Final Assignment (Version 1.3 – 4 September 2025)

ANALYSIS 8: SOFTWARE QUALITY (INFSWQ01-A | INFSWQ21-A)

Educational Period 4 [2024-2025]

URBAN MOBILITY backend system

To make it feasible as an assessment for this course, the following scenario is formulated to ensure that students have achieved at least the minimum level of the course learning outcomes, as defined in the course manual. Please note that this scenario might be very different in real world cases, which usually need other quality requirements. Normally such a system would involve many other requirements and components, but here you can limit yourself only to the given description.

Learning Objectives

The learning objectives of the assignment and mapping to the intended learning outcome of the course are listed below:

1. To understand the common mistakes of coders in input validation and communications with subsystems (LO2, LO3).
2. To apply the knowledge of input validation, SQL injection, and cryptography (LO1, LO4).
3. To partially build a secure system against various attacks initiated by user input (LO4).

Assignment

Introduction

Urban Mobility operates a network of shared electric scooters across the Rotterdam region. Travelers can unlock scooters directly by scanning a QR-code, or reserve them in advance via the mobile app.

In this assignment, you will design and implement a secure backend system that handles user authentication, scooter information, traveller data and admin monitoring. The focus of this assignment is on software security. Your system must follow best practices in input validation, encryption, logging, and secure role-based access.

This assignment consists of the design and implementation of a simple console-based interface in Python 3 for the mentioned backend system. This system should use a local database to store the data. You should use SQLite3 database for this purpose. Figure 1 Use Case Diagram provides an overview of the actors and their corresponding use cases.

