SECP2613: SYSTEM ANALYSIS AND DESIGN

System Documentation (SD)

Software Testing Documentation (STD)

Youth Ventures Student Portfolio Management System (StuPort)

Version 3.0

4th January 2024

Faculty of Computing

Lecturer:

DR. MUHAMMAD IQBAL TARIQ BIN IDRIS

Prepared by:

KOH SU XUAN A22EC0060 (SECTION 02 GROUP EXPLORER)

Revision Page

a. **Overview**

System Documentation (SD) aims to provide a comprehensive guide for:

I. Understanding, developing and maintaining a system,
II. Facilitating effective communication,
III. Knowledge sharing,
IV. Troubleshooting,
V. Compliance with standards,

ultimately ensuring system stability, reliability and future evolution. It serves as an important resource that fosters collaboration among clients, guiding development and supporting system operations while enabling efficient adaptation and enhancement over time.

Version 3.0 of SD covers System Testing Documentation (STD) of Youth Ventures Student Portfolio Management System (StuPort) which provides a detailed visualization of the requirements-based testing including functional requirements and non-functional requirements, black-box testing and white-box testing in order to provide a comprehensive approach to testing software functionality, behavior, and internal structure.

b. **Target Audience**

The targeted audience of SD of Youth Ventures Student Portfolio Management System (StuPort) are :

1. Client - Youth Ventures Asia : The SD serves as a reference for our client, Youth Ventures Asia to comprehend StuPort's functionalities, scope, limitations and expected outcomes. It helps in managing expectations and aligning the final product with their requirements.

2. Youth Ventures Asia's Clients - Partners, Organizers, Students, Lecturers etc. : SD provides insights into the functionalities, features and user interactions within StuPort,

ensuring that the StuPort system caters to the needs of Youth Ventures Asia's clients by offering a clear understanding of the system's capabilities and benefits.

3. Project Manager : With SD, project manager oversees the development process, understanding the project's scope, managing timelines and ensuring that the project aligns with the established requirements and goals.

4. Designer : Designer relies on SD to understand user requirements, functionalities and constraints by insights into the system's layout, user interface elements and user experience expectations provided to create intuitive and user-friendly interfaces for StuPort.

5. Database Administrator (DBA) : SD outlines the data requirements, storage structures and interactions with the database. It assists DBA in understanding the data models, relationships and constraints necessary for designing, implementing and maintaining the StuPort database efficiently.

6. Quality Assurance (QA) Tester : QA testers utilize SD to create test cases, scenarios and expected outcomes based on specified requirements, helping in validating StuPort's functionalities, ensuring that the software meets quality standards and performs as expected.

7. System Analyst : System Analysts refer to SD to comprehend the system's architecture, functionalities and dependencies. It aids in analyzing system requirements, identifying potential risks and proposing suitable solutions for StuPort.

8. Documentation Specialist : SD serves as a foundational resource in structuring and organizing comprehensive documentation for StuPort. It provides essential details, terminology and information necessary for creating user manuals, guides and other supplementary documents.

9. Developer : Developers rely on SD for detailed technical specifications, architectural diagrams, coding guidelines and integration requirements. It guides them in implementing StuPort's functionalities while adhering to the defined standards and specifications.

c. **Project Team Members**

Member Name	Role	Task	Status
KOH SU XUAN	- Database Administrator (DBA) - Quality Assurance (QA) Tester	Section A	Complete
		Section B	
		Section C	

d. **Version Control History**

Version	Primary Author(s)	Description of Version	Date Completed
3.0	KOH SU XUAN	Completed Section A	16/02/2024
3.0	KOH SU XUAN	Completed Section B	17/02/2024
3.0	KOH SU XUAN	Completed Section C	17/02/2024

Table of Contents

Section A Requirements-based Testing 1

A1 Functional Requirements 1

A1.1 Test Requirements (TR) 1

A1.2 Test Cases 2

A2 Non-Functional Requirements 4

A2.1 Test Requirements (TR) 4

A2.2 Test Cases 4

A3 Summary 5

Section B Black-box Testing 6

B1 Object Class 6

B1.1 Equivalence Partitioning and Boundary Value Analysis 6

B1.2 Test Cases 8

B2 Summary 9

Section C White-box Testing 10

C1 Methods Class 10

C1.1 Flow Graph 11

C1.2 Cyclomatic Complexity 12

C1.3 Test Cases 12

C2 Summary 13

Section A: Requirements-based Testing

A1 Functional Requirements

The chosen use case that I owned which has been written in the System Documentation is UC006: Edit Profile under M002: Profile Module of Youth Ventures Student Portfolio Management System (StuPort).

