Chapter 1 - Introduction

The final year project is the cornerstone of any undergraduate academic program, serving as the final evaluation for the award of a degree. Its importance is further highlighted by the fact that it represents the crystallization of knowledge of a graduate throughout their academic journey so far, and hence serves as a tangible demonstration of their skills to industry recruiters or for the pursuit of higher graduate studies.

At the University of Mauritius (UoM), final year projects are traditionally proposed either by the faculty staff or by the students themselves, and are then allocated accordingly, to a supervisor-student pair. This supervision process is structured as a feedback loop whereby students are expected to iterate over their ‘Dissertation’ - a comprehensive document describing the project’s aims, literature review, methodology and its results – based on their supervisor’s feedback.

Therefore, it is imperative that every aspect of the supervision process, from the meetings conducted for communication between the supervisor and students to the feedback given remain as high as possible, for they shape not only the student’s academic success but also the metrics that determines the credibility and reputation of an institution.

1.1. Problem Statement

According to a recent paper on project supervision (Stynes, P. and Pathak, P., 2022), the traditional supervisory process for final year projects is not scalable. While the research focuses mainly on the postgraduate/masters level, its underlying principles – the supervisor’s finite time and attention, and one-on-one mentorship – are directly applicable for undergraduate programs as well. This lack of scalability leads to significant administrative burdens for academics and degrades the quality of the supervisory feedback, particularly within institutions with high enrollment numbers. As a result of these operational inefficiencies,

the quality of final year projects may suffer, affecting the institution's overall academic output and its credibility. This can be an emerging problem within the University of Mauritius' (UOM) Faculty of Information and Communications Technology (FOICDT), whereby the total enrollment has increased from 916 in 2020 to 1339 in 2022 (University of Mauritius, 2025), representing an increase in 46.2%. This rapid growth places a significant strain on the academic staff, who have to adhere to the manual and inefficient processes.

58	**E700**	Post-Doctoral Fellowship	PT	2		
59	**FOICDT**					
60	**E311**	BSc (Hons) Information Systems	FT	4		961
61	**E311M**	BSc (Hons) Information Systems	FT	1		
62	**IC311**	BSc (Hons) Information Systems	FT	142		
63	**IC318**	BSc (Hons) Computer Science	FT	219		
64	**E318**	BSc (Hons) Computer Science	FT	3		
65	**E318M**	BSc (Hons) Computer Science	FT	9		
66	**IC318M**	BSc (Hons) Computer Science	FT	26		
67	**IC319**	BSc (Hons) Applied Computing	FT	136		
68	**E319**	BSc (Hons) Applied Computing	FT	10		
	E319M	BSc (Hons) Applied Computing	FT	8		

Figure 1: Sum of Total Enrollment for UoM's FOICDT 2019-2020

2	FACULTY	TOTAL		
3		T	M	F
4	Faculty of Engineering	996	635	361
5	Centre for Innovative and LifeLong Learning	954	669	285
6	Faculty of Law and Management	2833	891	1942
7	Faculty of Information Communication and Digital Technologies	1339	836	503
8	Mahatma Gandhi Institute	477	125	352
9	Faculty of Social Sciences and Humanities	1351	311	1040
10	Mauritius Institute of Education	13	11	2
11	Faculty of Agriculture	359	107	252

Figure 2: Sum of Total Enrollment for UoM's FOICDT 2021-2022

Supervisors who are already encumbered from managing multiple students, including different versions of each of their work, have seen their workload increase further. Much of it comprises of the menial tasks of managing communications via email, progress tracking, rescheduling meetings due to conflicting schedules and most notably, document management which includes the repetitive task of checking drafts, often consisting of thousands of words, for compliance with feedback.

Thus, a greater portion of the workload is dedicated to handling administrative overhead as opposed to their core responsibilities such as teaching, supervising and research, impacting the quantity and quality of feedback provided. Evidently, these affect the students as well since providing insufficient or vague feedback makes it more difficult to identify and resolve underlying issues, triggering a loss in motivation for both parties. Consequently, the probability that the project meets academic standards reduces significantly.

1.2. Proposed Solution

To address these inefficiencies and subsequently the challenges of scalability on the institutional level, this project proposes a web-based platform, Final Year Project Management System (FYPMS) which will be designed with the goal of automating unproductive and repetitive tasks and streamlining existing processes.

The core feature of the system will consist of project management - supervisors will be able to administer a list of their projects and set deliverable submission tasks for a given project. A deliverable re-submission workflow will be built-in where re-submissions are checked for compliance with prior feedback using artificial intelligence. Furthermore, a meeting scheduling subsystem will allow supervisors to import their calendar schedules in the system. It will then be used as a basis for scheduling meetings. For each project, the system will provide the functionality to generate a progress log report. Lastly, reminders for important events such as meeting dates and assignment submission deadlines will be sent by email.

Chapter 2 - Background Study

This chapter will consist of the literature review of the features and limitations of each solution. This will include a section on the evaluation of their features with respect to the proposed system features and an appraisal section. The background study will then end with a section that describes the proposed system and its constituent features in detail.

2.1. Literature review

The following solutions derived from both widely available commercial software and academic research from various universities, arranged in tabular form.

Existing Solutions	**Features**	**Limitations**
A Web Based Final Year Project Supervision System (Beekhy, 2013)	- Appointment scheduling system which uses the supervisor's calendar as a basis for scheduling meetings. - Document management system where supervisors can set tasks for deliverable submissions/re-submissions and track document changes. - Progress tracker with Gantt chart visualization to provide a complete picture of project's progress.	- Lack of AI-powered feedback compliance checking - Inability to send notifications to external services such as email or push notifications for important reminders
Creatrix Campus *(Creatrix Campus, 2025:-b, Creatrix Campus, 2025:-c)*	- Centralized document repository for change tracking and version control. - Customizable review and approval workflows to streamline dissertation management (Low-code configurations).	- No out-of-the-box solution for automated feedback checking; training is required for developers to implement and maintain custom review and approval workflows.

	- Reporting analytics for evaluating dissertation and thesis milestones for both students and supervisors. - Meeting management software includes meeting scheduling with conflict avoidance and attendance tracking.	- Enterprise-level pricing may present a significant financial barrier.
Moodle as a Final Year Project Management System *(Khamaruddin et al., 2018)*	- The system re-purposes Moodle's forum activity module to provide reminders with email notifications for important tasks and deadlines. - Assignment activity module in Moodle allows supervisors to set assignments where students can submit drafts and final documents. - Progress updates are tracked via the system's course administration module.	- Assignment activity module does not provide change tracking between submissions. - The system lacks a meeting planning and automatic feedback checking system.
Final Year Project Management System for Information Technology Programmes *(Leung et al., 2018)*	- The system has a File Sharing and Repository module to centralize document and code submissions. - Project Management module provides students with a shared workspace and a scheduler for them to create tasks and set deadlines.	- The system lacks a meeting scheduling and automatic feedback checking system. - Group-based submission may obscure work accountability for individual students.

	- Basic progress tracking feature for each student.	
Google Workspace for Education *(Google, 2025)*	- Google Classroom acts as the core of the system where supervisors can make classes for project students and create assignments with deadlines. - Google Classroom integrates seamlessly with Google Calendar, which can be used as a basis for meeting scheduling. - Google Docs allows keeping track of different document versions for version control and overall change tracking using “Suggest” mode. - Google Docs integrate with Google Assignment for feedback submission.	- Lack of customization flexibility to incorporate an AI solution for feedback compliance checking. - Inadequate progress tracking capabilities to provide an overview of project’s progress.
Web-Based Final Year Project Supervision Management System (FYPSMS) (Adeniyi et al., 2024)	- Student home page for uploading project work documents and viewing progress. - Supervisor dashboard to view progress for multiple students and project inspection page for viewing, downloading documents & providing feedback.	- The system lacks a dedicated meeting scheduling system and automatic feedback checking. - There is no feature to track changes between document submissions.

2.1.1. Comparison between Existing Solutions

Solutions	**Meeting Scheduling & Reminder Notification System**	**Deliverable Submission/ Re-Submission**	**Automatic Feedback Compliance Checking**	**Progress Tracking (progress log or visual)**	**Specific Features**
A Web Based FYP Supervision System	✓	✓	✗	✓	- Meeting scheduling on the basis of supervisor calendar. - Gantt Chart visualization for progress tracking
Creatrix Campus	✓	✓	?	✓	- Low code platform to create review and approval workflows for deliverables. - Meeting Attendance tracking
Moodle as a FYP Management System	✓	✗	✗	✓	- Re-purposing existing open-source Moodle modules to provide specific features
FYP Management System for IT Programmes	✗	✗	✗	✗	- Architecture for streamlining group-based project supervision
Google Workspace for Education	✓	✓	✗	✗	- Document versioning for visible change tracking - Prevent document

					editing/re-submission during feedback
Web-Based FYPSMS	✗	✗	✗	✓	- Dedicated dashboards for system administrators and external examiners

2.1.2. Appraisal

The ecosystem of digital solutions aimed at addressing the inefficiencies in academic management is already mature enough to justify their widespread adoption among colleges and universities around the globe. The extent of their integration with existing IT infrastructure vary significantly from one institution to another. For instance, some institutions use standard tools like Learning Management Systems for basic communication and deliverable handling. Others, however, adopt centralized full-fledged platforms to manage the entire supervision process from smart meeting scheduling to document management with progress tracking.

The University Of Mauritius currently uses Google Workspace for Education as a general education platform for a variety of courses in order to provide learning material, centralize assignment submissions or simply to broadcast reminders to particular cohorts. However, as Beekhy (2013) pointed out, there is no unanimously agreed upon methodology, and subsequently no centralized system supervisors use for the supervision process.

An evaluation of the project management systems listed so far shows that they tend to address areas concerning project allocation, meeting scheduling and managing deliverable submission. Most of them are adapted for the one-on-one supervision process although Leung et al. (2018) described a system that provides features dedicated for group-based supervision. Nonetheless, these systems have a clear functional gap – most of them lack some form of workflow automation for the deliverable submission. In that case, the feedback compliance checking process is absent in these solutions.

2.2. Proposed System Features

This section will present the proposed system features based on the solutions identified in the previous context and the application's intended purpose.

2.2.1. Role-Based Access Control

Firstly, the web platform will require a way to verify users of the system and determine which set of features of the system they will be allowed to access. For this, the system will have to provide authentication – verify users based on their credentials and categorize them as either one of the stakeholders below:

- Administrator(s)
- Supervisors
- Students

2.2.2. Administrator Dashboard

Since the system requires a secure way of determining the roles of users, they will be managed by an administrator (academic staff) via a dedicated dashboard inaccessible to the other users. The administrator's role will encompass user creation and role assignment.

2.2.3. Project Management & Progress Log Generation

Supervisors and students, on the other hand, will be concerned with the final year project(s) and meeting scheduling. Hence, supervisors will be given the ability to create and manage projects that are assigned to them by the university, including adding the corresponding students to them, joining projects or archiving completed projects. On the other hand, students may only be able to view the corresponding project or create a new one if they don't have one.

For administrative purposes, a progress log of project activities is required. Hence, a project progress log generation feature, based on prior meetings conducted and deliverable submissions, will be made available for individual projects and will be accessible by both the students and supervisors.

2.2.4. Deliverable Submission & Feedback

The supervision process is an iterative process where the supervisor reviews document drafts of the student's research on the project, which altogether, will contribute to the overall dissertation. Therefore, for a given project, the system will provide supervisors with the feature to create deliverable submission tasks with deadlines.

These tasks will be based on the activity box of Google Classroom (Fig 3), where initially students will be able to submit their deliverables and supervisors will provide feedback on their work. Then, the given feedback will be made visible in the corresponding task from which the deliverable was submitted, and also be sent as a notification to the student via email.

