VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY
INTERNATIONAL UNIVERSITY

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

WEB APPLICATION DEVELOPMENT

STUDY PLANNING

Course name: Web Application Development

Course ID: IT093IU

Theory class lecturer: Assoc. Prof. Nguyen Van Sinh

Laboratory class lecturer: MSc. Nguyen Trung Nghia

No.	Full name	Student ID	Role	Contribution
1	Phạm Tuấn Đăng Khoa	ITITIT22087	Leader	
2	Nguyễn Xuân An	ITITIT22001	Member	
3	Đặng Nguyễn Nhật Vy	ITITDK22102	Member	

Table Of Content

List of Table

CHAPTER 1: ACKNOWLEDGMENTS

This final project is the result of collective effort, and we wish to extend our sincere appreciation to all who contributed to its completion.

First and foremost, we would like to express our profound gratitude to our supervisors, Assoc. Prof. Nguyen Van Sinh and MSc. Nguyen Trung Nghia. Their expert guidance, scholarly insights, and steadfast encouragement were fundamental in shaping this research. We are deeply indebted to them for their patience and the high academic standards they inspired us to uphold.

We also wish to recognize the invaluable contributions of our team members. Their proactive initiative, combined with a shared commitment to excellence, was the cornerstone of this project. The synergy within the group not only enhanced the quality of this report but also made the research process a truly rewarding professional experience.

Furthermore, we extend our thanks to all individuals who provided technical assistance and valuable perspectives. Every contribution, regardless of its scale, played a vital role in refining the final outcome.

Finally, we dedicate this work to the spirit of collaboration. It is our hope that the knowledge and skills gained throughout this journey will serve as a robust foundation for our future professional endeavors.

CHAPTER 2: EXECUTIVE SUMMARY

1. Overview

In the context of modern web-based systems, web applications play an increasingly important role in supporting learning and daily activities. Today's learners are required to manage multiple subjects, overlapping deadlines, and various academic tasks, while traditional planning methods such as manual notes or static schedules no longer meet the demands for flexibility and personalization.

The Study Planning project is developed as a web application that supports users in planning and managing their learning process effectively. The system allows users to organize subjects, define learning goals, create study schedules, and track learning progress through an intuitive and user-friendly web interface. By being deployed on a web platform, the application enables users to access the system anytime and anywhere without dependency on a specific device or operating system.

In addition, modern learning environments emphasize self-directed learning and personal responsibility.Study Planning addresses this trend by providing features that allow users to

actively design and adjust their own learning plans according to individual needs. Instead of applying a fixed planning model, the system offers flexibility in managing study time, priorities, and objectives based on real-life situations.

Furthermore, personalization is a key factor in contemporary web application design. Study Planning enables users to customize their study plans according to their daily routines and learning pace, thereby improving the practicality and sustainability of the planning process. Through progress visualization, the system helps users monitor their achievements and maintain learning motivation.

In summary, Study Planning is a web-based application developed to address the challenges of personal study management in digital learning environments. The project demonstrates the practical application of concepts and techniques learned in the Web Application course, as well as the ability to design and implement a complete, functional, and extensible web system.

2. Objectives and Scope

2.1 Objectives

The primary objective of the Study Planning project is to develop a web-based application that supports learners in organizing and managing their academic activities effectively. In modern learning environments, students are increasingly required to handle multiple subjects, tasks, and deadlines simultaneously. The system aims to assist users in defining clear learning goals and structuring their study plans in a way that improves time management and learning efficiency.

Another important objective of the project is to enhance user engagement and learning consistency through progress tracking and feedback mechanisms. By allowing users to monitor their learning progress visually, the application encourages accountability and helps maintain motivation throughout the study process. This objective directly contributes to improving the overall learning experience and reducing the likelihood of incomplete or abandoned study plans.

In addition, personalization is a key objective of Study Planning. The application is designed to adapt to individual learning habits, daily routines, and study preferences rather than enforcing a rigid planning model. By enabling users to adjust schedules, priorities, and goals dynamically, the system aims to increase the practicality and long-term usability of the learning plans created within the application.

From a technical perspective, the project also aims to apply and demonstrate core concepts learned in the Web Application course. These include designing a structured web system,

managing user interaction, handling data persistence, and implementing core functionalities through a responsive and user-friendly interface.

2.2 Scope

The scope of the Study Planning project focuses on the development of a web-based system that facilitates personal study planning and progress management. The application supports digital data management related to users, subjects, learning goals, schedules, and task completion status. All interactions within the system are handled electronically through a web interface, ensuring accessibility and ease of use.

The project is technologically facilitated through standard web technologies, enabling users to access the system via a web browser without the need for specialized hardware or software. The application emphasizes core web application functionalities such as user interaction, data processing, and dynamic content rendering. The system is designed to operate across different devices, ensuring basic compatibility with both desktop and mobile environments.

Furthermore, the scope of the project includes enhancing the overall user experience by focusing on usability, clarity, and responsiveness. Key considerations include intuitive navigation, clear presentation of information, and efficient interaction flows when creating or updating study plans. While advanced features such as intelligent recommendations or analytics may be considered for future development, the current scope is limited to essential functionalities that align with the requirements and time constraints of a final project.

In summary, the scope of Study Planning is defined to deliver a functional and practical web application that supports personal study management while demonstrating the application of web development principles taught in the course.

3. Key Achievements and Milestones

Throughout the development of the Study Planning project, several key achievements and milestones were accomplished, reflecting the systematic implementation of the application within the scope of a web application course. The project began with the analysis of user requirements, where common challenges in personal study management were identified and translated into functional system requirements. This phase laid the foundation for designing a solution that aligns with real-world learning needs.

Following the requirement analysis, the overall system architecture and workflow were designed to ensure logical structure and efficient user interaction. Core components of the application, including user management, subject organization, study scheduling, and progress tracking, were implemented using web-based technologies. These functionalities

represent the essential features required for a practical and functional study planning system.

Another significant milestone was the development of a user-friendly interface that supports intuitive navigation and clear information presentation. Attention was given to usability and responsiveness to ensure that users can interact with the system effectively across different devices. The integration of progress visualization features further enhanced user engagement by allowing learners to monitor their achievements and study consistency.

Finally, the system underwent functional testing to verify the correctness and stability of the implemented features. This testing phase ensured that the core functionalities operated as expected and met the objectives defined at the beginning of the project. The successful completion of these milestones demonstrates the feasibility and effectiveness of the Study Planning Web application within the constraints of a final academic project.

4. Summary of Project Outcome

The Study Planning project achieved significant results in developing a web-based application that supports personal study planning and management. The system was implemented in accordance with the initial project objectives and provides users with core functionalities such as goal setting, study scheduling, plan personalization, and progress tracking through an intuitive web interface.

Through the development process, the project demonstrated the effective application of knowledge and skills acquired in the Web Application course, including system architecture design, user interaction handling, and web-based data management. The main features of the system operate reliably and meet the fundamental requirements of a complete web application within the scope of a final academic project.

In addition to its technical contributions, the project offers practical value by helping learners improve time management and study organization. Progress visualization enables users to gain a comprehensive view of their learning journey, thereby enhancing self-regulation and sustaining learning motivation.

Although certain limitations remain due to time and resource constraints, Study Planning establishes a solid foundation for future development. The achieved outcomes highlight the system’s potential for expansion and confirm the suitability of the proposed solution in supporting personal learning management through web-based technologies.

CHAPTER 3: INTRODUCTION

1. Goal

1.1 Customer Service

In order to improve accessibility and user convenience, Study Planning supports multiple account registration methods. In addition to the traditional account creation process on the website, users can easily register and log in using their Google or Facebook accounts. This approach reduces the time and effort required for account creation, lowers entry barriers for new users, and enhances the overall onboarding experience.

The application is designed to be completely free of charge, making it accessible to users of different age groups and backgrounds. By removing financial constraints, Study Planning ensures that learners can fully utilize the system's features without limitations, thereby promoting inclusivity and wider adoption.

Furthermore, the system emphasizes a user-friendly interface with simple navigation and clear presentation of information. The interface is designed to be intuitive and easy to use, even for users with limited technical experience. Important features such as study schedules, goals, and progress tracking are organized logically to allow users to interact with the system efficiently.

To support effective communication and user engagement, the application also integrates essential date-related information and contact features, enabling users to manage deadlines accurately and interact with the system in a structured manner. These design choices contribute to a positive user experience and align with modern web application standards.

Overall, by combining multiple registration options, free accessibility, and a user-centered interface, Study Planning enhances usability and ensures that the application meets the needs of a broad range of users in a modern web-based learning environment.

1.2 Platform Performance

Platform performance is a critical factor in ensuring the effectiveness and usability of the Study Planning system. The application is designed to deliver a smooth, responsive, and reliable experience that supports users in managing their study schedules without interruption or delay.

Fast page loading and efficient data processing: To achieve optimal performance, the system prioritizes fast page loading and efficient data processing. The interface is optimized to reduce unnecessary requests and ensure that pages load quickly, allowing users to access study schedules, progress tracking, and time management tools with minimal waiting time. This responsiveness is especially important for features such as timetable generation, real-time progress updates, and the study timer, which require immediate feedback to maintain user focus and productivity.

Stability and scalability: In terms of stability and scalability, the platform is structured to handle multiple users simultaneously without performance degradation. The system is capable of managing increasing volumes of user data, including personal preferences, study plans, and progress records, while maintaining consistent performance. This ensures that the application remains reliable during peak usage periods and can be expanded in the future to accommodate a larger user base.

Smooth navigation and seamless user interaction: The platform also emphasizes smooth navigation and seamless user interaction. Page transitions are designed to be intuitive and efficient, enabling users to move easily between features such as timetable creation, study time visualization, goal tracking, and the timer tool. This contributes to a user-friendly experience and reduces cognitive load during usage.

Data integrity and system reliability: Finally, the system is built with attention to data integrity and system reliability. User data is processed consistently to ensure accurate schedule recommendations and progress tracking. By combining performance optimization, system stability, and responsive design, Study Planning delivers a dependable platform that effectively supports personalized study planning and long-term learning efficiency.

2. A Brief Overview of The Report's Structure

This report is structured into multiple chapters that collectively present a comprehensive overview of the Study Planning project. The report begins with an executive summary and an introduction, which outline the project background, problem statement, and overall objectives. These sections provide readers with a clear understanding of the motivation and purpose behind the development of the system.

Subsequent chapters focus on project planning and requirements analysis, detailing the functional and non-functional requirements derived from user needs. This is followed by chapters on system design and architecture, where the overall structure, technologies used, and architectural decisions of the web application are explained.

The development chapter describes the implementation process, including core features such as personalized study schedule generation, timetable management, progress tracking, and timer functionality. This is complemented by the testing and quality assurance chapter, which discusses testing strategies, system validation, and measures taken to ensure reliability and performance.

Next, the report presents the deployment and implementation chapter, explaining how the system is deployed and prepared for real-world usage. The user documentation chapter

provides guidance on how users interact with the platform, highlighting the user-friendly interface and navigation flow.

The project evaluation and lessons learned chapters reflect on the outcomes of the project, assess its effectiveness in meeting the original objectives, and identify challenges encountered during development. These insights contribute to a deeper understanding of both technical and teamwork aspects of the project.

The report concludes with a conclusion chapter that summarizes the entire project and its achievements. Additional supporting materials are included in the appendices, while all sources of information, data, and references used throughout the report are properly cited in the references section to ensure academic integrity and transparency. Finally, the acknowledgments section expresses gratitude to supervisors, instructors, and individuals who provided guidance and support throughout the successful completion of the Study Planning project.

CHAPTER 4: PROJECT PLANNING

1. Project Management Method

1.1 Agile

The Study Planning project adopts the Agile methodology to support flexible and iterative development throughout the project lifecycle. Agile is particularly suitable for this web application project because system requirements may evolve during development as user needs become clearer through implementation and testing.

By applying Agile principles, the project is developed incrementally, allowing core functionalities—such as user registration, study schedule creation, progress tracking, and time management—to be implemented and refined in multiple iterations. This approach enables early detection of issues, continuous improvement of features, and timely adjustments without disrupting the overall project structure.

Agile also emphasizes collaboration and frequent communication among team members. Clear goals are defined at the beginning of each development iteration, ensuring that all members work consistently toward the project objectives. This iterative process improves development efficiency, enhances software quality, and reduces potential risks during implementation.

1.2 Scrum

Within the Agile methodology, the project utilizes the Scrum framework to manage and organize development activities effectively. Scrum provides a structured approach that supports transparency, regular evaluation, and continuous adaptation.

The development process is divided into several Sprints, with each Sprint focusing on specific functionalities. For example, initial Sprints concentrate on system architecture,

database design, and MVC structure implementation, while subsequent Sprints focus on developing features such as personalized timetable management, task tracking, progress visualization, and user interface optimization. Each Sprint results in a functional increment of the system that can be reviewed and tested.

Scrum is implemented based on its three core principles:

- Transparency: All project tasks, progress, and requirements are clearly documented and shared among team members, ensuring a common understanding of the system's current state.

- Inspection: At the end of each Sprint, the system is reviewed and tested to identify defects, evaluate performance, and verify that implemented features meet the defined requirements.

- Adaptation: Based on inspection results and feedback, necessary adjustments are made to improve functionality, usability, or system performance in subsequent Sprints.

Scrum is selected for this project because it supports efficient project management, improves team coordination, and enables continuous improvement. By combining Agile principles with the Scrum framework, the Study Planning Web project achieves a structured yet flexible development process that ensures timely delivery and a high-quality web application.

2. Project Timeline and Milestones

2.1 Gantt Chart

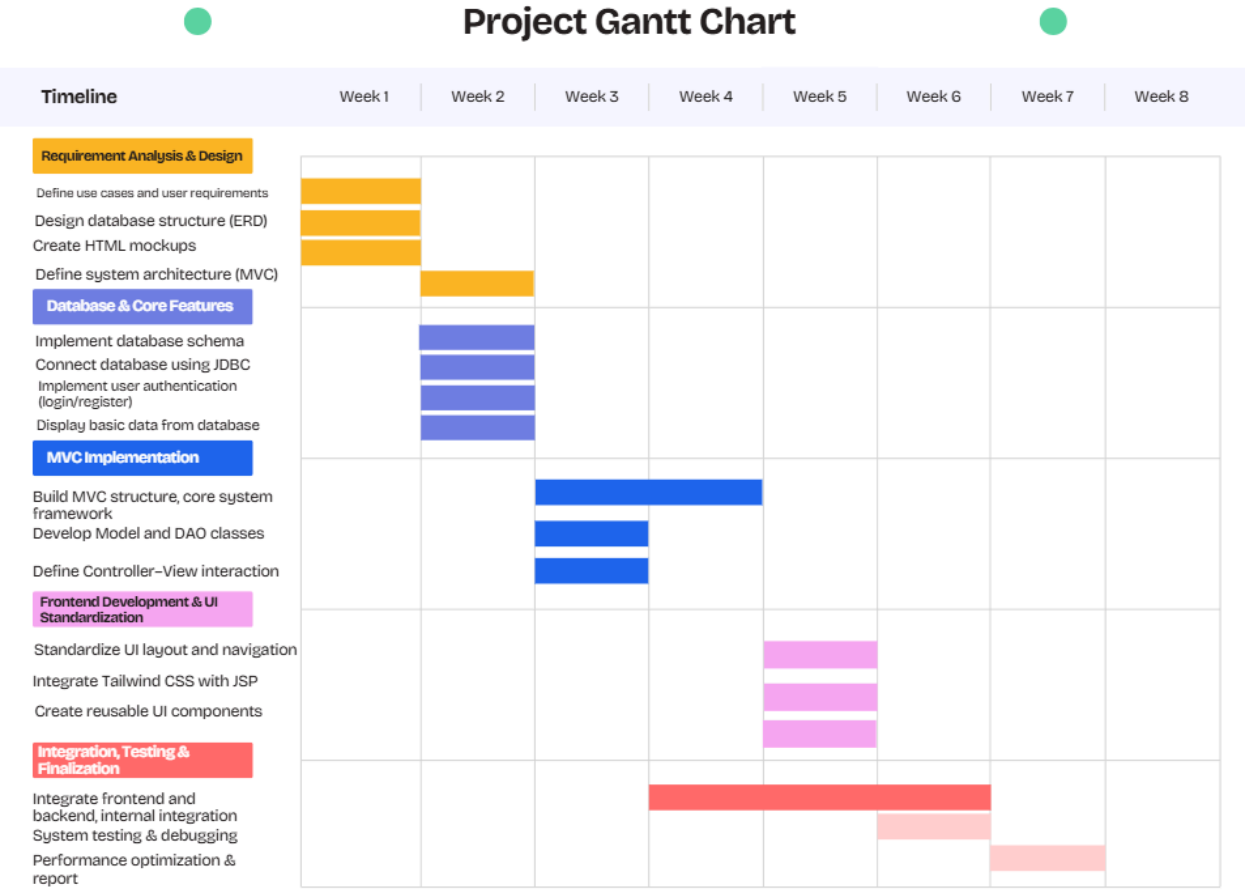

Figure 1: Gantt Chart

The project progress is rigorously managed through a Gantt chart with a total planned duration of 8 weeks, divided into specialized phases marked by distinct colors for easy identification. Tasks that have been completed are represented by solid, dark-colored blocks (Orange, Purple, Blue, Pink, and Red), while lighter-colored blocks indicate tasks that are either pending or currently in progress. Specifically, the project has successfully navigated through key milestones, including Requirement Analysis & Design (Weeks 1-2), Database & Core Features (Week 2), MVC Implementation (Weeks 3-4), and Frontend Development (Week 5), all achieved with a 100% completion rate. Currently, the project is focusing on the final phase, Integration, Testing & Finalization; within this stage, system integration has been finalized, while testing, debugging, and performance optimization (indicated by the light-shaded blocks) are being executed sequentially to ensure final delivery by the end of Week 8.

