جامعة الإمام عبد الرحمن بن فيصل
IMAM ABDULRAHMAN BIN FAISAL UNIVERSITY

Imam Abdulrahman Bin Faisal University
College of Computer Science & Information Technology
Department of Computer Science

CS 411 – Software Engineering
Term 1 – 2024/2025

Software Requirements Specification

For

Figure 1: save a beat

Save A Beat (for blood donation)

Version [1]

Team Members

Table of Contents

Revision History 4
1. Introduction 6
1.1 Objectives 6
1.2 Testing Strategy 7
1.3 **Scope** 8
1.4 Reference Material 8
1.5 Definitions and Acronyms 8
2. Test Items 10
2.1 Program Modules 10
2.2 Job Control Procedures 10
2.3 User Procedures 10
2.4 Operator Procedures 11
3. Features To Be Tested 11
4. Features Not to Be Tested 13
5. Approach 14
5.1 component testing 14
5.1.1 common Functions 14
5.1.2 Hospital function 15
5.1.3 Donor Functionalities 18
5.1.3.1 Update profile 18
5.1.3.2 Book and Schedule Donation Appointments 19
5.1.3.3 20
5.1.3.4 View Donation History 21
5.1.4 Blood Bank Functionalities 21
5.2 Integration Testing 24
5.2.1 Hospital Integration Test 24
5.2.2 Blood Bank Integration Test 25
5.2.3 Donors Integration Test 26
5.3 Conversion Testing 27

5.4 Job Stream Testing 27
5.5 Interface Testing 28
5.6 Security Testing 28
5.7 Recovery Testing 29
5.8 Performance testing 29
5.9 Regression Testing 30
5.10 Acceptance Test 30
5.11 Beta Testing 31
6. Pass / Fail Criteria 31
6.1 Suspension Criteria 31
6.2 Resumption Criteria 32
6.3 Approval Criteria 32
7. Testing Process 33
7.1 Test Deliverables 33
7.2 Testing Tasks 33
7.3 Responsibilities 34
7.4 Resources 34
7.5 Schedule 35
8. Environmental Requirements 35
8.1 Hardware 35
8.2 Software 36
8.3 Security 36
8.4 Tools 36
8.5 Publications. 36
8.6 Risks and Assumptions 36
8.6.1 Test Item Availability 37
8.6.2 Test Resource Availability 37
8.6.3 Time Constraints 37
9. Change Management Procedures 37
10. PLan Approval 38

Revision History

Name	Date	Reason For Changes	Version
All members	Sep 30, 2024	Prepared initial version	0.1
All members	Oct. 19, 2024	Updated section 3	0.2
All members	Oct. 25, 2024	Updated Section 4	0.3
All members	Oct.30 ,2024	Updated Section 1, 2 & 6	0.4
All members	Nov9 ,2024	Update Section 3	0.5
All members	Nov. 14, 2013	Complete review - Final version	1.0

Table 1:Revision History

Table of tables

Table 1:Revision History 4
Table 2:Testing Strategy Summary....7
Table 3:Terminologies and Definition 8
Table 4:Acronyms and Definition....9
Table 5:Features to be Tested with Description....12
Table 6:Features Not to be Tested with Reasoning....13
Table 7:Forget Password in Test Case 14
Table 8:Forget Password in Test Case 15
Table 9:Sign up Hospital in Test Case....15
Table 10:Log in Hospital in Test Case....16
Table 11:Hospital Blood Request in Test Case....17
Table 12:Hospital Blood Request Tracking in Test Case 17
Table 13:Hospital Blood Request Update Delete in Test Case....18
Table 14:Update Profile in Test Case 19
Table 15:Book and Schedule Donation in Test Case....20
Table 16:Create Account in Test Case....21
Table 17:View Donation History in Test Case 21
Table 18:Add New Blood Donation in Test Case....22
Table 19:Update Inventory in Test Case....23
Table 20:Manage Request in Test Case 23
Table 21:Manage Campaign in Test Case....24
Table 22:Hospital Integration in Test Case....25
Table 23:Blood Bank Integration in Test Case 26
Table 24:Donors Integration in Test Case....27
Table 25:Testing Resources 35
Table 26:Test Activity Scheduling....35
Table 27:Plan Approvals....38

1. Introduction

This Software Test Plan is a part of the “Save A Beat System” project that captures the testing vision, mission, and goal to accomplish... It offers a guide for testing deliverables and points out the parts of the system, test strategies, resources and time that is required to achieve the set goal of the system.

1.1 Objectives

The main purpose of this document is to describe how to test the “Save A Beat System” to confirm its functional efficiency, effectiveness, and robustness. This document specifies:

The parts and characteristics that are subject to testing.

Some of the specific kinds of testing are: Functional testing, Security testing Usability testing and many others.

As far as various responsibilities of testing process are concerned.

Essential resources; cost estimate; timeline.

