Bilkent University

Department of Computer Engineering

InternHub

Design Report

May 5, 2023

Group: HASAT

Deniz Tuna Onguner	22001788
Anıl İlağa	22002044
Hasan Ege Tunç	22003814
Alper Göçmen	22002948
Sarper Arda Bakır	21902781

Instructor: Eray Tüzün
Teaching Assistants: Yahya Elnouby & Muhammad Umair Ahmed

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the design of the Object-Oriented Software Engineering course – CS319.

CONTENTS

1. Introduction....1
1.1. Purpose of the system....1
1.2. Design goals....1
1.2.1. Usability....1
1.2.2. Performance....2
1.2.3. Reliability....2
1.2.4. Security....3
1.3. Top two design goals....3
2. High Level Software Architecture....4
2.1. Subsystem Decomposition....4
2.1.1. Presentation Layer....5
2.1.1. Application Layer....6
2.1.2. Database Interface Layer....7
2.1.3. Data Layer....8
2.2. Deployment Diagram....9
2.3. Hardware/Software Mapping....10
2.4. Persistent Data Management....11
2.5. Object Access Matrix....12
2.6. Boundary Conditions....13
3. Low Level Design....15
3.1. Object Design Trade-Offs....15
3.2. Final Object Design....16
3.3. Packages....17
3.3.1. External Packages....17
3.3.2. Internal Packages....18
3.4. Class Diagrams....19
3.4.1. Application Layer....19
3.4.2. Data Layer....20
3.4.3. Presentation UI Layer....21
4. References....22

1. Introduction

1.1. Purpose of the system

InternHub is a web-based application designed to simplify the process of managing student internship reports and evaluations. The system enables students to upload their internship reports to a central repository where they can be reviewed and evaluated by their respective instructors. Also, the secretary takes crucial role in internship management system. The system facilitates the job of secretary by digitalizing the system and removing most of the paperwork. Compared to current system, this system eliminates the use of other sources, and gather all documents in one place. Thus, this system provides a convenient and efficient way to manage the entire process of internship evaluation.

1.2. Design goals

The design goals for this web-based application are determined with the help of requirement analysis report. InternHub aims to remove the paperwork and digitalize the process. This maximizes the usability and performance to save time and effort for the instructors and the secretary. The system must be safe and reliable since it contains the personal information of students and some information of the companies. The system must be protected against any malicious attacks, and it should handle the crashes and bugs.

1.2.1. Usability

The InternHub team offers a web-based application that facilitates the use of the current system both for the first-time users and staffs. To achieve this, we provide a simple interface that includes all the operations that can be done by users and components that are self-explanatory. Moreover, InternHub provides a feature that displays a deadline dates, and announcements that informs users about current events.

1.2.2. Performance

Due to the sensitive nature of its operations, it is crucial that the InternHub performs efficiently even under heavy workloads. To ensure optimal performance, we have implemented certain measures. For instance, we have utilized async calls in the backend, particularly when querying data from the database. Similarly, in the frontend, API calls to the backend are mostly done asynchronously. In addition, we have aimed to minimize the number of transactions when retrieving data from the database in our backend, to avoid bottlenecks. To achieve this, we have implemented the repository design pattern so that methods can retrieve data from the database with just one request in most cases. We have also conducted an analysis to select the appropriate infrastructure setup from AWS to further enhance the application's performance.

1.2.3. Reliability

As the internship management process for a student lasts for one or two semesters, InternHub must be highly reliable for the entire duration. For instance, the reports uploaded by students and the evaluation forms written by instructors and secretary should not be lost for enough time. To ensure that, InternHub provides sufficient amount data storage to its user. Also, the system should handle the huge data entrances. For example, in the deadline times, there could be lots of students who uploads their reports to the system. Thus, InternHub is designed to handle these kind of huge data entrances at a specific time.

1.2.4. Security

To safeguard the privacy of students and companies, InternHub give access to some private information such as company grades given to students to only authorized personnel such as secretary and instructor. Furthermore, only users that is added by super-user can access to the system to prevent unauthorized access.

1.3. Top two design goals

For the InternHub, the most significant two design goals were chosen as usability and reliability. Security is not in top two, since lots of information are not confidential. Thus, there is no need to implement highly secure web-based application.

