
# CITS55026 – IT Capstone Project Specification and Plans

**Group**: 8
**Project**: AI Study Planner
**Client**: Maryam Vettoor, Paul Lloyd, Feba Chacko (UWA)

**Group Members:** Mei Li, Mengting Zhang, Sheikh Muhammad Hassan Ali, Vincent Ma, Srindidhi Perala, Zachary Baker, Jinyoung Lee

# Table of Contents

# 1: Problem Statement

## The Problem

The University of Western Australia (UWA) currently relies on a manual process for creating student study plans, which primarily involves using Excel spreadsheets and online information. This process is not automated and struggles with the complexity and non-uniform nature of the underlying course data.

The rules governing course structures, such as unit prerequisites, co-requisites, incompatibilities, and availability, are stored in a "user readable," natural language formats that are not easily parsed by traditional, logic-based software systems.

Previous attempts at creating an online study planner had some success but poses a lot of work for staff to redesign current systems and data to fit the stricter logical requirements.

This lack of an automated, intelligent tool creates challenges for the staff who need to create these planners by hand, the major cost being time. The lack of open access to these planners is also an opportunity to provide them to new prospective students.

## The Proposed Solution

This project aims to develop an intelligent study planner that leverages Generative AI to overcome the challenge of processing unstructured course data.

The system will function as a publicly accessible, open platform where a user can select a course and have the AI automatically generate a valid draft study plan.

The core purpose is to provide students with a flexible, user-friendly tool that can build a correct study plan while allowing for customisation and ensuring all course rules are met.

## Minimum Viable Product (MVP) Key Deliverables

The primary focus of the project is the development of the back-end study planning engine, with the user interface being a secondary priority. The client has agreed that the MVP will include the following core features:

- **AI Planning Engine**: The system will accept course data (initially for a few selected courses) including core units, availability, and prerequisite rules, provided in a CSV or XLSX format. The AI will process this data to generate a valid, foundational study plan.
- **Core Unit Prioritisation**: The initial AI-generated plan will prioritise the placement of core units correctly within the study plan.
- **Student Customisation Interface**: A simple, web-based front-end will be developed to visually represent the study plan. This interface will allow students to modify the plan by dragging and dropping units into different semesters.

- **Selection Validation**: When a student modifies the plan, the system will provide feedback. It will validate the changes against course rules and reject invalid moves, such as unmet prerequisites, unit unavailability, or exceeding the maximum number of units in a semester.
- **PDF Export**: As a primary output, users will be able to export their finalised study plan to a PDF document.
- **Open Access**: The tool will not require a user login, functioning as a publicly accessible platform for guest users.

## Minimum Viable Product (MVP) Out-of-Scope

The above mentioned deliverables are considered **in-scope**. In the interest of absolute transparency and communication the following areas are considered **out-of-scope**, at least for the purpose of the initial minimal viable product:

- **Prior Learning and Advanced Standing**: The system will not have functionality for students to input or account for previously completed units, credits, or advanced standing. The planner is designed from the perspective of a student undertaking a course from the beginning.
- **User Accounts and Persistent Storage**: The MVP will function as a guest-only platform. There will be no user login system, and study plans will not be saved on the server between sessions.
- **Full Course Catalogue**: The initial product will only support a few selected courses and their core units to serve as a proof-of-concept. It will not include the entire UWA course catalogue.
- **Advanced Natural Language Queries**: While the back-end uses AI to interpret rules, the front-end will not support advanced, free-text student queries like "add units that I might enjoy" or "show me a plan with no Friday classes". This functionality is designated as a "second-round feature" (see future phases below!).
- **Polished User Interface**: As agreed with the client, the primary focus is on the back-end AI engine. The user interface will be simple and functional but they did not specify any requirements beyond this.

## Assumptions

- The course data (units, prerequisites, etc.) provided by the client in the CSV/XLSX file will be accurate and complete for the courses in scope.
- The client will be available for fortnightly meetings and for ad-hoc questions via MS Teams as agreed.
- The XLS/CSV data provided by our client will maintain the same schema now & in the future. This should be communicated if it hasn't already, otherwise future imports will break.

## Constraints

- The project team will have no access to UWA's internal student information systems or databases.
- The project is to be completed within the 12-week semester timeline.
- The solution must be a standalone web application and will not be integrated into the existing UWA student portal for the MVP.

## Future Phases / Post-MVP Features

This section recognises the broader vision of our clients as well as a more polished and well-rounded product. As a reminder a MVP is only the absolute bare-bones requirement to achieve “go live”:

- **Natural Language Input**: Allowing students to type requests in plain English, such as "I prefer units with a lot of programming" or "Show me a plan that avoids Friday classes," and having the AI adjust the plan accordingly.
- **Broader Course Support**: Expanding the system to include foundational units, option units, minors, and a wider range of UWA courses.
- **User Accounts**: Implementing a login system for UWA staff to save and manage multiple study plan versions over time (for quick print & sending to students)
- **A more polished**, professional interface. Front-ends can always use more work, look better on tablets & mobiles, etc.

