IT Group Project

STI Time Table Management System

one page

By Yan Myo Aung, Moe Htet Yan, Khine Khant Soe, Htoo EntLwin

Introduction

Background

In today's academic environment, the capacity to properly plan, organize, and manage teaching schedules is essential for providing high-quality education. Educational institutions, particularly those with complicated multi-departmental systems, are finding it increasingly difficult to manage timetables that meet the needs of administrators, lecturers, and students all at once. Originally, a lot of schools have continued to use manual techniques, like spreadsheet trackers, paper-based timetables, or simple scheduling tools, which are frequently isolated and don't integrate with other academic systems. These methods lead to separated data storage, substantial administrative overhead, and difficulty implementing real-time changes when last-minute adjustments arise. In order to solve these inefficiencies, the STI Timetable Management System (STI-TTMS) was developed, providing a centralized, web-based platform that can manage the entire timetable creation and distribution lifecycle. In addition to intelligent conflict detection, responsive design, and real-time notifications, the system integrates role-based access for administrators, lecturers, and students to streamline scheduling procedures. Academic coordinators don't have to keep difficult manual records when cross-referencing important parameters like lecturer availability, room capacity, and course requirements thanks to STI-TTMS. Lessons are less likely to be missed or confused by out-of-date schedules when lecturers can access their most recent schedules from any device and students receive timely updates on their classes. STI-TTMS reduces scheduling conflicts, increases transparency, and boosts overall academic efficiency by combining scheduling tasks onto a single platform. In addition to easing the administrative staff's workload, this guarantees that learning activities are more effectively organized and shared with all parties involved.

Aim and Objective

The STI Timetable Management System (STI-TTMS) project's main goal is to create a web-based platform that is safe, effective, and easy to use for making, managing, and distributing academic schedules in educational institutions. With the help of an integrated digital solution that facilitates real-time updates, conflict detection, and multi-role access, this system aims to replace antiquated manual and disjointed scheduling procedures.

The following particular goals are defined by the project in order to accomplish this goal:

- To ensure accessibility for administrators, instructors, and students, design and develop a responsive web application that can adjust to desktop, tablet, and mobile devices with a consistent layout across platforms.
- To manage access to pertinent system features and safeguard private information, use secure role-based authentication for various user types (administrator, lecturer, and student).

- Give an administrator interface that includes tools for managing academic levels, time slots, venues, lecturers, students, and courses. It should also be able to create timetables without conflicts in less than five seconds.
- Give instructors the ability to quickly search for classes, view and create customized timetables, and verify assigned courses—all in less than three seconds.
- With a data retrieval time of less than two seconds, let students view and update their schedules based on their level, semester, and academic session.
- Include automated validation procedures to avoid invalid time ranges, duplicate course codes, and overlapping class times.
- Maintain a 99.9% uptime rate and responsive design for any device compatibility to ensure high system reliability.
- To guarantee data integrity and protection against data loss, put in place a secure database structure with frequent backups.
- Make it simple to export timetables with clear, user-friendly formatting in structured formats like Excel and PDF.

Method of solution and Justification

STI-TTMS is a web-based multi-role scheduling platform that was created using responsive design principles, secure authentication protocols, and contemporary web technologies to address the inefficiencies in the manual timetable management processes that are currently in use.

Technology Stack and Architecture

Frontend

A JavaScript framework like React.js or Vue.js and responsive HTML5/CSS3 used to create the frontend. This will guarantee a modular, reusable, and maintainable architecture that is based on components. The user interface will adhere to responsive design guidelines to guarantee accessibility on all screens, including mobile ones and large desktop monitors.

Backend

The robust PHP web framework, which was selected for its MVC architecture, integrated routing, validation, and security features, will be used to implement the backend. User authentication, timetable generation logic, and database interactions will all be managed by the backend.

Database

In a relational database MySQL will be used to store structured data, such as user information, courses, venues, time slots, and timetable records. Tables will be normalized to keep data clean, and indexes will be used to improve search and access efficiency.

Authentication and Security

Role-based access control will be used by the system to guarantee that only authorized features are accessible by administrators, instructors, and students. We will use session-based authentication to maintain secure login sessions. We'll use prepared statements, encryption, and input validation to guard against vulnerabilities like SQL injection and cross-site scripting (XSS).

Justification for Chosen Approach

- Efficiency: Automating scheduling lowers the possibility of human error and eliminates tedious manual tasks.
- Scalability: New features, like automated room assignment or integration with learning management systems, can be easily added thanks to the modular architecture.
- Security: Encryption and role-based access control safeguard private academic information.
- User Experience: Administrators, instructors, and students can all easily access the system from any device thanks to responsive, user-friendly interfaces.
- Maintainability: Long-term maintainability and the availability of resources for future development are guaranteed when widely used frameworks with robust community support are used.

STI-TTMS will offer a dependable, scalable, and user-centric solution that simplifies academic scheduling and improves communication between educational institutions by combining these technical and functional factors.

Analysis

This chapter presents the analysis and specifications of the proposed system, including the detailed functional and non-functional requirements, a prioritization matrix using the MoSCoW method, and high-level system models such as use case diagrams and class diagrams.

1) Requirements Specification

i. Functional Requirements

Functional requirements define the specific behavior and functionality that a system must support to meet user requirements and institutional objectives. As AltexSoft Editorial Team (2023) states, *“Functional requirements are product features that developers must implement to meet the user requirements under specific conditions.”*

For the web-based school timetable management system, functional requirements are categorized into three key user groups: administrators, lecturers, and students.

