UAT Plan

for

[Grow]

Index

[1. Scope 3](file:///C:\Users\ManaA\Downloads\UAT_TestPlan%20Templatev2%202.docx#_Toc139546475)

[1.1. Objectives and business requirements 3](file:///C:\Users\ManaA\Downloads\UAT_TestPlan%20Templatev2%202.docx#_Toc139546476)

[1.2. Scope 3](file:///C:\Users\ManaA\Downloads\UAT_TestPlan%20Templatev2%202.docx#_Toc139546477)

[2. Testing team 4](file:///C:\Users\ManaA\Downloads\UAT_TestPlan%20Templatev2%202.docx#_Toc139546478)

[3. Environmental requirements 5](file:///C:\Users\ManaA\Downloads\UAT_TestPlan%20Templatev2%202.docx#_Toc139546479)

[3.1. Hardware requirements 5](file:///C:\Users\ManaA\Downloads\UAT_TestPlan%20Templatev2%202.docx#_Toc139546480)

[3.2. Software requirements 5](file:///C:\Users\ManaA\Downloads\UAT_TestPlan%20Templatev2%202.docx#_Toc139546481)

[4. Test Scripts 1](file:///C:\Users\ManaA\Downloads\UAT_TestPlan%20Templatev2%202.docx#_Toc139546482)

# Scope

## Objectives and business requirements

In this section, outline the business requirements. In other words:

The goal is to create a functional application to better help students engage in studying. The application is to include a timer, tips and a planner.

*Example:*

*The goal of this user acceptance test is to ensure all the features of a website aimed at helping young people understand mental health work as designed.*

## Scope

In this section, outline the scope. This means:

* What is the pain point we’re trying to fix?
* What are we testing exactly, and what are we not testing?

The landing page of the application is simplified and easy to access and all menu (buttons) are functional.

For this UAT test, we’d like to test:

* Does the study planner page load correctly
* Is the date correct on the study planner page
* Is the clock accurate on the study timer page
* Testing if the task is added correctly

For the UAT test, we are not testing:

* Does the page look good?
* Does the tips/timer/planner page load properly?

*Example:*

*The collision detection algorithm has been refined to respond earlier and bring the robot to halt in a more controlled manner*

*For this UAT test, we’d like to test:*

* *Does the collision detection system identify solid objects*
* *Does the collision detection system begin responding earlier*
* *Does the collision detection system visual outputs work*

*For the UAT test, we are not testing:*

* *Other vehicle response mechanisms*
* *Does the collision response mechanism respond to mobile, irregular or transparent objects*

## System Diagrams

In this section, paste any drawings or diagrams that help the UAT team understand the program being tested. With each drawing include a brief explanation of how the drawing represents the application or system being tested.

*Example:*

*Storyboards, wireframes, flowcharts, schematics, pictorials, mood-boards, etc.*

# Testing team

In this section, list out members of your QA team and what their roles will be during UAT.

|  |  |
| --- | --- |
| **Name** | **Responsibilities** |
| Amber | UAT Coordinator |
|  |  |
|  |  |
|  |  |
|  |  |

*Example:*

|  |  |
| --- | --- |
| ***Name*** | ***Responsibilities*** |
| *Robert K. Wright* | *UAT Coordinator - handles communication between end users and QA team* |
| *Johannes Creusen* | *Design test cases for the accounting team* |
| *Stefan Kottila* | *Design test cases for the management team* |
| *Roxanne Gilbert* | *Create test data and write UAT reports* |
| *Claudia Decker* | *Set up staging + usability test cases and reports* |

# Environmental requirements

## Hardware requirements

What hardware has the solution been designed for and should be tested on.

If that is the case, outline the minimal and recommended requirements so the QA team can verify that the software runs on the testers’ machines.

Any Laptop/Computer that can access the internet and google.

Any mobile device that can access the internet and google.

*Example:*

* *Lenovo Desktop PC*
  + *Windows 10.*
  + *Intel I5 processor.*
  + *256gb SSD.*
  + *8 GB of RAM.*
  + *Intel GPU.*
  + *Ethernet NIC.*
  + *LED 1080p Monitor with HDMI connection.*
* *Google Pixel 5 - Mobile Phone*
  + *Android 11.*
  + *Qualcomm SM7250 Snapdragon.*
  + *1080 x 2340 pixels, 19.5:9 ratio.*
  + *4g NIC*

## Software requirements

If any extra software or dependencies must be downloaded and installed, list them here.

* Laptop/Computer
  + Google Chrome
  + Safari (if APPLE)
* Mobile Devices
  + Google Chrome
  + Safari (if APPLE)

*Example:*

* *Lenovo Desktop PC*
  + *Google Chrome.*
  + *Microsoft Edge.*
  + *Mozilla Firefox.*
  + *Mozilla Firefox.*
* *Android Mobile Phone - Pixel 5*
  + *Google Chrome.*

## Network requirements

Some software (design, video editing…) can be demanding on hardware specifications.

