Software Test Plan

for

<Emergency Response and Coordination App>

Version 1.0 approved

Prepared by <sami nabi>

Table of Contents

[Revision History 3](#_Toc136846938)

[1. TEST PLAN IDENTIFIER: AT-TP01.3 4](#_Toc136846939)

[2. REFERENCE MATERIALS 4](#_Toc136846940)

[3. INTRODUCTION 4](#_Toc136846941)

[3.1 Background to the Problem 4](#_Toc136846942)

[3.2 Solution to the Problem 4](#_Toc136846943)

[4. REQUEIREMNT SPECIFICATION 4](#_Toc136846944)

[4.1 System Features 4](#_Toc136846945)

[4.2 System Quality Attributes 5](#_Toc136846946)

[4.3 System Interface 5](#_Toc136846947)

[4.4 Project Requirements 5](#_Toc136846948)

[5. FEATURES NOT TO BE TESTED 5](#_Toc136846949)

[6. TESTING APPROACH 5](#_Toc136846950)

[6.1 Testing Levels 5](#_Toc136846951)

[6.2 Test Tools 5](#_Toc136846952)

[6.3 Meetings 5](#_Toc136846953)

[7. TEST CASES/TEST ITEMS 6](#_Toc136846954)

[8. ITEM PASS/FAIL CRITERIA 6](#_Toc136846955)

[9. TEST DELIVERABLES 6](#_Toc136846956)

[10. STAFFING AND TRAINING NEEDS 6](#_Toc136846957)

[11. RESPONSIBILITIES 6](#_Toc136846958)

[12. TESTING SCHEDULE 7](#_Toc136846959)

[13. PLANNING RISKS AND CONTINGENCIES 7](#_Toc136846960)

[14. APROVALS 7](#_Toc136846961)

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision** | **Date** | **Updated by** | **Update Comments** |
| 0.1 | 18-Dec-23 | Sami Nabi | System Interface |
| 0.2 | 19-Dec-23 | Sami Nabi | Project Requirement |
| 0.3 | 20-Dec-23 | Sami Nabi | Test Tools |
| 0.4 | 21-Dec-23 | Sami Nabi | Test Case |
| 0.5 | 21-Dec-23 | Sami Nabi | Cross Reference |
| 0.6 | 22-Dec-23 | Sami Nabi | Test Deliverables |
| 0.7 | 23-Dec-23 | Sami Nabi | Item Pass / Fail Criteria |
| 0.8 | 23-Dec-23 | Sami Nabi | Cross Reference |
| 0.9 | 24-Dec-23 | Sami Nabi | Responsibilities |

# TEST PLAN IDENTIFIER: AT-TP01.3

# REFERENCE MATERIALS

* Any reference documents with the test plan. For example: Software Requirement Specification (SRS) Document
* Software Quality and Testing Course PowerPoint Slides.

# INTRODUCTION

## 3.1 Background to the Problem

* In today's fast-paced world, emergencies and disasters can strike at any time, ranging from natural disasters like earthquakes and floods to man-made crises like accidents or terrorist incidents. The efficient coordination of emergency response efforts is critical to minimizing damage, saving lives, and ensuring swift recovery. However, traditional methods of communication and coordination often prove inadequate in the face of such challenges. Delays, miscommunication, and lack of real-time information sharing can lead to inefficient responses and increased casualties.
* The root cause of this problem lies in the lack of a unified and technologically advanced platform that facilitates seamless communication, collaboration, and coordination among various stakeholders involved in emergency response. This problem is of paramount importance as lives and property are at stake during emergencies, making the need for an effective solution crucial.

## 3.2 Solution to the Problem

Our proposed solution is an "Emergency Response and Coordination App" that addresses the aforementioned challenges by providing a robust and user-friendly platform for coordinating emergency response efforts. This app will allow different emergency response teams, such as first responders, medical personnel, law enforcement, and relevant authorities, to communicate, share real-time data, and collaborate efficiently during emergencies.

The app will provide features like real-time location sharing, incident reporting, resource allocation, task assignment, and communication tools (e.g., chat and notifications) to streamline coordination efforts. This solution is appropriate because it leverages the ubiquity of smartphones and the power of real-time data sharing to bridge communication gaps and enhance the effectiveness of emergency response.

# REQUEIREMNT SPECIFICATION

## System Features

**1. System Sign Up**

Functional Requirements:

1.1. Users can access the Sign Up page from the home screen.

1.2. Users must provide a unique username, valid email address, and a strong password.

1.3. The system validates the email address format and ensures that the username is not already taken.

1.4. Upon successful submission, a confirmation email is sent to the user for verification.

1.5. Users can verify their email by clicking on the verification link in the email.

1.6. After email verification, users are registered and can proceed to log in.

Priority: High

Pre-conditions: None

**2. System Login**

Functional Requirements:

2.1. Users can access the Login page from the home screen.

2.2. Users must provide their registered email and password.

2.3. The system validates the entered credentials against the database.

2.4. Upon successful login, users are directed to the User Interface.

Priority: High

Pre-conditions: User must be registered and have a verified email address.