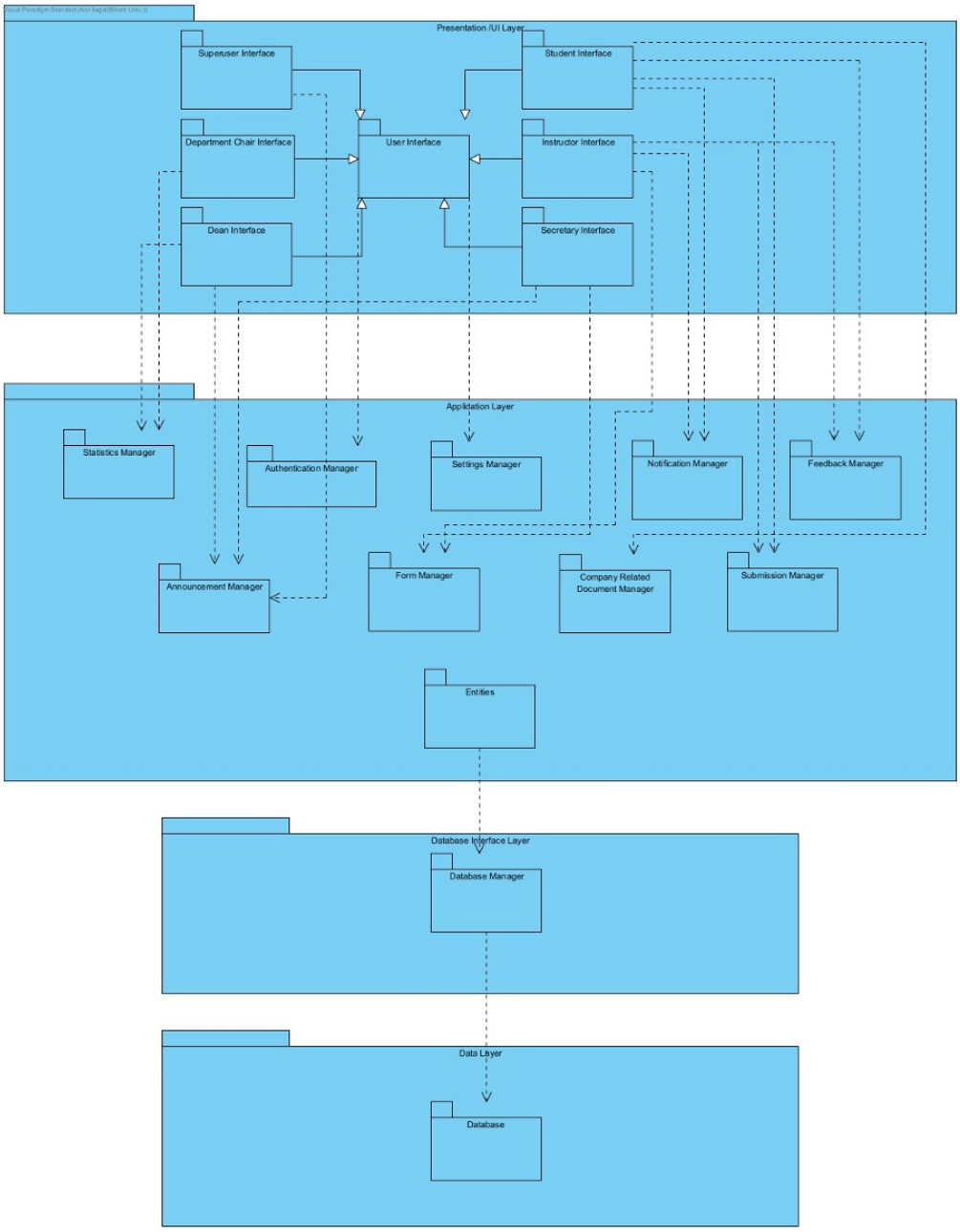
Secretary does a lot of job both with paperwork and digitalized documents using different platforms. Instructors are using these platforms for evaluation and making communication with secretary. The system aims to reduce the workload so it must be easy to use and adapt so we made user-friendly interfaces. Also, we gathered all process such as all necessary documents and information in one place. Therefore, InternHub provides an efficient workplace in terms of usability to its users.

The communication among students, instructors and secretary is a complex and long process. The reports and evaluation forms must be protected for enough time and data about these should not be lost. On the other hand, lots of students and instructors will use this system in one or two semesters until the evaluation done. Thus, on the deadlines, there could be huge data entrance to the system. InternHub provides necessary amount of data storage to protect forms and reports for necessary time. Also, the system handles huge data entrances at a specific time.

2. High Level Software Architecture

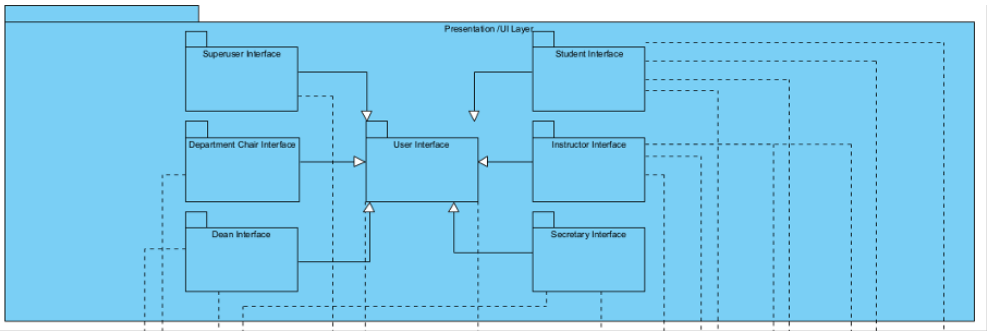
2.1. Subsystem Decomposition

We have implemented a four-layered architecture for our subsystem decomposition, comprising the Presentation Layer, Application Layer, Database Interface Layer, and Data Layer. This architectural approach divides the program structure into four distinct substructures, each encompassing a set of related packages. We believe that employing this architecture enhances the security, functionality, maintainability, and usability of our system.



2.1.1. Presentation Layer

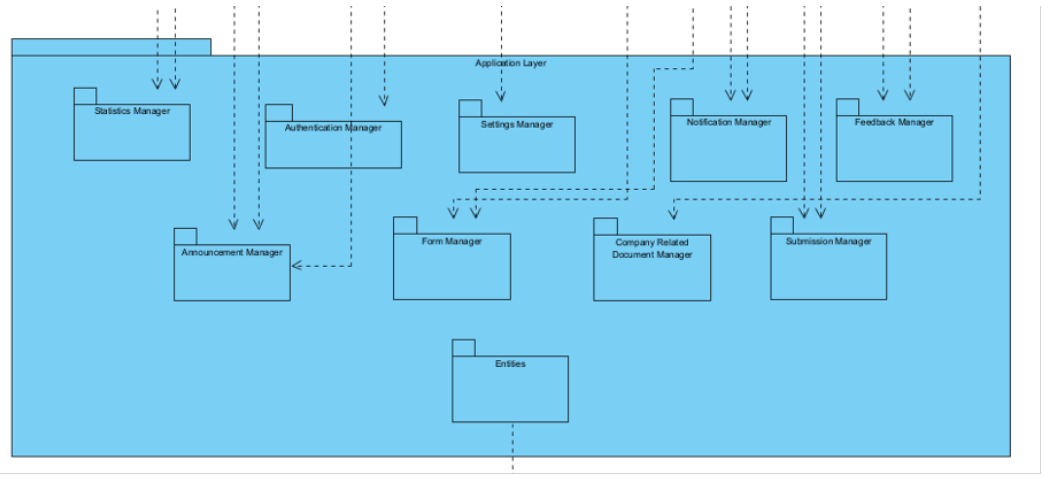
The Presentation Layer serves as the primary interface between the users and the system, facilitating their interaction. It allows users to perform various tasks such as uploading files and providing necessary inputs to the system. Once the inputs have been received, they are transmitted to the Application Layer for further processing.