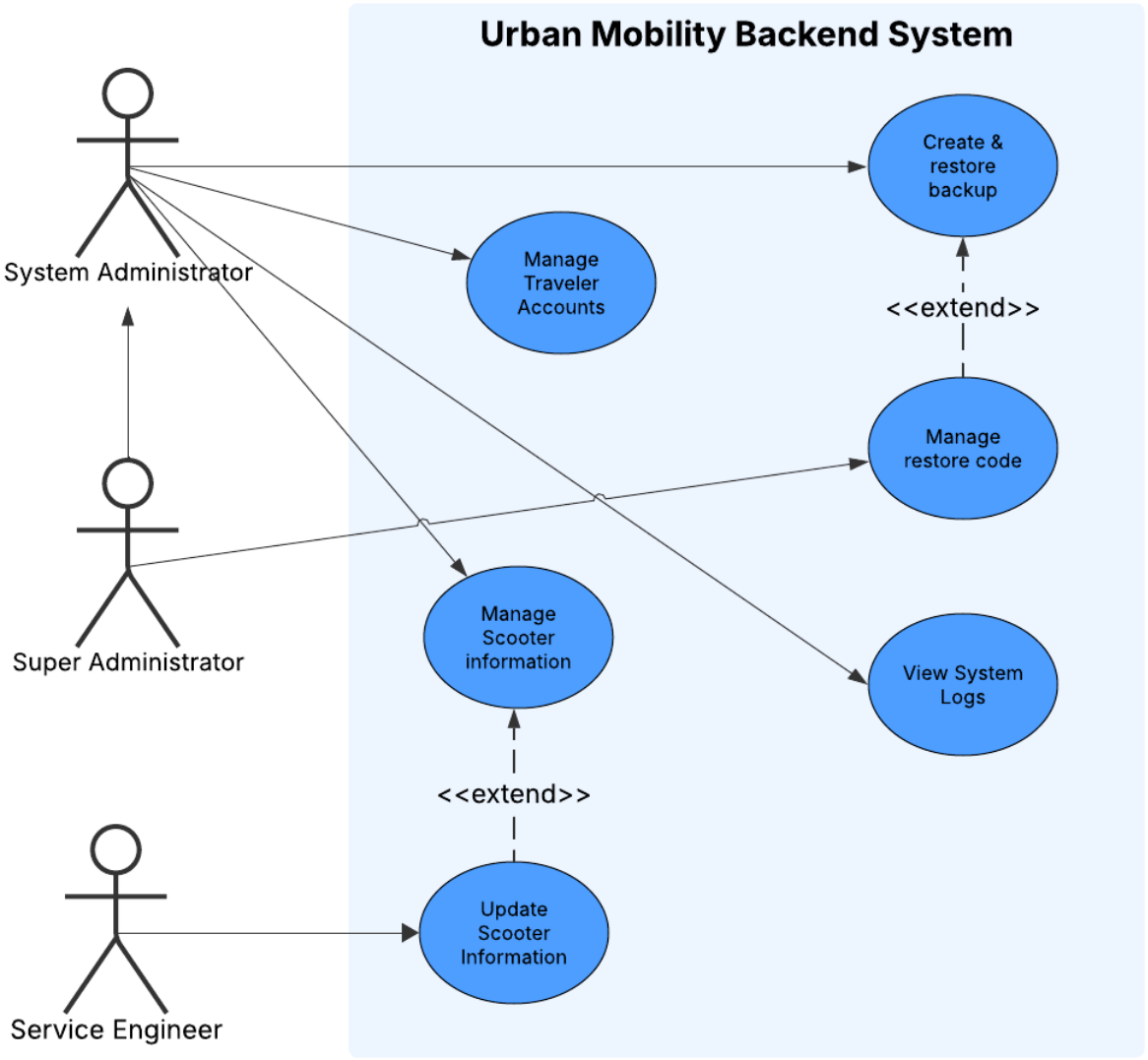

Figure 1 Use Case Diagram

Overview of the User Roles

Users of the backend system are the employees of Urban Mobility. In the system there are three User Roles, which are categorized in Table 1 User Roles.

User role	Created by	Remark
Super Administrator	Hard-coded	A Super Administrator has full control of the system. In practice, their main role is to create and manage System Administrator accounts.
System Administrator	The Super Administrator	A System Administrator can fully manage Traveler accounts & Service Engineer accounts and add, update and delete scooter information.
Service Engineer	The Super Administrator or the System Administrator	A Service Engineer can update some attributes of existing scooter information. They are not allowed to add or delete scooters.

Table 1 User Roles

The Super Administrator account is Hard-coded
For assessment purposes, the backend system must include a hard-coded Super Administrator account with a fixed username and password.

The fixed username must be: *super_admin*
The fixed password must be*: Admin_123?*

This approach is intentionally insecure and would not be acceptable in a real-world application. However, it simplifies access for instructors during the assessment process.

The customers of Urban Mobility are not users of the backend system
The customers of Urban Mobility (the travellers) are not users of the backend system. The users of the backend system and their corresponding Use Cases can be found in the following pages.

The data stored in the backend system

Traveller data

When a new traveller applies for an account, their information should be entered into the backend system. A new traveller can be registered in the backend system by a System Administrator and by the Super Administrator. The backend system then needs to automatically add the registration date and assign a unique customer ID to every new traveller. The registration date is the current time at the moment of entering the traveller data.

See Table 2 for the traveller data that should be entered into the backend system. For some traveller data attributes specific syntax rules are defined in the column 'Format'. The letter D means a digit (0-9), the letter X means an uppercase letter (A-Z).

Traveller data attribute	Description	Format
First Name	The traveller's given name.	
Last Name	The traveller's family name or surname.	
Birthday	The traveller's date of birth, used for identity verification and age-related restrictions.	
Gender	The traveller's gender, male or female.	
Street name	The name of the street where the traveller lives.	
House Number	The number of the traveller's residence on the street.	
Zip Code	The postal code corresponding to the traveller's address.	DDDDXX
City	The city where the traveller resides.	The system should generate a list of 10 predefined city names of your choice.
Email Address	The traveller's email address, used for communication and account management.	
Mobile Phone	The traveller's mobile phone number, used for account verification and notifications.	+31-6-DDDDDDDD Only DDDDDDDD to be entered by the user.
Driving license number	The unique number of the traveller's driving license, required for eligibility to rent scooters.	XXDDDDDDD or XDDDDDDDD

Table 2 Traveller Data

Scooter data

The backend system stores information about the scooter fleet. The Super Administrator and System Administrator can add new scooters, modify and delete existing ones. The Service Engineer can only modify some attributes of existing scooter information. When a new scooter is added to the fleet, the backend system should automatically add the in-service date. The in-service date is the current time at the moment of entering the scooter information.

