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