Introduction to Software Engineering

TÀI LIỆU KIỂM THỬ

Yêu cầu nhóm sinh viên hoàn thành tài liệu kiểm thử cho đồ án đã được giao theo biểu mẫu đính kèm.

Bộ môn Công nghệ phần mềm
Khoa Công nghệ thông tin
Đại học Khoa học tự nhiên TP HCM

MỤC LỤC

1 Member Evaluation Table **2**

1 Test plan **3**

1. Testing Objectives 3
2. Scope of Testing 4
3. Testing Techniques to be Applied 7
4. Completion Criteria 9

2 Test case **10**

2.1 Test Case List 10

2.2 Test Case Specifications 21

2.2.1 Test case 1 21

2.2.2 Test case 2 22

2.2.3 Test case 3 23

2.2.4 Test case 4 23

2.2.5 Test case 5 24

2.2.6 Test case 6 24

2.2.7 Test case 7 25

2.2.8 Test case 8 26

2.2.9 Test case 9 26

2.2.10 Test case 10 27

2.2.11 Test case 11 28

2.2.12 Test case 12 28

2.2.13 Test case 13 29

Testing Documentation

The document focuses on the following topics:

- Creating software testing documentation.
- Completing software testing documentation with the following contents:
  - Test plan
  - Test cases
- Understanding software testing documentation

1 Member Evaluation Table

MSSV	Họ Tên	% đóng góp (tối đa 100%)	Chữ ký
22127389	Nguyễn Phúc Thành	25	Thành
22127088	Phạm Quang Duy	25	Duy
22127060	Lê Hoàng Đạt	25	Đạt
22127270	Nguyễn Quang Minh	25	Minh

1 Test plan

[Trình bày kế hoạch kiểm thử dự án, cần nêu rõ nhóm dự định sẽ áp dụng những kỹ thuật kiểm thử nào, sẽ thực hiện trên các đối tượng (chức năng, tài liệu) nào của hệ thống]

1. Testing Objectives

- **Accuracy Verification**: Ensure that all functionalities, such as opening accounts, transactions, and reporting, work correctly according to the business rules.
- **Requirement Fulfillment**: Verify that all specified requirements are implemented accurately and that the system meets the expected user needs.
- **Error Detection**: Identify and resolve any defects or inconsistencies in the application to enhance the system's reliability and stability.
- **Functional Testing**:
    - Ensure that each feature of the application functions correctly according to the specifications. This includes validating user interactions, data processing, and system responses to both valid and invalid inputs.
    - Verify that all user scenarios and business logic are accurately implemented, ensuring that the application meets its intended purpose without functional errors.
- **Non-Functional Testing**:
    - **Performance Testing**: Assess the application’s responsiveness, speed, and stability under various load conditions, ensuring it performs well under peak usage.

- **Security Testing**: Identify and mitigate any potential vulnerabilities in the system to protect against unauthorized access, data breaches, and other security threats.
- **Usability Testing**: Evaluate the application's user interface and user experience to ensure it is intuitive, user-friendly, and meets the needs of the target audience.
- **Compatibility Testing**: Ensure that the application works across different devices, operating systems, and browsers, providing a consistent experience to all users.
- **Reliability Testing**: Confirm that the application operates consistently and accurately over time, with a low rate of failure or bugs.

2. Scope of Testing

- **Main Functionalities to be Tested**:
  - **Open a Savings Account**:
    - Test the process of opening a savings account, including verifying required fields such as customer information, account type, and initial deposit. **Including both early checking and late checking.**
    - Validate that the application enforces the minimum deposit money
    - Choose the correct term with appropriate interest rate.
    - Securing the idea that 1 customer can have multiple accounts.
  - **Create Deposit Slip**:
    - Test the creation of deposit slips, ensuring correct input of account details, deposit amount, and adherence to the minimum deposit rules.
    - Verify that deposits are accurately reflected in the customer's account balance.

- Verify that the slip is appropriate with the regulation such as only deposit to no period term.
- Securing the updation of new balance on the account.
- Storing information on deposit slips.