See Table 3 for the scooter data that should be entered into the backend system. For some scooter data attributes specific syntax rules are defined in the column 'Format'.

Scooter data attribute	Description	Format	Attribute can be edited by	
			Super Administrator & System Administrator	Service Engineer
Brand	The manufacturer of the scooter (e.g., Segway, NIU).		✓	
Model	The specific model name or number of the scooter.		✓	
Serial number	A unique identifier assigned to each scooter by the manufacturer.	10 to 17 alphanumeric characters	✓	
Top speed	The maximum speed the scooter can reach, measured in kilometres per hour (km/h).		✓	
Battery capacity	The total energy capacity of the battery, usually measured in watt-hours (Wh).		✓	
State of Charge (SoC)	The current battery charge level, expressed as a percentage.		✓	✓
Target-range SoC	The recommended operating range of the scooter's battery, defined by a minimum and maximum State of Charge (SoC). Staying within this range helps ensure that the battery is neither overly discharged nor overcharged, optimizing its lifespan.		✓	✓
Location	The current GPS coordinates (latitude and longitude) of the scooter. Must be a real-world location within the Rotterdam region. Must allow for at least 2-meter accuracy.	5 decimal places, e.g., latitude = 51.9225, longitude = 4.47917.	✓	✓
Out-of-service status	Indicates whether the scooter is currently unavailable for use (e.g. due to low battery, maintenance or technical issues).		✓	✓
Mileage	The total distance travelled by the scooter since it was first used, measured in kilometres.		✓	✓
Last maintenance date	The date when the scooter was last inspected or serviced for maintenance purposes.	ISO 8601 format: YYYY-MM-DD	✓	✓

Table 3 Scooter data

The column ‘Attribute can be edited by’ in Table 3 indicates which attribute can be edited by which user role. When there is a checkmark in the appropriate cell it indicates that the corresponding attribute can be edited by the corresponding user role. If there is no checkmark then the corresponding attribute cannot be edited the corresponding user role. For instance, the attribute ‘Serial number ‘ can only be edited by the Super Administrator and the System administrator, not by the Service Engineer.

User Interface

The backend system must have a user-friendly interface (easy, efficient, and enjoyable) that allows users (Super Administrator, System Administrators, and Service Engineers) to perform their functions easily and smoothly.
A console-based interface is the sufficient requirement; however, a graphical user interface (GUI) is optional and allowed, **but not necessary**, provided that it also meets the usability expectations.

Ensure that your user interface provides sufficient information for the user to work with it. For example, if you have a menu "1. Register new traveller" which should be chosen by pressing **'R'** or **'r'** or entering **1**, this should be clearly displayed to the user on the menus screen. Do not suppose that the user (and your teacher when testing and grading your assignment) should guess how to work with the user interface.

Note that the user interface would not be graded for flexibility or efficiency of use, but if your teachers cannot properly work with the system, it might not be possible for them to correctly assess your work.

Data (DB) File

The main functionality of the backend system is to store and manage the information of travellers and scooters in the system. In addition, the system needs to store information of the users of the backend system. For this purpose, you need to implement the database using SQLite library in Python "sqlite3".

Note that the sensitive data, including usernames and travellers' data, must be encrypted in the database. You must not store any password in the database, rather as you learned (or will learn soon) in the lessons, **you must only store the hash of password in the database**.

Stakeholders, Users, Authorization, Functions and Accessibility Levels

More details about the stakeholders of the system are explained below:

1. **Travellers**

   Travellers are not the users of the concerned backend system and have no role or interaction with this part of the application. The only relationship between travellers and the backend system is that their information is recorded and stored in the backend system by a Super Administrator and System Administrator.

