Andrew Tran

Scaffold

Table of Contents

[Section 1 - Identifying and Defining (15 Marks) 2](#_Toc183180441)

[1.1 — Client Scenario 2](#_Toc183180442)

[1.2 — Environmental Specification 2](#_Toc183180443)

[1.3 — List of Objectives 2](#_Toc183180444)

[1.4 — Performance Criteria 3](#_Toc183180445)

[1.5 — IPO (Input-Process-Output) 3](#_Toc183180446)

[1.6 — Gantt Chart 3](#_Toc183180447)

[Milestone 1: This should be completed by term 1, week 2. 3](#_Toc183180448)

[Section 2 - Research and Planning (10 Marks) 4](#_Toc183180449)

[2.1 — Research 4](#_Toc183180450)

[2.2 — Data Flow Diagram (DFD) 4](#_Toc183180451)

[2.3 — Interface Design (Wireframe) 4](#_Toc183180452)

[2.4 — Storyboard 4](#_Toc183180453)

[2.5 — Pseudocode 5](#_Toc183180454)

[Milestone 2: This should be completed by term 1, week 5. 5](#_Toc183180455)

[Section 3.1 - Development (10 Marks) 6](#_Toc183180456)

[3.1.2 — Logbook 6](#_Toc183180457)

[Milestone 3.1: This will be checked at various points each term. 6](#_Toc183180458)

[Section 4 - Testing and Maintaining (10 Marks) 7](#_Toc183180459)

[4.1 — Acceptance Testing 7](#_Toc183180460)

[4.2 — Load Testing 7](#_Toc183180461)

[4.3 — User Survey 7](#_Toc183180462)

# Section 1 - Identifying and Defining (15 Marks)

## 1.1 — Client Scenario

Define a clear scenario based on the client’s needs, including the application’s purpose, target audience, and primary goals. Detail the client's requirements and provide a narrative that encapsulates the problem that the application aims to solve. Explain the overall vision and the expected impact of the application.

|  |
| --- |
| **Overview**  Many teenagers and young adults (16-29) have reported a struggle in building and maintaining healthy habits for their future. This is due to a confusion on how to start their habits and a difficulty in exhibiting continual effort and hard work to reach their goals amongst external pressures like school exams and work deadlines.  A habit tracking application ‘TrackIt’ has been proposed to provide teenagers and young adults a tool that will easily create and track habits of their choice. TrackIt will provide a quick-use application that allows users to easily check in on their habit progress and identify what steps are next in cementing these behaviours into their daily lives. This application aims to solve the problems around starting up habits and provides a long-term solution for building them.  Through incentives and milestones users will be given constant motivation to continue in their habit while being able to view their progress and ultimately how close they are to reaching their end-goal. Although this application is designed for teens and young adults, due to their age group being the keenest on establishing healthy habits for the future, it is open to all users and is widely accessible for anyone willing to build a healthy habit. TrackIt’s main goals include simplifying the process of habit building, providing constant motivation to the user, fostering a sense of accountability within the individual, and prioritising the user’s privacy and security around their habits.  TrackIt will provide a virtual calendar that will track the user’s habits, allowing for a visual roadmap of habit progress. The user will first set up a habit which is an overall task that is logged in their personal profile. Examples of habits can be “read more”, “spend less time on my phone”, or things similar that are general outline of what they want to achieve. After this, the user will create ‘progress tasks’ which can be logged when the user makes progress on their habit. For example if the user has a “read more” habit, their progress tasks may include “read for 15min”, “read for 30min”, “read for 1 hour” and when they make progress they can log one of these tasks. Finally, the user can set up milestones which are based off their habits and progress tasks. The aforementioned example user may create a milestone “read everyday for a week” and if they log a progress task each month then their milestone will be completed.  **Advanced Features**  Additionally, TrackIt will be built with advanced features that will enhance the user’s experience and the app’s functionality. For example, TrackIt will include user personalisation that will allow the user to edit the UI of the app to be tailored to their preferences and create a more unique experience. Ideas of UI features that can be changed are system colours, profile pictures, bios, usernames, etc.  Another advanced feature of TrackIt is the Chatbot feature that allows the user to send any FAQs to an AI chatbot which will generate an appropriate response. This feature allows the user to receive support quickly and minimises the time the user must spend learning the app’s functionality. It can also be used to recommend relevant habits/milestones/tasks based on the user’s past history of created habits.  TrackIt will also contain gamification aspects as an advanced feature which will reward the user in-app points based on their consistency in completing tasks, hitting milestones and making progress on their habits. The user will then be able to save up points to spend on customisation features and different items. Another component of TrackIt’s gamification is a streak counter which will create a streak for the user if they complete tasks or milestones each day. This creates an incentive to log on each day and continue contributing to their habits. The streak will be reset if the user misses a day, encouraging them to make continual progress. |

## 1.2 — Environmental Specification

Outline the technical environment in which the application will operate. This includes hardware, software, network specifications, and other relevant environmental factors. Describe the operating systems, development tools, servers, and other infrastructure components necessary for the application.