CHAPTER 5: REQUIREMENTS ANALYSIS

1. Description of Project Requirements

1.1 Functional Requirements

Reg.ID	Requirements Name	Detailed Description
001	User Registration & Login	The system shall allow students to register, log in, and log out securely in order to manage personal study data and timetables.
002	Input Personal Data	The system shall allow students to input and update personal information such as name, academic year, major, personality traits, and study preferences (morning/evening, group/individual).
003	Input Study Tasks	The system shall allow students to add, edit, and delete study tasks including subjects, deadlines, priority levels, and available free time.
004	Generate Study Timetable (Decision Support)	The system shall send user data to the ML model (Flask API) and receive a personalized study timetable recommendation based on preferences and constraints.
005	View Study Timetable	The system shall display the generated timetable in a clear visual format such as a table or calendar view.
006	Edit/ Delete Tasks	The system shall allow students to modify or remove study tasks after a timetable has been generated.
007	Save Timetable	The system shall store personalized timetables in the database for future access and reuse.
008	Update Study References	The system shall allow students to update learning preferences and regenerate a new timetable accordingly.
009	View Performance Analytics	The system shall display study performance statistics such as total study time and subject-wise productivity using charts and dashboards (Chart.js).
010	Give Feedback Model	The system shall allow students to provide feedback on timetable recommendations to improve future suggestions.

011	Auto-adjust Timetable	The system shall automatically update the timetable when deadlines, tasks, or preferences change.
012	Recommend Learning Resources	The system shall recommend learning resources (videos, documents, links) based on subjects or current study tasks.
013	Send Notifications/ Reminders	The system shall notify students of upcoming study sessions or deadlines via web notifications or email.

Table 1: Functional requirements of Study Planning

1.2 Nonfunctional Requirements

Req.ID	Requirement Name	Detailed Description
001	Performance	The system shall provide fast response times, minimal downtime, and smooth interaction when generating timetables, loading dashboards, and displaying analytics.
002	Scalability	The system shall support a growing number of users and study records without performance degradation by using scalable backend and database architecture.
003	Security	The system shall ensure secure authentication, encrypted data transmission, secure session management, and protection of personal user data.
004	Usability	The system shall provide an intuitive, user-friendly interface with clear navigation and minimal learning curve for students.
005	Accessibility	The system shall follow web accessibility standards (WCAG) to support users with disabilities, including keyboard navigation and readable UI components.
006	Reliability	The system shall reliably save and retrieve user data and timetables without data loss or corruption.
007	Maintainability	The system shall be designed using MVC architecture to allow easy maintenance, updates, and feature extension.

008	Compatibility	The system shall be compatible with modern web browsers and responsive across different screen sizes.

Table 2: Nonfunctional requirements of Study Planning

2. User Stories

- As a Student
  + Manage Study Tasks
    - In order to organize my study efficiently, I want to add, edit, and delete study tasks, including subjects, deadlines, and priority levels.
    - In order to focus on the most important tasks, I want to filter and sort study tasks by subject, priority, or deadline.
  + Generate and View Timetable
    - In order to have a personalized study plan, I want the system to generate a timetable based on my preferences, available time, and deadlines.
    - In order to track my study schedule, I want to view the generated timetable in a clear visual format such as a table or calendar.
    - In order to adjust my plan when needed, I want to edit or remove tasks after the timetable has been created.
  + Manage Learning Preferences
    - In order to get the most suitable timetable, I want to input and update my learning preferences (morning/evening, group/individual, study style).
    - In order to improve future timetable recommendations, I want to provide feedback on the generated timetable.
  + Track Study Performance
    - In order to monitor my progress, I want to view analytics such as total study time and subject-wise productivity through charts and dashboards.
  + Notifications and Timetable Export
    - In order to stay on top of deadlines and study sessions, I want to receive notifications and reminders via web or email.

- As a System/ ML Model
  + Timetable Generation
    - In order to provide personalized recommendations, I want to process user data and generate optimized study timetables using ML algorithms.
    - In order to ensure timetables are always up-to-date, I want to automatically adjust schedules when tasks, deadlines, or preferences change.
  + Learning Resource Recommendations
    - In order to help students study effectively, I want to suggest learning resources (videos, documents, links) based on subjects and tasks.

3. Diagram

3.1 Context Diagram

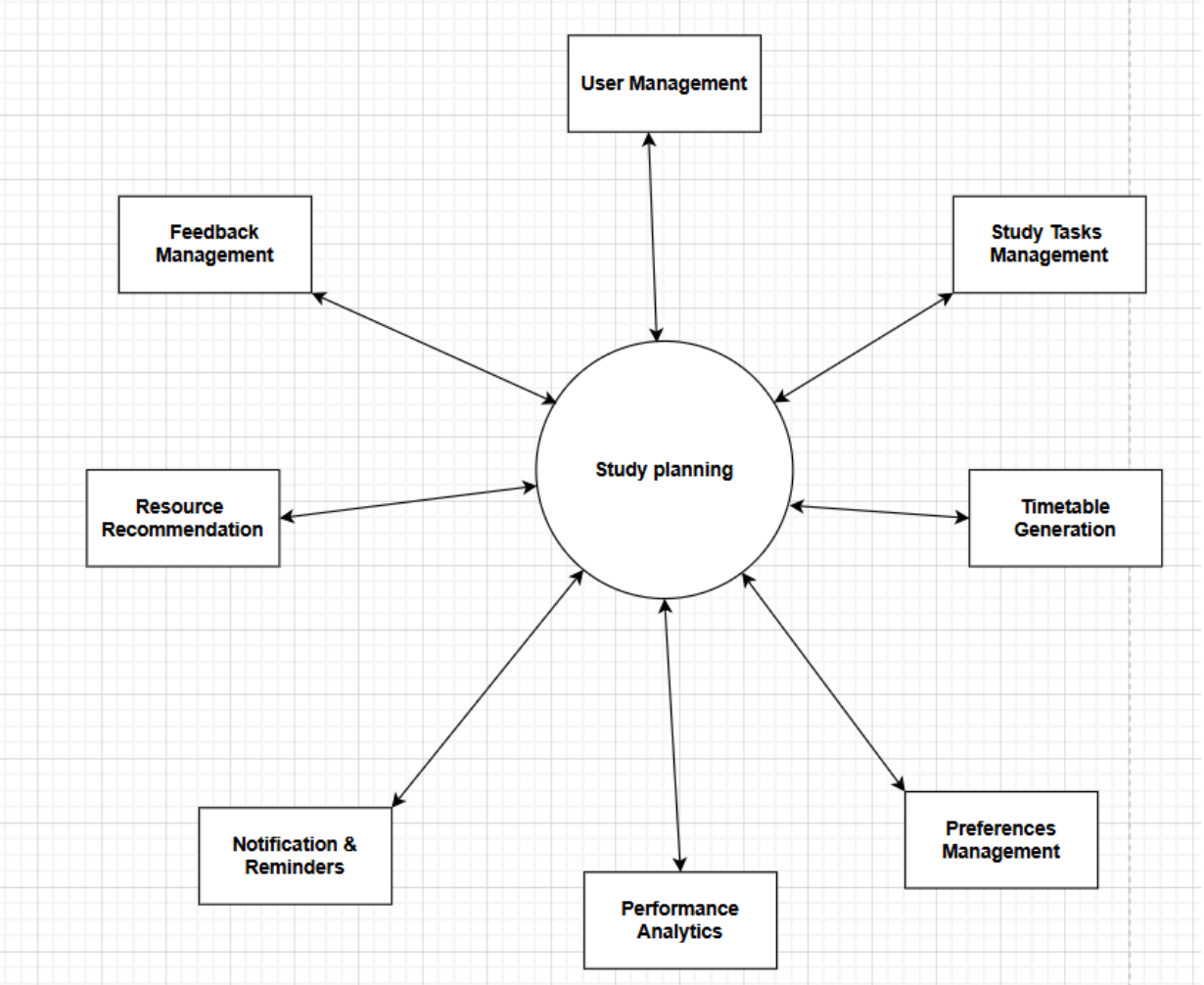

Figure 2: Context diagram

Action	Goal
User Management	Manage student accounts, including registration, login, logout, and profile updates.
Study Tasks Management	Manage students' study tasks, including adding, editing, deleting tasks, setting priorities, deadlines, and available time.
Timetable Generation	Generate personalized study timetables based on user data, learning preferences, and deadlines, using ML algorithms to optimize schedules.
Preferences Management	Allow students to input and update learning preferences (morning/evening, group/individual, study style) to adjust their timetables.

Performance Analytics	Collect and display study performance analytics, including total study time and subject-wise productivity, using charts and dashboards.
Notifications & Reminders	Send notifications and reminders to students about upcoming study sessions or deadlines.
Resource Recommendation	Recommend learning resources (videos, documents, links) based on subjects or current study tasks.
Feedback Management	Collect student feedback on generated timetables to improve future recommendations.

Table 3: Context diagram analysis

3.2 Use case diagram

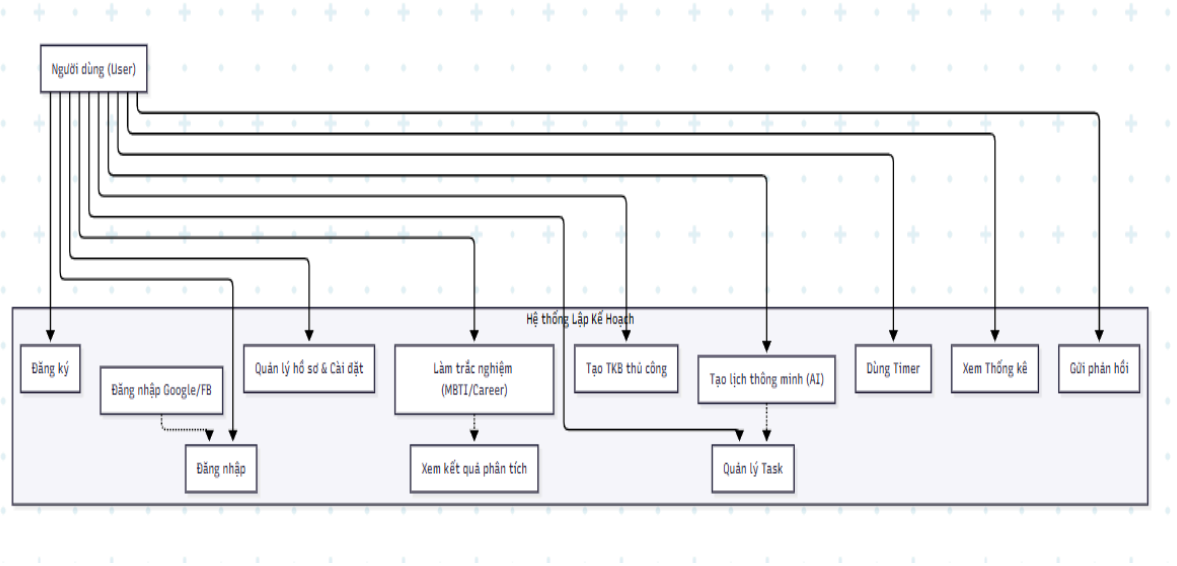

Figure 3: Use case diagram of Study Planning

3.2.1 Actor Analysis

Actor	Role
User (Student)	The primary user of the system, with the capability to input personal data, manage study tasks, track their timetable, and receive educational support suggestions.

Table 4: Actor Explanation

3.2.2 *Use Case Analysis*

Use case	Goal
Register / Login	Users register or log in to secure and store their personalized study path data.
Input Personal Data	Students provide background information such as major, preferences, and personality traits to personalize the system.
Input Study Tasks	Users add a list of specific subjects and tasks along with deadlines and their available free time.
Generation Timetable	The system collaborates with the ML Model to create a proposed timetable based on the provided input data.
View Timetable	Displays the optimized study schedule in a visual table or calendar interface.
Edit/ Delete Task	Allows users to flexibly modify content or remove study tasks after they have been created.
Performance Analytics	Collects and visualizes the user's study efficiency through statistical charts.

Table 5: Use cases explanation

3.2.3 *Relationship*

The use case "Generate Timetable" requires that the user has provided information in the "Input Personal Data" and "Input Study Tasks" use cases; therefore, "Generate Timetable" encompasses or directly depends on the data from these two input use cases.

The "Recommend Learning Resources" and "Send Notification" use cases expand the user experience after the timetable has been generated, supporting the effective execution of the study plan.

The relationship between the "User" and the "ML Model" through the "Generate Timetable" use case represents a two-way interaction: the user sends data requests and the model returns predicted results.

3.3 *Entity Relationship Diagram (ERD)*

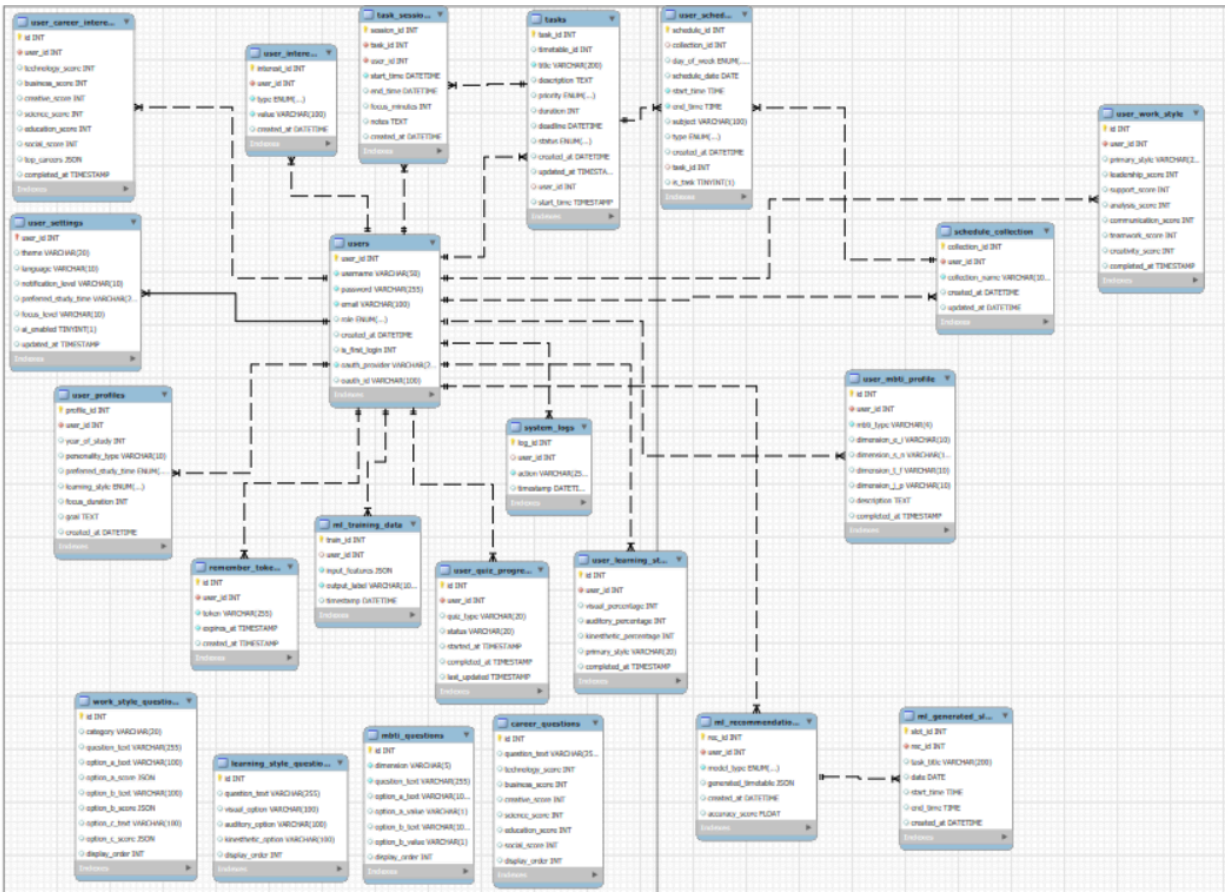

Figure 4: Entity Relationship Diagram (ERD)

3.3.1 Normal Form Analysis

Normal Form	Description
1 N.F.	The database does not have any multivalued tuples
2 N.F	All the non-key attributes depend on the primary key
3 N.F	There are no transitive dependencies between non-key attributes
BCNF	Every non-trivial functional dependency in the database depends on a candidate key

Table 6: Normal Form

3.3.2 Schemas and Table Analysis

Table	Attribute
users	user_id **INT** AUTO_INCREMENT **PRIMARY** KEY, username **VARCHAR(50)** NOT NULL UNIQUE, password **VARCHAR(255)** NOT NULL, email **VARCHAR(100)** NOT NULL UNIQUE, role **ENUM('student','admin')** DEFAULT 'student', created_at **DATETIME** DEFAULT CURRENT_TIMESTAMP, is_first_login **INT** DEFAULT 1, oauth_provider **VARCHAR(20)** NOT NULL DEFAULT LOCAL, oauth_id **VARCHAR(100)** UNIQUE DEFAULT NULL,
work_style_question	id **INT** AUTO_INCREMENT **PRIMARY** KEY, category **VARCHAR(20)** DEFAULT NULL, question_text **VARCHAR(255)** DEFAULT NULL, option_a_text **VARCHAR(100)** DEFAULT NULL, option_a_score **JSON** DEFAULT NULL, option_b_text **VARCHAR(100)** DEFAULT NULL, option_b_score **JSON** DEFAULT NULL, option_c_text **VARCHAR(100)** DEFAULT NULL, option_c_score **JSON** DEFAULT NULL, display_order **INT** DEAULT '0'
user_work_style	id **INT** AUTO_INCREMENT **PRIMARY** KEY, **FOREIGN** KEY (user_id) REFERENCES (user_id), primary_style **VARCHAR(20)** DEFAULT NULL, leadership_score **INT** DEFAULT '0', support_score **INT** DEFAULT '0', analysis_score **INT**DEFAULT '0', communication_score **INT** DEFAULT '0', teamwork_score **INT** DEFAULT '0', creativity_score **INT** DEFAULT '0', completed_at **TIMESTAMP** DEFAULT CURRENT_TIMESTAMP,
user_settings	user_id **INT** NOT NULL **PRIMARY** KEY theme **VARCHAR(20)** DEFAULT 'light' language **VARCHAR(10)** DEFAULT 'vi' notification_level **VARCHAR(10)** DEFAULT 'medium' preferred_study_time **VARCHAR(20)** DEFAULT NULL, focus_level **VARCHAR(10)** DEFAULT NULL, ai_enabled **TINYINT(1)** DEFAULT '1' updated_at **TIMESTAMP** DEFAULT CURRENT_TIMESTAMP,
user-schedule	schedule_id **INT** AUTO_INCREMENT **PRIMARY** KEY,