2. **Service Engineer**

   Service Engineers are employees of Urban Mobility who are responsible for the management of the scooter-fleet. Hence, they need to be able to manage some attributes of the existing scooter information. They are not allowed to add new or delete existing scooters. So, the minimum required functions of a Service Engineer in the system are summarized as below:

   - To update their own password
   - To update some attributes of scooters in the system
   - To search and retrieve the information of a scooter (check note 2 below)

3. **System Administrators**

   A System Administrator is a person who can maintain the system and perform some administration tasks on the application. They are IT technical people and not intended to work with the scooters. However, they should be able to perform all the functions of Service Engineers if needed. The minimum required functions of a System Administrator are listed below:

Same as Service Engineer:

- To update their own password.
- To update the attributes of scooters in the system
- To search and retrieve the information of a scooter (check note 2 below)

Specific for the System Administrator:

- To check the list of users and their roles.
- To add a new Service Engineer to the system.
- To update an existing Service Engineer account and profile.
- To delete an existing Service Engineer account.
- To reset an existing Service Engineer password (a temporary password).
- To update his own account and profile.
- To delete his own account.
- To make a backup of the backend system.
- To restore a specific backup of the backend system. For this purpose, the Super Administrator has generated a specific 'one-use only' code to restore a specific backup.
- To see the logs file(s) of the backend system.
- To add a new Traveller to the backend system.
- To update the information of a Traveller in the backend system.
- To delete a Traveller record from the backend system.
- To add a new scooter to the backend system.
- To update the information of a scooter in the backend system.
- To delete a scooter from the backend system.
- To search and retrieve the information of a Traveller (check note 2 below).

4. Super Administrator

Super Administrator is simply the owner or the manager of Urban Mobility. The main function of the Super Administrator is to define System Administrators and leave the system to them; however, they **should be able to perform all possible functionalities of the lower-level users** (System Administrator and Service Engineer).

The minimum required functions of a Super Administrator are listed below:

Same as Service Engineer:

- To update the attributes of scooters in the system
- To search and retrieve the information of a scooter (check note 2 below)

Please note: The credentials of the Super Administrator are hard-coded. Therefore, it is not required that he should be able to update his own password. The Super Administrator does not need a password change option in the user interface.

Same as System Administrator:

- To check the list of users and their roles.
- To add a new Service Engineer to the backend system.
- To modify or update an existing Service Engineer account and profile.
- To delete an existing Service Engineer account.
- To reset an existing Service Engineer password (a temporary password).
- To see the logs file(s) of the backend system.
- To add a new Traveller to the backend system.

- To update the information of a Traveller in the backend system.
- To delete a Traveller from the backend system.
- To add a new scooter to the backend system.
- To update the information of a scooter in the backend system.
- To delete a scooter from the backend system.
- To search and retrieve the information of a Traveller (check note 2 below).

Specific for the Super Administrator:

- To add a new System Administrator to the backend system.
- To modify or update an existing System Administrator account and profile.
- To delete an existing System Administrator account.
- To reset an existing System Administrator password (a temporary password).
- To make a backup of the backend system and to restore a backup.
- To allow a specific System Administrator to restore a specific backup. For this purpose, the Super Administrator should be able to generate a restore-code linked to a specific backup and System Administrator. The restore-code is one-use-only.
- To revoke a previously generated restore-code for a System Administrator.

Please note: The Super Administrator should not be able to restore a specific backup on behalf of a System Administrator (using a restore-code). The Super Administrator can only generate a restore-code, but only the intended System Administrator is allowed to use it to perform the actual restore.

Note 1: Service Engineers and System Administrators should have profiles, in addition to their usernames and passwords. Their profiles contain only first name, last name and registration date.

Note 2: The search function must accept reasonable data fields as a search key. It must also accept partial keys. For example, a user can search for a Traveller with a name “Mike Thomson” and customer ID “2123287421” by entering any of these keys: “mik”, “omso”, or “2328”, etc.