**3. Forgot Password**

Functional Requirements:

3.1. Users can access the "Forgot Password" link on the Login page.

3.2. Users must enter their registered email address for password recovery.

3.3. The system sends a password reset link to the user's email.

3.4. Users can click the link to reset their password and set a new one.

Priority: Medium

Pre-conditions: User must have a registered and verified email address.

**4. View User Interface**

Functional Requirements:

4.1. After successful login, users are directed to the User Interface.

4.2. The User Interface displays a dashboard with relevant information and options.

4.3. Users can view their borrowed books, fines, and account details.

4.4. Users can access the Book Category section and other library-related features.

Priority: High

Pre-conditions: User must be logged in.

5. Share Location

Functional Requirements:

5.1. Location Sharing Capability: Users can opt to share their current location within the application.

5.2. Location Privacy Settings: Users can set preferences to control the duration and audience for location sharing.

5.3. Map Integration: Integration with mapping services to display shared locations accurately.

5.4. Emergency Location Sharing: In case of an emergency, users can trigger an immediate location share with designated contacts or emergency services.

Priority: Medium to High

Pre-conditions: User must grant location access permission.

6. Live Chat

Functional Requirements:

6.1. Real-Time Messaging: Users can engage in live chat with library staff or other users.

6.2. Multiple Chat Support: Ability to manage multiple chat sessions concurrently.

6.3. Chat History: Saving and accessing chat history for future reference.

6.4. Notification Alerts: Users receive notifications for new messages or responses.

Priority: High

Pre-conditions: Users must have an active internet connection.

7. Call Emergency Response Team

Functional Requirements:

