Final Assignment

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

Educational Period 4 [2021-22]

Furnicor Family System (FFS)

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 apply the knowledge of input validation for both user-generated and server-generated data (LO1, LO4).
2. To experience the common mistakes of coders in input validation (LO2, LO3).
3. To partially build a secure input validator by coding practice (LO4).

Assignment

Introduction

In this assignment, we would like to make a simple system to store and manage the information of members of Furnicor retail outlet. Furnicor is a company that designs and sells ready-to-assemble furniture, kitchen appliances and home accessories, among other goods and home services. It currently has over 650,000 members worldwide. As Furnicor Family member, clients can enjoy many benefits, such as discounts on lots of products and free access to workshops and events, return their products indefinitely, get a free 14-day 'oops'-insurance, take a free cup of coffee or tea at the restaurant, etc. In order to enjoy all these exciting member benefits, a client must be a family member of Furnicor. They can be registered for free at a customer service desk. A membership management system is needed at the customer service to register new members.

This assignment consists of the design and implementation of a simple console-based interface in Python 3 for the mentioned system. The system should use a local database to store the information of members. You should use SQLite 3 database for this purpose. Figure 1 in the next page, depicts a general overview and the components of the systems.

Users are the employees of the company, which are categorized as below:

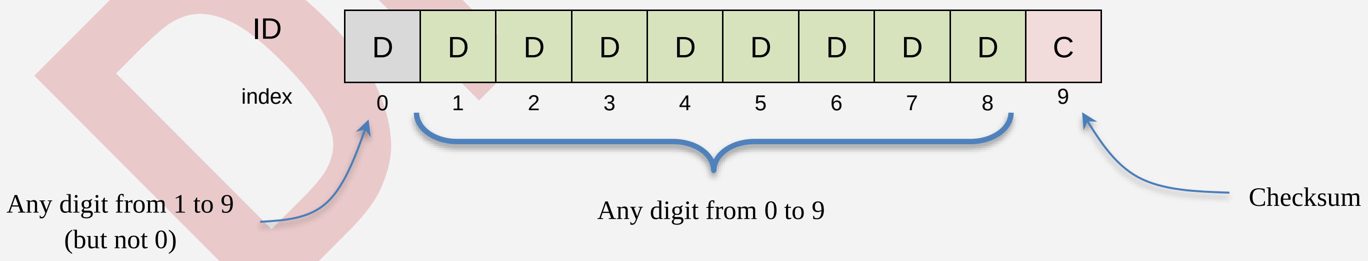
1. **Super Administrator** (Hardcoded) – A super admin has full control of the system.
2. **System Administrators** (to be defined by the Super Administrator only) – An admin who can manage advisors (register new advisor, modify or delete an advisor, etc.)
3. **Advisors** (to be defined by a system administrator or a super administrator) – An advisor can manage members in the system (register new members, modify, search or retrieve their information.)

Note that the members (or clients) are not users of the system (more details about the users and their roles can be found in following pages).

When new clients of the company request for membership, their information should be registered in the system, first. A new member can be registered in the system by an advisor (or a higher-level user, i.e., system admin or super admin). For a member, the following data must be entered to the system:

- First Name and Last Name
- Address (Street name, House number, Zip Code (DDDDXX), City (system should generate a list of 10 city names of your choice predefined in the system)
- Email Address
- Mobile Phone (+31-6-DDDDDDDD) – only DDDDDDDD to be entered by the user.

The system then needs to automatically add the registration date and assign a unique membership ID to every new member. The membership ID is a string of 10 random digits, not allowed to start with a Zero. The last digit on the right is a checksum digit, which must be equal to the remainder of the sum of the first 9 digits by 10.



Few examples are given below:

- Invalid ID number: **0**22328742**0** [a Zero at index 0 is not allowed]
- Invalid ID number: 522328742**4** [The checksum is not correct]
  (5+2+2+3+2+8+7+4+2) = 35 35 mod 10 = 5 ≠ **4**
- Valid ID number: 522328742**5**