Student-Side Functional Requirements

For students who interact with the timetable system's front-end portal, the requirements include:

- Registering for an account and logging in
- Viewing their personal timetable and course schedule
- Viewing class details such as subject name, lecturer, and location
- Searching and filtering timetable entries by day, week, or course
- Downloading or printing timetable schedules
- Receiving updates on timetable changes or announcements
- Viewing examination timetables
- Updating personal contact information

These functions ensure that students have clear, accessible, and up-to-date timetable information to support their academic activities.

Lecturer-Side Functional Requirements

For lecturers who manage or view their teaching schedules, the requirements include:

- Logging in securely to the lecturer portal
- Viewing assigned teaching timetables and lecture slots
- Viewing class details, including enrolled students, subjects, and locations
- Requesting timetable changes or swaps with other lecturers
- Adding class-related announcements or notes for students
- Viewing examination invigilation schedules

These functions enable lecturers to manage their teaching responsibilities effectively and stay informed of any schedule updates.

Admin-Side Functional Requirements

For administrators who manage the entire timetable system via an admin dashboard, the requirements include:

- Creating and managing user accounts for students and lecturers
- Creating, updating, editing, or deleting timetable entries
- Assigning subjects, rooms, and lecturers to time slots
- Managing academic calendars, including holidays and exam dates
- Approving or rejecting timetable change requests from lecturers
- Sending announcements and notifications to students and lecturers
- Generating and exporting timetable reports in various formats

These functions ensure that timetable data is managed centrally, accurately, and efficiently.

ii. MoSCoW Prioritization Technique

The MoSCoW method is a prioritization framework used in project management and software development to categorize and rank requirements or features (Strategic Roadmaps, n.d.). It consists of four prioritization categories – Must Have, Should Have, Could Have, and Won't Have (Andreev, n.d.):

- Must Have**:** Essential features without which the system cannot function
- Should Have**:** Important features that enhance usability but are not critical for core operation
- Could Have**:** Desirable features that improve the system's value but are not essential
- Won't Have**:** Features agreed to be out of scope for the current development cycle

This prioritization technique ensures that the most critical functions are delivered first within the development life cycle. For the school timetable project, the functional requirements for admins, lecturers, and students will be categorized using the MoSCoW technique as shown in the next section.

No.	Functional Requirements	Moscow Prioritization
1	Admin can manage academic calendar from dashboard	Must have
2	Admin can create, update, edit, or delete timetable entries	Must have
3	Admin can assign subjects, rooms, and lecturers to specific time slots	Must have
4	Admin can manage user accounts (students and lecturers)	Must have
5	Admin can approve or reject timetable change requests from lecturers	Should have
6	Admin can send bulk notifications and announcements to users	Should have
7	Admin can send bulk notifications and announcements to users	Should have
8	Admin can export timetable data in PDF or Excel formats	Should have
9	Admin can use automated timetable generation based on constraints	Could have
10	Admin can use role-based dashboard analytics	Could have
11	Admin can use AI-driven timetable optimization (planned for future phase)	Won’t have
12	Admin can use multi-school integration support	Won’t have
13	Lecturer can secure login to lecturer portal	Must have
14	Lecturer can view assigned teaching timetable	Must have
15	Lecturer can view class details (subject, location, enrolled students	Must have
16	Lecturer can view examination invigilation schedules	Should have
17	Lecturer can sync timetable with personal calendar apps (Google Calendar, Outlook)	Should have
18	Lecturer can sync timetable with personally	Could have
19	Lecturer can direct timetable editing without	Won’t have

	admin approval	
20	Student can secure login to student portal	Must have
21	Student can view personal timetable	Must have
22	Student can Download or print timetable	Should have
23	Student can receive announcements about class or schedule changes	Should have
24	Student can option to customize timetable view (themes, color codes)	Could have
25	Student can have timetable reminders via email or mobile notifications	Could have
26	Student can direct timetable editing	Won't have

Moscow Prioritization

iii. Non-Functional Requirements

Non-functional requirements describe **how** the system should perform rather than **what** it should do. These characteristics such as performance, usability, reliability, and maintainability are essential to delivering a high-quality, dependable timetable management system KVY Technologyprojectsinventory.com.

Below is a structured list of non-functional requirements tailored to the school timetable system:

1. **Performance**
    - The system should respond to user interactions (e.g., viewing or searching timetables) within **2 seconds** under normal load projectsinventory.com.
2. **Scalability**
    - The system must accommodate increasing numbers of users, timetables, and concurrent requests without performance degradation, supporting both vertical and horizontal scaling strategies projectsinventory.comtpptechnology.com.
3. **Reliability & Availability**

- The system should achieve **99.9 % uptime** during school terms, featuring robust error recovery, redundancy, and backup mechanisms to ensure continuous service projectsinventory.com+1.

