Bilkent University

Department of Computer Engineering

CS319 TERM PROJECT

Section 3

Group Lux Æterna

Bilkent Pandemic Manager

Design Report

Group Members:

- Remzi Tepe (21802713)
- Deniz Semih Özal (21802414)
- Arman Engin Sucu (21801777)
- Berkan Sivrikaya (21803052)
- Saitcan Başkol (21803589)

Supervisor: Eray Tüzün

1. Introduction

1.1. Purpose of the System

Our Project is a web-based application which tracks mainly HES codes, vaccinations and tests of Bilkent University students, instructors and workers. The application has some features for students and instructors such as adding covid-19 related files, selecting nearby students if one is affected by virus, using risk calculator to calculate corona risk, viewing covid-19 numbers in dorms, weekly infected numbers and news. Admin could control the accuracy of the files, dorms and courses.

1.2. Design Goals

The non-functional requirements from the analysis report set the design goals of the project. System of the project will be designed to have a usable interface along with maintainable and efficient features while it is secure and error-free which means it is reliable.

1.2.1. Usability

Since everyone including students, instructors and workers in Bilkent will be using the application, it should have an explanatory and usable interface. For example, names of the buttons, tabs and options will be easy to understand and to get interact with practically. So that, anyone can clearly see and take action about what they need to do in the application.

1.2.2. Maintainability

One of the most important aspects of this application is to be maintainable through the pandemic since it will last for a long time unfortunately. For instance, there might be new variants of the virus that need to be taken care of in different ways such as new test methods or new treatments. Also, the application needs to open for new features according to user feedback.

1.2.3. Efficiency

The application must be efficient since there will be intense data flow especially while uploading the photos of the vaccination cards and test results. To make it happen, the application will be run on cloud deployment. As a result, users will have no problems with the server about delays.

1.2.4. Security

Since the application stores personal data of users such as vaccination files, HES Codes, and test files, those data should not be seen and reached by other users. Also, since this application does not connect with Life Fits Into Home App (Government's application), some personal information of a user will be checked by an admin so securing information is a crucial aspect.

1.2.5. Reliability

Our project mainly tracks coronavirus status of people who belong to Bilkent University therefore the program must be error free since wrong information could lead to health issues in a school. For instance, if someone's coronavirus status is written as risk-free while it is risky, it would make a problem in a class or a dormitory.