7.1. Emergency Button: Accessible within the application for immediate contact with emergency response services.

7.2. Geolocation Integration: Automatically sends user location information alongside the emergency call.

7.3. Confirmation Prompt: Prompt for confirmation before initiating an emergency call to prevent accidental activations.

7.4. Emergency Protocol Information: Access to emergency protocols or guidance for users during critical situations.

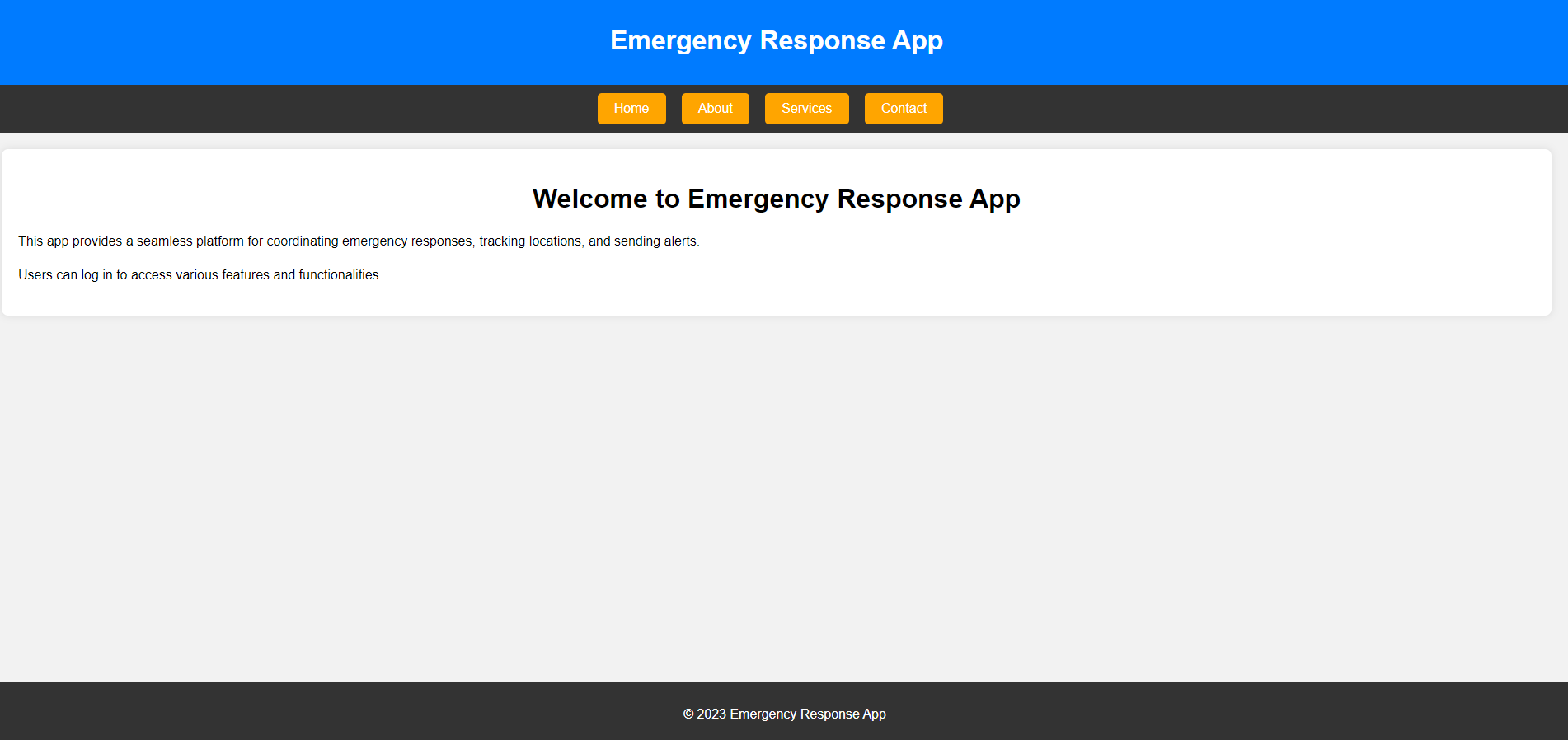
Priority: Critical

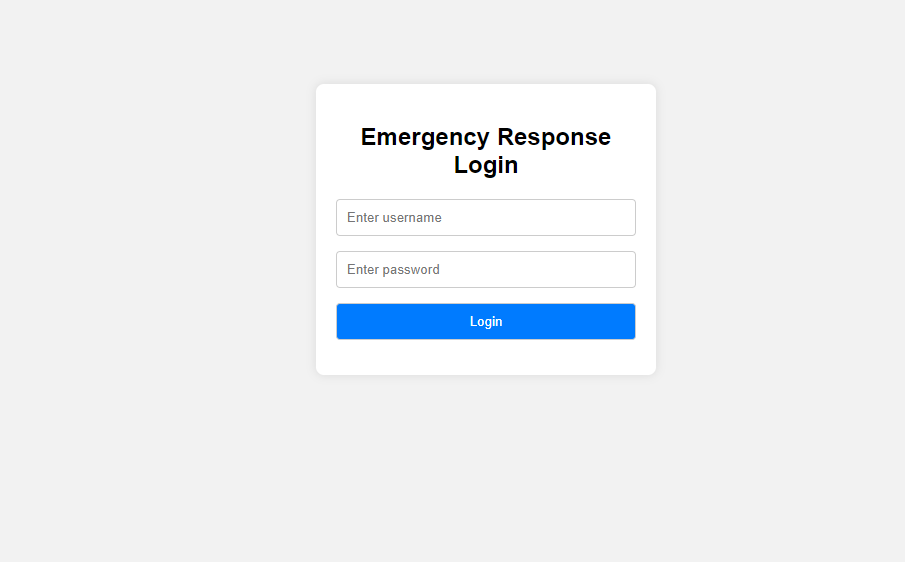
Pre-conditions: Users must have appropriate permissions and a stable network connection.