	FOREIGN KEY (collection_id) REFERENCES (collection_id), day_of_week **ENUM('Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun')** DEFAULT NULL, schedule_date **DATE** DEFAULT NULL, start_time **TIME** NOT NULL**,** end_time **TIME** NOT NULL**,** subject **VARCHAR(100)** NOT NULL DEFAULT NULL**,** type **ENUM('class', 'break', 'self-study', 'activity')** DEFAULT 'class', created_at **DATETIME** DEFAULT CURRENT_TIMESTAMP,, **FOREIGN** KEY (task_id) REFERENCES (task_id), is_task **TINYINT(1)** DEFAULT'0'
user_quiz_progress	id **INT** AUTO_INCREMENT **PRIMARY** KEY, **FOREIGN** KEY (user_id) REFERENCES (user_id), quiz_type **VARCHAR(20)** DEFAULT NULL, status **VARCHAR(20)** DEFAULT 'NOT_STRATED', start_at **TIMESTAMP** DEFAULT NULL, completed_at **TIMESTAMP** DEFAULT NULL, last_updated **TIMESTAMP** DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
user_profiles	profile_id **INT** AUTO_INCREMENT **PRIMARY** KEY, **FOREIGN** KEY (user_id) REFERRENCES (user_id), year_of_study **INT** DEFAULT NULL, personality_type **VARCHAR(10)** DEFAULT NULL, preferred_study_time **ENUM('morning', 'afternoon', 'evening', 'night')** DEFAULT NULL, learning_style **ENUM('visual', 'auditory', 'kinesthetic')** focus_duration **INT** DEFAULT '90' goal **TEXT** DEFAULT NULL, created_at **DATETIME** DEFAULT CURENT_TIMESTAMP,
user_mbti_profile	id **INT** AUTO_INCREMENT **PRIMARY** KEY, **FOREIGN** KEY (user_id) REFERRENCES (user_id), mbti_type **VARCHAR(4)** NOT NULL, dimension_e_i **VARCHAR((10)** DEFAULT NULL, dimension_s_n **VARCHAR(10)** DEFAULT NULL, dimension_t_f **VARCHAR(10)** DEFAULT NULL, dimension_j_p **VARCHAR(10)** DEFAULT NULL, description **TEXT** DEFAULT NULL, completed_at **TIMESTAMP** DEFAULT CURENT_TIMESTAMP,
user_learning_style	id **INT** AUTO_INCREMENT **PRIMARY** KEY, **FOREIGN** KEY (user_id) REFERRENCES (user_id), visual_percentage **INT** DEFAULT '0', auditory_percentage **INT** DEFAULT '0', kinesthetic_percentage **INT** DEFAULT '0',

	primary_style **VARCHAR(20)** DEFAULT NULL, completed_at **TIMESTAMP** DEFAULT CURENT_TIMESTAMP,
user_interests	interest_id **INT** AUTO_INCREMENT **PRIMARY** KEY **FOREIGN** KEY (user_id) REFERRENCES (user_id), type **ENUM('interest', 'dislike', 'hobby')** NOT NULL, value **VARCHAR(20)** NOT NULL created_at **DATETIME** DEFAULT CURENT_TIMESTAMP,
user_career_interests	id **INT** AUTO_INCREMENT **PRIMARY** KEY **FOREIGN** KEY (user_id) REFERRENCES (user_id), technology_score **INT** DEFAULT '0', business_score **INT** DEFAULT '0', creative_score **INT** DEFAULT '0', science_score **INT** DEFAULT '0', education_score **INT** DEFAULT '0', social_score **INT** DEFAULT '0', top_careers **JSON** DEFAULT NULL, completed_at **TIMESTAMP** DEFAULT CURENT_TIMESTAMP,
tasks	task_id **INT** AUTO_INCREMENT **PRIMARY** KEY, timetable_id **INT** DEFAULT NULL, title **VARCHAR(200)** NOT NULL, description **TEXT** DEFAULT NULL, priority **ENUM('low', 'medium', 'high')** DEFAULT 'medium', duration **INT** DEFAULT '90', deadline **DATETIME** DEFAULT NULL, status **ENUME('pending', 'in_progess', 'done')** DEFAULT 'pending', created_at **DATETIME** DEFAULT CURENT_TIMESTAMP, updated_at **TIMESTAMP** DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP, **FOREIGN** KEY (user_id) REFERENCES (user_id), start_time **TIMESTAMP** DEFAULT CURENT_TIMESTAMP,
task_sessions	session_id **INT** AUTO_INCREMENT **PRIMARY** KEY **FOREIGN** KEY (task_id) REFERENCES (task_id), **FOREIGN** KEY (user_id) REFERENCES (user_id), start_time **DATETIME,** end_time **DATETIME** DEFAULT NULL, focus_minutes **INT** DEFAULT NULL, notes **TEXT** DEFAULT NULL, created_at **DATETIME** DEFAULT CURENT_TIMESTAMP,
system_log	log_id **INT** AUTO_INCREMENT **PRIMARY** KEY, **FOREIGN** KEY (user_id) REFERENCES (user_id), action **VARCHAR(255)** NOT NULL

	timestamp **DATETIME** DEFAULT CURENT_TIMESTAMP,
schedule_collection	collection_id **INT** AUTO_INCREMENT **PRIMARY** KEY, **FOREIGN** KEY (user_id) REFERENCES (user_id), collection_name **VARCHAR(100)** NOT NULL, created_at **DATETIME** DEFAULT CURENT_TIMESTAMP, updated_at **DATETIME** DEFAULT CURENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
remember_tokens	id **INT** AUTO_INCREMENT **PRIMARY** KEY, **FOREIGN** KEY (user_id) REFERENCES (user_id), token **VARCHAR(255)** NOT NULL UNIQUE, expires_at **TIMESTAMP** NOT NULL, created_at **TIMESTAMP** DEFAULT CURENT_TIMESTAMP,
ml_traning_data	train_id **INT** AUTO_INCREMENT **PRIMARY** KEY **FOREIGN** KEY (user_id) REFERENCES (user_id), input_features **JSON** NOT NULL output_label **VARCHAR(100)** NOT NULL timestamp **DATETIME** DEFAULT CURENT_TIMESTAMP,
ml_recommendations	rec_id **INT** AUTO_INCREMENT **PRIMARY** KEY **FOREIGN** KEY (user_id) REFERENCES (user_id), model_type **ENUM('decision tree', 'knn', 'hybrid'),** generated_timetable **JSON** DEFAULT NULL, created_at **DATETIME** DEFAULT CURENT_TIMESTAMP, accuracy_score **FLOAT** DEFAULT NULL,
ml_generated_slots	slot_id **INT** AUTO_INCREMENT **PRIMARY** KEY, **FOREIGN** KEY (rec_id) REFERENCES (rec,_id), task_title **VARCHAR(200)** DEFAULT NULL, date **DATE** DEFAULT NULL, start_time **TIME** DEFAULT NULL, end_time **TIME** DEFAULT NULL, created_at **DATETIME** DEFAULT CURENT_TIMESTAMP,
mbti_questions	id **INT** AUTO_INCREMENT **PRIMARY** KEY, dimension **VARCHAR(5)** NOT NULL, question_text **VARCHAR(255)** NOT NULL, option_a_text **VARCHAR(100)** DEFAULT NULL, option_a_value **VARCHAR(1)** DEFAULT NULL, option_b_text **VARCHAR(100)** DEFAULT NULL, option_b_value **VARCHAR(1)**DEFAULT NULL, display_order **INT** DELAULT '0',
learning_style_questions	id **INT** AUTO_INCREMENT **PRIMARY** KEY question_text **VARCHAR(255)** DEFAULT NULL, visual_optopn **VARCHAR(100)** DEFAULT NULL, auditory_option **VARCHAR(100)** DEFAULT NULL, kinesthetic_option **VARCHAR(100)** DEFAULT NULL,

	display_order **INT** DELAULT ‘0’,
career_questions	id **INT** AUTO_INCREMENT **PRIMARY** KEY, question_text **VARCHAR(255)** DEFAULT NULL, technology_score **INT** DELAULT ‘0’, business_score **INT** DELAULT ‘0’, creative_score **INT** DELAULT ‘0’, science_score **INT** DELAULT ‘0’, education_score **IN** DELAULT ‘0’, social_score **INT** DELAULT ‘0’, display_score **INT** DELAULT ‘0’,

Table 7: Schemas and Table Analysis

From the above table, we can conclude some requirements as follows:

- **The users table** stores core identity and authentication data for the customer, such as their unique username, encrypted password, email address, and system role. It also manages OAuth integration and tracks whether it is the user's first login.
- **The remember_tokens table** stores unique security tokens and their expiration dates to maintain a persistent "Remember Me" session for the user across different browser visits.
- **The system_log table** stores a chronological record of specific actions performed by the user to support security auditing and system monitoring.
- **The user_settings table** stores personalized configuration data, such as the UI theme (light/dark), preferred language, and the toggle for AI-enabled scheduling features.
- **The user_profiles table** stores foundational student data, including the year of study, academic goals, and preferred learning styles (visual/auditory/kinesthetic) used to calibrate the AI engine.
- **The user_quiz_progress table** stores the tracking status (started, completed, last updated) for various onboarding surveys to ensure the user provides enough data for a personalized experience.
- **The user_mbti_profile table** stores the detailed outcome of the user's MBTI assessment, including scores for each dimension (E-I, S-N, T-F, J-P) and a descriptive summary of their personality.
- **The user_learning_style table** stores a quantitative breakdown of how the user absorbs information, recording percentages for visual, auditory, and kinesthetic learning channels.
- **The user_career_interests table** stores affinity scores for various professional sectors (Technology, Business, Social, etc.) and utilizes JSON to save their top matching career paths.
- **The user_work_style table** stores evaluation scores for professional soft skills like leadership, teamwork, and creativity, identifying the user's primary working persona.
- **The user_interests table** stores a list of specific interests, hobbies, or dislikes for each user to help the system recommend relevant supplemental academic resources.

- **The mbti_questions table** stores the standardized question set used to determine the user's personality traits across specific psychological dimensions.
- **The learning_style_questions table** stores the specific questions and multi-channel options used to identify the user's dominant information-processing style.
- **The work_style_question table** stores behavioral questions along with JSON-formatted scoring data to assess the user's professional habits and mindset.
- **The career_questions table** stores orientation questions that assign point values to different professional industries based on the user's choices.
- **The tasks table** stores individual study items created by the user, including the title, description, priority level, duration, and current status (pending/in progress/done).
- **The task_sessions table** stores data from actual focus sessions, recording the precise start and end times to measure real productivity against the user's goals.
- **The schedule_collection table** stores named containers for different sets of schedules, allowing users to switch between plans for different semesters or exam periods.
- **The user-schedule table** stores specific entries within a collection, such as class times, breaks, or self-study slots, and can be linked directly to a specific task.
- **The ml_recommendations table** stores the high-level output generated by the AI model (Decision Tree, KNN, or Hybrid), including the generated timetable in JSON format and its calculated accuracy score.
- **The ml_generated_slots table** stores the specific, granular time intervals produced by a recommendation, mapping task titles to specific calendar dates and times.
- **The ml_traning_data table** stores historical input features and actual user outcomes in JSON format, providing the dataset necessary to retrain and improve the AI model over time.

Relationship between these tables:

- One-to-One (1:1) Relationship
  - **User to UserSetting**: Each user owns exactly one settings record to manage UI preferences and the AI activation status (ai_enabled).
  - **User to UserProfile**: This 1:1 link stores core academic data, such as the year of study and personal goals, for each student.
  - **User to Psychological Profiles**: A user is directly connected to exactly one result record for each assessment type, including UserMBTIProfile, UserLearningStyle, UserCareerInterest, and UserWorkstyle

- One-to-Many (1:N) Relationship
  - **User to Tasks**: A single user can create and manage a list of multiple study tasks.
  - **User to ScheduleCollection**: Users have the right to create multiple schedule collections (e.g., Semester Schedule, Exam Review Schedule).
  - **ScheduleCollection to User-Schedule**: Each collection contains multiple specific time slots (classes, breaks, self-study) within a week.
  - **Task to TaskSession**: A single task can be executed through multiple focus sessions, allowing the system to track total actual time spent.
  - **MLRecommendation to MLGeneratedSlot**: An AI recommendation result

(generated via Decision Tree/KNN) is decomposed into several specific study time slots on the calendar.

- **User to SystemLogs & Training Data**: A user generates many activity logs and contributes multiple data points to support the retraining of the machine learning model.
- **User to RememberTokens**: A user can have multiple tokens to maintain login status across different devices or browsers.

- Many-to-Many (N:N) Relationship: In the **PlanZ** system, several Many-to-Many relationships are implemented through intermediate tables to maintain 3rd Normal Form (3NF) and data integrity
  - **User to Interest**: Handled by the user_interests table, allowing multiple users to share similar interests while each user maintains their own list.
  - **User to Survey Questions**: Managed via user_quiz_progress and result profiles. This allows the system to track which users have completed which standardized surveys.
  - **User to Specific Time Slots**: Facilitated by schedule_collection and user_schedule to map multiple users to various academic activities.

3.4 Class Diagram

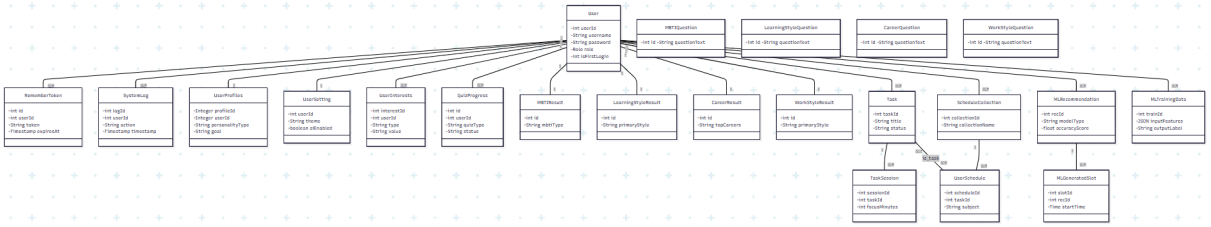

Figure 5: Class Diagram

From the above class diagram, we can summary some information as follows:

- **User:** Represents the system's users. It stores authentication credentials, including usernames, encrypted passwords, assigned roles, and first-time login status.
- **RememberToken:** Stores persistent session tokens to support "Remember Me" functionality and automatic re-authentication for returning users.
- **SystemLog:** Records a chronological history of user actions and system events for administrative monitoring, auditing, and security purposes.
- **UserProfiles:** Stores comprehensive personal data, including the user's identified personality type and their specific academic or personal goals.
- **UserSetting:** Maintains individual preference configurations, such as interface themes and the activation status of AI-assisted features.
- **UserInterests:** Stores categorized interests and preferences (types and values) to facilitate the customization of the user experience.
- **QuizProgress:** Tracks the real-time status of users as they engage with various assessments, recording the quiz type and the current stage of completion.
- **MBTIResult:** Stores the finalized results and categorized dimensions derived from the user's MBTI personality assessment.

- **LearningStyleResult:** Stores the analyzed results of the user's primary learning style survey (Visual, Auditory, or Kinesthetic).
- **CareerResult:** Stores career orientation outcomes, highlighting the most compatible professions for the user based on their career assessment data.
- **WorkStyleResult:** Stores the results of the user's work style evaluation, identifying their primary behaviors in professional environments.
- **MBTIQuestion:** A static repository of standardized questions used to evaluate and determine the user's MBTI personality type.
- **LearningStyleQuestion:** A static repository of questions designed to identify the user's dominant information-processing channels.
- **CareerQuestion:** A static repository of orientation questions used to assess interest across various professional sectors.
- **WorkStyleQuestion:** A static repository of behavioral questions used to determine a user's mindset and habits in a workplace context.
- **Task:** Represents specific assignments or objectives created by the user, including essential details such as titles, priorities, and completion status.
- **TaskSession:** Records data from individual focus intervals for each task, tracking actual duration and focus performance.
- **ScheduleCollection:** Represents organized groups or folders of schedules, allowing users to categorize plans by semester or purpose.
- **UserSchedule:** Stores specific timetable entries, including subjects and activity types, which can be directly linked to existing tasks.
- **MLRecommendation:** Stores high-level suggestions generated by machine learning models, including the model architecture used and the prediction's accuracy score.
- **MLGeneratedSlot:** Stores granular time intervals produced by the AI based on recommendations, including associations with specific tasks and start/end times.
- **MLTrainingData:** Stores the historical datasets used to refine AI models, containing input features and their corresponding target labels.

3.4 MVC Model

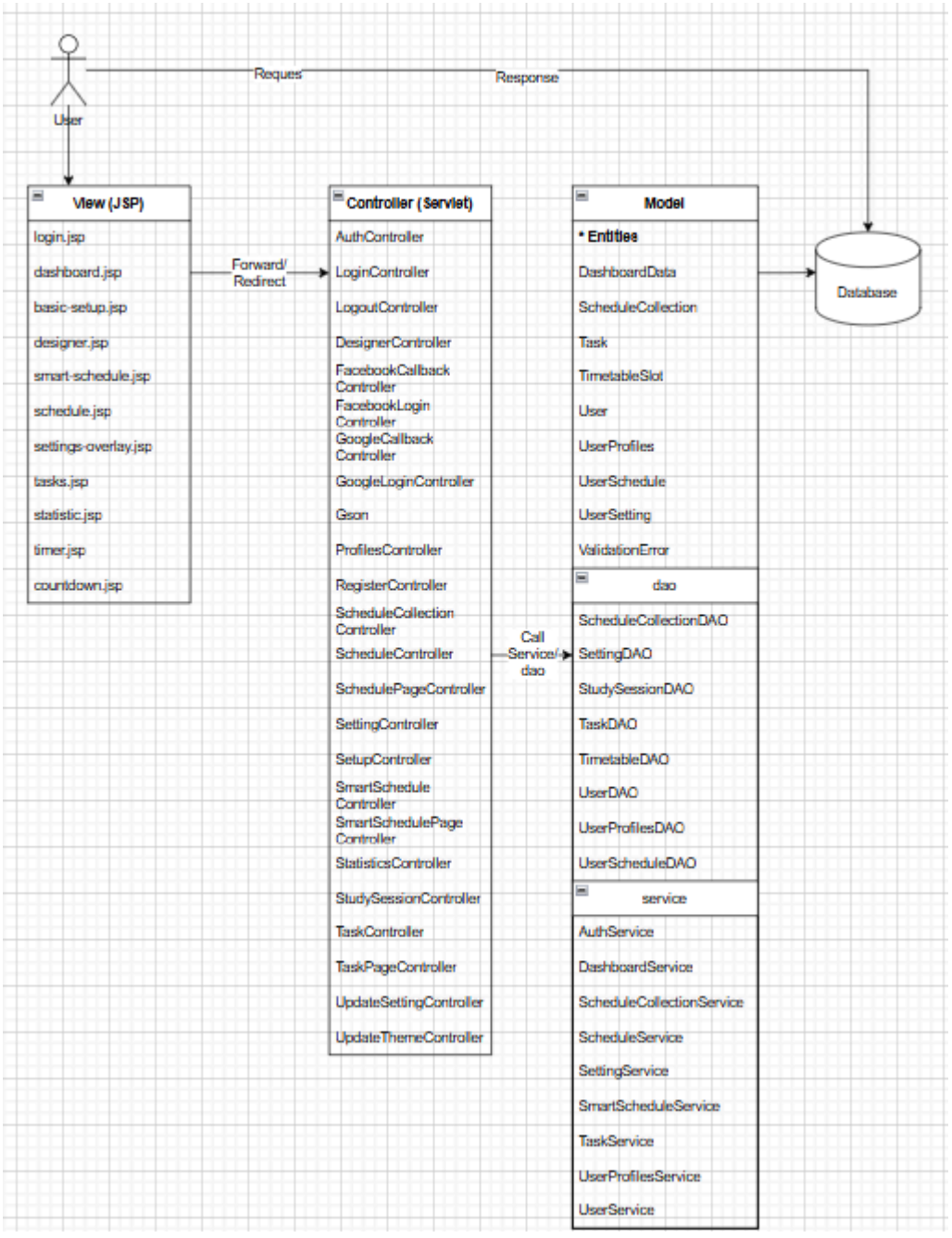

Figure 6: MVC model of Study Planning

- View (JSP - Presentation Layer)
  - **Function:** Serves as the interface that displays information and interacts directly with the user.
  - **Components:** Includes JSP pages such as login.jsp, dashboard.jsp, task.jsp,...
  - **Workflow:** Sends HTTP requests to the Controller and receives responses to render data for the user.
- Controller (Servlet - Coordination Layer)

- **Function:** Acts as the "brain" of the system, receiving requests from the View, invoking processing services, and determining the next page to display.
- **Components:** Servlets such as AuthController, TaskController, ScheduleController,...
- **Workflow:** Manages navigation logic (Forward/Redirect) between the View and the Model.

