**Ctrl-Alt-Elite**

**Mindlift**

**TEST PLAN**

Date: 10/18/2023

Table of Contents

[Introduction 3](#_Toc109387551)

[1.1 Objectives 3](#_Toc109387552)

[1.2 Team Members 3](#_Toc109387553)

[2 Scope 3](#_Toc109387554)

[3 Assumptions / Risks 4](#_Toc109387555)

[3.1 Assumptions 4](#_Toc109387556)

[3.2 Risks 4](#_Toc109387557)

[4 Test Approach 4](#_Toc109387558)

[4.1 Test Automation 4](#_Toc109387559)

[5 Test Environment 5](#_Toc109387560)

[6 Milestones / Deliverables 5](#_Toc109387561)

[6.1 Test Schedule 5](#_Toc109387562)

[6.2 Deliverables 5](#_Toc109387563)

# Introduction

The Test Plan has been created to communicate the test approach to team members. It includes the objectives, scope, schedule, risks and approach. This document will clearly identify what the test deliverables will be and what is deemed in and out of scope.

## Objectives

The purpose of the application is to allow a user struggling with depression, or any form of mental disorder, to communicate with an AI. The application, on top of the chat feature, also stores conversations for later viewing, seeks correlations between how a user is feeling and the events throughout their day, allows the user to store their mood throughout the day, and features a crisis detection and prevention ability.

## Team Members

|  |  |
| --- | --- |
| **Resource Name** | **Role *(examples are given below)*** |
| Austin Hoffman | Developer |
| Akash Patel | Developer |
| Hadia Bilal | Developer |
| Shawana Tahseen | Developer |
| Timmy Akindunni | Developer |

# Scope

The initial sprint will include ‘must have’ requirements. These and any other requirements that get included must all be tested.

The following sections indicate what is tested during each sprint. The scope of testing is determined at the beginning of the current sprint.

At the end of Sprint 1, a user must be able to:

1. Navigate through the app to each screen. R10
2. See what kind of content each screen will contain. R10

At the end of Sprint 1, the application must:

1. Be usable on both IOS and Android.

At the end of Sprint 2, a user must be able to:

1. Decide at the start of sprint 2

At the end of Sprint 3, a user must be able to:

1. Decide at the start of sprint 3

**Assumptions / Risks**

## Assumptions

1. The user has a mobile device that is listed as compatible and has used it long enough to be proficient in navigating apps and the Appstore.
2. The user speaks, reads, and writes English.
3. The user has some level of awareness regarding the performance of local AI.
4. The user has hands, or some way of interacting with onscreen UI.

## Risks

The following risks have been identified and the appropriate action identified to mitigate their impact on the project. The impact (or severity) of the risk is based on how the project would be affected if the risk was triggered. The trigger is what milestone or event would cause the risk to become an issue to be dealt with.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Risk | Impact | Trigger | Mitigation Plan |
| 1 | The local AI does not respond meaningfully | Medium | Feelings of disconnect between user and application. | AI’s response creativity will be increased and more training data will be given. |
| 2 | The local AI responds with harmful responses | High | A response that provides harmful advice, abuse, or triggering statements. | AI creativity will be reduced until training data has been confirmed safe, and model response quality is above the acceptable threshold. |
| 3 | AI performance is too low | Medium | Response time is greater than five seconds. | AI parameter amount will be reduced. Training set will be trimmed. |
| 4 | Application drains too much battery while not actively in use | Medium | Battery drain on device is significant, draining 2-3 times more battery than other sleeping applications. | Optimize the crisis detection feature. Lessen the number of sounds to look for to initiate crisis prevention. |
| 5 | Training set required is too large | Medium | The current training set does not produce responses in the AI that meet the quality threshold. | Gather another five thousand examples. |
| 6 | AI model becomes too large to use with GitHub | Low | Github does not allow upload of model. | Store the model in Git LFS. |
| 7 | Training a model from scratch becomes too demanding on time | High | Estimates of model training exceed allotted project time. | Curate and specialize a pretrained model. |

# Test Approach

The project is using an agile approach, with 3-week sprints. *Mention how you will conduct testing during the sprint in terms of the techniques you plan to do and when. Add a new subsection for each sprint.*

Sprint 1

Sprint 2

Sprint 3

## Test Automation

*Discuss the role of automated testing and how you plan to conduct any (including tools).*

## Test Cases (Black Box)

### Feature 1 (Name this based on the name in your Req doc)

*Have a table for the test cases needed to test the User Story*

| **Test Case ID** | **Description** | **Requirements Trace** | **Directions** | **Expected Output** |
| --- | --- | --- | --- | --- |
| TC-001 | UI Accessibility through the WWW. | List the user story/stories this is linked to | 1. Open a web browser and go to the address of the RRS. | The home page for the RRS should be displayed. |
|  |  |  |  |  |
|  |  |  |  |  |

