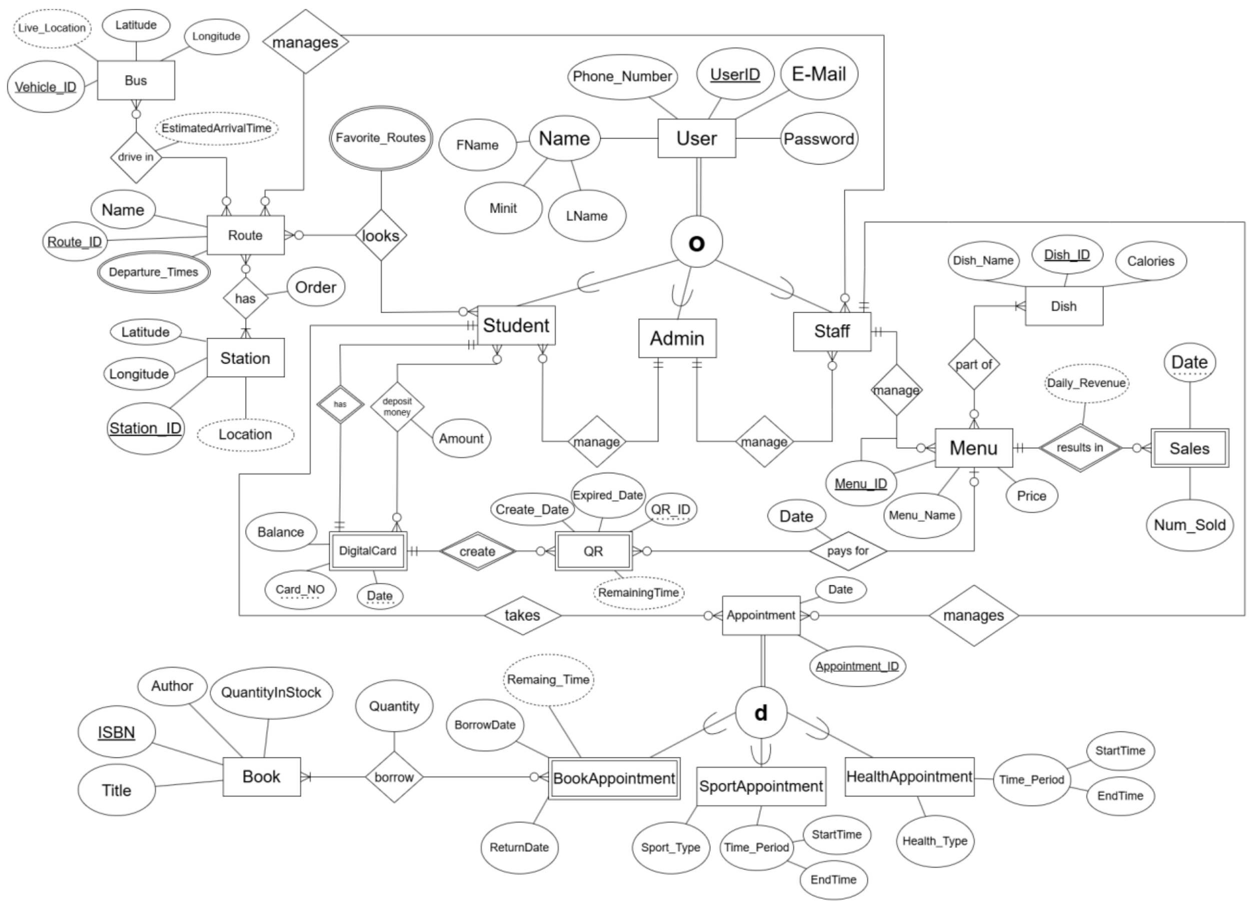
- **NFR-DATA-1:** Relationship: **drive in** (between **Bus** and **Route**)
  - A **Bus** can drive in zero, one, or many **Routes** (M:N).
  - A **Route** can be driven by zero, one, or many **Buses** (M:N).
  - Participation of **Bus** in the 'drive in' relationship is **optional** (Partial).
  - 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 zero, 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 zero, one, or many **Routes** (M:N).
    - A **Route** can be managed at by zero, 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 zero, one, or many **Routes** (M:N).
    - A **Route** can be looked at by zero, 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 zero, 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 zero, 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 zero, 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 zero, 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 zero, 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 zero, one, or many **Menu** entries (1:N).
    - A **Menu** entry can contain zero, 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 zero, 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 zero, 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 zero, 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 zero, 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 zero, 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).

Models


Live_Location
Latitude
Longitude
Vehicle_ID
Bus
manages
EstimatedArrivalTime
drive in
Favorite_Routes
Phone_Number
UserID
E-Mail
FName
Name
User
Password
Minit
LName
Name
Route_ID
Route
looks
Departure_Times
o
has
Order
Dish_Name
Dish_ID
Calories
Dish
Student
Admin
Staff
Latitude
Station
Longitude
part of
Date
manage
Daily_Revenue
has
deposit money
Station_ID
Location
Amount
manage
manage
Menu
results in
Sales
Menu_ID
Price
Menu_Name
Num_Sold
Expired_Date
Create_Date
QR_ID
Date
Balance
DigitalCard
create
QR
pays for
Card_NO
Date
RemainingTime
Date
takes
Appointment
manages
Appointment_ID
Author
QuantityInStock
Remaing_Time
Quantity
BorrowDate
d
ISBN
Book
borrow
BookAppointment
SportAppointment
HealthAppointment
Time_Period
StartTime
EndTime
Title
ReturnDate
Sport_Type
Time_Period
StartTime
EndTime
Health_Type