- **Create Withdrawal Slip**:
  - Test the withdrawal process, including restrictions such as minimum holding periods and rules for withdrawing from term and non-term accounts.
  - Confirm that the withdrawal slip reflects the correct deduction from the account balance and that interest calculations are accurate.
  - Securing that the account will close as soon as the balance is reset to 0.
  - Securing the regulation for minimum withdraw day.
  - Securing the correct accounting of interest money to maturity
- **Search for Savings Account Information**:
  - Test the search functionality for retrieving savings account details by various criteria (e.g., account number, customer name).
  - Ensuring the display of the account has been closed or open.
  - Ensuring the display of the transaction for each account.
  - Ensuring the searching of multiple types like name of the owner of account or account id.
  - Ensure that the system displays accurate and up-to-date information, including account balance, transaction history, and account status.
- **Generate Daily and Monthly Reports**:
  - **Daily Reports**: Test the generation of daily operational reports, including total deposits, withdrawals, and net balance changes.

        - **Monthly Reports**: Validate monthly reports summarizing account openings/closures and overall performance, ensuring that the data is correctly aggregated and presented.
    - **Modify Regulations**:
        - Test the ability to modify system regulations, such as changing minimum deposit amounts, interest rates, and other key parameters.
        - Ensure that these changes are reflected accurately **across all relevant functionalities** without disrupting existing account operations.
        - Ensuring the consistency of the system when changing regulation. (older account with interest still operate consistently)
- **Testing Documents**:
    - **BM1: Savings Account Form**: Ensure that all fields in the form are validated and saved correctly.
    - **BM2: Deposit Slip**: Validate the correct processing of deposits and ensure compliance with deposit rules.
    - **BM3: Withdrawal Slip**: Test adherence to withdrawal regulations and correct balance updates.
    - **BM4: Savings Account List**: Ensure that the list accurately displays all accounts and their details.
    - **BM5: Reports**:
        - **Daily Reports**: Verify that daily transactions are accurately captured and summarized.
        - **Monthly Reports**: Ensure that monthly summaries are comprehensive and error-free.
    - **BM6: Change Regulation**: Test the system's ability to update and enforce new regulations without affecting existing data.

3. Testing Techniques to be Applied

- **Functional Testing**:
  - **Objective**: Validate that each function works according to its specification.
  - **Approach**:
    - **Example**: Create test cases for the "Open a Savings Account" functionality. This includes verifying that the system correctly captures customer details, selects the appropriate account type, and validates the minimum deposit amount.
    - Execute tests to ensure the correct processing of user inputs and expected outputs, such as successful account creation with valid data and appropriate error messages for invalid inputs.
    - Use both valid and invalid inputs to test the system's robustness, such as entering an invalid ID number or a deposit amount below the required minimum.
- **Integration Testing**:
  - **Objective**: Ensure that the integration between different modules (e.g., deposit and withdrawal functions) works seamlessly.
  - **Approach**:
    - **Example**: Test the flow of data between the "Create Deposit Slip" and "Update Account Balance" modules. After a deposit is made, verify that the account balance is updated correctly and reflected in the account summary.
    - Validate that changes in one module are reflected correctly across the system, such as ensuring that a withdrawal accurately decreases the account balance and that the updated balance is correctly displayed in account inquiries.

- **Regression Testing**:
  - **Objective**: Ensure that new changes or bug fixes do not introduce new issues.
  - **Approach**:
    - **Example**: After implementing a new feature to change the interest rate for savings accounts, re-run previously executed test cases related to account creation, deposit, and withdrawal functionalities to ensure that existing functionalities remain unaffected.
    - Focus on areas of the system that are most likely to be impacted by changes, such as the calculation of interest during withdrawals after an interest rate change.
- **System Testing**:
  - **Objective**: Validate the overall behavior of the system in a real-world scenario.
  - **Approach**:
    - **Example**: Perform end-to-end testing that includes the entire process from opening a savings account, making deposits, performing withdrawals, and generating a monthly report. Ensure that all data is processed correctly and that the system behaves as expected throughout the user journey.
    - Test the system's performance, security, and usability in various environments, such as different browsers or devices, to ensure a consistent user experience.
- **User Acceptance Testing (UAT)**:
  - **Objective**: Ensure the system meets user needs and business requirements.
  - **Approach**:

