The Hong Kong Polytechnic University

Department of Computing

COMP4913 Capstone Project

Report (Proposal)

An online reservation system for teaching

Student Name:	HONG Xincong
Student ID No.:	24028069d
Programme-Stream Code:	61435-SYC
Supervisor:	Prof.WANG Qixin
Submission Date:	2025-10-16

Contents

Background and Problem Statement

In modern university education, most majors have a Capstone project. Students need to make appointments with their supervisor for a presentation or meeting. However, final year students are busy, and professors have their own priorities. Therefore, finding a mutually satisfactory time for both students and professors can be challenging. To address this, an online reservation system for teaching is required, aiming to digitize and automate the appointment process for both students and faculty.

The basic functionality of the system was completed by the graduates. Teachers could create time slots and students could book them online. However, when considering actual deployment, some key vulnerabilities were discovered:

- **Lack of authentication and security:** In the existing system, the teacher function does not verify identity, so anyone can create time slots without authorization. This vulnerability can be easily attacked by scraping, creating a lot of useless appointment times, and eventually causing the system to crash. In addition, anyone can enter a real teacher's name (such as "Henry") without the system verifying the user's identity. Malicious users could exploit this vulnerability to create fake meetings, mislead students, or even send false email notifications.

- **Email information is not detailed**: Detailed information about the subject or course is not displayed in the email. This may make confusion to students when they make multiple bookings at same time.

- **The password design for students is unreasonable**: Every time a teacher creates an appointment, a new temporary password is generated for each student who can make an appointment. Students must search for different passwords across multiple emails to log into the system. This design not only increases the user's operational burden but also reduces the system's convenience and maintainability.

- **Does not support centralized login:** students cannot access multiple functions or book multiple courses simultaneously with a single login. In the long term, the system may not only be applied in capstone project presentations and meeting bookings but also become scheduling platforms for all courses on campus. This would enable faculty and students to manage multi-disciplinary, multi-purpose teaching within a single interface, significantly enhancing the digitization and efficiency of the academic process.

- **Poor usability for teachers:** The current system requires teachers to enter the specific meeting code to check the meeting information. This design may work for a single meeting, but in real-world, teachers need to manage multiple courses, schedule multiple meetings, or give student presentations, it becomes significantly inefficient and inconvenient. Moreover, if teachers do not save their meeting code, the teacher may need to recreate the meeting.

- **Lack of input validation and exception handling:** In the existing system, teachers created timeslots without front-end or back-end validation constraints. When students submitted preferences, the server also lacked comprehensive validity checks. If the system receives NULL data, the errors will appear, such as "mysqli_sql_exception: Column 'timeslotid' cannot be null,", it causes the page to crash and exposing stack trace and file path information to the end user.

- **Sensitive Data Exposed in URL:** The system places the examID and password directly in the URL (e.g., status.php?examid=...&password=...). The password is short and unencrypted, making it easy to brute-force. A guessed password could allow unauthorized access to the entire meeting roster and schedule. Although teachers will not share the examID, it's integrated into the system's email notification system, and every student will have one.

- **Data Consistency Risk: Missing Transaction and Rollback in result.php:** In the current result.php, the system operates on three data tables simultaneously: writing the assignment results to result and updating the student status to studentexammatch and meetingtimeslots. However, these SQL statements are not placed in the same transaction, and no rollback is set. If a statement fail, for example, if result is successfully written but the update to studentexammatch fails, data inconsistency will occur. It is possible that a student's record in result will remain unchanged in studentexammatch. This will cause data confusion and make corrections difficult. This may cause problems with both atomicity and consistency.

- **The user interface (UI) design is not user-friendly:** The system doesn't provide a clear time slot display, it uses drop-down boxes to select time, it makes the booking process cumbersome. Students can't directly understand which time slots are available. On the teacher's side, there's no visual overview of the time slots, forcing teachers to rely on a raw list.

Objectives and Outcome

This project aims to enhance the existing online reservation system's architecture comprehensively, including Security, Data Consistency, Real-time Interaction, and user experience. Make the system more efficient, secure, and long-term scalable appointment platform. Although the existing system is already equipped with the basic function and scheduling method, it still has a lot of problems in insecure identity authentication and credential management, a lack of transaction control leading to inconsistent data, complex interface design affecting the user experience, and an unreasonable database structure. In the future, these pain points will be enhanced and carry out comprehensive improvements, with the following specific goals and achievements:

- **Real-time Booking Implementation:** In the existing system, after submitting the reservation preference, the students cannot receive the result instantly; they need to wait for the system update after the deadline. This may seem like just a delay in information feedback, but it may cause problems in actual teaching management. Students may make other important arrangements during their waiting, such as other presentations. When the system returns the final result, which students get the last time slot of previous preference(unluckily), and it overlaps the new arrangement, students may miss the opportunity to re-coordinate and can only passively give up or temporarily change the arrangements. The new system introduces a real-time booking mechanism that provides instant feedback and immediate slot confirmation, allowing students to know their booking status right after submission.

