Bilkent University

Department of Computer Engineering

CS319 Term Project

Section 1

Group 1A - "lobby"

Project short-name: Pandemica

Design Report

Group Members:

- Efe Beydoğan (21901548)
- Arda Önal (21903350)
- Mert Barkın Er (21901645)
- Emir Melih Erdem (21903100)
- Gökberk Beydemir (21902638)
- Eren Polat (21902615)

1. Introduction

Purpose of the System

Our project a is designed as a software that will ease everyone's lives in the university during difficult pandemic times. The project will be a medium for students, lecturers, TAs and health personnel to get together during rough times to inform each other of the latest developments regarding the pandemic and take precautions together. The web-based application will provide easy access to resources a person may need during tough pandemic times. These resources include the personal information of the user, as well as weekly reports, guidelines specified by the university, important announcements, statuses of other people in the user's registered courses, and much more. Also, by adding a feedback form we are hoping to receive feedback from the users about the app and increase service quality. The project will include a request form, which users can fill out and send to the university to make suggestions on pandemic related precautions (addition of disinfectants to certain places etc.). Our goal when developing this product is to save the inhabitants of the university from the hassle of finding every bit of information separately on the internet, and let everyone have easy access to crucial information.

Design Goals

The application is supposed to be a user-friendly system, with an easy to use interface. Our goal when developing this software is to ensure it's usability, reliability, security, supportability and maintainability. Also, as we have stated in our Analysis Report, there are a number of performance constraints we strive to fulfill, in order to deliver the best experience for the users of our application. Top two design goals we value the most are usability and security.

User-Friendliness / Usability

We want to make sure that even users who aren't familiar with using computers can find their way around our website, so we will make our app as usable as possible. To this end, we will try our best to let any user who knows how to read and write use the system without a user manual. All the buttons will be aligned and have self-explanatory titles. Every button that is visible to the user will be clickable.

Reliability / Security

The information that is displayed on the site should always be correct and a system crash should not result in any data loss to ensure the reliability of the system. Also, reports that the user generates should not be lost due to internet related issues etc. The system should not crash in input errors and warn the user that the input is incorrect.

The system will have confidential information of every user such as passwords, phone numbers, Bilkent ids, email etc. and this information should not be accessible to anyone who tries to obtain them. Any functionality of the website should work regardless of cybersecurity attacks.

Maintainability / Supportability

The website should be supported for Google Chrome version 68.x or higher, Opera version 80.x or higher and Microsoft Edge version 93.x or higher. The locations of labels and buttons should not change for every operating system and browser. The website should work on devices that have 2 GB RAM or higher. The database should be able to store data that is up to 1 T.

We will implement the system as neatly as possible to make it supportable, so future developers can easily build up on it and add new features. We aim for any changes to be easily added after deployment.

Performance

We will try to make the website as efficient as possible, performance wise. Thus, the users will be able to use it without running into problems very often and they will hopefully have a smooth experience.

Some performance constraints include:

- The system website should load in less than 2 seconds.
- Every user interaction should be processed and if there is output, the result should be displayed in less than 2 seconds.
- The time that takes for the website to retrieve information from the database should be less than 1 second.
- The HES code should be revalidated every 30 seconds.
- If there is a change in the covid status of a user, the change should be represented in less than 2 seconds.

- The course and section information should be validated every 30 seconds and if there is any change, it should be displayed in less than 2 seconds.
- The entire system data should be backed up every hour.