## User Stories

The following user stories brainstormed by the team will help translate the client's needs into functional goals for our project and guide the Agile development process.

***As a new UWA student**, I want the system to generate a valid foundational study plan based on my chosen course, so that I can get a clear and correct starting point without feeling overwhelmed by unit rules.*

***As a student planning my degree**, I want to be able to drag-and-drop units into different semesters, so that I can customize my study plan to fit my personal preferences and see the consequences of my changes.*

***As a student modifying my plan**, I want the system to immediately warn me if I make an invalid change (like missing a prerequisite or choosing an unavailable unit), so that I can avoid making mistakes that would disrupt my graduation timeline.*

***As a student**, I want to export my final study plan to a PDF, so that I have a permanent, shareable copy for my records or to discuss with a student advisor.*

***As a prospective student,** I want to access the study planner without needing a login, so that I can explore potential course paths at UWA before I enrol.*

***As a member of UWA Student Services**, I want an AI-powered tool that can interpret the complex, natural-language prerequisite rules, so that we can provide students with a reliable planning tool without needing to manually restructure all of the university's curriculum data*

## Snow Cards

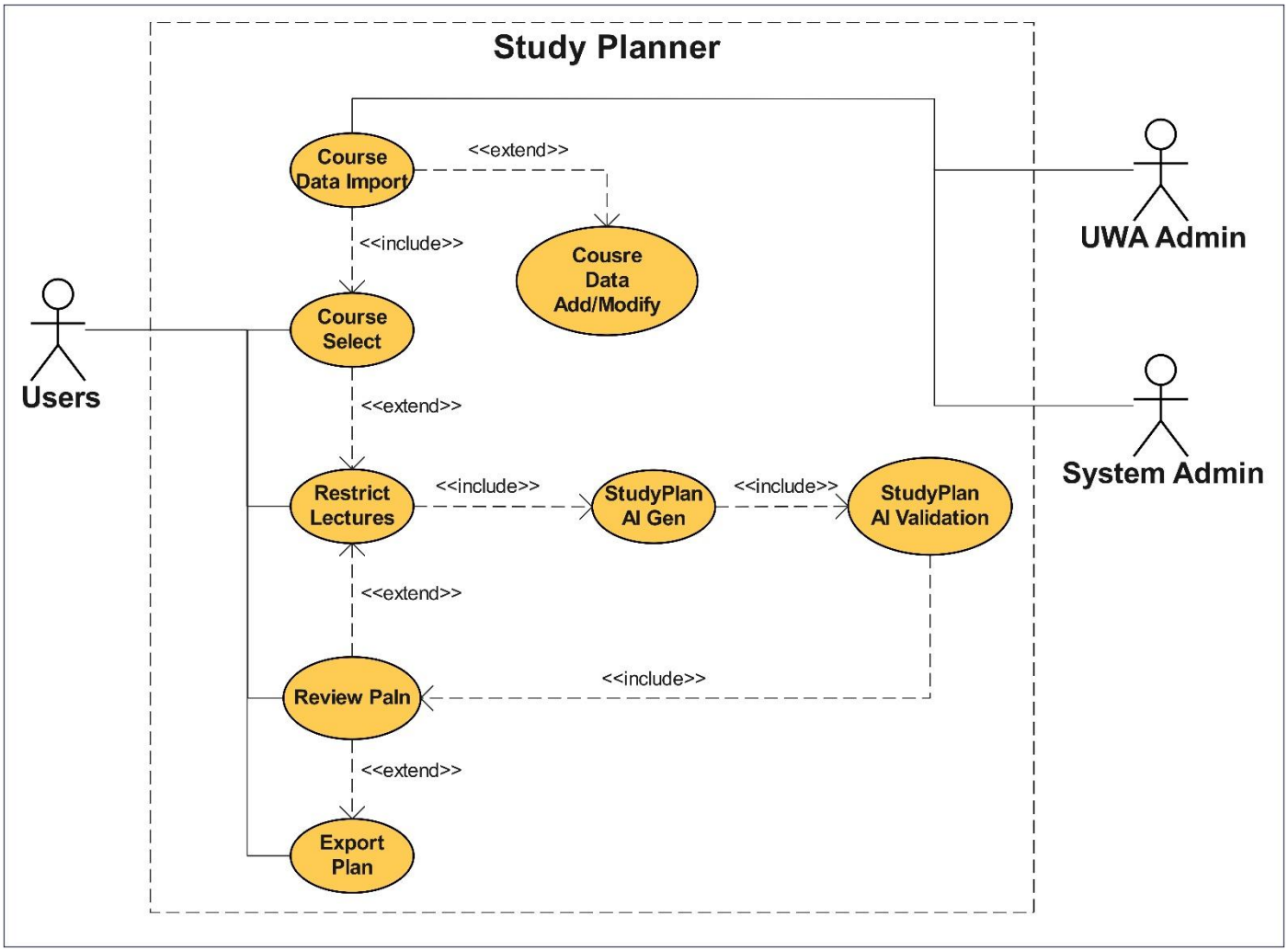
For what we consider the most important user-stories above, we have created a couple of snow-cards to drill a little deeper into the requirement. One is from the perspective of our customer (the University) and the other, the end user. More may be created later to assist in development.