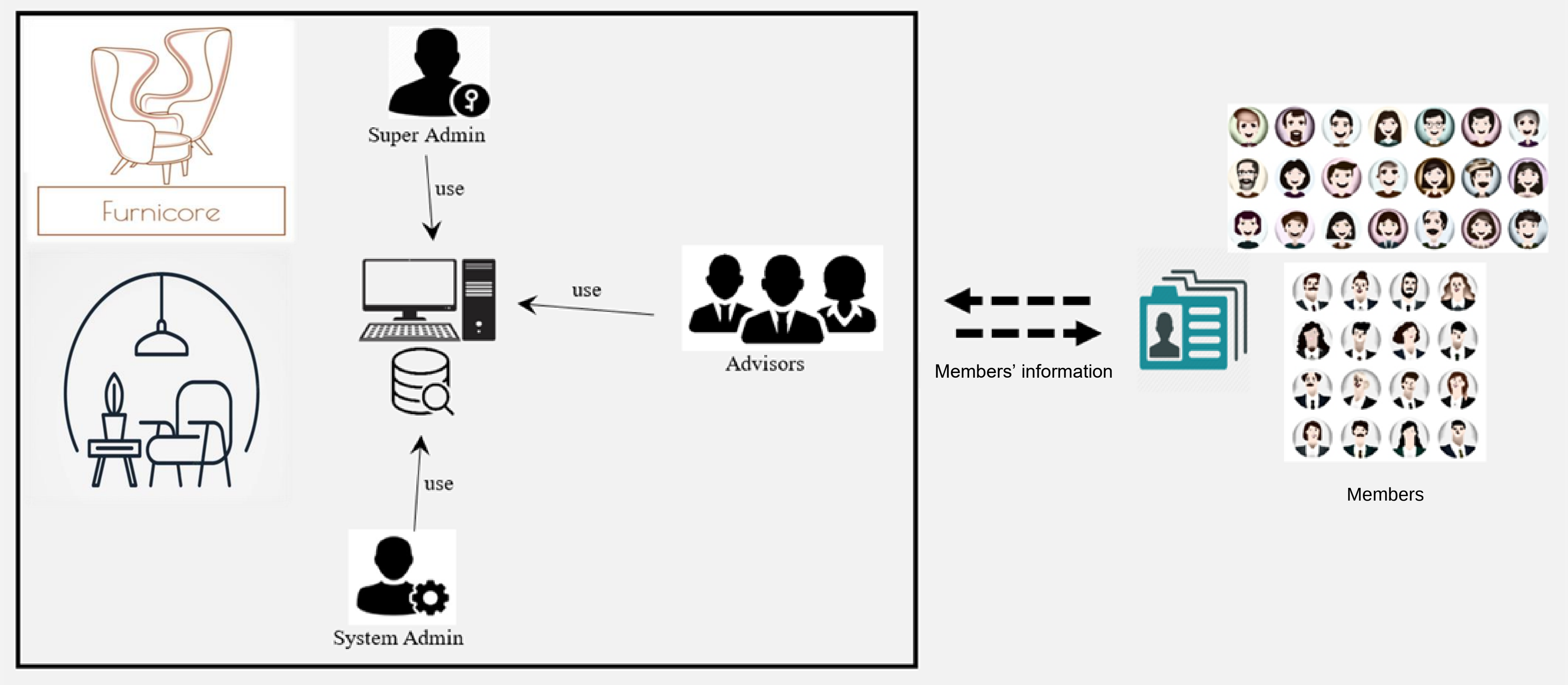


Figure 1. Overview of the System

User Interface

The minimum requirement for the user interface is a console-based interface with the possibility of menus or options to be chosen by the user.

The system must have a user-friendly (easy, efficient, and enjoyable) interface to allow the users (super-admin, system-admins, or advisors) to perform their functions, easily and smoothly.

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 member" 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 system is to store and manage the information of the members in the system. In addition, the system needs to store information of the users of the system.

For this purpose, you need to implement the database using SQLite library in Python sqlite3.

Note that the sensitive data, including usernames, passwords, and members' phones and addresses must be encrypted in the database.

Stakeholders, Users, Authorization, Functions and Accessibility Levels

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

1. **Members**

   Members are not the users of the system and have no role or function in the application. The only relationship between members and the system is that their information is recorded and stored in the system by a company advisor (System admin and super admin should be also able to manage members' data in the database).

2. **Advisors**

   Company advisors are employees of the company who are in direct contact with the members. They process the requests of the members. Hence, they need to be able to manage the member's information and data. For this purpose, when a new client requests for membership, an advisor needs to register the client's information in the system. So, the minimum required functions of an advisor in the system are summarized as below:

   - To update their own password
   - To add a new member to the system
   - To modify or update the information of a member in the system
   - To search and retrieve the information of a member

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 clients or members. However, for security reasons, they should be able to perform all the functions of advisors, if needed. The minimum required functions of an administrator are listed below:

   - To update their own password
   - To check the list of users and their roles

- To define and add a new advisor to the system
- To modify or update an existing advisor’s account and profile
- To delete an existing advisor’s account
- To reset an existing advisor’s password (a temporary password)
- To make a backup of the system and restore a backup
- To see the logs file(s) of the system
- To add a new member to the system
- To modify or update the information of a member in the system
- To delete a member's record from the database (note that an advisor cannot delete a record, but can only modify or update a member’s information)
- To search and retrieve the information of a member

4. **Super Administrator**

Super administrator is simply the owner or the manager of the company. The manager needs a super admin password through which can define a system administrator. Although the main function of the super admin is to define system admin(s), and leave the system to them; however, they **should be able to perform all possible functionalities of the lower-level users** (i.e., system admin and advisor).

In this assignment, to make it easier for your teacher to test and assess your work, a super admin must be hard coded with **username: superadmin**, **password: Admin321!**

Note that we know this is not a good development practice in terms of the quality and security of the system, but this is only to enable your teacher to easily test your system using this predefined hardcoded username and password.