4. **Security**
    - Sensitive data (e.g., personal info, credentials) must be encrypted both in storage and transit, with secure authentication and role-based authorization enforced projectsinventory.com+1.
    - Compliance with relevant educational privacy regulations should be maintained projectsinventory.com.
5. **Usability & Accessibility**
    - Interfaces must be intuitive and easy to navigate for administrators, lecturers, and students, minimizing training requirements.
    - The system should comply with **WCAG accessibility standards**, ensuring usability for individuals with disabilities projectsinventory.comtpptechnology.com.
6. **Maintainability**
    - The system should follow a modular design with clear documentation to support future updates, bug fixes, and enhancements projectsinventory.com.
7. **Compatibility & Interoperability**
    - The application should operate seamlessly across modern web browsers and devices (desktop, tablet, mobile) and support integration with systems like LMS via standard data formats (e.g., CSV, XML) projectsinventory.com+1.
8. **Data Integrity & Accuracy**
    - The system must ensure consistent and accurate timetable data by employing validation checks, transaction integrity, and appropriate error-handling projectsinventory.com.
9. **Compliance & Auditability**
    - The system must comply with institutional and legal requirements, including data retention policies. Activity logs and audit trails should be maintained for accountability and transparency.
    -

Implementation

Introduction

This chapter describes the implementation of the Timetable Management System (STI-TTMS), detailing the technical choices and execution steps taken after the design phase. It focuses on the selection of programming languages and frameworks, the system's architecture, data migration strategies from a manual system, and the training plans developed for end-users.

Choice of Programming Language

The project uses a modern full-stack development approach, incorporating a combination of client-side and server-side technologies to ensure a secure, responsive, and scalable web application.

- **Frontend:**
  - **React.js** serves as the core framework for building dynamic user interfaces for the administrator, lecturer, and student dashboards.
  - **TypeScript** was chosen to enhance code quality by providing type safety, which helps to reduce common errors and improve the maintainability of the large codebase.
  - **Tailwind CSS**, a utility-first CSS framework, was used to rapidly build a consistent and responsive interface, ensuring the application looks and functions well on all device types.
  - **Shadcn/ui** provides a set of accessible UI components that were customized to improve user experience and accelerate development time for elements like forms, tables, and buttons.
- **Backend:**
  - The backend is implemented using the **Laravel** framework, which acts as the application's core engine, handling all business logic, database operations, and authentication.
  - **PHP Laravel** was used to build a set of RESTful API endpoints that the React frontend communicates with. These APIs handle crucial functions such as user authentication, fetching timetable data, and performing CRUD operations for courses, lecturers, and venues.
  - **JWT (JSON Web Tokens)** are used to secure user sessions. The Laravel backend generates a JWT upon successful login, which the React frontend stores. This token is then included in subsequent API requests for secure, role-based authorization.
  - The backend also handles core operations like validating user input, managing authentication, and facilitating all CRUD (Create, Read, Update, Delete) operations for the system's entities.
  - **Laravel Blade** was used for some core administrative pages that required server-side rendering, such as the initial login pages and certain static administrative reports.

- **Database:**
    - **MySQL** provides a reliable relational database structure. All tables were normalized to ensure data consistency and reduce redundancy, which is critical for a system handling a large volume of interconnected data (e.g., students, courses, lecturers, and time slots).
- **Security:**
    - **JWT** was a fundamental component of the secure authentication system.
    - An **OTP (One-Time Password)** feature was designed to be sent to a user's registered email address, providing a secure mechanism for password recovery.

These technologies were integrated to implement a robust and efficient Timetable Management System that addresses the project's functional and non-functional requirements.

System Cutover from Development to Implementation Architecture

The STI-TTMS was developed in a local environment and was transitioned to a live deployment architecture through a structured cutover process.

- **Local Development Environment:**
    - The local development environment was set up with **MySQL hosted locally via XAMPP**.
    - The frontend, built with **React.js**, ran on a local development server.
    - The backend, using **Laravel**, ran on its built-in PHP server.
    - **VS Code** was used for code editing, and **Postman** was used to test the API endpoints.
    - **GitHub** was used for version control, managing commits and branches.
- **Implementation (Live) Architecture:**
    - The project is planned for deployment to a live environment after local testing is complete.
    - The frontend is planned to be hosted on a service like **Vercel**, enabling automatic deployment from GitHub.
    - The backend is planned to be hosted on a shared hosting or VPS platform, configured with Laravel environment settings to connect to the live MySQL database and mail server.
    - The database connection credentials in the Laravel `.env` file are adjusted to production settings.

This transition is planned to ensure a smooth connection between the frontend and backend, with domain routing and HTTPS (via an SSL certificate) planned for secure communication.

Data Migration

To prepare the STI-TTMS for its go-live date, a data migration process was performed. Unlike moving data between two digital systems, this process involved transitioning from antiquated, manual records to a structured digital format. The previous system relied on handwritten notes, spreadsheets, and various fragmented documents. The data migration process for STI-TTMS involved:

- **Data Preparation:** Consolidating all existing manual records for courses, lecturers, venues, and student enrollment into a single source, such as a master Excel spreadsheet.
- **Admin Account Setup:** Creating the initial administrator account to allow the business owner or key administrative staff to access the system's dashboard.
- **Manual Data Entry:** The first set of data, including all foundational information (courses, lecturers, venues, departments), was manually entered into the system using the administration forms.
- **Database Initialization:** The Laravel backend was used to generate the database schema using migration commands, preparing the database for the newly entered data.
- **File Organization:** Old, unstructured records were archived, as the new system would centralize all information digitally.

This process ensured that the new system had all the essential data required to operate effectively from day one.