| ID: 001 | AI Interpretation of Prerequisite Rules |
|---|---|
| User Story | As a member of UWA Student Services, I want an AI-powered tool that can interpret complex, natural-language prerequisite rules, so that we can provide students with a reliable planning tool without needing to manually restructure all of the university's curriculum data. |
| Conversation / Acceptance Criteria | 1. Given a unit with a simple prerequisite (e.g., "Prerequisite: CITS1401"), when the AI generates a plan, then CITS1401 must be scheduled in a semester before that unit.<br>2. Given a unit with a complex prerequisite (e.g., "Prerequisite: Enrolment in the Master of Professional Engineering and CITS2200"), when the AI validates a student's plan, then it must correctly identify if both conditions are met.<br>3. Given unstructured data from the Units.csv file, when the system starts, then it must successfully load this data into the database for the AI to query.<br>4. Given the system is live, when a student customizes their plan, then the AI's rule interpretation must be used to validate the changes. |
| Technical Notes & Considerations | This will be implemented using the multi-step AI pipeline discussed with the facilitator. Specifically, the "Prerequisite Analysis" step.<br>The database will need to store the prerequisite text verbatim from the CSV.<br>The prompt for this step needs to be carefully engineered to handle logical operators like "AND," "OR," and co-requisites.<br>The primary risk is AI inaccuracy (R-01). A separate, hard-coded validation layer might be needed as a backup for the most common prerequisite patterns. |

| ID: 002 | Student Plan Customisation via Drag-and-Drop |
|---|---|
| **User Story** | As a student planning my degree, I want to be able to drag-and-drop units into different semesters, so that I can customize my study plan to fit my personal preferences and see the consequences of my changes. |
| **Conversation / Acceptance Criteria** | 1. Given a study plan is displayed on the screen, when a student drags a unit from Semester 2. to Semester 1, then the UI should visually update to reflect the change if it's valid.<br>2. Given a student attempts to move a unit to a semester where its prerequisites are not yet met, when they drop the unit, then the move should be rejected and a clear error message must be displayed.<br>3. Given a student attempts to move a unit into a semester that is already full (e.g., more than 4 units), when they drop the unit, then the move should be rejected with an appropriate warning.<br>4. Given a student successfully moves a unit, when they click the "Export to PDF" button, then the exported PDF must show the new, updated layout of the plan. |
| **Technical Notes & Considerations** | The front-end will be built with HTML, CSS, and JavaScript, using a library like SortableJS or the native HTML Drag and Drop API to handle the user interaction.<br>Each time a "drop" event occurs, an API call will be made to the Flask back-end to trigger the real-time validation logic (FR-6)<br>The validation must be done in a reasonable time (NFR-3) to ensure a good user experience. This reinforces the need for an efficient database backend.<br>The UI/UX should be simple and intuitive, as usability (NFR-1) is a key requirement. |

## Use Case Diagram

Based on the user stories defined above, the following UML Use Case Diagram illustrates the key actors (students, administrators) and their interactions with the system. This diagram provides a visual summary of the functional scope of the MVP.




## Function & Non-Functional Requirements

Functional & non-functional requirements are implied in the above MVP deliverables and user-stories, however explicitly stating them is standard practice in software engineering and helps clarify the needs of the project as well as guide development practices.

### Functional Requirements

- **Data Ingestion**: The system shall be able to ingest and parse course curriculum data from a provided CSV or XLSX file.
- **AI Plan Generation**: The system shall use a Generative AI model (Chat GPT 3.5?) to produce a valid, personalized study plan based on user-selected courses.

- **Core Unit Prioritisation**: The AI-generated plan must correctly place all mandatory core units according to their prerequisites and availability.
- **Visual Plan Representation**: The system shall display the study plan in a clear, semester-by-semester visual format for the duration of the course.
- **Manual Customisation**: The system shall allow users to modify the plan by dragging and dropping units between semesters.
- **Real-Time Validation**: The system must re-evaluate the study plan upon modification and provide feedback on errors, including unmet prerequisites, unit unavailability, and credit overload.
- **PDF Export**: The system shall provide a function to export the current view of the study plan to a PDF file.