Risk mitigation strategies.

This plan ensures that the system meets the requirements specified in the Software Requirements Specification (SRS) and satisfies end-user needs.

1.2 Testing Strategy

No	Section	Description
1	Introduction	Is quite general and describes goals of the testing plan
2	Test Items	Specifies which parts of the system and what modules will be examined.
3	Features to Be Tested	Enumerates and says what feature alone and every possible combination may be tested.
4	Features Not to Be Tested	Describes what is outside the test scope of the functions and procedures.
5	Testing Approach	Explains how the methodologies such as unit testing, integration testing and acceptance testing are used
6	Pass/Fail Criteria	Sets up the criteria from which one can deduce whether a test case has been successful or not.
7	Testing Process	Describes the processes involved in conducting test as well as the examination of the outcomes
8	Environmental Requirements	Prescribes what type of hardware and software, as well as what type of network setup, needs to be in place in order to test.
9	Change Management Procedures	Describes how to deal with changes or increases during the testing process.
10	Plan Approvals	Shows persons to be possibly involved in the review/ approval of the test plan with the signature and date.

Table 2:Testing Strategy Summary

1.3 Scope

The type of testing that has been done on “Save A Beat System” includes; the functional testing, usability testing, security testing and the performance testing. The formative assessments are Component Testing, Integration Testing, Interface Testing, Security Testing, Acceptance Testing. Since the system implements Waterfall SDLC model, testing will respect phase endings and will correspond to the update frequencies, thus avoiding interruptions caused by abrupt changes.

1.4 Reference Material

[1] “IEEE Standard for Software Project”, The Institute of Electrical and Electronics Engineers, 1998

1.5 Definitions and Acronyms

Terminology	Definition
Availability	The degree to which the *Save a Beat* system is accessible to intended users when needed.
Data Flow Diagram (DFD)	This covers a diagram displaying how data flows within the system.
Pseudocode	Commenting to describe the overall shape of a program.
Interface	A connection is a link between items like systems, those using them or devices.
NetBeans	An integrated development environment (IDE) for Java programming, used to create and manage software projects.
Algorithm	A set of final sequential directions or steps which are calculated with accuracy to computerize or to solve certain types of problems.
Encrypted	The act of transformation of any piece of information in a way that makes the piece inaccessible to any other party.

Table 3:Terminologies and Definition

Acronyms	Meaning
SRS	Software Requirement Specification
IEEE	Institute of Electrical and Electronics Engineers
ID	Identification Number
PC	Personal Computer
SPMP	Software Project Management Plan
Ios	iPhone Operating System
SQL	Structured Query Language
UI	User Interface
DFD	Data Flow Diagram
SDLC	Software Development Life Cycle
AST	Application Security Testing Tools
OS	Operating System
JVM	Java Virtual Machine
GUI	Graphical User Interface
E-mail	Electronic Mail

Table 4:Acronyms and Definition

2. Test Items

All significant facets of testing involve donor dashboards, blood bank inventory, items requested by hospitals, notification displays, and maps.

2.1 Program Modules

User Interfaces: The focus is usability and navigation; thus, it will incorporate first user scenarios or cases to prove user friendliness as well as interface similarity. Others such as TestRail will be used for tracking results.

Data Management: Both new and existing operations, including donor sign up, blood stock data update, and emergency request processing, will be simulated through MySQL statements.

2.2 Job Control Procedures

Testing involves:

- Unit testing for isolated modules like donor registration.
- Integration testing for inter-module communication, such as donor and blood bank interactions.
- System testing for full end-to-end validation.

2.3 User Procedures

All user workflows (e.g., booking appointments, responding to donation campaigns) will undergo functional validation against the system's SRS.

2.4 Operator Procedures

Key procedures include:

- Disaster Recovery Testing: Ensures recovery from system failures and data loss.
- Performance Testing: Evaluates speed and responsiveness under high usage.
- Help Desk Testing: Assesses ticketing efficiency and resolution processes.

3. Features To Be Tested

This section involves the main features to be tested to guarantee the functionality of each interface. Furthermore, it explains the test design specifications to each interface.

Features	Design Specification
Common Interface	
Contact Support	A test to ensure users can send messages to the support team, confirm message entry (no empty) and receive appropriate feedback.
Forgot password	A test to verify that users can request a password reset, receive a reset link to email and register from that link.
Donors Interface	
Log In / Create Account	A test to authenticate donors can create accounts securely and log in with correct information.
Donation Booking	A test to ensure donation appointments are booked correctly based on date, location, and type.
Profile and Health Information	A test to ensure donors can update their profile details and check donation eligibility.
Hospital Interface	
Log In / Sign up	A test to authenticate hospital to create a secure account while ensuring compliance with legal requirements.