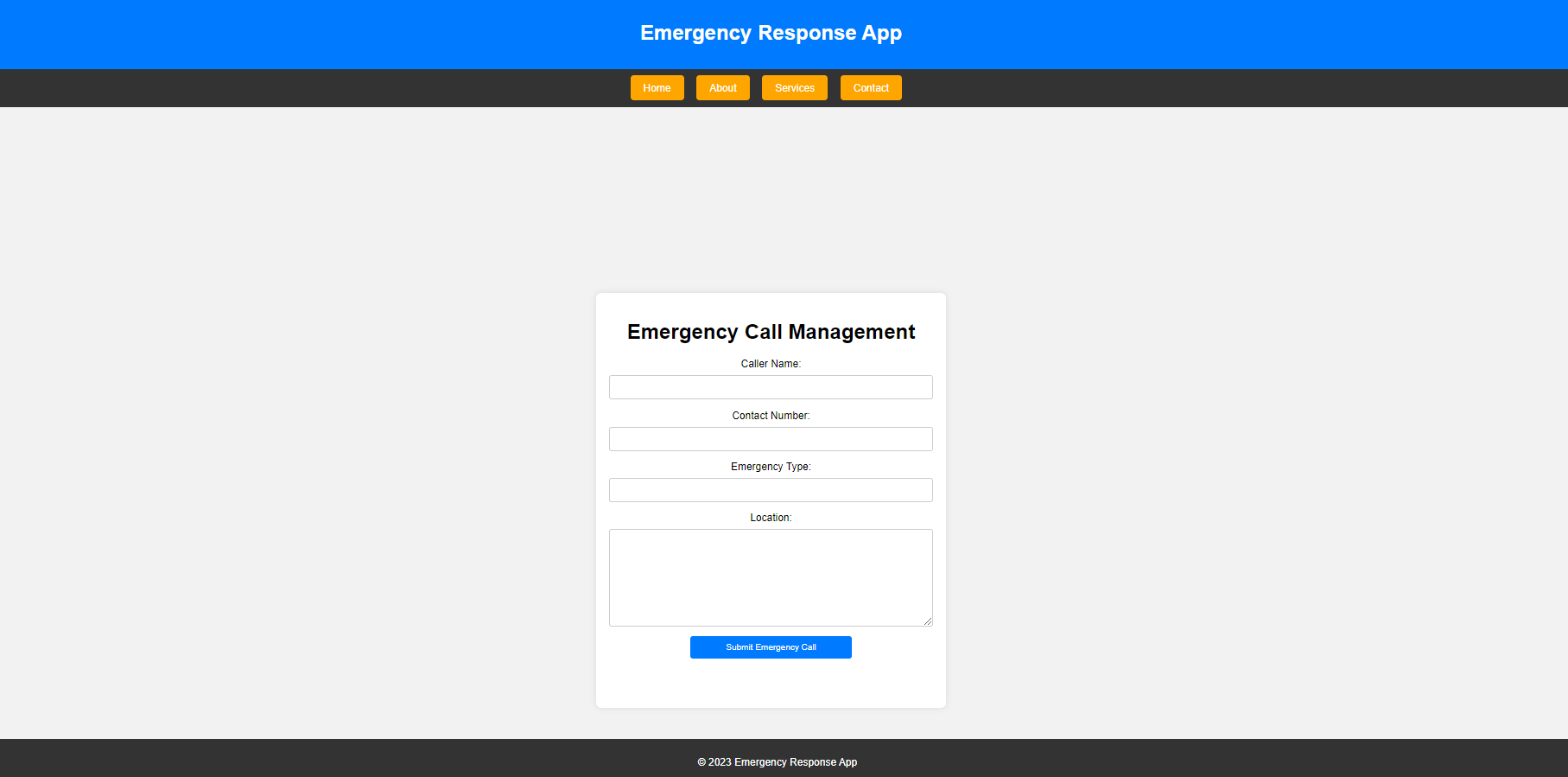
## System Quality Attributes

* + 1. **Usability:** This can be estimated with regards to usability. The system ought to be easy to understand. An everyday customer of our very own ought to have the alternative to post about his/her gadgets on a regular of 5 minutes and a restrict of 7 minutes. Our framework highlights are not difficult to advance as route is particularly basic. It's Easy for a new or uncommon user to determine out how to utilize the framework.
    2. **RELIABILITY AND CORRECTNESS:** No software is free from bug. Our software offers the actual output what user wants; its correctness has been ensured. Our software does not crash randomly as it has been tested and no false output is generated so It’s more reliable to use for our users.
    3. **Modularity**: Each system should be developed according to our modules. Our system consists of many modules and integrated to make it a complete system. So, we can easily identify the bug in any module and then we need to fix only that particular module we don't have to worry about other modules so it makes the work easier for our tester. In addition to this we can add new features to our system as it is built in modules.
    4. **Maintainability:** This means how without problems the maintenance team can perform their work. The major task of our maintenance team is to fix bugs, add something new or exchange some features. One of our maintenance programmers will be able to exchange any function with 24-man hours or much less of development effort.
    5. **Efficiency:** Main system quality features. Any given task in the system is measured in terms of time required to complete it. The efficiency of the application will be large.
    6. **TESTABILITY:** It means how easily the testing team can perform their work. Testability is highly dependent on modularity and since our system is built module wise the tester will go easy; they don't have to test every module to fix bugs.
    7. **FLEXIBILITY:** The effort required to change an operational program. A maintenance programmer with at least one year of experience supporting this product should be able to provide a new copy of output for the product, including code changes and testing, with no more than 2 hours of labor*.*

## System Interface







## Project Requirements

An algorithmic software cost estimating methodology is the Constructive Cost Model

(COCOMO). We will be using an organic software project type. It is a software project

that must be worked on in a hardware-dependent environment

**Construction Cost Model**

We are assuming that the SLOC (Source Lines of Code) that we require here after

analyzing all the components.

SLOC = 10,000 lines

Now we need to figure out the effort, development time, and required number of

people.