Training

The STI-TTMS was designed with a user-friendly interface to minimize the need for extensive training for students and lecturers. However, targeted training was required for administrative users who would be transitioning from manual scheduling to the new dashboard. The training plan was proposed as follows:

- **Admin Training:**
    - **Topics covered:**
        - Securely logging in and navigating the dashboard.
        - Creating, editing, and deleting courses, lecturers, students, and venues.
        - Using the timetable generation tool to create new schedules.
        - Updating and managing user accounts and roles.
    - **Materials:** A comprehensive PDF user manual with step-by-step instructions and screenshots was prepared.
    - **Sessions:** A two-day training schedule was proposed, including a theoretical presentation, a live system demonstration, and a hands-on session where trainees could practice using the system.
    - **Support:** A hyper-care period of three months was planned post-deployment to provide support and address any questions.
- **Lecturer and Student Training:**

- A simple PDF guide and a short video tutorial were created to help lecturers and students easily navigate their respective dashboards and view their timetables. The intuitive design of these interfaces made extensive training unnecessary.

Summary

The implementation of the STI-TTMS successfully transitioned the institution's scheduling from a manual to a digital platform. The strategic choice of a modern full-stack architecture, a structured cutover process, a plan for data migration, and a targeted training program for administrators were all crucial to ensuring a smooth adoption and a successful outcome for the project.

Conclusion

Project Summary and Achievements

The STI Timetable Management System (STI-TTMS) successfully delivers a centralized, role-based scheduling platform that addresses the inefficiencies of manual timetable management in academic institutions. Through the integration of pure PHP, HTML, CSS, JQuery, Bootstrap, MySQL, and Ajax-driven interactions, the system enables administrators to generate accurate, conflict-free timetables, while lecturers and students can securely view their schedules from any device.

The project achieved all core objectives:

- Efficient timetable generation with automated conflict detection for courses, lecturers, and venues.
- Role-based access control, ensuring that administrators, lecturers, and students interact only with relevant data and features.
- Responsive design, allowing the platform to function seamlessly across desktops, tablets, and smartphones.
- Robust backend architecture, ensuring secure authentication, optimized database queries, and maintainable code.

Comprehensive testing — including unit testing, integration testing, and user acceptance testing — validated the platform’s functionality, reliability, and usability. All test scenarios, from timetable creation to role-specific dashboard access, passed successfully with minimal adjustments required.

Lessons Learned

The development process provided several technical and project-management insights:

1. Conflict detection logic is both essential and complex — careful database design and algorithm optimization were key to ensuring speed and accuracy.

2. Data migration from manual records requires structured planning to avoid inconsistencies and missing information.

3. Clear communication with stakeholders ensured that the system aligned with actual academic workflows.

4. Frontend–backend integration using well-structured APIs and Ajax significantly improved the user experience.

Challenges Encountered

- Avoiding timetable conflicts while accommodating constraints such as lecturer availability, room capacity, and course requirements.

- Optimizing query performance for faster timetable generation under realistic academic data loads.

- Designing intuitive interfaces for three distinct user roles without overcomplicating navigation.

Further Work

Although the current STI-TTMS meets all primary requirements, future enhancements could further improve usability and adoption:

- Automated notifications via email or SMS when timetable changes occur.

- Preference-based scheduling, allowing lecturers to indicate available or preferred teaching times.

- Export and print options for timetables in PDF/Excel formats.

- Dedicated mobile applications for Android and iOS platforms.

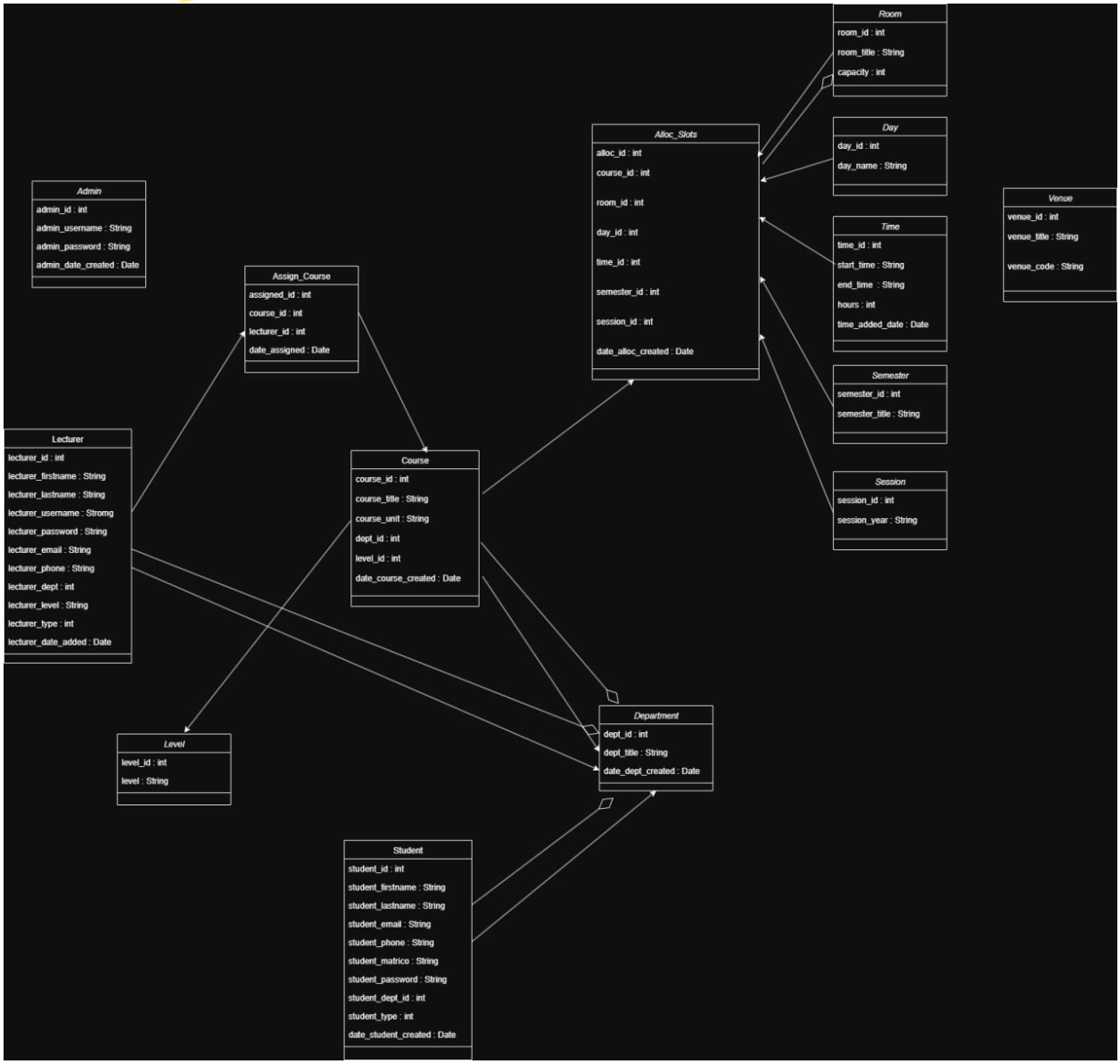
- Integration with learning management systems (LMS) for unified academic planning.