Create Blood Request	A test to validate that hospitals can create valid blood requests specifying blood type, quantity, and urgency level.
Track Blood Requests	A test to verify that the request tracking interface accurately displays the status of on process requests
Update/Delete Blood Request	A test to validate that hospitals can Update and delete blood requests.
Blood Bank Interface	
Log in /Create account	Ensure Blood Bank staff can securely create accounts and log in to the system.
Update Inventory	Ensure updates to the inventory (e.g., marking blood units as used, dispatched, or expired) are accurate.
Add New Donation	A test to confirm that new blood donations are logged with correct details, including blood type, Quantity, Date of Donation and donor ID.
Manage Requests	A test on the handling of incoming blood requests from hospitals.
Manage Campaigns	A test to validate the ability to add, search and view blood donation campaigns.
Add Campaign	A test to confirm that the staff can come up with donation campaigns and invite donors to the campaign.

Table 5:Features to be Tested with Description

4. Features Not to Be Tested

This section involves features that don’t require testing because of its low priority among the interfaces or isn’t needed for user interactions. This ensures the focus remains on the interface with the important functionalities.

Features Not to Be Tested	Reasoning
Campaign Summary	Given that the user is only viewing campaign data, no testing is required.
Donor Home Dashboard	The dashboard simply displays generated data (e.g., upcoming events, donation history), so testing is not necessary.
Donation History	As it only displays already recorded donation data, it does not require testing.
Support and Resources	As it provides static information (FAQs, guidelines, etc.), so no testing is required.
Upcoming Campaigns and Events	As it simply lists generated campaign data, which does not require testing.
Request History Interface	As it only shows recorded data and does not require testing.
My Profile Interface	Since this interface only displays and allows basic editing of profile information, it does not require testing.

Table 6:Features Not to be Tested with Reasoning

5. Approach

5.1 component testing

5.1.1 common Functions

TEST ID	Forgot_Password_01
PREREQUISITE	The user has an account, and all their information must be stored in the database.
TEST PROCEDURE	These test procedures apply to donors, hospitals, and blood banks. 1. Press "Reset Password" without entering an email 2. Enter an invalid email format 3. Enter a valid email address
EXPECTED RESULT	1. An error message should appear" please fill in all the required field" 2. An error message should appear" please enter valid email" 3. A reset password email should be sent successfully.
ACTUAL RESULT	Same as expected results.
VERIFIED (YES/NO)	YES

Table 7:Forget Password in Test Case

5.1.1.2 Contact Support

TEST ID	Contact_Support_01
PREREQUISITE	All the users have the privilege to send message
TEST PROCEDURE	These test procedures apply to donors, hospitals, and blood banks. 1. Press sends without entering a message 2. Enter a message in text message
EXPECTED RESULT	1. An error message should appear" please enter a message then send"

	2. The message will be sent successfully.
ACTUAL RESULT	Same as expected results.
VERIFIED (YES/NO)	YES

Table 8:Forget Password in Test Case

5.1.2 Hospital Functionalities

5.1.2.1 Sign up interface

TEST ID	Sign_Up_Hospital
PREREQUISITE	No PREREQUISITE
TEST PROCEDURE	These test procedures apply to hospitals, The user clicks on “Sign up” button by using each of the following options: 1. The user enters wrong format in hospital name 2. The user enters invalid email address 3. The user leaves empty fields 4. Passwords dose not match 5. Invalid password not following the criteria 6. The user enters all the fields correctly
EXPECTED RESULT	The results of the previous procedures are: 1. 1. error message appear” incorrect Data type in hospital name” 2. 2. error message appear "please enter valid email address” 3. 3. error message appear "please fill in all required fields” 4. 4. error message appear "passwords dose not match” 5. 5. error message appear "password should contain at least 8 characters and at least one upper case letter” 6. Account is successfully created
ACTUAL RESULT	Same as expected results.
VERIFIED (YES/NO)	YES

Table 9:Sign up Hospital in Test Case

5.1.2.2 Hospital log in

TEST ID	Hospital_Log_In_01
PREREQUISITE	The user has an account, and all their information must be stored in the database.
TEST PROCEDURE	These test procedures apply to hospitals, The user clicks on “login” button by using each of the following options: 1. The user enters invalid name/password 2. The user leaves empty fields 3. The user enters all information correctly
EXPECTED RESULT	The results of the previous procedures are: 1. Error messages appear " Please enter valid name/password " 2. Error messages appear "Please fill in all the required fields " 3. The user successfully logs in to their account
ACTUAL RESULT	Same as expected results.
VERIFIED (YES/NO)	YES

Table 10:Log in Hospital in Test Case

5.1.2.3 Hospital Create Blood request

TEST ID	Hospital_Blood_Request_01
PREREQUISITE	The hospital logs in to their account
TEST PROCEDURE	These test procedures apply to hospitals, The user clicks on “Submit Request” button by using each of the following options: 1. The user enters an invalid quantity, such as alphabetic characters instead of numeric values. 2. The user leaves empty fields 3. The user enters all information correctly