The Presentation Layer comprises specific views for different user roles, including the User package, Superuser package, Dean package, Secretary package and Chair package. The General User package communicates with the Settings Manager and Authentication Manager, facilitating the management of user settings and user authentication. The Superuser, Dean, and Secretary packages can also communicate with the Announcement Manager, which is responsible for managing system announcements. In addition, the Dean and Chair interfaces communicate with the Statistics Manager, which provides statistical data related to the system.

2.1.1. Application Layer

The Application Layer comprises manager packages that can modify existing data using the data received from the Presentation Layer. These managers act as intermediaries between the Presentation Layer and the Database Manager, facilitating communication between the two.



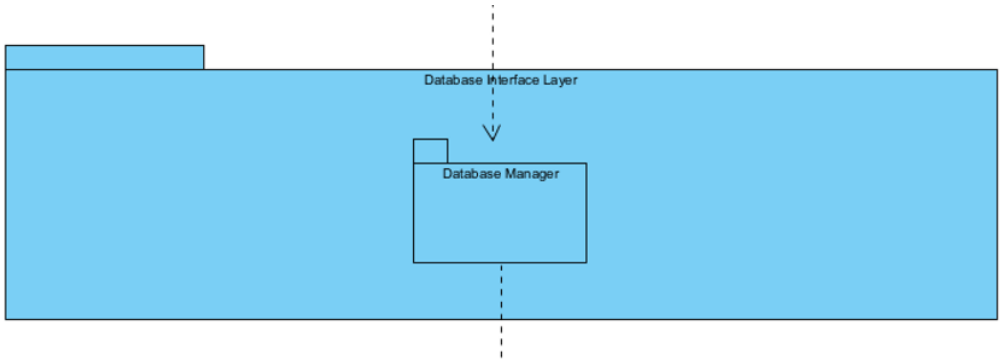
- **Statistics Manager**
  - Contains control objects related to statistics.
- **Announcement Manager**
  - Contains control objects related to make announcements.
- **Authentication Manager**
  - Contains control objects related to authentication.
- **Settings Manager**
  - Contains control objects related to settings.
- **Report Manager**
  - Contains control objects related to reports.
- **Form Manager**
  - Contains control objets related to forms.
- **Feedback Manager**
  - Contains control objets related to feedbacks.
- **Submission Manager**
  - Contains control objets related to submissions.

· **Entities**

- This package is responsible for defining the structure and behavior of the entities used by InternHub system.
- This package serves as the foundation of the system's data model and is a critical component of the overall project.

2.1.2. Database Interface Layer

The Database Interface Layer consists solely of the Database Manager package, which is responsible for managing database instances. This layer acts as a transactional intermediary between the Application Layer and the Data Layer, facilitating the exchange of data between the two layers. It provides a means for the Application Layer to interact with the Data Layer without having to deal with the intricacies of database management directly.



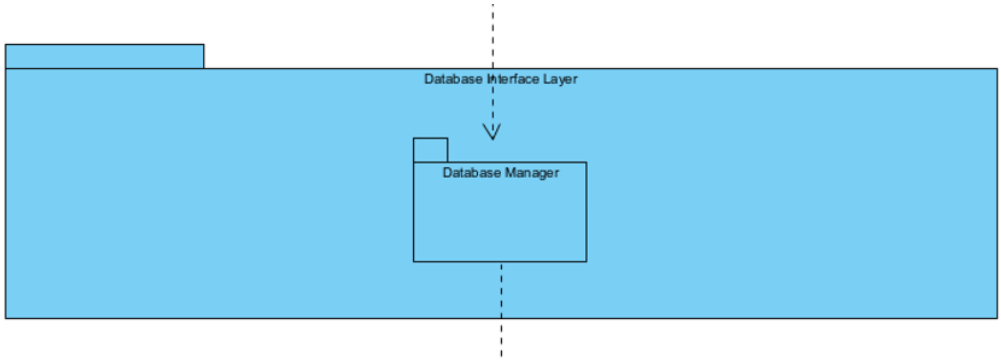
· **Database Manager**

- This package provides a set of methods and functions that allow the system to access and manipulate the data stored in the database.
- This package is responsible for managing database connections, handling data transactions, and executing database queries and commands.
- Serves as a bridge between the Application Layer and the Data Layer, allowing the two layers to communicate and exchange data without having to interact with the database directly.

- This package helps to ensure the security and integrity of the system's data by enforcing database constraints and managing access control.