Log

The system should log all activities. All suspicious activities must be flagged, and the system needs to produce an alert/notification for unread suspicious activities once a System Administrator or Super Administrator is logged in to the system. The content of the log file(s) must be encrypted and should be only readable through the system interface, by the System Administrator or Super Administrator. It means that it should not be readable by any other tool, such as file explorer, browser or text editor.

A log should be structured similar to the following sample:

No.	Date	Time	Username	Description of activity	Additional Information	Suspicious
1	12-05-2021	15:51:19	john_m_05	Logged in		No
2	12-05-2021	18:00:20	superadmin	New admin user is created	username: mike12	No
3	12-05-2021	18:05:33	…	Unsuccessful login	username: “mike12” is used for a login attempt with a wrong password	No
4	12-05-2021	18:07:10	…	Unsuccessful login	Multiple usernames and passwords are tried in a row	Yes
5	12-05-2021	18:08:02	superadmin	User is deleted	User “mike12” is deleted	No
...	...	...	...	...	...	...

Note that the structure above is just a sample. You may choose your own desired format, but the information given above are the minimum information needed in the log file.

The OWASP Logging Cheat Sheet could be used for further reading.

Encryption of sensitive data

As mentioned before, all sensitive data in the database, including usernames, and traveller phones and addresses, as well as log data must be encrypted. For this encryption, you must use a symmetric algorithm.

Additional Clarification: At any point in time, whether the application is running or not running, any user with any text editor (outside of the Urban Mobility application) must not be able to see any meaningful data in the database or log file [unless they can decrypt the file(s)]. So, decryption and encryption of the files on start and exit is not an acceptable solution.

Passwords

Note that any form of password (encrypted or unencrypted) is not allowed to be stored in the database or any other data file in the backend system. Instead, you must only store hash of passwords in the system. For this purpose, you are allowed to use a third-party library.

Backup

The System Administrator and Super Administrator should be able to create a backup of the backend system. The Super Administrator can restore the backend system from any backup, the System Administrator only from a specific backup.

The backup must include the database (users, scooters and travellers' information). The backup should be in **zip** format. Note that the sensitive data in the DB file must already be encrypted. Thus, no additional encryption is needed when you are creating the backup zip file. The system must support multiple backups.

Usernames and Passwords

All Usernames and Passwords (except for the Super Administrator which is hardcoded) must follow the rules given below:

- **Username**:
  - must be unique and have a length of at least 8 characters
  - must be no longer than 10 characters
  - must be started with a letter or underscores (_)
  - can contain letters (a-z), numbers (0-9), underscores (_), apostrophes ('), and periods (.)
  - no distinction between lowercase and uppercase letters (case-insensitive)

- **Password**:
  - must have a length of at least 12 characters
  - must be no longer than 30 characters
  - can contain letters (a-z), (A-Z), numbers (0-9), Special characters such as ~!@#$%&_-+=`|\(){}[]:;'<>,.?/
  - must have a combination of at least one lowercase letter, one uppercase letter, one digit, and one special character

Grading

The assignment will be evaluated as either PASS or FAIL. To successfully pass the course, students must pass the assignment together with passing the exam.

Your assignment might be graded by your teacher (the teacher in your course timetable) or another teacher (Babak, Ahmad, René or Nanne).

Students will receive feedback from the teachers during the presentation. We suggest you note all the comments, in case you cannot successfully pass the assignment, you can use the comments and feedback to apply in the next chance.

Your assignment will be assessed according to the following marking Scheme. To successfully pass the assignment you need to meet the following assessment criteria:

- **You must get C1 and C2 at least as Satisfactory (L2 or L3)**, and
- **You must get C3 and C6 at least as Satisfactory (L1)**, and
- **You must get C4 and C5 at least as L1**, and
- **You must get a minimum of 10 points in total**.