EXPECTED RESULT	The results of the previous procedures are: 1. Error messages appear " Mismatch error in quantity " 2. Error messages appear "Please fill in all the required fields " 3. Request successfully created
ACTUAL RESULT	Same as expected results.
VERIFIED (YES/NO)	YES

Table 11:Hospital Blood Request in Test Case

5.1.2.4 Hospital Request Tracking

TEST ID	Hospital_Blood_Request_Tracking_01
PREREQUISITE	The hospital logs in to their account,
TEST PROCEDURE	These test procedures apply to hospitals 1. The user enters an invalid Request ID (not in the database) 2. The user enters an invalid Request ID (alphabetic character) 3. The user enters valid Request ID
EXPECTED RESULT	The results of the previous procedures are: 1. Error messages appear " Please enter valid Request ID" 2. Error messages appear "Mismatch data type in Request ID" 3. Timeline Successfully shows the request details.
ACTUAL RESULT	Same as expected results.
VERIFIED (YES/NO)	YES

Table 12:Hospital Blood Request Tracking in Test Case

5.1.2.5 Hospital Update/Delete Request / Same as create Request

TEST ID	Hospital_Blood_Request_Update_Delete_01
PREREQUISITE	The hospital logs in to their account, hospital have previous requests
TEST PROCEDURE	These test procedures apply to hospitals 1. if the user types the invalid Request id (does not exist in the system or text instead of numeric). 2. If the user types valid ID
EXPECTED RESULT	The results of the previous procedures are: 1. error message appears “Please enter valid Request ID. 2. User can update the request information
ACTUAL RESULT	Same as expected results.
VERIFIED (YES/NO)	YES

Table 13:Hospital Blood Request Update Delete in Test Case

5.1.3 Donor Functionalities

5.1.3.1 Update profile

TEST ID	Update Profile_01
PREREQUISITE	Donor is logged into the system. Existing profile details are available for update.
TEST PROCEDURE	1- Donor navigates to the "Update Profile" section. 2- Performs the following scenarios: (a) Updates all profile fields with valid data. (b) Leaves required fields (e.g., phone number) blank.

	(c) Attempts to update with a duplicate phone number (already in the system).
EXPECTED RESULT	(a) Successfully updates profile details and saves changes. (b) Displays error message: "This field is required." (c) Displays error message: "Phone number already exists."
ACTUAL RESULT	Same as expected results
VERIFIED (YES/NO)	yes

Table 14:Update Profile in Test Case

5.1.3.2 Book and Schedule Donation Appointments

TEST ID	Book Appointment_01
PREREQUISITE	Donor is logged in. Available appointment slots exist in the system
TEST PROCEDURE	1- Donor navigates to the "Book Appointment" section. 2- Performs the following scenarios: (a) Books an appointment with valid details. (b) Attempts to book an appointment for an already booked time slot. (c) Leaves required fields blank during booking

EXPECTED RESULT	(a) Successfully books the appointment and updates the schedule. (b) Displays error message: "This day is full please choose another day." (c) Displays error message: "This field is required.
ACTUAL RESULT	Same as expected result
VERIFIED (YES/NO)	YES

Table 15: Book and Schedule Donation in Test Case

5.1.3.3 Create Account

TEST ID	Create Account_01
PREREQUISITE	The system is accessible to new users. Valid user details (e.g., name, email, phone number, EID, and blood type) are available.
TEST PROCEDURE	1- User navigates to the "Sign Up" or "Create Account" page. 2- Performs the following scenarios: (a) Fills in all required fields with valid information and submits. (b) Leaves mandatory fields blank (e.g., email or password). (c) Enters a phone number exceeding 10 digits. (d) Enters an ID exceeding 10 digits. (e) Enters an unrecognized blood type
EXPECTED RESULT	(a) Successfully creates the account and redirects to the dashboard. (b) Displays error message: "This field is required." (c) Displays error message: "Phone number cannot exceed 10 digits." (d) Displays error message: "ID cannot exceed 10 digits." (e) Displays error message: "Invalid blood type. Please select a valid blood type.

ACTUAL RESULT	Same as expected results.
VERIFIED (YES/NO)	YES

Table 156:Create Account in Test Case

5.1.3.4 View Donation History

TEST ID	View Donation History_01
PREREQUISITE	Donors have previous donation records in the system. Donor is logged in
TEST PROCEDURE	1- Donor navigates to the "Donation History" section. 2- Views donation records for the following scenarios: (a) Donation history exists. (b) No donation records are found
EXPECTED RESULT	(a) Successfully displays donation history, including date, type, and quantity. (b) Displays message: "No donation history available.
ACTUAL RESULT	Same as expected results.
VERIFIED (YES/NO)	YES

Table 167:View Donation History in Test Case

5.1.4 Blood Bank Functionalities

5.1.4.1 Add New Blood Donation

TEST ID	Add New Blood Donation_01
PREREQUISITE	• Blood bank admin is logged in. • Donor details and blood type are known and valid.
TEST PROCEDURE	1- Admin navigates to the "Add New Donation" option. 2- 2-Inputs the following scenarios: (a) All valid donor details. (b) Missing donor ID. (c) Invalid blood type. (d) Negative or zero blood quantity.

