Cs319 Term Project

Project Design Report

Gym Management System

Group Members

- Servet Gülnaroğlu | 21902474
- Efe Kerem Kesgin | 21902857
- Khashayar Amini | 21903613
- Cemil Mert Özdemir | 21803303
- Efe Karaköyülü | 21901510

Instructor

- Eray Tüzün

Teaching Assistants

- Elgun Jabrayilzade
- Muhammad Umair Ahmed

1.Introduction **2**
1.1 Purpose of the system 2
1.2 Design Goals 2
1.2.1 Security and Privacy 2
1.2.2 User-Friendly UI Design 3
1.2.3 High Performance Oriented 3
1.2.4 Modularity and Practical Maintainability 3

2. High-level software architecture **4**
2.1 Subsystem Decomposition 4
2.2 Hardware/Software Mapping 5
2.3 Persistent Data Management 6
2.4 Access control and security 6
2.5 Boundary conditions 7
2.5.1 Initialization 7
2.5.2 Termination 7
2.5.3 Failure 8

3. Low-level design **8**
3.1 Object design trade-offs 8
3.2 Final object design 9
3.3 Layers 11
3.3.1 User Interface Management Layer 11
3.3.2 Web Server Management Layer 12
3.3.3 Data Management Layer 13
3.4 Packages 15
3.4.1 Packages by developers 15
3.4.1.1 Entities 15
3.4.1.2 Controller 15
3.4.1.3 Security 16
3.4.1.4 Repositories 16
3.4.1.5 Services 16
3.4.1.6 ReservationManagement 16
3.4.1.7 AnnouncementManagement 16
3.4.1.8 ProfileManagement 16
3.4.1.9 WorkoutProgramManagement 16
3.4.1.10 UserManagement 16
3.4.2 External Libraries 16
3.4.2.1 Node-postgres 16
3.4.2.2 Nodemailer 16
3.4.2.3 Moment 17
3.4.2.4 Bycrypt 17
3.5 Design Patterns 17
3.5.1 Singleton Design Pattern 17

4. Glossary & References **17**

1.Introduction

1.1 Purpose of the system

Our project is a web-based gym management system application. The aim of this project is to develop a web-based gym management application for sport facilities located in Ihsan Dogramaci Bilkent University. This application will allow students and gym personnel to interact and manage their gym related tasks online. Because of limited equipment and resources, and also because of the social distancing due to sars 2 coronavirus, it is needed for each student to book a time slot in order to use the sport facilities and their equipment. Students can also ask for a workout program from the gym instructors if they wish. The current system is not synchronized and has its own shortcomings such as lack of control and hardship in communication between involved individuals. Our aim is to improve the existing system and make it more modern and more effective.

1.2 Design Goals

The application aims to have many types of users like instructors and students and each user may attempt to use the application for a different purpose. The main goal for the application is to allow all kinds of users to take minimal time to understand the system and start using it accordingly.

The design goals of the project are decided according to the non-functional requirements from the analysis report. Our project system should have a user-friendly and understandable interface that can be used by both facility staff and university students. Since this system will contain private information about both staff and students, the system must be secure and sustainable. Considering the intensity of use of the existing sports facilities, the system should have a good performance without possible crashes and should be optimized.

1.2.1 Security and Privacy

Security measures have an important place in all web applications to prevent security breaches. The system will also require substantial safety precautions. Since our application aims to help only Bilkent University members such as students and Bilkent University Sports Center staff, only Bilkent e-mails will be accepted during

registration. Admin accounts which will be for the Instructors, Sport Center Head and Admins will be premade and given access to properties such as taking reservation and creating personal programs for students. These separations between normal users and admin accounts will prevent security issues such as accessing other users' phone numbers, emails or booking times.

1.2.2 User-Friendly UI Design

UI Design of the application will be easy to understand and functional specially for the Sport Center staff in order to prevent difficulties they are facing while taking reservations. According to our field research and feedback from the sport center workers, the current system that they are using is hard to understand for the new workers and difficult to use. To make a user-friendly application we made pages come in front of the user gradually, users will be able to understand the functions of each button and the way to achieve their purpose so they do not think about what they should do at which stage. At the reservation stage, users will just click at most 2 times in each page. Even a user with no initial knowledge will be able to use this application without having a hard time. This will be achieved through a minimalistic UI design approach that will not overwhelm the user with an abundance of functions or choices to be made.

1.2.3 High Performance Oriented

As the project is a web-based application, the response time between the server and database should be fast and at most 2 seconds. This is very critical for our program because of the reservation feature. Reservation needs quick response times like 1 second at average, in order to prevent multiple reservations in the same slot. When users refresh the page, the information that was on the reservation slots should be cached and only new information should be demanded from the database. As such, pages should not take more than 1 second to load.

1.2.4 Modularity and Practical Maintainability

Our web program will be developed using object oriented methodology, so we will be able to make changes easily and have better control of the program. Students and the reservation system will be controlled through inheritance by subclasses. As

an example the reservation system will have several subclasses such as gym reservation and open air facilities reservation. Also, reservation system class can be also a subclass of sport centers class, which is to choose which sport center that users want to make reservations such as East Campus Sports Hall, Main Center Sport Hall or Dormitories Sport Hall.

