Bilkent University

Department of Computer Engineering

CS 319 Term Project Internship Management System

Project short-name: InternHub

Design Report

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

Instructor: Eray Tüzün
Teaching Assistant(s): Yahya Elnouby & Muhammad Umair Ahmed

Design Report
May 21, 2023

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

CONTENTS

1.0 Introduction

1.1 Purpose of The System

1.2 Design Goals

2.0 High-Level Software Architecture

2.1 Subsystem Decomposition

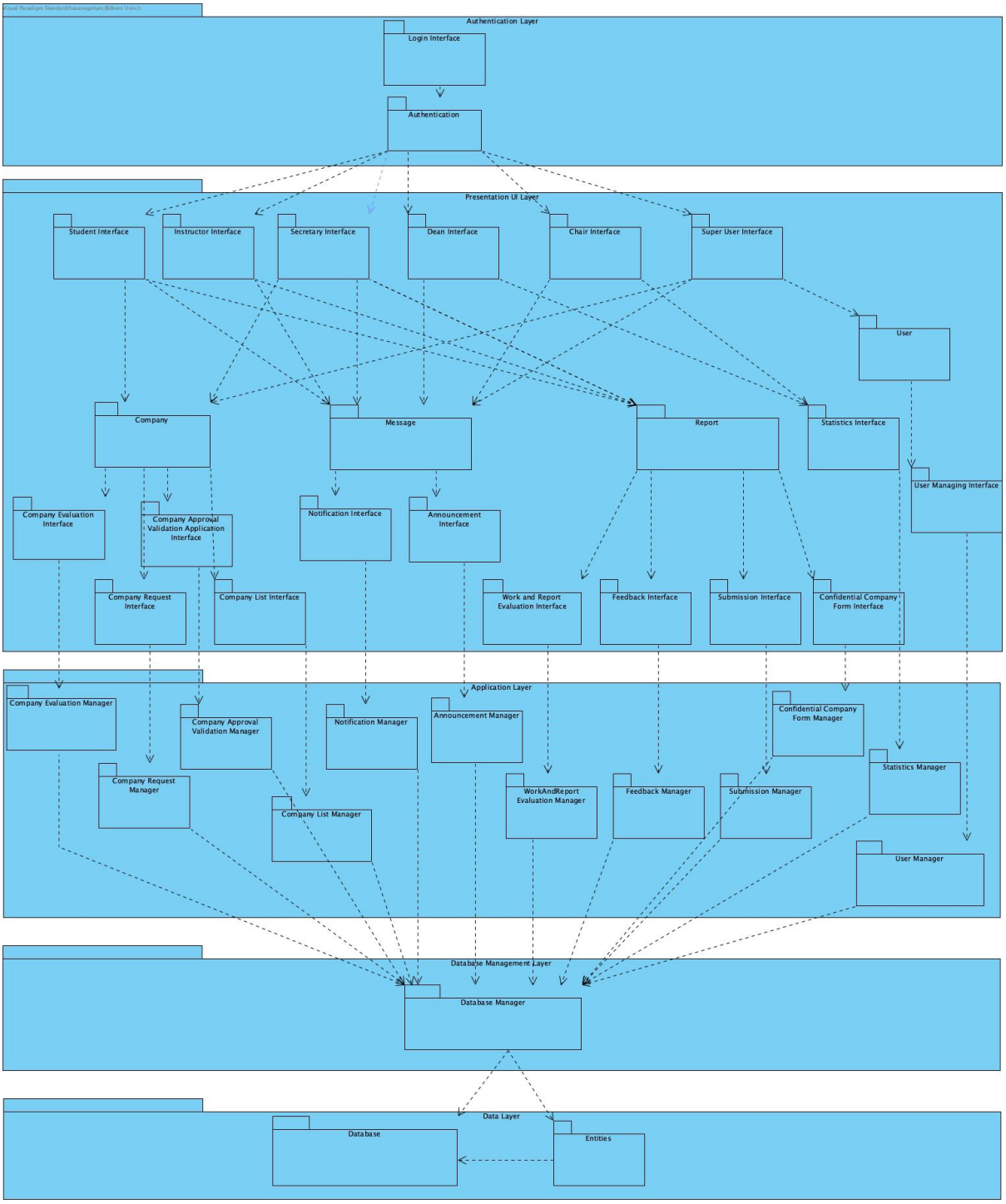

Figure-1: *Subsystem Decomposition Diagram of InternHub*

For high resolution please refer to the link below:
https://drive.google.com/file/d/16XdNalifSSlFuoRRawyom2DtTLWgpoP2/view?usp=share_link

For the internship management system, five layered architectural is implemented as seen above that decomposes system into layers that are authentication layer, presentation layer, application layer, database management layer, and data Layer from top to bottom. This architectural approach divides the program structure into five distinct substructures, each possessing its own classes. We believe that employing this architecture enhances the security, functionality, maintainability, and usability of the system, all of the nonfunctional requirements will be mentioned below as the focus of layers concern them.

2.1.1 Authentication Layer

Authentication layer of InternHub consists of a login interface and authentication subsystem. InternHub users encounter with the login page, and depending on the data provided by the authentication subsystem directs either them to the proper interface regarding their roles; thereby, connecting the authentication layer to the presentation layer or not allowing authorization by displaying the right message (i.e. user does not exist). Hence, the authentication layer serves for security.