- Model (Data & Business Logic Layer)
  - **Entities:** Object classes representing data structures, such as User, Task, Schedule, UserProfiles,..
  - **DAO (Data Access Object):** Performs direct Create, Read, Update, and Delete (CRUD) operations with the database, including UserDAO, TaskDAO, ScheduleDAO,...
  - **Service:** An intermediate layer containing business logic that connects the Controller to the DAO to ensure clean, maintainable code.

3.5 Sequence Diagrams

3.5.1. Login

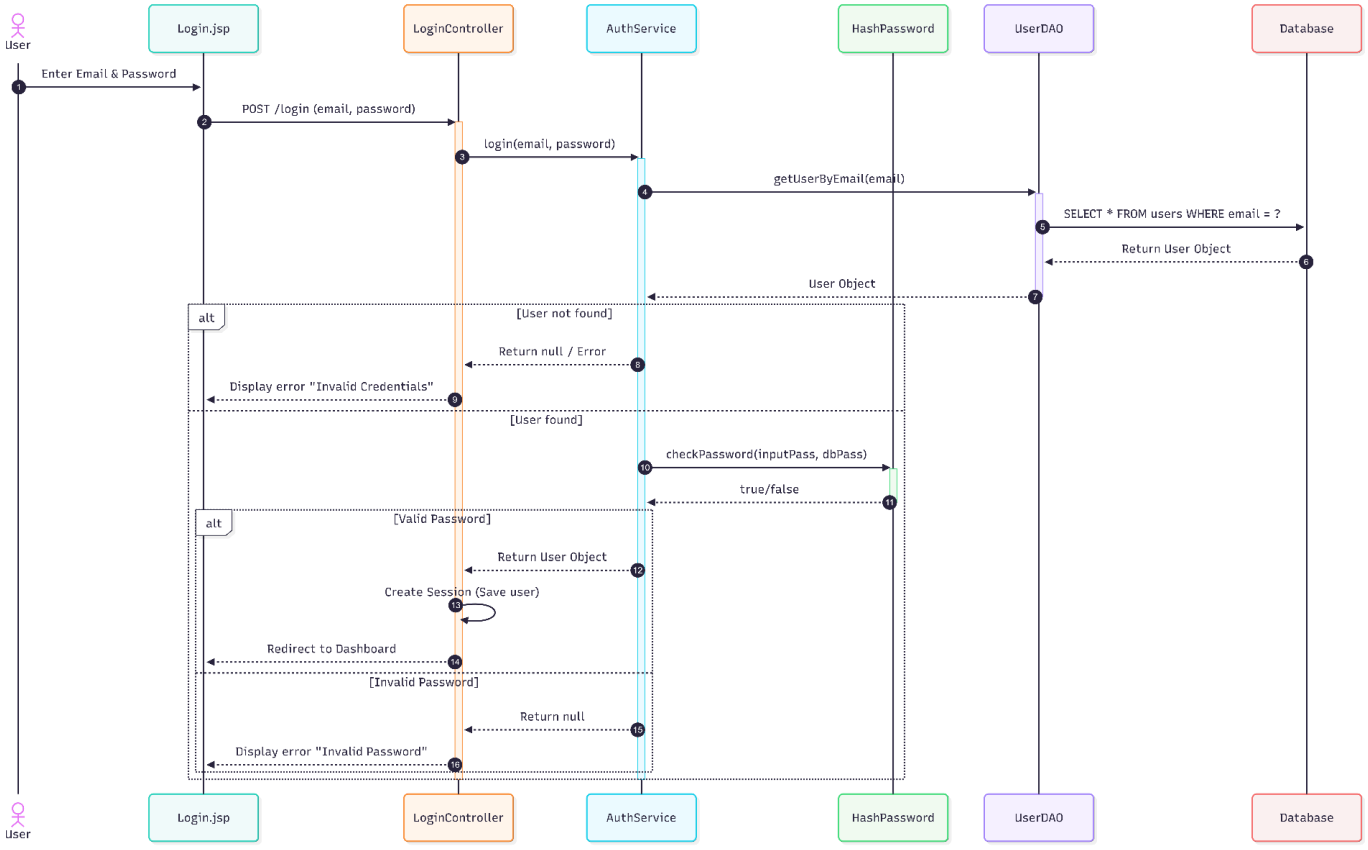

Figure 7: Sequence diagram for "Login"

- **Actors**

+ **User**: The end-user who initiates the authentication process to access the system.

- **Objects**

+ **Login.jsp (View)**: The user interface that presents the login form where the user enters their email and password.

+ **LoginController (Controller)**: The component that handles the HTTP request from the user, coordinates the logic between the View and the Service, and manages the final response (redirect or error).

+ **AuthService (Service)**: The component containing the core business logic for authentication. It acts as a bridge between the Controller and the Data layer.

+ **UserDAO (Data Access Object)**: The component responsible for communicating directly with the database to execute queries (e.g., finding a user by email).

+ **Database**: The persistent storage system that holds user account information.

+ **HashPassword (Util)**: A utility object used to verify security, specifically checking if the input password matches the hashed password stored in the database.

- **Control Flow**

+ The process begins when the **User** submits the login form. The **LoginController** receives this request and delegates the task to the **AuthService**.

+ The **AuthService** attempts to retrieve user data by asking the **UserDAO** to query the **Database**.

+ **Scenario A (User not found)**: If the database returns no results, the process halts, and the system informs the user that the information is incorrect.

+ **Scenario B (User found)**: If the user exists, the system proceeds to verify the password. The **AuthService** uses the **HashPassword** utility to compare credentials.

  - If the password is **incorrect**, the login fails, and an error is displayed.
  - If the password is **correct**, the **LoginController** creates a session to log the user in and redirects them to the Dashboard.

- **Interactions**

+ **User initiates login**: The User interacts with the *Login.jsp* form to submit their credentials (email and password).

+ **Controller calls login function**: The *LoginController* captures the POST request and calls the login() method within the *AuthService*.

+ **Database lookup**:

- The *AuthService* invokes UserDAO.getUserByEmail(email).
- The *UserDAO* executes a SELECT query against the *Database*.
- **User not found**: If the database returns null, the Service returns an error, and the Controller updates the View with an "Invalid Information" message.

+ **Credentials Validation**:

- **User found**: If the user record is retrieved, the *AuthService* calls HashPassword.checkPassword().
- **Invalid User (Wrong Password)**: If the password check returns false, the Service returns null, and the Controller displays an "Incorrect Password" error.
- **Valid User**: If the password check returns true, the Service returns the full User object. The *LoginController* saves this object into the Session and redirects the User to the Dashboard page.

3.5.2. User Register

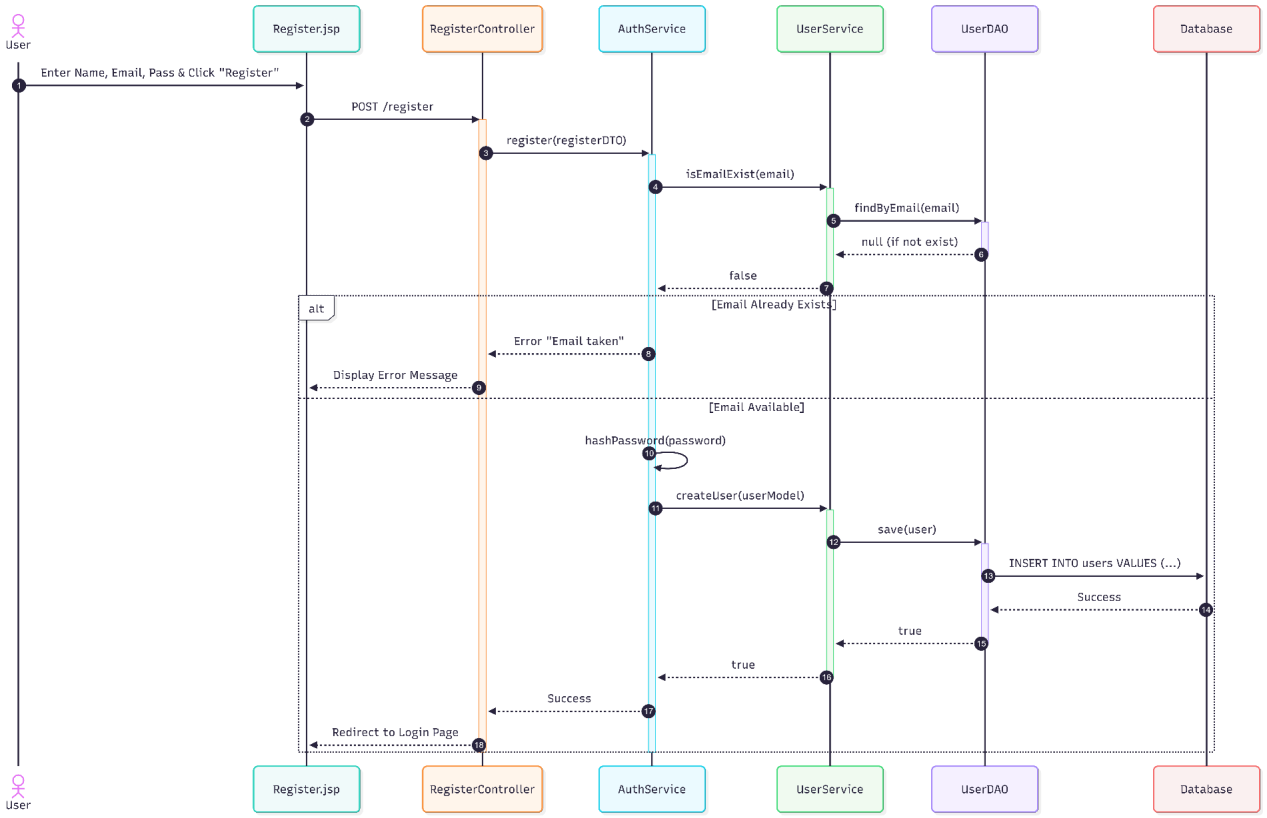

Figure 8: Sequence diagram for "User Register"

- **Actor:**

+ **User**: A new visitor who wants to create an account to start using the study planning system.

- **Object:**
    + **Register.jsp (View)**: The user interface providing the sign-up form where users input their full name, email address, and password.
    + **RegisterController**: The component responsible for handling the registration HTTP request and directing the flow to the appropriate services.
    + **AuthService**: The primary service for authentication logic, orchestrating the registration process and handling security measures like password hashing.
    + **UserService**: A domain-specific service that manages user entity logic, such as checking for duplicate accounts and creating new user records.
    + **UserDAO**: The Data Access Object that executes SQL commands to insert the new user's information into the database.
    + **Database**: The persistent storage system containing the users table.
- **Control flow:**
    + The process begins when the user submits the registration form. The **RegisterController** receives the data and delegates the processing to **AuthService**.
    + **AuthService** first validates the uniqueness of the account by asking **UserService** to check if the provided email already exists in the system.
        - **Scenario A (Duplicate)**: If the email is already taken, the process halts, and the controller reloads the view with an error message.
        - **Scenario B (Success)**: If the email is unique, the service encrypts the password. Then, it calls **UserService** to persist the new account. The **UserDAO** inserts the record into the **Database**. Finally, the system redirects the user to the Login page.
- **Interactions:**
    + **User initiates registration**:
        - The user interacts with the Register.jsp form by entering their credentials and clicking the "Register" button.
    + **Controller handles request**:
        - The RegisterController intercepts the POST request, extracts the form data (DTO), and invokes AuthService.register().
    + **Email Validation**:
        - The AuthService calls UserService.isEmailExist() to verify the email.

- UserService queries UserDAO.
- **Email Taken**: If the database returns an existing user, the service returns an error state. The controller then displays an "Email already in use" message to the user.

+ **Account Creation**:

- **Email Available**: If the email is valid, AuthService hashes the password using a security utility.
- It then calls UserService.createUser(), which triggers UserDAO.save().
- The DAO executes the INSERT SQL statement to store the new user in the **Database**.

+ **Completion**:

- Upon successful database insertion, the RegisterController sends a redirect response, moving the user to the **Login Page** to sign in with their new account.

3.5.3. Load Dashboard

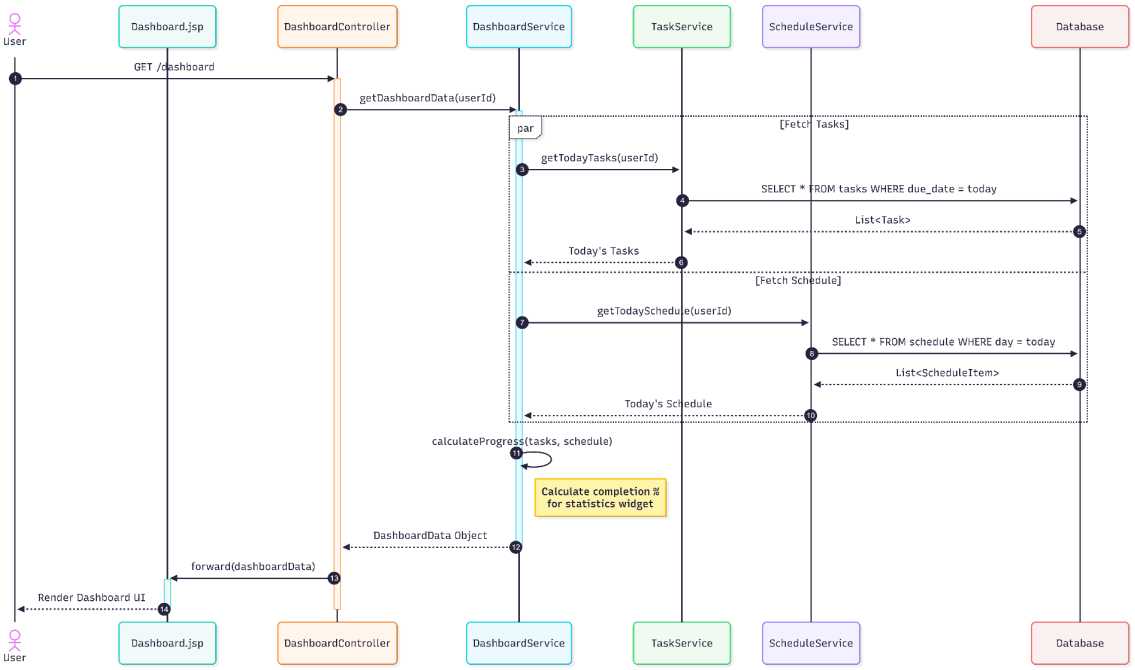

Figure 9: Sequence diagram for "Load Dashboard"

- **Actor:**

+ **User**: The logged-in user accessing their main homepage.

- **Object:**

+ **DashboardController**: The entry point that handles the request to view the dashboard.

+ **DashboardService**: A facade service that aggregates data from multiple sources (Tasks and Schedules).

+ **TaskService**: Retrieves task-related data.

+ **ScheduleService**: Retrieves time-table related data.

+ **DashboardData**: A Data Transfer Object (DTO) containing all necessary information for the view.

- **Control flow:**

+ When the user accesses the dashboard URL, **DashboardController** requests data from **DashboardService**.

+ The **DashboardService** executes parallel (or sequential) operations:

  - Calls **TaskService** to get tasks due today.
  - Calls **ScheduleService** to get the classes/events for the current day.

+ The service then processes this raw data (e.g., calculating how many tasks are "Done" vs "Pending" to show a progress bar).

+ The consolidated DashboardData object is returned to the Controller, which forwards it to **Dashboard.jsp** for rendering.

- **Interactions:**

+ **Data Aggregation**: This flow demonstrates the system's ability to combine data from different modules (tasks table and schedules table) into a single view.

+ **Visualization**: The View uses the aggregated data to render charts, lists, and progress bars immediately upon page load.