        - **Example**: Involve end-users in testing key functionalities, such as generating monthly reports and changing account regulations. Gather feedback on usability and correctness, particularly focusing on how easy it is for users to understand and navigate the system.
        - Make adjustments based on user feedback before the final release, such as improving the clarity of report formats or enhancing the navigation flow for modifying account details.
- **Unit Testing**:
    - **Objective**: Validate that individual components (units) of the application, such as functions, methods, or classes, operate correctly in isolation.
    - **Approach**:
        - **Example**: Develop and execute unit tests for the function that calculates interest on savings accounts. This would involve testing different scenarios, such as calculating interest for different account types (e.g., 3-month, 6-month) and ensuring that the calculation logic handles edge cases correctly.
        - Ensure that each unit meets its design specifications and behaves as expected before integrating it with other units, such as verifying that the interest calculation function correctly interacts with the overall account balance update process.

4. Completion Criteria

- **Test Case Execution**: All test cases must be executed, and critical functionalities should pass without any defects.

- **Defect Resolution**: All critical and high-priority defects must be resolved before release. Medium and low-priority defects should be documented and addressed in future updates.
- **Requirement Fulfillment**: Ensure that all functional and non-functional requirements are fully implemented and tested.
- **Sign-off**: Obtain sign-off from stakeholders, including the testing team, project managers, and end-users, confirming that the system is ready for production.

2 Test case

2.1 Test Case List

[Liệt kê tên các test case, các đối tượng test và diễn giải]

STT	Tên test case	Đối tượng test	Ý nghĩa
1	Checking insert term	change regulation	Testing whether the term has been inserted down to database and update to other form or not
2	Checking interest rate when insert new term	change regulation	Check if interest rate is numeric or not and report error if it is not
3	Checking if the term is already taken or not	change regulation	Check if we insert the same term twice

			or insert the already existence form
4	Check if the minimum deposit has been updated or not	change regulation	check if the other form has updated to new minimum deposit or not.
5	check the minimum deposit has been wrote down to DB	change regulation	check if the minimum deposit has been updated down to DB
6	Check Minimum deposit day has the correct format	change regulation	check if it is numeric or not if it is not report error
7	Check current deposit day	change regulation	Check if the form display the correct current minimum deposit day
8	Check minimum day format	change regulation	check if the minimum day format is inputted correctly with numeric form if not report error
8	check minimum withdraw day write down to DB	change regulation	check if the minimum withdrawal day has been updated to DB yet.
9	check new minimum withdraw day has updated to other form yes or not	change regulation	check if the minimum withdraw day has integrated

			as soon as it is changed to other form
10	Check new interest rate has been write down to DB	change regulation	Check if the new interest rate has been write down to DB and updated to other form
11	Check whether all the terms option is displayed	change regulation	Check whether all the terms option is displayed to choose and change interest rate
12	Check whether the customer name contains numbers.	Create Account	This test ensures that the customer's name only includes alphabetic characters, as numbers in a name are generally not valid.
13	Check whether the ID contains letters.	Create Account	This test verifies that the ID (CMND/CCCD) consists solely of numeric digits, as IDs should not contain letters.
14	Check whether the ID has enough digits.	Create Account	This test checks that the ID meets the required length, ensuring it has the

			correct number of digits to be valid.
15	Check whether the deposit amount contains letters.	Create Account	This test confirms that the deposit amount field contains only numeric values and no letters, ensuring proper data entry.
16	Check whether the deposit term type is valid (can be adjusted for user selection).	Create Account	This test ensures that the selected deposit term type is one of the valid options available for users.
17	Check whether the deposit amount is less than the minimum required amount.	Create Account	This test checks that the deposit amount meets or exceeds the minimum required value, ensuring compliance with deposit rules.
18	Check whether the account opening date is before or after the current date.	Create Account	This test verifies that the account opening date is valid by ensuring it is not set to a date in the past or far in the future.
19	Check whether the customer name matches the ID recorded in the database.	Create Account	This test confirms that the name provided matches

			the name associated with the ID in the database, ensuring consistency and correctness.
20	Check whether the address matches the ID recorded in the database.	Create Account	This test ensures that the provided address corresponds with the address linked to the ID in the database, verifying identity and preventing fraud.
21	Check whether the account information exists.	Deposit money	This test verifies that the account information provided is present in the system. It ensures that transactions or inquiries are made on valid and existing accounts.
22	Check whether the deposit date is before the account opening date.	Deposit money	This test ensures that the deposit date is not earlier than the account opening date, maintaining logical consistency in the transaction timeline.