2. High-level software architecture

2.1 Subsystem Decomposition

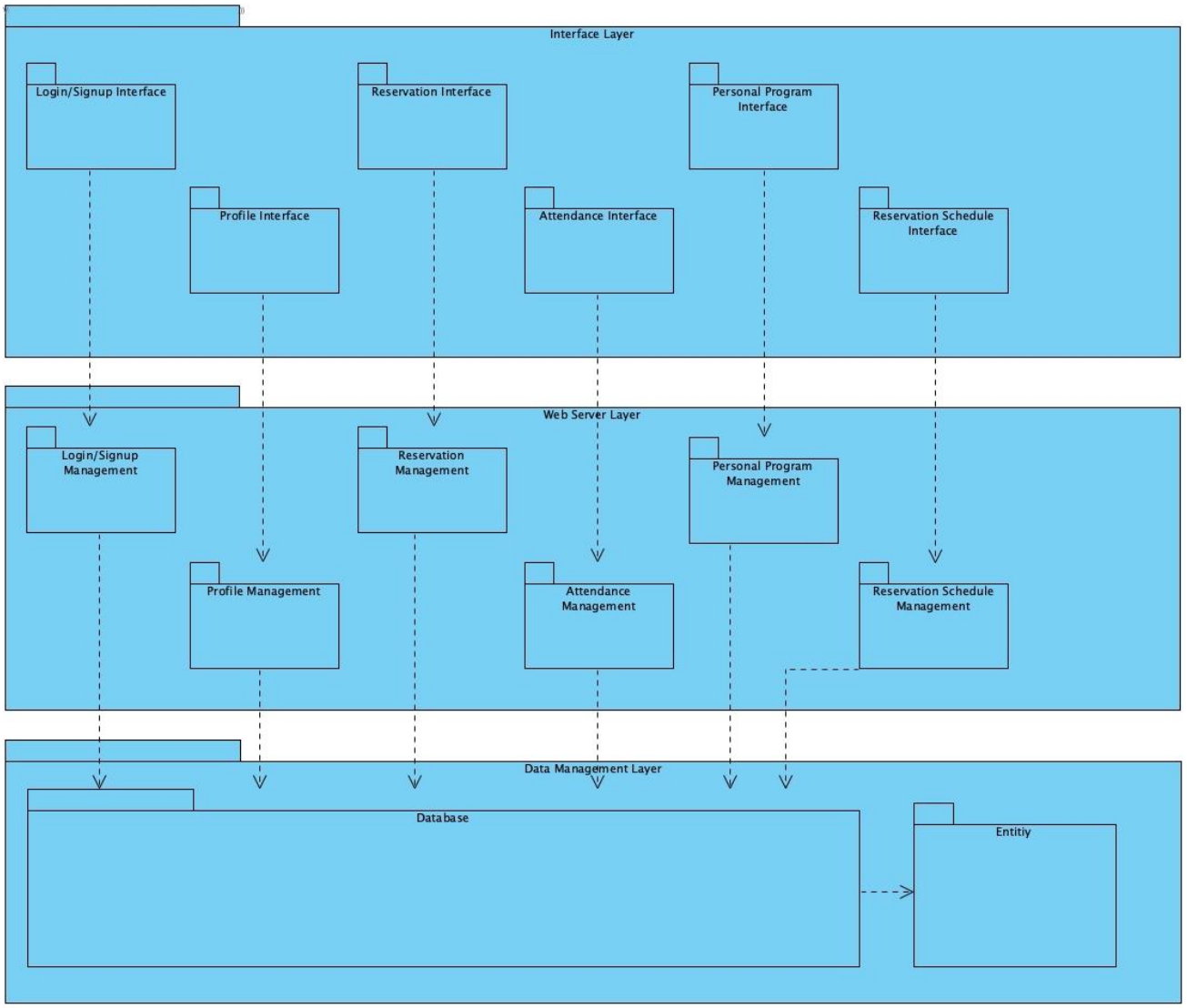


Fig. 1. Subsystem Decomposition

In order to achieve maintainability, our system uses 3 layers. In Fig.1 it can be seen that there are Interface Layer, Web Server Layer, and Data Management Layer.

The interface layer contains the pages of the website. There are 7 pages that users can access through the website. Each page has its controller in the backend that allows users to interact with the program by clicking the interactive parts such as buttons and navigation bars. Every subsystem in the interface layer includes the pages and functions of the pages. Pages are designed in a user-friendly way so that users will be able to understand the functions of each button and the way to achieve their purpose.

The Web Server Layer has the operations sent by the server in the back-end. This layer has several parts to allow stronger and flexible subsystems. Each subsystem in the web server layer manages the functionality of a particular system, such as the Reservation Management subsystem. The Reservation Management subsystem contains the service and controller classes related to the reservation process. Every subsystem depends on the database.

The Data Management Layer contains the Database and Entity subsystem. Database has the repository interfaces that can communicate with database and entity.

2.2 Hardware/Software Mapping

Our Sport Center Management System does not require any hardware component manipulation. Since the project aims to be a web program, there will be no strict operating system requirements and users will access the program through their browsers. The only thing that is needed is a computer, a mobile phone, or any device that supports a web browser. Users can use our website from any web browser such as Chrome, Mozilla, Firefox, IE9, Safari, Opera and so on.

For computers, it is needed to have sufficient input components to interact with the interface of the project. Those components are mouses and keyboards.

For mobile phones, since the user will interact with the interface by touching the screen, a properly functioning touch screen is needed.

The program will be tested using Google Lighthouse performance test and it will aim to reach a score above 90 out of 100.