3.5.4. Create Schedule

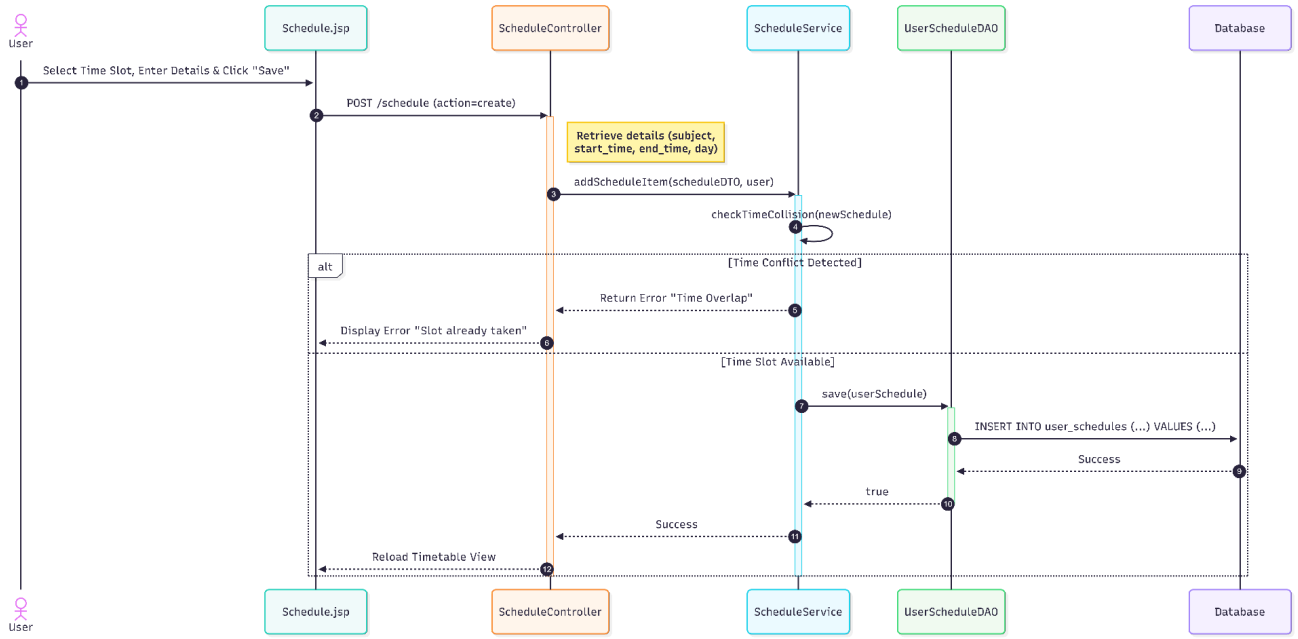

Figure 10: Sequence diagram for "Create Schedule"

- **Actor:**
  - + **User**: The student or user who manually adds a specific class, exam, or activity to a specific time slot in their weekly timetable.
- **Object:**
  - + **Schedule.jsp (View)**: The visual calendar interface where the user selects a day and time range to input activity details.
  - + **ScheduleController**: The component that handles the HTTP request for creating a new schedule item.
  - + **ScheduleService**: The core logic component responsible for ensuring the new activity does not overlap with existing commitments (Collision Detection).
  - + **UserScheduleDAO**: The Data Access Object that manages the persistence of schedule records in the database.
  - + **Database**: Stores the user_schedules table containing time slots and activity details.
- **Control flow:**
  - + The process starts when the user defines a time slot and submits the details. The **ScheduleController** receives this data.

+ The controller delegates the processing to **ScheduleService**. The service's primary role here is **Validation**: it checks if the requested time range (Start Time to End Time) conflicts with any existing activities for that specific day.

+ **Scenario A (Conflict)**: If the time slot overlaps with another activity, the service returns an error immediately, preventing the save operation. The user is notified of the conflict.

+ **Scenario B (Success)**: If the time slot is free, the service calls **UserScheduleDAO** to persist the new schedule item. Upon success, the controller refreshes the calendar view to display the new item.

- **Interactions:**

+ **User initiates creation**: The user interacts with the Schedule.jsp interface (e.g., clicking a "New Class" button or dragging on the calendar) and submits the form.

+ **Controller handles request**:

    - The ScheduleController extracts parameters like subject, day_of_week, start_time, and end_time.
    - It calls ScheduleService.addScheduleItem().

+ **Collision Detection**:

    - The ScheduleService performs a logic check (often querying existing schedules for that day) to ensure: New_Start < Existing_End AND New_End > Existing_Start.
    - **Conflict Found**: Returns an error message regarding the overlap.

+ **Database Operation**:

    - **No Conflict**: The ScheduleService invokes UserScheduleDAO.save().
    - The DAO executes the INSERT SQL statement to the **Database**.

+ **Completion**: The system redirects the user back to the updated timetable view, showing the newly blocked-out time slot.

3.6.5. Design Schedule

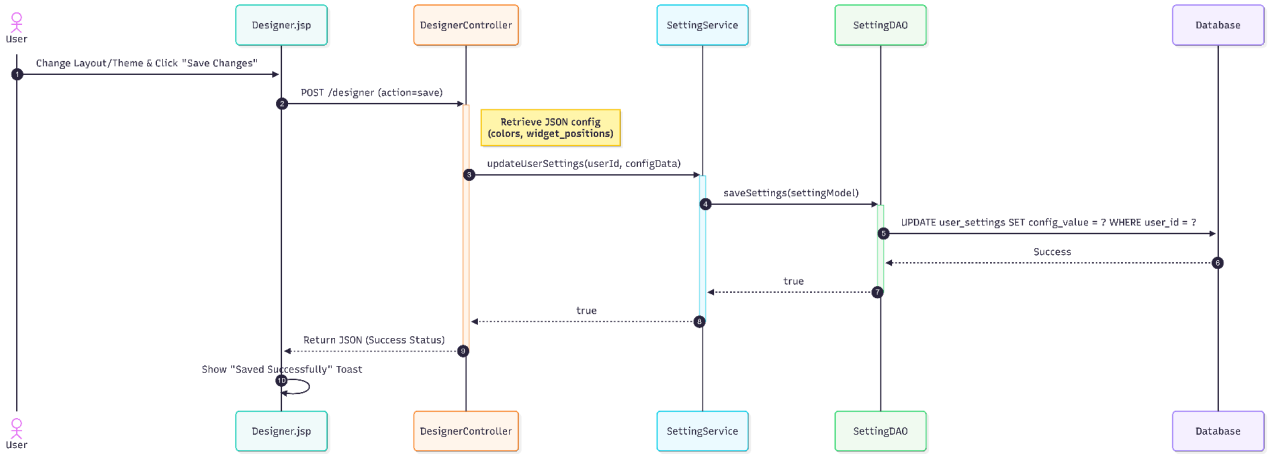

Figure 11: Sequence diagram for "Design Schedule"

- **Actor:**
  - **User**: The user who wants to personalize their dashboard, such as changing the color theme or rearranging layout widgets.
- **Object:**
  - **Designer.jsp (View)**: The customization interface where users can interact with UI elements (drag-and-drop or color pickers).
  - **DesignerController**: The controller that handles the saving of user interface preferences.
  - **SettingService**: The business logic component that processes the configuration data before saving.
  - **SettingDAO**: The Data Access Object responsible for updating the user's settings in the database.
  - **Database**: Stores the user_settings table where preference data (often in JSON format) is kept.
- **Control flow:**
  - The user makes changes on the frontend and clicks "Save". The **Designer.jsp** sends an asynchronous request (usually AJAX/Fetch) containing the new configuration to the **DesignerController**.
  - The controller parses the configuration data and passes it to the **SettingService**.
  - The service calls the **SettingDAO** to update the existing record for that user.

+ Once the **Database** confirms the update, the success signal propagates back up to the Controller, which sends a JSON response to the browser to notify the user.

- **Interactions:**

+ **User interactions**: The user modifies the visual aspects of the application on Designer.jsp.

+ **Request Handling**: The DesignerController receives the POST request with the new configuration parameters (e.g., theme color code, layout coordinates).

+ **Data Persistence**:

  - The SettingService prepares the data model.
  - The SettingDAO executes the SQL UPDATE command to persist the changes in the **Database**.

+ **Feedback**: Since this is often a background save (AJAX), the page does not reload. Instead, the Controller returns a success status, and the View displays a small notification (Toast) saying "Saved Successfully".

3.5.6. Add Task

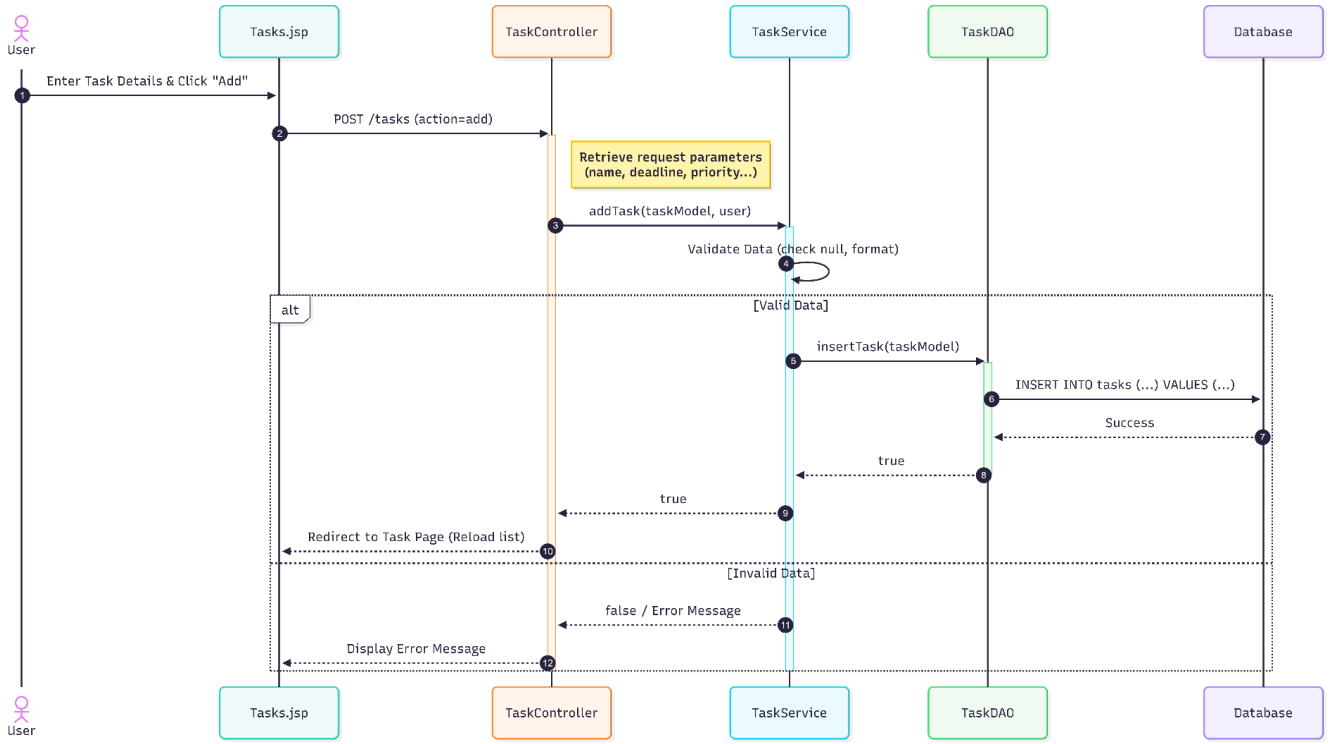

Figure 12: Sequence diagram for "Add Task"

- **Actor:**
  - + **User**: The individual who wants to add a new study task to their planner.
- **Object:**
  - + **Tasks.jsp (View)**: The interface providing the input form for task details (name, deadline, priority).
  - + **TaskController**: The component that handles the form submission request.
  - + **TaskService**: The logic component that validates the input data and coordinates with the DAO.
  - + **TaskDAO**: The data access object responsible for inserting the new task into the database.
  - + **Database**: Stores the task records.
- **Control flow:**

+ The user fills out the form and submits it. The **TaskController** intercepts the request and extracts the data.
+ The controller passes the data to the **TaskService**. The service first validates the information (e.g., ensuring the deadline is in the future).
+ If valid, the service calls **TaskDAO** to save the task. Upon success, the controller refreshes the page to show the new list.
+ If invalid, the service returns an error, and the controller displays a message to the user without saving to the database.

- **Interactions:**
  + **User inputs data**: The user enters task parameters and sends a POST request.
  + **Validation**: The TaskService checks if the inputs meet the requirements.
    - **Invalid**: Returns an error message immediately.
    - **Valid**: Proceed to database insertion.
  + **Database Insertion**: The TaskDAO executes an INSERT SQL command.
  + **Response**: The system redirects the user back to the task list view upon success.

3.6.7. Smart Schedule Generation

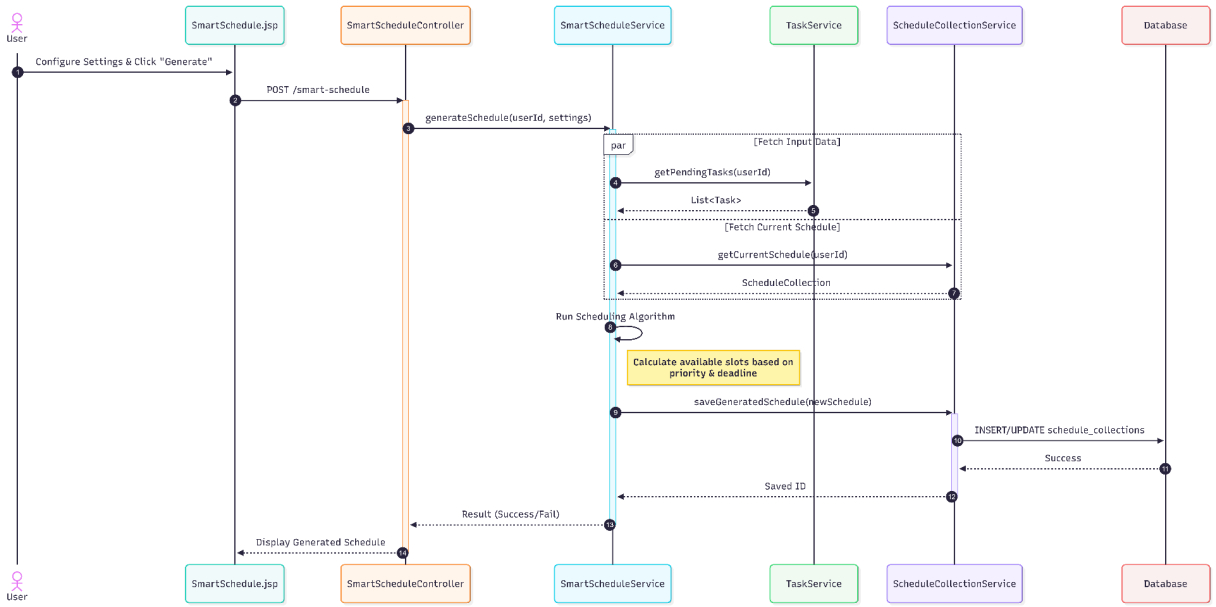

Figure 13: Sequence diagram for "Smart Schedule Generation"

- **Actor:**

    + **User**: The user requesting the system to automatically organize their study schedule.

- **Object:**

    + **SmartSchedule.jsp (View)**: The interface where users configure generation settings (e.g., study hours, intensity).

    + **SmartScheduleController**: Receives the generation request.

    + **SmartScheduleService**: Contains the complex algorithm to arrange tasks into time slots.

    + **TaskService**: Helper service to retrieve the user's unfinished tasks.

    + **ScheduleCollectionService**: Helper service to manage schedule data storage.

- **Control flow:**

    + The user initiates the process via the UI.

    + The **SmartScheduleService** gathers necessary data in parallel: it asks **TaskService** for pending tasks and **ScheduleCollectionService** for existing time constraints.

    + The service runs the scheduling algorithm to map tasks to available time slots based on priority.

    + The result is saved to the **Database** via **ScheduleCollectionService**.

    + Finally, the generated schedule is returned to the View for the user to see.

- **Interactions:**

    + **Data Aggregation**: The SmartScheduleService acts as an orchestrator, pulling data from multiple sources (TaskService and ScheduleCollectionService) before processing.

    + **Algorithm Execution**: The core logic happens within SmartScheduleService where it calculates optimal slots.

    + **Persistence**: The generated schedule is saved to the database immediately after calculation.

    + **Visualization**: The SmartScheduleController retrieves the result and renders it on SmartSchedule.jsp.

CHAPTER 6: DESIGN AND ARCHITECTURE

1. Overview of the System's Architecture

The Study Planning Web Application is designed as a web-based Decision Support System (DSS) that integrates a Java-based web application with a Python-based machine learning engine. The system follows a client–server architecture, in which the presentation layer, application layer, and intelligence layer are logically separated to ensure modularity and scalability.

Customer-side (Frontend):
The frontend serves as the user interface that enables students to interact with the system visually. It is implemented using JavaServer Pages (JSP), HTML, and CSS, and is designed to be intuitive and responsive. Through the interface, users can manage their personal profiles, define tasks, set time constraints, and view AI-generated study timetables. User actions on the frontend are transmitted to the backend via HTTP requests for processing and data retrieval.

Server-side (Backend):
The backend is responsible for handling business logic, session-based authentication, and communication with both the database and the AI engine. It is developed using Java Servlets and follows the MVC architectural pattern, incorporating Controller, Service, DAO, and Model layers. The backend aggregates user and task data, sends structured JSON requests to the Python-based AI service via RESTful APIs, and processes the returned timetable recommendations for presentation on the user interface.

Database:
MySQL is used as the relational database management system to store persistent data, including user profiles, task information, scheduling constraints, and historical activity records. Java Beans (POJOs) are used as entity models to represent database tables within the Model layer of the MVC architecture.

Machine Learning Service:
The intelligent component of the system is implemented as a standalone Python service using FastAPI. Machine learning models, such as Random Forest–based predictors, are trained to estimate productivity levels and generate personalized timetable recommendations. The AI service operates independently and communicates with the Java backend through JSON-based REST APIs.

2. Key Design Decisions and Justifications

User-centered interface design:
The user interface is designed to be simple and task-focused, allowing students to input information and interpret recommended schedules without technical expertise. A consistent layout and responsive design principles are applied to improve usability across devices.

Modular and scalable architecture:
The adoption of the MVC pattern and the separation of the machine learning engine as an independent microservice allow the system to scale and evolve without tightly coupling components.

AI-driven personalization:
Personalized timetable generation based on user profiles, task history, and time constraints enhances learning efficiency and user engagement.

Secure session management:
Session-based authentication ensures that only authorized users can access personal schedules and data, maintaining privacy and system integrity.

Technology interoperability:
Using RESTful APIs for communication between Java and Python enables seamless interoperability between different technologies and simplifies future system extension.

CHAPTER 7: DEVELOPMENT

1. Description of the Development Process

The Study Planning Web Application was developed using a lightweight Agile–Scrum approach, adapted specifically for a student academic project. Instead of applying Agile theory rigidly, the development process focused on iterative implementation, frequent refinement, and continuous integration of new features based on evolving project requirements.

