CSC3170 Project

Correspondence: Yuwei Xu

Email: yuweixu@link.cuhk.edu.cn

2025/09/30

The project will be completed individually and will require the submission of a demo video, a final report, and the project code. All deliverables should be packaged and submitted to Blackboard by **23:59**, **23 Nov 2025. No delay is permitted.** The report should be no longer than **five pages**, and the demo video should not exceed **five minutes.** Please package the **report, demo video, and code together for submission**. The project takes 30% of your final grade. Plagiarism is strictly prohibited, and all code submissions will be checked for duplicates to ensure originality.

The project must include both a client and a server component. That means at least two separate processes must run and communicate with each other via the network (e.g., sockets, HTTP, RPC). **In this way, a clear separation between frontend and backend is essential.** While third-party libraries (such as SQLite) may be used, the client must not directly manage them.

This is an individual project. You will be responsible for designing, implementing, and testing both client and server parts on your own.

Background:

Assume that you have been recruited by CUHK Shenzhen to join a development team and are responsible for database design. You may choose one organization from the provided scenarios that you are interested with, and proceed with the development based on your selection.

Make reasonable and realistic assumptions regarding the organization's operations, including its underlying entities, attributes, and relationships. Ensure that all assumptions are clearly stated and justified.

Project Requirements

Your project must satisfy the following requirements. To illustrate each requirement, we provide a simple example using a student dormitory management system. For your specific implementation, you are free to choose **different scenarios in candidate list** and have more complex functions:

1. **Analyze the organization's requirements.**

   You must define at least **2 distinct roles** within your system and implement a minimum of **9 different functions.**

   For example, in a student dormitory management system, there could be roles such as "Student" and "Administrator".

   Student Functions: Students should be able to log in, submit maintenance requests, and provide feedback.

   Administrator Functions: Administrators should be able to manage dormitory information, update student details, and more.

2. **Design the database schema.**

   Identify relevant entities, attributes and relationships, and any constraints and attributes, clearly labeling primary keys, foreign keys. You should list the schema for each table in you report.

   For example, in the student dormitory management system, there are several data tables such as `Dormitory_Information`, `Student_Information` and so on:

   `Dormitory_Information:`

Dormitory No.	Building No.	Floor No.	Dormitory Door No.	Count of Beds	Count of Vacant bed

   `Student_Information:`

Student ID	Password	Name	Gender	Major	Dormitory No.

   Note: Red indicates the primary key, and Dormitory No. in `Student_Information` referencing the primary key in `Dormitory_Information.`

3. **Populate the Database with Realistic Data:**

Populate the schema with a reasonable amount of realistic data to make sure that each function works properly. You are responsible for acquiring, crawling, or generating appropriate data for your project.

For example, in a student dormitory management system, the `Student_Information` table should contain reasonable data:

Student ID	Password	Name	Gender	Tel. No.	Major	Dormitory No.
21383	123456	Ming Li	Male	83482912	Physics	M101
21384	Asd123	Ali Ng	Female	89122384	Math	F101
21385	234567	Johnson	Male	84628190	Chemistry	M102

Hint: For testing purposes, generating around 30-50 data records with an LLM is acceptable. Use this data to validate your system functions. Additionally, please consider your design for extreme data scenarios to verify the robustness of client.

4. **Develop SQL Queries for the Designed Functions:**

Write SQL queries that implement the functions you have designed based on the table schemas. Ensure the queries align with the operations required by the system.

For example, in a student dormitory management system, when a student attempts to log in, the system should use an SQL query to check whether the `Student_ID` and `Password` from the table `Student_Information` match the information provided by the user:

User Input:

```
Student_ID: 21383
Password: 123456
```

SQL Query:

```
SELECT Student_ID
FROM Student_Information
WHERE Student_ID = 21383 and Password = 123456;
```

Note: This is a simple query for demonstration purposes. In a real-world scenario, you can **hash and salt passwords** to avoid storing them in plain text and prevent direct password comparisons. Instead, the system should compare hashed values to enhance security. A more secure approach would involve:

a) Storing a hashed version of the password.