2. High-Level Software Architecture

2.1. Subsystem Decomposition

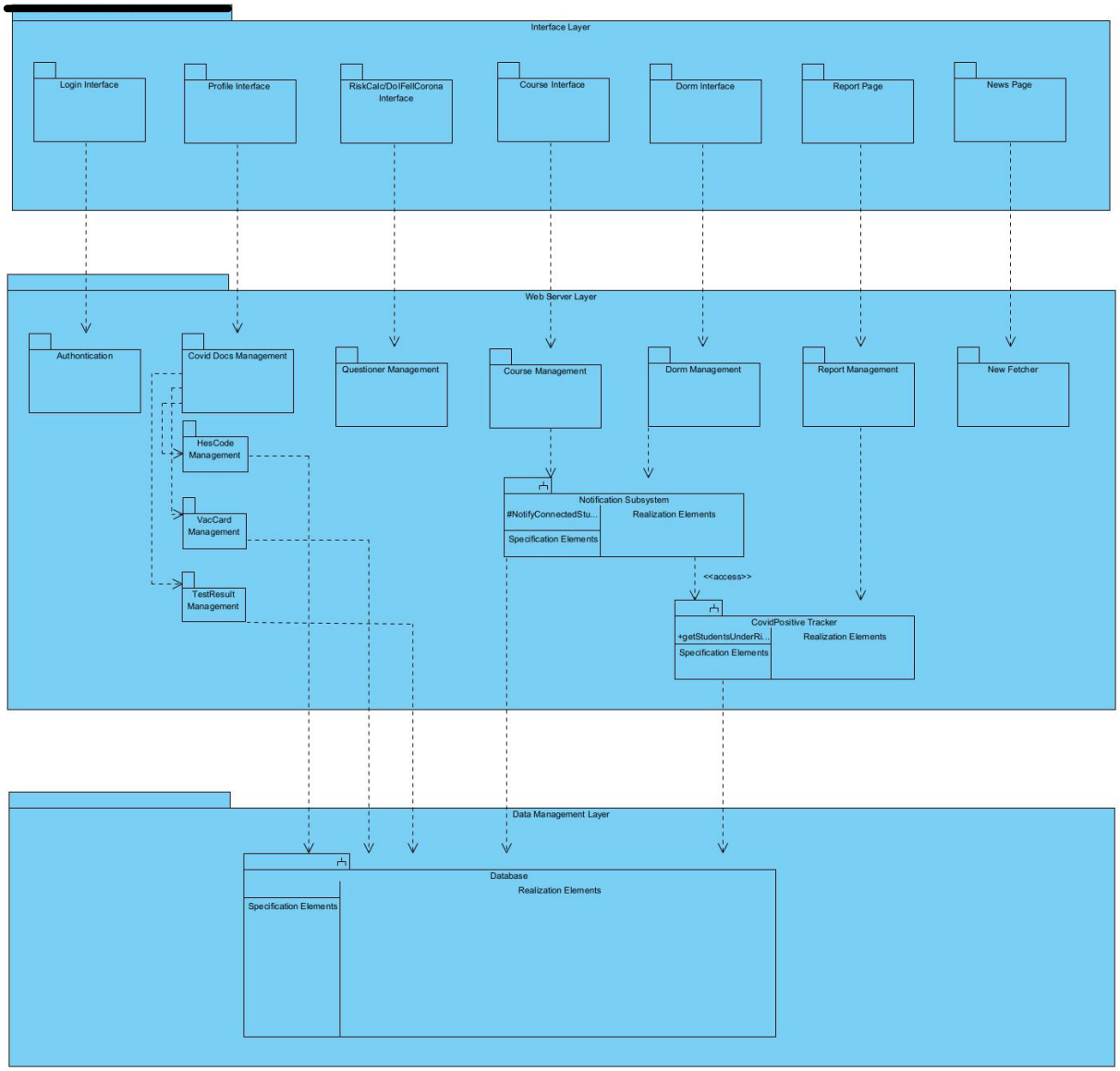


Figure 2.1. Subsystem Decomposition of The System

We have followed a 3-layered architectural style to design our project. The first layer is the User Interface Layer which represents the presentation layer of the project. This layer acts like the boundary object and it connects with the controller object. The interface's primary job is to interpret actions and outcomes into something that the user can comprehend.

The second layer is the Web Server Layer which represents the business logic layer of the project. This layer acts like the controller object and it provides the connection between boundary and entity objects. This layer manages the program by coordinating it, processing commands, and making logical judgments and assessments.

The third layer is the Data Management Layer which represents the data access layer of the project. This layer acts like the entity object and it receives messages from the controller. In this layer, the information is stored and retrieved from the database or file system. The information is subsequently transmitted back to the web server layer for processing before being returned to the user.

2.2. Hardware/Software Mapping

This project does not require any specialized hardware components to run successfully. Since this is a web application, users could access websites via various browsers. Hardware systems should be enough to run the applications. We thought to create only web applications for now, so personal computers or any device compatible with the web is comfortable. However, applications could be deployed on mobile as well in later. Some browsers which are compatible with our project are Google Chrome (version: 96.0.4664.45), Firefox (version: 91.0), Microsoft Edge (version: 96.0.1054.34), Safari (latest version) and so on.

2.3. Persistent Data Management

In this project PostgreSQL will be used as a database. PostgreSQL has been chosen, since it is a relational database which enables to show relations between objects. Moreover, PostgreSQL has a user-friendly interface that makes basic features to apply easily. Furthermore, our backend team has most experience with PostgreSQL. All of the entity objects in the project will be deployed to the database as tables by Hibernate. Then by using AWS, the database will be deployed to the cloud which can be edited by the web server layer’s services and JPA repositories on requests of the UI layer. Through the UI layer, the user can transmit his/her request by GET, POST, DELETE and PUT(CRUD operations). Therefore, users can edit the database objects via the UI layer.

2.4. Access Control and Security

In our application, security is very crucial since personal files need to be preserved and should be unreachable from other users. Users will be assigned different user credentials according to their user types. There are four types of users: instructor, student, staff, and admin. User interface could be changed according to user types. The resources a user can reach is determined from one's permission. Since our application does not have a chance to fetch information from Life Fits in Home app, information accuracy procedures will be done by an admin like checking Covid-19 Files. Last but not least, for every page according to enable authorization, tokens will be transmitted via cookies page to page.

2.5. Boundary Conditions

2.5.1. Initialization

Our application does not require any installation from users since it is a web application and runs on a web server. Users will be given passwords and their access permission will be changed according to their user types. All data of users will be stored in the database and datas will be fetched while the user is active on the website. All information about the user would be saved in the database.

2.5.2. Termination

Users could terminate the site by manually clicking the logout button or the active account will be terminated after a while if there is no motion in application. Admin could terminate the account of a user as well.

2.5.3 Failure

In order to be redundant we use the synchronous replication and streaming replication features of postgresql database. Also in order to prevent data loses failover features of postgresql are implemented. Moreover cloud solutions for redundancy are in our tool set such as Azure Advanced Threat Protection.