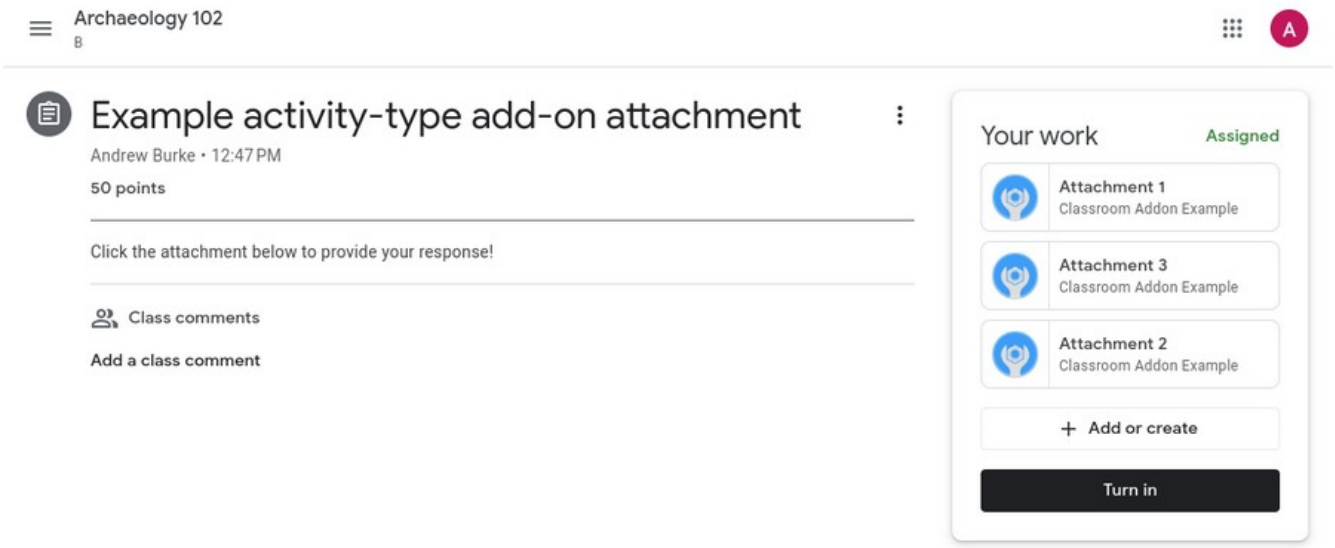

Figure 3: Google Classroom Activity Example

2.2.5. AI Feedback Compliance Analysis

After providing feedback, the supervisor expects the student to iterate over it and produce a version of their work more aligned with the correct requirements. Unfortunately, this can rapidly spiral out of control and overwhelm the supervisor especially if they have to review feedback on multiple tasks for multiple projects.

The solution is to enhance the deliverable re-submission process by adding a workflow where the new deliverable version is checked for compliance with the previous feedback criteria through the use of artificial intelligence. Then, the list of feedback criteria is displayed as a list to the student, who may chose to either re-do their work or override certain feedback criteria and submit the new version.

2.2.6. Meeting Scheduling

Based on Bheeky's (2013) meeting scheduling module, the meeting scheduling component aims to streamline the traditional way of scheduling appointments. Supervisors will have a calendar which is shared to all students under their supervision. Either a student or supervisor can book a meeting (organizer) with a supervisor or student respectively (attendee). The organizer-attendee pair will strictly be a supervisor-student pair. The attendee can chose to accept/reject/postpone a meeting.

2.2.7. Notification & Reminder System

In order to keep the students and supervisors aware of important events, reminders will be sent as email notifications. Furthermore, these reminders will be displayed in a section visible when either user logs back into the system.

Chapter 3 - Analysis

This chapter will focus on the analysis of the system proposed in the previous chapter, evaluating the architectural considerations of the system and the technologies that will be used as the basis of their implementation.

It will first consist of a list of the architectural considerations identified and then followed by an in-depth analysis of the potential solutions/paradigms that will be used for each one of them.

3.1. List of Key Architectural Considerations

The following is a list of the system aspects that require a careful consideration of their underlying technology(s).

- Application Architecture
- Programming Language / Web Development Framework
- Meeting Scheduling & Reminder Notification
- Database
- Authentication & Session Management Pattern
- Artificial Intelligence for Feedback Compliance Checking

3.2. Architectural Considerations

This section will discuss the architectural considerations identified in the previous one, dedicating three sub-sections for each consideration:

- Evaluation Criteria
- Comparison of the Features of the different Choices
- Justification for the Choice selected

3.2.1. Application Architecture

The application architecture serves as the foundation of the entire system – every subsequent considerations and design decisions will be dependent on it. Thus, a poor choice will significantly increase the cost of the evolution of the software later in the development life-cycle.

3.2.1.1. Evaluation Criteria

The following evaluation criteria have been selected and filtered from common criteria used in choosing the correct web application architecture *(Bass, Clements & Kazman, 2021)*, in order to suit the context of the solution to be developed.

- **Scalability –** It broadly refers to the ability to increase the system/component's capabilities in order to handle an increasing number of users.
- **Fault Isolation/Availability –** It is to the extent the system's features remain available to users in the event of faults.
- **Development Speed –** It describes the development effort and speed in order to provide a fully-functional system

3.2.1.2. Application Architecture Comparison

The three most common architectures for developing web applications are presented for selection:

	Monolithic	**Service Oriented Architecture (SOA)**	**Microservices**
Description	Monolithic architecture is where all components are tightly coupled and run as a single service	SOA is an architectural style that supports integration of business processes as linked services that may be	Microservices is a design that structures an application as a collection of smaller, independent services (Newman, 2015)

		accessed when needed over a network (Mehta, Lee, & Shah, 2006)	
Scalability	Lack of scalability since scaling one component requires scaling the entire application due to their tight coupling	Service components are loosely coupled and can be scaled separately	Highly scalable since every service is decentralized
Fault Isolation/ Availability	A single failure can bring the entire application down	Faults affect only the layer they occur in since there is less coupling among services	Faults only affect the services they occur in without impacting the rest of the application
Development Speed	Easiest and Fastest to develop and debug early since services interact directly without needing an interface to decouple them	Moderate development speed since services need to interact with each other using web services but can be developed in the same application	High development complexity and slowest development speed since services are isolated and require managing their orchestration.

3.2.1.3. Application Architecture Choice

Service-Oriented Architecture (SOA) will be chosen as the application architecture due to the following reasons:

1. Due to application requirements of handling large user traffic, it will need be to be scalable both vertically and horizontally *(see appendix).* As such, the service components needs to be loosely coupled to allow the system design to evolve to a microservices architecture later as an example of horizontal scaling.
2. While all three architectures adhere to the logical separation of concerns presented by the 3-Tier Architecture *(see appendix)*, SOA is preferable over Monolithic design since a fault in one layer (e.g presentation layer) does not affect other layers.

3. SOA is also preferable compared to microservices architecture since it has a lower development complexity and faster development and debugging speed.

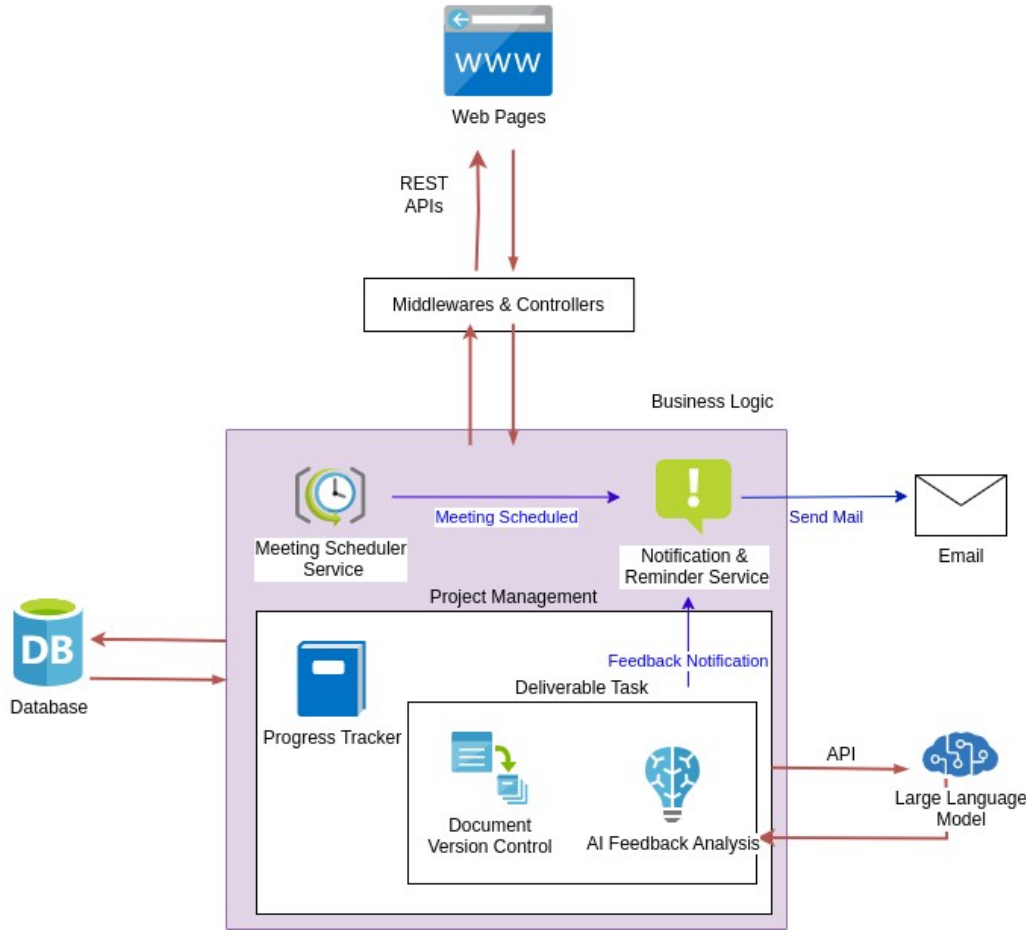

Figure 4: Architecture of the Proposed System

3.2.2. Programming Language / Web Development Framework

The programming language/framework to be used will directly affect how the components will be built, the ease of inter-component communication and how suitable will the application be for supporting the functional and non-functional requirements of the system.

3.2.2.1. Evaluation Criteria

The choice of the development framework is dependent on the following criteria commonly used to demarcate programming languages:

- **Programming Language Features –** Characteristics of the underlying language such as type safety, garbage collection for memory safety and automatic memory management, whether the language is compiled or interpreted.
- **Ecosystem -** Availability of third-party libraries for facilitating the development of custom application components, support for integration with external services such as databases and APIs and documentation quality.
- **Built-in Abstractions –** Provision of in-built features that handle tasks (such as sanitizing input to guard against XSS *(see appendix)*) without having to write the low level logic. This reduces development time and human error for critical code sections.
- **Response Latency & Concurrency –** If the framework offers a web server, response latency is the speed it can process requests. On the other hand, concurrency is the ability of the server to handle multiple simultaneous user requests.

3.2.2.2. Development Framework Comparison

The three most suitable web development frameworks have been identified and listed below for comparison:

	.NET Core (C#)	**Laravel (PHP)**	**Next.js (Javascript/Typescript)**
Programming Language Features	Its underlying language, C# is a strongly-typed, memory-safe and compiled high level language with support for a wide variety of operations and features.	PHP is an interpreted language with optional type safety, built specifically for web development. It has libraries and language features to support tasks such as session	Javascript, officially known as ECMAScript, is an interpreted (JIT-compiled *(see appendix)*) language that is used for both client-side and server-side logic. Its flexible syntax allows for rapid prototyping. Type

		management & input validation.	safety is provided by Typescript, which is a super set of Javascript.
Ecosystem	.NET offers an ecosystem of packages via Nuget package manager, including both community-developed and Microsoft-backed libraries.	Laravel provides an ecosystem of both official and community-developed packages through the Composer dependency manager for PHP.	Next.js makes use of the node package manager (npm) which manages the largest registry of packages and therefore, provides the largest ecosystem of readily available tools.
Built-in Abstractions	.NET has built-in middlewares for Authentication, logging and testing with built-in features against CSRF & XSS *(see appendix)* attacks. (ASP.NET Core, 2025) It also provides .NET Entity Framework ORM *(see appendix)* for database abstraction.	Laravel has out-of-the-box CSRF *(see appendix)* protection and middlewares for request JSON schema validation and logging. (Laravel Docs, 2025) It provides Eloquent ORM for database abstraction.	The built-in abstractions provided by Next.js are primarily concerned with optimizing performance for website rendering on the client.
Response Latency & Concurrency	C# is compiled into a highly optimized intermediate language and then executed, ensuring high performance. Furthermore, its multi-	PHP has high overhead per request and thus a high response latency due to architectural considerations and blocking I/O operations.	The underlying runtime, Node.js uses a single-threaded event loop for non-blocking IO, which is efficient for concurrent network requests (Node.js Docs, 2025)

	threaded nature allows it to efficiently handle multiple I/O requests simultaneously. As a result, it can tolerate high loads.	Laravel Octane is required to achieve lower latencies and higher concurrency support via persistent memory and asynchronous processing. (Laravel Octane, 2025)	Next.js improves website performance for the client by providing features such as Server-Side rendering *(see appendix)* (Next.js Docs, 2025)

3.2.2.3. Choice of Framework

.NET core will the be the most appropriate choice for the development framework based on the criteria given and the following additional justifications:

I. The web platform must be designed handle thousands of concurrent users, especially when nearing the submission deadlines, during which deliverable submissions may soar considerably. This will be a common criteria in determining the most suitable technology to use. Buljić et al. (2025) showed that .NET Core consistently outperforms other web frameworks in response latency and handling concurrent HTTP requests.