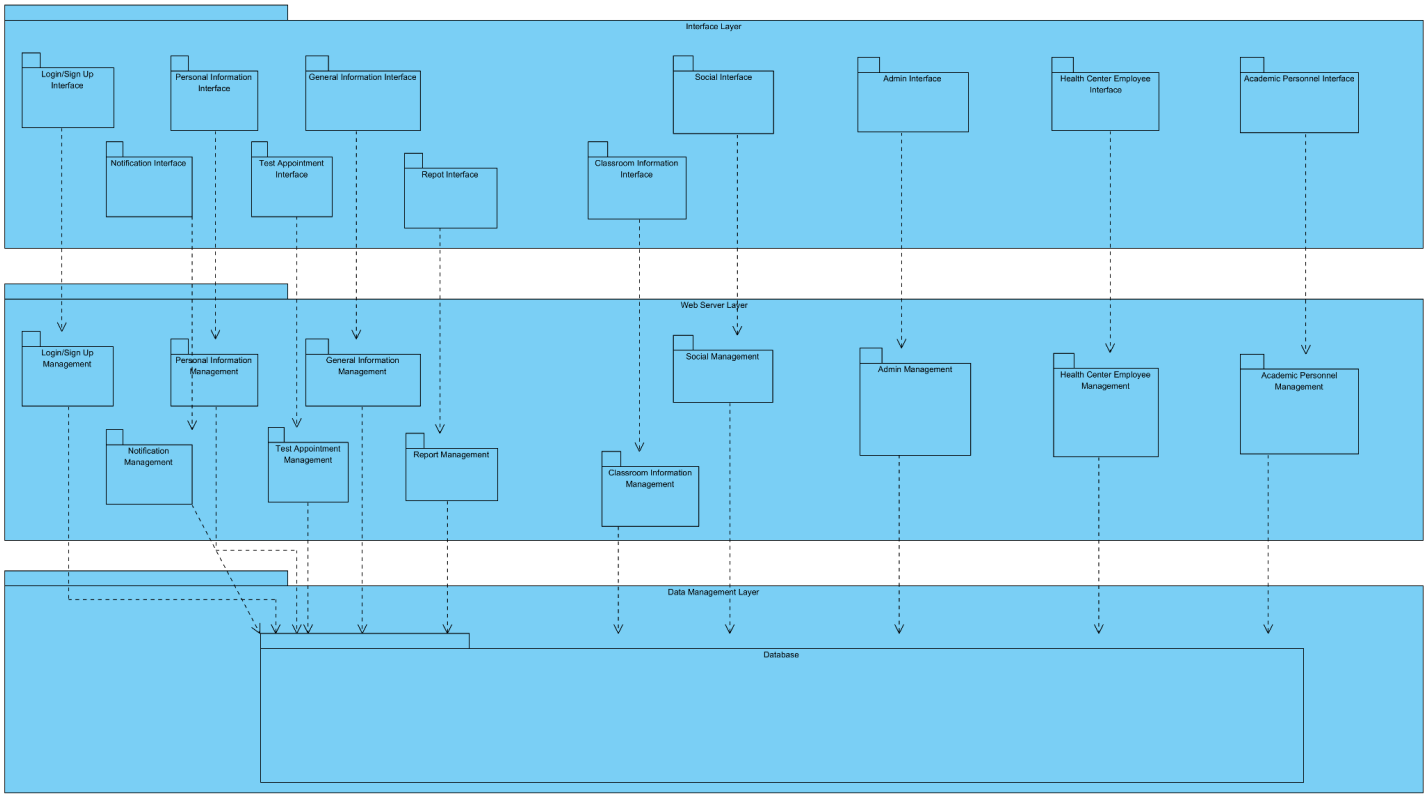

Figure 2.1: Subsystem Decomposition Diagram

2. High Level Software Architecture

Subsystem Decomposition

In the given Subsystem Decomposition Diagram, “Interface Layer” corresponds to the User Interface of our website. This layer constitutes the boundary objects of our project. All of the systems in this layer depict the UI elements users can interact with. We have decided these systems based on our use cases and functionalities we intend on offering to the users of our software. All of these UI elements will be designed according to our

“User-Friendliness / Usability” design goal, and we will make sure they meet the standards for easy usage.

The middle “Web Server Layer” contains all the systems for managing systems in the “Interface Layer”. All of these operations will be done in the backend. For example, the “Personal Information Management” system will fetch the information of the user from the database in order to be displayed correctly in the frontend to the user. All of the systems in this layer access the database to fetch information for the users of the software and fulfill their needs. The separation of core functionalities allow for a maintainable and supportable system so version changes and/or developer changes are more easily accommodated.

The final and bottom layer is the “Data Management Layer”. The database will be used to store every important information such as login credentials, vaccination cards, user information etc. This helps with our Performance and Reliability/Security criteria since the database lets us securely store data and quickly fetch it whenever needed.