3. Low-Level Design

3.1. Object Design Trade-Offs

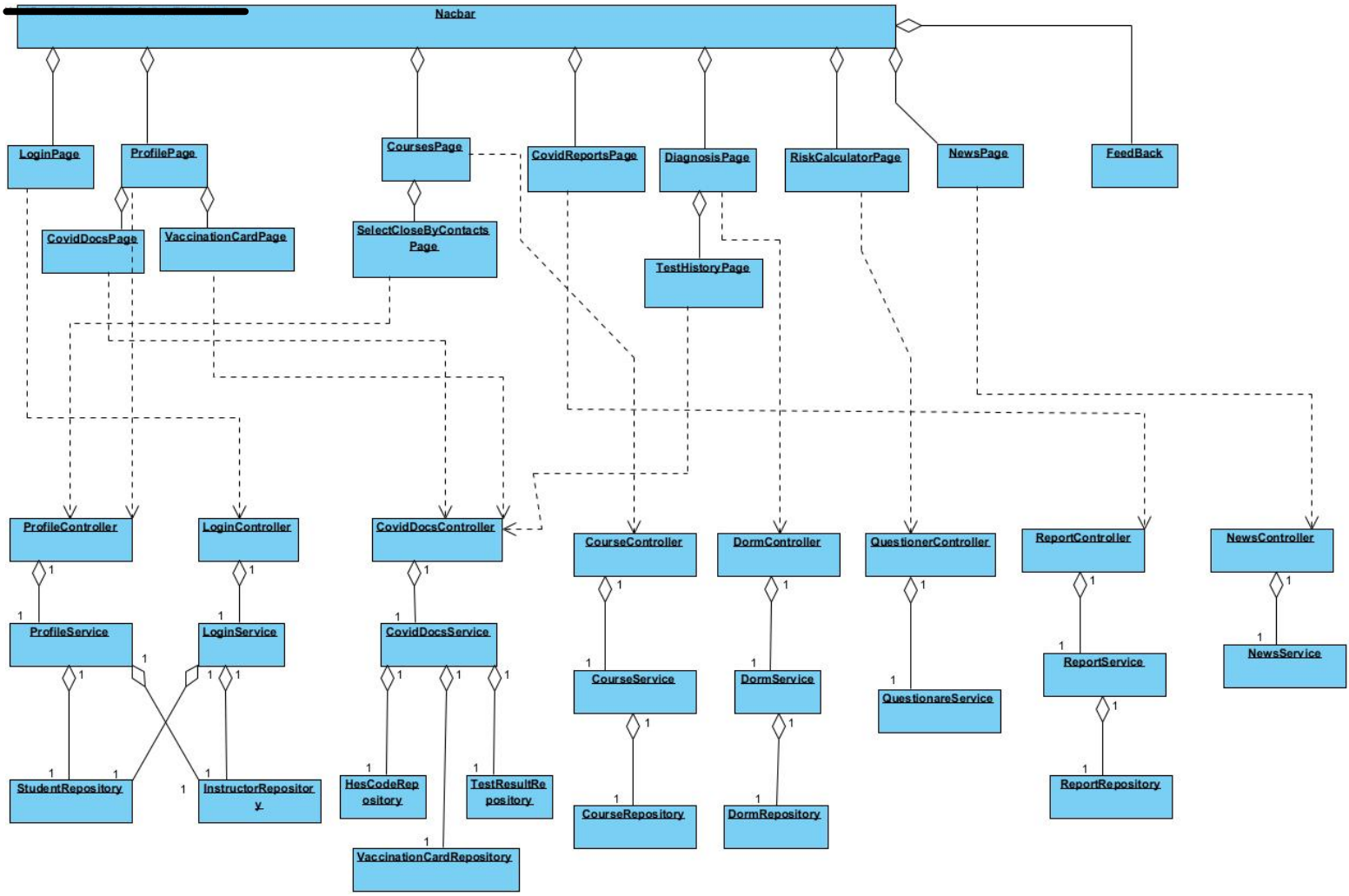
Maintainability versus Memory: Since everyone will be uploading their vaccination cards, Covid-19 test results and their medical reports, there will be a burden on memory. On the other hand, when people start to upload their information, the application will be maintainable and all the Covid-19 cases will be tracked easily.

Functionality versus Usability: The application will have components that only Bilkent students and instructors can actually understand and interact with. So, the application will be useless off-campus. It will decrease the usability of the application.

Cost versus Security: The application will be coded on all free frameworks and programs. Therefore, the cost will be too low. There will be only a domain cost. However, it might create some security issues since the application will almost have no protection against data theft.