2.1.3. Data Layer

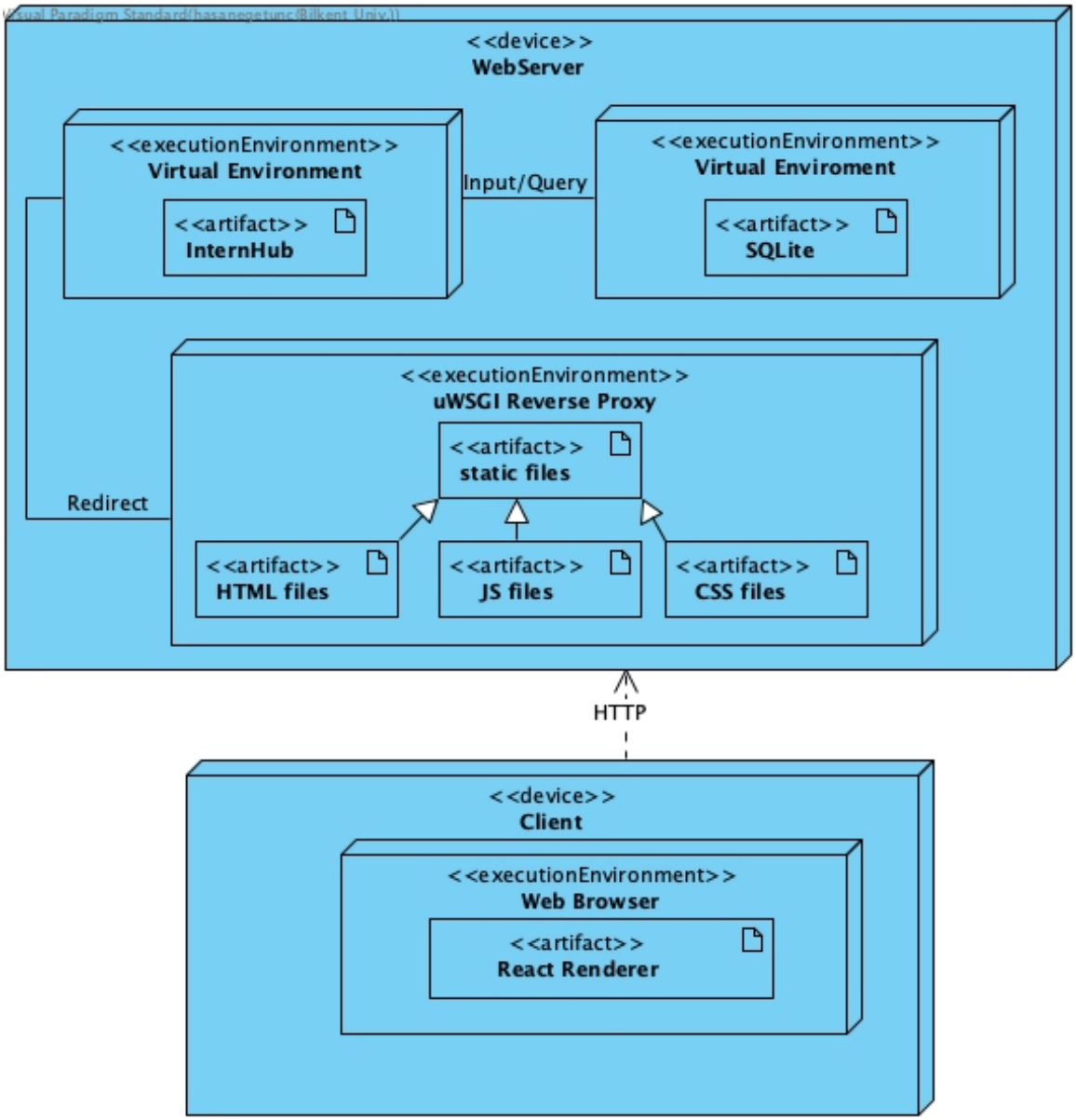
The Data Layer consists solely of the Database package, which is responsible for storing data. It represents the lowest level of the system architecture and provides the necessary functionality to access and manipulate data stored in the database. The Data Layer interacts with the Database Interface Layer through the Database Manager package, which facilitates the communication between the layers.



· **Database**

- The Database package is the lowest level of the system architecture, responsible for storing and retrieving data used by the system.
- This package provides a set of methods and functions that allow the system to interact with the database and manipulate the data stored within it.
- Data is organized and stored within the database according to a specific schema, which defines the structure of the database and the relationships between different tables and entities.
- The performance and scalability of the system are heavily dependent on the design and implementation of the Database package, which must be optimized to handle large volumes of data and high levels of traffic.

2.2. Deployment Diagram



InternHub is hosted on a web server, and this server also contains database to minimize required interactions to contribute the performance. Existence of a device that is able to run ES5 or newer JavaScript version on its web browsers is sufficient to use InternHub. Reverse proxy mechanism is benefitted to deliver the HTTP requests made by user and also to deliver the response given by system to the user again. In addition, reverse proxy presents required static files to users so that React Renderer can work.

2.3. Hardware/Software Mapping

InternHub is a small to medium-scaled web application that requires at least one server to host the backend of the InternHub and users must have a device with browser access/internet connection to use the application. As InternHub adopts the client-server model, it is optimized for modern browsers that support at least HTML5, CSS3, and ES5 on computers and mobile phones.

Since InternHub users frequently view reports and forms, the server should be powerful in terms of CPU and RAM. Moreover, InternHub is officially supported to run on a GNU/Linux environment, which can be another hardware specification constraint. Lastly, we should consider that the SQLite database runs on the same server as InternHub.

Considering the given requirements, we can pick an AWS EC2 instance with the following specifications:

m6g.xlarge:

- 4 vCPUs (AWS Graviton2 64-Core ARMv8 2.5 GHz)
- 16 GB RAM
- 500 GB SSD (For storing company related documents, report submissions, forms and logs.)

This setup is expected to handle up to 1650 concurrent users in a year, that is because, a total number of students admit to the engineering faculty is 550 in a year, and since 3rd and 4th grade students receive internship related courses we need to, at least, double this number, and also there must be some students that are taking these courses again, so we can triple 550 to set an upper bound. Each student is assumed to upload two internship reports, each of which is presumed to be 20MB in size, and other forms filled by secretary and instructor for a particular student is also assumed to be 10MB in size, then we need to store 50MB data for 1650 students in a year, so calculation gives nearly 80GB of SSD requirement for annual data storage. Assuming Bilkent University wants to store the reports of a particular student for 5 years, we can determine SSD size as 400GB. Since 1GB of SSD costs 0.10$ per month, so we have 40$ SSD cost and approximately $200 cost AWS EC2 intance cost, so total cost of 240$. Even though AWS is scalable, since we made an upper approximation, we will not possibly need rescaling.

As for the software stack, InternHub is built using the Django web framework, which runs on Python 3.10.3. The frontend is designed to work with HTML5 and CSS3. SQLite is used as the database solution for the development environment, as it does not require a running database server like MySQL or PostgreSQL, and the database is held in a single file, making it easy to track.

2.4. Persistent Data Management

SQLite is a popular choice for developers who need a lightweight and self-contained database management system. It is a relational database that uses SQL to manage and query data, making it a familiar choice for those who have worked with other SQL-based databases.

One advantage of using SQLite is that it does not require a separate server to run. Instead, the database is stored in a single file on the local machine, making it easy to manage and share among our team members. This also makes it a good choice for small to medium-sized projects that do not require a high level of traffic. Considering this fact and regarding that our project will encompass the engineering students only, SQLite seems to be plausible choice .

Another advantage of SQLite is its simplicity. It is easy to set up and does not require much configuration, making it a good choice for our team to increase progress speed.