When a User, whether a Student or a Lecturer, accesses the profile section, they are presented with their current profile information. After selecting the “Edit Profile” button and upon making the desired changes, the User clicks the "Save" button and triggers a validation process by the system. If any of the modifications are considered to be invalid, an error message prompts the User to make the necessary changes. Conversely, valid updates are securely stored in the database. Following this process, the User is redirected to the profile section to review the updated information. In the case of a Master Administrator, to edit a student or lecturer's profile, they navigate directly to the respective user's profile table within the database. Subsequent modifications are made and upon selecting "Save", the system promptly updates and stores the edited profile information in the database. A successful message is displayed to signify the completion of the operation.

A1.1 Test Requirements (TR)

Table 1. List of Functional Test Requirements

Use Case (UC)	**TR ID**	**Test Requirements**
UC006: Edit Profile	TR**001**	Validate that the system properly displays the current profile information for editing.
	TR**002**	Validate that the system allows the user to modify profile details.
	TR**003**	Validate that the system properly saves valid updates to the profile in the database.
	TR**004**	Validate that the system displays an error message for invalid changes and prompts the user to correct them.

	TR**005**	Validate that the system redirects the user to the profile section to view the latest information after saving the updates.

A1.2 Test Cases

Table 2. List of Functional Test Cases

TR ID	**Case No.**	**Data Entered**	**Expected Result**
TR001	**TC$_{\text{TR001}}$_01**	Navigate to the profile section and select the "Edit Profile" button.	The user should be able to access the edit profile section.
	TC$_{\text{TR001}}$_02	Verify that all existing profile information is displayed in the edit profile form.	All existing profile information should be visible and editable in the edit profile form except for fixed information such as email.
	TC$_{\text{TR001}}$_03	Attempt to edit a specific profile detail and confirm it reflects in the edit profile form.	The edited profile detail should be updated in the edit profile form.
TR002	**TC$_{\text{TR002}}$_01**	Change the address in the profile to a new valid address.	The address should be successfully updated to the new valid address.
	TC$_{\text{TR002}}$_02	Update the phone number with a new valid phone number.	The phone number should be updated to the new valid phone number.
	TC$_{\text{TR002}}$_03	Try to modify the course information and confirm the change is allowed.	The course information should be successfully modified.
TR003	**TC$_{\text{TR003}}$_01**	Save the edited profile with valid changes.	The system should save the edited profile with valid changes without any errors.

	TC_{TR003}_02	Confirm that the changes made are reflected in the profile section after saving.	After saving, the changes made should be immediately visible in the profile section.
	TC_{TR003}_03	Check the database directly to ensure that the updated profile information is stored correctly.	Directly querying the database should reveal that the updated profile information is stored correctly.

A2 Non-Functional Requirements

The chosen non-functional requirement that has been written in System Documentation is Security. The Youth Ventures Student Portfolio Management System (StuPort) implements robust security measures to protect sensitive user data. Based on users' request, the StuPort system ensures data confidentiality, integrity and availability, with features such as role-based access control and encryption.

A2.1 Test Requirements (TR)

Table 3. List of Non-Functional Test Requirements

Non-functional	**TR ID**	**Test Requirements**
Security	TR001	Ensure that the system implements role-based access control to restrict unauthorized access to sensitive student data.
	TR002	Validate that the system encrypts sensitive user data stored in the database to maintain data confidentiality.
	TR003	Verify that the system logs and monitors user activities to detect and prevent unauthorized access or suspicious behavior.

A2.2 Test Cases

Table 4. List of Non-Functional Test Cases

TR ID	**Case No.**	**Data Entered**	**Expected Result**
TR001	**TC_{TR001}_01**	Try to access student profile information without logging in.	Access should be denied and the system should prompt the user to log in.
	TC_{TR001}_02	Log in with a student account and try to access administrative settings.	Access should be denied and the system should display a message indicating insufficient permissions.

	TC$_{\mathbf{TR001}}$_03	Log in with a master administrator account and attempt to edit student profile information.	Access should be granted and the master administrator should be able to edit student data as intended.

A3 Summary

The requirements-based testing strategy is appropriate for system-level testing. This strategy involves testing the system's functionality against the specified requirements to ensure that it meets the intended objectives. Since requirements-based testing focuses on verifying that the system functions as expected based on the defined requirements, it is well-suited for testing the overall behavior and performance of the system. Additionally, it helps ensure that the system meets user needs and expectations while adhering to defined specifications. Thus, the requirements-based testing strategy is appropriate for system-level testing of Youth Ventures Student Portfolio Management System (StuPort).