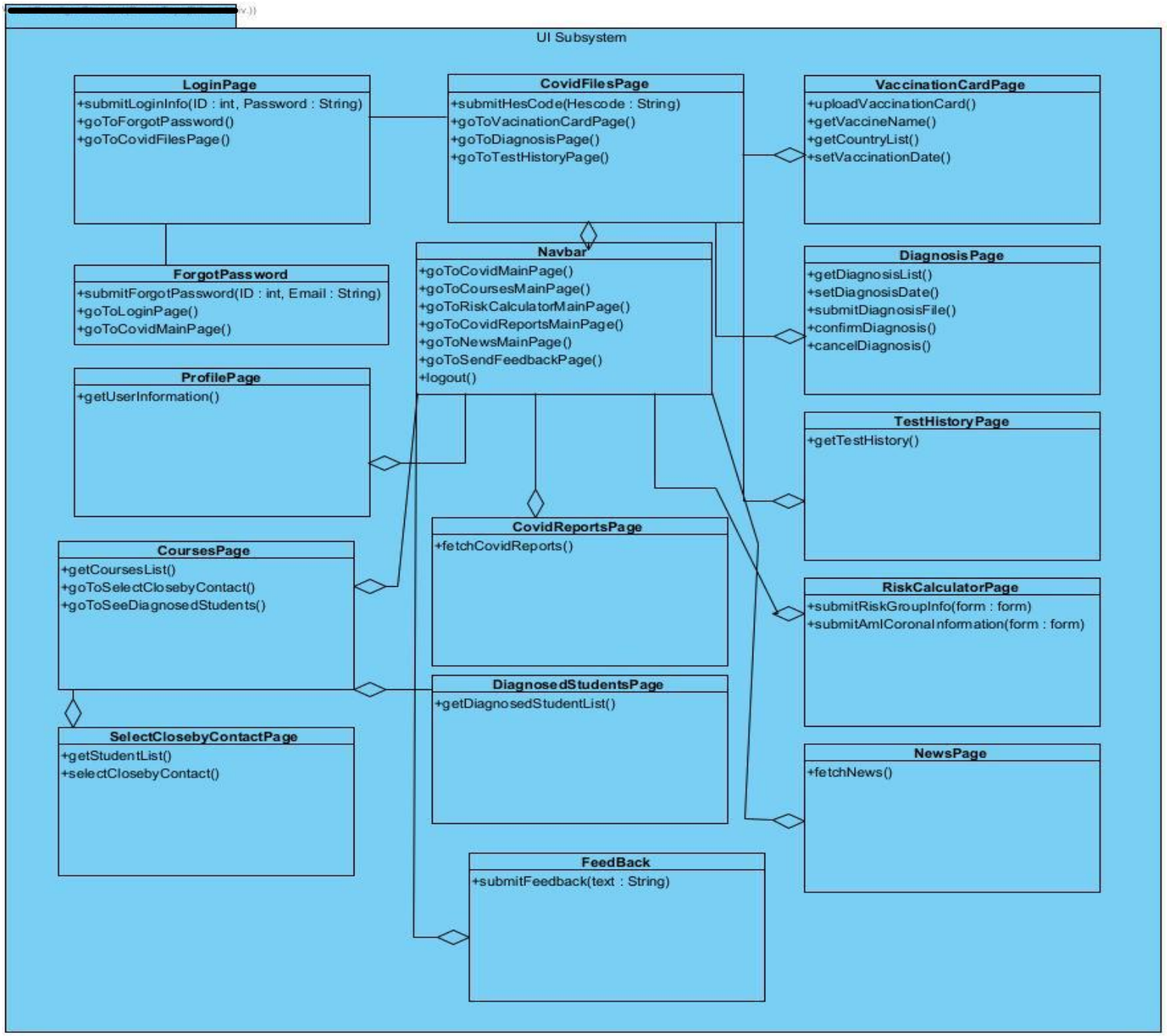
Expandability versus Performance: The application can be adapted to wider areas such as cities with a few changes in the code since it is basically for not only Bilkenter but also for humans in general. However, there is a problem that if it is implemented for wider areas the performance of the application can decrease rigidly since there will be a huge amount of data flow in between the system and user.