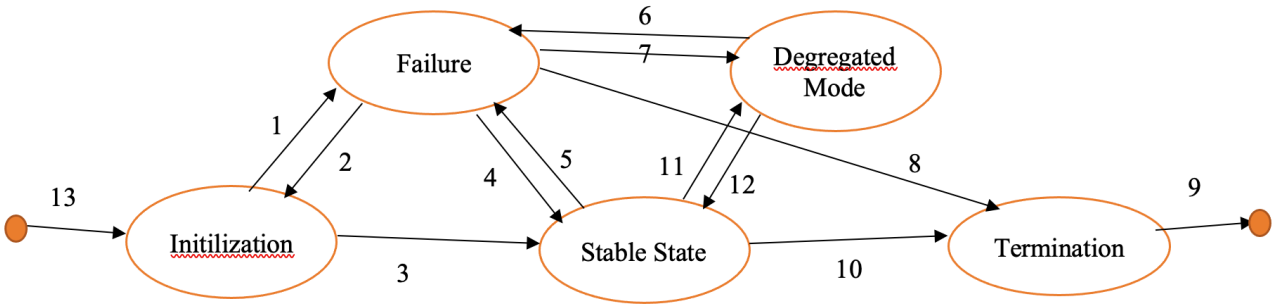
In addition, SQLite also supports many of the same data types and commands as other SQL-based databases, making it a versatile choice for a wide range of applications. And since it is an open-source database, there are many resources available for developers who need help or guidance.

Finally, since Django framework will be planned to be used in the project, and this framework is most popularly used with SQLite database, and initializes a project connected to this database, SQLite outstands among other databases. So, we decided using SQLite as a database in our project.

2.5. Object Access Matrix

	Authentication	Profile	Notification	Announcement	Placement	CompanyForm	Submission	Feedback	View	Settings
InternHubUser	login() forgetPassword()	-	-	-	-	-	-	-	-	-
Student	login() forgetPassword()	getRole() addContact() removeContact() getContact()	getMessage() getDate() viewNotification()	getContent() getDate() viewAnnouncement()	-	applyForCompanyApprovalValidation() requestCompany() evaluateCompany()	uploadReport() deleteReport() saveReport() editReport() requestExtension() viewSubmission()()	viewFeedback() getStatusMessage() getGivenDate() getSubmission()	view_companies_list() view_instructor() viewNotification() viewSubmission() viewFeedback() viewAnnouncement()	changePassword()
Instructor	login() forgetPassword()	getRole() addContact() removeContact() getContact()	getMessage() getDate() viewNotification()	getContent() getDate() viewAnnouncement()	-	-	edit_WEF() save_WEF() create_WEF() submit_WEF() edit_GF() save_GF() create_GF() submit_GF() create_submission() set_due_date()	give_feedback() getGivenDate() getSubmission()	view_companies_list() view_students() viewNotification() view_submissions() view_feedbacks() view_WEFs() view_GFs() viewAnnouncement()	changePassword()
DepartmentSecretary	login() forgetPassword()	setRole() getRole() addContact() removeContact() getContact()	getMessage() getDate() viewNotification()	getContent() getDate() makeAnnouncement() setContent() setDate() viewAnnouncement()	reassingStudentToInstructor() deleteStudentFromInstructor() assignStudentToInstructor()	add_company() remove_company() approve_company_request() reject_company_request() approveCAVA() rejectCAVA() markedAsConsideredCEBS() createConfidentialCompanyForm() submitConfidentialCompanyForm()	create_submission() set_due_date()	-	view_companies_list() view_students() view_instructor() viewStatistic() viewAnnouncement()	changePassword()
Dean	login() forgetPassword()	getRole() addContact() removeContact() getContact()	getMessage() getDate() viewNotification()	getContent() getDate() makeAnnouncement() setContent() setDate() viewAnnouncement()	-	-	-	-	view_companies_list() view_students() view_instructor() viewStatistic() viewAnnouncement()	changePassword()
Chair	login() forgetPassword()	getRole() addContact() removeContact() getContact()	getMessage() getDate() viewNotification()	getContent() getDate() makeAnnouncement() setContent() setDate() viewAnnouncement()	-	-	-	-	view_companies_list() view_students() view_instructor() viewStatistic() viewAnnouncement()	changePassword()
SuperUser	login() forgetPassword()	setRole() getRole() addContact() removeContact() getContact()	getMessage() getDate() viewNotification()	getContent() getDate() makeAnnouncement() setContent() viewAnnouncement()	add_user() delete_user()	-	-	-	view_companies_list() view_students() view_instructor()	changePassword()

2.6. Boundary Conditions



Checklist for #1
- Did any sudden database interruption occur?
- Did any uSWGI deployment problem occur?

Checklist for #2
- Can any sudden database interruption be fixed?
- Can uSWGI be redeployed again?

Checklist for #3
- Is database connection maintained appropriately?
- Can users utilize the functionality appointed to them from the system without any trouble?
- Can HTTP requests be handled conveniently?
- Can HTTP response be sent back to the users as expected?
- Does reverse proxy method work correctly?

Checklist for #4
- Can database connection be fixed?
- Did server or host fix its troubles?
- Did the storage expand from host?

Checklist for #5
- Were database connection lost?
- Were the server or host experiencing troubles?
- Did the storage reach its allocated amount?

Checklist for #6
- Did malicious attack penetrate the system?
- Did plugin issue even make the server unaccessible?

Check list for #7
- Were malicious attack solvable?
- Can the server be made accessible again?

Checklist for #8
- Can malicious attack not be solved?
- Is the system unable to be made accessible to the users?
- Is the database connection unable to be established?
- Are the troubles experienced by users unsolvable?

Checklist for #9
- Will the system be totally shut down?

Checklist for #10
- Is database connection interrupted intentionally?
- Are the functionalities given to users hindered by choice?
- Is the system not receiving any HTTP requests?
- Is the reverse proxy method not working anymore?
- Is uSWGI undeployed deliberately?

Checklist for #11
- Did some malicious attacks happen (DDOS)?
- Did some plugin issue occur?
- Did a lot of bandwidth – data transfer occur at once?
- Are users unable to make uploads to the system?