## Non-Functional Requirements

- **Usability**: The system shall be designed for a first-time student user, with an intuitive interface that requires no prior training.
- **Accessibility**: The system shall be a publicly accessible web platform and will not require any user authentication or login.
- **Performance**: Real-time validation feedback on user modifications should be near-instantaneous, with any noticeable delay being unacceptable for a smooth user experience.
- **Scalability**: For the MVP, the system must support a single user session effectively. (Note: Scalability to multiple concurrent users is a post-MVP concern).
- **Extensibility**: The system's back-end should be designed in a way that allows for the addition of new courses and rules in the future.
- **Security**: The system will not handle or store any personal student data, mitigating significant security risks.

# 2: Client Communication and MVP Agreement

## Evidence of Communication

On **Tuesday, July 29, 2025, at 1:00 PM**, customer representatives Maryam Vettoor and Feba Chacko held a meeting to gather first requirements. The meeting was held in a hybrid style, with some people attending in person at the Ian and Gerty Ewan Collaboration Room and others joining remotely through Microsoft Teams.

The team gathered a lot of information about the project's goals, functional and non-functional needs, and the client's expectations for the MVP during this session. The conference was completely recorded, and:

- **Meeting Minutes**: a summary of the main topics and choices made.
- **Full Transcript**: a record of the whole conversation, including clarifications and the reasons for the requirements.
- **Initial Interview Questions Document**: made to help gather needs.

We have saved these materials in the FILES part of our Teams group, where they can be used to prove that we worked with clients.

Files that can be used in Teams FILES section:

- 2025-07-29 - Initial Interview Questions.docx
- 2025-07-29 - Meeting Minutes.docx
- 2025-07-29 - Meeting Transcript.docx
- 2025-07-29 - Meeting Recording.mp4
- 2025-08-03 - Meeting Minutes.docx
- 2025-08-08 - Meeting Minutes (Group Facilitator).docx
- 2025-08-15 - Communication with Customer Stakeholder.pdf

A formal progress email was issued on **August 15, 2025**, to confirm agreement and next steps.

## Agreed Communication Plan

To maintain alignment and transparency throughout the project lifecycle, the following protocol has been agreed with the client.

- **Main Communication Channel**: A dedicated Microsoft Teams channel will be the main place for all daily communication, short questions, and file sharing.
- **Scheduled Meetings to Check on Progress**: Every two weeks, there will be formal project review meetings that everyone who is important to the project has agreed to attend. If there are urgent problems or important milestones, further ad-hoc meetings may be planned.

- **Updates on Progress**: After each major project phase or deliverable is finished, a concise, organised report will be sent out. Reports will list finished tasks, tasks that are still going on, problems that need client feedback, and forthcoming milestones.
- **Attendance at Meetings**: Only the team members who need to be there will be at each client meeting. After the meeting, the rest of the group will get updates.

## MVP Agreement

The following is the MVP scope that was designed during the first meeting with our client. This specifications document will be sent to our client no later then Tuesday 19th August to review and confirm.

The main deliverable is a working AI-powered study planner that:

- Accepts course and unit data from stakeholders in CSV or XLSX format.
- Makes a personalised study plan that puts key modules first based on rules, prerequisites, and availability.
- Allows students to make simple changes (like dragging and dropping unit schedules) and checks them right away.
- Gives you the validated plan in PDF format.

Limitations on the project's scope:

- No login system, thus guests can use it as an open platform.
- Concentrate on a few courses with important units to test the idea's viability.
- Core functionality is more important than UI development.
- The main purpose of AI integration is to interpret and validate rules. Advanced natural language features will be added in the future.

Out of Scope for MVP:

- Full integration with university systems.
- Saving study plans over and over again.
- Advanced AI plain language requests, like "Add all units with a lot of programming." These are set aside for possible future versions.

# 3: Project Planning and Management

## Project Management Tools

- **MS Teams**: Used for all client communication and internal team collaboration and meetings. Also used for file-storage for DOCX, XSLX, PDF and similar files.
- **GitHub**: Used for source code version control, code documentation, and task management via GitHub Projects/Issues.
  *GitHub Project:* https://github.com/meiliyuri/AI-Study-Planner-Group8
  *GitHub Project repo:* https://github.com/meiliyuri/AI-Study-Planner-Group8.git
- **Atlassian Jira**: Project management & task assignment
  *Jira project:* https://cits5206-2025s2-gourp8.atlassian.net/jira/software/projects/ASP/summary