Constructive Cost Model

Software project type: Organic; [p=1.05]

Coefficient<Effort Factor> = 2.4

Effort = PM

So, P = 1.05 and T = 0.35

SLOC = 10,000 Lines

Persons-months, PM = Coefficient<Effort Factor> \* (SLOC / 1000) P = 2.4 \* (10000/1000) ^1.05

= 26.92

Development time, DM = 2.50 \* (PM) ^T

= 2.50 \* (26.92) ^0.35

= 7.91 = 8 months

Required number of people, ST = PM/DM

= 26.92/8

= 3.36 = 4 people

Total Development time: 8 months

Total working hours needed: (8\*22) \*8=1,408 hours; (1 month = 22 working day &

per day working time 8 hours)

Requirement analysis & Documentation times needed: 22\*8=176 hours

Times needed for Ui/Ux designing: (1\*22) \*8=176 hours

Times needed for developing system: (3\*22) \*8= 528 hours

Times needed for Testing & Debugging: (1.5\*22) \*8=264 hours

Revision time: (1.5\*22) \*8= 264 hours

For develop the software:

* Developer team of 6 engineers.
* Software Quality assurance team of 3 engineers.
* One Business Analyst
* One Ui/Ux Designer

Cost analysis:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Post** | **Total Hour** | **Salary/hour** | **Employee no** | **Total cost (bdt)** |
| BA Analyst | 176 | 300 | 1 | 52,800 |
| UI/UX designer | 176 | 130 | 1 | 22,800 |
| Developer | 528 | 280 | 6 | 8,87,040 |
| SQA Engineer | 264 | 230 | 3 | 1,82,160 |
|  | Total Cost | | | 11,44,800 |

All Cost Include like Other staff salary, office rental cost, Maintenance cost, Review cost, Market Promotion cost, lunching website cost

|  |  |
| --- | --- |
| **Cost Items** | **Total cost (bdt)** |
| Salary of the core employee | 11,44,800 |
| Other staff salary | 45,000 |
| Office rental cost | 3,00,000 |
| Maintenance cost | 10,000 |
| Review Cost | 4,000 |
| Market Promotion cost | 40,000 |
| Lunching website cost | 16,000 |
| Grand Total | 15,59,800 |

Profit: 25% of total expense = 15,59,800\* 25% = 3,89,950 TK

So Total budget = 15,59,800+ 3,89,950 = **19,49,750 TK**

# FEATURES NOT TO BE TESTED

List the areas/features that will not be specifically addressed in testing. Either all testing in these areas will be indirect as a result of other testing efforts under client control and outside of the scop of this project (e.g., interoperable systems)

**Users can contact system admin**: This feature was not been tested because user can contact emergency response team, due to low risk.

**Simulations or Training Modes:** While essential for user training, the testing of simulation or training modes might not be as critical as testing live emergency response functionalities.

**FAQ:** Ensuring that the Help or FAQ section functions as intended is crucial, but testing each and every possible query or situation in-depth may not be the top priority, particularly if it includes non-essential general information that is not needed in an emergency.

# TESTING APPROACH

## Testing Levels

SYSTEM/INTEGRATION After completing the test of all the smallest part, the system must be connected together. The 100% tested all parts of the units are converted into a module where another test is done. It’s called the integration testing. After adding all modules together, the regression testing is done to the units but in here, the main purpose is to test the interface of the system that all the connections are working perfectly or not. Also, it can be called data flow test.

* **Unit Testing:** Unit testing is the most common part of a testing where a small part of the project is going to be tested by the developers. After developing a small part of the project, this small part or portion is tested to get understand the system is working properly or not. In here test personal should find out the correctness of the inaccessible code.
* **Integration Testing:** Then in the second phase we will do the integration. In this test we will ensure that all software modules are logically integrated and tested as a group. Our project consists of several software modules written by four programmers. The purpose of this level of testing is to find errors in the way different software modules interact when integrated in this step, we will follow the "bottom-up integration" approach.
* **System Testing:** Then we will do system testing. Through system testing we will check full-featured, fully integrated systems. Then we will verify if it meets all the requirements. Black-box testing falls under this condition. So, at this level, we will follow the "Black Box Testing" technique.
* **Acceptance Testing:** The last phase of our testing is acceptance testing. We will perform this check to check the acceptability of our products. This test will be completed to check if any errors have been missed throughout the functional testing phase. At this level, we will follow the "Black Box Testing" technique. After that, we can run unit tests again. The customer will do ACCEPTANCE testing with the help of the test manager and development team leader. After the System/Integration test is completed, the acceptance test will run in parallel with the existing manual ZIP/FAX process for one month.

## Test Tools

**Design:**

1. **Visual Studio :** Helps in designing and prototyping user interfaces.