Deployment Diagram

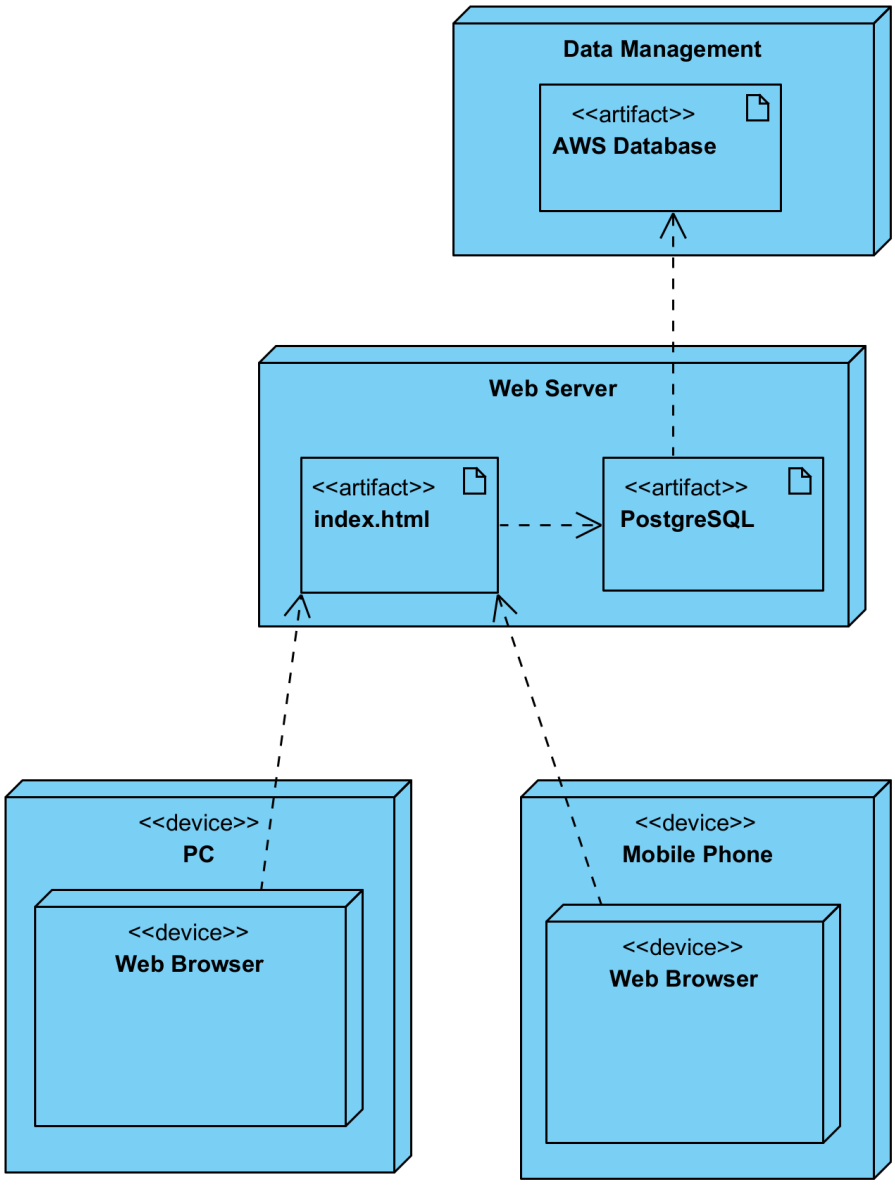

Figure 2.2: Deployment Diagram

The PC and Mobile Phone nodes use the following subsystems:

- Login/Sign Up Interface
- Notification Interface
- Personal Information Interface
- Test Appointment Interface
- General Information Interface
- Report Interface
- Classroom Information Interface

- Social Interface
- Admin Interface
- Health Center Employee Interface
- Academic Personnel Interface

Web Server node uses the following subsystems:

- Login/Sign Up Management
- Notification Management
- Personal Information Management
- Test Appointment Management
- General Information Management
- Report Management
- Classroom Information Management
- Social Management
- Admin Management
- Health Center Management
- Academic Personnel Management

Data Management node uses the following subsystems:

- Database

Hardware/Software Mapping

This project is web based which means that supported web browsers are required to use the website. The supported web browsers are Google Chrome version 68.x or higher, Opera version 80.x, Safari 10.x higher or higher and Microsoft Edge version 93.x or higher as indicated in the Analysis Report. Any device that supports these browsers will be sufficient to use the software. Hence, our software does not distinguish between computers and mobile phones which allows the software to be used more easily by providing more device types that the software could be run on.