3.2 Final Object Design



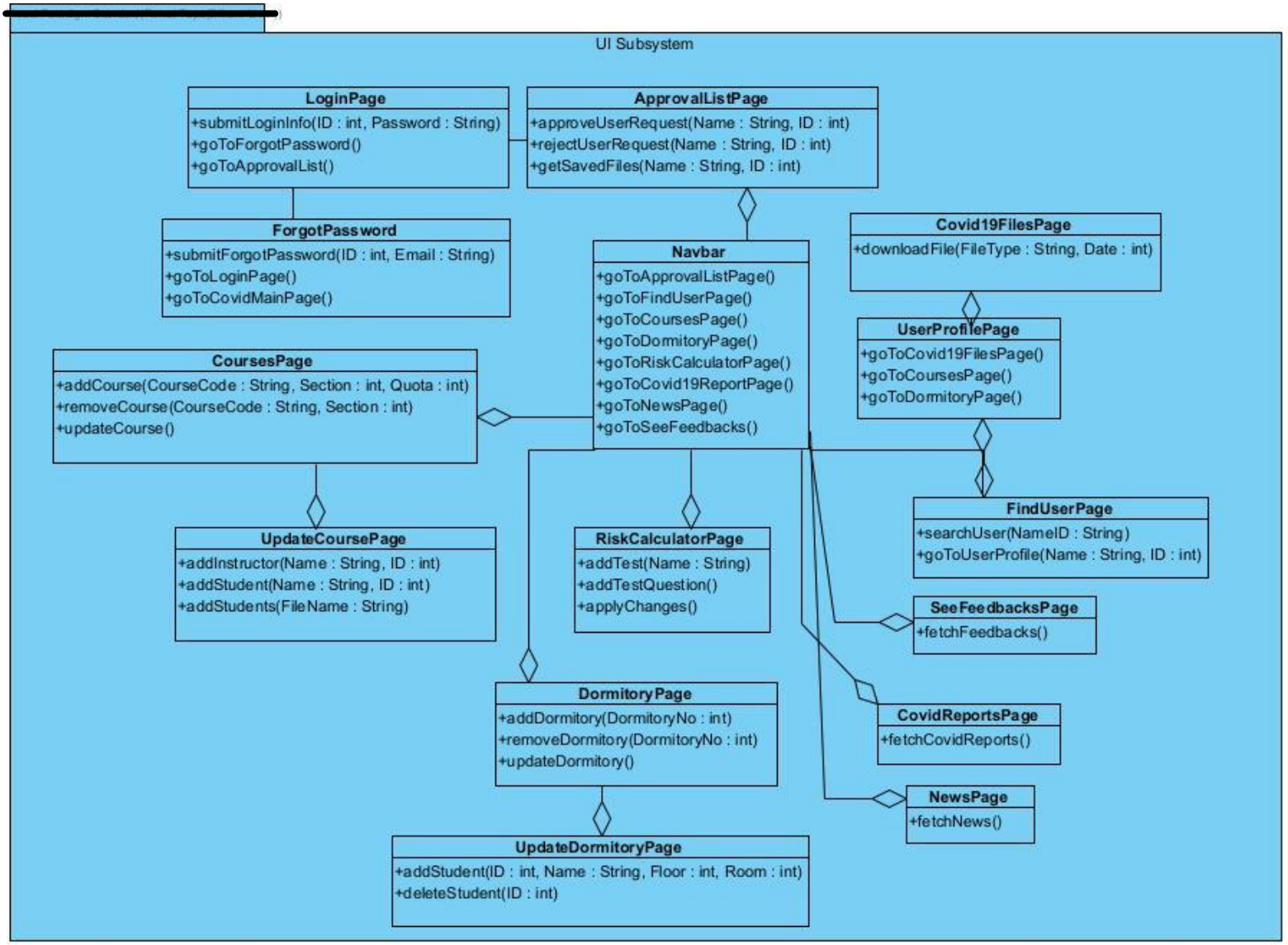
3.3. Layers

3.3.1. User Interface Management Layer



This user interface layer acts as a boundary object between the user and the application layer. Classes in this layer are in HTML format. Those classes work with the Web Server Layer. The layer runs without any problems since HTML has dynamic and mutable objects in this layer.