Section B: Black-box Testing

(beta)

B1 Object Class

Two chosen object classes that have been written in System Documentation are lecturer and profile. Object class lecturer contains attributes such as lecturer's id, email, full name, telephone number, address, institution, gender race and age while profile which only hold by two types of user, lecturer and student, consists of attributes such as user's profile id, email, name, gender, race, age, date of birth, profile image, position, headline, about, country and city state.

B1.1 Equivalence Partitioning and Boundary Value Analysis

Table 5. Equivalence Partition and Input Range

Object class	**Attributes**	**Equivalence Partition and Input Range**	
		Equivalence Partition	**Input Range**
lecturer	l_id	Valid positive integers.	Any integer within the valid range of 11 digits.
	l_email	Valid email format.	Non-empty string up to 255 characters.
	l_fName	Non-empty string.	Non-empty string up to 255 characters.
	l_telephone_no	Valid telephone number format.	Any integer within the valid range of 11 digits.
	l_address	Non-empty string.	Non-empty string up to 255 characters.
	l_institution	Non-empty string.	Non-empty string up to 255 characters.
	l_gender	Valid gender values.	Female, Male.
	l_race	Non-empty string.	Non-empty string up to 255 characters.

	l_age	Valid positive integers within a reasonable range.	Any integer within the valid range of 2 digits.

Object class	**Attributes**	**Equivalence Partition and Input Range**	
		Equivalence Partition	**Input Range**
profile	p_id	Valid positive integers.	Any integer within the valid range of 11 digits.
	p_email	Valid email format.	Non-empty string up to 20 characters.
	p_name	Non-empty string.	Non-empty string up to 50 characters.
	gender	Valid gender values.	Female, Male.
	race	Non-empty string.	Non-empty string up to 20 characters.
	age	Valid positive integers within a reasonable range.	Any integer within the valid range of 2 digits.
	dob	Valid date formats.	Any valid date value.
	profileimage	Valid file path for an image.	Any valid path value.
	position	Non-empty string.	Student, Lecturer.
	headline	Valid text content.	Any text content.
	about	Valid text content.	Any text content.
	country	Non-empty string.	Non-empty string up to 50 characters.
	citystate	Non-empty string.	Non-empty string up to 50 characters.

B1.2 Test Cases

Table 6. Object Class Based Test Cases

Object name: lecturer

Method name: createLecturer()

Case No.	**Equivalence Class**	**Representative (BVA)**	**Expected Result**
TC001	Valid email format.	Valid email format. Eg. yvlect@gmail.com	Valid email address accepted.
TC002	Valid telephone number format.	Valid telephone number format. Eg. 01234567890	Valid telephone number accepted.
TC003	Valid gender values.	Male/ Female Eg. Male	Valid gender values accepted.
TC004	Valid positive integers within a reasonable range.	Valid positive integers within a reasonable range. Eg. 30	Valid age accepted within the specified range.

Object name: profile

Method name: editProfile()

Case No.	**Equivalence Class**	**Representative (BVA)**	**Expected Result**
TC001	Valid email format.	Valid email format. Eg. yvlect@gmail.com	Valid email address accepted.
TC002	Valid gender values.	Male/ Female Eg. Male	Valid gender values accepted.

TC003	Valid positive integers within a reasonable range.	Valid positive integers within a reasonable range. Eg. 30	Valid age accepted within the specified range.
TC004	Valid date formats.	Valid date formats. Eg. 2000-02-28	Valid date of birth accepted.
TC005	Valid file path for an image.	Valid file path. Eg. images/users/yvlect@gmail.com/65a7c743bbe563.25840094.png	Valid profile image path accepted.

B2 Summary

The black-box testing strategy is appropriate for system-level testing. Black-box testing focuses on verifying the functionality of the system without looking at its internal structure or implementation details. Since black-box testing evaluates the system based on its specified requirements and expected behavior, it helps ensure that the Youth Ventures Student Portfolio Management System (StuPort) meets user needs and client, Youth Ventures Asia’s needs and functions correctly in various scenarios. Additionally, black-box testing allows testers to identify potential defects or discrepancies between the system's actual behavior and its intended functionality. Therefore, executing black-box testing at the system level helps validate the overall behavior and performance of the StuPort system.

Section C: White-box Testing

(alpha)

C1 Methods Class

The chosen method from the class(es) that I owned which has been written in System Documentation is the editProfile() method from the Profile Module.