- **Data Consistency and Atomicity Assurance:** The core design structure in the backend uses transaction control, allowing all modifications to be executed in one transaction. If booking procedure causes any errors, the system can trigger rollback automatically. By doing this, atomicity and consistency can be guaranteed. In order to further meet the demand for immediate feedback in a high-concurrency environment, the system will use pessimistic locks to prevent conflicts when a lot of students make the reservation at the same time.

- **Strengthening system security protection and identity verification mechanisms:** In order to improve security and convenience, the new system will change to use a single password sign-on mechanism. Students and teachers only need their personal account to log in to the system, allowing them to use one account to make the bookings for multiple courses. The One-Time Password will also be equipped, used for secondary verification to ensure the authenticity of the login user.

- **Multi-course Integration and Scalability:** Implement multiple course integrations, and teachers can create multiple time slots for different purposes. Students can book their related time slots of the campaign (Students have already been added to this campaign by the teachers).

- **Database Redesign and Optimization:** Clearly distinguish between user, role, course time, and student reservation data tables. By doing this, the system can support centralized management of multiple teachers, multiple courses, and multiple time periods. Moreover, the system implements the user permissions levels by role-based access control to ensure different users just can use functions and data under their scope of responsibility.

- **UX/UI Redesign:** Design the user interface, making the system compliant with the requirements of Human Computer Interaction, such as consistent layout and color scheme, and allowing the user to finish the task in the minimal steps. Allowing teachers to create, modify, and view the time slot more easily. The student side displays the available time slot, availability, and deadlines clearly. And allow them to check their reservation status instantly. The new design will be easier to use.

Final

- The final goal is to apply to the computing department.

Project Methodology

The main technology stack is Docker + FastAPI(Python) + MySQL + HTML + jQuery.

System Architecture Overview: The system adopts a frontend and backend architecture separation and interacts with data through APIs. Also using the database to store data. The docker will be used in the deployment environment.

HTML + jQuery + Ajax（Frontend Development）

Interactive UI:

- Student side: Using table to display available time slot. Also use the pop-up window to confirm the reservation operation.
- Teacher's side: Providing a visual time management panel for quick viewing, creating, modifying, and deleting time slots.
- The overall layout adopts a responsive design, compatible with various device sizes (desktop and mobile).

Real-time Interaction: The fronted uses Ajax to call the API to communicate with the backend, and the backend returns the JSON after performing the logic.

FastAPI + Python（Backend Development）

- **Modularity:** Each function (creating a time slot, canceling an appointment, querying a student list, etc.) exists as an independent module, reducing code coupling.
- **Asynchronous processing (ASGI architecture):** can better handle high concurrency situations [1].
- **RESTful API design:** Use a unified /api/... path and standard HTTP methods (such as GET /api/timeslots/{teacher_id}, POST /api/bookings) to implement a standardized structure, making it easy to call and maintain.

MySQL and pessimistic locking (Ensure atomicity and consistency in high concurrency situations [2].):

- **The new database table structure includes:**
  1. users (user information and role permissions)
  2. booking_time_slot (course time slot information)
  3. students_booking (student booking mapping)
  4. time_slot_group_student (course group binding)
- **When students simultaneously attempt to reserve a time slot**
  1. Start a transaction (START TRANSACTION);
  2. Lock the target record using SELECT ... FOR UPDATE;
  3. Check the remaining slots (vacancy);
  4. If a slot is available, update and commit the transaction (COMMIT); otherwise, roll back (ROLLBACK).

Hight Concurrency Testing: Use Python's threading to build a stress test script to simulate 300 users' booking at same time.