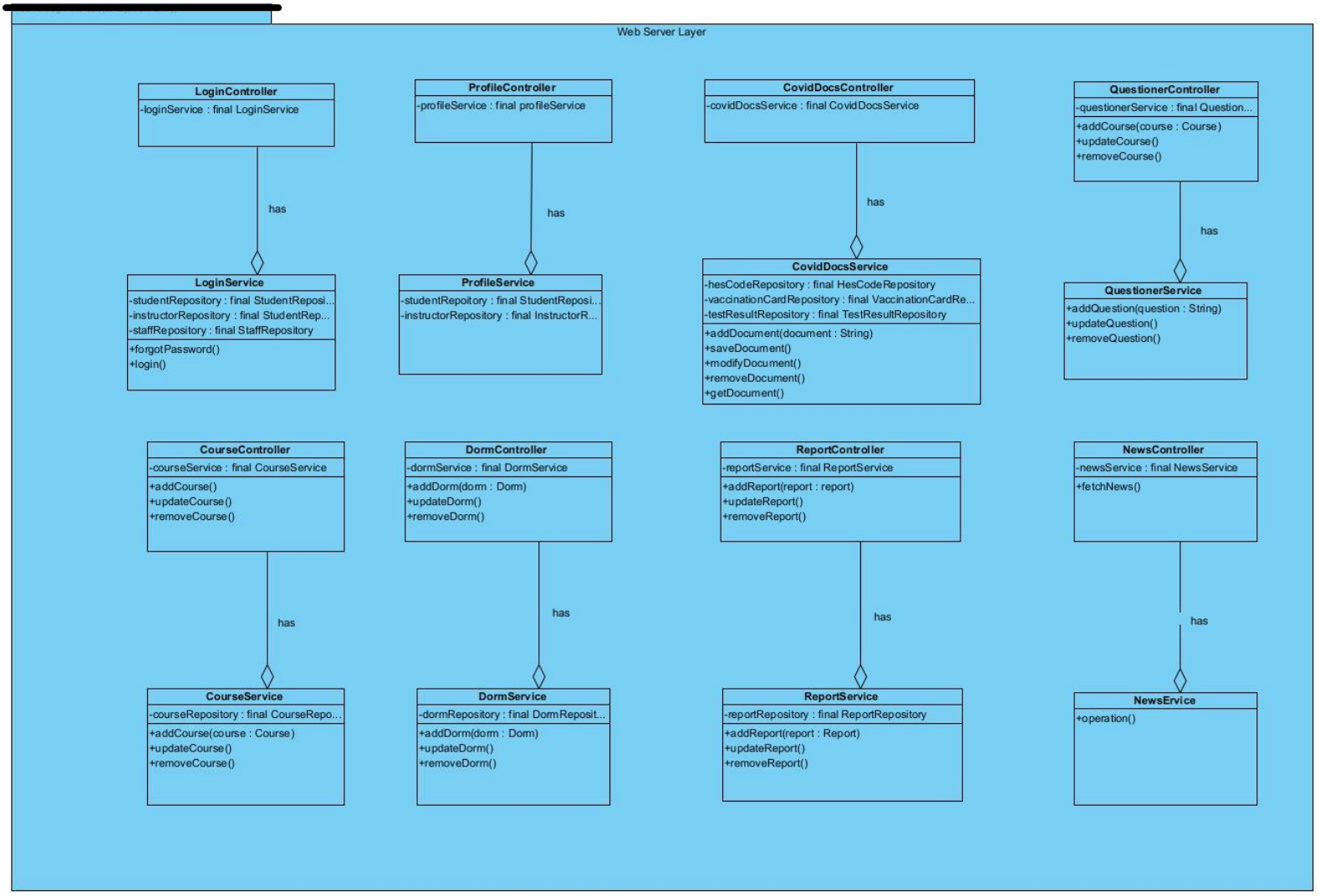
3.3.2. Admin Interface Management Layer



Admin Interface Management Layer acts as a boundary between the user and the application layer. All classes are in the format of HTML. Many of those classes will be working with Web Server Layer. Because the layer will be coded in HTML, the objects in the classes will be dynamic and changeable while there is no problem in running.