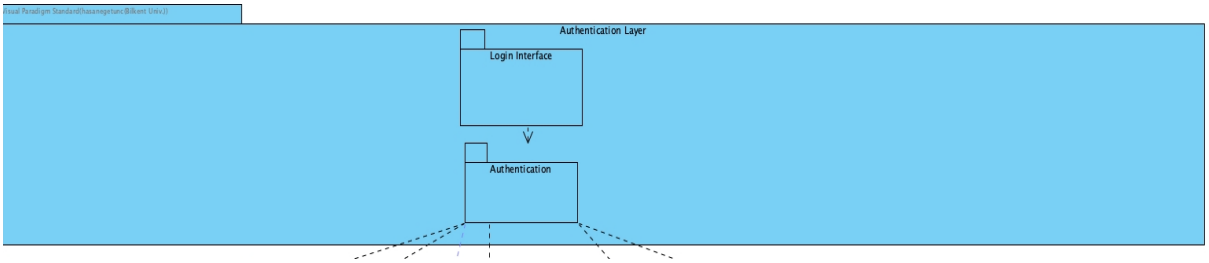

Figure-2: *Authentication Layer of InternHub*

2.1.2 Presentation UI Layer

Presentation layer of InternHub includes related user interfaces for all actors. User interfaces communicate with the subsystems -Company, Messages, Report, and User- depending on the actor type. See the dependency rows below for a better understanding. Each subsystem has its own interfaces, for example, company subsystem has company evaluation, company request, company list, and company approval validation application interfaces, which may alternate in functionality depending on the actor. To illustrate, the company approval validation application interface sends a form to the student to carry out the application; on the other hand, it displays the list of all applications made to the department secretary. Indeed, the two of them could be presented as independent interfaces, but since they belong to the same functionality of the same subsystem, it is preferred in this way. A similar argument holds for other subsystems as well. For instance, the feedback interface cannot be the same for the instructor and student. Finally, the user subsystem is where user entities are kept and manipulated (i.e. creation or deletion), which can only be performed by the superuser. To conclude user interfaces communicate with other subsystems via interfaces provided by these subsystems' interfaces, so the presentation layer serves for functionality and usability.