II. The ecosystem and built-in abstractions for .NET are more aligned with the requirements of the project – robust middlewares for authentication, testing and logging will reduce development time considerably while its ecosystem has official libraries for a range of tasks including LLM interactions and cron job scheduling.

Framework	Request Type	Average (ms)	Min (ms)	Max (ms)	Std. Dev.	Throughput (req/sec)	Received KB/sec	Sent KB/sec	Avg. Bytes
Django	HTTP POST	52	11	115	13.43	5.1	1.87	1.12	377.8
	HTTP GET	12	3	63	10.75	5.1	55.74	0.63	11257.3
	Total	32	3	115	23.75	10.1	57.6	1.76	5817.5
.NET	HTTP POST	1	0	37	1.84	5.1	1.25	1.13	250
	HTTP GET	1	0	5	0.58	5.1	53.65	0.63	10765
	Total	1	0	37	1.36	10.2	54.87	1.76	5507.5
Express.js	HTTP POST	12	6	40	3.34	5.1	1.41	1.12	282.8
	HTTP GET	3	1	9	1.32	5.1	55.96	0.63	11247.7
	Total	7	1	40	5.24	10.2	57.35	1.76	5765.2

Fig 2. Metrics made in Apache JMeter

Figure 5: Server Side Framework Performance Metrics (Buljić et al., 2025)

3.2.3. Meeting Scheduler & Reminder Notification

Reminders from important events such as upcoming deliverable submission deadlines and meetings should be made visible to the relevant parties until the event has occurred, upon which they are discarded.

3.2.3.1. Evaluation Criteria

The evaluation criteria of the algorithm used to manage the life-cycle of the reminders is chosen on the basis of the criteria below. They have been shortlisted based on the context of the application's features and other considerations.

- **Scalability –** Ability to handle higher reminder workloads without straining much on available resources.
- **Reliability –** Ensuring that the service always functions as intended. This primarily concerns the ability of sending notifications at variable notice periods.
- **Development Cost/Complexity**

3.2.3.2. Comparison of Strategies

	Client-side Reminder Life-cycle Management	Cron Job Scheduling
Description	Management of the life-cycle of reminders is purely initiated from the client. The client will thus need to have the corresponding logic to periodically check and alert the server on any state changes of the reminders.	A recurring job is run periodically to check and update the state of reminders and send notifications. Any updates on the reminders' states will need to sent to the client and handled accordingly.
Scalability	Since the server will only be responsible for CRUD operations, the scalability of this strategy is	The job needs to query the database periodically. This can become a bottleneck if there are a large number

	dependent only on the performance of the client-side logic	of reminders, slowing down other database operations as well.
Reliability	Since the service won't work when the clients are offline, it is not possible to send recurrent email notifications within different notice periods	Highly reliable since the service will always be running on the server and supports sending reminder notifications at different notice periods
Development Speed / Complexity	Low to moderate complexity since the client only needs an asynchronous event loop to periodically check for the state of the reminders	Moderate to high complexity since it involves managing a cron job scheduling service and informing clients of state updates.

3.2.3.3. Choice of Scheduling Strategy

The Client-side Reminder Life-cycle Management strategy will be chosen due to the following reason:

The ability to send email notification reminders at variable notice times is not a necessity and hence can be traded off for the first option in order to benefit from higher scalability and lower development complexity.

3.2.4. Database

The database is a critical component in ensuring the persistence of data as per the requirements of the system. Its importance is comparable to the choice of the system architecture because a poor choice may become a bottleneck in the system's performance and scalability.

3.2.4.1. Evaluation Criteria

The database to use will be selected based on the criteria as commonly stipulated by database vendors *(Belagatti, 2025)*, curated to suit the use cases of the system:

- **Performance**
- **Security & Reliability**
- **Scalability**
- **Cost & Licensing**
- **Development Framework Integration**
- **Large Object Storage & Retrieval**

3.2.4.2. Database Comparison

	PostgreSQL	Microsoft SQL Server	MySQL
Performance	PostgreSQL offers higher performance than MySQL for frequent update and delete operations. It offers less performance than MySQL for frequent read operations. (AWS -	SQL Server offers the highest performance due to several optimization capabilities. Supports advanced indexing techniques	MySQL is less performative on for heavy update workloads than PostgreSQL. It outperforms PostgreSQL in frequent read operations (AWS - PostgreSQL vs

	PostgreSQL vs MySQL, 2025) Supports more advanced indexing techniques as compared to MySQL.		MySQL, 2025)
Security & Reliability	ACID *(see appendix)* compliant with support for advanced indexes for reliability	Extensive feature set for ensuring security and reliability including ACID compliance	ACID compliance depends on the storage engine used
Scalability	Scales both vertically (increasing resources for a single server) and horizontally (increasing number of servers)	Scales both vertically and horizontally	Scales better horizontally than vertically
Cost & Licensing	Free & Open-source	Requires licensing for large scale enterprise environments	Free & Open-source
Development Framework Integration	.NET Core support provided by Npgsql, which is an open-source data provider package.	Microsoft provides an official data provider for SQL Server and native libraries are specifically optimized for interaction with SQL Server.	.NET Core support provided by MySqlConnector, which is open-source.
Large Object Storage & Retrieval	PostgreSQL uses TOAST (PostgreSQL: Docs – TOAST, 2025) optimization to compress large objects before writing to disk.	Less storage compression efficiency offered as compared to PostgreSQL.	No inherent compression techniques for large objects and has less storage efficiency than PostgreSQL.

	TOAST also makes retrieval slightly faster than MySQL by making records cache-friendly.		

3.2.4.3. Choice of Database

In spite of SQL Server being the most advanced of the databases listed, PostgreSQL will be selected due to SQL Server's expensive licensing costs. The decision is also based on the additional reasoning:

I. Since deliverables may often be text documents ranging from 5-150 pages long, PostgreSQL will be more suitable for handling large text objects due to its TOAST optimization features, optimizing storage utilization. Read operations will also be slightly more efficient.

II. Despite the consensus that web applications have a higher read-write ratio, the number of write operations can significantly increase when taking into account the meeting scheduling and reminder system. Therefore, PostgreSQL will be most suited for handling this change in workload.

3.2.5. Authentication & Session Management Pattern

This architectural consideration addresses a fundamental aspect of the system – security. A weak authentication & session management pattern can result in vulnerabilities such as session hijacking and/or can be a major performance bottleneck.

3.2.5.1. Evaluation Criteria

The authentication & session management pattern of choice is selected based on the application needs, organized as the following criteria:

- **Implementation Complexity**
- **Performance & Scalability**
- **Security**
- **Convenience**

3.2.5.2. Comparing Authentication & Session Management Patterns

	Session-Based Authentication	**Token-Based Authentication**
Description	It involves creating a record on the server associated with a user session and sending back a sessionID token in the form of an HttpOnly cookie	It involves creating a JWT *(see appendix)* access token that contains user credentials and permissions and sending it back to the client, which is subsequently stored in local storage. The access token is included in the Authorization header of future HTTP requests or as an HttpOnly cookie
Implementation Complexity	Very simple to implement.	High complexity if refresh token and token revocation logic is added.

	Provided in .NET Core as services.AddAuthentication().AddCookie() middleware	Provided in .NET Core as services.AddAuthentication().AddJwtBearer() middleware
Performance & Scalability	The need to query a session store for every request affects performance and scalability.	Scalable since server only needs to validate access token and extract user data and permissions on each request.
Security	XSS and CSRF security achieved by setting HttpOnly to true and SameSite attributes to strict respectively. Sessions are revocable.	Access tokens must be short-lived since they cannot be revoked on the server without adding complex revocation logic. XSS and CSRF security achieved by storing them as an HttpOnly cookie with SameSite attribute set to strict
Convenience	User has to re-enter credentials every time the session expires.	User has to re-enter credentials once the access token expires. However, if refresh tokens are used, access tokens can be refreshed, allowing for sessions to persist

3.2.5.3. Choice of Authentication & Session Management Pattern

Token-based authentication **without revocation and refresh logic** will be selected based on the following reasons:

I. Implementation complexity for session based authentication with this simple type of token based authentication is comparable due to the availability of built-in middlewares in .NET Core.

II. Since performance at scale under large number of concurrent users is one of the more important metrics of the application, token-based authentication is preferred since

tokens are stateless and hence scalable. This architectural evidence is supported by the adoption of JWT in systems like API Gateways for its ability to enable statelessness and scalability (Gupta et al., 2025)

III. The lack of revocation for JWT tokens is an acceptable risk since even if an unauthorized user hijacks the device with the open session, they will not be able to perform significant changes before token expiry. This is provided that the token has a short expiration time (5-15 minutes)

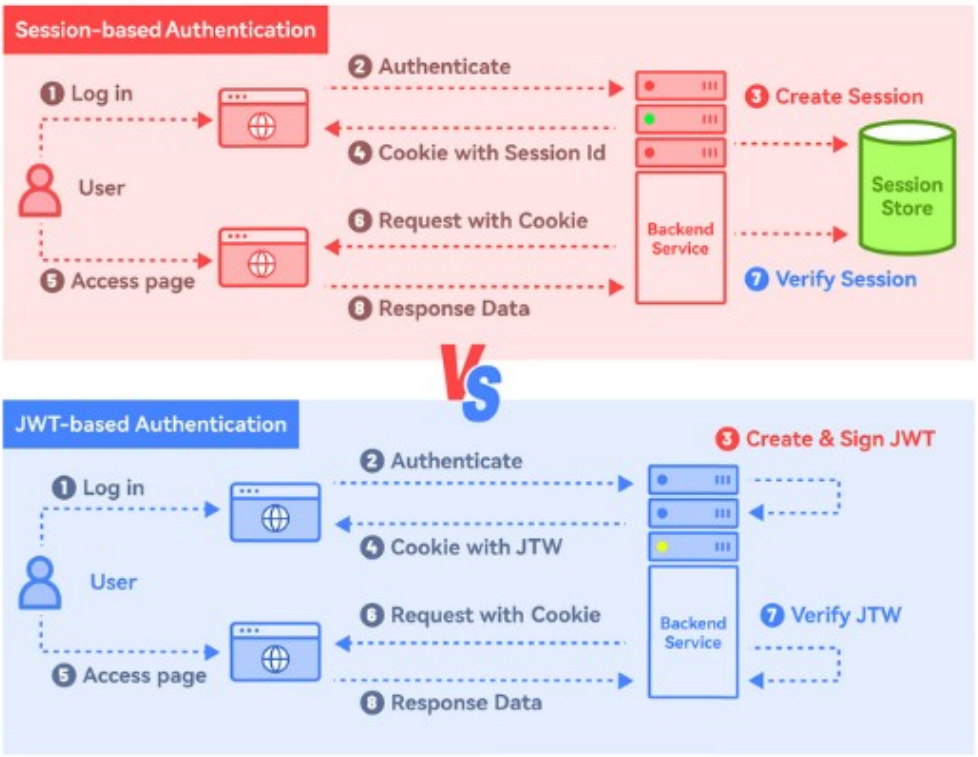

Figure 6: Session vs JWT Authentication (ByteByteGo, 2025)

3.2.6. Artificial Intelligence for Feedback Compliance Checking

The selection of the AI component determines the accuracy and effectiveness of the compliance checks. The process in question – feedback compliance – is a natural language inference/textual entailment task where a model is required to determine the relationship between a premise (document text) and the hypotheses (feedback criteria).

Large Language Models are state of the art systems used to perform this task.

3.2.6.1. Evaluation Criteria

The large language models will be selected based on the criteria

- **Performance**
- **Latency / Speed**
- **Context Window**
- **Licensing & Cost**

3.2.6.2. Comparison of Large Language Models

Large Language Models	**Anthropic: Claude 3 Opus**	**OpenAI: GPT-4o**	**Gemini 2.5 Flash**
Performance	Top-tier performance on many LLM benchmarks	Top-tier performance	High performance on benchmarks
Latency / Speed	High latency	Low latency	Lowest latency
Context Window	200,000 tokens	128,000 tokens	1,000,000 tokens
Licensing & Cost	Highest cost per token	Moderate costs per token	Lowest costs per token

3.2.6.3. Choice of Large Language Model

Gemini 2.5-Flash will be chosen because it is the best fit for the most of the criteria set

3.3. Requirements Specification

This section will specify the functional & non-functional requirements of the system.

