Software Requirements Specification
For
Syllabank

07/July/2025

Prepared By :
Chanpirun Srorn , Channon Teng , Vattanac Ny

Paragon International University
8 St. 315, Boeung Kok 1, Toul Kork, Phnom Penh

Table of Contents

Revision History

Version	Date	Name	Description
0.1	28/06/2025	Chanpirun Scrom, Chanon Teng, Vattanac Ny	Initial draft
0.2	29/06/2025	Chanpirun Scrom, Chanon Teng, Vattanac Ny	Second draft
0.3	30/6/2025	Chanpirun Scrom, Chanon Teng, Vattanac Ny	Third draft
0.4	1/7/2025	Chanpirun Scrom, Chanon Teng, Vattanac Ny	Fourth Draft
0.5	2/7/2025	Chanpirun Scrom, Chanon Teng, Vattanac Ny	Fifth Draft

1 Introduction

1.1 Overview

At Paragon International University , students rely on syllabi to understand course goals,expectations,grading and materials.However,accessing syllabi is currently difficult due to the lack of a centralized system. The provost's office, responsible for managing syllabi , stores them manually on Google Drive and often has to request missing documents from lecturers.This makes it challenging for students , especially those applying for transfers,exchanges or further education.

A proper system is needed to improve access and management of syllabi for both academic progress and accreditation requirements.Therefore, a new evaluation syllabi system is being processed to implement new features that rely on the new syllabus upcoming year.

According to evaluation of a new syllabi system, all information follows a new syllabus that will exist in the next academic year.However It is a difficult and inconvenient process of handling the syllabi and syllabi compilation for other purposes.

1.2 Goals and Objectives

The goal of this SRS is to support the development of an upgraded syllabus management system as a proof of concept to demonstrate improvements in instructor usability and prepare for the academic structure and changes planned for the next academic year

The Goals of this SRS are:

1. Enhance the instructor experience in syllabus creation:
    - Provide a more intuitive and streamlined form interface that simplifies the process of creating and submitting syllabi, reducing the redundancy and manual work.
2. Prepare for the upcoming academic year changes:
    - Ensure the system design accommodates the revised syllabus format and evaluation criteria set to take effect in the next academic year.
3. Develop a functional proof of concept for system evaluation
    - Build and deliver a working model to demonstrate the core features, validate design decisions, and gather feedback.

The Objectives of this SRS are:

- Redesign the syllabus to improve the usability and reduce tasks for the instructors.

- Implement form logic that supports upcoming syllabus formats during the upcoming transition period.
- Demonstrate system viability and potential impact through real-use testing.
- Support decision-making for future implementation based on insights gained during the POC or proof-of-concept phase.

1.3 Scope

This project defines the boundaries of a **proof-of-concept syllabus management system** intended to improve the instructor workflow for creating and submitting course syllabi at Paragon International University. It also aims to reflect the upcoming syllabus format changes for the next academic year.

The system will provide instructors with a streamlined digital form to submit syllabi. It will validate the required fields, align with updated formatting standards, and store submitted documents in a centralized location accessible to authorized academic staff. However, this proof of concept is limited in scope and does not represent the final deployment version.

The system **will** support instructors in:

- Filling out, editing, and submitting syllabi using a redesigned interface.
- Receiving guidance on required information based on the new academic structure.
- Submitting syllabi to a central repository for review by academic administrators.

The system **will not**:

- Be deployed university-wide; access is limited to a controlled test group.
- Replace existing syllabus storage methods (e.g., Google Drive) during the testing phase.
- Automatically approve, evaluate, or modify submitted syllabi.
- Provide public access to syllabi or student-facing features.
- Include integration with other institutional systems (e.g., LMS, accreditation systems).

The purpose of this project is to **demonstrate and evaluate core improvements** to the syllabus creation process and gather feedback for future iterations. Decisions about full-scale deployment, advanced features, or third-party integrations will be made after the proof-of-concept evaluation.

1.4 Definition

This section defines terms, acronyms, and abbreviations used throughout the Syllabus Management System SRS. These definitions help ensure consistent understanding across all stakeholders involved in the development and evaluation of this proof-of-concept system.

General Terms

- **Academic Staff (Faculty)** – University staff involved in academic activities, including lecturers, heads of department (HoDs), deans, and the Provost's Office.
- **Actor** – A user or system that interacts with the Syllabus Management System (e.g., Student, Lecturer, Dean).
- **Approve Syllabi** – Final confirmation performed by the Provost's Office for syllabi to be accepted into the official repository.
- **Compile** – The act of collecting and organizing multiple syllabi for institutional use or external requests.
- **Dean (of Faculty)** – Oversees all departments within their faculty and coordinates with HoDs to ensure timely and accurate syllabus submissions.
- **Head of Department (HoD)** – Academic authority responsible for reviewing syllabi submitted by lecturers and providing initial validation ("vouching").
- **Lecturer** – Creates and submits syllabi for assigned courses; interacts with the system to edit, track, and respond to reviews.
- **May** – Indicates an optional feature or action. For example, "The student may request a public link."
- **Must / Shall** – Indicates a required condition or feature; a mandatory element of the system.
- **Non-affiliated Individuals** – External entities such as university admission officers or partner institutions who may request syllabi via secure links.
- **Provost's Office** – Academic leadership responsible for ensuring quality assurance, syllabus compliance with accreditation, and final approval.
  Request Form – A submission from students or academic staff to request access to or share syllabi.
- **Role-Based Access Control (RBAC)** – A system model where permissions are assigned to users based on their role (e.g., student, dean, admin).
- **Syllabus** – A formal academic document outlining course details, learning outcomes, grading, and content structure.
- **Student** – End-user of the system, primarily interested in accessing or requesting syllabi related to their enrolled courses.
- **Vouch Syllabi** – The process by which an HoD confirms that a submitted syllabus is acceptable for review by a Dean.