Persistent Data Management

In the implementation of the project, PostgreSQL will be used because some of our team members worked as a backend software engineer part-time in a company and they are the most knowledgeable team members about

databases. They are going to lead the backend team and this was his decision. In the database, there will be tables for entities such as students, staff, securities, academic personnel, medical employees, admins, classroom plans, courses, friend requests, neighbors, forms, examination reports, vaccination information, test results, announcements and guidelines. There will also be relational tables that are indicated in the class diagram with many to many relationships. Each table will store the attributes of each entity in the class diagram as database attributes in their table. The primary keys of tables will be selected from their attributes such that it uniquely identifies every tuple in the table for example, the id attribute of the different users etc. Every subsystem in the web server layer will interact with the database subsystem that is located in the data management layer.

Access Control and Security

We promise absolute security and reliability in our software. For example, the data that is displayed to the users should always be correct and dependable data, based on official numbers announced by the university. Also, users' personal information will be under protection through the database. In order to ensure access control, we have different roles in our site and all of these roles are able to access different functionalities.

All of the user types are able to login/sign up, view their notifications, personal information, test appointments and general information regarding the situation of the pandemic in the university. Also, every user type is allowed to send feedback, violation and request reports.

In addition, student roles have access to the social and classroom interfaces. Students are able to add their friends on the site to view their friends' situations as well. Also, students have the ability to enter the seat selection interface, where they can select the seats they are sitting at in every one of their sections. The system will automatically determine their neighbors using this information, and the students will be saved from the hassle of learning the names/ids of all of their neighbors in their classrooms.

Admins have the access to the "Admin Interface" as we put it. This interface includes the functionalities of sending notifications, viewing every user's info, editing the general information about the university such as case numbers etc. and viewing every report created by other users and reviewing them.

Health Center employees can view an interface specifically designed for them, where they can view test appointments made by users, enter test

results, upload physical examination reports and manually update users' risk statuses if need be.

Finally, Academic Personnel also have access to a special interface where they can view their sections' information, such as who is allowed in the class or not, as well as seating plans according to the input from the students when they select which seat they are sitting at.

Boundary Conditions

Initialization

Since this project is a web based application, it doesn't require any installation. Only a browser and an active internet connection are needed in order to view the site. However, users who aren't registered cannot use any functionality offered by the site. To access functionalities, users must have a valid account, authorized through the database.

Termination

The website subsystems work altogether which means that if a single subsystem is terminated, the whole application terminates. If an admin terminates any subsystem, all data that is being used in that subsystem created by the users are saved to the database.

Failure

The intended failures caused by the developers will return different HTTP codes depending on the situation. Front-end will show a pop-up describing the situation and why the operation failed (e.g. "This student is already your friend so you cannot send a friend request to them."). Our project will be hosted on AWS, the unintended failures will be handled by AWS.

3. Low-Level Design

Object Design Trade-Offs

Maintainability versus Performance

The application will be coded using OOP, therefore will have objects for almost every portion of the project. While this will make performance a bit lower, maintainability will be better since developers will be able to make changes easier, saving developers hours that they would spend trying to understand the system.

Memory versus Performance

Since the project will be coded using OOP, there will be a lot of memory usage caused by the amount of classes. However, AWS has memory optimization that lowers the memory usage which we will be using, increasing our memory optimization at the cost of performance.

Security versus Usability

The application will use JSON Web Tokens for authorization to preauthorize functionalities based on the user's type. The tokens will be generated per user when that user logs in, and it will be stored in the local storage and sent whenever a user tries to use a functionality. This increases security, making sure that the users can only do things they are allowed to.