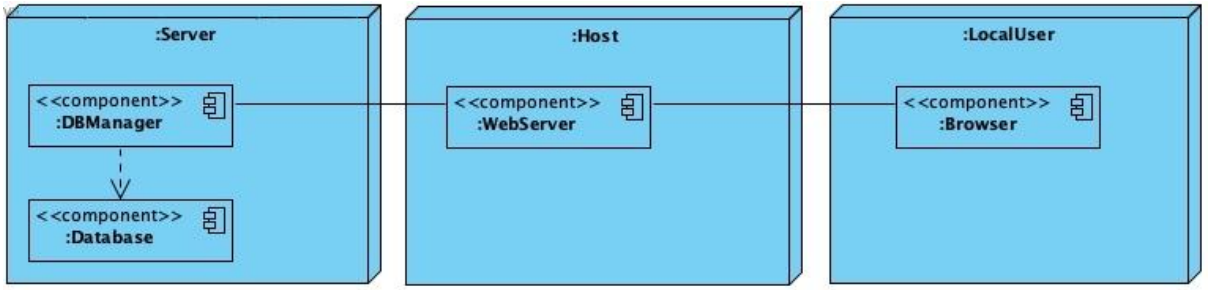


Fig.2: Deployment Diagram

2.3 Persistent Data Management

In our project, PostgreSQL will be used [1]. PostgreSQL is a open source database management system. Since the tables that PostgreSQL provides are relational, it will be easy to handle manipulation of data once the cascading and restriction options are set properly. Also, our team is comfortable with PostgreSQL compared to other database management systems. In the database, we plan to store informations such as dates (account creation dates), account informations, reservations, id of most of the classes such as user id, instructor id, sports center id and such. Other than that, we plan to store announcements, schedules, instructors information and even exercises. With these storing options, it will be easier for us to create a connection between all of these entities. It gives us the freedom about managing and editing data.

It will use React Framework. Another advantage of PostgreSQL is that PostgreSQL servers can be easily connected to any React App.

2.4 Access control and security

Our gym management application ensures security for users and gives access control. To make the application secure, we will deploy an authentication system to the application. Current user presence query will be made. We will use cookies to provide sessions to the users. The parts users can access will be determined by their type. Whether they are admin, user, instructor etc. As we will hash users' passwords' the application will be secure to possible data leaks. Also, cookies are used to protect against cross-site forgeries, which is another way to improve security.

	Admin	Instructor	Student	System
Sign In	X	X	X	X
Sign Up			X	
View Facilities Page	X		X	X
View Sport Activity Page	X		X	X
View Reservation Page			X	X
View Profile Page	X	X	X	X
View Workout Program Request Page		X		X
View Attendance Management Page	X	X		X
View Attendance Page	X	X		X
Create Announcement	X	X		X
Update Workout Programs		X		X

2.5 Boundary conditions

2.5.1 Initialization

Because our application is a web based application, it does not need any installation. It only requires the user to have an internet connection and javascript enabled on their browser. Users can access our app using either their computers or mobile phones. Users also need to have a valid account to access all parts of our application. To initialize frontend as we use React framework we need to use related commands in react like “npm start”. For the backend we use node.js and with the nodemon library we use just the “nodemon” command to run the server.

2.5.2 Termination

Our application works as a single system. The only way for our application to terminate is admins terminating the app completely. This can happen in case of an emergency or in the time of maintenance. To terminate our application, we should

terminate both the frontend and backend. To kill running servers “ctrl + c” command is enough.

2.5.3 Failure

In case of failure caused by a communication between the user's system and our database, the system will try to access the database three times in the next 10 seconds. If the application could not access the database or there was a communication problem, a message will be shown to the user.If the problem couldn't be solved, admins will manually fix the problem. Ideally, a failure will not result in any damage or data loss.

3. Low-level design

3.1 Object design trade-offs

Security vs. Usability: Our application will be used by students and staff of Bilkent university. This exclusivity requires heavier security barriers. This can make it more difficult for the users to create new accounts, manage their accounts and use the application in general.

Performance vs. Reliability: In order to have better performance and faster response time, our application does contact the database all the time and uses a cache system. This approach will make our app run faster but it can cause problems in case of connection loss or cache information not being up to date with the database.

Modularity vs. Simplicity: Our web application will be developed using object-oriented methodology and using this methodology, we will be able to achieve a modular and easier to maintain program. Application is designed to be as modular as possible. This approach makes it easier to maintain the application and add new features if needed. This approach will make the system more complicated at the same time. This loss of simplicity can be more visible as the application grows and gets new features.

