**Batch: A-3 Roll No.: 16010122104**

**Experiment / assignment / tutorial No. 7**

|  |
| --- |
| **Title: Designing test plan document for Mini Project** |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Aim:** To learn and understand the way of developing the software by classical methods of software engineering. Planning and monitoring, testing, validating of the project using tools and prepares a document for the same by using the concept of software engineering

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**CO:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

1. Roger Pressman, Software Engineering: A practitioners Approach, McGraq Hill, 2010 ,6th edition

2. Ian Somerville , Software Engineering , Addison Wesley,2011,9th edition

1. http://en.wikipedia.org/wiki/Software\_requirements\_specification

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Test Plan Template:**

ManSeek: Mental health App.

**Prepared by:**

Kashish Mamania

17/10/2024

**TABLE OF CONTENTS**

1.0 INTRODUCTION

2.0 OBJECTIVES AND TASKS  
2.1 Objectives  
2.2 Tasks

3.0 SCOPE

4.0 Testing Strategy  
4.1 Alpha Testing (Unit Testing)  
4.2 System and Integration Testing  
4.3 Performance and Stress Testing  
4.4 User Acceptance Testing  
4.5 Batch Testing  
4.6 Automated Regression Testing  
4.7 Beta Testing

5.0 Hardware Requirements

6.0 Environment Requirements  
6.1 Main Frame  
6.2 Workstation

7.0 Test Schedule

8.0 Control Procedures

9.0 Features to Be Tested

10.0 Features Not to Be Tested

11.0 Resources/Roles & Responsibilities

12.0 Schedules

13.0 Significantly Impacted Departments (SIDs)

14.0 Dependencies

15.0 Risks/Assumptions

16.0 Tools

17.0 Approvals

**1.0 INTRODUCTION**

Overview  
ManSeek is a mental health and wellness application designed to help users track their moods, sleep patterns, and mindfulness activities. The application features user registration, mood tracking, sleep tracking, guided mindfulness sessions, and crisis support options.

Functions

* User Registration and Profile Management: Users can create and manage their profiles.
* Mood Tracking: Users can log, edit, and view their mood history.
* Sleep Tracking: Users can log their sleep hours and view sleep statistics.
* Guided Mindfulness Sessions: Users can participate in guided sessions for relaxation and stress relief.
* Crisis Support: Provides users with access to crisis helpline numbers and resources.

**2.0 OBJECTIVES AND TASKS**

2.1 Objectives

* Define testing tasks and responsibilities.
* Serve as a communication vehicle among stakeholders.
* Provide a service level agreement for the project stakeholders.
* Ensure comprehensive testing coverage of all functionalities.

2.2 Tasks

* Testing: Unit Testing, System and Integration Testing, Performance and Stress Testing, User Acceptance Testing, Automated Regression Testing, Beta Testing.
* Post-Testing: Reporting results, tracking issues, and documenting findings.
* Problem Reporting: Identifying and documenting any defects or issues found during testing.

**3.0 SCOPE**

General

The test plan covers all functionalities of the ManSeek application, including user registration, mood and sleep tracking, mindfulness sessions, and crisis support features.

Tactics

To accomplish the testing objectives:

* Coordinate with development and user experience teams to access the application and its features.
* Schedule testing sessions with relevant stakeholders to ensure functionality is verified across different user perspectives.
* Notify key individuals to represent their respective areas during testing activities.

**4.0 TESTING STRATEGY**

The overall testing approach includes various testing methodologies aimed at ensuring comprehensive coverage of all features.

**4.1 Unit Testing**

**Definition**  
Unit testing will validate individual components of the application for correct functionality.

**Participants**

* Developers
* QA Engineers

**Methodology**

* Test scripts will be written by developers for each component.
* The sequence of events will include writing tests, executing them, and reviewing results.

**4.2 System and Integration Testing**

**Definition**  
System and integration testing will validate the interactions between different modules and the overall system.

**Participants**

* QA Engineers
* Integration Specialists

**Methodology**

* Test scripts will be written by QA Engineers.
* The sequence will include executing integration tests after unit tests are passed and assessing the system's overall functionality.

**4.3 Performance and Stress Testing**

**Definition**  
Performance testing assesses how the application performs under load, while stress testing evaluates how it handles extreme conditions.

**Participants**

* Performance Test Engineers

**Methodology**

* Performance scripts will be developed to simulate multiple users accessing the application.
* The sequence will involve running the application under increasing loads and measuring response times.

**4.4 User Acceptance Testing**

**Definition**  
User acceptance testing confirms that the system meets end-users' needs and requirements.

**Participants**

* End-users
* Product Manager

**Methodology**

* Test cases will be written by QA Engineers in collaboration with end-users.
* End-users will execute the test scripts to validate functionality against requirements.

**4.5 Batch Testing**

**Definition**  
Batch testing will assess the performance of multiple data entries processed together.

**Participants**

* QA Engineers

**Methodology**

* Test scripts will be written to handle batch processing scenarios.
* The process will involve executing test cases to verify the system can handle batch data inputs efficiently.