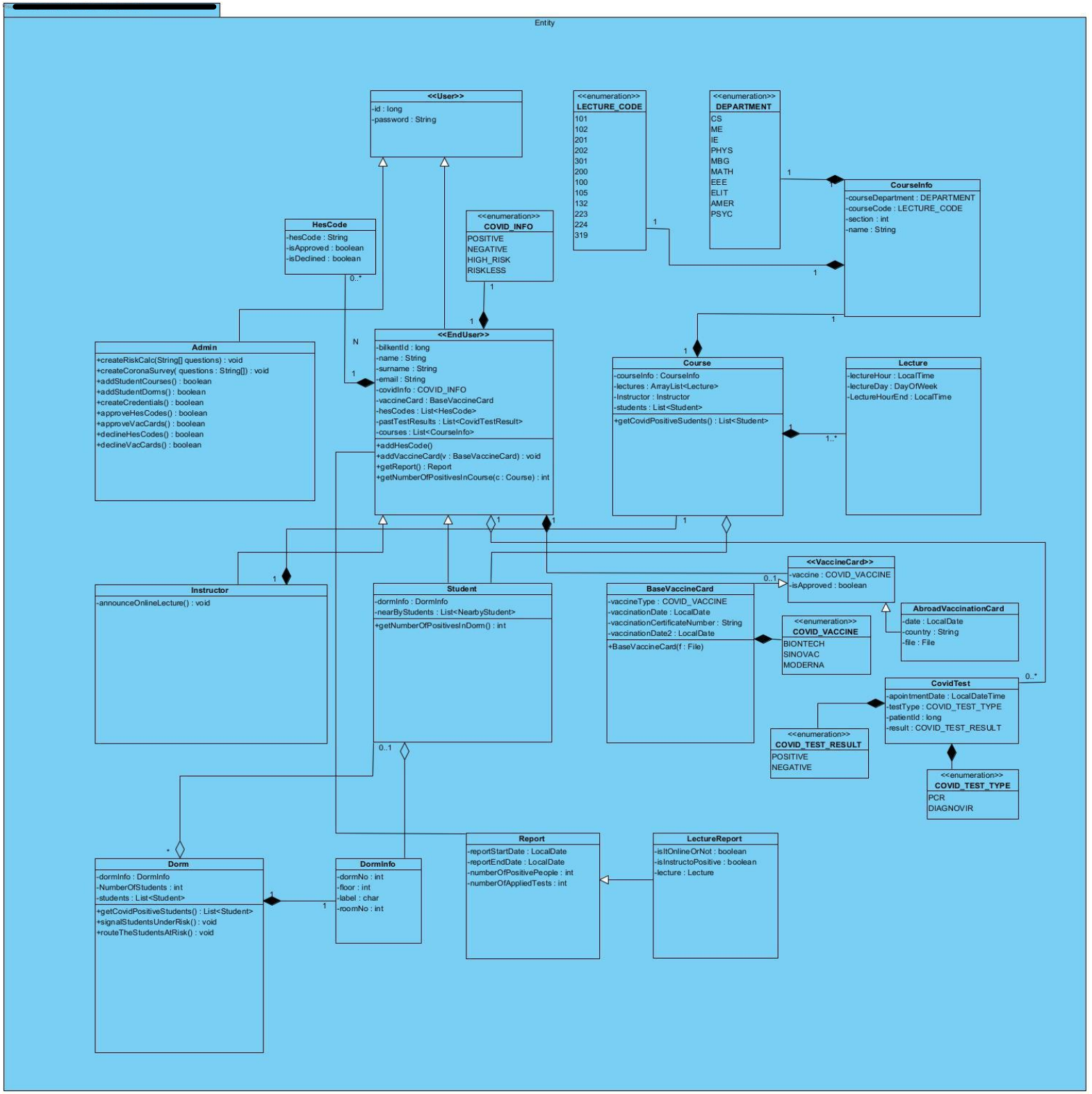
3.3.3. Web Server Layer

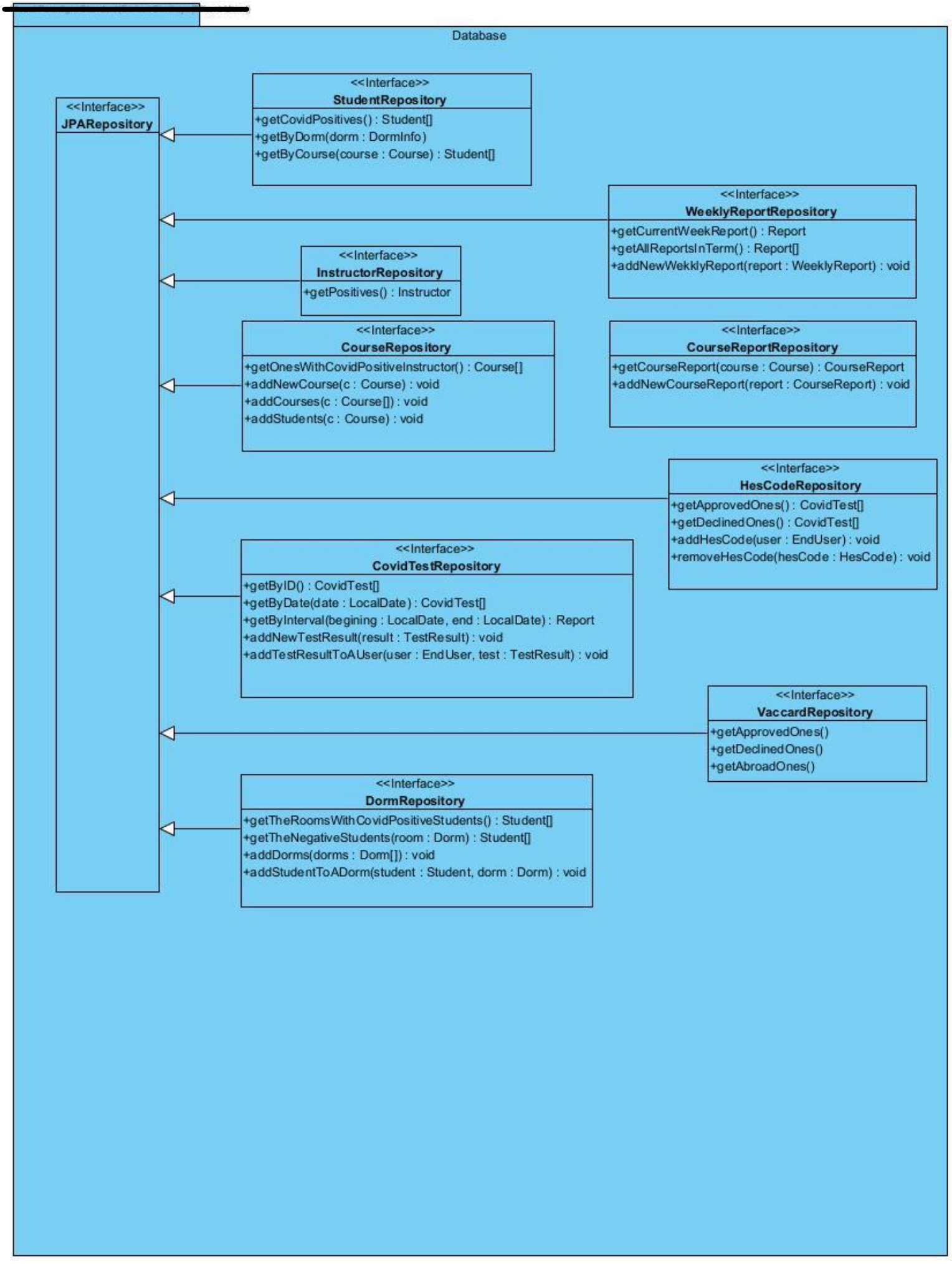
This layer represents the Web Server Layer of the project and it provides a connection between Presentation Layer (User Interface) and Data Access Layer (Database). It gets the request from the upper layer and sends it to the lower level.