23	Check whether the deposit date is before the current date.	Deposit money	This test confirms that the deposit date is not set in the future, ensuring that all deposit transactions are recorded with valid dates.
24	Check whether deposits are accepted only for the "no periods" savings type.	Deposit money	This test ensures that deposits are only allowed for the specific savings type "no periods," preventing invalid or unapproved deposit transactions for other savings types.
25	Check whether the account is already closed.	Deposit money	This test checks the status of the account to verify if it has been closed. It prevents further transactions or actions on accounts that are no longer active.
26	Check whether the account name and savings type are automatically displayed when the account number is entered.	Withdraw money	This test ensures that when a user enters the account number, the system automatically displays the associated account

			name and savings type to facilitate the withdrawal process.
27	Check whether a withdrawal amount field is available for non-term savings accounts.	Withdraw money	This test verifies that for non-term savings accounts, a field is available to input the withdrawal amount, allowing flexibility in the withdrawal process.
28	Check whether only a withdrawal button is available for term savings accounts.	Withdraw money	This test confirms that for term savings accounts, the user is only provided with a withdrawal button, indicating that the entire balance must be withdrawn.
29	Check whether the entire balance must be withdrawn for term savings accounts.	Withdraw money	This test ensures that for term savings accounts, users are required to withdraw the full amount, maintaining the terms and conditions of the account type.
30	Check whether the withdrawal amount is calculated with interest	Withdraw money	This test verifies that the interest is properly calculated on the withdrawal

	based on the number of rollovers.		amount, accounting for the number of times the term has been renewed.
31	Check whether the withdrawal amount for non-term savings accounts is based on the current balance.	Withdraw money	This test confirms that for non-term savings accounts, the withdrawal amount is correctly determined based on the available balance in the account.
32	Check whether interest is calculated after withdrawing the current balance.	Withdraw money	This test ensures that after withdrawing a portion of the balance, the interest on the remaining balance is recalculated according to the new total.
33	Check whether the withdrawal amount is less than the current balance.	Withdraw money	This test verifies that the withdrawal amount does not exceed the current balance, preventing overdraft or invalid transactions.

34	Check whether the account is closed after withdrawing the entire balance.	Withdraw money	This test ensures that the account is automatically closed when the entire balance has been withdrawn, maintaining proper account management.
35	Check whether the withdrawal date for non-term savings accounts is more than 15 days after the most recent deposit.	Withdraw money	This test confirms that withdrawals from non-term savings accounts are only allowed if at least 15 days have passed since the most recent deposit, in accordance with the account rules.
36	Check if sort on every colum work correctly	View Account	To check whether the column sorting running in the backend is correct. Check whether the column sorting is running correctly in the backend, specifically whether it is following the required order and sorting the correct column.

37	Check for defauts order	View Account	Check if the default sorting remains consistent after different sorting operations.
38	Check for correct information displayed on screen	View Account	Check if the account information stored in the database matches the information displayed on the page. Ensure that the information is consistent and hasn't been altered.
39	Check for huge number of account	View Account	Test saving a large number of accounts in the database to see if the page can handle and display this large volume of accounts effectively.
40	Check if search account work correctly	View Account	Check if searching by name or ID returns the correct results as expected, and after performing the search, verify whether the information remains intact and unchanged.