	(e) Future donation date.
EXPECTED RESULT	(a) Successfully adds a donation record and updates inventory. (b) Displays error message: *"Donor ID is required."* (c) Displays error message: *"Invalid blood type."* (d) Displays error message: *"Quantity must be greater than zero."* (e) Displays error message: *"Donation date cannot be in the future."*
ACTUAL RESULT	Same as expected results.
VERIFIED (YES/NO)	YES

Table 178:Add New Blood Donation in Test Case

5.1.4.2 Update Inventory Table

TEST ID	Update Inventory Table_01
PREREQUISITE	Blood inventory records exist in the system.
TEST PROCEDURE	1- Admin selects the "Update Inventory" option. 2- Performs the following scenarios: (a) Marks a unit as dispatched. (b) Marks a unit as expired. (c) Updates the quantity of an existing blood type. (d) Attempts to update with invalid data (e.g., negative quantity).
EXPECTED RESULT	(a) Successfully marks the unit as dispatched and updates inventory. (b) Successfully marks the unit as expired and reduces inventory. (c) Successfully updates the quantity.

	(d) Displays error message: *"Invalid quantity. Please enter a valid number."*
ACTUAL RESULT	Same as expected results.
VERIFIED (YES/NO)	YES

Table 189:Update Inventory in Test Case

5.1.4.3 Manage requests

TEST ID	Manage Requests Table_01
PREREQUISITE	Requests exist in the system.
TEST PROCEDURE	Admin selects the "Manage Requests" option. 1 - Performs the following scenarios: (a)Approves an emergency request. (b)Rejects an emergency request.
EXPECTED RESULT	(a) Successfully approves the emergency request and updates inventory. (b) Marks the request as rejected in the system.
ACTUAL RESULT	Same as expected results.
VERIFIED (YES/NO)	YES

Table 190:Manage Request in Test Case

5.1.4.4 Manage campaign

TEST ID	Manage Campaign Table_01

PREREQUISITE	System is logged in with admin access.
TEST PROCEDURE	Admin selects the "Manage Campaigns" option. Performs the following scenarios: 1- Creates a new campaign with valid details (e.g., date, target). 2- Attempts to create a campaign with invalid details (e.g., invalid date). 3- Search for an existing campaign by year
EXPECTED RESULT	1- Successfully creates a new campaign and stores it in the database. 2- Displays error message: "Invalid date. Please enter a valid campaign date. 3- Successfully retrieves the campaign details based on the search criteria.
ACTUAL RESULT	Same as expected results.
VERIFIED (YES/NO)	YES

Table 201:Manage Campaign in Test Case

5.2 Integration Testing

Integration testing verifies that all components and functions are properly connected, and work as expected.

Technique:

- Incremental testing is conducted to identify bugs and errors efficiently.

Completion Criteria:

- The system operates cohesively and performs as intended.
-

5.2.1 Hospital Integration Test

TEST ID	Hospital_Home_Page_01
PREREQUISITE	The hospital logs in to their account

TEST PROCEDURE	The following procedures are required to be tested one by one, integrating with the respective components: • Access the hospitals "Quick Request "option and perform its respective component test case again. • Access the hospitals "History Request "option and perform its respective component test case again. • Access the hospitals "Track Request "option and perform its respective component test case again. • Access the hospitals "Edit Request "option and perform its respective component test case again. • Access the hospitals "My profile "option and perform its respective component test case again.
EXPECTED RESULT	The results of the above separate procedures are the same as the results of each component in the previous subsection **5.1.2 component testing**
ACTUAL RESULT	Same as expected results.
VERIFIED (YES/NO)	YES

Table 212:Hospital Integration in Test Case

5.2.2 Blood Bank Integration Test

TEST ID	Blood Bank Homepage_01
PREREQUISITE	The user is an admin and has already signed into their account.
TEST PROCEDURE	The following procedures are required to be tested one by one, integrating with the respective components: a) Access the "Add New Donation" option and perform its respective component test case again. b) Access the "Update Inventory Table" option and perform its respective component test case again. c) Access the "Manage Requests" option and perform its respective component test case again.

	d) Access the “Manage Campaigns” option and perform its respective component test case again.
EXPECTED RESULT	The results of the above individual procedures align with the results for each component detailed in **5.1.4 Component Testing**.
ACTUAL RESULT	Same as expected results.
VERIFIED (YES/NO)	YES

Table 223:Blood Bank Integration in Test Case

5.2.3 Donors Integration Test

TEST ID	Donor_Homepage_01
PREREQUISITE	The donor is logged into their account.
TEST PROCEDURE	a) Access the "Update Profile" option and perform its respective component test case again. b) Access the "Create Account" option and perform its respective component test case again. c) Access the "Book and Schedule Donation Appointments" option and perform its respective component test case again. d) Access the "View Donation History" option and perform its respective component test case again.