3.3.1. Functional Requirements

The functional requirements of the system will be grouped based on their categories as sub-sections

3.3.1.1. User Management

Requirement ID	Description
FR1	The system shall have as the initial user an administrator, who shall be able to view, create users with their credentials and assign their roles.
FR2	Users shall be able to prove their role either as a (I) student (ii) supervisor or (iii) administrator through the use of their credentials in the system.

3.3.1.2. Project Management

Requirement ID	Description
FR3	Supervisors shall be able to view, search, create, update or archive projects. Students shall only be able to view their corresponding project or a create a single project.
FR4	A supervisor shall be able to assign a student to a created project or join a project created by a student.
FR5	A project shall consist of a list of Task entries arranged in reverse chronological order. Each

Requirement ID	Description
	Task entry shall allow a user to (I) access the Task and (ii) display its corresponding deadline and status from missing/completed.
FR5	For a project, the supervisor shall be able view, create, update and delete Tasks for deliverable submissions.
FR6	Creation and modification of a Task in a project by a supervisor shall create a reminder based on the deadline of the Task and notify the student assigned to the project by email.
FR7	Supervisors and students shall be able to generate and download a progress log report of the project based on prior meetings conducted and deliverable submissions.

3.3.1.3. Deliverable Task

Requirement ID	Description
FR8	A Task shall be an interface that provides the deliverable file and displays the corresponding feedback criteria for the current deliverable version.
FR9	The system shall provide a student with a function to upload or remove a deliverable file pre-submission for a specific Task.
FR10	The system shall provide a student with a function to submit the deliverable file for the Task.

Requirement ID	Description
FR11	The system shall only store the latest version of the submitted deliverable file and its corresponding feedback criteria.
FR12	The system shall provide the supervisor with a function to submit a list of feedback criteria for the latest deliverable submitted in a Task.
FR13	The system shall prevent a student from removing/modifiying the submitted deliverable file while the supervisor provides feedback.

3.3.1.4. AI Feedback Compliance Checker

Requirement ID	Description
FR14	During re-submission of a deliverable, the output of the LLM during feedback submission shall be used as context with the text delta of the new submission to evaluate the feedback criteria.
FR15	The system shall display a loading state for all LLM invocations until the operation either terminates successfully or unsuccessfully.
FR16	The system shall send email notifications to the supervisor and student for deliverable submission and feedback provision respectively.

3.3.1.5. Meeting Scheduling

Requirement ID	Description
FR17	The system shall allow all booked meetings

Requirement ID	Description
	to be visible to all users.
FR18	A user (organizer) shall be allowed to book a meeting between another user (attendee), with the two parties being strictly of a supervisor-student pair.
FR19	Upon booking a meeting, a reminder shall be created for both the organizer and attendee and an email is to be sent to notify the attendee.
FR20	The attendee shall be allowed to accept, reject or postpone a meeting request, creating a reminder for both parties and notifying the organizer by email.

3.3.2. Non-Functional Requirements

Requirement ID	Description
NFR1	The system shall enforce Role-Based Access Control *(see appendix)* by only authorizing users to access resources and views based on the permission sets associated with their roles.
NFR2	The system shall provide a user-friendly interface.
NFR3	All server requests except for LLM invocations shall have a response time of less than 500ms.
NFR4	The server CPU usage shall be less than 80% for 1000 concurrent users.

Chapter 4 - Design

This section will provide the necessary specifications for the system's design, including both structural and behavioral UML diagrams, arranged in the following order:

1. Use Case Diagrams
2. Class Diagrams
3. Sequence Diagrams
4. Entity Relationship Diagram

4.1. Use Case Diagrams

From the user's perspective, the system can be divided into 4 sub-systems:

- Meeting Scheduling Subsystem
- Project Management Subsystem
- Deliverable Submission Subsystem
- Administrator Subsystem

4.1.1. Meeting Scheduling Subsystem

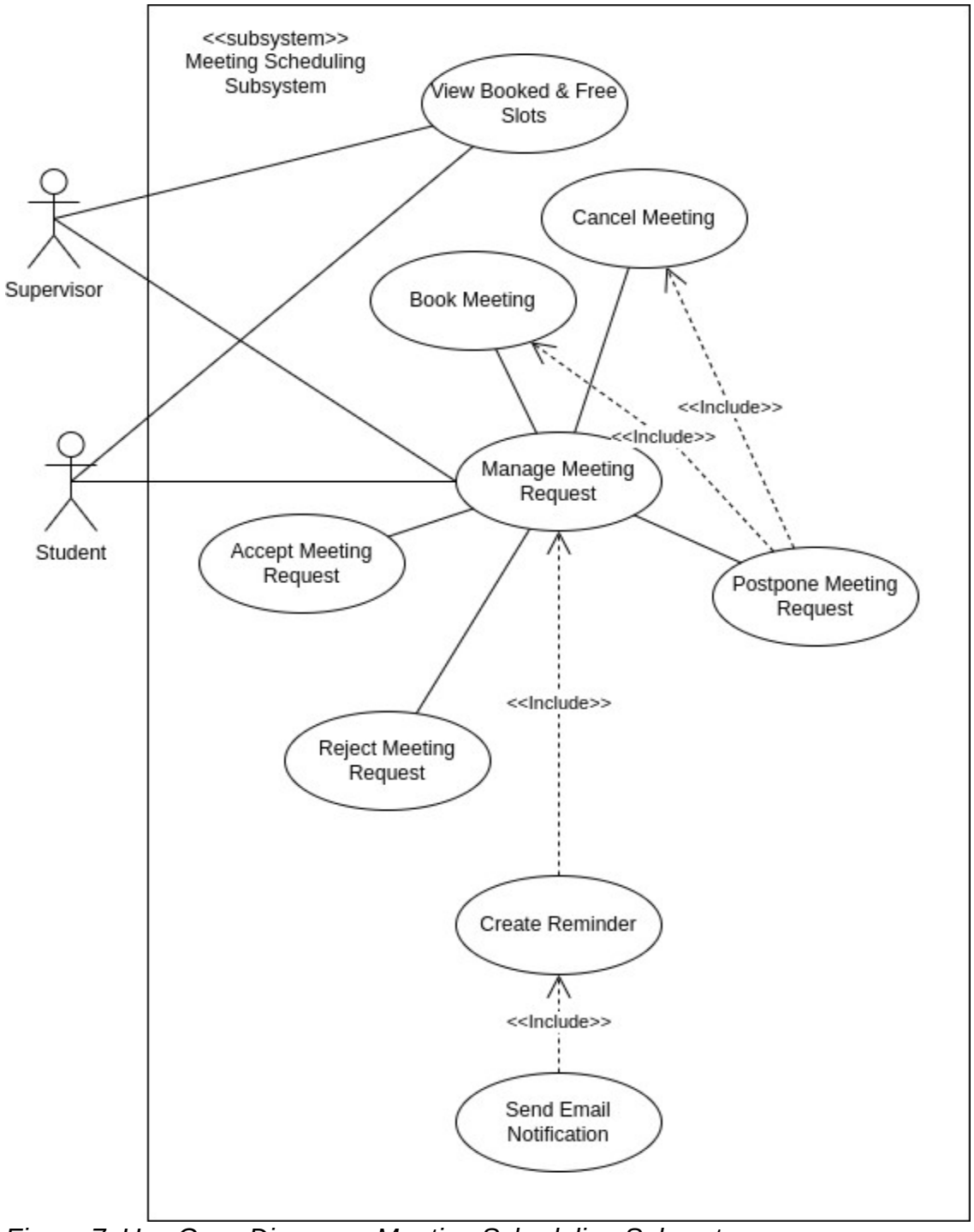

Figure 7: Use Case Diagram - Meeting Scheduling Subsystem

4.1.2. Project Management Subsystem

Figure 8: Use Case Diagram - Project Management Subsystem

4.1.3. Deliverable Task Submission Subsystem

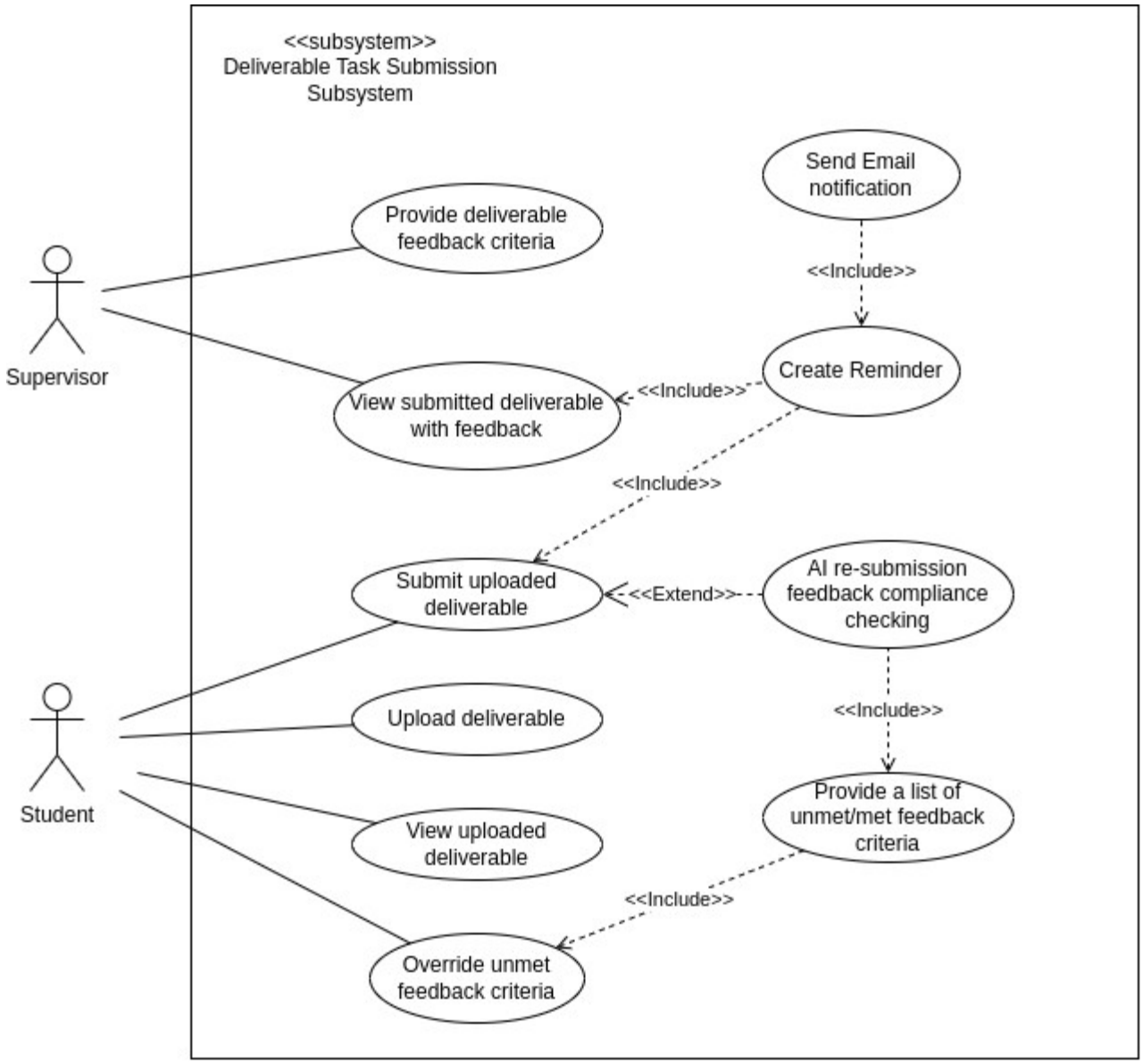

Figure 9: Use Case Diagram - Deliverable Task Submission Subsystem

4.1.4. Administrator Subsystem

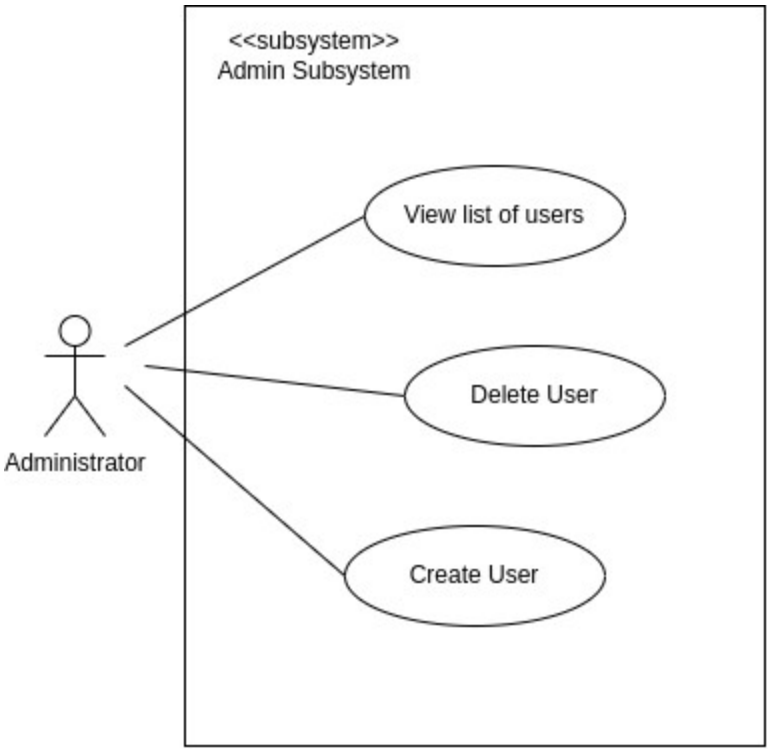

Figure 10: Use Case Diagram - Administrator Subsystem

4.2. Class Diagrams

For readability, the class diagram of the application will be separated into smaller class diagrams, categorized as follows:

- **Class Diagram (Domain)**– It depicts the domain model (see appendix) of the application.
- **Class Diagrams (Subsystems)** - Class Diagrams that depict the relationship (dependencies & associations) between the classes at the application level.