Critical Evaluation Design Chatper 3 need to add critical evaliton chapter

The requirements captured at the start of the project were largely fulfilled, with additional minor usability improvements identified and implemented during UAT. The design choices — particularly the Laravel–React stack — balanced performance, maintainability, and scalability for the university context. The project stayed close to its original plan, with minor timeline extensions during testing to fine-tune conflict-detection algorithms.

From a quality perspective, the STI-TTMS has demonstrated stability, security, and adaptability for future expansion, making it a viable long-term scheduling solution for academic institutions.

Class diagram line be careful



Sequence Diagram for STITTMS
Admin

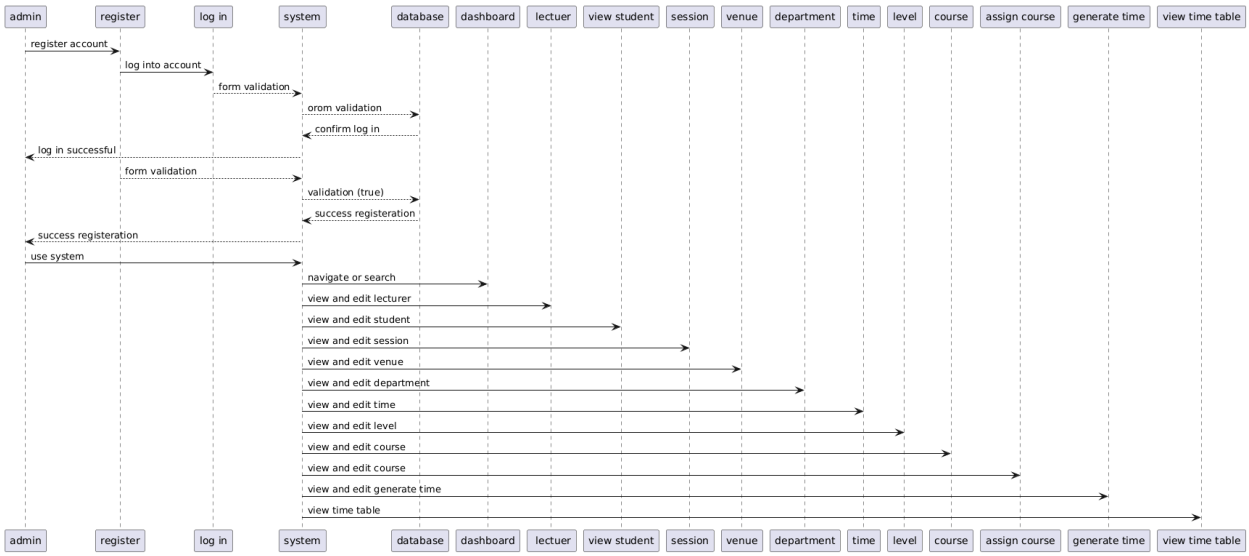


Figure : Sequence diagram of Adminstration

This diagram illustrates the process of how an administrator registers, logs in, and interacts with the system to manage various entities.

- **Registration and Login:** The admin first registers an account. The system validates the form, confirms the registration, and then allows the admin to log in. During login, the system validates the credentials and confirms a successful login.
- **System Interaction:** Once logged in, the admin can navigate the dashboard. The diagram shows the admin interacting with the system to **view and edit** lecturers, students, sessions, venues, departments, time, and level. The admin can also **view and edit** courses and assigned courses.
- **Timetable Generation:** A key interaction is the process to **view and edit** the timetable generation and then finally **view** the generated timetable.