Checklist for #12
- Can website be protected from malicious attacks?
- Cab plugin issues be fixed?
- Can bandwidth – data transfer amount be reduced?
- Is the system open to uploads from users?

Checklist for #13
- Is database connected to server?
- Is run server command executed?
- Is contact environment initialized?
- Is uSWGI deployed?

3. Low Level Design

3.1. Object Design Trade-Offs

I. **Usability** versus **Security:** No two gate authentication is preferred in InternHub; that is because, there is no private data requiring strict security checks is stored in the system. Absence of two gate authentication increases usability by allowing quick log of user to the system, however, reduces security.

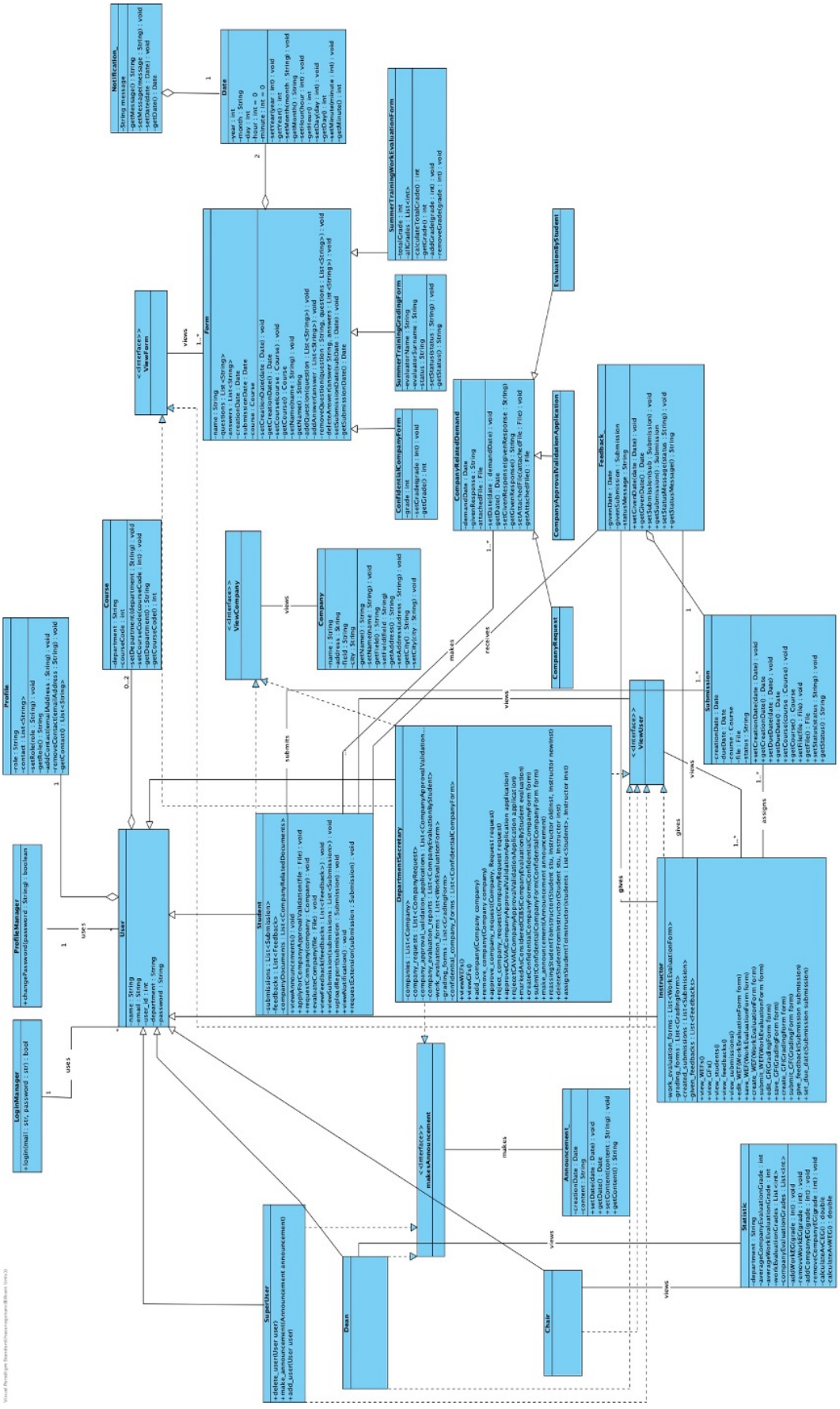
II. **Functionality** versus **Usability:** InternHub allows students to not only upload their internship reports to the system and receive feedback from instructors, but also makes it feasible for students to request company related issues, such as company approval validation application, evaluating company and requesting addition of company into the company list.

III. **Maintainability** versus **Performance:** Object oriented design of InternHub decreases performance of the system because managers in application layer needs to interact with entities which reduces performance, but increases maintainability since the system is decomposed into sub-compartments.

IV. **Reliability** versus **Cost:** Using of AWS EC2 instance and allocating big SSD in size to store multiple copies increase reliability, but also increases cost which is calculated as 240$.

V. **Functionality** versus **Performance:** Students capability to upload various reports and instructors ability to create multiple forms increase functionality, but reduces performance since created documents overload the database.

3.2. Final Object Design



3.3. Packages

3.3.1. External Packages

Django: A high-level web framework for Python is provided by this package, allowing for the quick building of safe and dependable websites. To control backend logic and communicate with both the React frontend and the SQLite database, we have selected Django [1].

React: This package is used to build the application's front end and provides a declarative and effective method for developing user interfaces [2].

SQLite: We utilize this package as our preferred database option since it provides a compact and serverless solution for preserving data pertinent to the system's defined entities. Django and SQLite collaborate on data management [3].

Django Storage and Django Storages: These packages offer an abstraction layer for handling files, support for various storage backends, and quick file uploading and downloading [4].

Axios: This package provides a user-friendly and effective method for data collecting through HTTP calls to our Django backend, which is used in our React frontend [5].

React RouterThe package provides robust routing functionality for our React application, enabling it to manage navigation and render components based on specific URLs [6].

AWS: The package is utilized to establish attachment properties between users and remote servers [7].

3.3.2. Internal Packages

3.3.2.1. User Screens Package: This package is the parent of all boundary classes.

3.3.2.1.1. Dean Screens Package: This package involves all boundary classes of dean.

3.3.2.1.2. Students Screens Package: This package involves all boundary classes of students.

3.3.2.1.3. Instructor Screens Package: Involves all boundary classes of instructors.

3.3.2.1.4. Secretary Screens Package: Involves all boundary classes of secretary role.

3.3.2.1.5. Chair Screens Package: Involves all boundary classes of chair role.

3.3.2.1.6. Superuser Screens Package: Involves all boundary classes of superusers.

3.3.2.2. Announcement Package: Involves all boundary classes related to the announcements.

3.3.2.3. Notification Manager Package: Involves all boundary classes related to the notifications.

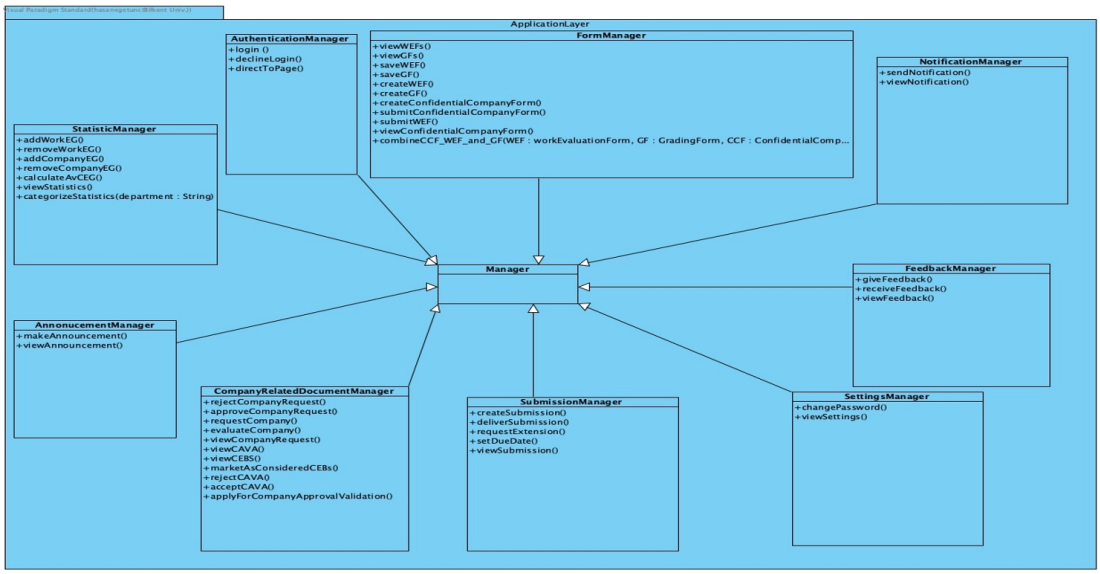
3.3.2.4. Authentication Manager Package: Involves all boundary classes related to the user authentication.