**4.6 Automated Regression Testing**

**Definition**  
Regression testing will ensure that new changes do not adversely affect existing functionalities.

**Participants**

* Automation Test Engineers

**Methodology**

* Automated test scripts will be created for core functionalities.
* Tests will be executed after every major change or release.

**4.7 Beta Testing**

**Participants**

* Selected end-users

**Methodology**

* Users will be invited to test the application in a real-world environment, providing feedback on usability and functionality.

**5.0 HARDWARE REQUIREMENTS**

Computers: Windows or Mac OS based systems with at least 8GB RAM.

Modems: Reliable internet connection for testing online functionalities.

**6.0 ENVIRONMENT REQUIREMENTS**

**6.1 Main Frame**

* **Hardware**: Servers running Linux/Windows OS.
* **Software**: Required applications, including the ManSeek application, database management systems, and necessary security protocols.
* **Security**: High-level security measures to protect user data.

**6.2 Workstation**

* Testers will need access to configured workstations equipped with necessary testing tools.

**7.0 TEST SCHEDULE**

**8.0 CONTROL PROCEDURES**

Problem Reporting

* Issues identified during testing should be logged using a standardized issue tracking system.
* An incident report form will be attached as an appendix to the Test Plan.

Change Requests

* Modifications will require approval from the Project Manager and will be documented for transparency.

**9.0 FEATURES TO BE TESTED**

**9.0 FEATURES TO BE TESTED**

* User registration
* Mood tracking
* Sleep tracking
* Guided mindfulness sessions
* Crisis support functionality

**10.0 FEATURES NOT TO BE TESTED**

External integrations that are not part of the current release.

Features scheduled for future development cycles.

**11.0 RESOURCES/ROLES & RESPONSIBILITIES**

**QA Manager**: Oversee the testing process.

**Test Engineers**: Execute test cases and report findings.

**Developers**: Assist in resolving identified issues.

**12.0 SCHEDULES**

**13.0 SIGNIFICANTLY IMPACTED DEPARTMENTS (SIDs)**

Development Team

Product Management

Customer Support

**14.0 DEPENDENCIES**

Availability of test environments.

Access to user data for testing.

**15.0 RISKS/ASSUMPTIONS**

**Risk**: Delays in user feedback may extend testing timelines.

**Contingency Plan**: Schedule additional testing sessions if necessary.

**16.0 TOOLS**

**Automation Tools**: Selenium

**Bug Tracking Tool**: JIRA.

**17.0 APPROVALS**

**Post Lab Descriptive Questions:**

1. **Distinguish between Black Box and White Box Testing**
2. Consider following scenario: An institute is interested in developing a Library Information System (LIS) for the benefit of students and employees of the institute. LIS will enable the members to borrow a book (or return it) with ease while sitting at his desk/chamber. The system also enables a member to extend the date of his borrowing if no other booking for that particular book has been made. For the library staff, this system aids them to easily handle day-to-day book transactions. The librarian, who has administrative privileges and complete control over the system, can enter a new record into the system when a new book has been purchased, or remove a record in case any book is taken off the shelf. Any non-member is free to use this system to browse/search books online. However, issuing or returning books is restricted to valid users (members) of LIS only.

The final deliverable would a web application (using the recent HTML 5), which should run only within the institute LAN. Although this reduces security risk of the software to a large extent, care should be taken no confidential information (e.g. passwords) is stored in plain text.

**R1: New User Registration and Profile Management**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | Summary | Dependency | Pre-condition | Post-condition | Execution Steps | Expected Output |
| TC1 | Verify that a new user can register successfully | None | User is not registered | User is registered | 1. Navigate to the registration page 2. Enter valid details 3. Click 'Register' | Registration successful, redirected to profile |
| TC2 | Verify that the user cannot register with an already registered email | None | Email already exists | User is not registered | 1. Navigate to registration page 2. Enter existing email 3. Click 'Register' | Error: "Email already registered" |
| TC3 | Verify that a user cannot register without mandatory fields | None | No data entered | User is not registered | 1. Leave fields blank 2. Click 'Register' | Error: "Please fill all required fields" |
| TC4 | Verify that a user receives a confirmation email post-registration | TC1 | User registered | Confirmation email sent | 1. Complete registration 2. Check email | Confirmation email received |
| TC5 | Verify that the user can update their profile after registration | None | User is logged in | Profile is updated | 1. Navigate to profile 2. Update details 3. Save changes | Profile updated successfully |
| TC6 | Verify that a user cannot update profile with invalid email format | None | Invalid email entered | Profile not updated | 1. Enter invalid email 2. Click 'Save' | Error: "Invalid email format" |
| TC7 | Verify that a user can change their password | None | User is logged in | Password is changed | 1. Navigate to profile 2. Change password 3. Confirm | Password updated successfully |