**Development:**

1. **Integrated Development Environment (IDE):**
   * **Visual Studio Code:** Lightweight code editor with robust features.
   * **Notepad :** Notepad used for Coding HTML and CSS Design
2. **Project Management:**
   * **Jira:** For agile project management, issue tracking, and bug tracking.
   * **BigGantt :** Used For Gantt Chart

**Testing:**

1. **Testing Frameworks:**
   * **Jira (already mentioned above):** Can be used for bug tracking as well.

## Meetings

The testing team will meet every week and judge the system quality and preparing for the next step. The team leader will meet the developer and testers timely and check the progress of the system. Then the project manager will check the overall process of the system time to time. These two meetings are arranged on different weeks. Emergency meeting is called as needed for emergency situation.

# TEST CASES/TEST ITEMS

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Project Name: Emergency Response and Coordination App | | | Test Designed by: Nesar Uddin | | |
| Test Case ID: FR\_1 | | | Test Designed date: 22 December 2023 | | |
| Test Priority (Low, Medium, High): Medium | | | Test Executed by: Nesar Uddin | | |
| Module Name: Login Session | | | Test Execution date:24 December 2023 | | |
| Test Title: verify login with valid username and password | | |  | | |
| Description: Test website login page | | |  | | |
| Precondition (If any): User must have valid username and password | | | | | |
| Test Steps | Test Data | Expected Results | | Actual Results | Status (Pass/Fail) |
| 1. Go to the website 2. Enter username 3. Enter password 4. Click submit | Username: 99999999999  Password: 321 | User should login into the application | | As expected, | Pass |
| Post Condition: User is validated with database and successfully login to account. The account session details are logged in the database. | | | | | |

Table 1: Test Case for **Login Session**

Table 2: Test Case for **User Profile Update**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Project Name: Emergency Response and Coordination App | | | Test Designed by: Nesar Uddin | | |
| Test Case ID: FR\_2 | | | Test Designed date: 22 December | | |
| Test Priority (Low, Medium, High): Medium | | | Test Executed by:MEHRUB KHAN | | |
| Module Name: User Profile Update | | | Test Execution date: 24 December 23 | | |
| Test Title: Verify user profile update functionality | | |  | | |
| Description: Test the ability to update user profile information | | |  | | |
| Precondition (If any): User must be logged in and have access to profile settings | | | | | |
| Test Steps | Test Data | Expected Results | | Actual Results | Status (Pass/Fail) |
| 1. Go to the profile settings page 2. Modify the user's name and email 3. Save the changes | Updated Name: John Doe  Updated Email: johndoe@example.com | The user profile should reflect the updated information. | | Updated information is reflected in the user profile. | Pass |
| Post Condition: User is updated user name and email in the database. | | | | | |

Table 3: Test Case for **Emergency Alert Broadcast**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Project Name: Emergency Response and Coordination App | | | Test Designed by: Nesar Uddin | | |
| Test Case ID: FR\_3 | | | Test Designed date: 22 December 2023 | | |
| Test Priority (Low, Medium, High): Medium | | | Test Executed by: Nesar Uddin | | |
| Module Name: Emergency Alert Broadcast | | | Test Execution date: 24 December 23 | | |
| Test Title: Test emergency alert broadcasting functionality | | |  | | |
| Description: Verify the system's capability to broadcast emergency alerts to users | | |  | | |
| Precondition (If any): Emergency alert functionality enabled | | | | | |
| Test Steps | Test Data | Expected Results | | Actual Results | Status (Pass/Fail) |
| 1. Initiate an emergency alert  2. Send the alert to all registered users/devices | Emergency message: "Fire Alert at Building A" | All registered users/devices should receive the emergency alert promptly. | | Emergency alert received by all registered users/devices. | Pass |
| Post Condition: User Notify about emergency Alert Broadcast | | | | | |

Table 4: Test Case for **Location Tracking During Emergency Response**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Project Name: Emergency Response and Coordination App | | | Test Designed by: Nesar Uddin | | |
| Test Case ID: FR\_4 | | | Test Designed date: 22 December 2023 | | |
| Test Priority (Low, Medium, High): Medium | | | Test Executed by: MEHRUB KHAN | | |
| Module Name: Location Tracking during Emergency Response | | | Test Execution date: 24 December | | |
| Test Title: Validate location tracking during emergency response | | |  | | |
| Description : Test the accuracy of location tracking for emergency responders | | |  | | |
| Precondition (If any): Emergency response feature activated, responders logged in | | | | | |
| Test Steps | Test Data | Expected Results | | Actual Results | Status (Pass/Fail) |
| 1. Activate emergency response mode for a responder  2. Track the responder's location in real-time | Responder's assigned location: Building B, Floor 3 | The system should accurately display the real-time location of the responder. | | Real-time location updates accurately reflect the responder's movements.. | Pass |
| Post Condition: User can tracking location during Emergency Response | | | | | |

# ITEM PASS/FAIL CRITERIA

**Test Case: Login**

* Pass Criteria:
  + User enters valid credentials (username/email and password).
  + Upon entering valid credentials, the user is successfully logged in.
  + After successful login, the user is directed to the app's dashboard or designated landing page.
* Fail Criteria:
  + User enters invalid credentials (incorrect username/email or password).
  + The login process fails, and the user receives an error message indicating invalid credentials.
  + The system allows access with incorrect credentials or without proper authentication.