EXPECTED RESULT	The results of the above individual procedures align with the results for each component detailed in 5.1.3 Component Testing.
ACTUAL RESULT	Same as expected results
VERIFIED (YES/NO)	YES

Table 234:Donors Integration in Test Case

5.3 Conversion Testing

Conversion testing ensures that data is transferred seamlessly to a new system, acting like a data transformation process for the project. It safeguards the integrity of the data, ensuring that no information is lost or corrupted during the transition. The goal is to verify that the converted data is fully functional and integrates correctly into the new system, ensuring a flawless evolution of the data.

Technique:

- Conducting various tests to guarantee a smooth transfer of data from the old system to the new one.

Completion Criteria:

- Conversion testing is considered complete when the transferred data matches the expected output and functions accurately within the new system.

5.4 Job Stream Testing

Job Stream Testing ensures that the application operates correctly in the production environment by verifying that all tasks are executed in the prescribed order. The output of one task serves as the input for the next.

Technique:

- A job stream test will be conducted to ensure the correct flow of information between consecutive tasks.

Completion Criteria:

- The flow of information between tasks must be accurate, ensuring correct input, processing, and output at each stage.

5.5 Interface Testing

Interface testing focuses on identifying faults and ensuring that all interactions within the Save A Beat app work seamlessly, with data flowing correctly between components and all errors properly handled. This testing ensures that the user interface interactions, such as navigating between hospital home pages, creating blood requests, and managing request function as expected without errors.

Technique:

- User interfaces will be tested by simulating interactions, verifying that data flows correctly between components and the correct information is displayed. Each interface will be checked to ensure that user actions, such as navigating through different sections of the app, trigger the expected outcomes.

Completion Criteria:

- Interface testing will be considered successful when all user interactions and data flows operate correctly, ensuring the system functions as intended without errors or failures.

Special Consideration:

- After any feedback is received, the development team will address necessary adjustments and revalidate the interfaces to ensure they are functioning properly.

5.6 Security Testing

Security testing is essential for ensuring that sensitive and private user information, such as IDs, is safeguarded from unauthorized access. Since security is a critical non-functional requirement of the software, this testing must be conducted regularly to maintain a high level of protection and verify that all user data is secure. To ensure that the application provides robust security measures, preventing unauthorized access and protecting user information from breaches.

Techniques:

- Restrict access to users with valid credentials, such as a national ID, password.
- Deny access to the system if incorrect credentials are entered.
- Ensure all actions within the application are performed securely.

Completion Criteria:

- Security testing is complete when the system verifies that only authorized users can access it and ensures that user data remains safe and protected from unauthorized actions.

5.7 Recovery Testing

Recovery testing ensures that the system can effectively recover from failures or unanticipated events, serving as a safety net for the project. It focuses on verifying the system's ability to restore operations and maintain data integrity after setbacks, such as server failures or unexpected shutdowns. This testing ensures that backup and recovery procedures are robust and functional, safeguarding the system's reliability. To verify that the system can recover seamlessly after a failure, ensuring that operations continue without data loss or disruption. The system must have a backup plan in place to restore data and operations. For example, in the event of a server failure, the system should automatically migrate activities to a failover server, send alerts to notify the team, and allow for swift resolution.

Techniques:

- Simulate failures by disconnecting the internet or shutting down the server while a function is running and observe the system's recovery process.
- Ensure that reprocessing transactions from the last known stable point is possible.

Completion Criteria:

- Recovery testing is successful when the system restarts without issues, all data is restored from the backup, and normal operations resume.

5.8 Performance testing

Performance testing ensures that it remains reliable, fast, and accessible across different platforms and under various load conditions. This testing phase will guarantee user satisfaction and instill confidence in the app's ability to deliver seamless and accurate performance during critical operations.

Technique:

- Evaluate the app's response time under normal and high usage conditions.
- delivering the same speed and functionality on all devices.
- Test the app's accessibility at different times.
- Evaluate the app's ability to handle increasing user numbers or data volume.

Completion Criteria:

- The app operates successfully under all tested conditions without crashing.
- Response times consistently meet acceptable criteria, regardless of load or platform.
- Throughput remains stable and does not degrade with increasing user activity.
- The app demonstrates successful fault recovery, ensuring continued availability.

5.9 Regression Testing

Regression testing ensures that changes or updates to the system, such as modifications to code or the addition of new features, do not negatively impact existing functionality. This process validates that the software's functional and non-functional attributes remain intact after changes are implemented. If performance issues or errors arise due to these changes, they are referred to as regressions.