  They are further arranged as follows:

  1. Meeting Scheduling Subsystem
  2. Project Management Subsystem
  3. Deliverable Task Submission Subsystem
  4. Progress Log Subsystem
  5. Notification & Reminder Subsystem
  6. Administrator Subsystem

In order to reduce clutter, the method signatures for the classes will intentionally be left as incomplete, only highlighting the dependencies instead. The actual method signatures are covered in the following section on Sequence Diagrams.

4.2.1. Class Diagram (Domain)

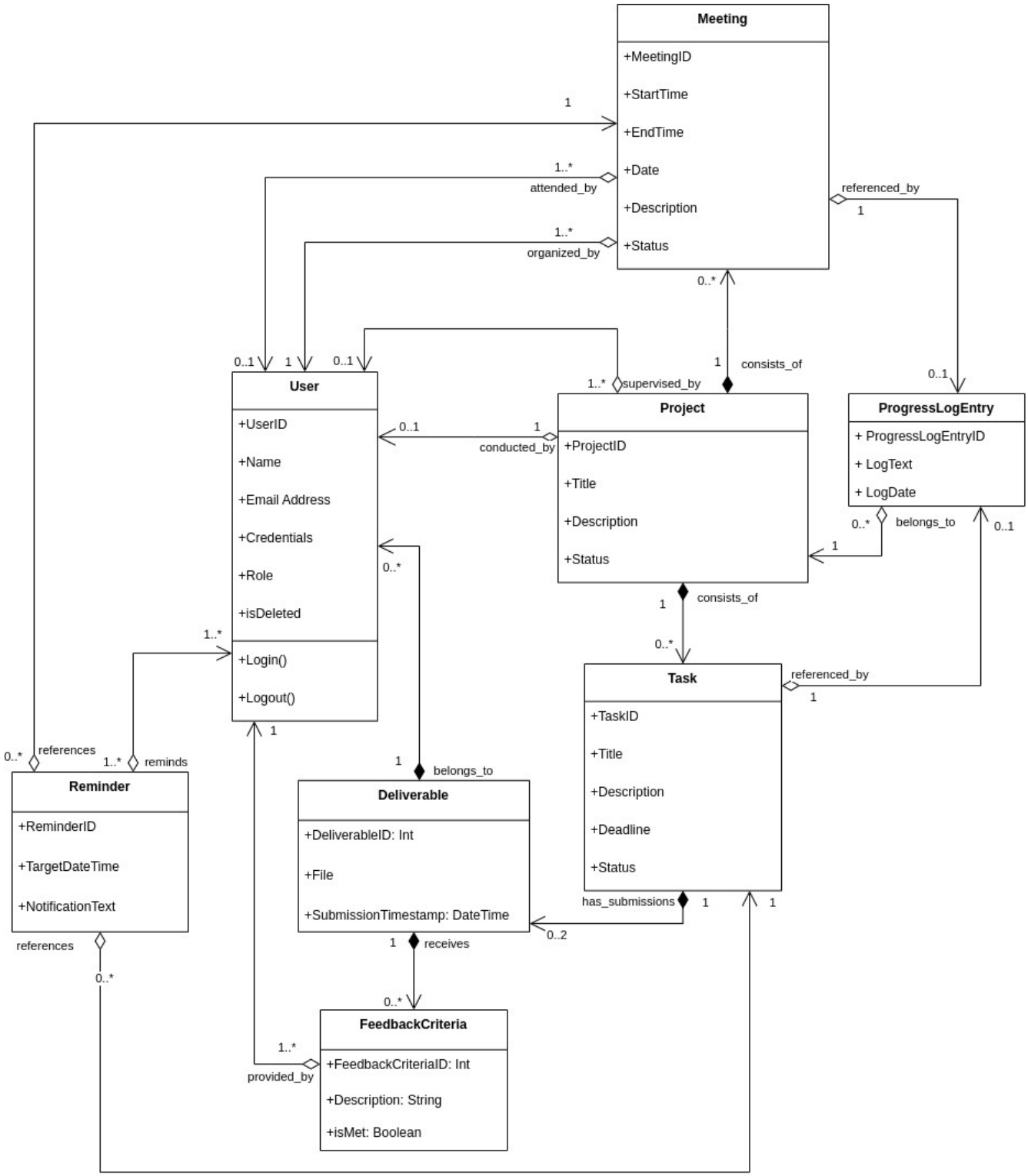

Figure 11: Class Diagram (Domain)