Grading Table

Does the functionality of the submitted code match the assignment description?	Result
Functionality of the system as described. • If unsatisfactory, the assignment is FAIL and could not be evaluated for grading. • If satisfactory, then the table below will be used for grading.	(Unsatisfactory/Satisfactory)

Criteria and Points		Unsatisfactory		Satisfactory	
C1	Authentication and Authorization for users are properly implemented (Users access level)	L0	L1	L2	L3
C2	All inputs are properly validated.	L0	L1	L2	L3
C3	The system is secured against SQL injection.	L0		L1	
C4	Invalid inputs are properly handled.	L0	L1	L2	L3
C5	All activities are properly logged and backed up.	L0	L1	L2	L3
C6	Students can properly demonstrate and explain the system.	L0		L1	

C1, C2, C4, and C5:
- **L0:** Not implemented / very basic attempts **[0 point]**
- **L1:** Poor implementation / Major problems **[1 point]**
- **L2:** Minimum requirements are implemented / Minor problems **[2 points]**
- **L3:** Meet the requirements / Good implementation **[3 points]**

C3:
- **L0:** Not implemented / Poor implementation / Major problems **[0 point]**
- **L1:** Minimum requirements are implemented / Minor problems **[1 point]**

C6:
- **L0:** presentation is not satisfactory **[0 point]**
- **L1:** presentation is satisfactory **[1 point]**

Marking Scheme

An example of criteria and marking scheme is given in the table below. Please note that not all criteria are written in this table, but you can find all requirement and criteria in the assignment description which are already explained in detail. This table is to give you an idea of how the assessment procedure is.

Criteria	Unsatisfactory		Satisfactory	
	L0 (0 point)	L1 (1 point)	L2 (2 point)	L3 (3 points)
C1	Authenticating does not exist, or it is not working properly. Authorization is not implemented or at a very basic level.	Authentication is based on username and passwords. Usernames and PWs do not conform the given format and are not hashed. Application code has hard-coded role checks. Lack of centralized access control logic. There are some bugs or major problems.	Authentication has proper error messages. Authentication data are stored in an encrypted file using proper mechanism. Passwords are hashed. Authorization is implemented based on user roles and is centralized. No bugs or major problems.	Authentication has a secure recovery mechanism. It is protected against multiple wrong tries. Authorization is fully implemented based on the user's actions, without bugs or major problems.
C2	Input Validation is not implemented or at a very trivial level. There are many bugs or errors, which let Input Validation be bypassed easily.	Input Validation is implemented, but not for all input types, or contains few bugs and errors. Input Validation can be still bypassed. Blacklisting or mixed mechanism is used.	Input Validation is complete for all input types and does not allow bypassing. Whitelisting is used for all inputs without any flaw. There is no bug or error.	Input Validation is fully implemented and there are signs of following good practices in validation, such as checking for NULL-Byte, range and length, Validation Functions, etc.
C4	Invalid inputs are not handled, or at very basic level, with many bugs or errors.	There are some attempts of invalid input handling, but not correctly implemented. The reactions to different types of inputs are not suitable.	Invalid inputs are properly handled, without bugs or major problems. However, there might be very few improper reactions or minor improvements needed.	Invalid inputs are very well handled, and there is evidence of following good practices in response to different types of inputs.
C5	Logging, Backup and restore are not implemented, or there are major issues.	Logging, Backup and Restore are partially implemented. There are some bugs or shortcomings.	Logging, Backup and Restore are fully implemented. All suspicious incidents are logged. However, it could be still improved.	Logging, Backup and Restore are complete and well formatted, and there is evidence of good practices.

Criteria	Unsatisfactory	Satisfactory
	L0 (0 point)	L1 (1 point)
C3	The system is not secure (or partially) against SQL Injection. The SQL queries are not consistent throughout the code. There are coding bugs or issues.	The system is fully secure against SQL Injection. Appropriate mechanism and coding practices are used. SQL queries and codes are consistent in the final product.
C6	Students cannot properly run and demonstrate the system, or cannot explain it, or answer the technical questions. There is no evidence of original work or satisfactory contribution by the student.	Students can properly run and demonstrate the system and provide relevant answers to the majority of the technical questions. The work is evidently original, and there is evidence of sufficient contribution by the student.