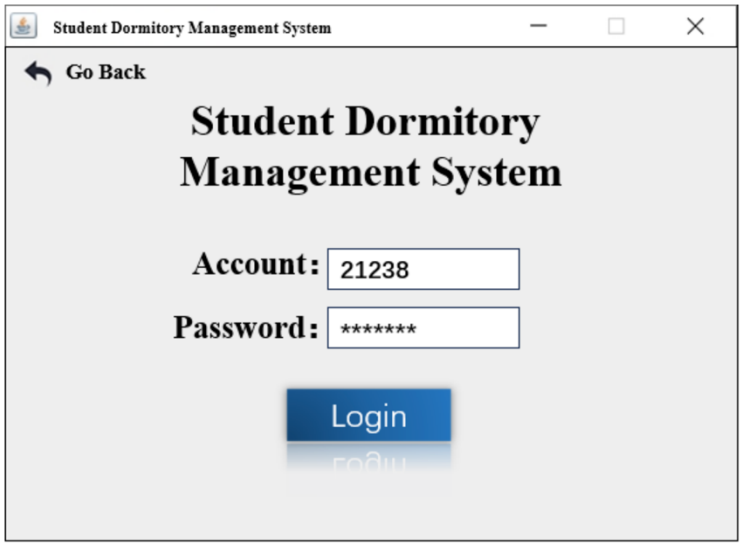
b) Using a hashing function (such as `bcrypt`, `SHA-256`, or `Argon2`) when users enter their password.

c) Comparing the stored hash with the hash of the input password.

5. **Reasonable front-end and back-end design**.

The back-end system should be designed to meet all project requirements effectively and efficiently. The front-end should offer a smooth user experience, with an intuitive and easy-to-use interface. You may choose to implement the front-end using a Command Line Interface (CLI) or develop a Web or Mobile App interface.

For example, in a student dormitory management system, a basic implementation of a student login screen might look like the following:



The following can be a CLI implementation of a student login screen:

```
Welcome to the Student Dormitory Management System
Please enter your Student ID:
> 21383
Please enter your password:
> ********
Login Successful. Welcome, Xiaoming!
```

Or a more elegant one:

Note: You can use CLI to implement the front-end, while the web design and APP design will have extra points, up to 10% of the total project score, but will not exceed the maximum total project score, that is to say, without a graphical interface, you can still get full marks, but a good graphical interface could help you to get a high score!

Project Scenario List:

We provide you with five alternative organizations here, you should only choose the items listed below!

1. CUHK(SZ) Dormitory Management System

As CUHK(SZ)'s campus continues to expand and its student population continues to grow, traditional dormitory management models are no longer able to meet the demands of modern university management. Paper-based registration, manual assignments, and verbal communication lead to inefficient dormitory allocation, delayed maintenance responses, and even dormitory conflicts and wasted resources. To improve dormitory management efficiency, optimize the student living experience, and reduce the workload of the back-office department, the CUHK(SZ) Dormitory Management System was developed. Through a simple and intuitive online platform, the system digitizes the entire process of dormitory allocation, application processing, and maintenance management.

The following are the main concepts and some simple requirements, the specific implementation needs to be expanded in detail:

Student:

a) Register, log in, and modify personal information (e.g., contact information)

b) View individual dormitory assignment details (building number, room number, roommate information)

c) Submit dormitory adjustment requests (e.g., room change)

d) View dormitory fee overview (e.g., utilities)

e) Submit or modify dormitory maintenance requests

Administrator:

a) Log in and modify account information

b) Approve or reject student dormitory adjustment requests

c) View dormitory status (e.g., room availability, occupancy status, payment status)

d) Process dormitory maintenance requests and update status

2. CUHK(SZ) Library Management System

With the improvement of people's knowledge level, libraries have become an indispensable part of daily life, and book management has become difficult and important. The volume of book stock and business is huge, and it is not feasible to rely only on the traditional bookkeeping management; and library management system databases have the advantages that cannot be compared with manual management, such as: search and query, easy to locate, high reliability, large storage, good confidentiality, low cost, and so on. These advantages can greatly improve the efficiency of library management. Libraries need to provide detailed information on books and library inventory for schools or the community in need and need to establish a huge database.

The following are the main concepts and some simple requirements, the specific implementation needs to be expanded in detail:

Reader:

a). Reader registration, login account, modify account information

b). Query book information, borrow books

c). Query the information of borrowed books, return borrowed books, and extend borrowed books.

Librarian:

a). Log in to your account and make changes to your account information

b). Manage library books (modify, update, add or delete).

c). View book information and borrowed status

d). Manage reader account information (query, modify, add, delete)

e). Viewing reports (checking loan and return records, checking unreturned books, etc.)

f). View library logs (including reader operations and administrator operations).

Library Director (only one):

a). It has the authority of all librarians, based on which it can add and delete librarians.

3. CUHK(SZ) food ordering system

With social progress and technological development, the needs and forms of food service industry are undergoing rapid changes. The traditional way of ordering food is gradually being replaced by modernized ordering systems. This transformation not only brings about the convenience of ordering food for consumers, but also provides a more efficient management and service model for restaurant operators. Online food ordering systems are able to connect consumers and restaurants via the Internet, enabling users to select, order and pay for food at any time and place via web or mobile applications.