If that is the case, outline the minimal and recommended requirements so the QA team can verify that the software runs on the testers’ machines.

* Laptop
  + Access to Home Network
* Mobile
  + 4G or 5G network

*Example:*

* *Lenovo Desktop PC*
  + *NBN Fibe to the Node network.*
* *Android Mobile Phone - Pixel 5*
  + *Telstra 4g network.*

# Test Scripts

This section is more important than it seems—it is crucial that both the QA team and the testers know what features must be tested, especially if you’re testing a lot at once.

**4.0 Study Planner Page**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test** | **Describe the feature being tested** | **Describe the user input or test data** | **Describe the pass criteria** |  |
| 1.0 | Loading Study Planner Page | 1. User clicks the “Study Planner” button | 1. User should see Study Planner title. 2. User should see a date in the corner (PC/Laptop) or in the center (Mobile) 3. User should see a text box with “Enter Task” 4. User should see an option to select the due date 5. User should see the add task, return and help button. | Tester name:   |  |  | | --- | --- | |  | PASS | |  | FAIL |   Observations: |
| 1.1 | Date is accurate and correct | 1. User clicks the “Study Planner” button 2. Compare the date with your computer/phone date | 1. The date should be the same as your system date | Tester name:   |  |  | | --- | --- | |  | PASS | |  | FAIL |   Observations: |
| 1.2 | Clock is accurate and correct | 1. User clicks the “Study Timer” button 2. Compare the date with your computer/phone date | 1. The time should the same as your system date | Tester name:   |  |  | | --- | --- | |  | PASS | |  | FAIL |   Observations: |

Tip: Write step-by-step, detailed but concise instructions on how to test the feature.

**0.4.1 – StudyPlanner-TaskList**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test** | **Describe the feature being tested** | **Describe the user input or test data** | **Describe the pass criteria** |  |
| 2.0 | Adding Task | 1. User inputs something into “Add Task Bar” 2. User inputs something or leave blank in “Due Date” selector 3. User press Add Task | 1. User should see a task card with the task entered previous and the date (if user entered one) in a white box 2. User sees a delete button in the white box | Tester name:   |  |  | | --- | --- | |  | PASS | |  | FAIL |   Observations: |
| 2.1 | Deleting Task | 1. User press Delete Task Button | 1. User should see task card disappear leaving other task cards on screen | Tester name:   |  |  | | --- | --- | |  | PASS | |  | FAIL |   Observations: |
| 2.2 | Loading Previous Tasks | 1. User inputs something into “Add Task Bar” 2. User inputs something or leave blank in “Due Date” selector 3. User press Add Task 4. User reloads window | 1. User should see previous task card remaining on the page | Tester name:   |  |  | | --- | --- | |  | PASS | |  | FAIL |   Observations: |

*Example*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Test*** | ***Describe the feature being tested*** | ***Describe the user input or test data*** | ***Describe the pass criteria*** |  |
| *1.1* | *Video plays in home page footer* | 1. *User starts at index.html* 2. *User scrolls down to bottom of page* 3. *User clicks play bottom on video in purple footer* | 1. *User see homepage* 2. *User see’s video thumbnail displayed in footer with arrow playbutton* 3. *User clicks play button* 4. *User see’s video playing and hears sound* | *Tester name:*   |  |  | | --- | --- | |  | *PASS* | |  | *FAIL* |   *Observations:* |
| *1.2* | *Addition calculator works* | 1. *User types addme.py into bash shell* 2. *Enter data as follows:*  |  |  | | --- | --- | | *1* | *2* | | *0* | *5* | | *-4* | *-4* | | *-2* | *4* | | *Program outputs as follows:*   |  |  |  | | --- | --- | --- | | *1* | *2* | ***3*** | | *0* | *5* | ***5*** | | *-4* | *-4* | ***-8*** | | *-2* | *4* | ***2*** | | *Tester name:*   |  |  | | --- | --- | |  | *PASS* | |  | *FAIL* |   *Observations:* |
| *1.3* | *Robot stops before collision* | 1. *Place robot on ground* 2. *Place a box 1000mm in front of robot, ensure robot is in line with the object and it will collide* 3. *Turn robot on* | 1. *Robot should accelerate to full speed* 2. *When robot is 300mm from object robot should begin deaccelerating.* 3. *When robot is 50mm from object robot should coem to a complete standstill and the red LED start flashing* | *Tester name:*   |  |  | | --- | --- | |  | *PASS* | |  | *FAIL* |   *Observations:* |