Abbreviations and Acronyms

- **API** – Application Programming Interface
- **CS** – Computer Science
- **DB** – Database
- **HoD** – Head of Department
- **ICT** – Information and Communication Technology
- **IPO** – Input–Process–Output (model used to structure system functionality)
- **PDF** – Portable Document Format
- **PoC** – Proof of Concept
- **RBAC** – Role-Based Access Control
- **SRS** – Software Requirements Specification
- **UAT** – User Acceptance Testing
- **UI** – User Interface
- **UX** – User Experience
- **2FA** – Two-Factor Authentication

1.5 Document Conventions

- This document uses : Arial font at 12 point size with single line spacing and 1-inch margins on all sides.
- Heading and important information are formatted in bold for emphasis.
- Each SRS document is labeled with a distinct identifier or version number for tracking purposes.
- A revision history table is included to record any updates made to the SRS, noting the date of change , version number and a summary of the modifications.

1.6 Assumptions

The following assumptions form the basis for the requirements described in this document. These assumptions are external conditions that are beyond the direct control of the development team but are expected to hold true for the successful implementation and evaluation of the system:

- Authorized users(Ex: instructors,HoDs,deans) have valid institutional accounts and email access.
- Paragon University’s network allows on-campus and remote access to the system.
- Lecturers and staff use up-to-date web browsers on working computers.

- Email and 2FA systems are reliable for notifications and login.
- No integration with LMS,SIS or HR systems is required during this phase. It means the system runs independently.

2. General Design Constraints

2.1 Product Environment

The Syllabus Management System (SyllaBank) will be deployed and executed locally on a single developer or test machine, using localhost as its runtime environment. This setup is intended solely for demonstration, testing, and feedback purposes within a non-networked, standalone environment.

The system environment is defined as follows:

- Execution Platform: The web application will run locally on a computer (Windows/macOS/Linux) with sufficient resources. Users and testers will interact with the system through a web browser by accessing http://localhost.
- No Internet or LAN Requirement: The system operates independently of any external or institutional network. All services are self-contained within the host machine.
- Database: A local instance of PostgreSQL is used to store application data including user accounts, syllabi metadata, and submission status.
- File Storage: All syllabus files are saved on the local file system of the host machine. No cloud or shared drive is involved.
- Email Simulation: Email notifications (e.g., reminders, OTP) are simulated using Mailpit, which captures and displays emails locally for testing purposes.
- Authentication: User login and optional 2FA are handled entirely within the local instance. No external identity provider or mail server is involved.
- Access Control: The system enforces Role-Based Access Control (RBAC) for managing permissions, even within the local test context.
- Limitations: This environment does not support multi-user access, integration with institutional systems (e.g., LMS, SIS), or remote testing. It is intended for proof-of-concept validation only.

2.2 User characteristics

The Syllabus Management System (SyllaBank) will serve multiple categories of users, each with different roles, responsibilities, and levels of technical familiarity. Understanding these users helps guide decisions in user interface design, access control, feature visibility, and system feedback mechanisms.

A. Lecturers

- **Responsibilities**: Create, edit, and submit syllabi; respond to comments; update drafts as needed.
- **Domain Knowledge**: High – Lecturers have deep knowledge of the academic domain but may vary in their familiarity with digital tools.
- **Technical Background**: Moderate – Most lecturers are proficient in using standard productivity tools (Word, Excel, email), but their experience with form-driven platforms or custom web apps may vary.
- **Education**: University graduates; typically hold at least a Master's degree.
- **Priority**: **High** – As primary content contributors, the interface must prioritize ease of syllabus creation and clarity in feedback handling.

B. Heads of Department (HoDs)

- **Responsibilities**: Review and vouch for submitted syllabi, assign courses to lecturers, approve student requests.
- **Domain Knowledge**: Very high – Strong familiarity with academic policies, departmental structures, and curriculum standards.
- **Technical Background**: Moderate – Comfortable using web systems for approval, comments, and coordination.
- **Education**: Master's or doctoral level; administrative experience.
- **Priority**: **High** – Their review and coordination actions affect downstream approval processes and must be intuitive and efficient.

C. Dean

- **Responsibilities**: Final evaluation and acceptance of syllabi within their faculty; oversee consistency and compliance.
- **Domain Knowledge**: Very high – Responsible for academic integrity and inter-departmental alignment.
- **Technical Background**: Moderate to low – May have administrative support; interfaces should be clean, guided, and avoid unnecessary complexity.
- **Education**: Doctorate or senior academic standing.
- **Priority**: **Medium to High** – Their input is critical but less frequent than lecturers or HoDs.

D. Provost's Office

- **Responsibilities**: Approve syllabi for archival, handle external syllabi requests, oversee academic compliance and documentation.
- **Domain Knowledge**: Expert – Central academic authority with full knowledge of institutional and accreditation requirements.
- **Technical Background**: High – Typically works with internal systems, documents, and data platforms regularly.
- **Education**: Senior administrative staff; highly experienced.
- **Priority**: **Medium** – Needs advanced search, bulk export, and oversight tools, but less frequent day-to-day interaction.