Submission

Deliverable

The delivery to be handed in must consist of **one zip-file**, named as below:

studentnumber1_studentnumber2_studentnumber3.zip

The zip-file must contain:

1. A one-page **pdf document**, called **um_members.pdf**, containing **Names** and **student numbers** of the team (maximum **3** students per team),
2. A directory called **src**, containing all the **code files** and the **data files**, including one main file **um_members.py**. Starting the system should be done by running **um_members.py**.

⚠ Important Notes

1. **Do not** include any **bulky** Python system files in the delivery.
2. The code must **only** use **standard library modules,** plus **sqlite3**, **re** (regular expression) and any **cryptography** and **hash** library of your choice.
3. The code must run **error-free** (on a standard Windows or MAC PC). If needed, the code should only write to a temporary storage subfolder of the current folder, on the local machine.
4. The code should **only** write to **temporary storage** directories on the local machine, meaning on the current (running) folder or a subfolder of it.
5. We encourage you to work in a **team of 2 or 3 persons**. However, individual work is also acceptable, if you prefer to do it individually, or you are not able to make a team (e.g., retakers).
6. If submitted as a group work, **every member/student in the group** is responsible for the whole code and must be able to explain and justify the implementation.
7. When working in a team, **only one team member (the team leader)** submits the assignment, and all group members submit a group-info message with names and student numbers of the entire team, clearly indicating who is the team leader.
8. Part time-students can form a group only with part-time students and full-time students are allowed to form a group with only full-time students. Retake students are free to form a team with non-retake students (the only restriction is FT with FT, and PT with PT).

How to submit?

- **Students** can only submit it via the appropriate channels in MS Teams.

  Please note that submission through chat or email is not accepted. If you confront any problem in submission, contact your teacher before the deadline to assist you. As the submission dates (first chance and second chance) are usually on weekend, it is obvious that any communication with your teacher during the weekend might not be replied. Hence, during the working days of the submission week, ensure that you know how to submit it, and everything correctly works for you. Please do not leave any problem or issues with your submission on the submission day, otherwise you may miss on-time submission.

Deadline

Submission deadline for the **First Chance** is **20 JUNE 2025.**

Submission deadline for the **Second Chance** is **26 October 2025**.

Presentation

The presentation will be planned within the next three weeks after the submission deadline. It is mandatory for grading.

- Each presentation (for a submission, either individual work or group submission) will be scheduled for maximum 30 minutes.
- The presentation will be placed physically in Wijnhaven 107. The room will be announced later.
- An online form will be available few days before the submission deadline, with some time slots.

  Links of presentation scheduling form are provided on the course team on MS Teams. Ask your teacher to help if you couldn't find it.

  A student or a group can select their preferred time slot for the presentation. Because we have a restriction to complete the grading of assignment before the end of the education year, ensure that you timely choose your preferred time slot and register your name in the form, otherwise, you have to only pick one from the remaining time slots. Although we do our best to facilitate it, but we are not sure if we can provide additional time slots, if you miss your presentation.

What is the presentation?

You can see the criteria C6 in the Grading Table and Marking Scheme to see what the expectation for the presentation is.

You should run the system and demonstrate the functionalities of the system. You will be asked to explain how you have implemented the requirements of the system. You don't need to prepare any slides for the presentation.

For example, we may ask you questions like these: how you have implemented SQL queries? Is your application secured against SQL injection attacks and why? Where have you implemented the input validation layer? Which mechanism is used in the application to implement input validation? Why you used this mechanism? Are you protecting it against buffer-overflow attack and how? Are you protecting it against Null-byte attack and how? How do you deal a user-generated and server-generated invalid inputs? How do you decide that an activity in your log is suspicious or normal? Show the data in the database (you can install a third-part software or plugin for this), Show the code for the user authentication, …

You will be guided by your teachers during the presentation what you need to do.