Functionality versus Usability

Our project is supposed to be used by everyone in Bilkent University, ranging from the professors to the non-academic staff. This forces the project to have many functionalities that are not available for everyone since there are many user types with many different user authorities, which decreases the usability of the software.

Layers

User Interface Layer

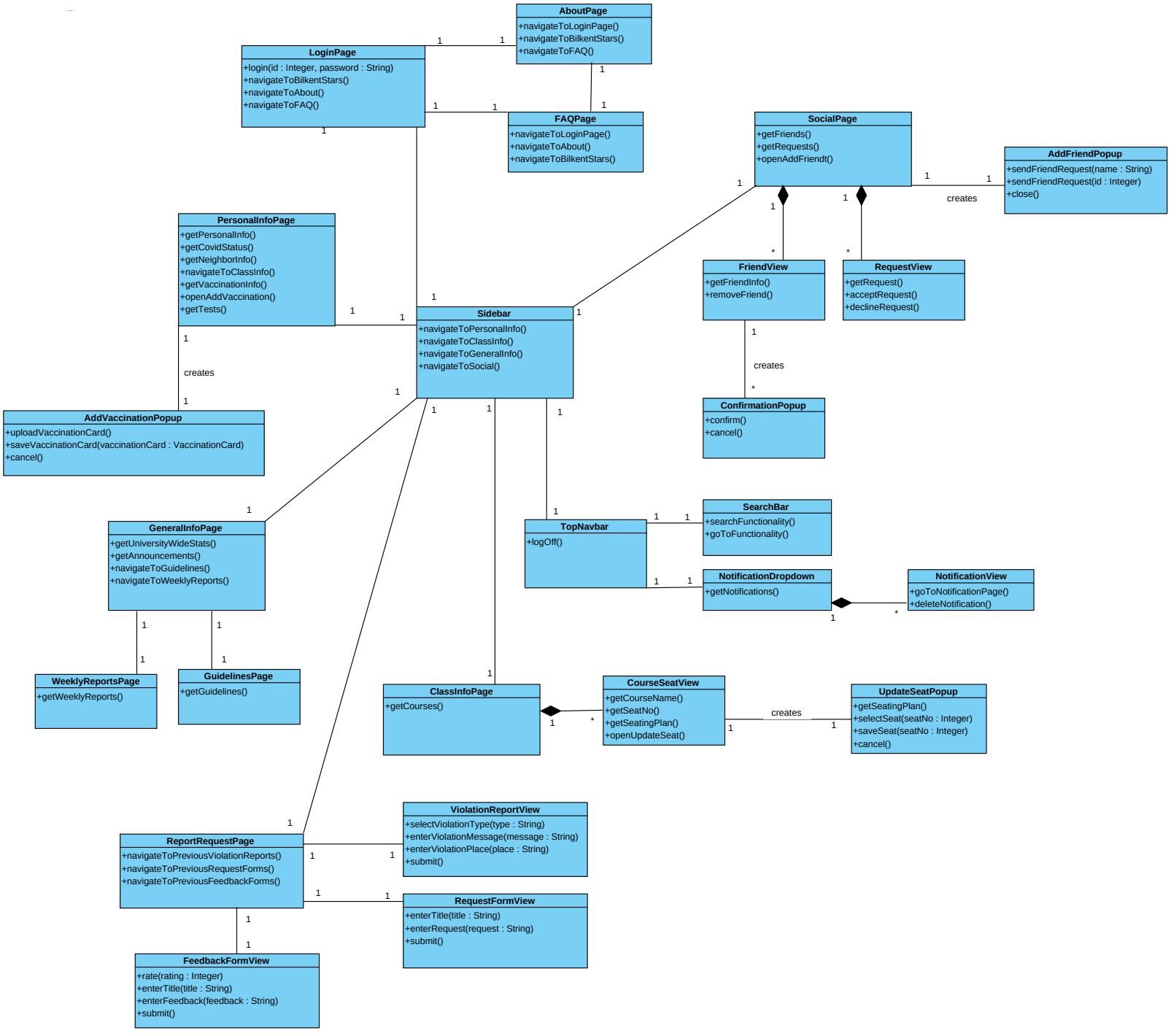

Figure 3.1: User Interface Layer Class Diagram

Web Server Layer

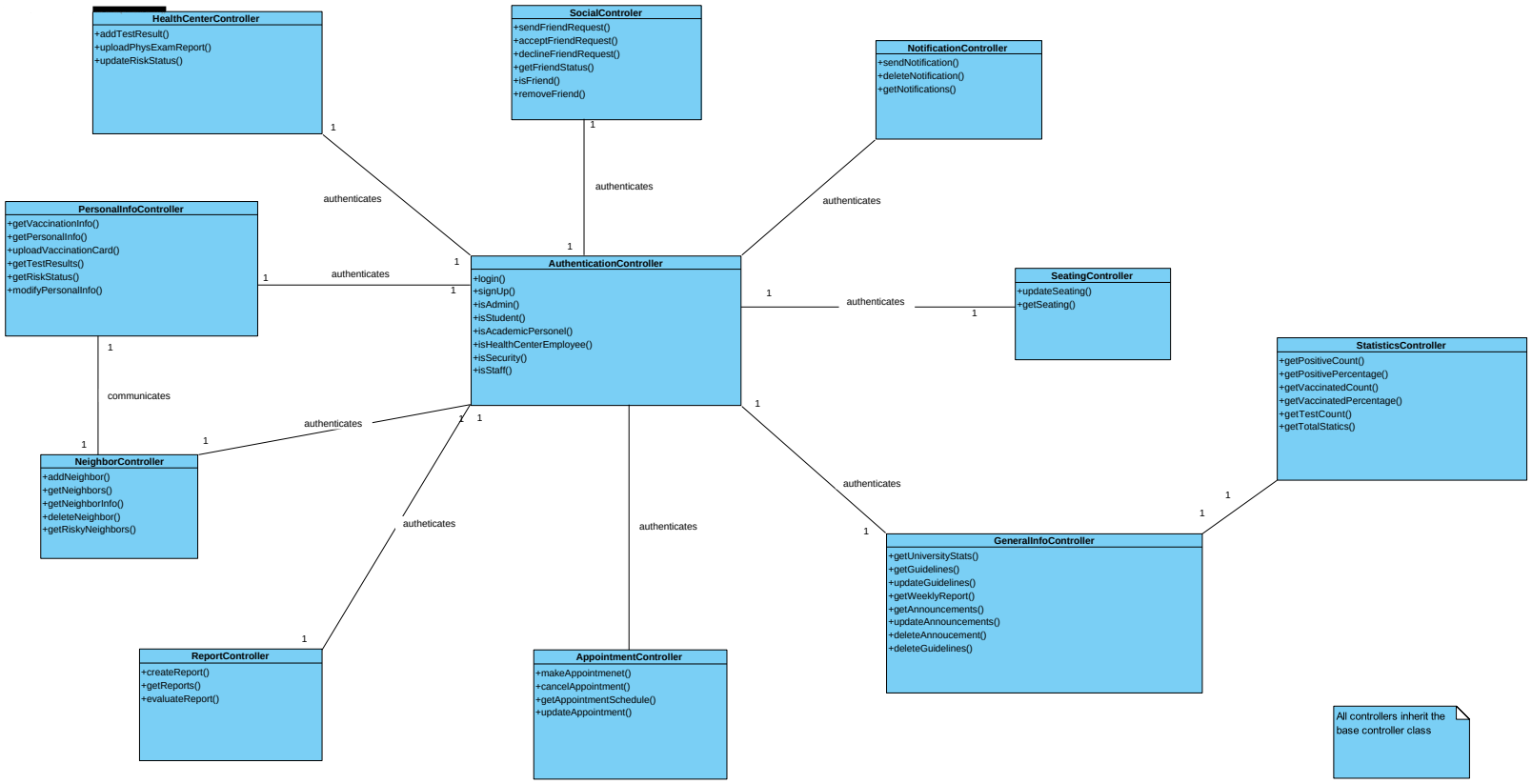