41	Check if look for account transaction work correctly	View Account Transaction	Check if the information of the person being searched matches the correct account associated with that person.
42	Check if information in account transaction is correct	View Account Transaction	
43	Check if information in account transaction is same with another page	View Account Transaction	Check if the account information retrieved during the transaction search matches the information displayed on other pages.
44	Check if transaction is correct	View Account Transaction	Check if transactions are properly saved in the database, and verify that the transaction amounts and other transaction details are correct.
45	Check if sort function in account transaction work correctly	View Account Transaction	Check if the sort functionality is working correctly, ensuring that the sorting aligns with the selected criteria

			and properly adjusts between ascending and descending orders as specified.
46	Check data integrity after sort	View Account Transaction	Check if the information remains consistent and unchanged after performing multiple sorts and reloading the data.
47	Check if the data is correct	Daily Report	Verify that the information for the selected date matches the data stored in the database.
48	Check if the data is interity	Daily Report	Check whether the information remains consistent after reloading the page or performing other operations.
49	Check if chart is drawn correcly	Daily Report	Check if the chart is synchronized with the information for that specific day.
50	Check if the data is correct	Monthly Report	Verify that the information for the selected date matches the data

			stored in the database.
51	Check if the data is interity	Monthly Report	Check whether the information remains consistent after reloading the page or performing other operations.
52	Check if chart is drawn correcly	Monthly Report	Check if the chart is synchronized with the information for that specific day.

2.2 Test Case Specifications

[Sinh viên chọn và trình bày đặc tả của vài (9-10) test case quan trọng nhất]

2.2.1 Test case 1

Test case	Checking insert term
Related Use case	*Adjust regulation*
Context	*As bank employee*
Input Data	*12 months*
Expected Output	*Updated to other form new term*
Test steps	1. *Click to change regulation* 2. *Choose insert term* 3. *Fill the form with value "12 months"* 4. *Click submit button* 5. *Go to create Account* 6. *Open the term box in the create Account and check if the 12 months term has been updated or not*

Actual Output	*New term 12 months appears in the create account term box.*
Result	*Passed*

2.2.2 Test case 2

Test case	Check if the minimum deposit has been updated or not
Related Use case	*Adjust regulation*
Context	*As bank employee*
Input Data	*update minimum deposit to 200000*
Expected Output	*The new minimum deposit will be updated to all forms.*
Test steps	1. *11click to change regulation from home page* 2. *choose change minimum deposit money form* 3. *fill the form with value 200000* 4. *click submit button* 5. *go to create account or deposit money* 6. *fill the money box with a value 150 000.* 7. *the screen should display error due to 150 000 < 200 000 new minimum deposit*
Actual output	*Screen display error due to money not satisfy minimum deposit regulation*
Result	*Passed*

2.2.3 Test case 3

Test case	Check for correct information displayed on screen
Related Use case	*View Account*
Context	*As bank employee*
Input Data	*The data from database*

Expected Output	*The information on screen is similar to input*
Test steps	1. *Query Manualy using DBMS* 2. *Save information from DBMS* 3. *Open viewaccount page* 4. *Check if viewaccount's information is match with saved information*
Actual output	*The information displayed on the viewaccount page matches the information stored in the database.*
Result	*Passed*

2.2.4 Test case 4

Test case	Check if sort function in account transaction work correctly
Related Use case	*View Account*
Context	*As bank employee*
Input Data	*The data from database*
Expected Output	*The information is sort correctly on selected column and order*
Test steps	1. *Go to viewaccount page* 2. *Choose column and order for sorting* 3. *Observe if the selected column is sorted correcly*
Actual output	*The sorted information displayed on the viewaccount page is sorted correcly*
Result	*Passed*

2.2.5 Test case 5

Test case	Check data integrity after sort
Related Use case	*View Account*

Context	*As bank employee*
Input Data	*The data from database*
Expected Output	*The data after sort multiple times*
Test steps	1. *Get constant information of viewaccount from DBMS* 2. *Go to viewaccount page* 3. *Choose the column and order for sorting* 4. *Then check if for every single row have data integrity* 5. *Do it multiple times*
Actual output	*The data is correctly match after every single sort*
Result	*Passed*

2.2.6 Test case 6

Test case	Check if search account work correctly
Related Use case	*View Account*
Context	*As bank employee*
Input Data	*The data from database*
Expected Output	*The data after search is match with data from database*
Test steps	1. *Query manually for specific account_ID on DBMS* 2. *Go to viewaccount page* 3. *Search for same account_ID searched before* 4. *Check if the data after search is match with data from DBMS*
Actual output	*The data match with data from DBMS*
Result	*Passed*

2.2.7 Test case 7

Test case	Check whether the address matches the ID recorded in the database.