E. Students

- **Responsibilities**: View, request, and download syllabi for completed courses; optionally request public links for external sharing.
- **Domain Knowledge**: Low to moderate – Familiar with course content, but not administrative procedures.
- **Technical Background**: High – Digital natives; comfortable with online forms, documents, and web portals.
- **Education**: Undergraduate level.
- **Priority**: **Medium** – Interface must be simple, self-explanatory, and focused on clarity and feedback.

2.3 Mandated Constraints

The following design and implementation choices are mandated by the academic requirements, development context, or institutional expectations, and must not be altered during this proof-of-concept phase:

- Development must occur locally on a single machine using localhost without reliance on institutional or external servers.
- The system must be developed using the following technologies:
    - Frontend: Next.js with Material UI (MUI)
    - Backend: Laravel (PHP framework)
    - Database: PostgreSQL
- Email Testing: Mailpit for capturing local outbound email
- No external integrations are permitted, including:
    - University authentication systems (SSO, LDAP, etc.)

    - Google Drive or existing syllabus storage platforms
    - Learning Management System (LMS) or Student Information System (SIS)
- Two-Factor Authentication (2FA) must be implemented using institutional email but simulated through Mailpit during testing.
- The system must use Role-Based Access Control (RBAC) to define user permissions.
- The product must be presented as a standalone proof of concept, and not positioned for production use or university-wide deployment at this time.

These constraints are mandatory and define the limits of flexibility available to the development team during the design, implementation, and evaluation of the system.

2.4 Potential System Evolution

The Syllabus Management System (SyllaBank) is designed as a **proof of concept**, but its modular architecture supports future growth. Although the current implementation is limited to a local environment with a controlled set of features, several potential areas for system evolution have been identified to guide long-term planning and future development.

A. University-Wide Deployment

- Transitioning from a local prototype to a hosted production environment accessible across the university.
- Supporting secure multi-user access via Paragon's internal network or VPN.

B. Integration with Existing Systems

- Integration with Paragon's **Learning Management System (LMS)** and **Student Information System (SIS)** for automatic syncing of course, lecturer, and student data.
- Linking to **Google Drive or institutional cloud storage** for backup or archival purposes.

C. Public Link and Verification Enhancements

- Extending the public syllabus sharing mechanism with **expiring, trackable links** and optional **PDF watermarks**.
- Adding a **verification and authentication layer** for non-affiliated institutions accessing shared syllabi.

D. Richer Feedback and Evaluation Features

- Adding **collaborative editing** and **comment threads** for syllabus drafts.
- Expanding the review process with **rubrics** or **checklists** for academic standards compliance.

E. Mobile Accessibility

- Developing responsive views or a dedicated mobile app for lecturers and students to access and manage syllabi on smartphones or tablets.

F. Automation and Notification Improvements

- More advanced email logic (e.g., escalating reminders, report summaries).
- Scheduled reports or alerts for missing or outdated syllabi.

G. Enhanced Role Management

- Support for **multi-role users** (e.g., lecturer + HoD) with flexible permission resolution.
- Administrative tools to manage role assignments and user activity logs.

H. Analytics and Reporting

- Dashboards for administrators showing syllabus submission rates, delays, and departmental compliance statistics.

3. Nonfunctional Requirements

Nonfunctional requirements define system-wide qualities and constraints that are not specific to individual features but are essential for the overall performance, usability, and maintainability of the Syllabus Management System (SyllaBank). These requirements ensure that the system not only functions correctly but also meets expectations for reliability, user experience, and future adaptability.

A. Usability

- The system shall provide an intuitive user interface suitable for academic staff with varying levels of technical expertise.
- Form fields, action buttons, and navigation components must follow a consistent layout and design language using Material UI.
  Tooltips, inline validation, and error messaging should guide users through the submission process with minimal confusion or training.

B. Reliability

- The system shall maintain operational consistency during testing periods, with an uptime goal of 95% within local execution conditions.
- In case of failure (e.g., unexpected shutdown), no data loss shall occur for submitted syllabi, user sessions, or role assignments.

C. Performance

- Page responses for typical actions (e.g., form submission, syllabus loading, role switching) shall complete within 2 seconds under local conditions.

- The system shall maintain consistent performance on machines with standard development specifications (e.g., 8 GB RAM, local PostgreSQL server).

D. Maintainability

- The system architecture shall follow a modular design pattern, separating concerns between frontend (Next.js), backend (Laravel), and data storage (PostgreSQL).
- All code should be readable, documented, and follow naming conventions that support future debugging, extension, and code review.

E. Security

- Role-Based Access Control (RBAC) must enforce permissions at every endpoint and UI level to prevent unauthorized access or role escalation.
  Passwords shall be securely hashed using Laravel's built-in hashing before being stored.
- Two-Factor Authentication (2FA) shall be implemented using simulated email OTP via Mailpit, ensuring basic verification in test environments.

F. Portability

- The system shall be operable across major operating systems (Windows, macOS, Linux) with minimal setup.
- All components must run on a local machine without requiring internet access or remote services.

G. Availability

- Although deployed in a local environment, the system shall be available for testing for at least 8 hours per working day during the evaluation period.

Any maintenance or restart procedures shall not exceed 15 minutes per incident and must not occur more than twice per week during active use.

H. Testability

- The system shall support functional and user acceptance testing (UAT) with clear input/output scenarios and form validation rules.
- All email functions shall be testable using Mailpit, and the system shall log actions (e.g., submissions, approvals) for test review.

3.1 Usability Requirements

Usability is a key focus of the SyllaBank proof-of-concept, as the system aims to streamline workflows and reduce friction for academic staff involved in syllabus management. The interface should allow all key actions to be performed efficiently, with minimal steps and no need for training or support. The following requirements guide the system's design:

A. Efficiency

- The system interface allows a lecturer to complete and submit a syllabus form in 10 minutes or less, assuming the course content is ready.
- A Head of Department (HoD) can review and vouch for a submitted syllabus in under 5 minutes.
- Key interface pages load within 2 seconds on standard local development hardware.
- Users should not need more than 5 navigation steps to reach or complete common actions.
- Form input errors preserve previous data, allowing corrections and resubmission within 2 minutes.

B. Effectiveness

- Form submissions are validated to ensure all required fields are completed and follow the updated academic format.
- Each user role sees only relevant content and actions based on role-based access permissions.
- The system interface maintains a consistent layout, form structure, and label naming to reduce confusion.

C. Satisfaction

- The system uses familiar patterns and Material UI design to ensure ease of use without requiring formal training.
- Clear confirmation messages, validation feedback, and tooltips guide users through all primary tasks.

D. Accessibility

- UI elements follow contrast and legibility standards as provided by the Material UI framework.
- Color is never the sole indicator for important states; icons or text are always included.

3.2 Operational Requirements

Operational requirements define the conditions under which the SyllaBank system is expected to function. These include runtime conditions, environment dependencies, maintenance expectations, and routine procedures necessary for successful operation of the system during its proof-of-concept phase.

A. Deployment Environment

- The system shall operate on a **local development machine** using `localhost` without requiring internet access.
- Supported operating systems include Windows, macOS, and Linux, provided they meet the minimum hardware and software requirements (e.g., Node.js, PHP, PostgreSQL).
- All services (frontend, backend, database, and email simulator) must run concurrently on the same machine without external dependencies.

B. Startup and Shutdown

- The system shall be startable using a defined sequence of commands or scripts (e.g., `npm run dev`, `php artisan serve`, `postgres start`) documented in the setup guide.
- System shutdown shall gracefully terminate all services, ensuring data integrity and session cleanup.

C. Maintenance

- Routine maintenance such as resetting the database, clearing test entries, or updating configuration files must be possible through simple administrative actions or scripts.
- The system shall include instructions for manual data backup (e.g., exporting `.sql` dumps or local file storage) to prevent loss during reinstallation of updates.

D. Error Handling

- In case of application crashes or service failure, the system shall log errors locally in designated log files (`/logs` or Laravel's `storage/logs`) to assist with debugging.
- UI-level validation and error messages shall guide users to resolve common input or permission issues without technical support.

E. Usage Limit

- The system is intended for **single-user or small-group testing only**. Concurrent access by multiple users is not supported or guaranteed during this phase.
- Operational sessions are limited to **internal testing hours** and are not expected to exceed 8 hours of continuous use.

F. Data Handling

- All syllabus files, user data, and configurations are stored locally and not synchronized with any cloud or external service.
- File access permissions shall be restricted to the local machine and the currently logged-in developer or tester.

3.3 Performance Requirements

- Response Time : The system responds to user actions and queries within an acceptable time frame.For instance : upload syllabus , edit syllabus have one or two seconds to enable uploading and editing.

- Loading Time : The application should load less than 1 second when launching or switching between screens.

3.4 Security Requirements

Although the SyllaBank system is currently developed and deployed in a controlled local environment, basic security requirements must still be upheld to ensure proper handling of academic data, user credentials, and access privileges. The following security requirements reflect the role-based structure, authentication process, and local-only operation defined in earlier sections of this document.

A. Role-Based Access Control (RBAC)

- The system shall enforce strict RBAC policies to restrict access to features and data based on the user's assigned role (e.g., Lecturer, HoD, Dean, Provost).
- Users shall only be able to view or perform actions relevant to their role. For example:
    - Lecturers may only edit their own syllabi.
    - HoDs may only review and vouch for syllabi in their department.
    - Provost staff may approve but not edit submitted syllabi.

B. Authentication

- All users must log in using institution-issued email addresses and passwords.
- User credentials shall be stored securely using Laravel's built-in hashing system, and shall never be stored in plain text.

- Failed login attempts shall result in appropriate error feedback, but must not reveal whether a specific email exists in the system.

C. Two-Factor Authentication (2FA)

- A simulated 2FA mechanism shall be implemented using OTP codes sent via Mailpit.
- OTP verification shall be required during login for roles with elevated privileges (e.g., Dean, Provost).
- OTP codes shall expire after a defined time window and must be randomly generated per session.

D. Data Protection

- All user data and syllabus files shall be stored locally on the development machine and protected by the operating system's file permission system. Access to the local database and storage directories shall be restricted to authorized users of the host machine only.

E. Session Management

- The system shall enforce session expiration after a predefined period of inactivity (e.g., 30 minutes).
  Users must re-authenticate after session timeout or manual logout.

F. Input Validation

- All user inputs must be validated both client-side and server-side to prevent injection attacks, malformed data, or unauthorized submissions.
- Uploaded files must be restricted to allowed types (e.g., PDF) and scanned for potential threats if implemented in a production phase.

G. Logging and Auditing (Local Only)

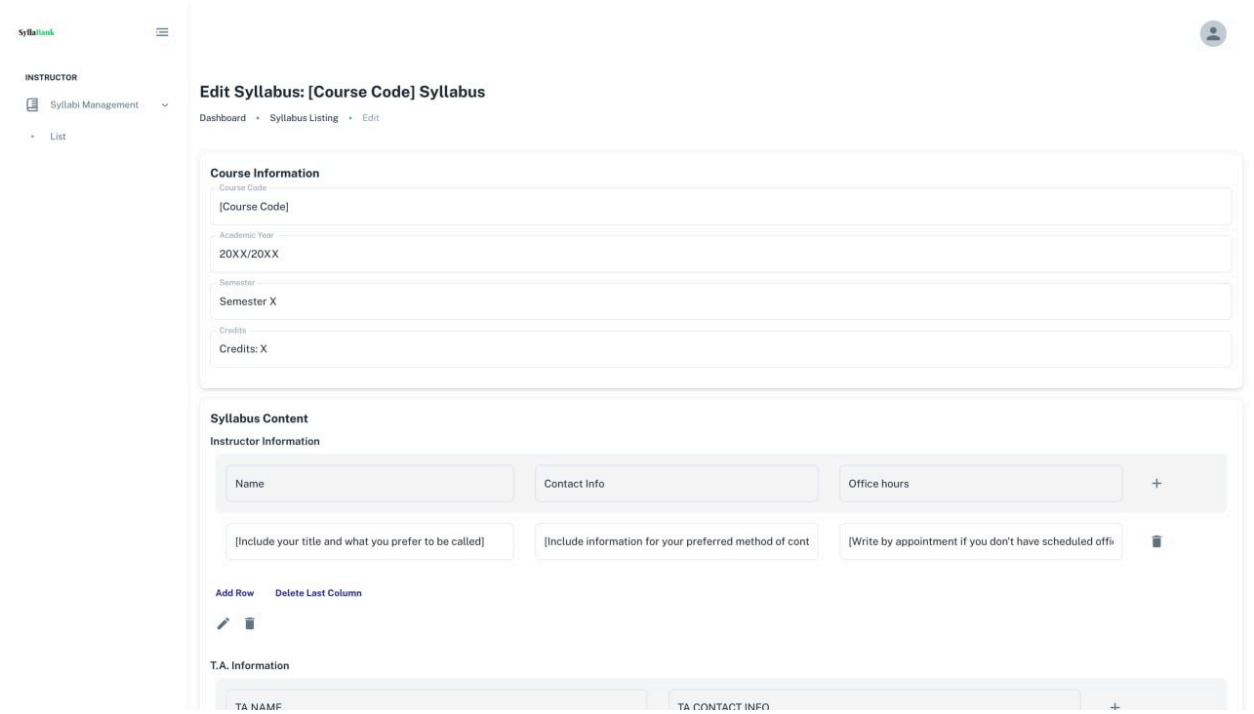
- All login events, role-switching actions, and submission approvals shall be logged locally for audit and debugging purposes.
- Access to logs shall be restricted to administrative developers during the PoC phase.

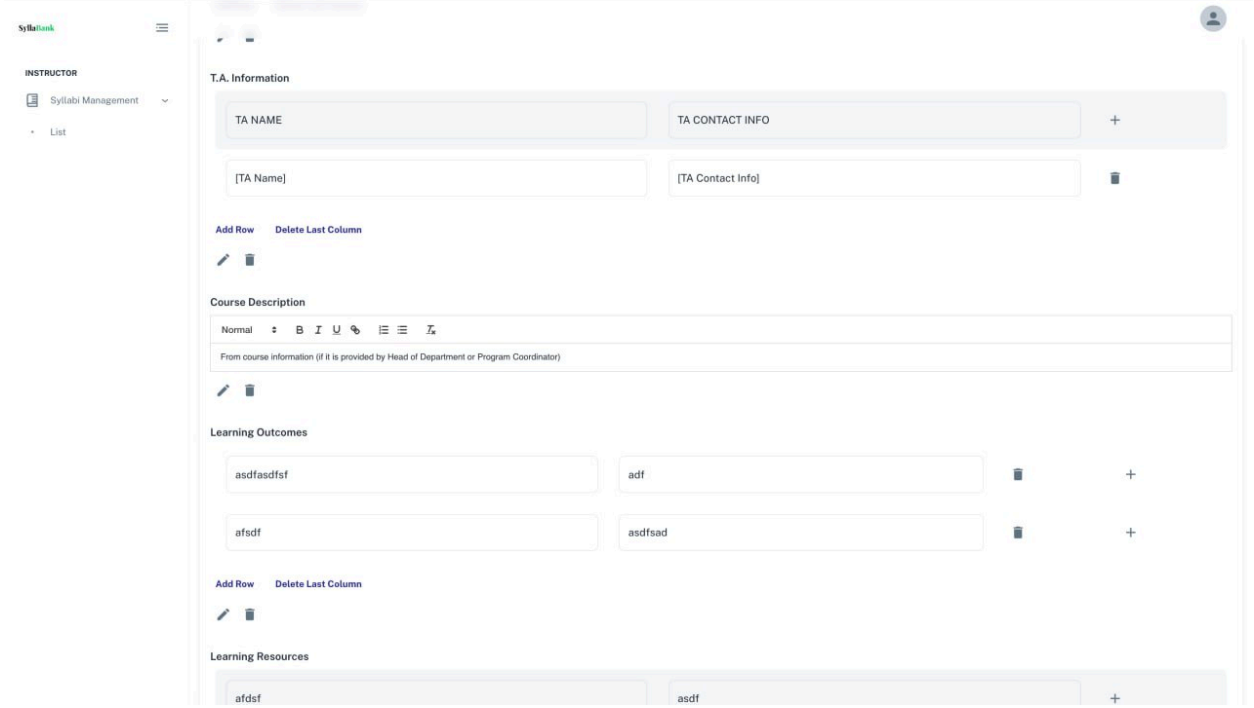
H. No External Exposure

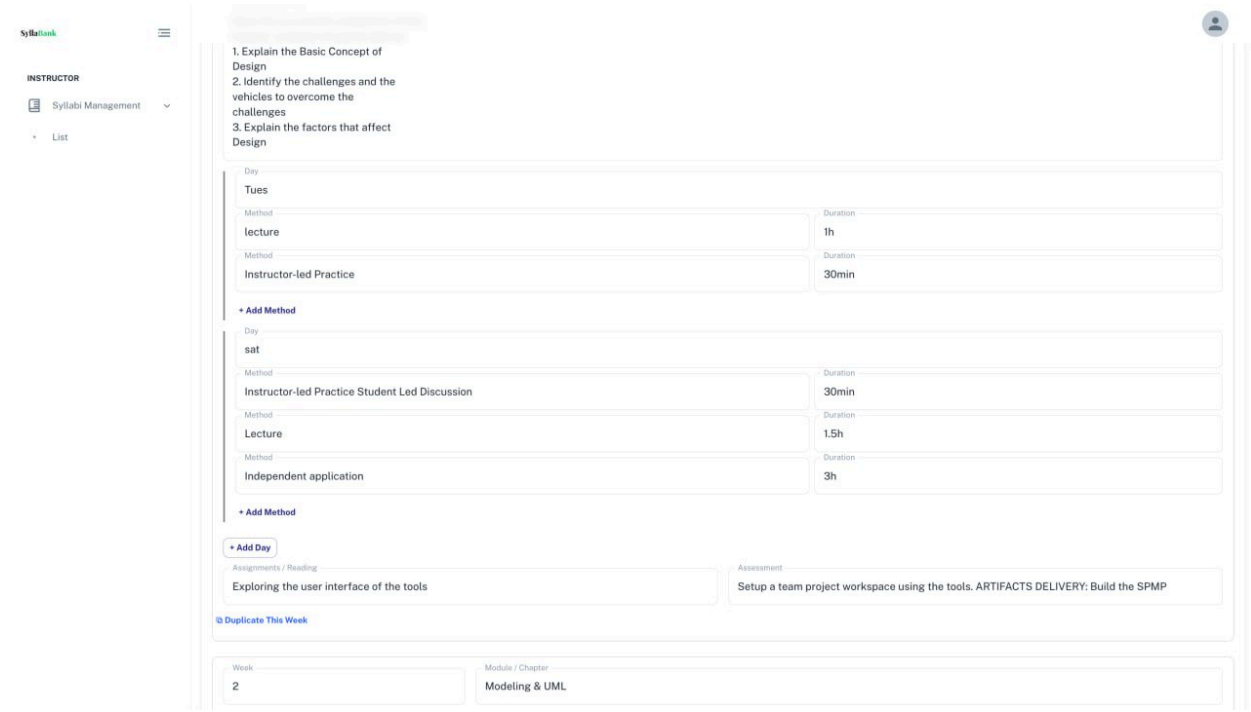
- The system shall **not** expose any API or interface to the public internet.
- Access is limited to localhost and the current test environment, preventing remote attacks or data leakage.

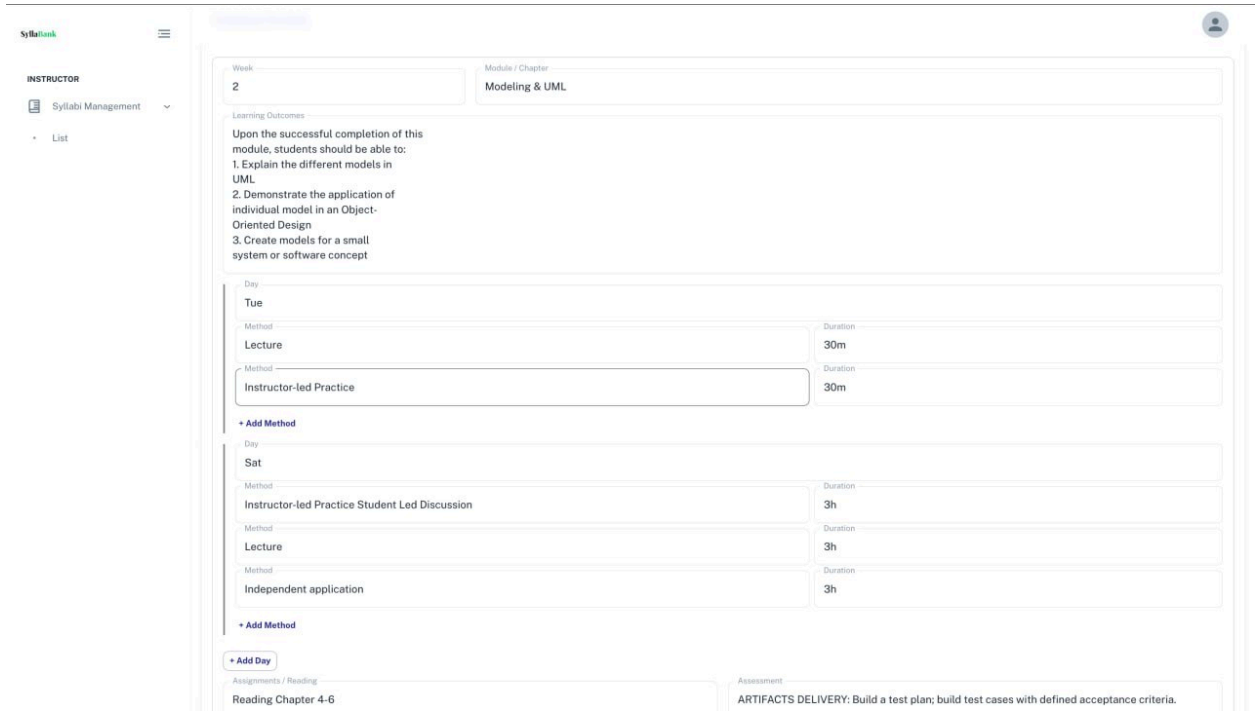
3.5 User Interface

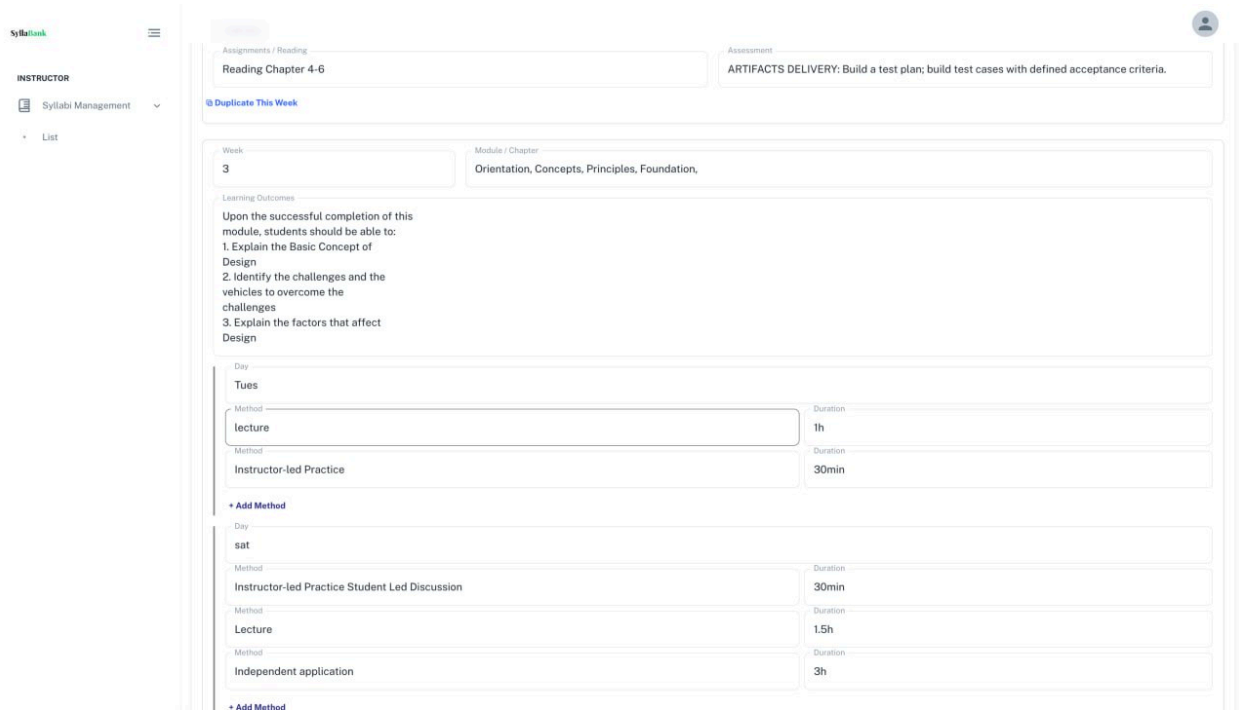
The user interface should be a simple and intuitive interface that allows the user to easily navigate the system, in addition, the interface should contain white background and the font should be good for the UI for the system. The figures below show the process of the improved edit syllabus function to support the next academic year.


SyllaBank
INSTRUCTOR
Syllabi Management
List
Edit Syllabus: [Course Code] Syllabus
Dashboard • Syllabus Listing • Edit
Course Information
Course Code
[Course Code]
Academic Year
20XX/20XX
Semester
Semester X
Credits
Credits: X
Syllabus Content
Instructor Information
Name
Contact Info
Office hours
[Include your title and what you prefer to be called]
[Include information for your preferred method of cont
[Write by appointment if you don't have scheduled offi
Add Row
Delete Last Column
T.A. Information
TA NAME
TA CONTACT INFO


SyllaBank
INSTRUCTOR
Syllabi Management
List
T.A. Information
TA NAME
TA CONTACT INFO
[TA Name]
[TA Contact Info]
Add Row
Delete Last Column
Course Description
Normal
From course information (if it is provided by Head of Department or Program Coordinator)
Learning Outcomes
asdfasdfsf
adf
afsdf
asdfsad
Add Row
Delete Last Column
Learning Resources
afdsf
asdf


SyllaBank
INSTRUCTOR
Syllabi Management
List
1. Explain the Basic Concept of Design
2. Identify the challenges and the vehicles to overcome the challenges
3. Explain the factors that affect Design
Day
Tues
Method
lecture
Duration
1h
Method
Instructor-led Practice
Duration
30min
+ Add Method
Day
sat
Method
Instructor-led Practice Student Led Discussion
Duration
30min
Method
Lecture
Duration
1.5h
Method
Independent application
Duration
3h
+ Add Method
+ Add Day
Assignments / Reading
Exploring the user interface of the tools
Assessment
Setup a team project workspace using the tools. ARTIFACTS DELIVERY: Build the SPMP
Duplicate This Week
Week
2
Module / Chapter
Modeling & UML


SyllaBank
INSTRUCTOR
Syllabi Management
List
Week
2
Module / Chapter
Modeling & UML
Learning Outcomes
Upon the successful completion of this module, students should be able to:
1. Explain the different models in UML
2. Demonstrate the application of individual model in an Object-Oriented Design
3. Create models for a small system or software concept
Day
Tue
Method
Lecture
Duration
30m
Method
Instructor-led Practice
Duration
30m
+ Add Method
Day
Sat
Method
Instructor-led Practice Student Led Discussion
Duration
3h
Method
Lecture
Duration
3h
Method
Independent application
Duration
3h
+ Add Method
+ Add Day
Assignments / Reading
Reading Chapter 4-6
Assessment
ARTIFACTS DELIVERY: Build a test plan; build test cases with defined acceptance criteria.



3.6 Software Interface

The Syllabank system is developed using a Next.js frontend and a Laravel backend, which communicate via RESTful APIs. All data exchanges are in JSON format, and authentication is handled via Laravel Sanctum.

4. System Features

4.1 Enhanced Course Schedule Input

4.1.1 Description and Priority

This feature transforms the traditional flat course schedule input into a flexible and deeply structured 3-tier system: **Week** → **Day** → **Hour**. Each hourly block supports granular input fields including:

- Topic
- Delivery Method
- Assignments
- Assessments
- Learning Outcomes

This allows instructors to create syllabi that align precisely with institutional standards for instructional planning, enabling richer curriculum detail and improved policy compliance.

Priority: High

4.1.2 Use Case: Input Detailed Weekly Schedule

- **Actor**: Instructor
- **Precondition**: User is authenticated and authorized to create/edit a syllabus
- **Flow**:
    1. Create or edit “Syllabi”
    2. Scroll down to "Course Schedule" section
    3. For each week,input specify instructional days
    4. For each day, define instructional hours
    5. For each hour, fill out topic, activity, assignment, assessment, Learning Outcome
    6. Save as draft or submit

Postcondition: The course schedule is saved in a fully structured, queryable format supporting administrative validation, rendering, and export.

4.2 Dynamic Section Formatting

4.2.1 Description and Priority

Instructors can now toggle between **paragraph mode** and **table mode** for key syllabus sections. This enables flexibility in how content is authored and presented, depending on the nature of the information or the preference of the instructor. Table mode supports adding/removing rows and columns dynamically, making the interface more adaptable for structured input.

Priority: High

4.2.2 Affected Sections:

- Instructor Info
- TA Info
- Course Description
- Course Objectives
- Learning Outcomes
- Learning Resources

- Assessment
- Course Policies
- Course Schedule

4.2.3 Use Case: Switch Format

- **Actor**: Instructor
- **Precondition**: Editing an existing syllabus or creating a new one
- **Flow**:
    1. Create or edit “Syllabi”
    2. Scroll down to section
    3. Select “edit” or a pen symbol
    4. Allow user to select between table and paragraph
    5. Table appears with option to add/remove rows/columns
    6. Inputs data in cell-based editor

Postcondition: Content is saved in the selected format and rendered accordingly in both preview and published views.

4.3 Enhanced Learning Outcome Input

4.3.1 Description and Priority

This feature upgrades the "Learning Outcomes" section to include **explicit linkage between Learning Outcomes** and **Program Learning Outcomes (PLOs)**. It provides a structured table-based interface where CLOs can be mapped to one or more PLOs, supporting curriculum alignment, accreditation preparation, and institutional reporting.

Priority: High

4.3.2 Use Case: Add Learning Outcome with PLO

- **Actor**: Instructor
- **Precondition**: Editing an existing syllabus or creating a new one
- **Flow**:
    1. Scroll to “Learning Outcomes” section
    2. Select “edit” or a pen symbol on Learning Outcome section
    3. Allow Instructor to Enters one or more PLO description in the table

Postcondition: Outcome are saved with mapped PLO references, enhancing the syllabus for institutional use and outcome-based education audits.

4.4 Feature: Enhanced Syllabus Preview Renderer

4.4.1 Description and Priority

This feature enhances the instructor and reviewer experience by rendering the entire syllabus including complex nested schedule data in a visually organized, print-friendly layout. It reflects paragraph/table formatting, includes PLO mappings, and renders weekly schedules with full structural fidelity.

- **Priority:** High

4.4.2 Use Case: Preview Formatted Syllabus

Actor: Lecturer
Precondition: Logged-in user with Lecturer role
Basic Flow:

1. Click “Preview”
2. System renders full syllabus with detailed schedule
3. Allow print/export