The minimum required functions of a super administrator are listed below:

- To check the list of users and their roles
- To define and add a new advisor to the system
- To modify or update an existing advisor’s account and profile
- To delete an existing advisor’s account
- To reset an existing advisor’s password (a temporary password)
- To define and add a new admin to the system
- To modify or update an existing admin’s account and profile
- To delete an existing admin’s account
- To reset an existing admin’s password (a temporary password)
- To make a backup of the system and restore a backup (members information and users’ data)
- To see the logs file of the system
- To add a new member to the system
- To modify or update the information of a member in the system
- To delete a member's record from the database (note that an advisor cannot delete a record, but can only modify or update a member’s information)
- To search and retrieve the information of a member

Note 1: Advisors and system admins should have profiles, in addition to their usernames and passwords. Their profiles contain first name, last name, and registration date.

Note 2: The search function must accept any data field as a search key (member ID, first name, last name, address, email address, and phone number). It must also accept partial keys. For example, a user can search for a member with a name “Mike Thomson” and member ID “5223287425” by entering any of these keys: “mik”, “omso”, or “2232”, etc.

Log

The system should log all activities. All suspicious activities must be flagged, and the system needs to produce an alert for unread suspicious activities, once a system administrator or super administrator is logged in to the system. Log file(s) must be encrypted and should be only readable through the system interface, by system administrator or super admin. 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.	Username	Date	Time	Description of activity	Additional Information	Suspicious
1	john_m_05	12-05-2021	15:51:19	Logged in		No
2	superadmin	12-05-2021	18:00:20	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	superadmin	12-05-2021	18:08:02	User is deleted	User “mike12” is deleted	No
...	...	...	...	...	...	...

Note that the table 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.

Encryption

As mentioned before, all sensitive data in the database, including usernames, passwords, and members phones and addresses, as well as log data must be encrypted. For this encryption, you can employ a simple encryption algorithm, such as Caesar cipher, Vigenère cipher, etc. It is not allowed to use any third-party library or module, and you must implement it yourself.

However, if you are willing (and know how) to use a hash function for password, you are allowed to use a third-party library for this purpose (only for the hash function).

Backup

The system administrator and super administrator should be able to create a backup of the system and restore the system from a backup. This backup must include the database (users and members information) and the log file(s) and should be in **zip** format. Note that the log files and sensitive data in the DB file must be already encrypted, and no additional encryption is needed when you are creating the backup zip file.

Usernames and Passwords

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

- **Username**:
  - must be unique and have a length of at least 6 characters
  - must be no longer than 10 characters
  - must be started with a letter
  - can contain letters (a-z), numbers (0-9), underscores (_), apostrophes ('), and periods (.)
  - no distinguish between lowercase or uppercase letters

- **Password**:
  - must have a length of at least 8 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

User Manual

A User Manual should be provided to explain how to use the system, for all types of users. Any information needed for the users of the system should be clearly provided in this document, including commands, shortcuts, usernames and passwords (if any), etc. This manual may contain screenshots, examples, diagrams, if needed. This information may also be used by your teacher to run, test, and assess the system.

Do not suppose that the teacher will explore your code to find out how to run the system.

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.

Students will receive feedback from the teachers via Google Classroom, if needed.

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 as Satisfactory (L2 or L3)**, and
- **You must get C3 and C6 as Satisfactory (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

Assignment will be evaluated according to the marking scheme, below.

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. PWs are longer than 8 characters and are 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 unreadable file by a text editor. There is no bug or major issue. 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.	Input Validation is complete for all input types and does not allow bypassing. Whitelisting is used. 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.	Logging, Backup and Restore are partially implemented. There are some bugs.	Logging, Backup and Restore are fully implemented. All suspicious incidents are logged. However, it could still be improved.	Logging, Backup and Restore are complete, and there is evidence of good practices.

Criteria	Unsatisfactory	Satisfactory
	L0 (0 point)	L1 (1 point)
C3	The system is not secure against SQL Injection.	The system is secure against SQL Injection
C6	Students cannot properly run and demonstrate the system, or cannot explain it, or answer the technical questions.	Students can properly run and demonstrate the system and provide relevant answers to the majority of the technical questions.

Submission

Deliverable

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

studentnumber1_studentnumber2.zip

The zip-file must contain:

1. A **pdf document**, called **ffs.pdf**, containing:
   a. **Names** and **student numbers** of the team (maximum **2** students per team),
   b. A **User Manual** explaining, with examples, how to use the system. **This is needed for your teacher to test and assess your code.**
2. A directory called **src**, containing all the **code files** and the **data files**, including one main file **ffs.py**. Starting the system should be done by running **ffs.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 **hash** library of your choice (if needed).
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 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. 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. **Feedback will be given to the team leader only**, who will then communicate it to the other team members.
7. You are **<u>not allowed</u>** to work together with someone who has a different group. Here we mean the group in your schedule. (e.g., a student from group INF2A can have a teammate only from group INF2A).

How to submit?

- **Students** can only submit it via the appropriate channels in MS Teams. (Submission slots will be available later).
  Please note that submission through chat or email is not accepted. If you confront any problem in submission, contact your teacher to assist you. As the submission dates (first chance and second chance) are 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 work 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 **19 June 2022.**

Submission deadline for the **Second Chance** will be announced later.

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 on 13 June, with some time slots.

  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.

  Please check this document again on teaching week 6 and 7 to see the link for the presentation form.