Figure 3.2: Web Server Layer Class Diagram

Database Layer Entity Class Diagram

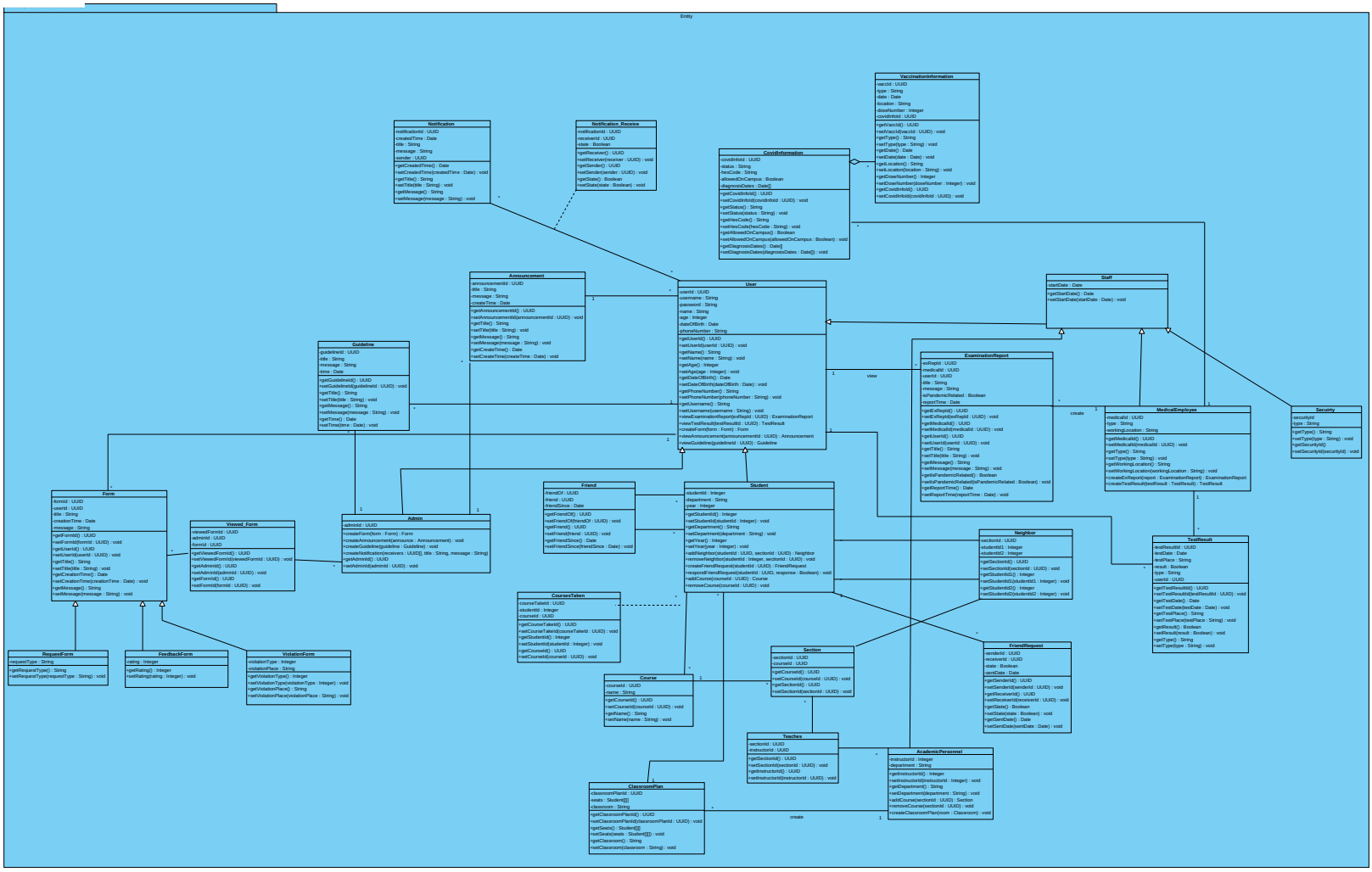

Figure 3.3: Database Layer Entity Class Diagram

The classes in the entity class diagram represent the database tables and relations.