4.2.2. Meeting Scheduling Subsystem

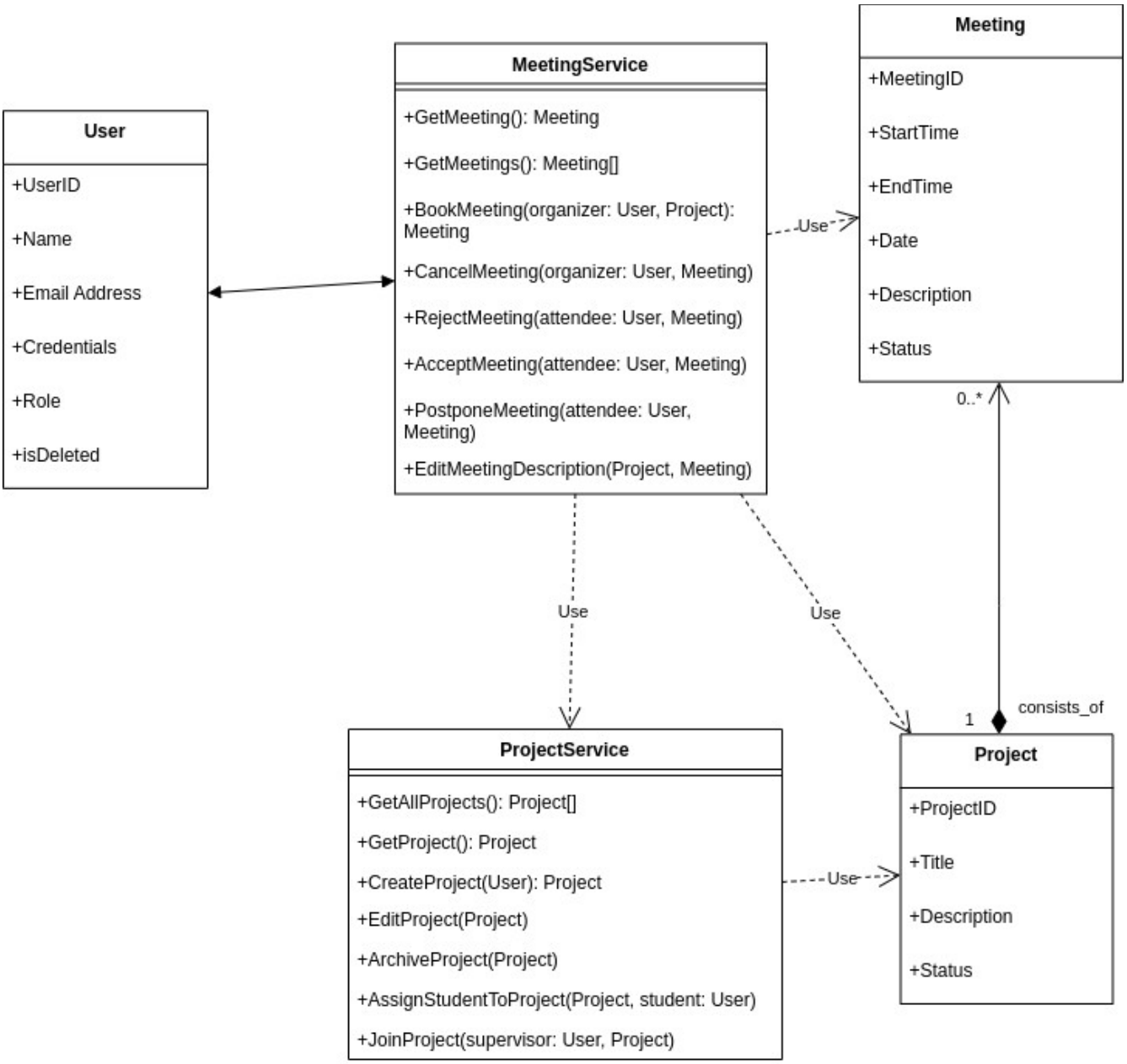

Figure 12: Class Diagram- Meeting Scheduling Subsystem

4.2.3. Project Management Subsystem

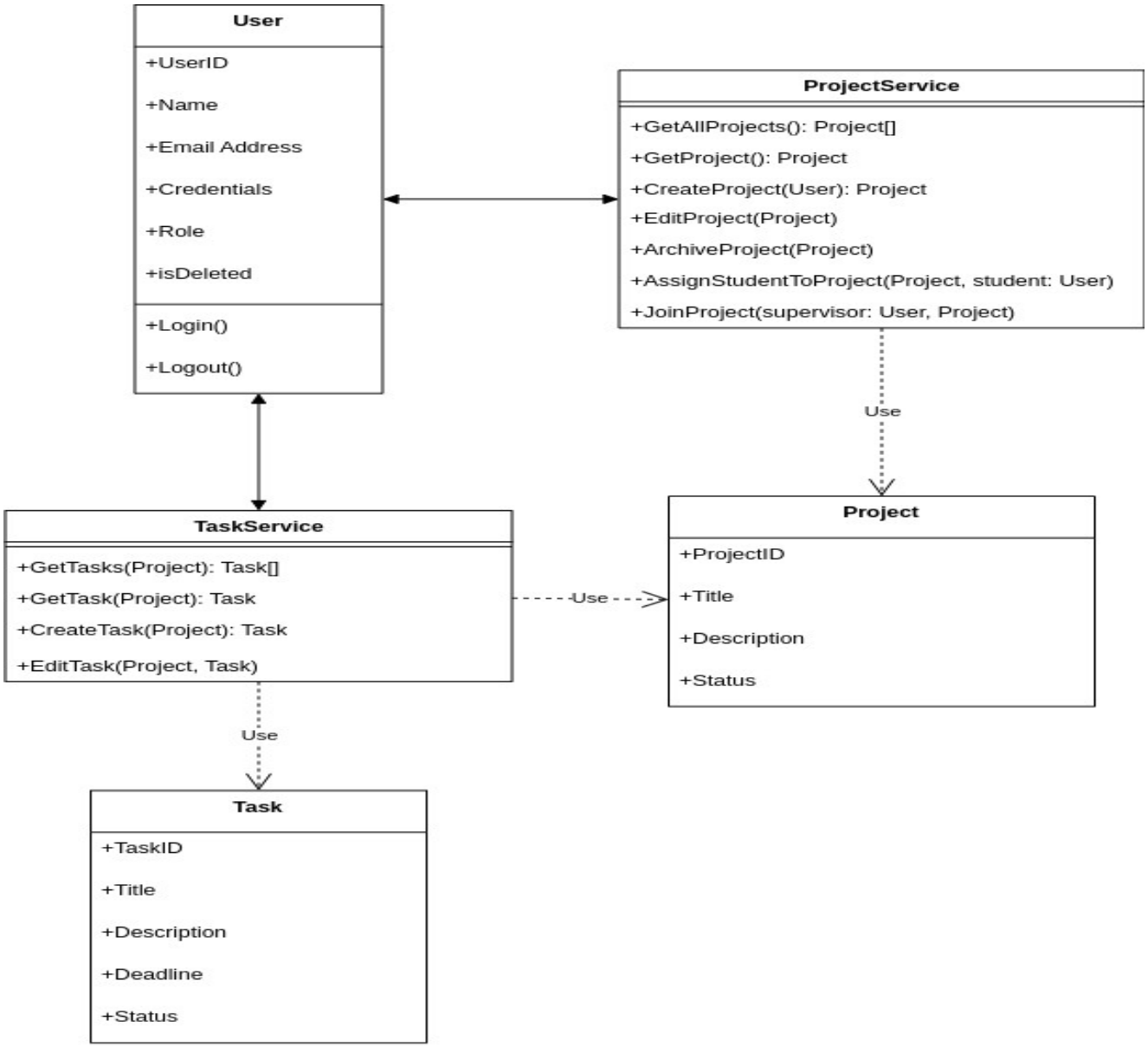

Figure 13: Class Diagram - Project Management Subsystem

4.2.4. Deliverable Task Submission Subsystem

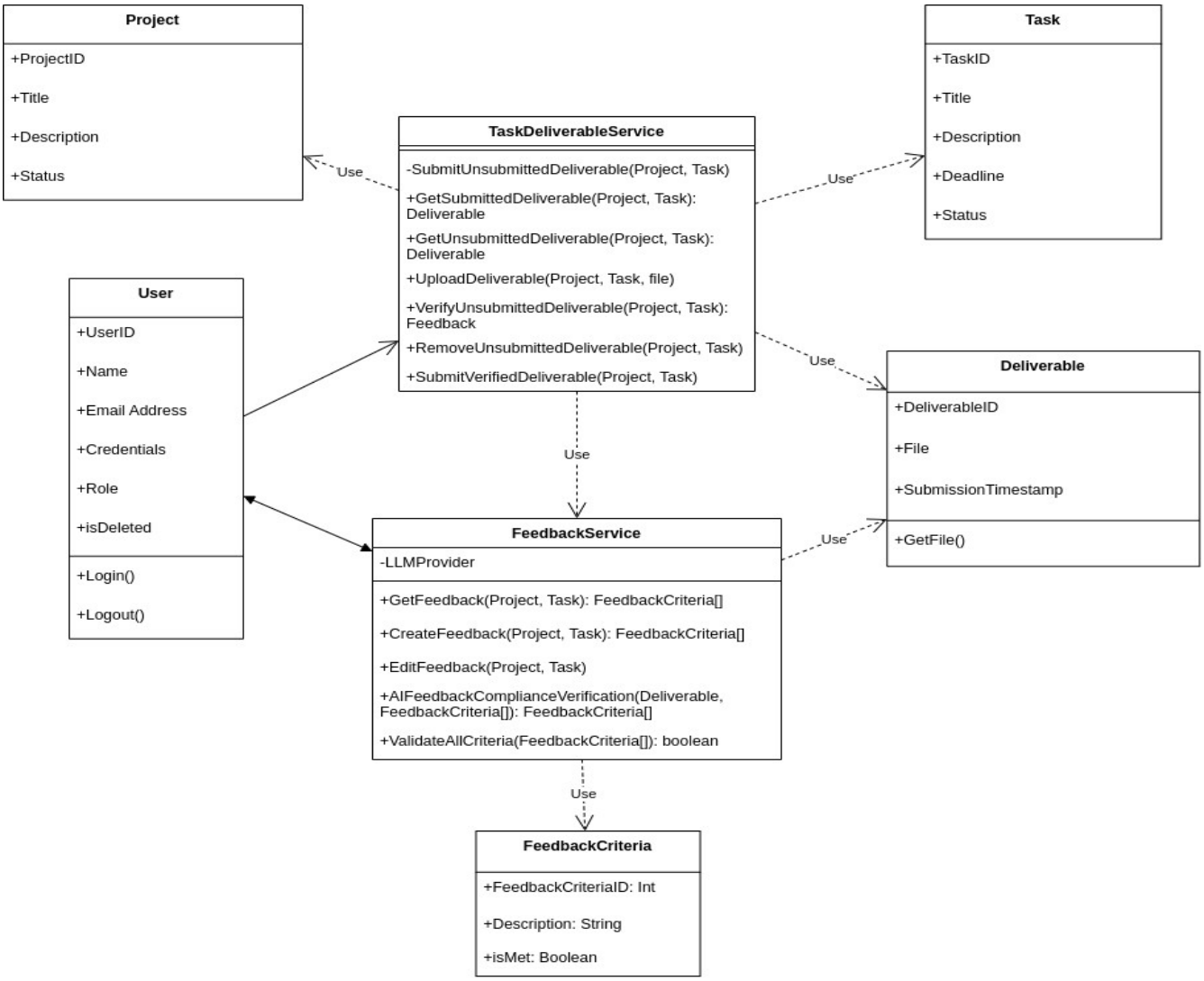

Figure 14: Class Diagram - Deliverable Task Submission Subsystem

4.2.5. Progress Log Subsystem

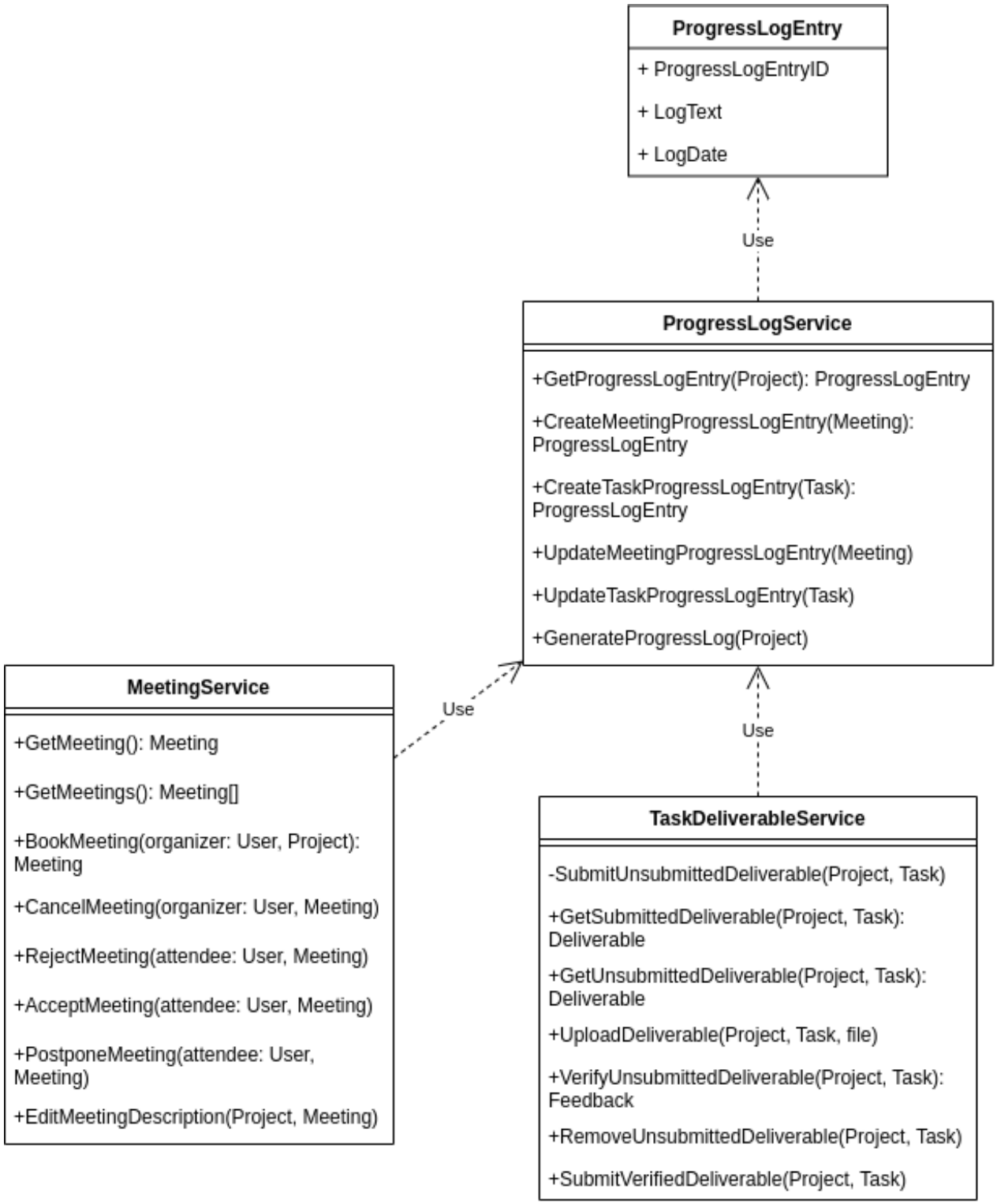

Figure 15: Class Diagram - Progress Log Subsystem

4.2.6. Notification & Reminder Subsystem

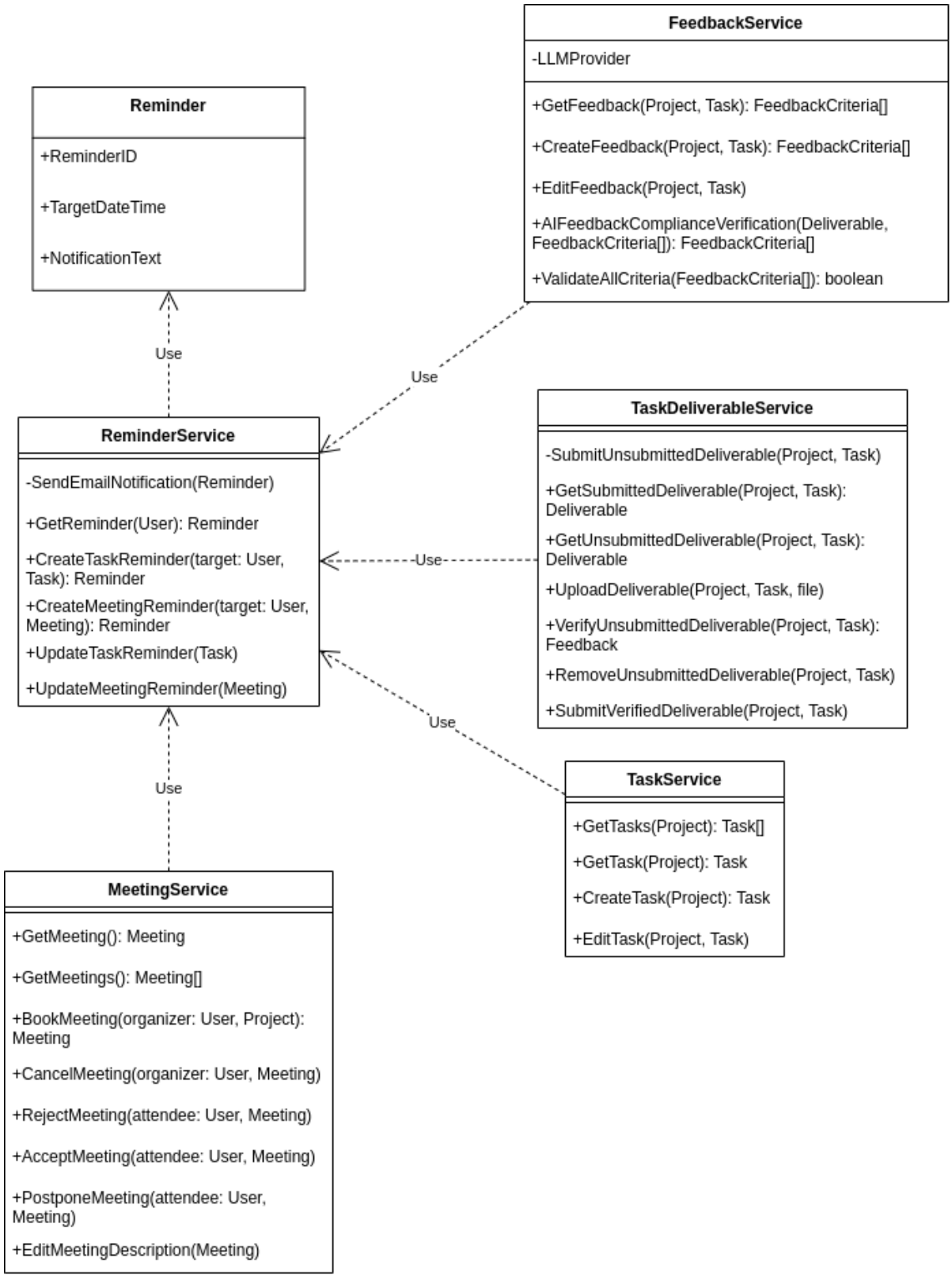

Figure 16: Class Diagram - Notification & Reminder Subsystem

4.2.7. Administrator Subsystem

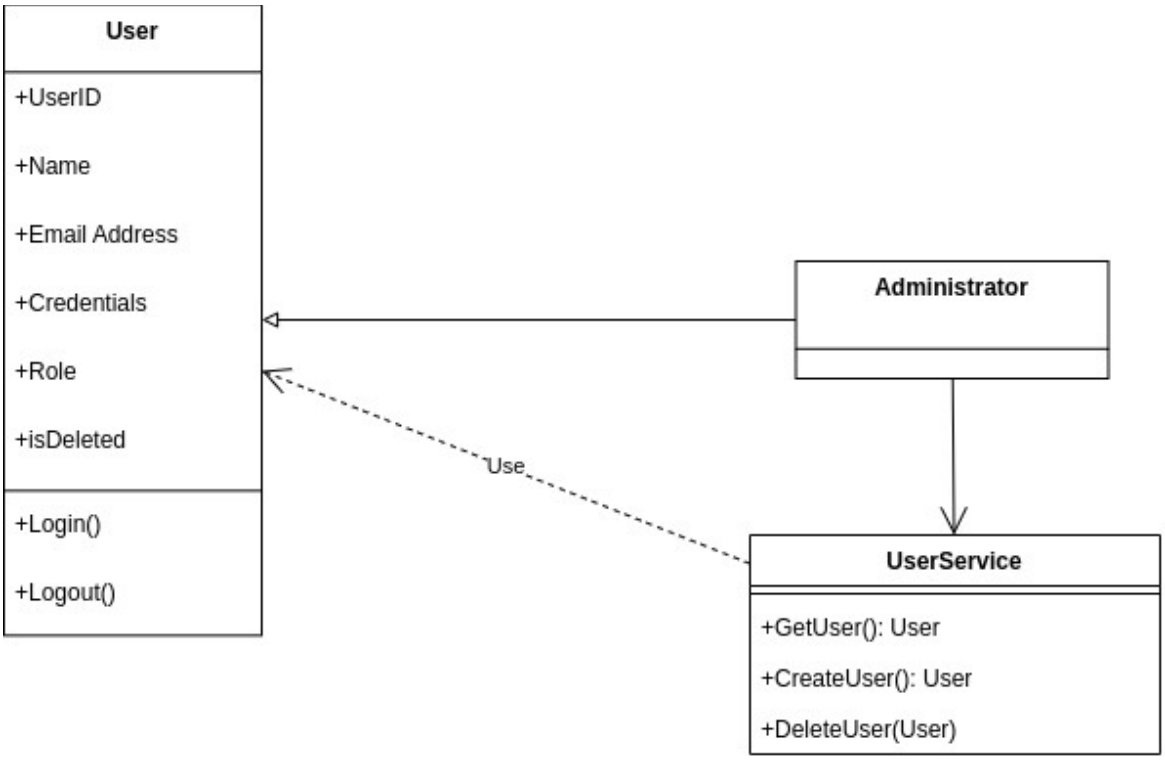

Figure 17: Class Diagram - Administrator Subsystem

4.3. Sequence Diagrams

They aim to provide more detail to the class diagram methods shown in the previous section, highlighting further method invocations and object life-cycle management

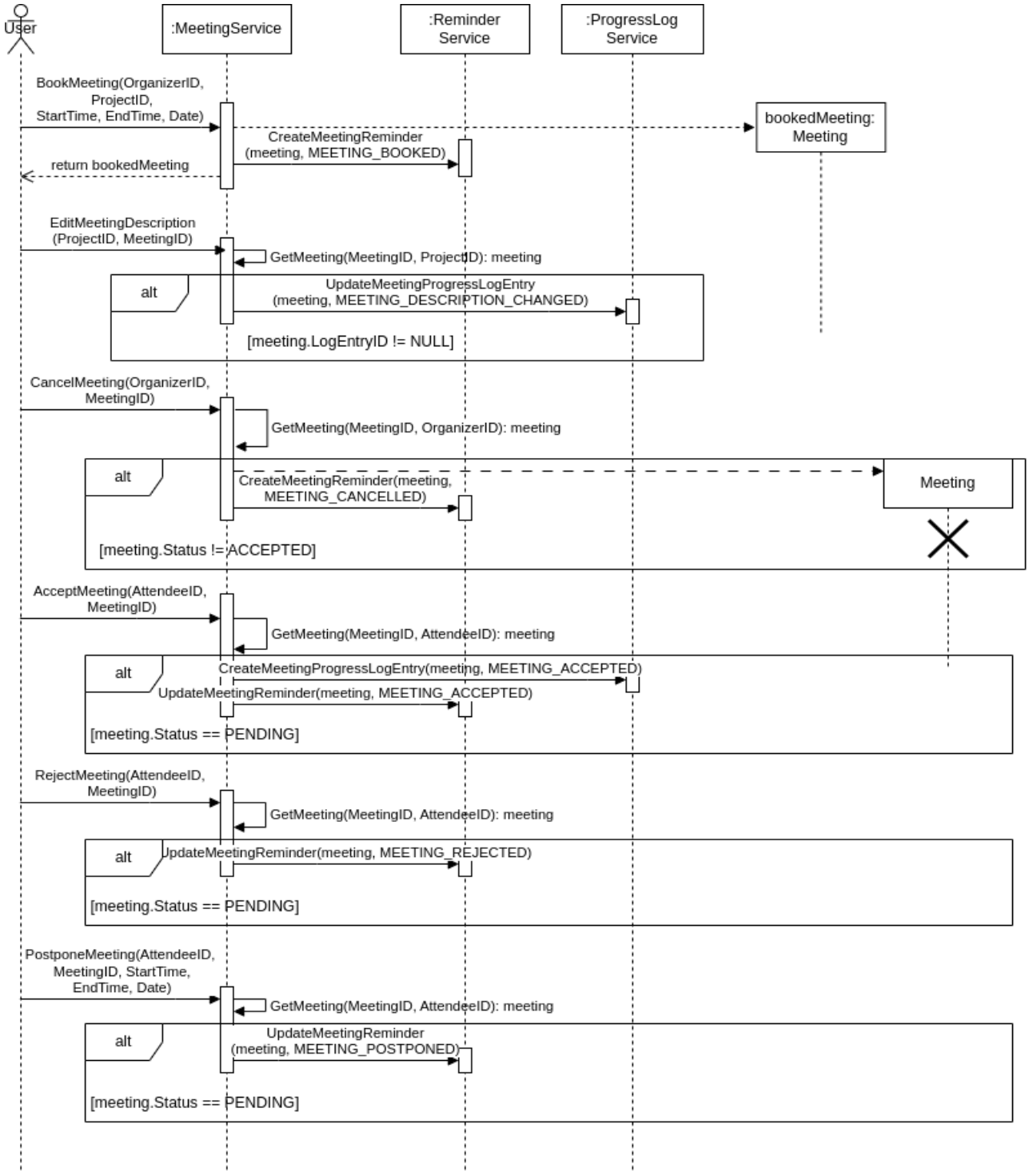

Figure 18: Sequence Diagram: Meeting Scheduling Subsystem

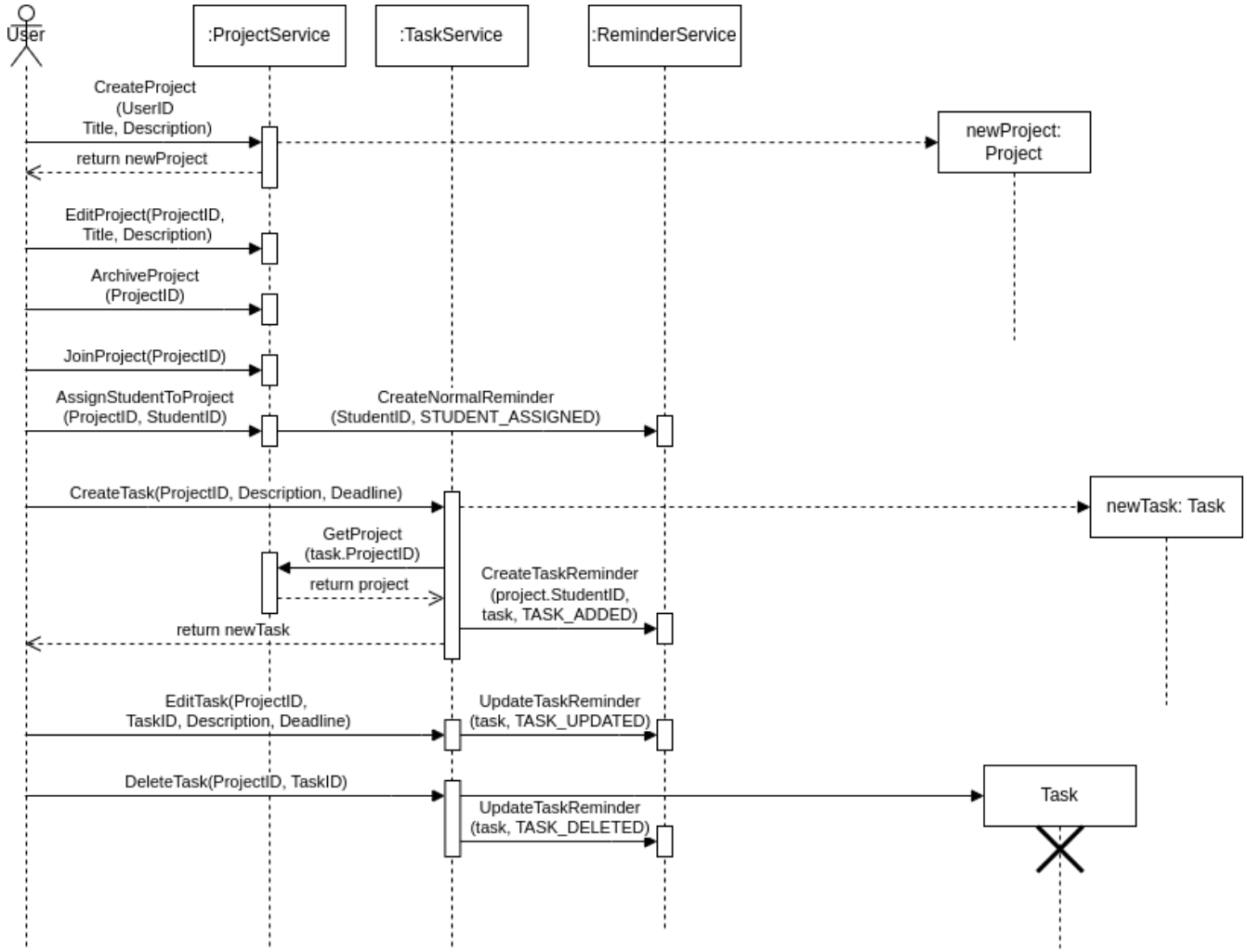

Figure 19: Sequence Diagram: Project Management

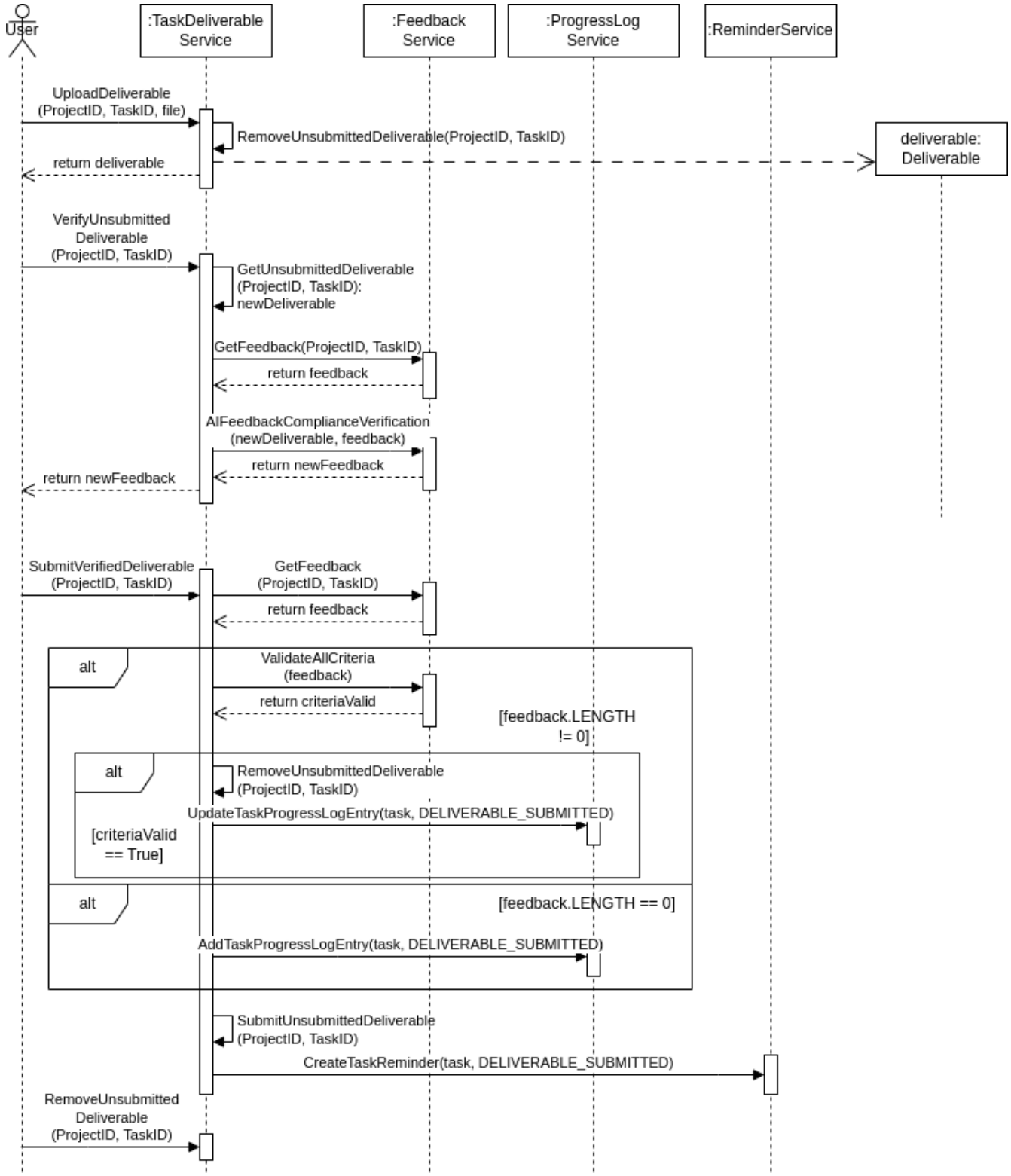

Figure 20: Sequence Diagram: Task Deliverable Management

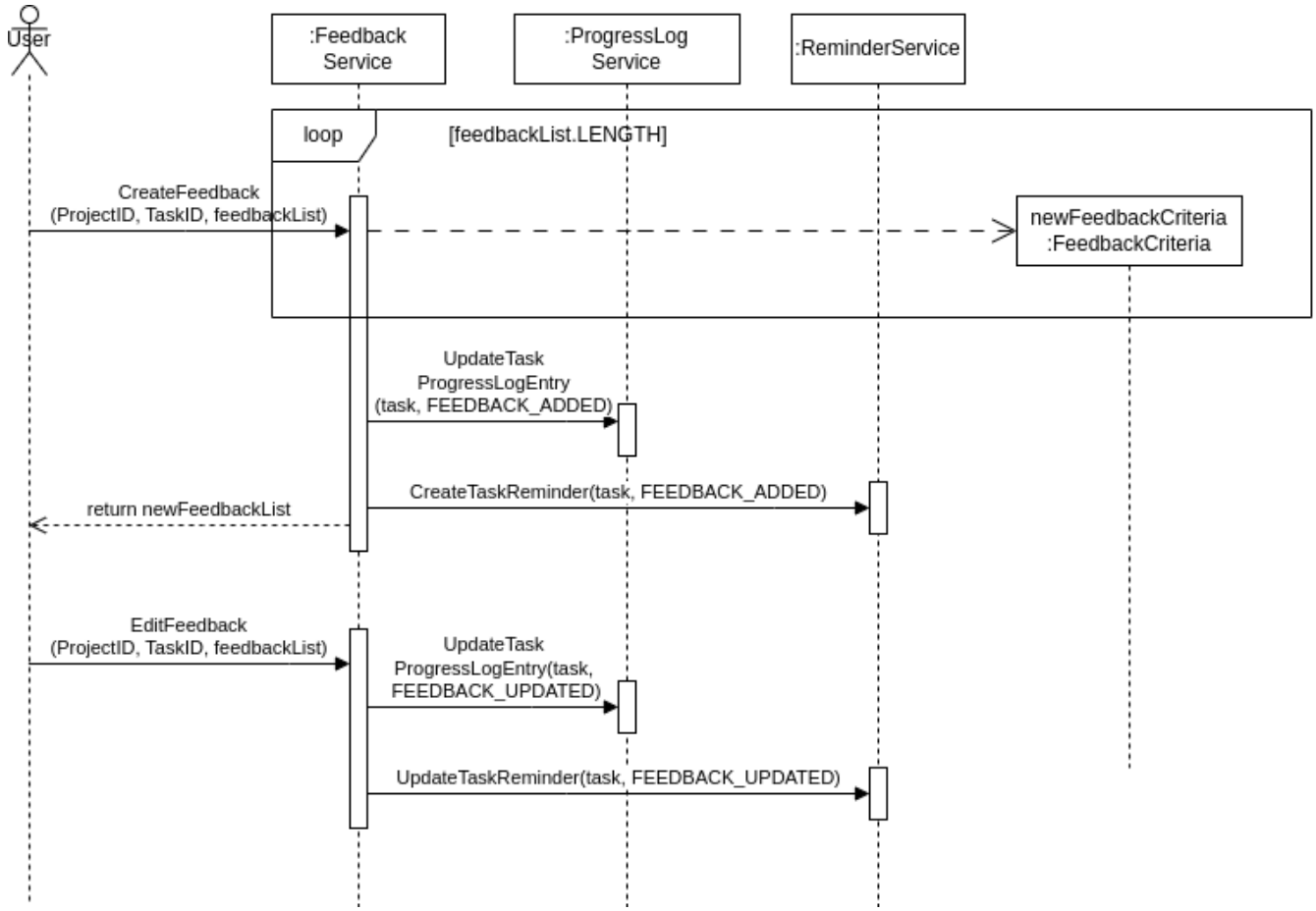

Figure 21: Sequence Diagram: Feedback Management

4.4. Entity Relationship Diagram

The ERD depicts the database tables, their columns and their relations with one another based on the domain class diagram.

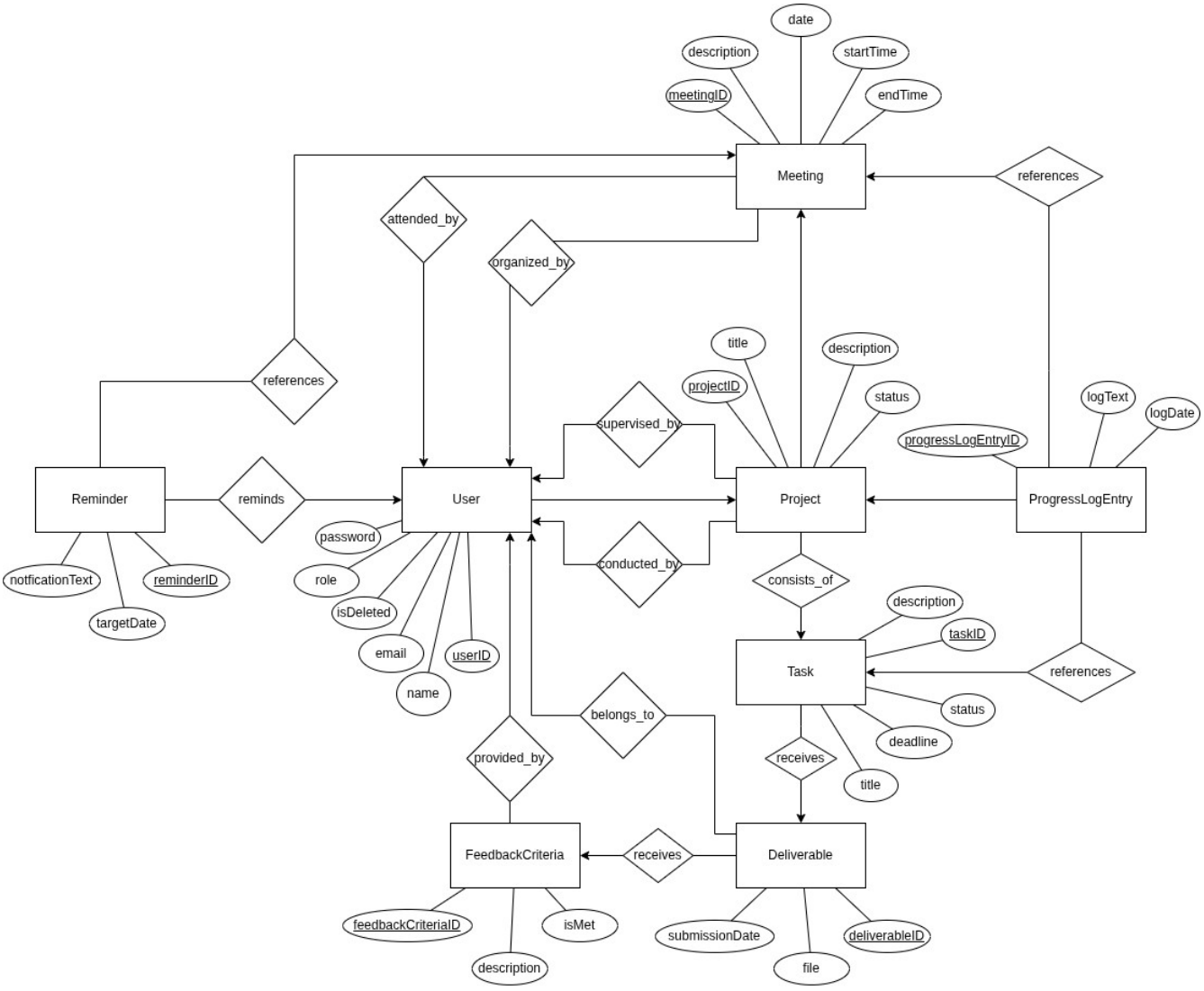

Figure 22: Entity Relationship Diagram

Bass, L., Clements, P. & Kazman, R. (2021). *Software Architecture in Practice* (4th ed.). Addison-Wesley Professional.

https://learn.microsoft.com/en-us/nuget/what-is-nuget

https://learn.microsoft.com/en-us/aspnet/core/fundamentals/middleware/?view=aspnetcore-9.0

https://laravel.com/docs/12.x/csrf

Buljić, I., Kadusic, E., Cvijanovic, T., Hadzajlic, N. and Zivic, N. (2025) Comparative Performance Analysis of Leading Backend Frameworks for Developers. In: (Title of Conference/Proceedings/Book - *Inferred*) **2025 International Scientific Conference on Information Technology and Data Related Fields (INFOTEH)**. (Date/Month of Conference - *Inferred*) Tuzla, Bosnia and Herzegovina: INFOTEH, pp. 1–5. doi: 10.1109/INFOTEH64129.2025.10959250.

Chacon, S., & Straub, B. (2014). *Pro Git* (2nd ed.). Apress. (For Git storage and object model).

Garcia-Molina, H., Ullman, J. D., & Widom, J. (2009). *Database Systems: The Complete Book* (2nd ed.). Pearson Prentice Hall. (For database storage and data structure management).

https://aws.amazon.com/compare/the-difference-between-mysql-vs-postgresql/

https://www.postgresql.org/docs/current/storage-toast.html

https://assets.bytebytego.com/diagrams/0333-what-s-the-difference-between-session-based-authentication-and-jwts.png

Mehta, M. R., Lee, S., & Shah, J. R. (2006). Service-Oriented Architecture: Concepts and Implementation. *Proceedings of the Information Systems Educators Conference (ISECON) 2006*, 23(35).

Newman, S. (2015). *Building Microservices: Designing Fine-Grained Systems*. Sebastopol, CA: O'Reilly Media.

Gupta, M., Sharma, S., Rathi, M., & Singh, A. (2025). Secure API Gateway with Rate Limiting and JWT Authentication. *International Journal for Research in Applied Science and Engineering Technology (IJRASET)*, 13(4), pp. 3559-3562.

Belagatti, P. (2025) *How to Choose the Right SQL Database*. SingleStore. Available at: https://www.singlestore.com/blog/how-to-choose-the-right-sql-database/ (Accessed: 19 November 2025).

Appendix

A **domain model** is a conceptual model of the domain that incorporates both behavior and data. (Fowler, Martin: Analysis Patterns, Reusable object models, Addison-Wesley Longman, 1997. ISBN .)