Lecture

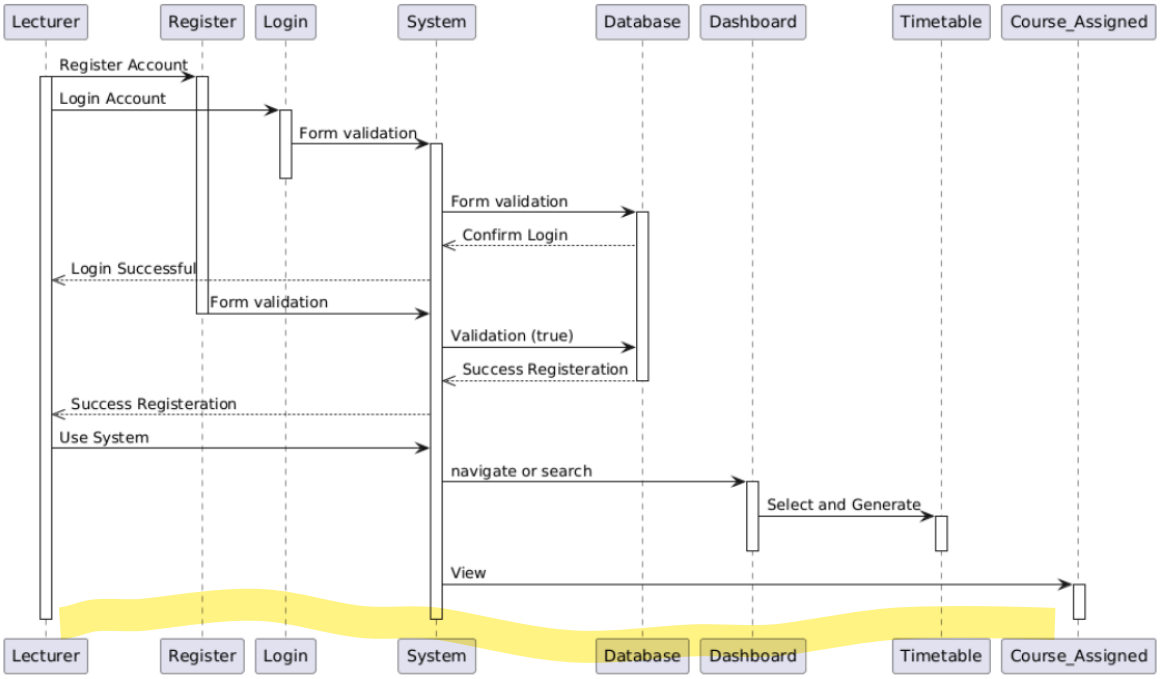


Figure : Sequence diagram of Lecturer

This diagram details the process for a lecturer to register, log in, and view their assigned information.

- **Registration and Login:** Similar to the admin, a lecturer can register and log into the system. The process involves form validation and confirmation by the system.
- **System Interaction:** After a successful login, the lecturer can use the system to **navigate or search**. The diagram highlights the lecturer's ability to **view** their assigned timetable and courses.
- **Timetable Generation:** The lecturer can also **select and generate** their timetable, which is then displayed to them.

Student

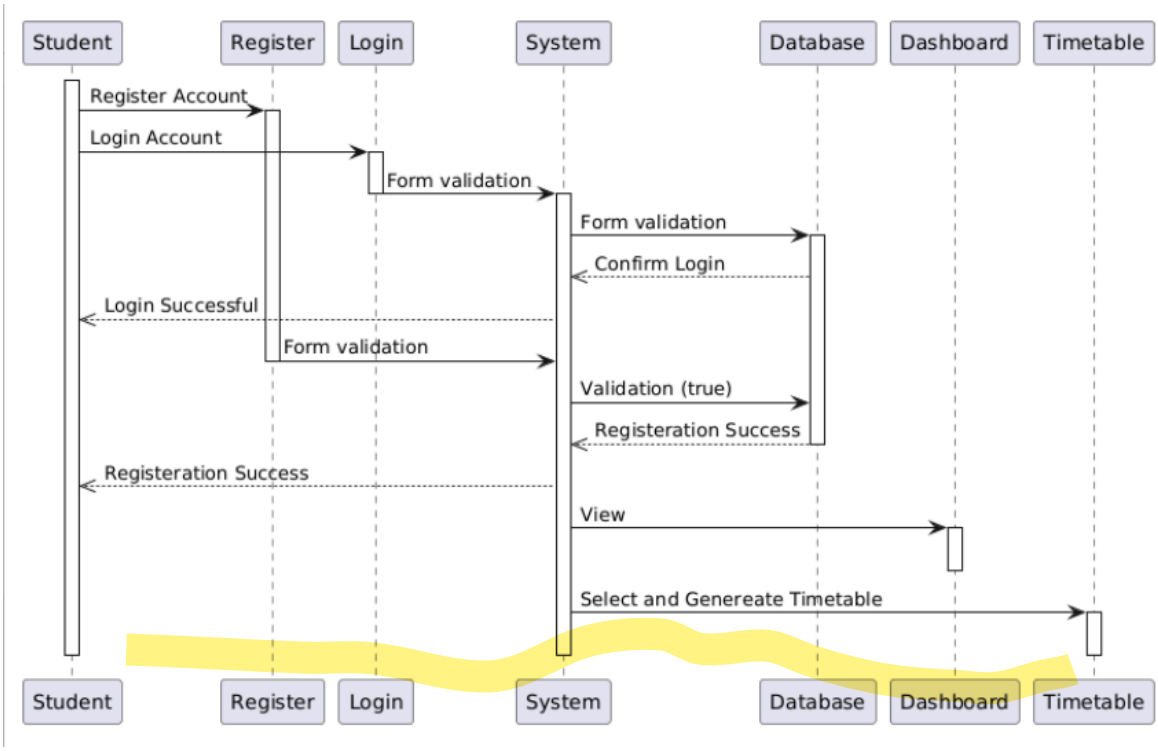


Figure : Sequence diagram of Student

This diagram outlines the student's journey from registration to viewing their academic timetable.