**Test Case: User Profile Update**

* + Pass Criteria:
    - User can successfully update their profile information (name, email, etc.).
    - Updated information reflects correctly in the user's profile.
  + Fail Criteria:
    - User updates the profile, but the changes do not get saved/displayed.
    - The updated information is incorrect or doesn't match the input.

**Test Case: Emergency Alert Broadcast**

* + Pass Criteria:
    - The emergency alert is successfully broadcast to all relevant users/devices.
    - Users receive the alert promptly without delays.
  + Fail Criteria:
    - The emergency alert fails to reach all intended recipients.
    - Users receive the alert with significant delays or not at all.

**Test Case: Location Tracking during Emergency Response**

* + Pass Criteria:
    - The app accurately tracks the location of emergency responders in real-time.
    - Location updates are frequent and precise.
  + Fail Criteria:
    - Location tracking shows incorrect or outdated information.
    - There are delays or inconsistencies in updating the responder's location.

# TEST DELIVERABLES

For your "Emergency Response and Coordination App" Software Quality and Testing Project, here's a list of test deliverables or documents/materials that would be important to deliver along with the testing process:

1. **Test Plan:**
   * Document outlining the testing approach, scope, objectives, resources, schedule, and overall strategy for testing the emergency response app.
2. **Test Cases:**
   * Detailed test cases covering functional, non-functional, integration, and regression test scenarios. This includes inputs, actions, expected results, and any preconditions/post-conditions.
3. **Test Scripts:**
   * Automated test scripts created using testing frameworks/tools for automated testing purposes.
4. **Test Data:**
   * A set of data used for testing different scenarios, including both valid and invalid inputs.
5. **Test Logs/Reports:**
   * Records of test execution activities, including test results, issues encountered, steps taken, and their resolutions.
6. **Defect Reports:**
   * Documents containing details of identified defects, including their severity, priority, steps to reproduce, and status.
7. **Traceability Matrix:**
   * A document mapping the relationship between test cases and requirements, ensuring all requirements have associated test cases.
8. **Testing Environment Setup Guide:**
   * Instructions on how to set up the testing environment, including hardware, software, configurations, and dependencies required for testing.
9. **User Acceptance Testing (UAT) Documentation:**
   * Materials for end-users to conduct UAT, including test cases, scenarios, and instructions.
10. **Test Summary Report:**
    * A comprehensive report summarizing the testing activities, including test coverage, test execution results, defects found, and overall assessment of the product quality.
11. **Release Notes:**
    * Documentation highlighting the changes, enhancements, fixes, and any known issues associated with the specific release/version of the app.
12. **Training Materials (if applicable):**
    * Documentation or guides to train testers or end-users on using the app and performing testing activities effectively.

# STAFFING AND TRAINING NEEDS

This section shows how to staff the test jobs and prepare them for the work. Staffing is set for the duration of the project. It's realistic to assume that the vast majority of the staff will agree to do some testing. The following occupations are recognized:

Project Manager: Responsible for maturing the complete execution of the Web website. This includes creating requirements, managing the seller relationship, and overseeing the testing cycle.

Test Manager: Responsible for fostering the expert test strategy, examining the test deliverable, dealing with test cycles, collecting measurements and reporting progress to the Project Manager, and recommending when testing should be completed.

Test Engineer: Planning tests, creating test methods, creating test information, running tests, preparing occurrence reports, examining episodes, writing mechanized test strategies, and detailing measurements to the test administrator are all responsibilities of this position.

The test manager and test specialists should be familiar with the website development life cycle. Because this project is being developed in a traditional manner, this is a nonexclusive depiction of Staffing and Training requirements. As a result, the names of conscious people for each project aren't given.