The choice of an Agile-based development model was driven by the dynamic nature of the system. Throughout the development lifecycle, the project requirements changed frequently due to the inclusion of multiple functional components such as user authentication, survey-based data collection, dashboard visualization, and AI-driven timetable recommendation. Under such conditions, a traditional Waterfall model would have limited flexibility and delayed feedback, whereas an iterative Agile approach enabled continuous improvement and rapid adaptation.

The system was developed incrementally through a sequence of short development cycles, referred to as sprints. Each sprint focused on delivering a functional subset of the system that could be implemented, tested, and demonstrated independently. The main development phases were organized as follows:

- **Sprint 1:** Implementation of user registration and login functionalities
- **Sprint 2:** Development of basic user setup and survey/quiz modules
- **Sprint 3:** Construction of the personal dashboard interface
- **Sprint 4:** Integration of the AI-based timetable recommendation service
- **Sprint 5:** User interface refinement and performance optimization

Each sprint followed a simplified workflow including requirement identification, implementation, testing, and review. This incremental process ensured that the system remained functional at all times while allowing errors and design issues to be detected early and corrected efficiently.

2. User-Oriented Development

Although the project did not strictly follow a formal user-centered design framework, the development process consistently maintained a user-oriented perspective. System features were designed and refined based on how students would interact with the application in real academic contexts.

User requirements were initially defined in the form of simple user stories, such as the ability to log in using third-party authentication, view a personalized study timetable, or receive scheduling recommendations based on personal constraints. These user stories guided the prioritization of development tasks across different sprints.

User experience considerations played a central role during implementation. Interface layouts, navigation flow, and feature accessibility were continuously reviewed to ensure that users could complete tasks intuitively without technical knowledge. Feedback obtained during internal testing and demonstrations was used to adjust interface complexity, improve clarity, and enhance overall usability.

In addition, the integration of the AI recommendation component required iterative evaluation from a user perspective. Timetable outputs were reviewed to assess whether the suggested schedules were practical, understandable, and aligned with user expectations. This evaluation process allowed the system to gradually improve both functional accuracy and perceived usefulness.

3. Technologies

This section outlines the technologies applied during the development phase, focusing on their practical role in implementation rather than architectural design, which has been discussed in Chapter 6.

Programming Languages

- **Java:** Backend development, business logic implementation, session management
- **Python:** Machine learning model development, data processing, timetable recommendation
- **HTML, CSS, JavaScript:** Frontend structure, styling, and client-side interactions

Frameworks and Libraries

- **Java Servlets (Jakarta EE):** Handle HTTP requests and responses, MVC controller layer

- **JSP & JSTL:** Dynamic web page rendering, separation of presentation and logic
- **FastAPI:** Expose machine learning services via RESTful APIs
- **scikit-learn:** Machine learning model training and inference
- **pandas & NumPy:** Data preprocessing, numerical computation, feature handling

Tools Used

- **NetBeans:** Java backend development, Servlet configuration, MVC implementation
- **Visual Studio Code:** Frontend development, Python machine learning implementation
- **MySQL Workbench:** Database management, schema design, query testing
- **Git & GitHub:** Version control, branch-based collaboration, feature integration
- **Manual Testing:** Functional testing, form validation, workflow verification
- **System Logging:** Runtime error detection, data flow validation, debugging support

CHAPTER 8: TESTING AND QUALITY ASSURANCE

1. Testing Methodologies Employed

Testing activities in the Study Planning Web Application were conducted to ensure functional correctness, system stability, and the reliability of AI-generated study planning recommendations. Given the academic nature and limited deployment scope of the project, the testing strategy mainly relied on manual testing while applying multiple testing levels to evaluate different components of the system. The emphasis was placed on validating expected system behavior and ensuring consistent data flow across the application rather than executing extensive automated test suites.

1.1. Unit Testing

Unit testing was conducted to verify individual functional components of the system before integration with other modules. This testing level ensured that each component operated correctly in isolation, thereby reducing the risk of error propagation during later development stages.

In this project, unit testing primarily focused on backend logic and AI processing functions. Backend testing covered Java Servlets acting as controllers, the service layer responsible for business logic, and the DAO layer managing interactions with the MySQL database. For the AI component, testing targeted data preprocessing modules for survey and task inputs, as well as timetable generation functions within the Python-based machine learning service.

Due to time and scope constraints, unit testing was performed mainly through manual methods. Individual methods and API endpoints were tested by manually sending requests, observing system responses, and comparing them with expected results.

The main objectives of unit testing in this project were to:

- verify the correctness of core backend functionalities such as authentication and data handling;
- ensure consistency between input data and AI-generated outputs;
- detect errors early prior to module integration;
- reduce debugging complexity and support maintainability;
- confirm the independent and stable operation of each module.

Typical unit testing activities included validating login authentication logic, checking survey data preprocessing before transmission to the AI service, and verifying the structure of JSON responses returned by the AI recommendation API.

1.2. Integration Testing

After unit testing was completed, integration testing was conducted to assess the interaction between major system components. This stage focused on verifying that individual modules functioned correctly when combined within the complete system.

Integration testing evaluated interactions between the JSP-based frontend and the Java Servlet backend, communication between the Java backend and the Python-based AI service through RESTful APIs, and data exchange between backend services and the MySQL database. Particular attention was given to ensuring that data flowed consistently across components, that API requests and responses were processed accurately, and that database read and write operations were executed correctly.

In addition, integration testing verified that AI-generated timetables were accurately retrieved and displayed on the user dashboard. Typical integration scenarios included a user submitting survey information, the backend processing and forwarding this data to the AI service, receiving a generated timetable, and presenting the results on the dashboard interface. Session management mechanisms were also tested to ensure that users who successfully logged in could access protected dashboard resources. Through this process, integration testing helped identify issues that only emerged when system components began operating together.

1.3. User Acceptance Testing (UAT)

User Acceptance Testing (UAT) was conducted during the final phase of the project to evaluate how well the system met the needs and expectations of real users. Participants involved in UAT included members of the development team as well as friends and student users who tested the application in realistic usage conditions.

The evaluation focused on assessing whether the AI-generated timetables were reasonable and aligned with users' actual study schedules, whether the dashboard interface was intuitive and easy to use, and whether the overall user flow—from login to survey completion and dashboard viewing—was clear and understandable. Feedback was also collected to determine whether the AI recommendations met user expectations.

UAT provided valuable insights that helped reduce deployment risks, improve user experience, and refine AI recommendation behavior based on real-world feedback.

2. Test Results and Bug Tracking

2.1. Test Results

Testing covered core functional workflows such as authentication, survey completion, dashboard interaction, and AI-based recommendation generation, as well as selected boundary conditions including incomplete survey inputs and unauthorized dashboard access. The system demonstrated stable behavior under normal usage conditions, with several minor interface and logic issues identified and resolved during early testing phases.

2.2. Bug Tracking

Bug tracking was performed through manual testing, observation of system logs, and feedback from test users. Identified issues were categorized into user interface bugs, logic-related errors, and inconsistencies in AI-generated outputs. Bugs were fixed incrementally, and affected workflows were re-tested to ensure system stability.

2.3. Regression Testing

Regression testing was conducted manually after bug fixes and feature updates to confirm that previously implemented functionalities continued to operate correctly. Core workflows such as login, survey submission, dashboard access, and AI recommendation generation were re-evaluated to prevent unintended side effects.

3. Quality Assurance Measures Taken to Ensure Software Reliability

3.1. Code Review

To maintain code quality, internal code reviews were conducted within the development team prior to integration. These reviews focused on verifying logical correctness, maintaining consistent coding style, and improving clarity and maintainability.

3.2. Testing Strategy

The testing strategy prioritized manual testing due to the moderate project scale, the need for flexibility during development, and limitations in available time and resources. Unit testing, integration testing, and user acceptance testing were combined to provide sufficient coverage across different system components.

3.3. Security and Data Integrity

Quality assurance measures also included basic security and data integrity checks. These measures focused on session management, access control for protected dashboard resources, and validation of user input data. The objective was to safeguard users' personal study data and prevent unauthorized access.

3.4. Performance and Stability Testing

Basic performance and stability testing was conducted to evaluate system responsiveness. Testing focused on response time when invoking the AI service and data loading time on the

dashboard interface. Results showed stable system behavior and no crashes within the tested data range.

CHAPTER 9: USER INTERFACE & API

1. User Interface

1.1. "Sign up" Page

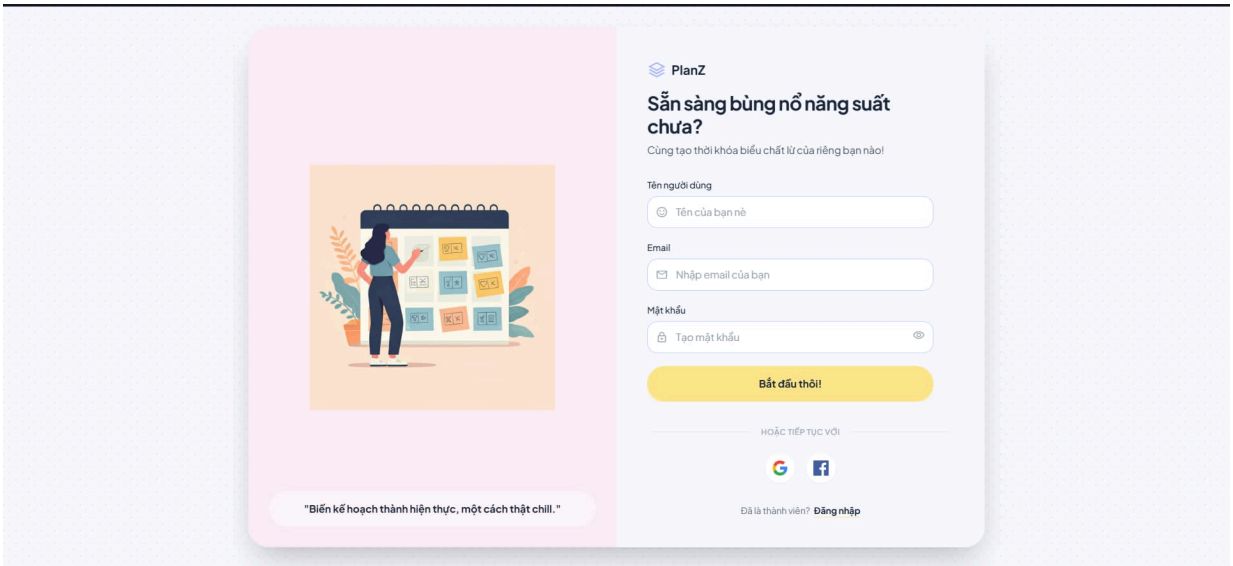

Figure 14: Illustrator of "Sign up" page

An illustrative image of a student interacting with a planning board is used to help users clearly visualize the core value of the application. The message *"Turn plans into reality, in a truly chill way"* is subtly placed to reduce learning pressure and create a relaxed user experience. The Study Planning logo and slogan are prominently displayed on the registration form on the right side, which includes input fields such as Email, Username, and Password. In addition, for greater convenience during the registration and usage process, social sign-up options such as Google and Facebook are provided directly below the form.

To register an account on Study Planning, users need to complete the form on the right by entering their email address, username, and password. After that, they can click the "Sign up" button to complete the registration process. Alternatively, users may choose to sign up using their Facebook or Google accounts via the icons located below the form. If the user already has an account, they can select "Sign in" to access the system.

1.2. “Sign in” Page

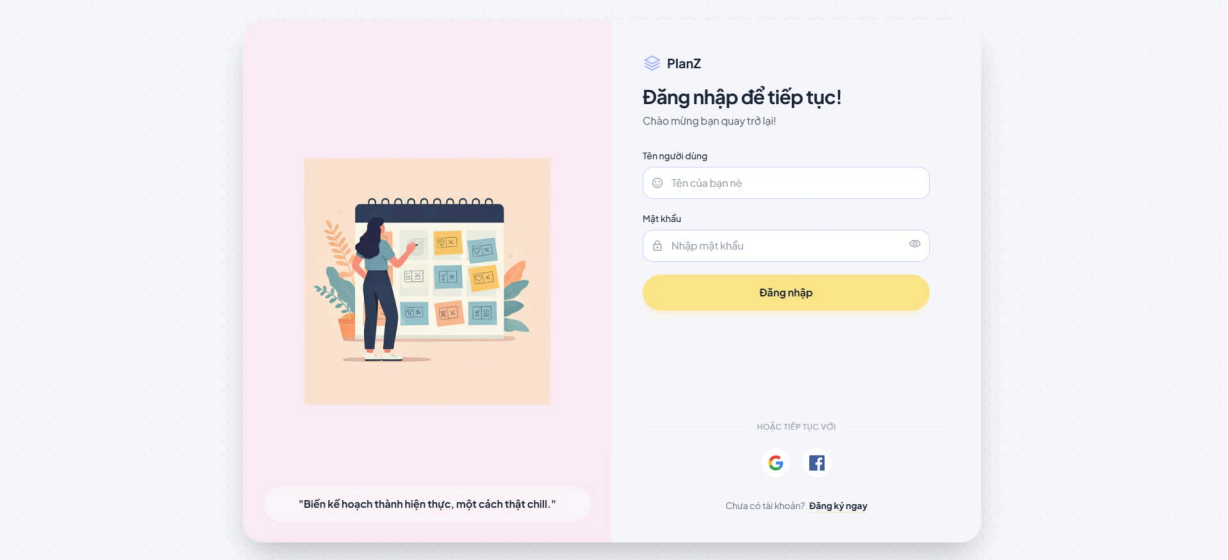

Figure 15: Illustrator of “Sign in” Page

The Study Planning login page is designed with a modern and well-balanced layout, emphasizing a user-centered approach and strong brand identity. To enhance usability, the authentication form is simplified to optimize performance, requiring only a username and password, allowing returning users to access their accounts quickly. The password field includes a show/hide password icon to ensure a balance between convenience and security. In addition, quick authentication options via Google and Facebook are provided directly below the main form, enabling users to join or access the platform without the need to remember additional login credentials. To facilitate a smooth transition for new users who need to create an account, a clear message — *“Don’t have an account? Sign up now”* — is placed at the bottom of the page.

To access the system, users simply need to enter their username and password, then click the “Sign In” button. Furthermore, new users can easily join the platform by selecting “Sign up now”, or by using the Google and Facebook buttons for quick social login.

1.3. “Basic setup” Page

Figure 16: Illustrator of “Basic-setup” Page (First Page)

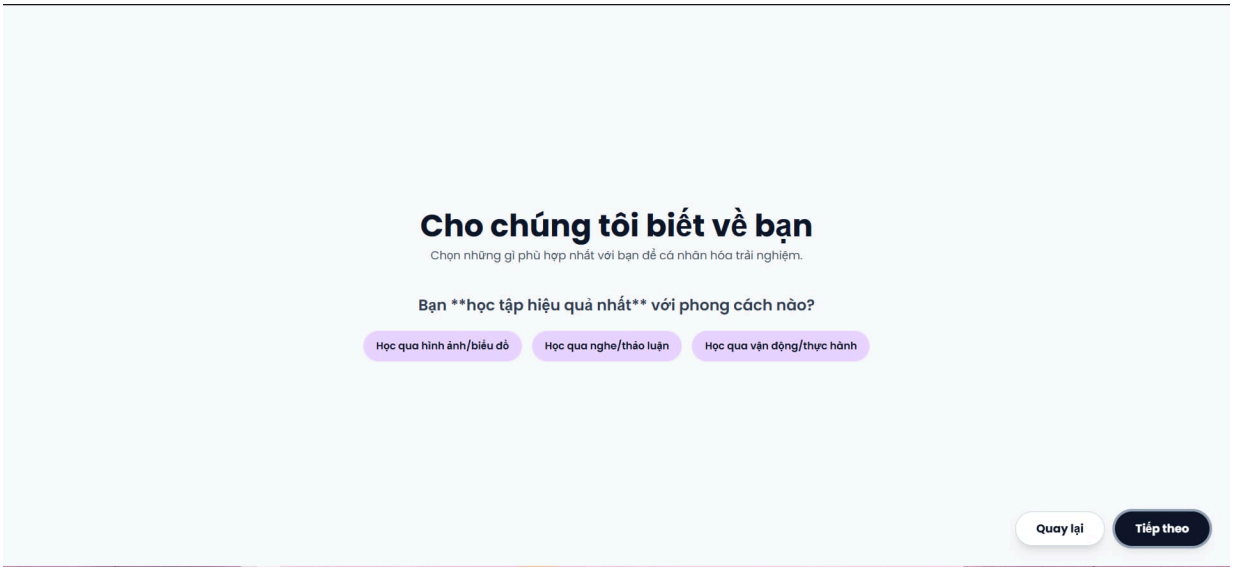

Figure 17: Illustrator of “Basic-setup” Page (Second Page)

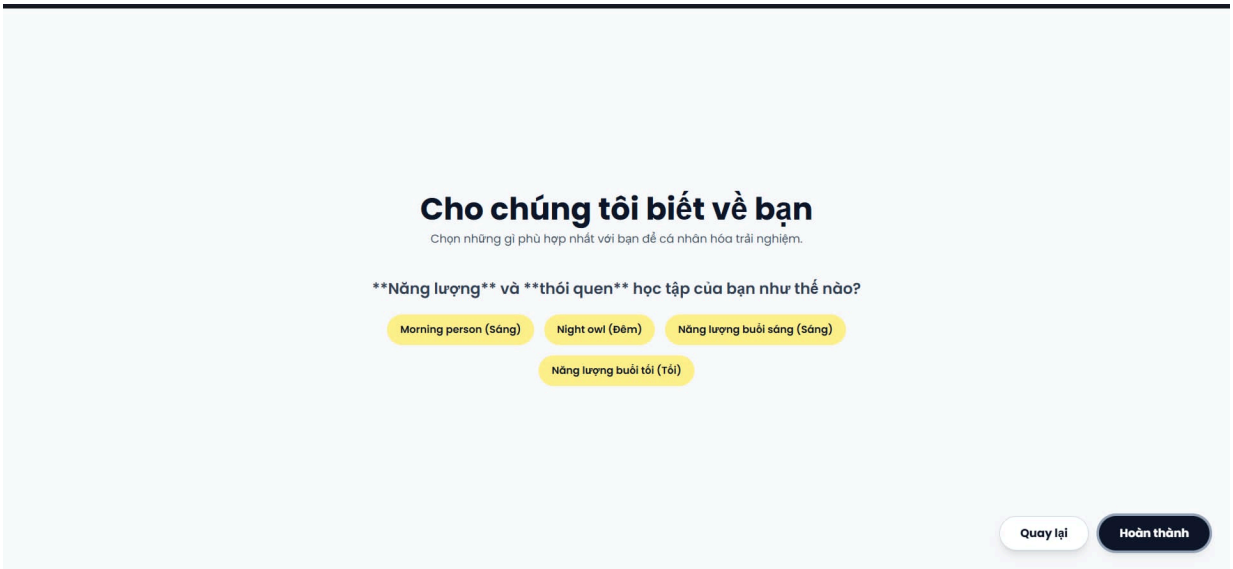

Figure 18: Illustrator of "Basic-setup" Page (Last Page)

This interface sequence is displayed after a user completes the registration process for the first time. The first page focuses on defining the user's short-term learning goals, where users select options such as improving GPA, exam preparation, certificate study, or interview preparation. The second page allows users to choose their preferred learning style, including visual learning, listening and discussion-based learning, or hands-on and practical learning. The final page in this sequence collects information about the user's energy levels and study habits, specifically whether the user is more productive in the morning or evening.