3.2 Final object design

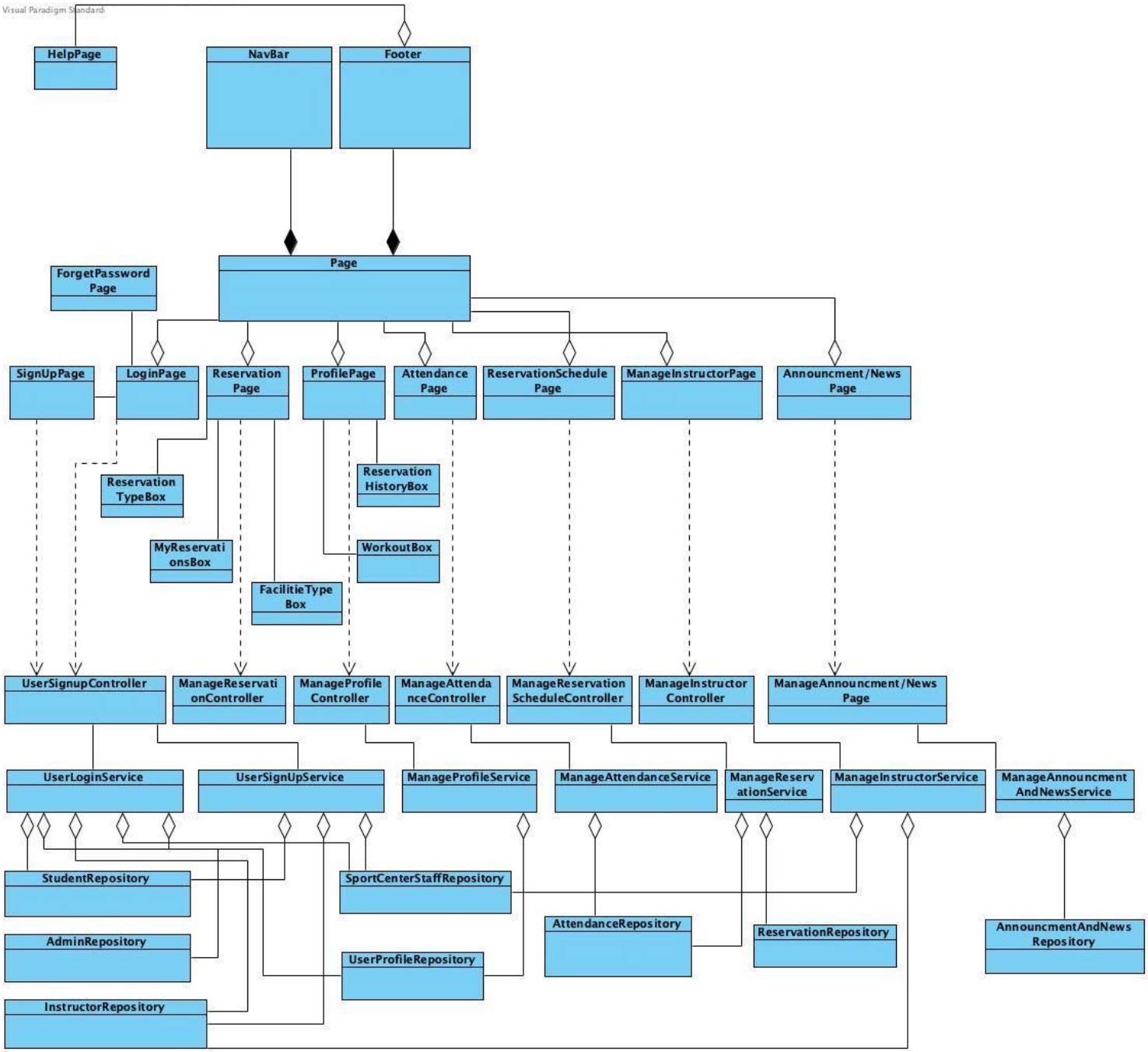
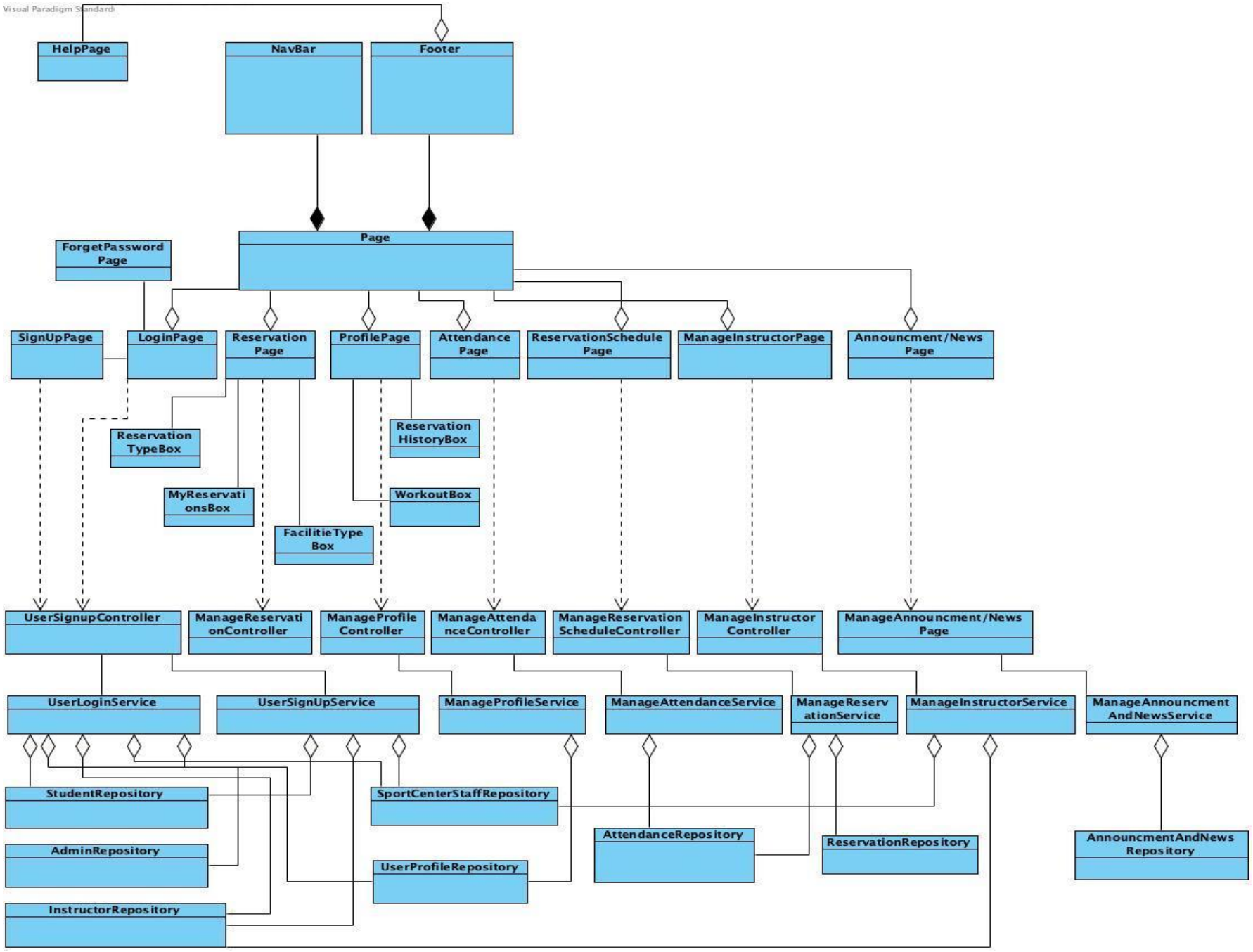