3.3.2.5. Settings Manager Package: Involves all boundary classes related to the settings.

3.3.2.6. Statistics Manager Package: Involves all boundary classes related to the statistics.

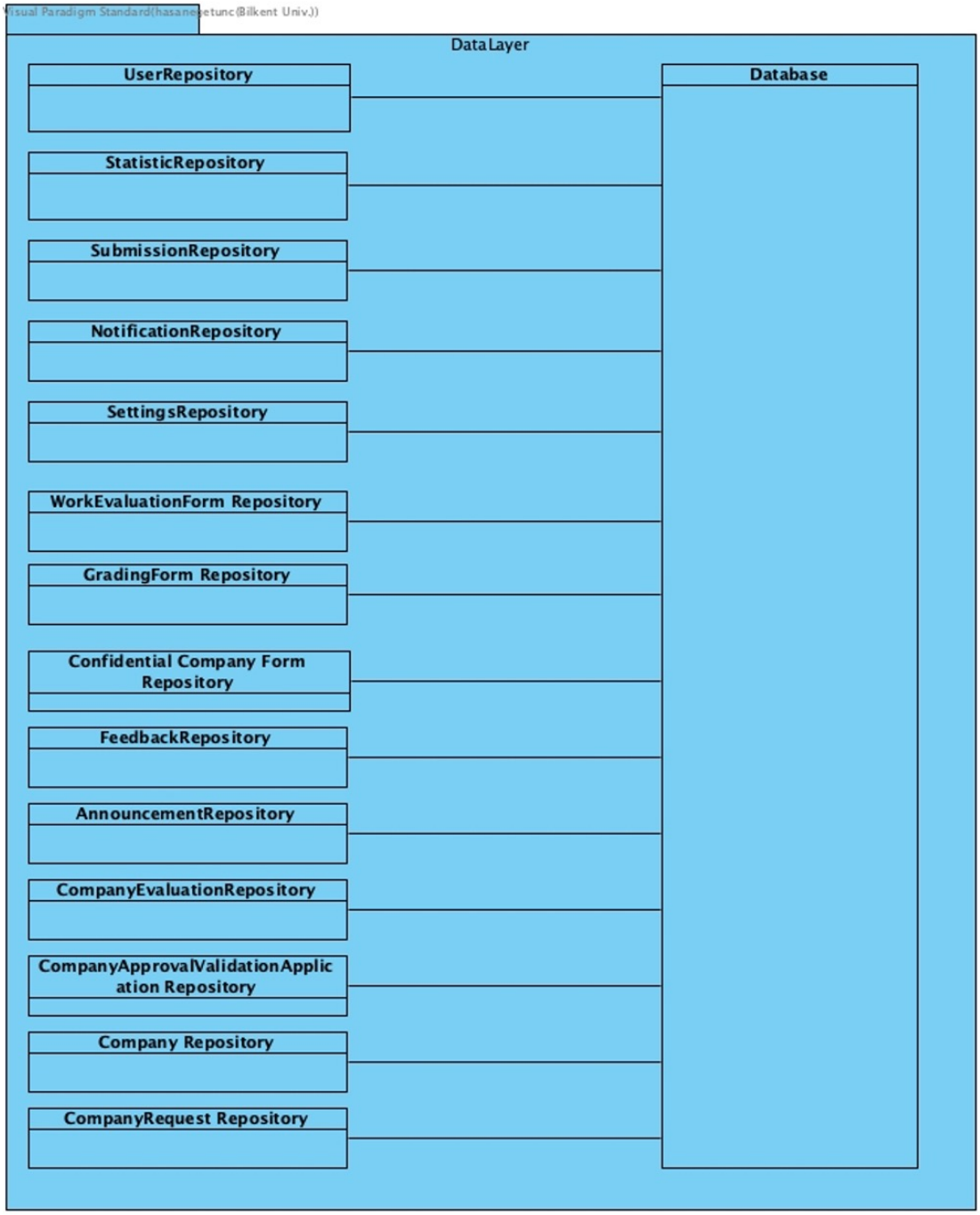
3.4. Class Diagrams

3.4.1. Application Layer

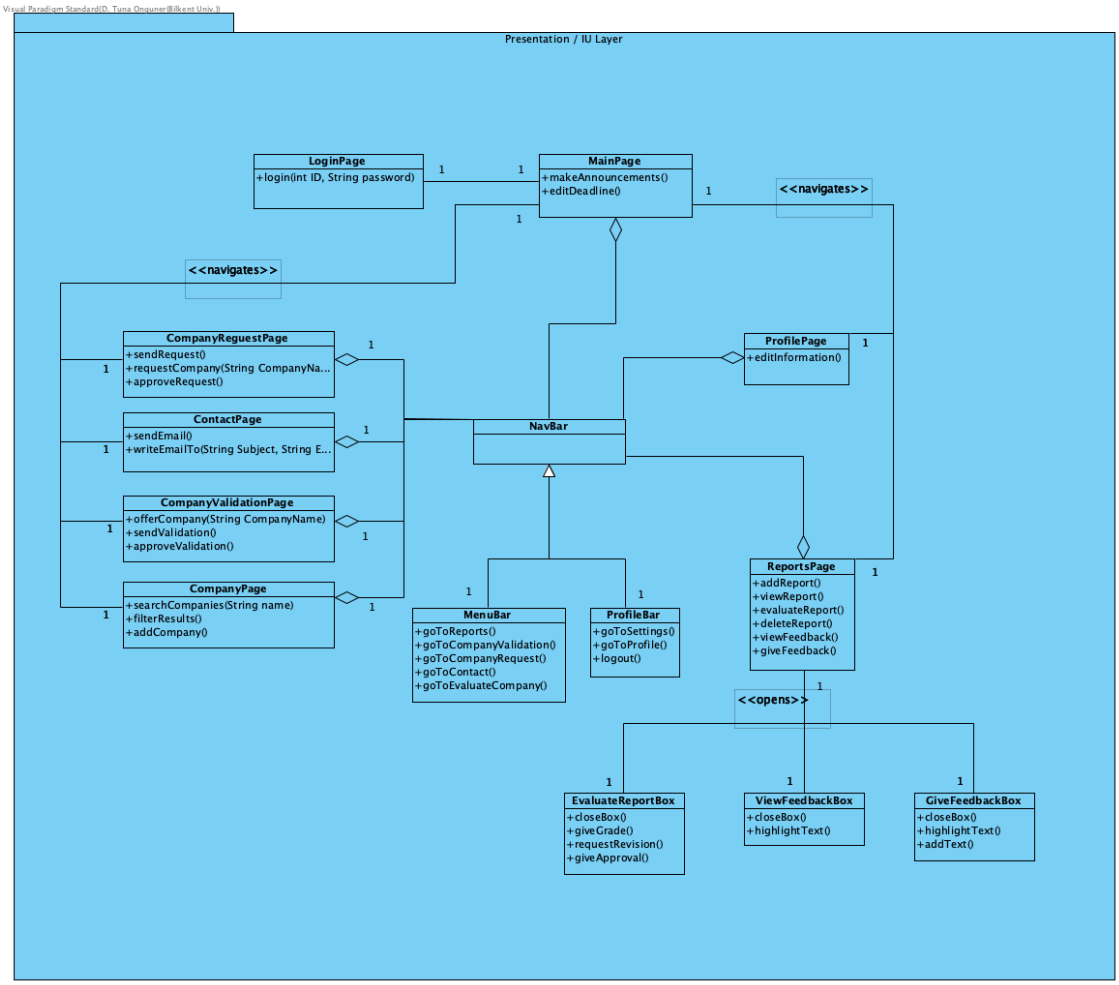


Methods included in manager classes given above. Note that, nearly all of above methods are contained in the final object design within the some classes that are supposed to fulfill the corresponding particular action or shown with associations. Users interact with the presentation layer, then, presentation layer interacts with the entities through these managers. Finally, it is important to note that entities including these methods are only the users, because users of the application also become entities that store relevant data regarding their behaviors.

3.4.2. Data Layer



3.4.3. Presentation UI Layer



4. References

[1] “Django,” *Django Project*. [Online]. Available: https://docs.djangoproject.com/en/3.2/. [Accessed: 05-May-2023].

[2] *React*. [Online]. Available: https://reactjs.org/. [Accessed: 05-May-2023].

[3] *SQLite documentation*. [Online]. Available: https://www.sqlite.org/docs.html. [Accessed: 05-May-2023].

[4] “Storages¶,” *django*. [Online]. Available: https://django-storages.readthedocs.io/en/latest/. [Accessed: 05-May-2023].

[5] “Promise based HTTP client for the browser and node.js,” *Axios*. [Online]. Available: https://axios-http.com/. [Accessed: 05-May-2023].

[6] “Home V6.11.1,” *Home v6.11.1*. [Online]. Available: https://reactrouter.com/. [Accessed: 05-May-2023].

[7] “Cloud computing services - amazon web services (AWS).” [Online]. Available: https://aws.amazon.com/. [Accessed: 05-May-2023].