3.3.4. Data Management Layer

Data Management Layer includes Database and Entity subsystems. In database subsystem JpaRepository interface extensions are there to manage entities whereas Entity subsystem contains the entity classes.




Database
<<Interface>>
JPARepository
<<Interface>>
StudentRepository
+getCovidPositives() : Student[]
+getByDorm(dorm : DormInfo)
+getByCourse(course : Course) : Student[]
<<Interface>>
WeeklyReportRepository
+getCurrentWeekReport() : Report
+getAllReportsInTerm() : Report[]
+addNewWekklyReport(report : WeeklyReport) : void
<<Interface>>
InstructorRepository
+getPositives() : Instructor
<<Interface>>
CourseRepository
+getOnesWithCovidPositiveInstructor() : Course[]
+addNewCourse(c : Course) : void
+addCourses(c : Course[]) : void
+addStudents(c : Course) : void
<<Interface>>
CourseReportRepository
+getCourseReport(course : Course) : CourseReport
+addNewCourseReport(report : CourseReport) : void
<<Interface>>
HesCodeRepository
+getApprovedOnes() : CovidTest[]
+getDeclinedOnes() : CovidTest[]
+addHesCode(user : EndUser) : void
+removeHesCode(hesCode : HesCode) : void
<<Interface>>
CovidTestRepository
+getByID() : CovidTest[]
+getByDate(date : LocalDate) : CovidTest[]
+getByInterval(begining : LocalDate, end : LocalDate) : Report
+addNewTestResult(result : TestResult) : void
+addTestResultToAUser(user : EndUser, test : TestResult) : void
<<Interface>>
VaccardRepository
+getApprovedOnes()
+getDeclinedOnes()
+getAbroadOnes()
<<Interface>>
DormRepository
+getTheRoomsWithCovidPositiveStudents() : Student[]
+getTheNegativeStudents(room : Dorm) : Student[]
+addDorms(dorms : Dorm[]) : void
+addStudentToADorm(student : Student, dorm : Dorm) : void

3.4 Packages

3.4.1 Packages Introduced by Developers

3.4.1.1 Models

This package contains the classes, packages and enumerators required for the solution domain.

3.4.1.2 Repositories

This package contains the classes that extend the org.springframework.data.jpa.repository.JpaRepository interface which we need to generate queries that run on our sql database.

3.4.1.3 Controller

This package includes classes with required endpoints that handle according http get or post requests.

3.4.1.4 Services

This package has the service classes where our business logic occurs as well as the basic methods to handle controller classes appropriately.

3.4.1.5 Exception

This package has exception classes to handle exception occurs in service classes as well as controller classes

3.4.1.6 Security

This package contains the classes required to handle authentication and authorization.

3.4.2 External Library Packages

All of our dependency packages are added via maven package manager.

3.4.2.1 org.projectlombok

This library allows us to generate required functions and constructors of our objects with specific annotations such as @Data, @getter, @setter, @toString, @equals and @hash functions.

3.4.2.2 org.springframework.boot

This library allows us to build our project without concerning ourselves about managing server properties of a web application.

3.4.2.3 org.postgresql

This library allows our application to manage a given postgresql server with given declarations.

3.4.2.4 org.springframework.session

Without being tethered to an application container-specific solution, Spring Session makes it simple to handle clustered sessions.

3.4.2.5 org.springframework.security

This library enables security restrictions to the project.Its primary goal is to handle authorization and authentication at the Web request as well as the method invocation level.

3.4.2.6 org.springframework.jpa

This library enables Java to manage relational data. It allows the project to access and persist data between Java objects and the relational database.

3.4.2.7 org.springframework.web

This library enables us to build web applications including RESTful applications using Spring MVC.

3. Glossary & References

“Spring makes java simple.,” *Spring*. [Online]. Available: https://spring.io/. [Accessed: 28-Nov-2021].

E. Paraschiv, “Spring with Maven,” *Baeldung*, 09-Sep-2020. [Online]. Available: https://www.baeldung.com/spring-with-maven. [Accessed: 28-Nov-2021].

PostgreSQL, 28-Nov-2021. [Online]. Available: https://www.postgresql.org/. [Accessed: 28-Nov-2021].

H. Koushik, “AWS Fundamentals: Beginners Guide,” *Medium*, 13-May-2021. https://medium.com/age-of-awareness/aws-fundamentals-beginners-guide-ffea402596fb