## Project Methodology

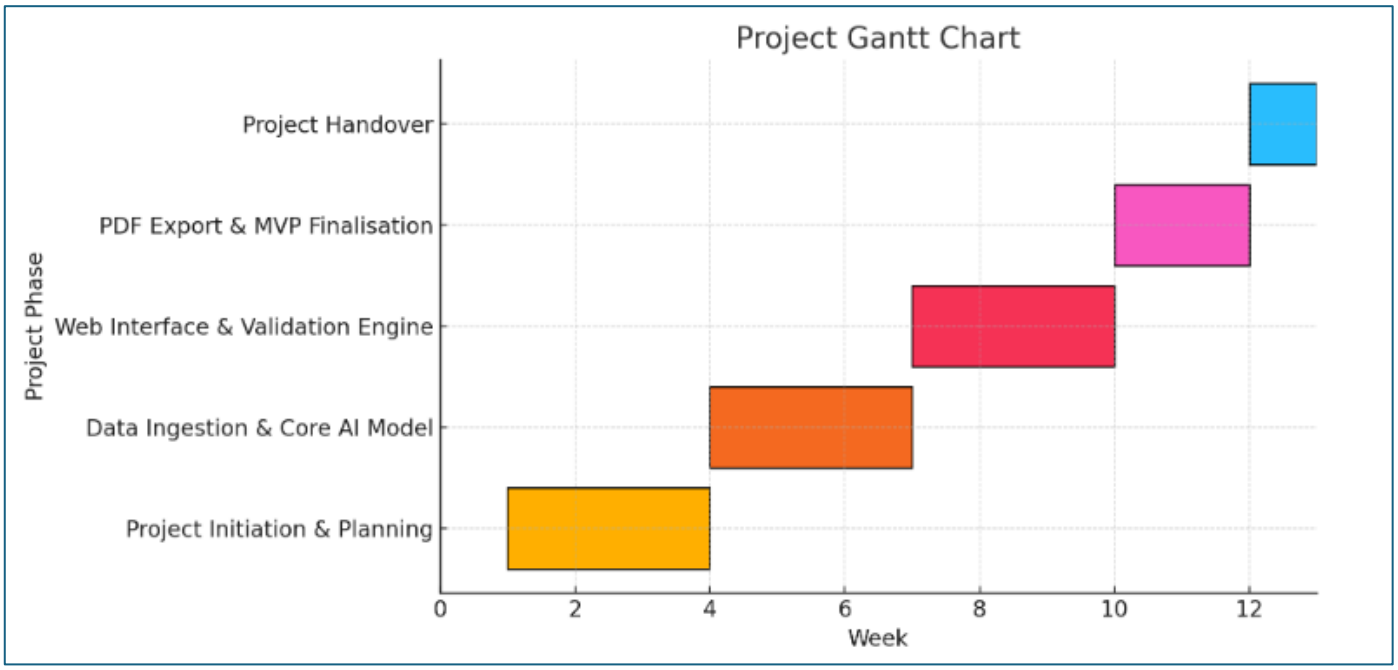
The team will adopt an **Agile development methodology**. The project will be broken down into a series of sprints, each focusing on a specific set of features from the MVP.

This approach aligns with the client's request for iterative reporting and allows the team to adapt to challenges and feedback throughout the development lifecycle.

### High-Level Project Phases & Timeline

| Phase | Sprint Focus | Key Tasks | Estimated Timeline |
|---|---|---|---|
| 1 | Project Initiation & Planning | Finalise and submit Project Spec & Plans document. Set up GitHub repository, Jira & and project boards. Assign initial roles and responsibilities. | Week 1-3 |
| 2 | Data Ingestion & Core AI Model | Develop Python/Flask backend structure. Create a module to parse course data from CSV/XLSX. Build and train the initial Gen AI model to generate a basic plan from the data. | Week 4-6 |
| 3 | Web Interface & Validation Engine | Develop the simple web-based front-end (HTML/CSS/JS. Implement drag-and-drop functionality for units. Develop the real-time validation logic to check for errors on modification. | Week 7-9 |
| 4 | PDF Export & MVP Finalisation | Integrate a library to export the final study plan to PDF. Conduct comprehensive testing of all MVP features. Refine documentation and prepare for final submission. | Week 10-11 |
| 5 | Project Handover | Final presentation and demonstration to the client. Submit all final deliverables and documentation. | Week 12 |

## Project Gantt Chart




## Team Roles and Responsibilities

- **Project Facilitator/Lead:** Zachary Baker
  Responsible for internal team coordination, removing development obstacles, and ensuring the smooth execution of agile processes.
- **Client Communication Lead:** Mei Li
  The main point of contact for client communication, responsible for scheduling meetings, conveying feedback, and managing client expectations.
- **Back-End/AI Development Lead:** Vincent Ma, Srinidhi Perala, Jinyoung Lee
  Responsible for the architectural design and implementation of databases, server logic, and AI model integration and validation. Collaboration with 3 team members will be essential for this role.
- **Front-End Development Lead:** Sheikh Muhammad Hassan Ali, Jinyoung Lee, Mengting Zhang
  Responsible for the design and implementation of the user interface, including HTML/CSS/JS development, ensuring application usability and interaction experience.
- **Testing/QA Lead:** Sheikh Muhammad Hassan Ali, Srinidhi Perala
  Responsible for developing and executing test plans, managing bug tracking, and ensuring the quality and stability of the final product.
- **Documentation Lead:** Mengting Zhang
  Responsible for writing, organizing, and archiving all project documentation, including meeting minutes, technical documents, and user guides.