This multi-step onboarding process plays a crucial role in collecting input data for the machine learning model, which is used to generate an optimized study timetable tailored to each individual user.

To complete this survey, users select the options that best match their preferences on each page, then click the "Next" button located at the bottom of the page to proceed to the next step. If users wish to modify their previous selections, they can click the "Back" button located at the bottom left of the page, next to the "Next" or "Finish" button. Once all steps are completed, users can finalize the process by clicking the "Finish" button on the last page.

1.4. "Statistic" Page

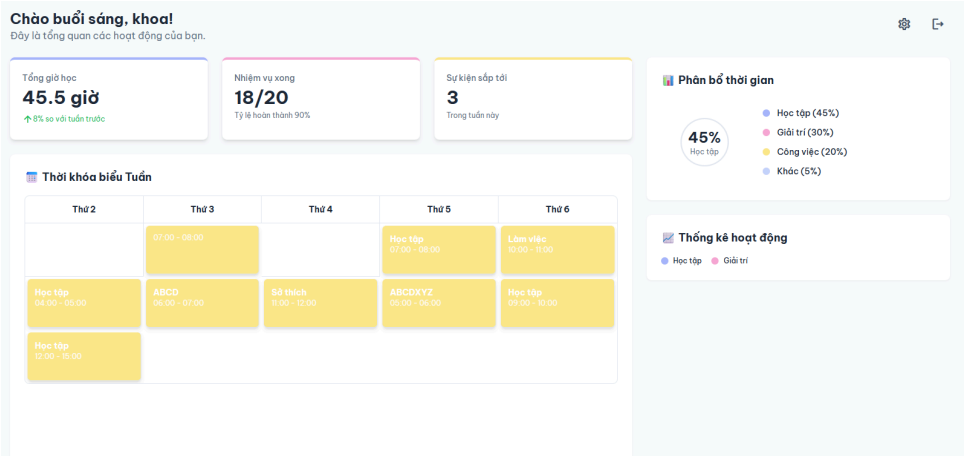

Figure 19: Illustrator of "Statistic" Page

The Dashboard interface is designed as a personal control center, helping users (students) comprehensively track their study progress and manage time intuitively. A personalized greeting, "Good morning, khoa!" (Chào buổi sáng, khoa!), along with a general status line, creates a friendly atmosphere and encourages users to start their day.

The system provides key metrics through three quick-stat cards at the top:

- **Total Study Hours:** Displays accumulated hours (45.5 hours) along with a growth indicator compared to the previous week (up 8%).
- **Tasks Completed:** Tracks the success rate of goals (18/20 tasks, corresponding to 90%).
- **Upcoming Events:** Reminds the user of important milestones scheduled for the current week.

The lower section of the page is divided into two main columns:

- **Left Column - Weekly Timetable:** Displays the schedule from Monday to Friday with visual yellow blocks, allowing users to easily grasp study and work time slots throughout the day.
- **Right Column - Allocation and Statistics:** * The **Time Allocation** donut chart helps users identify the balance between Study (45%), Entertainment (30%), Work (20%), and other activities (5%).
    - The **Activity Statistics** area allows for reviewing fluctuations in activities over time to make appropriate adjustments.

Settings and logout icons are subtly placed in the upper right corner, ensuring convenience without distracting from the user's focused workspace.

1.5. “Schedule Creation” Page

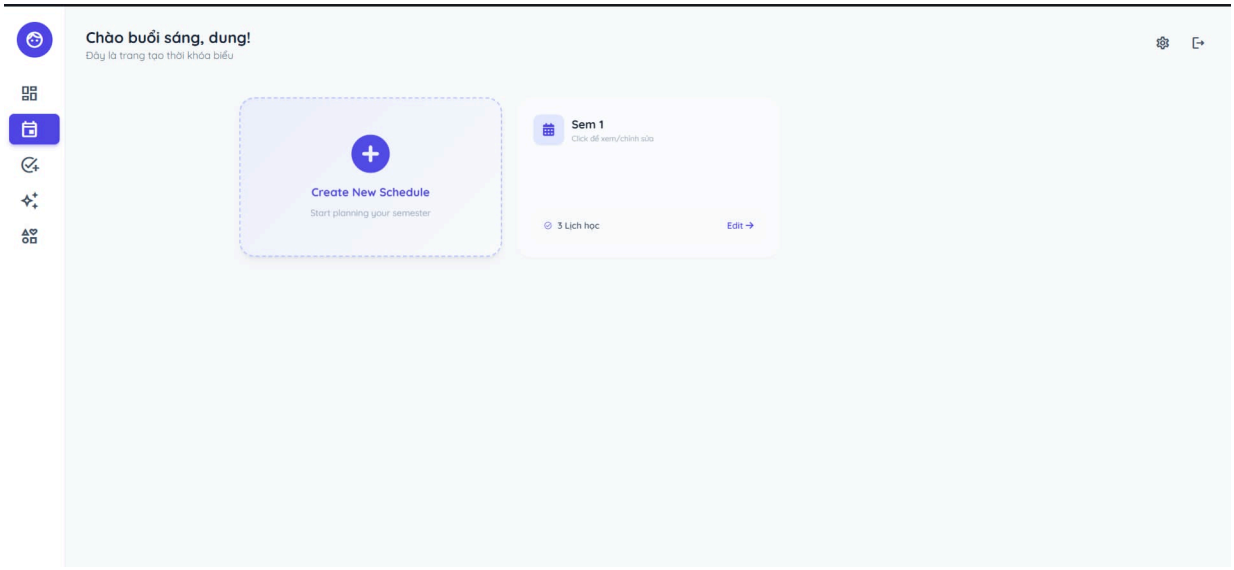

Figure 20: Illustrator of “Schedule Creation” Page

This interface allows users to organize and manage multiple study plans or semester schedules in a clean, card-based format. It emphasizes simplicity, enabling students to switch between different academic contexts effortlessly.

Key features of this page include:

- **Create New Schedule Card:** A prominent action card featuring a large plus (+) icon and the call-to-action "Start planning your semester," encouraging users to initiate a new planning cycle.
- **Existing Schedule Cards:** Individual cards representing saved plans, such as "Lịch học mới 2" and "Lịch học mới".
- **Schedule Details:** Each plan card displays the number of specific entries (e.g., 6 schedules or 3 schedules) contained within that plan.
- **Quick Navigation:** Each card includes an "Edit" button with a directional arrow, providing a direct path to view or modify the details of that specific schedule.

1.6. “Schedule” Page

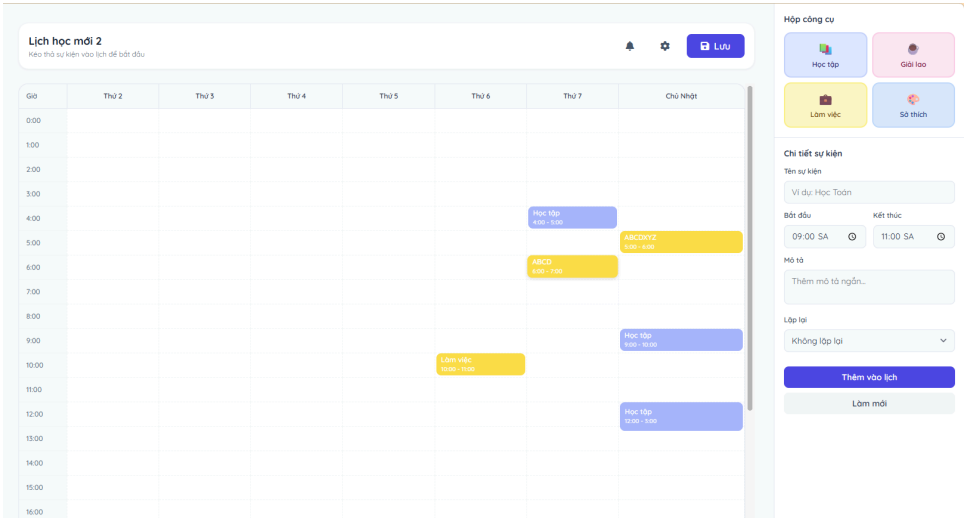

Figure 21: Illustrator of “Schedule” Page

The Detailed Schedule Editing page provides a comprehensive workspace for users to build their weekly routine with precision. The interface is split into a main calendar view and a sidebar for event details.

Key components of this interface include:

- **Main Planning Grid:** A detailed 24-hour vertical timetable spanning from Monday to Sunday, where users can drag and drop activities.
- **Toolbox (Hộp công cụ):** Four color-coded categories—Study (Học tập), Relaxation (Giải lao), Work (Làm việc), and Hobbies (Sở thích)—allow for quick classification of events.
- **Event Details Form (Chi tiết sự kiện):** A sidebar form used to input specific information for a selected slot, including event name, start/end times, and a description.
- **Recurrence and Action Buttons:** Users can set events to repeat and use the "Add to schedule" (Thêm vào lịch) or "Reset" (Làm mới) buttons to manage their entries.
- **Save and Control:** A "Save" (Lưu) button at the top right ensures all changes are synchronized with the user's dashboard.

1.7. "Task Addition" Page

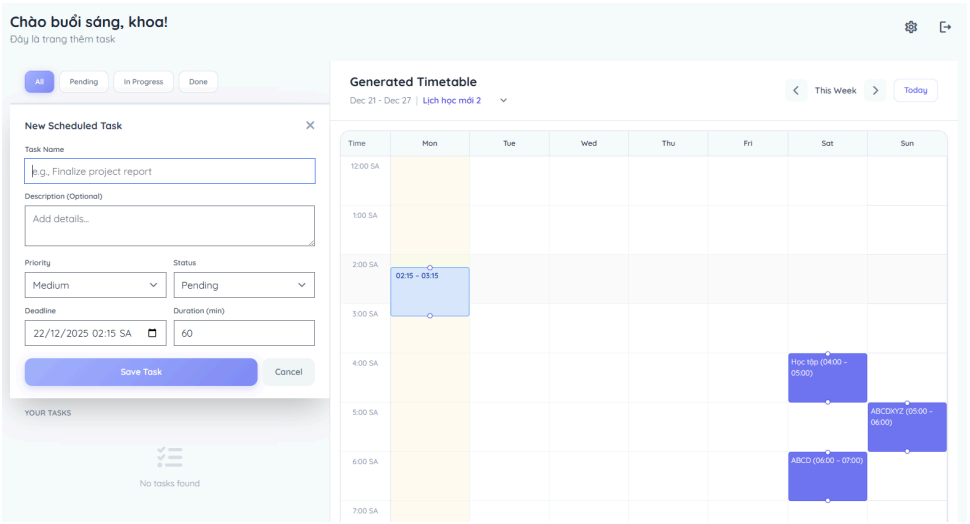

Figure 22: Illustrator of "Task Addition" Page

The Task Addition page allows users to integrate specific, actionable tasks into their generated timetable. It combines a task creation form with a weekly calendar view to ensure deadlines are visible and manageable.

Key features of this page include:

- **New Scheduled Task Form:** A dedicated modal on the left where users enter the Task Name, Description, and set key parameters such as Priority (e.g., Medium) and Status (e.g., Pending).
- **Time Management:** Users can specify a precise Deadline date and time, as well as the expected Duration in minutes, before selecting "Save Task".
- **Generated Timetable:** A central grid that displays the week's schedule, showing how new tasks fit between existing blocks like "Học tập" and "ABCDXYZ".
- **Task Filtering:** Quick-filter buttons at the top left allow users to view tasks by status: "All," "Pending," "In Progress," or "Done".

1.8. "Smart Schedule Creation" Page

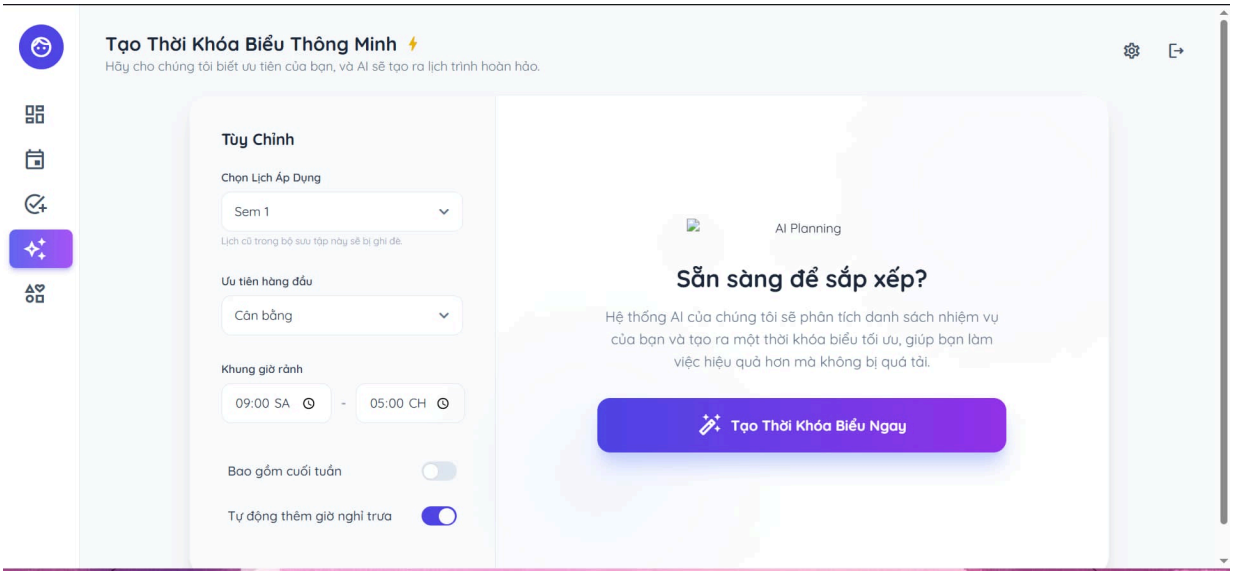

Figure 23: Illustrator of "Smart Schedule Creation" Page

The Smart Schedule page serves as the core interface where users interact directly with input configurations to trigger the system's decision-support engine. The UI is designed with a professional and intuitive layout, divided into two distinct functional zones:

- Customization Control Panel (Left Side):
  - **Schedule Selection:** Allows users to select a specific semester or collection (e.g., "Sem 1") to ensure data is organized systematically.
  - **Priority Settings:** Features a dropdown menu (e.g., "Balanced") that allows users to guide the AI on how to arrange the density of their workload.
  - **Time Management:** Integrates a free-time selector and smart toggles, such as "Include weekends" or "Auto-add lunch breaks". This reflects the core value of creating a healthy and "chill" learning path.

- Confirmation and Action Area (Right Side):
  - **Guidance Messaging:** Use the prompt "Ready to organize?" along with a brief description of the AI analysis process to build trust and transparency with the user.
  - **Primary Call-to-Action (CTA):** The **"Create Timetable Now"** button utilizes a prominent purple gradient effect to direct users toward the most important action on the page.

1.9. "Profile" Page

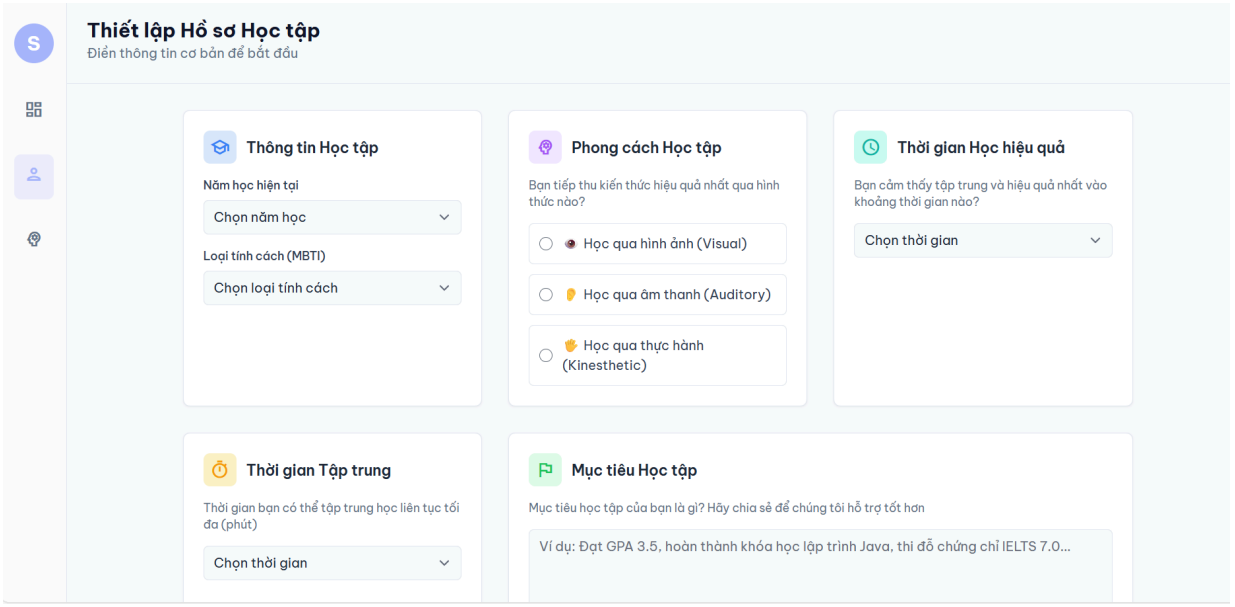

Figure 24: Illustrator of "Profile" Page

The Study Profile Setup (Thiết lập Hồ sơ Học tập) page is a comprehensive data-collection interface designed to refine the system's personalization capabilities. The layout follows a modular grid system, organizing complex information into digestible, white-space-rich cards to maintain the platform's "chill" aesthetic.