Database Layer Repository Class Diagram

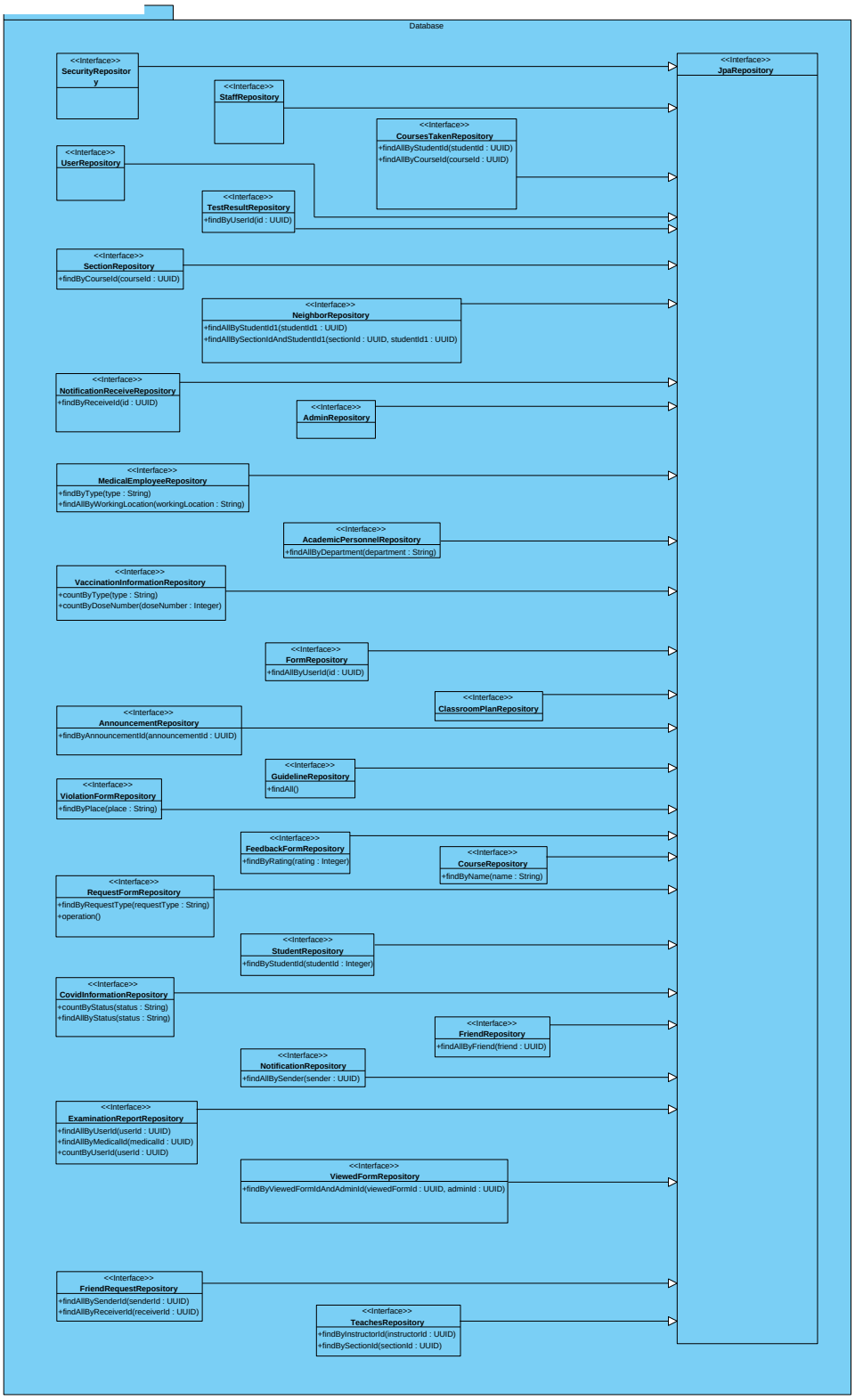

Figure 3.4: Database Layer Repository Class Diagram

Packages and Frameworks

1. jQuery
2. Bootstrap
3. Angular Framework
4. Java Spring
    a. springframework.data.jpa
    b. springframework.security
    c. springframework.postgresql
    d. springframework.jdbc