Technique:

- Regression tests are conducted after integrating new components or making modifications to the system. These tests help identify any bugs early in the integration process, ensuring stability and compatibility within the system's environment.

Completion Criteria:

- The testing is successful when all components function seamlessly as a cohesive system, with no errors or negative impact on existing functionality.

5.10 Acceptance Test

Acceptance testing is aimed at ensuring the software meets the client's needs and is ready for use. During this phase, the client assesses the system's effectiveness and provides feedback on its performance. Necessary adjustments are made based on this feedback before the software is officially released.

Technique:

- The client directly interacts with the software, using its features and functionalities in real-world scenarios.
- The client evaluates the system's usability, functionality, and compliance with specified requirements.
- Feedback is provided on functional and non-functional attributes, including any usability concerns, missing features, or areas for improvement.

Completion Criteria:

- The system fully satisfies all functional and non-functional requirements specified by the client.
- The software is error-free, stable, and ready for deployment in the user's environment.
- All client-recommended changes have been implemented, tested, and validated.

5.11 Beta Testing

Beta testing is a type of user testing where a pre-release version of the software is provided to a select group of end-users to validate its functionality, reliability, and compatibility. This process helps ensure the software is error-free, meets business and functional requirements, and aligns with customer expectations. To identify faults, failures, and defects in the software, validate that it meets the customer's needs, and confirm that it adheres to business standards and acceptance criteria.

Technique:

- Users test the software, report issues, and share their findings with the development team. The development team then analyzes the feedback, resolves bugs, and optimizes performance.

Completion Criteria:

- Beta testing is successful when all system features and functionalities have been thoroughly tested, proven to work as expected, and meet the customer's requirements and business standards.

6. Pass / Fail Criteria

6.1 Suspension Criteria

Suspension criteria describe the events or issues that may cause testing to be temporarily paused. These include:

- **Critical Functional Defects**: Severe issues that impact core functionalities, such as the failure of blood inventory management, which renders the system unusable.
- **Unavailability of Hardware or Software**: If essential hardware or software is unavailable or malfunctions, preventing the continuation of testing.
- **Build Failures**: If a new build introduces defects that affect testing operations or prevent the test from proceeding.
- **Lack of Test Resources**: If necessary, resources, such as testing personnel, environments, or test data, are unavailable.
- **Timeline Delays**: When project timelines change in a way that affects the testing schedule.
- **Connection Issues**: Network or connectivity issues that disrupt the testing process, especially for cloud-based components.

- **Customer-side Enhancements**: If updates or changes from the customer side are required before testing can proceed.
- **SRS Deviations**: When the system's functionality deviates from the defined Software Requirements Specification (SRS), causing a mismatch between expected and actual results.

6.2 Resumption Criteria

Resumption criteria outline the conditions that must be met for testing to resume after being paused. These include:

- **Resolution of Critical Issues**: All serious problems identified during testing have been fixed and validated, such as issues with blood request handling or donor eligibility checks.
- **Stable Application Builds**: The new application build is stable, ensuring the reliability and accuracy of future tests.
- **Availability of Resources**: Essential hardware, software, and testing resources (e.g., test data, environments, personnel) are now available and ready for use.
- **Timeline Adjustments**: The project schedule has been reviewed, and necessary adjustments or accommodations have been made to ensure testing can continue without affecting overall project deadlines.
- **Restored Connectivity**: Any connection issues have been resolved, ensuring stable communication between testing components and external systems.
- **Customer-side Enhancements Implemented**: All required customer-side updates have been implemented, and any necessary adjustments to the testing strategy or process have been made.
- **Compliance with SRS**: The system is now functioning as defined in the Software Requirements Specification (SRS), and any previously identified deviations have been addressed.

6.3 Approval Criteria

The approval criteria establish the conditions that must be met to deem the test results valid and acceptable. These include:

- **Error Resolution**: Any errors or failures discovered during testing must be fully addressed and fixed to ensure accurate results.

- **Test Result Validation**: Test results will be considered valid only after comparing them to the predefined criteria and confirming that all functionalities meet the specifications.
- **Risk Mitigation**: The goal is to minimize risks and prevent any unwanted issues from impacting the hardware or software.
- **Formal Review**: A formal review of the test results will be conducted to ensure all issues are resolved, and the testing process is completed accurately and thoroughly. This review will confirm that the application meets its requirements and is ready for deployment.

7. Testing Process

This section outlines the various components involved in the testing process, including test deliverables, tasks, responsibilities, resources, and the schedule.

7.1 Test Deliverables

The key deliverables of the testing process for the **'Save A Beat'** blood bank system are:

- **Software Test Plan (STP)**: The document that outlines the testing strategy, objectives, and approach for the system.
- **Test Cases**: Detailed scenarios for testing the functionalities, such as blood inventory management, donor eligibility checks, and hospital request processing.
- **Test Reports**: Documentation that records the results of each test, including pass/fail statuses and any discovered defects.
- **Defect Logs**: Records of any issues encountered during testing, including the severity, steps to reproduce, and the resolution status.
- **Test Summary Report**: A final report summarizing all testing activities, results, and recommendations.