The following are the main concepts and some simple requirements, which need to be expanded in detail for specific implementation:

User:

a). Users need to be able to register, login to their account and make changes to their account information

b). Users can place orders from different stores

c). Users can modify orders to a certain extent

d). Users can view order information (including unshipped, shipped, completed)

e). Users need to confirm receipt of shipped orders.

Administrator:

a). Administrators can log in and modify account information

b). Add and delete delivery staff

c). Manage store information (add, delete, modify)

d). Manage orders (assign unshipped orders, view order information, etc.)

4. **CUHK(SZ) Course Selection System**

With the rapid development of higher education and the widespread use of information technology, traditional course selection methods no longer meet the needs of modern university education. Traditional course selection methods often rely on paper forms or simple spreadsheets, leading to problems such as inefficiency, lack of transparency, and frequent conflicts. Modern universities require an efficient course selection system that provides students with a convenient course selection experience and provides effective management tools for academic administrators.

The following are the main concepts and some simple requirements, the specific implementation needs to be expanded in detail:

Student:

a) Register, log in, and modify personal information (e.g., contact information)

b) View available course information (course title, credits, time, and location)

c) Select a course, with the system automatically checking for time conflicts

d) View a list of selected courses and their status (e.g., confirmed, pending review)

e) Modify selected courses (drop or change courses) during the add/drop period

Administrator:

a) Log in and modify account information

b) Manage course information (add, delete, and modify course details)

c) Process student course registration requests (review, confirm, or reject)

d) View course registration status (e.g., number of students, full-time status)

5. **CUHK(SZ) Visitor Management System**

As CUHK(SZ) accelerates its internationalization and increases its openness to the campus, the number of campus visitors (including parents, alumni, partners, and temporary visitors) has increased significantly. Traditional paper-based registration methods are not only inefficient and prone to information omissions, but also pose security risks, such as delayed visitor identity verification and visitor time conflicts. To improve campus security management efficiency and visitor experience, the CUHK(SZ) Visitor Management System was developed. Through a simple and secure online platform, the system digitizes the entire visitor reservation, approval, and registration process.

The following are the main concepts and some simple requirements, the specific implementation needs to be expanded in detail:

User:

a) Register and log in to an account

b) Entering basic contact information (e.g., name, phone number, and affiliation).

c) Schedule a visit time, location (e.g., academic building), and purpose of visit (e.g., tour, meeting).

d) View reservation status (approved, pending review, denied) and receive system notifications.

e) Modify or delete reservation information, while being mindful of any conflicts with the current reservation status (re-review required after review).

Administrator (Staff):

a) Log in to the system and modify personal account information.

b) Approve or deny visitor reservation requests (real-time processing).

c) View all reservation records (filter by date and location).

d) Generate daily visitor statistics reports (e.g., number of visitors, popular locations).

Project scoring rules:

1. Requirements analysis (10%)

--Comprehensiveness of understanding and analysis of project requirements.

--Whether the functionality to be considered is fully described and understood.

--Whether the objectives of the project, the user groups and their needs can be clarified.

2. Design Accuracy (10%)

--Whether the database design meets the standardization requirements and effectively avoids data redundancy.

--Whether the best practices of database design, such as appropriate indexing, proper data type selection, etc., are reasonably used.

3. Reasonable Data Population and Query Implementation (10%)

--Whether the data is sufficient to cover all defined entities and relationships

--Whether the diversity of the data can support the testing of various queries.

--Whether the SQL query design is good to ensure that the system functions can be realized, and ensure the efficiency and correctness.

4. Reasonable front-end and back-end design, program usability (35%)

--Whether the design of the back-end API is clear, whether it supports the front-end functional requirements, and whether the response time of the API is reasonable.

--Whether the front-end provides good user interaction design, whether the interface is intuitive and easy to use, we accept you to use cmd as the front-end, but please pay attention to the user experience, to ensure that the program is easy to use!

5. Presentation and documentation (35%)

--Does the documentation document the development process, design decisions, and how to use the system?

-- Does the demo clearly show the main functions of the system, is it smooth, and is the demo user familiar with his/her own project?

6. *Additional Bonus Points* (up to 10%)

--If you choose to use web or app as the front-end user interface, we will provide you with a maximum of 10% bonus points based on the aesthetics and usability of your

design, which will be added to the final project grade, but will not exceed a maximum of 100%. (Please be assured that investing excessive effort in the UI is unnecessary. A simple, clear, and functional interface is valued and sufficient for this project.)

Documentation Requirements:

The document should contain the following sections.

1). the overall project description

2). the requirements analysis

3). the database and SQL design

4). front and back-end design and implementation

5). the summary and harvest

Demo Requirements:

1). An explanation of the database design, especially how it meets the project requirements.

2). Demonstrate the flow of key functions, especially those that are complex or core.

3). Ensure that the demo is well organized and flows clearly, and is controlled to be no more than 5 minutes in length.