Related Use case	*Create Account*
Context	*As bank employee*
Input Data	*The data from input*
Expected Output	*Print error under address : "the input address doesn't match the address from database: [input address in database]"*
Test steps	1. *Input "Customer's Name" which has already existed in database.* 2. *Input "ID Number" which has already existed in database.* 3. *Input different address in "Address"* 4. *Choose "Initial Deposit" and "Term"* 5. *Click "Submit"*
Actual output	*"The input address doesn't match the address from database: [input address in database]"*
Result	*Passed*

2.2.8 Test case 8

Test case	Check whether the customer name matches the ID recorded in the database.
Related Use case	*Create Account*
Context	*As bank employee*
Input Data	*The data from input*
Expected Output	*Print error under address : "the input customer name doesn't match the address from database: [input address in database]"*

Test steps	1. *Input different "Customer's Name" from which has already existed in database.* 2. *Input "ID Number" which has already existed in database.* 3. *Input "Address" which has already existed in database.* 4. *Choose "Initial Deposit" and "Term"* 5. *Click "Submit"*
Actual output	*"The input customer name doesn't match the customer name from database: [input customer name in database]"*
Result	*Passed*

2.2.9 Test case 9

Test case	Check whether deposits are accepted only for the "no periods" savings type.
Related Use case	*Deposit Money*
Context	*As bank employee*
Input Data	*The data from input*
Expected Output	*Print error under "Saving book's ID": "* Only accept deposits for "non-term" savings. The savings type of the book is 3 months"*
Test steps	1. *Input "Saving book's ID" which has already existed in database.* 2. *Input "Deposit Amount"* 3. *Click "Submit"*
Actual output	*"* Only accept deposits for "non-term" savings. The savings type of the book is 3 months"*
Result	*Passed*

2.2.10 Test case 10

Test case	Check whether the account is already closed.
Related Use case	*Deposit Money*
Context	*As bank employee*
Input Data	*The data from input*
Expected Output	*Print error under "Saving book's ID": "* Saving book's Closed"*
Test steps	1. *Input "Saving book's ID" which has already existed in database but has been closed in the past.* 2. *Input "Deposit Amount"* 3. *Click "Submit"*
Actual output	*"* Saving book's Closed"*
Result	*Passed*

2.2.11 Test case 11

Test case	Check whether the account is closed after withdrawing the entire balance.
Related Use case	*Withdraw Money*
Context	*As bank employee*
Input Data	*The data from input*
Expected Output	*Status of Account in "Account Lookup" is changed into "Inactive"*
Test steps	1. *Input "Saving book's ID" which has already existed in database and pass all withdrawal regulations.* 2. *Input "Withdraw Amount" equal to "Old Balance"*

	3. Click "Submit"
Actual output	*Status of Account in "Account Lookup" is changed into "Inactive"*
Result	*Passed*

2.2.12 Test case 12

Test case	Check whether the withdrawal date for non-term savings accounts is more than 15 days after the most recent deposit.
Related Use case	*Withdraw Money*
Context	*As bank employee*
Input Data	*The data from input*
Expected Output	*Print error under "Withdraw date": "* Can only withdraw at least 15 days after the last transaction"*
Test steps	*1. Input "Saving book's ID" which has already existed in database for under 15 days or which has nearest transaction under 15 days.* *2. Input "Withdraw Amount"* *3. Click "Submit"*
Actual output	*Can only withdraw at least 15 days after the last transaction*
Result	*Passed*

2.2.13 Test case 13

Test case	Check whether the account name and savings type are automatically displayed when the account number is entered.
Related Use case	*Withdraw Money*

Context	*As bank employee*
Input Data	*The data from input*
Expected Output	*Displayed name and savings type automatically*
Test steps	*1. Input "Saving book's ID" which has already existed in database.*
Actual output	*Displayed name and savings type automatically*
Result	*Passed*