**R2: User Login**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | Summary | Dependency | Pre-condition | Post-condition | Execution Steps | Expected Output |
| TC1 | Verify that a registered user can login with correct credentials | None | User is registered | User is logged in | 1. Enter valid email/password 2. Click 'Login' | User logged in, redirected to home page |
| TC2 | Verify that an unregistered user cannot login | None | User is not registered | User is not logged in | 1. Enter unregistered credentials 2. Click 'Login' | Error: "Invalid credentials" |
| TC3 | Verify that a registered user cannot login with incorrect password | None | User is registered | User is not logged in | 1. Enter correct email, wrong password 2. Click 'Login' | Error: "Incorrect password" |
| TC4 | Verify that a user is locked out after 3 failed login attempts | TC3 | User has failed login 3 times | User is locked out | 1. Enter wrong password 3 times 2. Try again | Error: "Account locked after 3 failed attempts" |
| TC5 | Verify that a user receives a password reset email after requesting | None | User is registered | Reset email sent | 1. Click 'Forgot Password' 2. Enter email 3. Check email | Password reset email received |
| TC6 | Verify that user cannot login with an expired session | None | User logged in, session expired | User is logged out | 1. Wait until session expires 2. Attempt an action | Error: "Session expired, please login again" |
| TC7 | Verify that user can logout successfully | None | User is logged in | User is logged out | 1. Click 'Logout' | User logged out, redirected to login page |

**R3: Mood Tracking**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | Summary | Dependency | Pre-condition | Post-condition | Execution Steps | Expected Output |
| TC1 | Verify that the user can log their mood for the day | None | User is logged in | Mood logged | 1. Go to mood tracking page 2. Select mood 3. Submit | Mood successfully logged |
| TC2 | Verify that user cannot log mood more than once a day | None | Mood already logged | Error displayed | 1. Log mood once 2. Try logging again | Error: "Mood already logged for today" |
| TC3 | Verify that a user can view mood history for the last week | None | Mood data available | History displayed | 1. Go to mood history 2. Select date range | Mood history displayed |
| TC4 | Verify that user can edit a mood entry within the same day | None | Mood logged today | Entry updated | 1. Log mood 2. Edit entry 3. Submit | Mood entry updated |
| TC5 | Verify that mood entries are synchronized across devices | None | User logged in on multiple devices | Entries synchronized | 1. Log mood on one device 2. Check another device | Entry synchronized |
| TC6 | Verify that user receives reminders to log their mood daily | None | User is registered, logged in | Reminder received | 1. Set reminder 2. Wait for reminder | Daily reminder received |
| TC7 | Verify that user can filter mood history by time range | None | Mood history available | Filter applied | 1. Navigate to mood history 2. Apply date filter | Filtered history displayed |

**R4: Sleep Tracking**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | Summary | Dependency | Pre-condition | Post-condition | Execution Steps | Expected Output |
| TC1 | Verify that user can manually log their sleep hours | None | User is logged in | Sleep hours logged | 1. Go to sleep tracking page 2. Enter sleep hours 3. Submit | Sleep logged successfully |
| TC2 | Verify that user can sync sleep data from a wearable device | None | Device connected | Data synchronized | 1. Connect wearable 2. Sync data | Sleep data synced successfully |
| TC3 | Verify that user can view sleep history for a week | None | Sleep data available | History displayed | 1. Navigate to sleep history 2. Select week | History displayed |
| TC4 | Verify that user can receive sleep tips based on logged data | None | Sleep data available | Tips provided | 1. Navigate to sleep tips 2. View recommendations | Personalized sleep tips shown |
| TC5 | Verify that user receives sleep improvement notifications | None | User is logged in | Notification received | 1. Enable notifications 2. Wait for tip | Notification received with tips |
| TC6 | Verify that user can set a bedtime reminder | None | User logged in | Reminder set | 1. Go to settings 2. Set reminder 3. Wait for reminder | Bedtime reminder received |
| TC7 | Verify that user can view average sleep duration for the last month | None | Sleep data available | Average displayed | 1. Go to sleep history 2. View monthly average | Average sleep time displayed |

**R5: Guided Mindfulness and Relaxation Techniques**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | Summary | Dependency | Pre-condition | Post-condition | Execution Steps | Expected Output |
| TC1 | Verify that user can start a guided mindfulness session | None | User logged in | Session started | 1. Go to mindfulness page 2. Select session 3. Click 'Start' | Session started successfully |
| TC2 | Verify that user can pause a mindfulness session | TC1 | Session ongoing | Session paused | 1. Start session 2. Click 'Pause' | Session paused |
| TC3 | Verify that user can resume a paused session | TC2 | Session paused | Session resumed | 1. Click 'Resume' | Session resumed successfully |
| TC4 | Verify that user receives a completion message at the end of a session | TC1 | Session completed | Message displayed | 1. Complete session 2. View completion | "Session Completed" message shown |
| TC5 | Verify that user can download session statistics | TC1 | Session completed | Stats downloaded | 1. Complete session 2. Click 'Download' | Session statistics downloaded |
| TC6 | Verify that user can track progress over multiple sessions | TC5 | Sessions completed | Progress tracked | 1. Complete multiple sessions 2. View progress tracker | Progress tracking shown |
| TC7 | Verify that user can exit a session early without saving progress | TC1 | Session ongoing | Session exited | 1. Start session 2. Click 'Exit' | Session exited without saving progress |