Fig. 3. Final Object Design


Visual Paradigm Standard
HelpPage
NavBar
Footer
Page
ForgetPassword Page
SignUpPage
LoginPage
Reservation Page
ProfilePage
Attendance Page
ReservationSchedule Page
ManageInstructorPage
Announcment/News Page
Reservation TypeBox
Reservation HistoryBox
MyReservati onsBox
WorkoutBox
FacilitieType Box
UserSignupController
ManageReservati onController
ManageProfile Controller
ManageAttenda nceController
ManageReservation ScheduleController
ManageInstructor Controller
ManageAnnouncment/News Page
UserLoginService
UserSignUpService
ManageProfileService
ManageAttendanceService
ManageReserv ationService
ManageInstructorService
ManageAnnouncment AndNewsService
StudentRepository
SportCenterStaffRepository
AdminRepository
AttendanceRepository
ReservationRepository
AnnouncmentAndNews Repository
UserProfileRepository
InstructorRepository

3.3 Layers

3.3.1 User Interface Management Layer

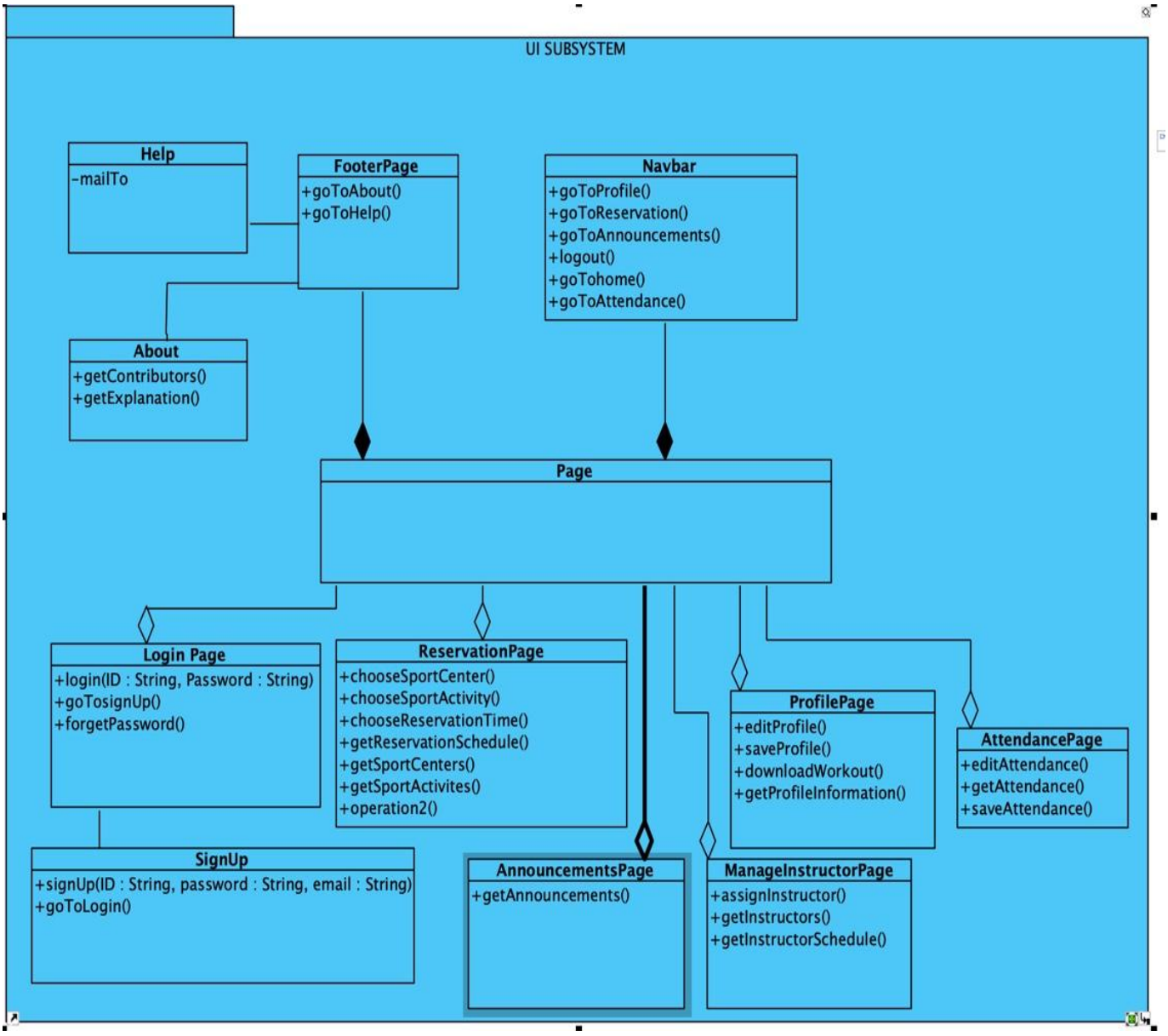


Fig. 4. User Interface Management Layer

The Interface Layer is the outermost layer in our application design. It operates the user and application interaction. The classes that are included in this layer serve as HTML components in the HTML file format and they are communicating with the Web Server Management Layer.