### Feature n (Name this based on the name in your Req doc )

*Have a table for the test cases needed to test the User Story. A sample is below*

| **Test Case ID** | **Description** | **Requirements Trace** | **Directions** | **Expected Output** |
| --- | --- | --- | --- | --- |
| TC-101 | Prepare to request a reservation at a restaurant. | RES-002  RES-003  RES-005  RES-006 | 1. Execute TCB-004. | All fields necessary to fill in the data specified in the directions are present. |
| TC-102 | Request a reservation at a restaurant. | RES-001 | 1. Execute TC-101. 2. Click “Continue” button. 3. Click “Make Reservation!” button. | A table reservation confirmation page should be displayed. Otherwise, a page should be displayed saying that the reservation could not be made. |
| TC-103 | Cancel a previous reservation. | RES-007 | 1. Successfully make a reservation using TC-102. 2. Click “Return to Restaurant” button. 3. Click “Modify / Cancel Existing Reservation” button. 4. Enter confirmation number received from TC-003 in the “Reservation Confirmation Number” text box. 5. Enter the email address used in TC-003 in the “Email Address” text box. 6. Click “Modify/Cancel Reservation” button. 7. Click “Cancel Reservation” button. | A reservation cancellation confirmation page should be displayed. |

## Test Cases (White Box)

3.3.1..n Name by Feature (like above)

*Organize the test cases using the table below. Be sure to have a directory in your repository for your test suite that follows a naming scheme that matches the items in the table below.*

*Like section 3.2, you will add to the table as you progress through each sprint.*

| **Test Case ID** | **Description** | **Directions/Goals** | **Expected Output** |
| --- | --- | --- | --- |
| TC-1001 | Selecting a restaurant to visit. | Follow TCB-001 when there are no restaurants available. Does the system handle not having a restaurant to select? | Should load the home page without errors. However, there are no database checks for null pointers so if the DB fails, there will be an error. |
| TC-1002 | Attempt to go to a restaurant page without selecting a restaurant. | Follow TCB-001 only do not select a restaurant. Does the system detect that no restaurant is selected. | The system should force the user to select a restaurant. Should fail however since there is no null pointer check in the code. |
| TC-1003 | Check site links. | Browse all available pages clicking on each link available to make sure they point to pages. Also, check graphics as well. | Each page should link properly to intended pages. |
| TC-1004 | Ensure data validation is working properly. | Follow TCB-006 only with improper input by leaving fields blank. Also attempt to use improperly formatted email and phone numbers, i.e. The last name for both fields. | System should check to make sure that fields are filled in and warned about otherwise. If there is an improper format for the email and phone number, that should be marked as well. |
| TC-1005 | Attempt to use the system when the database is down. | Access any page without the database running. Check to make sure the error seems reasonable. | Although not a requirement, if the database connection cannot be made, a reasonable error message would be nice. |
| TC-1006 | Attempt to use the system when JavaScript is disabled in the user’s browser. | Can you still make a reservation with blanks in the field if you disable JavaScript? See how verbose the error handling is. | Since JavaScript does the error handling, not the code, disabling JavaScript should allow bad input. |
| TC-1007 | Attempt to use apostrophes in text fields whose values get used in SQL statements. | Follow TCB-006 but use O’Conner as the last name.  Checking to make sure data checking includes escaping characters. | SQL uses the apostrophe character as a special character, so does the system escape the apostrophe character if it appears in a name to prevent a bad SQL call. |
| TC-1008 | Bounds checking on the Erlang implementation. | Use the Erlang page and try using a blank field or negative numbers. | Should fail since there is no error handling to round or check the input to the Erlang implementation. |
| TC-1009 | Add an incentive directly to the database and ensure it is displayed. | 1. Open a web browser and go to <http://rrs.se.rit.edu/rrs/sql.jsp>. 2. Enter the SQL statement “INSERT INTO incentives (restaurant\_id, start\_time, end\_time, description) VALUES (0, sysdate, TO\_DATE(‘MM/DD/YYYY’, ‘12/31/2002’), ‘<B>TC-1009 Incentive</B><BR><I>This is a new incentive!</I>’)” 3. Execute TCB-001. | The incentive just entered should be displayed on the page. |

# Test Environment

*For example, A new server is required for the web server, the application and the database.*

# Test Schedule

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task Name *(sample is below, focus on spring 1 to start)*** | **Start** | **Finish** | **Effort** | **Comments** |
| *Test Planning* |  |  |  |  |
| *Review Requirements documents* |  |  |  |  |
| *Create initial test estimates* |  |  |  |  |
| *Learn new test resources* |  |  |  |  |
| *First deploy to QA test environment* |  |  |  |  |
| *Functional testing – Sprint 1* |  |  |  |  |
| *Iteration 2 deploy to QA test environment* |  |  |  |  |
| *Functional testing – Sprint 2* |  |  |  |  |
| *System testing* |  |  |  |  |
| *Regression testing* |  |  |  |  |
| *Usability Testing* |  |  |  |  |
| *Resolution of final defects and final build testing* |  |  |  |  |
| *Deploy to Staging environment* |  |  |  |  |
| *Performance testing* |  |  |  |  |
| *Release to Production* |  |  |  |  |