- **Registration and Login:** The student follows the same registration and login process as the other users, with the system handling form validation and confirmation.
- **System Interaction:** After a successful login, the student can use the system to **view** their timetable on the dashboard.
- **Timetable Generation:** The student can also **select and generate** their personalized timetable.

Testing

Introduction

This chapter presents the testing methodology applied to the STI Time Table Management System (STI TTMS) to verify its functionality, usability, and reliability. Testing was performed to ensure that the system meets all specified functional and non-functional requirements, including accurate timetable generation, user role management, and system performance. The testing process combined multiple levels of validation—from isolated components to the complete integrated system—providing confidence that the system is robust and user-friendly.

Testing Strategy

To thoroughly validate STI TTMS, a multi-level testing strategy was implemented, including Unit Testing, Integration Testing, and User Acceptance Testing (UAT). Each level focuses on different aspects of the system to ensure comprehensive quality assurance:

- **Unit Testing**: This involves testing individual functions and classes in isolation to confirm they behave as expected. Examples include verifying that login validation correctly authenticates users and that timetable generation produces accurate scheduling data.

- **Integration Testing**: Integration testing ensures that multiple modules work together correctly. For example, it verifies that the timetable creation module interacts properly with the database and that updates are reflected accurately in the user interface.

- **User Acceptance Testing (UAT)**: UAT involves real users—such as administrators, lecturers, and students—who test the system in realistic scenarios. This phase confirms that the system meets user needs and expectations in practical use.

Unit Testing

Unit Test 1: Lecturer Registration and Timetable Initialization

Field	Description
Unit Test ID	UT-01
Test Class	`LecturerController` – handles lecturer registration
Designed By	Yan Myo Aung
Data Source	New lecturer registration form data
Objective	Verify that registering a new lecturer
Tester	Yan Myo Aung
Test Case	Register a new lecturer through the admin interface
Description	Confirm that upon registration
Tasks	1. Navigate to Admin Dashboard 2. Select 'Add Lecturer' 3. Enter valid details 4. Submit form
Expected Result	System generates the lecturer and displays a success message
Actual Result	Actual results matched the expected results. The lecturer was created successfully without errors.

add some ss

Unit Test 2: Timetable Generation Algorithm

Field	Description
Unit Test ID	UT-02
Test Class	`TimetableGenerator` – responsible for creating conflict-free timetables
Designed By	Yan Myo Aung
Data Source	Course, lecturer, and room availability data
Objective	Ensure that timetable generation algorithm produces valid, conflict-free schedules
Tester	Yan Myo Aung
Test Case	Run timetable generation with sample input data
Description	Validate that the generated timetable has no overlapping sessions and all courses are assigned
Tasks	1. Provide sample data for courses, lecturers, rooms 2. Execute generation function 3. Review generated timetable
Expected Result	Timetable is generated without conflicts and includes all assigned courses
Actual Result	Actual results matched the expected results. The timetable was generated successfully without conflicts.

Integration Testing

Integration Test 1: Lecturer Assignment and Timetable Update Workflow

Field	Description
Integration Test ID	IT-01
Modules Involved	Lecturer Registration, Course Assignment, Timetable Generation, Timetable Display
Objective	Verify that lecturer assignment updates are correctly reflected in the generated timetable
Data Source	Lecturer and course details entered via admin interface
Description	Confirm that assigning a course to a lecturer triggers timetable regeneration and updates UI
Tasks	1. Register a lecturer 2. Assign courses to lecturer 3. Generate timetable 4. Verify updated timetable on UI
Expected Result	Timetable shows assigned courses and no conflicts, visible correctly to all user roles
Actual Result	Actual results matched the expected results. The updated timetable reflected all course assignments accurately.

Integration Test 2: Role-Based Access Control

Field	Description
Integration Test ID	IT-02
Modules Involved	Authentication, Authorization, Dashboard Views
Objective	Ensure that users see appropriate interfaces and data based on their roles (Admin, Lecturer, Student)
Data Source	Test user accounts with different roles
Description	Login as different roles and confirm UI elements and accessible functions correspond correctly
Tasks	1. Login as Admin 2. Login as Lecturer 3. Login as Student 4. Observe differences in views and permissions
Expected Result	Each user role accesses only permitted features and data
Actual Result	Actual results matched the expected results. All user roles accessed only their permitted areas without issues.

User Acceptance Testing (UAT)

UAT was conducted by a group of target users including faculty administrators, lecturers, and students. Participants completed typical tasks such as logging in, viewing timetables, and requesting timetable changes. Feedback indicated the system is intuitive and meets user expectations. Minor usability enhancements identified during UAT were promptly addressed.

Summary

The testing phase confirmed that TTMS meets its functional and usability goals. Unit and integration tests validated that individual modules and their interactions work correctly, while UAT verified the system's effectiveness in realistic scenarios. These results demonstrate that STI TTMS is stable, reliable, and ready for deployment.