Table 7. Methods Class

Entity Name	Profile
Method Name	editProfile
Input	Modified Profile Information
Output	Latest Profile Information
Algorithm	1. Start 2. Retrieve the profile information from ProfileDA. 3. Display the current profile information. 4. User select the “Edit Profile” button. 5. Edit and modify the profile details. 6. Select the “Save” button. 7. Validate the edited profile details. 8. If the profile details is valid: a. Update the edited profile details. b. Store the updated profile details in ProfileDA. c. Display the latest profile information. 9. Else if the profile details is invalid: a. Display error message. b. Handle the correction of profile details appropriately. c. Resubmit the edited profile details. d. Continue with Algorithm Step 7. 10. End

C1.1 Flow Graph

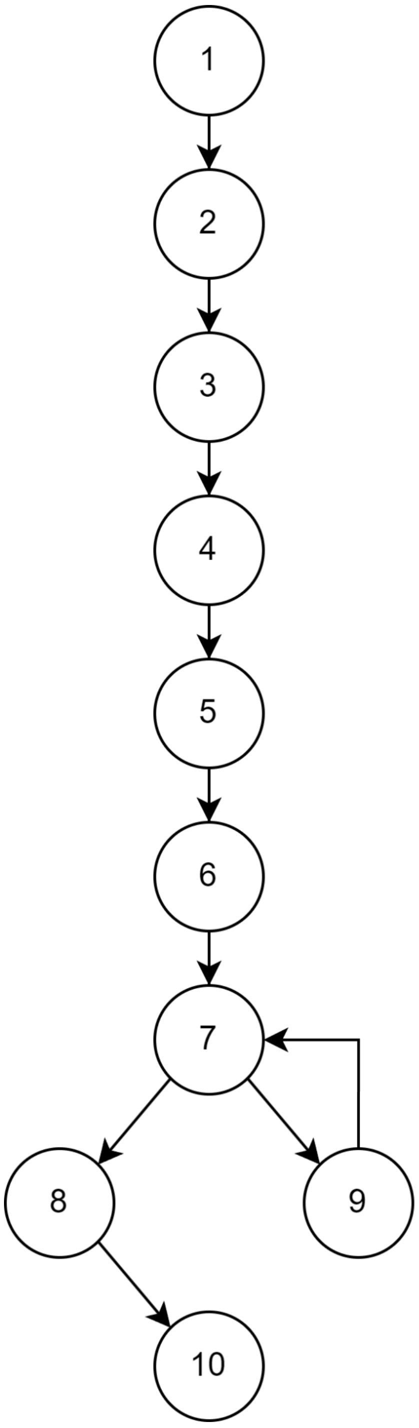

Figure C1.1.1: Flow Graph of editProfile() Method Class

C1.2 Cyclomatic Complexity

Formula 1 : V(G) = E - N + 2P
E (Number of Edges) = 10,
N (Number of Nodes) = 10,
P (Number of Connected Components) = 1.
∴ V(G) = 10 - 10 + 2(1) = 2

Formula 2 : V(G) = L - N + 2
L (Number of Edges) = 10,
N (Number of Nodes) = 10.
∴ V(G) = 10 - 10 + 2 = 2

Formula 3 : V(G) = P + 1
P (Number of Connected Components) = 1.
∴ V(G) = 1 + 1 = 2

The cyclomatic complexity for the editProfile() method is calculated using three different formulae. The resulting complexity is 2, indicating that there are two independent paths through the method.

C1.3 Test Cases

Table 8. Independent Path Based Test Cases

Case No.	**Independent Path**	**Data* for Test Cases**	**Expected Result**
TC001	1-2-3-4-5-6-7-8-10	Modified profile information with valid data.	Latest profile information is displayed after successful editing.
TC002	1-2-3-4-5-6-7-9-7-8-10	Modified profile information with invalid	Error message is displayed for

Case No.	Independent Path	Data* for Test Cases	Expected Result
		data, then corrected with valid data.	invalid data, profile details are corrected by the user and the latest profile information is displayed after successful editing.

C2 Summary

The white-box testing strategy is appropriate for unit testing. This strategy involves examining the internal logic and structure of the code to ensure its correctness and efficiency. In this case, white-box testing would be suitable for testing the "editProfile" method as it allows us to verify the implementation details of the algorithm and ensure that it behaves as expected under different conditions. Additionally, white-box testing helps identify any potential errors or bugs in the code, making it an essential part of the testing process of a developed system. Thus, the white-box testing strategy is appropriate for unit testing of the Youth Ventures Student Portfolio Management System (StuPort).