**Note:** Roles can be shared, and certain tasks (like back-end and front-end development) will require collaboration between team members.

## Project Resources and Repositories

To support collaboration and maintain consistency, we use an integrated toolchain. All final project deliverables, including source code and key documentation, are stored in the GitHub codebase as the official project archive. The table below summarizes the main uses and access methods for each platform.

| Resource | Primary Purpose | Access Link |
|---|---|---|
| **GitHub Repository** | Official Project Archive.<br>Stores the finalized source code and project documents.<br>Contains the official record of meeting minutes.<br>Serves as the authoritative reference for the project team. | https://github.com/meiliyuri/AI-Study-Planner-Group8 |
| **Atlassian Jira** | Agile Project Management.<br>Manages the Product Backlog and user stories.<br>Plans and tracks Sprint progress.<br>Reports and fixes bugs. | https://cits5206-2025s2-gourp8.atlassian.net/jira/software/projects/ASP/summary |
| **MS Teams** | Daily Communication & Collaboration.<br>Used for instant messaging within the team and with clients.<br>Conducts online meetings and quick discussions.<br>Shares draft documents and facilitates informal collaboration. | For daily communication; relevant communication records can be found in Section 2 of the documentation. |

The GitHub repository follows a standardized directory structure to keep all project assets organized and maintainable:

```
AI-Study-Planner-Group8/
├── README.md          # Project overview and resource links
├── Project-Plan/      # Planning documents and specifications
├── Meeting-Notes/     # Archived meeting minutes
├── src/               # Application source code
└── resources/         # Non-code assets (e.g., test data)
```

# 4: Risk and Technology Assessments

## Technology Assessment & Justification

### Chosen Technology Stack:

- **Backend**: Python with the Flask web framework.
- **Frontend**: Standard HTML, CSS, and JavaScript.
- **AI/ML**: A suitable Generative AI model/library (e.g., via API) – *TBA in 1st development sprint*
- **Database:** SQLite

### Justification

The client gave the team freedom to choose the stack, so we selected technologies balancing familiarity, speed of development, and alignment with AI use cases.

- Python/Flask: Chosen for the backend as the team already has strong Python foundations from previous units (CITS1401, CITS5505). Flask is lightweight and well-suited for rapid prototyping.
- AI/ML: Generative AI is essential for interpreting unstructured prerequisite rules. OpenAI's models (e.g., GPT-3.5/4) provide reliable APIs and proven performance in natural language processing.
- Database: SQLite is sufficient for MVP due to the limited course scope, but performance risks exist if scaled. Future iterations will likely migrate to PostgreSQL/MySQL to handle larger datasets and concurrent queries more efficiently.
- Frontend: A minimal HTML/CSS/JS stack keeps the MVP simple. More advanced frameworks (e.g., React) could be adopted in later phases for scalability and better user experience.

This stack ensures quick development for the MVP while leaving room for scalability and future-proofing.

## AI Implementation and Prompt Strategy

This specific component of our product warranted its own section in our planning report after receiving feedback from our facilitator (Frank) and considering how important it is to the core product.

A single-prompt approach to the LLM is insufficient for this task's complexity. We will instead implement a multi-step AI pipeline, where each step has a specific function and a tailored prompt.

This "chain-of-thought" process makes the system more accurate, predictable, and easier to debug.

The AI will be used in two distinct phases: initial plan generation and real-time validation of user modifications. The prompts are constructed programmatically by our system logic based on user actions in the GUI.

A note on the initial plan generation: During testing we will debug to see if we need a 'default' plan for each course to start from, or if this can be done on the fly. It may prove more performant to have a default either manually created or created by an Ai job overnight. TBA.

## Phase 1

**Initial Plan Generation (Triggered by Course Selection)**

When a user selects a course from the interface, the system queries the database for all mandatory units and their rules, then sends a structured prompt to the LLM.

- System Action: User selects "Master of Information Technology" from a dropdown menu.
- Prompt Template:

```
Generate a valid, standard study plan for the "{Course Name}".

# Units to Schedule
{
  "units": ["CITS5501", "CITS5502", "CITS5503", ...]
}

# Rules
{
  "CITS5503": {
    "prerequisite": "CITS5501",
    "availability": ["Semester 1"]
  },
  ...
}

Ensure all prerequisites are met before a unit is scheduled. The standard workload is 4 units per semester. Respond with the plan in JSON format.
```

- Expected LLM Output: A structured JSON object representing the semester-by-semester plan, which is then rendered visually in the front-end.

## Phase 2

**Plan Validation (Triggered by Drag-and-Drop)**

When a user moves a unit, the system sends the entire proposed new plan to the LLM for validation.