7.2 Testing Tasks

The primary tasks involved in the testing process include:

1. **Review of Software Requirements and Design Documentation**: Ensuring that all functionalities defined in the **Software Requirements Specification (SRS)** and **Software Design Specification (SDS)** documents are understood and ready for testing.
2. **Test Case Development**: Creating detailed test cases based on the SRS and SDS to ensure all aspects of the system are covered.
3. **Setting Up Testing Environments**: Configuring both software and hardware environments to ensure they align with the requirements for testing (e.g., database setup, network configurations).
4. **Execution of Test Cases**: Running the test cases using various testing methods, including functional, performance, and security testing.
5. **Defect Logging and Tracking**: Documenting and tracking any defects encountered during testing.
6. **Troubleshooting and Resolution**: Investigating and resolving any issues or errors discovered during testing.
7. **System Modification**: Implementing necessary system changes based on testing results and retesting to verify fixes.
8. **Final Review**: Conducting a final review of the testing results and preparing the Test Summary Report.

7.3 Responsibilities

All team members share responsibility for various aspects of the testing process, which includes:

- **Test Planning**: Creating and maintaining the Software Test Plan and associated documents.
- **Test Design and Execution**: Developing and executing the test cases based on requirements.
- **Error Management**: Identifying, logging, and resolving errors that arise during testing.
- **Test Reporting**: Documenting the results of the testing process, including the identification of defects and their resolutions.
- **System Updates**: Modifying the system based on testing feedback and retesting after changes are made.

7.4 Resources

Resource Category	Description
Human	All team members, depending on their area of expertise.

Hardware	High-speed Wi-Fi connection and a computer with sufficient resources to support testing activities.
Software	a) Windows and iOS operating systems for cross-platform compatibility. b) MySQL Workbench for database connection and query testing. c) NetBeans IDE for development and testing integration.

Table 245:Testing Resources

7.5 Schedule

The testing process will follow this timeline, with activities planned and completed at various stages of the development lifecycle:

Task	Date
Review Software Requirements Specification (SRS)	Nov. 9, 2023
Review Software Design Specification (SDS)	Nov. 18, 2023
Develop Test Cases	Nov. 20, 2023
Execute Testing (Functional, Performance, Security)	Nov. 21, 2023
Troubleshoot Errors	Nov. 23, 2023
Modify the System (as needed based on test results)	Dec. 1, 2023
Develop the Software Testing Plan (STP)	Dec. 1, 2023

Table 256:Test Activity Scheduling

8. Environmental Requirements

8.1 Hardware

The system is designed to operate on a range of modern devices, including:

- **Mobile Devices:** Android devices running version 14 or later and iOS devices running version 16 or later.
- **Computers:** Any system with at least 4 GB of RAM and a dual-core processor.

8.2 Software

- **Operating Systems:**
  - Windows 11 or macOS for development and testing environments.
  - iOS and Android for mobile application deployment.
- **Development Tools:**
  - NetBeans.
  - MySQL Workbench for database management.

8.3 Security

To ensure data confidentiality and system security during testing:

- Application security testing tools (AST) will be employed to detect and mitigate vulnerabilities.
- Sensitive user credentials (e.g., IDs, passwords, medical history) will be encrypted and stored securely.

8.4 Tools

- **Development:** NetBeans and MySQL Workbench.
- **Testing:** Platforms for unit and integration testing, such as JUnit.

8.5 Publications.

The documentation required for conducting testing includes:

- **Software Requirements Specification (SRS)**
- **Software Design Specification (SDS)**
- **Software Project Management Plan (SPMP)**

8.6 Risks and Assumptions

To ensure high-quality, error-free software, follow the Software Development Life Cycle (SDLC) procedures. Rapid development and modification might lead to changes in test plans,

posing dangers. The following list outlines potential hazards, assumptions, and contingency plans for each one.

8.6.1 Test Item Availability

All testable items, including modules and database connections, must be fully operational before testing begins.

8.6.2 Test Resource Availability

Developers, testers, and users must be available to provide feedback during critical phases.

8.6.3 Time Constraints

Testing must align with the project schedule to ensure timely delivery.

9. Change Management Procedures

All changes to the system are documented, reviewed, and implemented carefully to avoid disruptions. Change requests include a description, reason, and impact. The project manager and stakeholders review and approve changes based on feasibility and alignment with project goals. Approved changes are first tested in a development environment to ensure they work correctly without causing new problems. All changes are recorded in a change log for tracking, and regular reviews ensure they meet project objectives. This process helps maintain focus and minimize risks.

10. Plan Approvals: