Proposed System Features

The digital centralized platform will be designed with the goal of alleviating administrative tasks pertaining to dissertation supervision in mind, with emphasis on streamlining the manual and time-consuming process of deliverable feedback compliance checking. Coupled with other features provided by the project management system, the expected outcome is to improve supervision quality and also ensure more constructive meetings.

These proposed features of the system are categorized and described below:

Role-Based Access Control

- **Role Assignment –** Users can be assigned roles of supervisor or student by an administrator. The administrator can specify the roles of users before they have registered in the system.
- **Authentication & Authorization -** Users need to provide their credentials to register their identity, and re-enter them, if necessary, on subsequent visits to the platform. Their assigned roles will be used as the basis for determining which parts of the system they will have access to.

Project & Assignment Management

- **Project -** Each project will include a meeting scheduling, progress log and reminder section as well as a list of assignment entries with their completion status.
- **Assignment –** Each assignment will allow for submission/re-submission of deliverables and for the provision of feedback for a given deliverable submission. For re-submissions, an interface will be provided to access the version history of the previous deliverable documents.
- **Assignment Document Version Control –** Users can select previous versions of deliverables along with their feedback. These deliverables and feedback are available for download.
- **Project Management –** Supervisors can view a list of their projects in the form of entries describing the overall progress of the project and student working on it. They will also be able to update and delete projects. A student can only view the project they are assigned to.

- **Assignment Management –** Supervisors will be able to create, update and modify assignments in a given project. Reminders for assignments are created/updated upon creation or modification of an assignment respectively. Students can only view and access assignments.
- **Assignment Feedback –** Assignments requiring feedback will have their corresponding assignment entry status' changed to remind the supervisor to provide feedback. For a given assignment, a supervisor can provide feedback on the latest deliverable submission.
- **Deliverable Re-submission with AI Feedback Compliance Checking –** Natural Language Processing techniques, not excluding the use of Large Language Models, will be used to assess whether the new deliverable submitted complies with previous feedback.
- **AI Feedback Compliance Evaluation Overriding –** After the automated feedback compliance process, an interface will be provided to override the results and prompt the user to either proceed with submission or abort the operation.

Progress Tracking

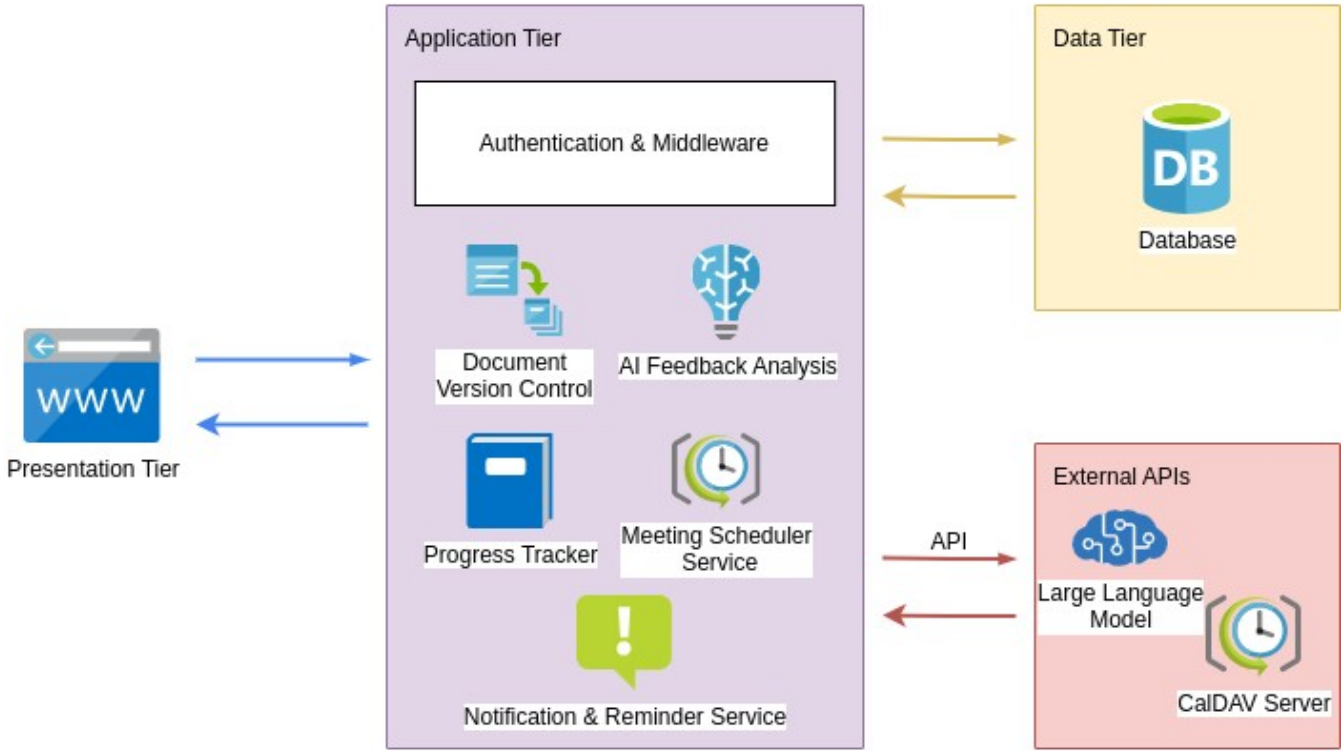
- **Progress Log –** Each deliverable submission with its corresponding feedback and date is recorded into a progress log. The user will be able to download a report of the progress log, with optional metrics such as meeting attendance and assignment completion.

Reminder Notification & Meeting Scheduling

- **Calendar Sharing –** Supervisors can share their calendar with students to reserve meetings in advance and improve attendance rates.
- **Email Reminders –** Meeting and assignment notification reminders are sent automatically via email within a notice period.

Evaluation & Choice of Tools for Implementation

Web Application Architecture



The architecture of choice for the application will be a 3-Tier Architecture, where there is separation of concerns in terms of presentation tier for user experience, application tier for business logic and the data tier for abstracting data access.

Comparison with Monolithic & Micro-services Architecture

Compared to a Monolithic design in which components are tightly coupled, loose coupling of components in the 3-Tier Architecture allows the system to be scalable and faults are isolated within the respective tiers, making maintenance easier. Additionally, it avoids the need of developing and maintaining full-fledged HTTP API endpoints for layers to communicate with one another – a requirement in Micro-services architecture.

Database

	MySQL	SQL Server	PostgreSQL
Cost & Licensing	Open-source	High licensing costs for professional use	Open-source
Architecture	RDBMS (see appendix)	RDMS	ORDMS (see appendix)
Performance	Optimized for read-heavy operations	Optimized for large data volumes in enterprise environments	Optimized for complex queries and write-heavy operations
Concurrency	Low locking granularity (table and row level locking) that affects concurrency	Sophisticated control and concurrency mechanisms	Sophisticated control and concurrency mechanisms
Ecosystem	Extensive community support	Heavily integrated with Microsoft technologies such as .NET and features comprehensive documentation	Moderate community support

The database chosen is MySQL since it has extensive community support, is open-source and does not contain excessive features which will end up being unused.

Application Server Tier Framework

	.NET (C#)	Express.js - ECMAScript/Javascript	Django (Python)
Performance	Fastest performance provided by compilation of C# code	Medium performance since ECMAScript code runs on the V8 engine which supports JIT compilation (see appendix)	Slowest performance as python code is purely interpreted by default CPython interpreter
Development Speed & Experience	Moderate development speed Extensive integration with IDEs (see appendix) such as Visual Studio provide guardrails and features to enhance development experience	Fastest development speed since Express.js requires minimal boilerplate code to setup and run. Requires Typescript for type safety	High development speed due since Django requires minimal setup and boilerplate to run Type safety not inherently supported
Ecosystem	Smaller but more stable	Largest community	Moderate community

	ecosystem backed by Microsoft	ecosystem with Node Package Manager (NPM)	ecosystem with largest support for AI and machine learning toolkits
Security	Provides robust out-of-the-box solutions for authentication and security	Requires 3rd party packages such as Passport for handling different authentication strategies.	Provides robust out-of-the-box solutions for security and user management
Database Access	Provides Entity Framework Core, which is a robust ORM (see appendix) that simplifies database interaction	Relies on 3rd party ORMs for database access	Includes a robust ORM that simplifies database interaction

.NET is chosen as Application Tier framework due to the availability for a wide range of out-of-the-box solutions and extensive security, concurrency and performance features it provides, as compared to Express.js and Django.

Presentation Tier Framework

	ASP.NET Razor Pages	React.js
Development Model / Paradigm	Server-Side Rendering .NET as prerequisite for application tier framework and leads to tight coupling of presentation logic with server side logic	Client-Side Rendering Dedicated server required for deployment UI is decoupled from server logic and React adheres to a declarative paradigm where state updates implicitly updates user interface components
Ease of Development	Uses HTML, CSS and ECMAScript to define web pages and state management is simple since server-side data does not have to be fetched via API requests.	Uses JSX to define user interface components. State management is done through React hooks and server-side data must be accessed through API requests.
Ecosystem	Fewer but stable community libraries for complex components	Massive support for community libraries for all types of components
User Experience	User interface not as responsive since web pages often require full page reloads	User interface is highly responsive since React optimizes UI State updates
Maintainability	Lacks modularity	Highly modular

The Presentation Tier framework of choice will be .NET Razor pages since the web pages are largely expected to be static and it avoids the need of deploying another server. Moreover, it has a high degree of interoperability with .NET as Application Tier framework, simplifying development.

Document Version Control

	Integrated Database Table	Dedicated Internal Git Repository
Implementation	Simple to implement Involves storing an original document copy and text diffs for each document re-submission in a database table	High complexity in understanding and implementation of dedicated Git repository Requires deploying and maintaining a dedicated Git server or service and accessing it using a Git client library
Document Retrieval & Storage	Requires formulating complex SQL queries to reconstruct the document state for a specific submission iteration Files are stored in the database	Git uses efficient algorithms to retrieve the document state for a specific iteration/commit Documents are stored in the file system
Storage Efficiency	Good storage efficiency since only differences between initial and subsequent versions are stored	Highest storage efficiency since Git uses sophisticated diff and compression algorithms to store the differences
Performance	Performance tends to suffer with large number of document revisions	Advanced optimizations ensure performance remains consistent across multiple document versions/revisions

Document version control will be handled through custom application logic operating on an integrated database table rather than a dedicated Git repository. The primary reason is that the majority of operations will consist of document retrieval and storage, and thus won’t require advanced features such as branching and rebase, offered by Git. Furthermore, document will be centralized in the database as opposed to Git which stores the documents in the file system, and thus will lack the benefits provided by the DBMS.

Meeting Scheduler Service

	Custom Application Server Logic	Google Calendar API Integration	Radicale (Open-source)
Server Side Implementation	High Complexity since it requires implementing all features from user calendar tables and conflict resolution algorithms from scratch	Moderate Complexity as it requires API key setup and use of Oauth authentication (see appendix) to store credentials for Google accounts	Moderate complexity as Radicale is a CalDAV (see appendix) server that requires deployment and integration via CalDAV HTTP protocols
Client Side Implementation	High complexity since custom logic is required to integrate the calendar data in an interface for users	Low complexity since frameworks such as React supports components that handles interface logic given JSON data from the API calls.	High complexity since custom logic is required to develop an interface that makes use of the CalDAV data
Features	Extendable	Comprehensive features for calendar scheduling tasks	Robust features for most calendar scheduling tasks
3rd Party Integration	None	Requires users having a Google Account	None
Cost	None	API calls are free up to a certain limit after which the client is billed	None

Using an open-source CalDAV server offers marginal advantage over the other options and hence will be backbone of the Meeting Scheduler Service.

Notification & Reminder Service

	Custom Application Server Logic	External Email Service – SMTP (see appendix)	3rd Party Notification Platform
Implementation	Highest complexity since requires writing sophisticated algorithms such as a queuing system and retry logic	Low complexity since it only requires configuration of SMTP client with a mail server’s credentials	Moderate complexity since it requires API key setup and integration of a dedicated SDK (see appendix)
Cost & Scalability	No cost and scalability depends on component	No cost since leverages existing email infrastructure and scalability depends on	API costs involved

	architecture	the email server	

SMTP will be used as the choice for the Notification & Reminder Service due to its low complexity for implementation and ability to leverage existing email infrastructure

AI Feedback Compliance Checker

	Custom Natural Language Processing Algorithm	Large Language Model (LLM) - Google Gemini API
Implementation	High Complexity Requires assigning labels to data manually, model training and refinement	Moderate Complexity Requires API key setup and workflows making use of prompt engineering techniques
Cost	High initial costs due to computing resources required for training but low running costs once deployed	API costs involved
Security & Privacy	Highest control since all data processing is done within the application server	Low control since data privacy is dependent on the data practices and policies of the LLM provider

Standard supervised classifiers, which include Naive Bayes, Logistic Regression and Bert-based neural networks work under the assumption that a document has a finite, pre-defined set of classes and won’t work as intended with unseen classes. Therefore, the custom NLP algorithm must make use of zero-shot learning, which aims to check the presence of an arbitrary feature. Another approach is to leverage LLMs to check for feedback compliance through prompt engineering techniques such as Retrieval Augmented Generation (RAG). The choice of implementation is still to be decided.

Appendix

RDBMS - A software system that enables you to define, create, maintain, and control access to relational databases.

ORDBMS - An object–relational database management system, is a database management system (DBMS) similar to a relational database, but with an object-oriented database model: objects, classes and inheritance are directly supported in database schemas and in the query language.

JIT Compilation (V8 Engine) - Just-In-Time (JIT) compilation is a compilation process in which code is translated from an intermediate representation or a higher-level language... into machine code at runtime, rather than prior to execution. This approach combines the benefits of both interpretation and ahead-of-time (AOT) compilation.

AOT Compilation - Ahead-of-Time (AOT) compilation is a technique where source code (or an intermediate representation of it) is compiled into native machine code before the program is executed. This is in contrast to Just-in-Time (JIT) compilation, which occurs at runtime.

IDE (Integrated Development Environment) - An integrated development environment (IDE) is a software application that provides comprehensive facilities for software development.

ORM - Object-relational mapping is the process of abstracting the connection between programming language entities (objects) and their corresponding database elements. It is a software layer that translates object data to the underlying database, abstracting database details from the programmer.

Entity Framework Core (EF Core) - Entity Framework (EF) Core is a lightweight, extensible, open source and cross-platform version of the popular Entity Framework data access technology. EF Core can serve as an ORM, which enables .NET developers to work with a database using .NET objects.

OAuth Authentication - OAuth is a protocol for extending user authorization across multiple applications without sharing the user's identity authentication data with those applications.

CalDAV - CalDAV models calendar events as HTTP resources in iCalendar format, and models calendars containing events as WebDAV collections.

WebDAV - Web Distributed Authoring and Versioning is a set of extensions to the Hypertext Transfer Protocol (HTTP), which allows user agents to collaboratively author contents directly in an HTTP web server by providing facilities for concurrency control and namespace operations.

SMTP - A TCP/IP protocol used for sending electronic mail messages between servers.