- System Action: User drags "CITS5503" from "Year 1, Semester 2" to "Year 1, Semester 1".
- Prompt Template:

```
Validate the following study plan against the provided rules. Is the plan valid? If not, identify the specific unit and the rule that is broken.

# Proposed Plan
{
  "Year 1, Semester 1": ["CITS5501", "CITS5502", "CITS5504", "CITS5503"],
  "Year 1, Semester 2": [...]
}
```

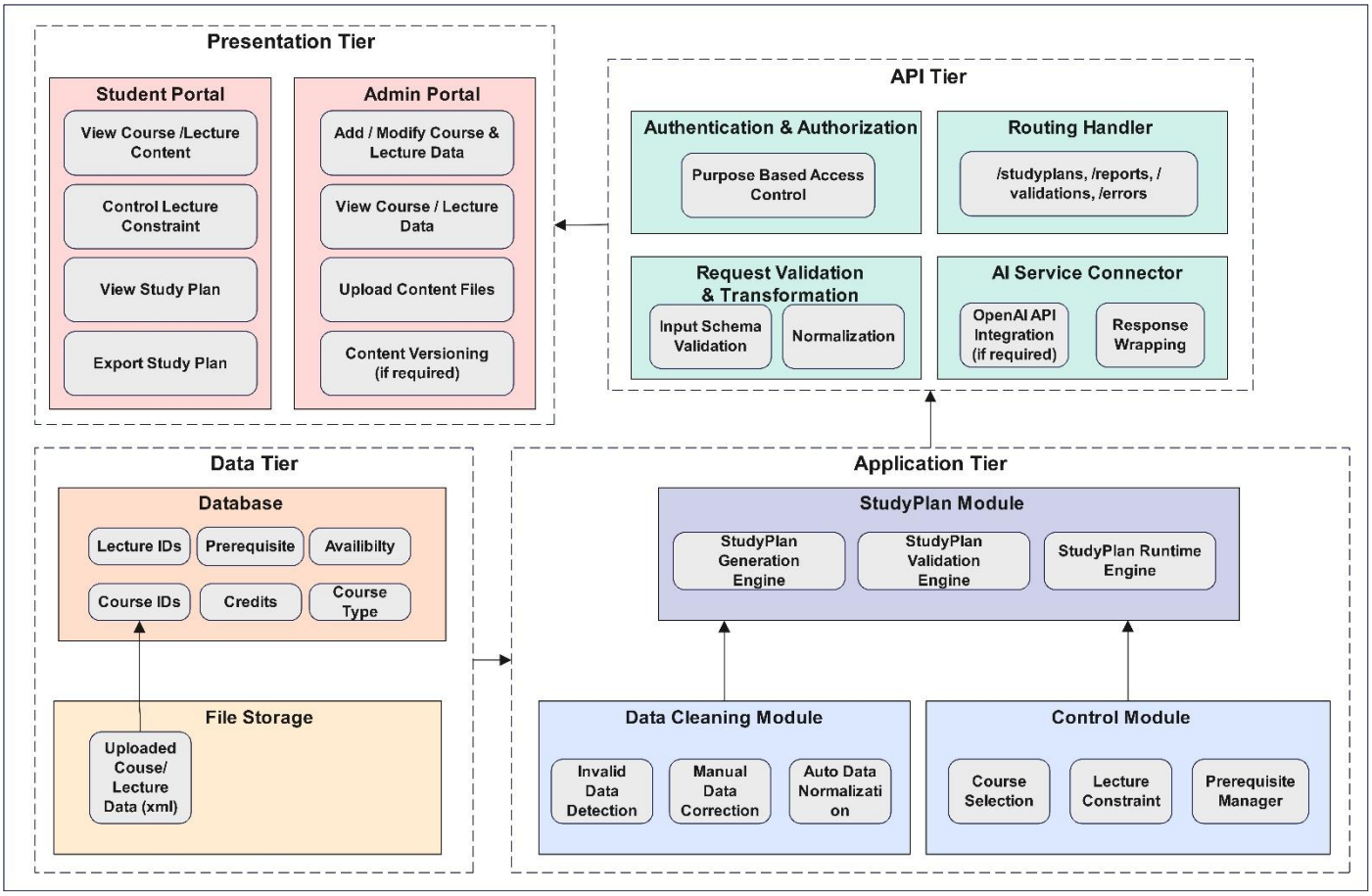
```
# Rules
{
  "CITS5503": {
    "prerequisite": "CITS5501",
    "availability": ["Semester 1"]
  },
  ...
}

Respond in JSON format with a 'isValid' boolean and a 'reason' string.
```

- Expected LLM Output: { "isValid": false, "reason": "Validation failed for CITS5503: The prerequisite unit CITS5501 must be completed in a prior semester." }. The system then uses this response to show an error to the user in the GUI.