3.3.2 Web Server Management Layer

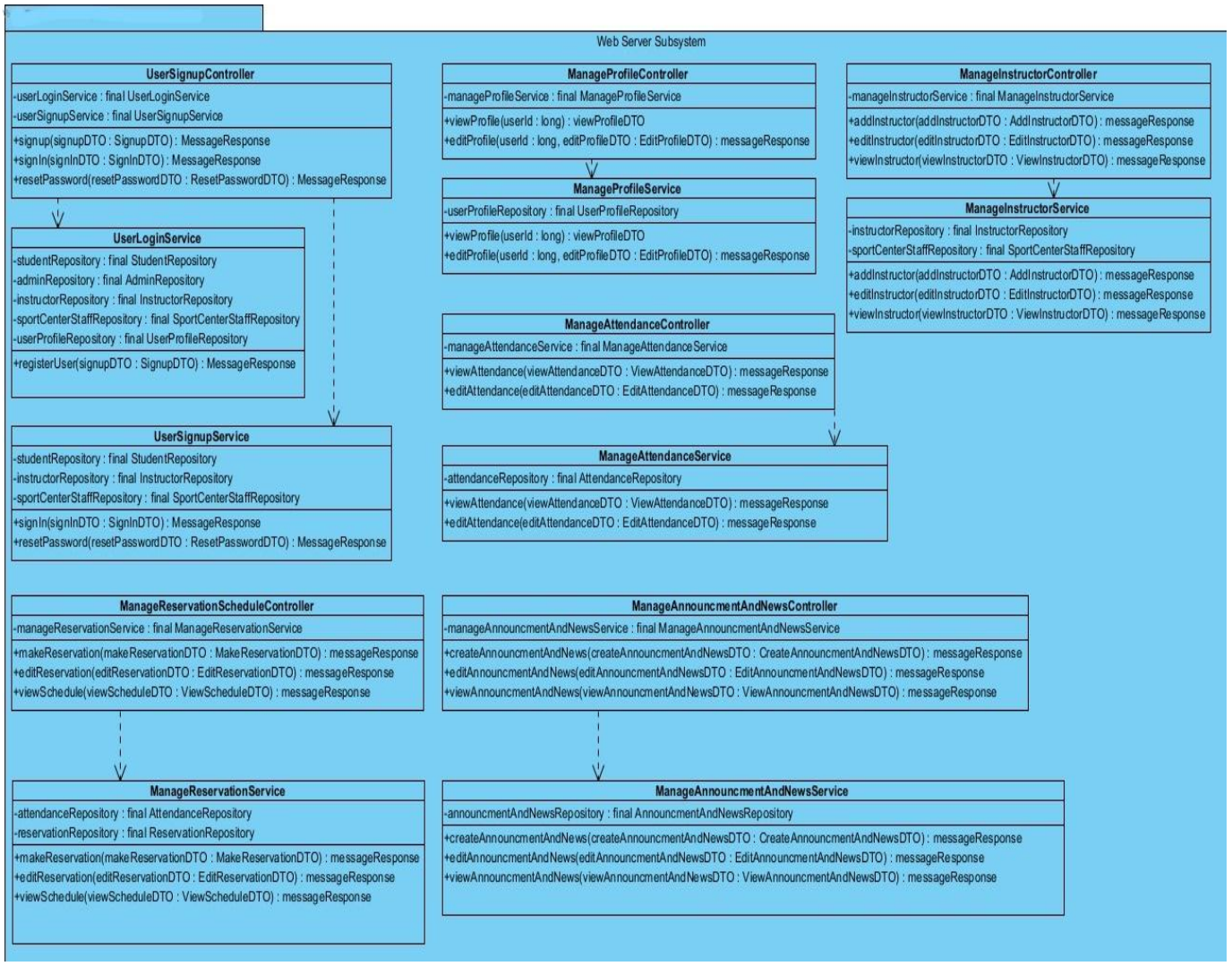


Fig. 5. Web Server Management Layer

This layer represents the Web Server Layer in our project. This layer supports the main User Interface Layer and is responsible for linking user actions to the Data Management Layer. It manages the request from the user interface and returns the requested data to the Data Management Layer.