Real-Time Booking Module Design

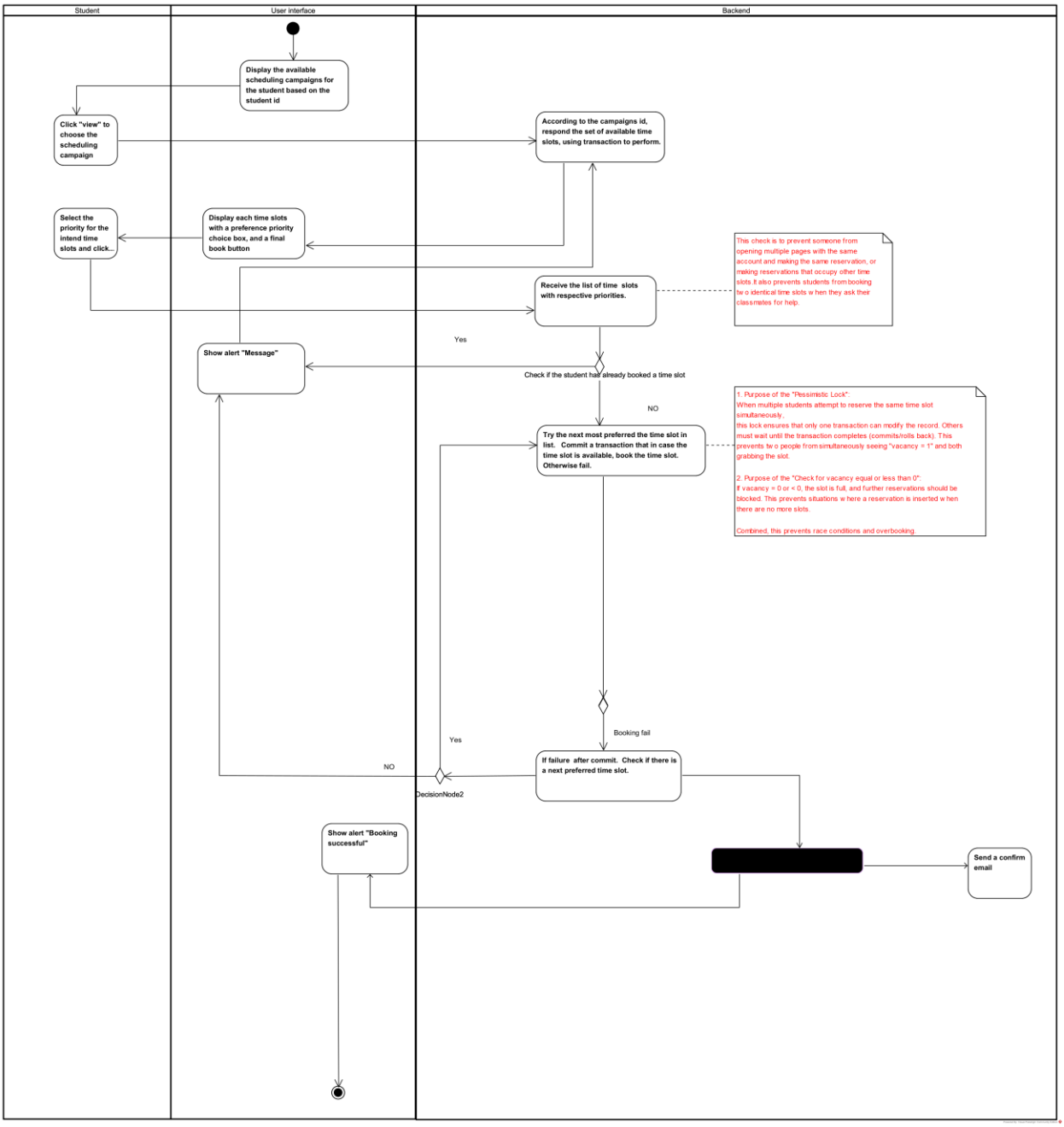

Figure 1. Core Booking Logic with Transaction Control and Pessimistic Lock

Project Schedule

Phase	Time Period	Tasks & Expected Milestones / Conclusions
Phase1	Stember2025-October 2025	1. Analysis of the existing system problem (Security, Data Consistent, and User interface). 2. Identify direction of Improvements and new functions. 3. Design the system structure. 4. Draw the Activity diagram which is the system core logic for reservation. 5. Finish the project proposal
Phase2	November 2025-January 2026	• **Backend development:** 1. Build a FastAPI + MySQL development environment. 2. Implement the logic according to the activity diagram. 3. Implement user authentication. • **Front-end development:** 1. Use HTML + jQuery to build interactive interfaces. 2. Realize real-time communication and reservation feedback between front-end and back-end through Ajax. 3. Enhance the user interface and achieve responsive design. • **Finish Interim Report**
Phase3	February-March 2026	1. Complete development of all functional modules (teacher and student). 2. Perform system testing (High concurrency and rollback testing): Write a multi-threaded script that can stimulate 300 students to make the reservation at the same time. 3. Verify whether transaction control and pessimistic locking mechanisms could prevent overbooking under high concurrency.
Phase4(Final)	**April 2026**	1. Set up and deploy the website in a server environment (Modify Docker when necessary). 2. Complete the final report including architecture design, core code, and performance results.

Resources Estimation

Hardware	• Development Laptop: Used for local development, testing, and debugging of the FastAPI backend and frontend interfaces. • Server: Hosting the deployed web application (Docker + FastAPI + MySQL)
Software	• FastAPI (Python Framework) • MySQL Database • Docker • GitHub • Visual Studio • SMTP Email Service • Excel

Reference

[1] J. Wang, "Python analysis of FastAPI optimization in high-concurrency applications," *CSDN Blog*, Oct. 10, 2025. [Online]. Available: https://blog.csdn.net/2501_91172624/article/details/146887465. [Accessed: Oct. 19, 2025].

[2] Oracle Corporation, *MySQL 8.4 Reference Manual – "17.7.2.4 Locking Reads"*, MySQL, 2025. [Online]. Available: https://dev.mysql.com/doc/refman/8.4/en/innodb-locking-reads.html. [Accessed: Oct. 19, 2025].