## UML System Diagram

The following UML System Diagram provides a high-level view of the AI Study Planner layered architecture. It illustrates how the Presentation Tier interacts with the Application Tier through the API Tier, and how these components integrate with the external AI service and the database.



## Skills Assessment

**Team Strengths**

- Strong foundation in Python, Flask, and Agile web development after completing CITS5505 Agile Web Development, and also earlier CITS1401 Computational Thinking with Python
- Prior experience with parsing structured data and implementing validation logic.

**Skill Gaps**

- Limited prior exposure to AI model integration and prompt engineering.
- Limited knowledge of managing API rate limits, caching, and fallback strategies.

**Mitigation**

- Team will complete official AI API tutorials and share findings in regular group meetings dedicated to programming.
- Buffer time allocated in Sprint 1 for experimentation and training.
- Regular workshops and check-ins with the facilitator to refine prompt design and handling AI inconsistencies.

## Risk Assessment and Mitigation Plan

**Risk Matrix**

| Likelihood | Impact | Category |
|---|---|---|
| Low | Low | Minor inconvenience |
| Medium | Medium | Manageable with extra effort |
| High | High | Threatens MVP delivery |

**Risks & Mitigations**

| Risk ID | Risk Description | Likelihood | Impact | Mitigation Strategy |
|---|---|---|---|---|
| **R-01** | AI Model Inaccuracy: The AI may generate incorrect or logically flawed study plans due to the complexity of natural language rules. | Medium | Medium | The client has confirmed that the AI does not need to be perfect and has a tolerance for some errors. A separate validation layer will be implemented to check the AI's output against hard-coded rules (e.g., prerequisites, availability) to prevent critical errors. The scope is limited to a few courses initially to ensure the model can be trained and tested on a manageable dataset. |
| **R-02** | Scope Creep: The project expands beyond the agreed-upon MVP, adding features like advanced natural language queries ("I like programming units") that threaten the timeline | Medium | High | A clear MVP has been defined and agreed upon with the client. Additional features have been explicitly designated as "second-round features" and are not part of the MVP. The agile project plan will prioritise tasks essential for the MVP. Any new feature requests will be added to a backlog for future consideration. |

| | | | | |
|---|---|---|---|---|
| **R-03** | Data Complexity: The provided CSV/XLSX data may be more unstructured or ambiguous than anticipated, making it difficult for the AI to interpret correctly. | Medium | Medium | Maintain open and frequent communication with the client via the dedicated MS Teams channel to seek clarification on any ambiguous data points. Start with the simplest course structures first to build a robust parsing foundation before moving to more complex ones. |
| **R-04** | Cybersecurity Threats: As a publicly accessible web application, the system could be vulnerable to common web attacks. | Low | Low | The risk is inherently low as the system is designed to be open, requires no user login, and does not store any sensitive or personal student data. All data processed by the system is based on publicly available university course information. Standard security best practices, such as input sanitisation, will be implemented to mitigate risks from any user-provided text. |
| **R-05** | AI Response Inconsistency: The generative AI model may produce different outputs even for the same identical inputs, which making users confused and reducing their trust in the system | Medium | High | Fine-tune the AI model's queries to reduce inconsistency, and clearly communicate to users that the generated plan is an assistant as a one of several valid options. |
| **R-06** | External AI API downtime or deprecation: the system depends on third-party AI APIs, which may become temporarily uncontrolled unavailable in an important session, which causing the study planner to fail for user. | Medium | High | Where possible, the system can provide a different AI APIs options for users to select, or design the system to handle API failures and fully inform users if the AI APIs can not work in uncontrolled situation in advance. |
| **R-07** | Data format changes: the format or structure of the input data provided by the client might change during operation. | Medium | Medium | Build a structure (schema) of input data with a negotiation with clients to reduce the risk or maintain open communication with the client about data updates. |
| **R-08** | Incomplete or changing client requirements: the client may refine or shift their expectations during developing, leading to rework or scope ambiguity | Medium | High | Use Agile methods to develop with sprint review for early feedback as having regular meetings with the client, and maintain a change request log for anything that affects scope or timeline. |
| **R-09** | Inaccurate AI Step-Output: One step in the AI pipeline (e.g., intent parsing) may fail or produce an incorrect output, causing the entire chain to fail. | Medium | High | Implement validation checks and error handling between each step of the AI chain. If a step produces an unexpected output, the system will log the error and can either stop or ask the user for clarification. This granular approach makes debugging much easier than a single-prompt method. |
| **R-10** | Over-reliance on a Single LLM: A specific LLM may not be good at all tasks in the pipeline (e.g., good at creative assembly but bad at strict JSON formatting). | Medium | Medium | The multi-step architecture allows for the possibility of using different models or prompts for different steps. For example, a stricter, more deterministic model could be used for parsing, while a more creative model could be used for the final plan assembly. It may introduce technical debt though, so should be measured. |