Appendices

Appendix 1

No.	Functional Requirement	MoSCoW Prioritization
1	Users (Admins, Lecturers, Students) can register accounts with email, username, and password	Must Have
2	Users can securely log in and log out of the system	Must Have
3	Admins can manage lecturers (add, edit, delete, view)	Must Have
4	Admins can manage student records (add, edit, delete, view)	Must Have
5	Admins can add, edit, and delete course details	Must Have
6	Admins can assign and unassign courses to lecturers	Must Have
7	Admins can manage departments and academic levels	Must Have
8	Admins can manage venues (add, edit, delete, set capacity)	Must Have
9	Admins can configure and manage time slots (start/end time, lecture hours)	Must Have
10	Admins can generate timetables by selecting course, lecturer, venue, date, and time	Must Have
11	Admins can view and download timetables in PDF or Excel format	Must Have
12	Lecturers can view their upcoming classes from the dashboard	Must Have
13	Lecturers can generate their personalized timetable based on session, semester, and level	Must Have
14	Lecturers can view a list of assigned courses with department details	Must Have
15	Students can refresh or regenerate their timetable to reflect latest changes	Must Have
16	Lecturers can search their assigned courses and timetable entries	Should Have

17	Admins can search for lecturers, courses, and venues without navigating menus	Should Have
18	System prevents allocation conflicts (course, lecturer, room)	Should Have
19	Admins can allow specific course-batch combinations	Should Have
20	Admins can freeze timetables to prevent further edits	Should Have
21	System supports multiple faculties and departments	Should Have
22	Students can search timetable by course code or lecturer name	Could Have
23	Admins receive alerts for room capacity conflicts	Could Have
24	Students can subscribe for timetable change notifications	Could Have
25	Lecturers receive notification of class changes via email	Could Have
26	System supports multi-language interface	Won’t Have
27	Integration with external Learning Management Systems	Won’t Have
28	Automated allocation of rooms based on class size	Won’t Have

Appendix 2 - Database Table Definitions

Table Name	Key Fields	Description
admin	admin_id, admin_username, admin_password, admin_date_created	Stores administrator login details and account creation date.
alloc_slots	alloc_id, course_id, room_id, day_id, time_id, semester_id, session_id, date_alloc_created	Tracks allocated timetable slots, linking courses, rooms, days, time periods, semesters, and sessions.
assigned_course	assigned_id, course_id, lecturer_id, date_assigned	Tracks which lecturers are assigned to which courses.
batches	batch_name, batch_dept, size	Stores information about student batches including department and size.
config	Name, value	Stores system configuration settings.
course	course_id, course_title, course_unit, dept_id, level_id, date_course_created	Contains the list of courses offered, linked to department and level.
courses	course_id, course_name, fac_id, allow_conflict	Stores courses with faculty linkage and a flag for whether scheduling conflicts are allowed.
day	day_id, day_name	Stores days of the week.
department	dept_id, dept_title, date_dept_created	Stores academic departments.
depts	dept_code, dept_name	Stores department codes and names for faculty linkage.
faculty	uName, fac_name, pswd, level, dept_code, dateRegd	Stores faculty login details and department associations.
lecturer	lecturer_id, lecturer_firstname, lecturer_lastname, lecturer_username, lecturer_password, lecturer_email, lecturer_phone, lecturer_dept, lecturer_level,	Stores lecturer profiles and login credentials.

	lecturer_type, lecturer_date_added	
level	level_id, level	Stores academic levels .
rooms	room_id, room_title, capacity	Stores available teaching rooms and their capacities.
semester	semester_id, semester_title	Stores semester information.
session	session_id, session_year	Stores academic sessions by year.
slots	table_name, day, slot_num, state	Defines available timetable slots for a timetable table.
slot_allocs	table_name, day, slot_num, room, course_id	Stores the allocation of slots to courses and rooms.
student	student_id, student_firstname, student_lastname, student_email, student_phone, student_matricno, student_password, student_dept_id, student_type, date_student_created	Stores student records and login details.
time	time_id, start_time, end_time, hours, time_added_date	Stores time periods for timetable slots.
timetables	table_name, days, slots, duration, start_hr, start_min, start_mer, allowConflicts, frozen, active	Stores general timetable configuration settings.
venue	venue_id, venue_title, venue_code, date_venue_created	Stores venue details for classes.

Use Case Description for Admins

Use Case	Actor	Description
Login	Admin	Login Admin Authenticates an administrator to access the system dashboard.
Manage Lecturers	Admin	Manage Lecturers Admin Add, update, or remove lecturer profiles and contact information.
Manage Students	Admin	Manage Students Admin Add, update, or remove student profiles and details.
Manage Departments and details	Admin	Manage Departments Admin Create and update department names and codes.
Manage Courses	Admin	Manage Courses Admin Add, edit, or delete course titles, codes, and unit values.
Assign Courses	Admin	Assign Courses to Lecturers Admin Link courses to specific lecturers for teaching duties.
Allocate Courses	Admin	Allocate Courses to Time Slots Admin Schedule courses into rooms, days, and time slots.
Manage Venues and room capacity	Admin	Manage Venues Admin Add or update room names and capacities.
Set timetable Settings	Admin	Configure Timetable Settings Admin Set number of slots per day, start time, and duration.
Timetable control	Admin	Freeze Timetable Admin Lock a timetable to prevent accidental edits.

Use Case Description for Lecturers

Use Case	Actor	Description
View assigned timetable	Lecturer	View Assigned Timetable Lecturer Check teaching schedule by day, time, and room.
Searching timetable	Lecturer	Search Timetable Lecturer Search by course code or session year.

Use Case Description for Students

Use Case	Actor	Description
View Timetable	Student	View Timetable Student View personal or departmental timetable for the current semester.
Search Timetable	Student	Search Timetable Student Search timetable by course or lecturer.

code
reference
code setup(xampp install/-----)