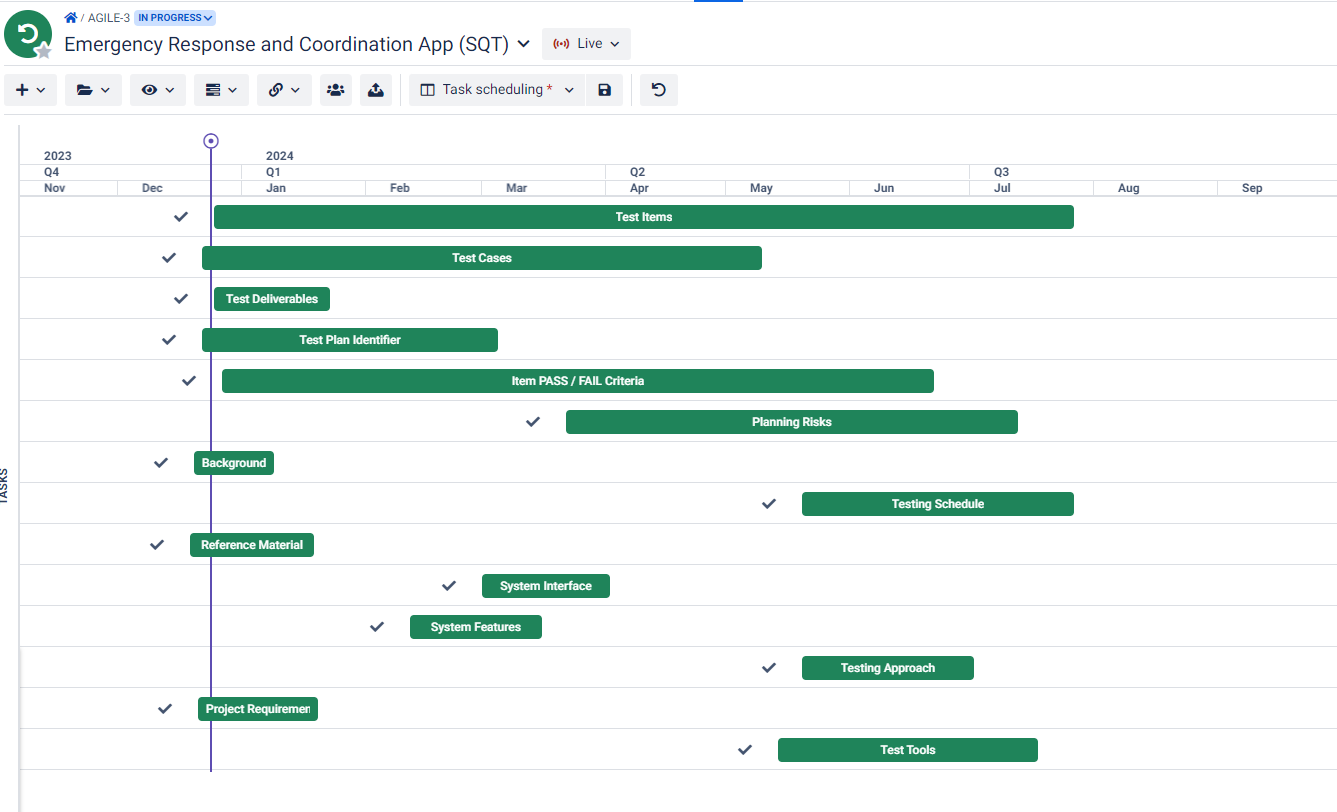
# RESPONSIBILITIES

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | TM | PM | Dev Team | Test Team | Client |
| Acceptance Test Documentation & Execution | X | X |  | X | X |
| System Test Documentation & Execution |  |  | X | X |  |
| Unit Test Documentation & Execution |  |  | X | X |  |
| System Design Reviews |  |  | X | X | X |
| Details Design Reviews |  |  |  |  |  |
| Test Procedures and rules | X | X | X |  |  |
| Screen & Report Prototype reviews |  |  | X | X | X |
| Change control regression testing | X | X | X | X |  |

# TESTING SCHEDULE

# 

**Figure : Project Schedule**



**Figure : Project Schedule (Gantt Chart )**

# PLANNING RISKS AND CONTINGENCIES

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S/N** | **Risk Factor** | **Possibility** | **Impact on Testing Activities** | **Mitigation Plan** |
| 01 | Insufficient Test Coverage | High | 1. Inadequate identification of defects leading to higher post-release issues | Develop a comprehensive test strategy, including risk-based testing. Prioritize critical features and functionalities for thorough testing. Regularly review and update test coverage. |
| 02 | Unstable Testing Environment | High | Disruptions in test execution, leading to unreliable test results. | Stabilize the environment by addressing infrastructure issues. Use virtual environments where possible for consistency. Implement version control for configurations. |
| 03 | Requirement Changes | High | Test cases becoming obsolete leading to rework and delays. | Maintain open communication channels with stakeholders. Implement change control processes to manage and track requirement changes. Regularly update test documentation. |
| 04 | Tool Incompatibility | Medium | Disruption in test automation impacting productivity | Regularly update tools, ensure compatibility with different environments, and have fallback options available. |