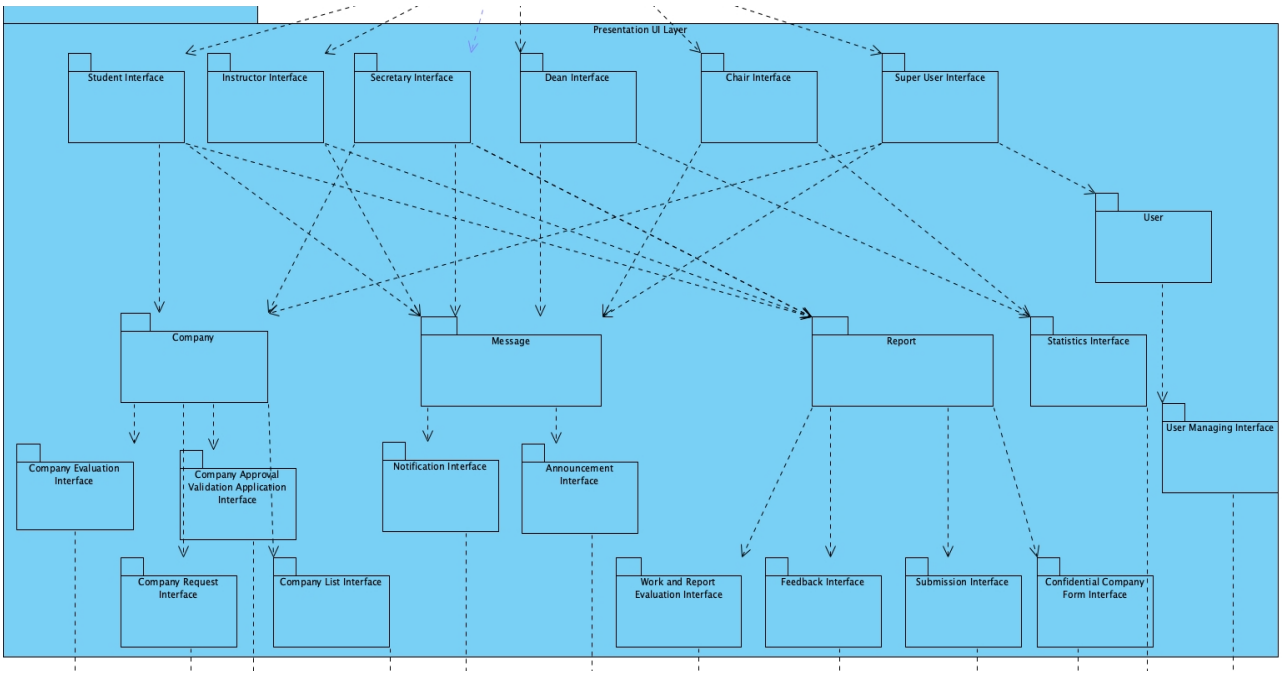

Figure-3: *Presentation Layer of InternHub*

2.1.3 Application Layer

Application layer is where the management of operations carried through interfaces in the presentation layer occurs. In other words, backend operations are performed in this layer. For instance, when an appropriate actor makes an announcement by clicking the related button, the announcement manager decides how to create this announcement, where to store it, and on which users' home page it should be displayed, with which data is included in it. In short, the application layer manages the operations performed on user interfaces, so the application layer serves for functionality

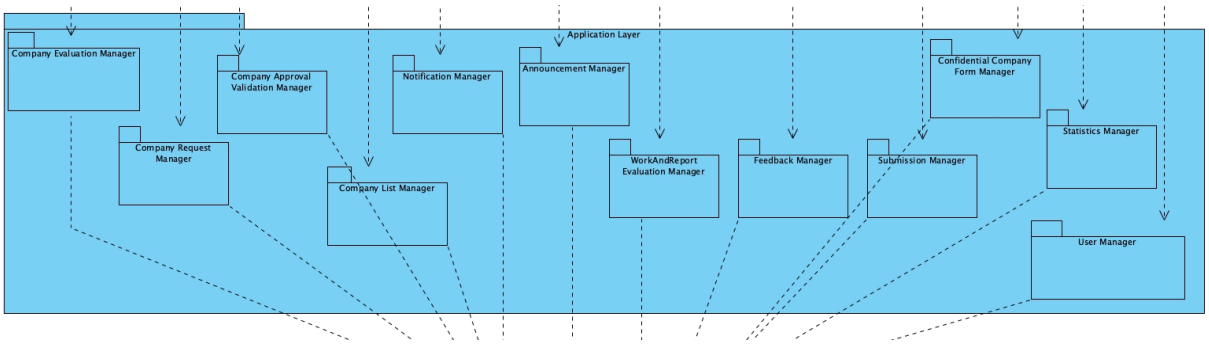

Figure-4: *Application Layer of InternHub*

2.1.4 Database Manager Layer

As illustrated in the previous section with an announcement example, the application layer may want to create new objects and store them properly. Obviously, these are database operations, so the application layer needs to communicate with the database regarding all its managers. Instead of communicating with each manager to the database separately, they can talk to a single database manager, which then performs the changes on

the database and entities. In other words, this layer acts as a transactional intermediary between the application and data layers, facilitating the exchange of data between the two layers. Indeed, this layer serves for maintainability because when an update is required on the database, making necessary changes on the database manager is sufficient, which prevents one from changing every single manager in the application layer.

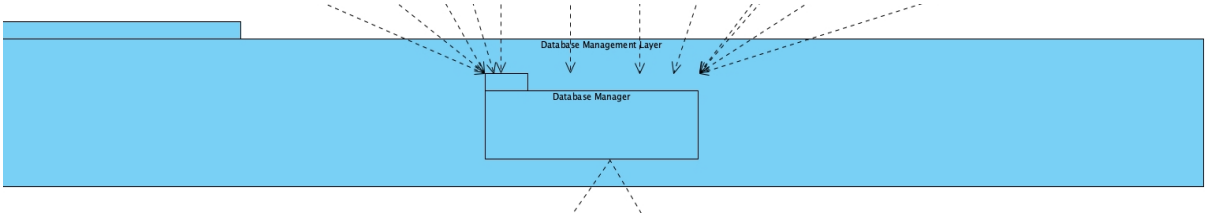

Figure-5: *Database Manager Layer of InternHub*

2.1.5 Data Layer

Data layer is the base layer on which all layers are constructed, and it performs changes on the existing database and entities based on the commands reaching it from the database manager.

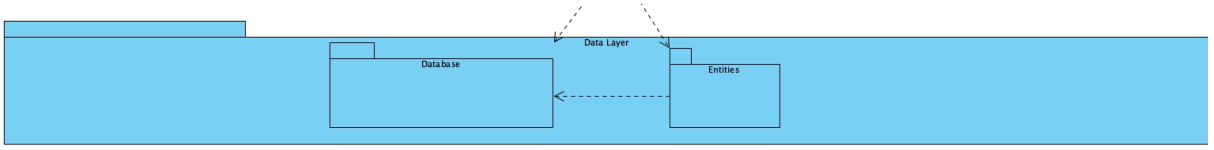

Figure-6: *Data Layer of InternHub*

2.2 Deployment Diagram

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 the total number of students admitted to the engineering faculty is 550 in a year. Since 3rd and 4th-grade students receive internship-related courses, we need to double this number at least, and also, there must be some students retaking these courses, 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 the secretary and instructor for a particular student are also assumed to be 10MB in size. Then we need to store 50MB of data for 1650 students a year, so the calculation gives nearly 80GB of SSD requirement for annual data storage. Assuming Bilkent University wants to store the reports of a particular student for five years, we can determine SSD size as 400GB. Since 1GB of SSD costs 0.10$ per month, we have 40$ SSD cost and approximately $200 cost AWS EC2 instance cost, so the total cost of 240$. Even though AWS is scalable, since we made an upper approximation, we will not possibly need rescaling.

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. 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 high traffic. Considering this fact and that our project will encompass engineering students only, SQLite seems to be a 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, many resources are available for developers who need help or guidance.

Finally, SQLite outstands among other databases because Django framework is being used in the project, and this framework is most popularly used with SQLite database by initializing a project connected to this database. In addition, Django reflects the changes made on object classes to existing databases with 2 to 3 lines of code. So, we decided to use SQLite as a database for the project.

2.5 Object Access Matrix

	Authentication	Profile	Notification	Announcement	Placement	CompanyForm	Submission	Feedback	View	Settings
InternHubUser	login()	-	-	-	-	-	-	-	-	-
Student	login() forgetPassword())	add_contact() remove_contact()	receiveNotification() filter_last_five_notifications()	filter_last_five_announcements_in_department()	-	apply_for_company_approvalvalidation() request_company() evaluate_Company() filter_companies_by_department()	make_submission() request_extension()	get_given_feedbacks() get_status_of_last_feedback()	ViiewCompaniesList() view_notifications() view_submissions() view_feedbacks() view_announcementst()	changePassword()
Instructor	login() forgetPassword())	add_contact() remove_contact()	receiveNotification() filter_last_five_notifications()	filter_last_five_announcements_in_department()	-	CreateWorkAndReportEvaluationForm UpdateWorkAndReportEvaluationForm	set_deadline() assign_submission()	give_feedback() set_feedback_status()	view_students() view_notification() view_submissions() view_WEFs() view_announcement()	changePassword()
DepartmentSecretary	login() forgetPassword())	add_contact() remove_contact()	receiveNotification() filter_last_five_notifications()	makeAnnouncement() filter_last_five_announcements_in_department()	assign_instructor_to_internship() assign_instructors_to_internships_randomly()	add_company() remove_company() approve_company_request() reject_company_request() filter_company_requests_by_department() approveCAVA() rejectCAVA() filter_cava_by_department() mark_company_evaluation_as_considered() CreateConfidentialCompanyForm UpdateConfidentialCompanyForm	-	-	view_companies_list() view_students() view_instructor() view_Announcement() view_WREs() view_CCFs() view_final_PDFs()	changePassword()
Dean	login() forgetPassword())	add_contact() remove_contact()	receiveNotification() filter_last_five_notifications()	makeAnnouncement() filter_last_five_announcements_in_department()	-	-	-	-	view_students() view_instructor() view_statistic() view_announcemenst()	changePassword()
Chair	login() forgetPassword())	add_contact() remove_Contact()	receiveNotification() filter_last_five_notifications()	makeAnnouncement() filter_last_five_announcements_in_department()	-	-	-	-	view_companies_list() view_students() view_instructor() view_statistics() view_announcements()	changePassword()
SuperUser	login() forgetPassword())	add_contact() remove_contact()	receiveNotification() filter_last_five_notifications()	makeAnnouncement() filter_last_five_announcements_in_department()	create_user() delete_user()	-	-	-	view_companies_list() view_students() view_instructor() view_announcements() view_users()	changePassword()

Figure-X: *Object Access Matrix of InternHub*

In the above diagram, parameters of methods are not shown, the application layer (3.4.2) and the presentation layer (3.4.3) can be referred to see the parameters. Camel case notations imply that Django classes are used to achieve related functionality.

3.4 Class Diagrams

Diagrams will be represented in this section based on the separation of layers. Each layer will appear with the classes they involve. This section can be considered as partitioning of the final object design (3.2).

3.4.1 Application Layer

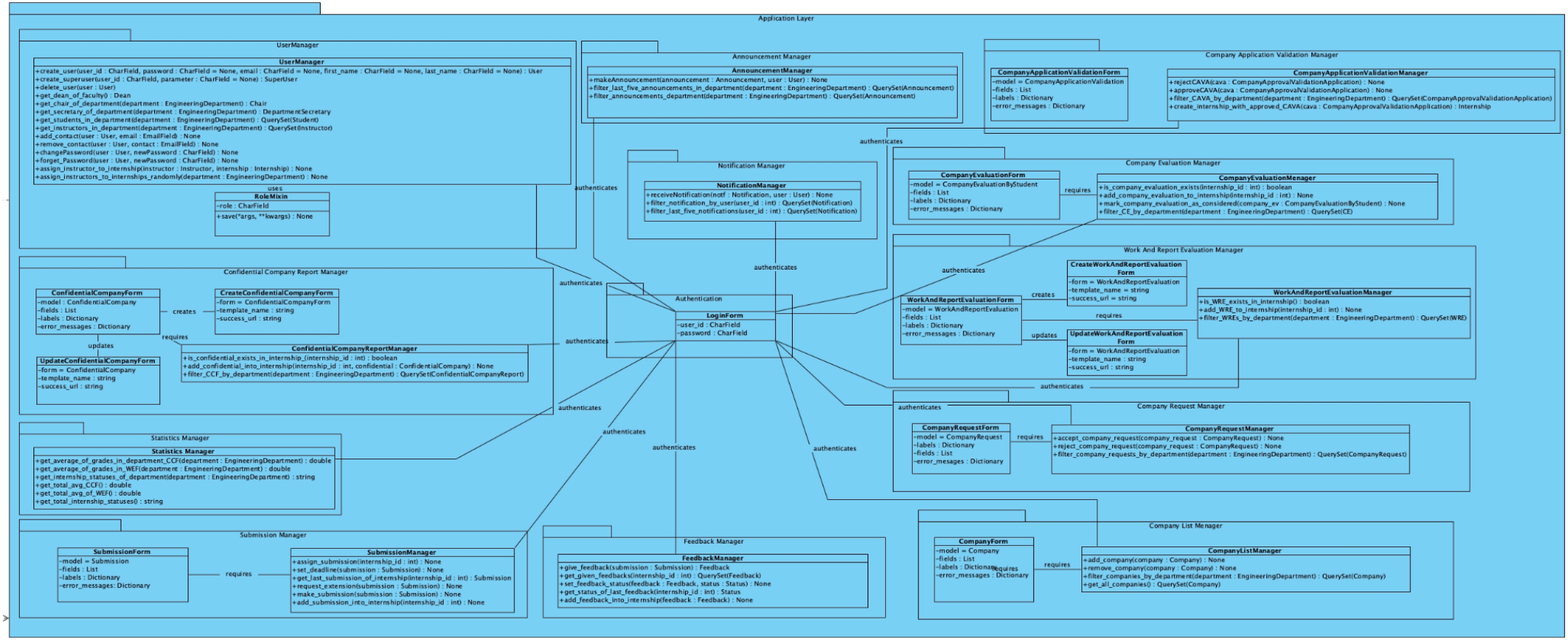

Figure-X: *Application Layer of InternHub*

Recall that in the subsystem decomposition diagram, the application layer was divided into the managers (2.1.3). The above diagram illustrates the classes involved in these managers with the functions provided. Authentication serves as the base, that is because, without valid authorization, none of the above managers will be accessible.

3.4.2 Data Layer

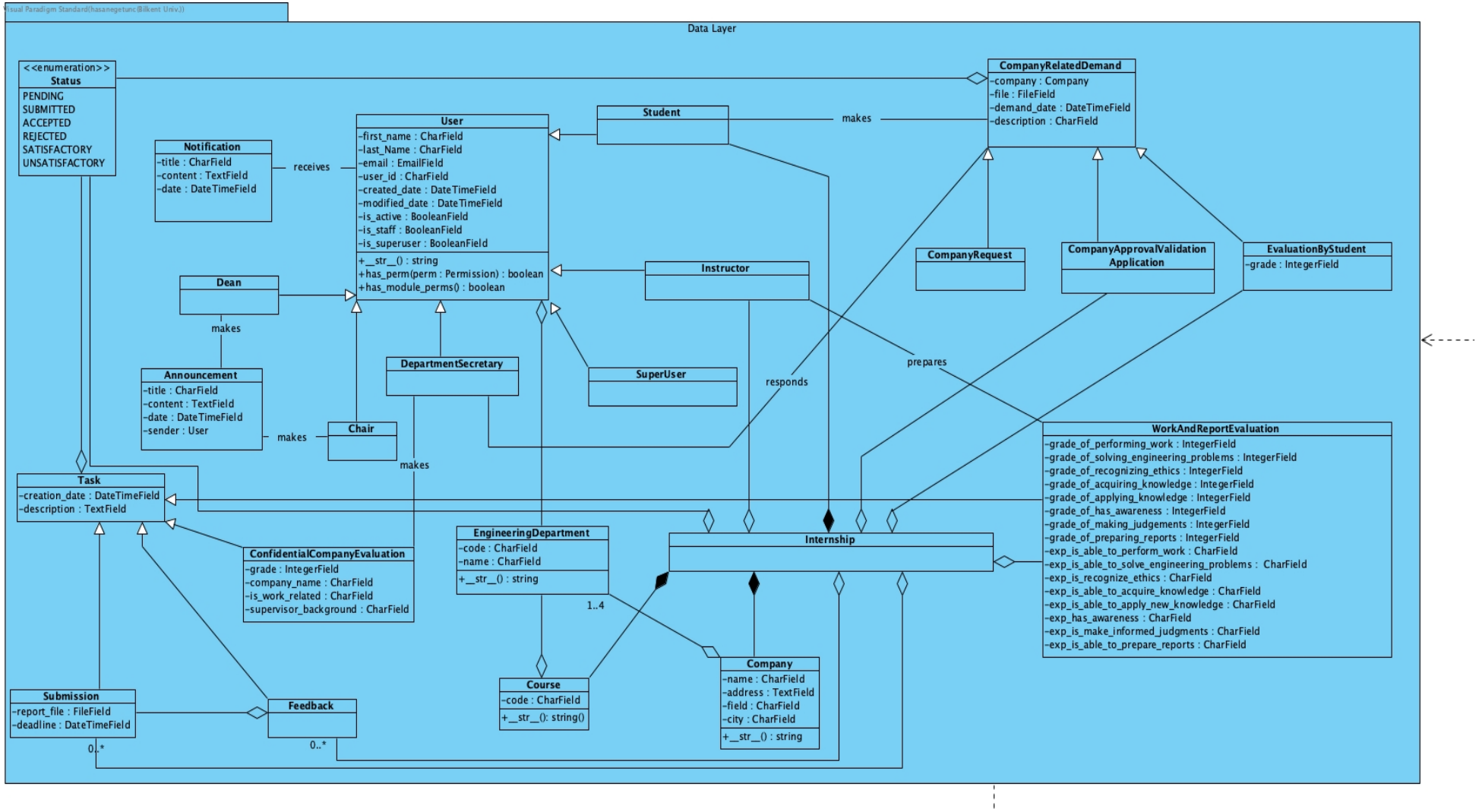

Figure-X: *Data Layer of InternHub*

The above diagram illustrates the data layer of InternHub, this layer's class diagram consists of entity classes. Django's model class data types are used to denote the variable types.

3.4.3 Presentation UI Layer

3.5 Design Patterns

4.0 Improvement Summary

5.0 Glossary

AWS: Amazon Web Services
CAVA: Company Approval Validation Application
CCF: Confidential Company Form
CE: Company Evaluation (by Student)
CSS: Cascading Style Sheets
ES5: ECMAScript 5
EC2: Elastic Compute Cloud **GNU:** A free and open-source operating system and software ecosystem developed by the GNU Project. **HTML:** Hypertext Markup Language **RAM:** Random Access Memory **SSD:** Solid-State Drive **vCPU:** Virtual Central Processing Unit **WRE:** Work and Report Evaluation

6.0 References