3.3.3 Data Management Layer

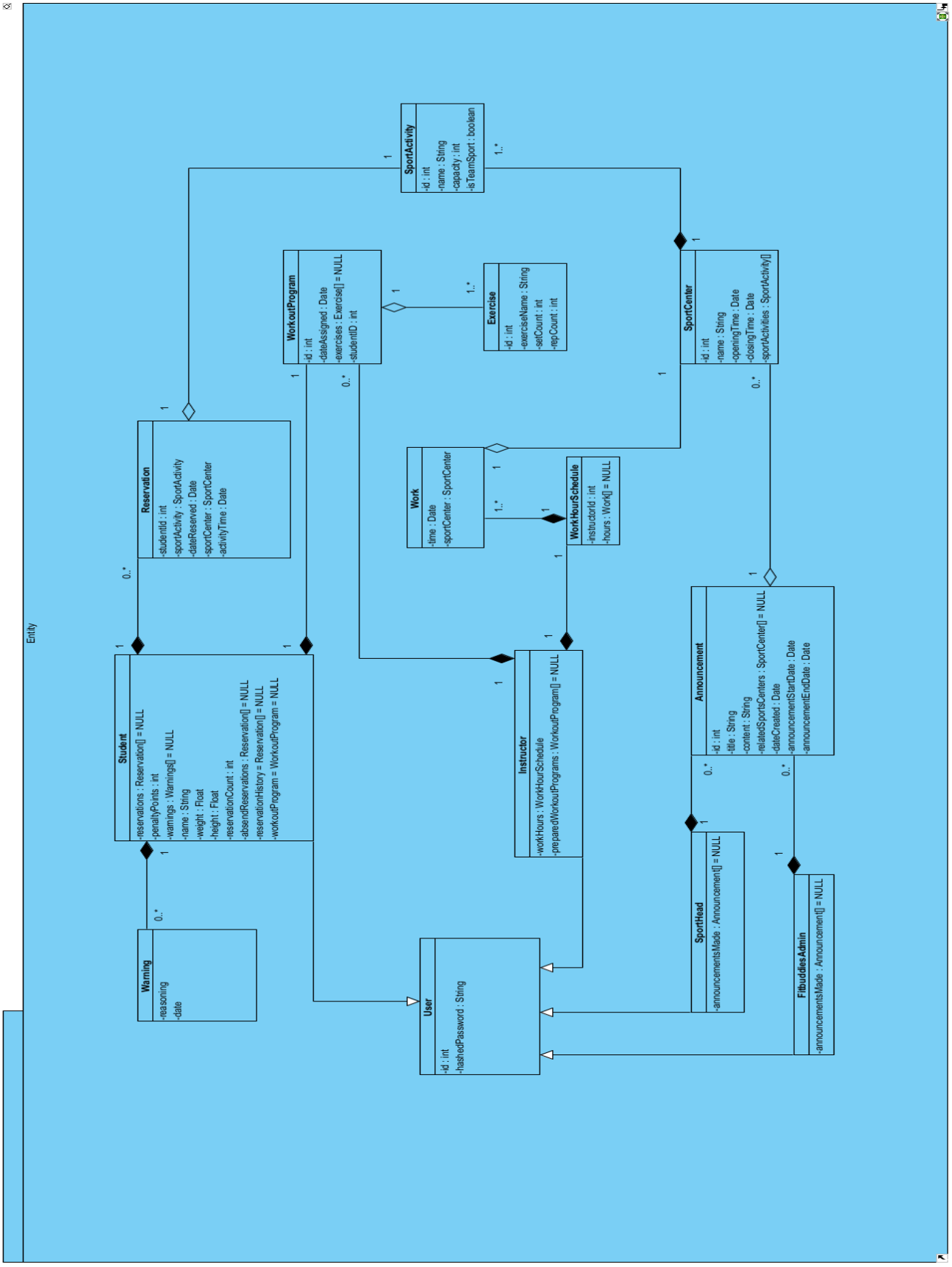


Fig. 6. Data Management Layer

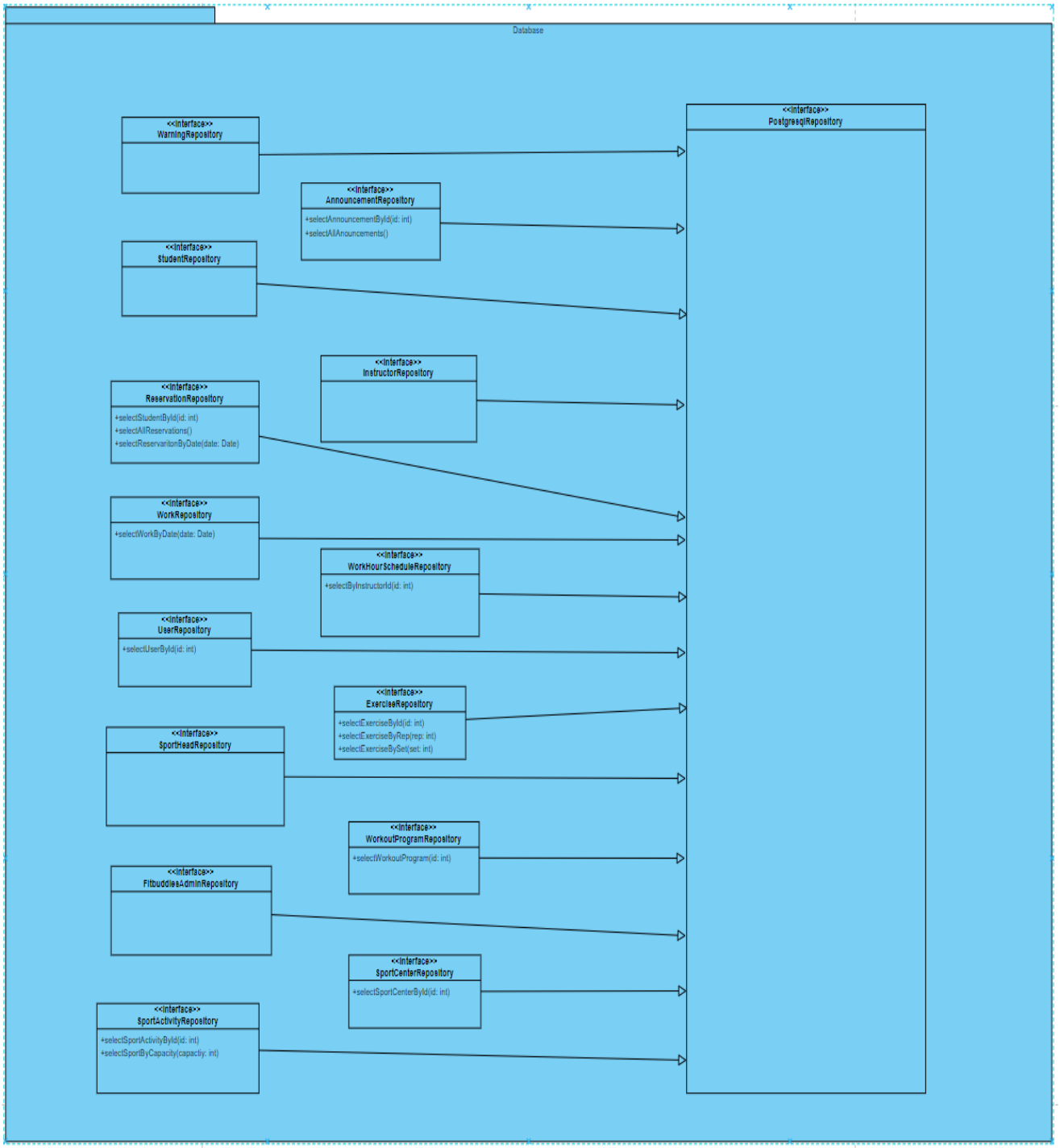


Fig. 7. Database Repositories Diagram

This layer is the Data Management layer which is the innermost layer in our application design. It supports the Web Service Layer. This layer handles the Web Service Layer and consists of two systems. It has Entity and Database subsystems. Entity subsystems have the entity classes and the database subsystem has database communications.