- Header and Navigation:
  - The page is titled **"Thiết lập Hồ sơ Học tập"** with a sub-header encouraging users to provide basic information to begin.
  - A persistent sidebar allows users to navigate between the main dashboard and specific profile settings.

- Data Collection Modules (Cards):
  - **Academic Information (Thông tin Học tập):** Features dropdown menus for selecting the current academic year and MBTI personality type to better understand user behavior.
  - **Learning Style (Phong cách Học tập):** Uses radio buttons for users to identify as Visual, Auditory, or Kinesthetic learners.
  - **Efficiency Survey (Thời gian Học hiệu quả):** A dropdown for users to select their peak productivity hours.
  - **Focus Capacity (Thời gian Tập trung):** Allows users to define their maximum continuous study duration in minutes.
  - **Learning Goals (Mục tiêu Học tập):** A large text area for qualitative input, such as achieving a specific GPA or finishing a professional certification.

1.10. "Work Timer" Page

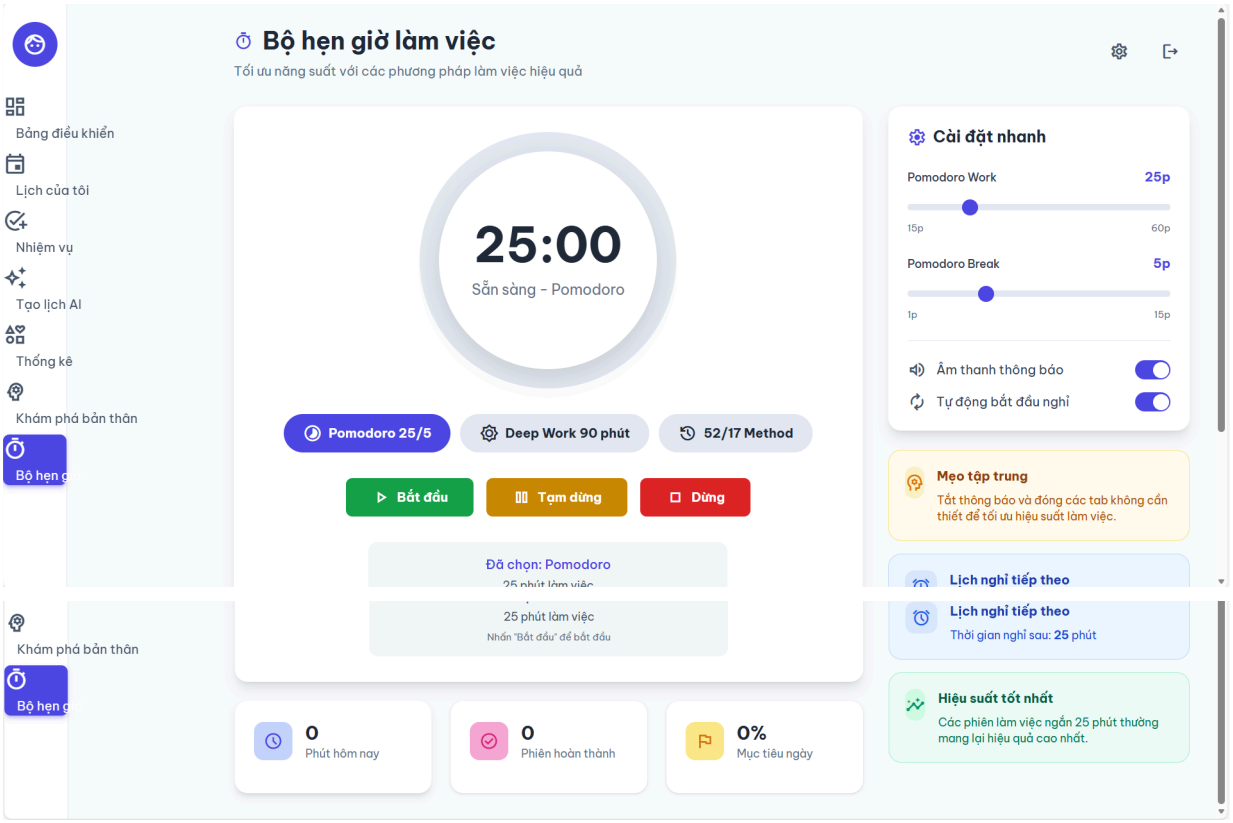

Figure 25: Illustrator of "Work Timer" Page

The **Work Timer** interface is designed to optimize user focus and productivity through structured work methods. The design follows a modern, clean aesthetic using a soft blue and white color palette, ensuring a distraction-free environment for users.

- Main Timer Dashboard:
  - **Countdown Display:** A prominent circular timer showing the remaining time (defaulting to **25:00** for Pomodoro).
  - **Method Selectors:** Quick-toggle buttons to switch between three scientifically backed methods: **Pomodoro 25/5**, **Deep Work 90 minutes**, and the **52/17 Method**.
  - **Control Buttons:** Large, color-coded action buttons for **Start** (Green), **Pause** (Orange), and **Stop** (Red) to ensure intuitive navigation.

- Quick Settings (Cài đặt nhanh):
  - **Adjustment Sliders:** Users can customize their "Work" and "Break" durations using interactive sliders (e.g., Work from 15p to 60p).
  - **System Toggles:** Functional switches for "Notification Sounds" and "Auto-start Break" for personalized automation.

- Statistics & Progress Tracking:

- **Summary Cards:** Located at the bottom, these cards track daily performance: "Total Minutes Today," "Completed Sessions," and "Daily Goal Progress (%)".

- Productivity Insights (Side Panel):
  - **Tips & Recommendations:** Informational cards providing focus tips (e.g., closing unnecessary tabs) and next-scheduled break reminders.
  - **Performance Analysis:** A smart-insight card highlighting that "25-minute sessions bring the highest efficiency" for the user.

2. Application Programming Interface (API)

Authentication & User API Endpoints	/auth/signup	Register a new account.
	/auth/signin	Log in to the system.
	/auth/logout	Terminate the session and log out.
	/user/profile	Retrieve and update personal profile information.
	/user/setting	Update user preferences (Theme, free time slots).
Task & Schedule API Endpoints	/task/create	Adds a new study task to the system.
	/tasks	Retrieves the user's list of created tasks.
	/task/update	Edit task information or status.
	/task/delete	Remove a task from the list.
	/schedule/current	Retrieves the user's active timetable.
	/schedule/save	Saves fixed academic or personal time slots.
Survey & Quiz API Endpoints	/quiz/questions	Fetches questions by type (MBTI, Career, etc.).
	/quiz/submit	Submits answers, calculates scores, and saves results.
	/quiz/status	Checks the completion progress of various surveys.

Work Timer API Endpoints	/timer/sesion	Starts/Ends a work session to log productivity data.
AI Integration API Endpoints	/ai/generate	Sends data to Flask to generate an AI schedule.
	/ai/recommend	Retrieves the time slots suggested by the AI.

Table 8: API of Study Planning

CHAPTER 10: PROJECT EVALUATION

1. Evaluation Criteria Used to Measure Project Success

The evaluation of the Study Planning Web Application was conducted using a multi-dimensional approach that reflects both technical and academic project standards. Rather than focusing solely on final completion status, the project was assessed based on several key criteria, including schedule performance, quality of implemented features, resource constraints, and stakeholder satisfaction. This evaluation framework is appropriate for an academic software project, where learning outcomes, system feasibility, and functional validity are considered as important as full feature completion.

Given the limited timeframe and scope of a final-year academic project, the evaluation emphasizes the extent to which core objectives were achieved, the stability of implemented functionalities, and the system's potential for future enhancement.

Schedule Performance: Schedule performance was evaluated by comparing the planned project timeline with actual development progress. The project did not fully adhere to the original schedule due to increased complexity in AI integration and dashboard analytics. While core milestones were completed, several advanced features were postponed to future work. Despite this deviation, essential system components were delivered with acceptable stability, reflecting a balanced trade-off between timeline adherence and implementation quality.

Quality of Implemented Features: The quality of implemented features was assessed through manual unit testing, integration testing, and user acceptance testing. Core system workflows demonstrated reliable and consistent behavior under normal usage conditions. Although automated testing was limited, the applied testing approach was sufficient to ensure functional correctness and system stability within the project's academic scope.

Resource and Constraint Considerations: Project development was conducted under constraints related to time, manpower, and data availability, which are typical for student-based academic projects. These limitations affected the level of AI optimization and feedback complexity. However, available tools and frameworks were effectively utilized to deliver a modular, maintainable, and functionally complete system.

Stakeholder Satisfaction: Stakeholder satisfaction was assessed primarily through internal evaluation by the development team and feedback collected from test users. These stakeholders played a critical role in validating usability, functionality, and perceived system usefulness. Their feedback provided qualitative insights that supported both system refinement and the evaluation of project success.

2. Assessment of Whether Project Goals Were Achieved

2.1. Overall Goal Achievement

The Study Planning project did not fully achieve all objectives defined at the beginning of the development lifecycle. However, the core goals related to building a functional, user-oriented study planning system were partially and meaningfully achieved. The project successfully demonstrates the feasibility of integrating task management, time tracking, and AI-assisted timetable generation within a web-based platform.

While certain advanced objectives were deferred due to time and resource constraints, the implemented system fulfills the foundational requirements of a personalized study planning application.

2.2. Successfully Implemented Functionalities

The following key functionalities were fully implemented and operated reliably at the time of evaluation:

- User registration and login with session-based authentication
- Study task management, including task creation, modification, and deletion
- AI-based timetable generation based on user characteristics and constraints
- User profile updates reflecting personal characteristics and study preferences
- Study time counter for tracking learning duration
- Dashboard features, including study time analysis, completed task visualization, and time distribution insights

These features collectively support the main use cases identified in the requirements analysis and represent the essential functional value of the system.

Unachieved and Partially Achieved Objectives

Some objectives were only partially achieved. In particular, the AI recommendation mechanism has not yet been optimized to handle complex or highly dynamic study behaviors. In addition, the feedback loop for improving AI recommendations remains relatively simple due to limited data and development time. These limitations indicate clear directions for future enhancement.

3. User Feedback and Satisfaction

Feedback from test users was generally positive regarding system usability, task management, and dashboard visualization. AI-generated timetables were considered helpful as general guidance, although users expressed the need for stronger personalization. Overall satisfaction was higher for core functionalities than for advanced AI features.

CHAPTER 11: LIMITATIONS AND FUTURE WORK

1. Project Limitations

Despite achieving key functional goals, the project exhibits several limitations that affect its completeness and scalability. The AI component remains at an early development stage and has not been fully optimized for all user scenarios. The chatbot feature is incomplete and currently provides limited interaction and guidance.

The system also lacks a comprehensive feedback mechanism that allows users to adjust or refine AI-generated recommendations. This limits the system's ability to learn from user behavior and improve over time.

From a user experience perspective, frontend design elements such as color schemes, visual consistency, and layout refinement require further improvement. Additionally, the absence of onboarding support, such as tutorials or usage guidance, may reduce usability for first-time users.

2. Future Work and Improvements

Future development should prioritize enhancing the intelligence and adaptability of the AI components. Improving machine learning models and completing the chatbot functionality would enable more interactive, context-aware, and personalized study planning support.

Incorporating a user feedback module is another important direction for future work. Allowing users to rate schedules, submit feedback, or adjust recommendations would support iterative learning and significantly improve system effectiveness.

From a user experience perspective, future enhancements should focus on refining frontend design elements, including color schemes, visual hierarchy, and responsive layout improvements, to create a more engaging and modern interface.

Additionally, implementing onboarding features such as user guides, tooltips, or step-by-step tutorials would improve accessibility for new users and reduce the learning curve when interacting with the system.

Overall, addressing these limitations in future work would enhance system usability, intelligence, and scalability, moving the project closer to a fully deployable and user-centered study planning platform.

CHAPTER 12: TESTING

12.1. Test case 1: Login

Pre-conditions

- The user account for which the login is being tested must already exist in the system
- The website or application hosting the login functionality should be accessible and operational
- The tester should have the correct login credentials for the user account being used in the test

Step	Action	Expected System Response	Pass/Fail	Comments
1	Access the website at the first time	Show the user the page in guest mode	Pass	
2	Click the "Đăng nhập" button in guess page	Display the “Login” page	Pass	
3	Enter username and password	Username and password are properly shown in the fields	Pass	
4	Press the "Đăng nhập" button on login page	Load email and password into the database for checking	Pass	
5	Check the account's validity	Compare the database with the supplied account. Send the user to the login page with an error message if the account is invalid.	Pass	

Table 9: Table of Test case 1 “Login”

Post-conditions
- Enables users to utilize additional features according to their account and access their account home page.

12.2. Test case 2: Register

Pre-conditions
- The website or application hosting the registration functionality should be accessible and operational. - Information required for registration, such as email address, username, password, and any additional fields, should be available to the tester. - The tester should use a unique email address or username that has not been previously registered

Step	Action	Expected System Response	Pass/Fail	Comments

1	Access the website at the first time	Show the user the page in guest mode	Pass	
2	Press the “Đăng ký ngay” button in guest page	Display the “Register” page	Pass	
3	Enter name, email, password			
4	Press the “Bắt đầu thôi!” button in “Register” page	Load this information into the database	Pass	
5	Save user information into the database	New user data, including email address and password, is inserted into the system database.	Pass	
6	Navigate to “Login” page for checking			
7	The user enters the email and password for the account just registered	Show the "Successful" message and navigate to Dashboard or Homepage	Pass	

Table 10: Table of Test case 2 “Register”

Post-Conditions

- A new user account should be successfully created in the system
- If the registration process involves email verification, a confirmation email should be sent to the registered email address
- Upon successful registration, the user may be redirected to their dashboard or the application's home page

12.3. Test case 5: Creating schedule

Pre-conditions

- The FurniScape website is accessible and operational.
- User is logged in with a valid account.
- Design elements/items are available in the system to be dragged and dropped.

Step	Action	Expected System Response	Pass/Fail	Comments
1	Access the website	There are basic information of schedules.	Pass	
2	Press the “Lịch của tôi” button	Display all available schedules	Pass	
3	Press the “Create new schedule” button	Display the design shcedule page	Pass	
4	Drag and drop elements which users wants onto the schedule			
5	Press “Lưu” button	The element is saved in calendar and display the design schedule page.	Pass	

Table 11: Table of Test case 5 “Creating Schedule”

Post-Conditions

- The new schedule is successfully saved to the database.
- The created schedule appears correctly on the user's calendar view.
- The design schedule page is displayed with the saved changes.

12.4. Test case 6: View and edit schedule

Pre-conditions

- FurniScape website is accessible and the user is logged in.
- At least one schedule exists in "Lịch của tôi" (My Schedule).
- **The schedule contains existing elements** (furniture/items) available for editing.

Step	Action	Expected System Response	Pass/Fail	Comments
1	Access website	The website homepage is displayed.	Pass	
2	Click on “Lịch của tôi” button	Display all available schedules	Pass	
3	Click on “edit” button			

4	Click element available in schedule	Display the information of elements.	Pass	
5	Change somethings which users want			
6	Click on “Lưu” button	The element is saved in calendar and display the design schedule page.	Pass	

Table 12: Table of Test case 6 “View and Edit Schedule”

Post-conditions
- The changes to the element are successfully updated in the database.
- The "Design Schedule" page is displayed with the updated element information.

12.5. Test case 7: Create tasks

Pre-conditions
- The FurniScape website is accessible and the user is logged in.
- **At least one schedule exists** in the system for the user to select.
- The "Nhiệm vụ" (Tasks) page is operational.

Step	Action	Expected System	Pass/Fail	Comments

		Response		
1	Access website	The website homepage is displayed.	Pass	
2	Click on “Nhiệm vụ”	Display the task page	Pass	
3	Choose schedule which users want	Details of the elements of schedule are displayed.	Pass	
4	Click the blank in schedule	Display form consist of information of task which should be filled	Pass	
5	Fill information and save tasks	Display task on the schedule	Pass	
5	Click the tasks available on schedule	User can change somethings which they wants	Pass	

Table 13: Table of Test cast 7 “Create tasks”

Post-conditions

- The new task is created and saved successfully to the database.
- The task is visually displayed on the selected schedule timeline.
- Any modifications to existing tasks are updated and saved.

12.6. Test case 8: Generate Schedule by AI

Pre-conditions:

- The FurniScape website is accessible and the user is logged in.
- The "AI Schedule Creation" feature is active/enabled in the system.
- User has sufficient permissions to create new schedules.

Step	Action	Expected System Response	Pass/Fail	Comments
1	Access website	The website homepage is displayed.	Pass	
2	Click on "Tạo lịch AI"	Display the smart schedule pagpage	Pass	
3	Fill in the requirements and generate schedule	Calendars are created based on users' personal preferences by AI.	Pass	
4	Click "Chấp nhận và Lưu"	Save the data which is created by AI	Pass	

Table 14: Table of Test case 8 "Generate Schedule by AI"

Post-conditions

- The AI-generated schedule is successfully saved to the user's account database.
- The system redirects the user to the "My Schedule" or Calendar view.
- The new schedule is visible with all AI-suggested details.

CHAPTER 13: CONCLUSION

1. Reiteration of Key Achievements

This project presented the design and development of a Study Planning Web Application that integrates task management, time tracking, and AI-assisted timetable generation within a web-based decision support framework. Despite limitations in time and resources, the project

successfully demonstrated the feasibility of combining a Java-based web application with a Python-based machine learning service to support personalized study planning.

Key achievements include the implementation of a modular system architecture, reliable user authentication, effective task management, and a functional dashboard that provides meaningful insights into study time and task completion. The integration of an AI-based timetable generation mechanism, although not fully optimized, represents an important step toward intelligent and personalized academic support systems. Collectively, these outcomes satisfy the core objectives of the project and reflect the successful application of theoretical knowledge to a practical software development context.

2. Closing Remarks

While the system does not yet represent a fully mature or commercial-ready product, it provides a solid foundation for future enhancement and research. The project highlights both the potential and challenges of applying artificial intelligence to real-world study planning scenarios, particularly under academic constraints.

Overall, this work contributes to a deeper understanding of web-based system development, AI integration, and user-centered design. The Study Planning Web Application serves not only as a functional prototype but also as a valuable learning experience, offering clear directions for future improvement and continued development.