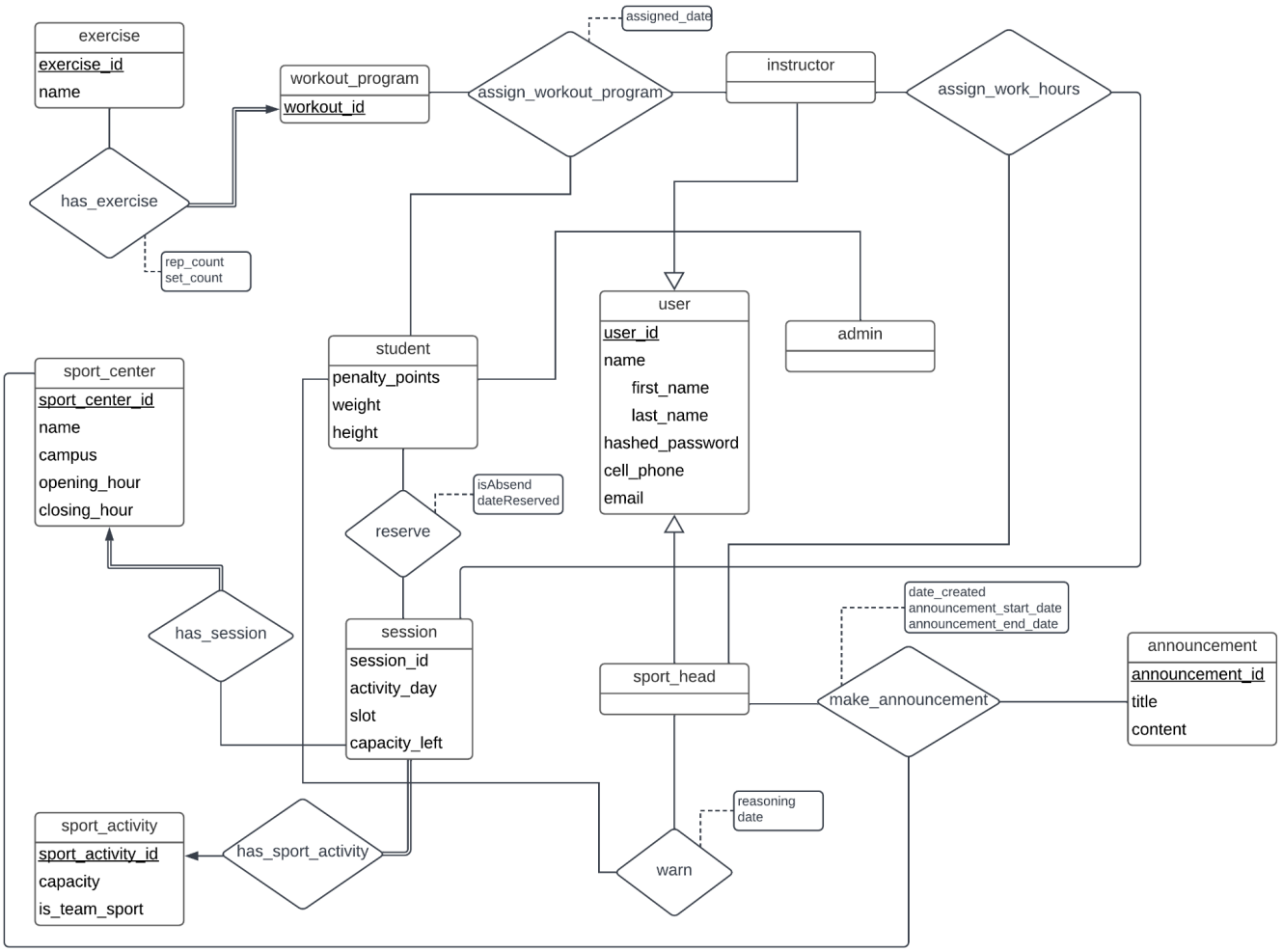


Fig. 8. E/R Diagram

3.4 Packages

Our system uses external libraries and internal packages to operate.

3.4.1 Packages by developers

3.4.1.1 Entities

This package includes the methods and objects used to run the processes for the website.

3.4.1.2 Controller

This package includes controller classes for the website.

3.4.1.3 Security

This package includes security classes for the website.

3.4.1.4 Repositories

This package includes operations for the database.

3.4.1.5 Services

This package includes all service classes to ensure 3-tier architecture.

3.4.1.6 ReservationManagement

This package includes classes that are responsible for managing the reservation system of the website.

3.4.1.7 AnnouncementManagement

This package includes classes that are responsible for managing announcement system for the website

3.4.1.8 ProfileManagement

This package includes classes that are responsible for managing profiles for the website.

3.4.1.9 WorkoutProgramManagement

This package includes classes that are responsible for editing workout programs for the users.

3.4.1.10 UserManagement

This package includes classes that are responsible for managing users for the website.

3.4.2 External Libraries

3.4.2.1 Node-postgres

This library will be used in order to have communication between the website and the database.

3.4.2.2 Nodemailer

This library will be used in order to send mails to the users of the website.

3.4.2.3 Moment

This library will be used in order to edit and manage dates of the website.

3.4.2.4 Bycrypt

This library will be used in order to secure users credentials.

3.5 Design Patterns

3.5.1 Singleton Design Pattern

Singleton design pattern is used to restrict instantiation of classes to objects. In our application, we will create database connections through repositories and access to those repositories via services and controllers (web server). For this case, those classes that are on the controller layer should have only one instance and those should be created during the launch of the application. Thus, we will use the Singleton design pattern for our controller (web server) layer classes.

4. Glossary & References

- “Postgresql”. The PostgreSQL Global Development Group.[Online]. Available: https://www.postgresql.org/
- B. Bruegge and A. H. Dutoit, Object-oriented software engineering: Using UML, patterns, and Java. Boston: Prentice Hall, 2010.
- UML: Universal Markup Language. It is a language being used while designing the project.
- OOP: Stands for Object Oriented Programming.