**Hardware**

|  |  |  |
| --- | --- | --- |
| **Image** | **Part** | **Specification** |
| Crucial RAM 16GB DDR4 2666 MHz CL19 Desktop Memory CT16G4DFRA266 :  Amazon.com.au: Computers | RAM for immediate storage and retrieval of user data throughout app’s use | 16.0 GB |
| AMD Ryzen 7 7840U Processor - Benchmarks and Specs - NotebookCheck.net Tech | AMD Ryzen 7 7840U for sufficient processing and operation of app | 3.30 GHz |
| Vansuny 512GB SATA III SSD Internal Solid State Drive 2.5” Internal Drive  Advanced 3D NAND Flash Up to 500MB/s SSD Hard Drive for PC Laptop : ... | Storage (HDD and SSD) for long-term storage of user data and permanent saving | 475 GB |
| AMD Radeon 780M Specs | TechPowerUp GPU Database | Radeon 780M Graphics for ideal app quality and best visual experience | 3.30 GHz |
| Framework Laptop 13 Pre-Built Quick Start Guide - Framework Guides | Power Supply for sufficient powering of application and to have the best battery usage | Framework Power Adapter 60W |
| How to replace a broken laptop screen | PCWorld | Monitor for visual display of application including UI and graphics | 2256 x 1504px, 3:2 Display |
| Mainboard Replacement Guide - Framework Guides | Motherboard to connect all hardware components and receive best operating experience | Framework FRANMDCP07 |
| Razer Pro Click Optical Wireless Ergonomic Mouse (RZ01-02990100) -  Umart.com.au | Mouse for ease of user experience and navigation / completing tasks (optional) | Razer Pro Click Wireless Mouse (16,000 dpi) / (however any model is acceptable) |

**Software**

|  |  |  |
| --- | --- | --- |
| **Image** | **Part** | **Specification** |
| Microsoft unveils Windows 11, a new user experience that brings you closer  to what you love – Microsoft Malaysia News Center | Windows 11 for ideal operating experience | 64-bit operating system |
| Python (programming language) - Wikipedia | Programming language for server-side programming | Python (version 3.13.2) |
|  | Backend framework to load and operate the development version of the application | Node.js (version 22) |
| Main Features of React That Developers Must Know | Medium | Frontend framework to load and operate the development version of the application | React.js (version 18.3.1) |
| VS Code Blog - Microsoft for Developers | Development Tool to run the application and export | VS Code (version 1.96) |

**Network**

|  |  |  |
| --- | --- | --- |
| **Image** | **Part** | **Specification** |
|  | Internet Speed | 327.8 Mbps upload, 61.4 Mbps download, WiFi |
| Largest Contentful Paint (LCP): Google's metric for revolutionizing ranking  | iSmartFrame | Largest Contentful Paint (LCP) | 2.94s |
| Cumulative Layout Shift – How to Measure and Fix CLS | Cumulative Layout Shift (CLS) | 0.10 |
|  | Router for facilitating internet access | Telstra Smart Modem Gen 2 NBN |
| Cable Matters 10Gbps Snagless Long Cat 6 Ethernet Cable 100 ft (Cat 6  Cable, Cat6 Cable, Internet Cable, Network Cable) in Black : Amazon.com.au:  ... | Fibre optic internet cables | 4 GHZ Bandwidth, 2km length, 850 nm wavelength |

## 1.3 — List of Objectives

Enumerate the specific objectives that the application aims to achieve. These should be clear, measurable goals that directly address the client’s needs. The objectives could include functionality, user engagement, performance targets, and other relevant criteria.

**Functional Features**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Priority** | **Description** |
| User can register with an account | 1 | An account can be created for a new user, securely saving their email and password, and saving all their future activity to this account |
| User can create a habit | 2 | The system can store a habit for the user account that will appear in the user’s dashboard including the title and ideal duration |
| User can make milestones | 3 | Custom milestones can be set by the user which appear on their personal calendar within the app |
| User can set up progress tasks | 4 | User can set up tasks for themselves that will indicate progress |
| User can edit habits/milestones/tasks | 5 | All habits, milestones, and tasks created by the user can have their details edited |
| User can complete progress tasks | 6 | The user can record that they did one of their progress tasks that day and have this marked on their personal calendar |
| User can achieve milestones | 7 | The user can check-off a milestone and have the app render it completed |
| User can delete habits/milestones/tasks | 8 | Any habit, milestone, or task created by the user can be deleted from their calendar and dashboard |
| User can start a streak | 9 | When the user successively records progress or achieves milestones, the app will record these as a streak |
| User’s streak can be lost | 10 | If a streak is interrupted by a day of non-progress the streak is reset back to zero |
| User can personalise their profile | 11 | The user has a profile page which can be customised with profile pictures, interests, and profile colours. |

**Non-Functional Features**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Priority** | **Description** |
| User can create an account within five minutes of clicking register | 1 | The account creation should be seamless and very quick as minimal data is collected, and load times should be quick |
| Habit/milestone/progress task creation is effective | 2 | User should have little-to-no difficulty creating habits, milestones, and progress tasks for their account measured by user feedback surveys |
| Application can support 100+ users concurrently | 3 | The application should have no trouble fulfilling the requests of 100+ users simultaneously with no delays |
| Fast API response times | 4 | API responses should be no more than 500ms for any query |
| End-to-end encryption | 5 | All user data should be encrypted in storage (SHA-256) and during transmission (HTTPS) |
| AI recommended habits/milestones/tasks are relevant | 6 | In-app AI recommends habits based on the user’s history to be relevant. For example, if a user creates habits around reading it may recommend a task “read for 30 minutes a day” |
| Application can be opened from desktop in a fast manner | 7 | Application opens within 2 seconds of the icon being clicked. |
| Deleted habits are backed up | 8 | Any habit that is deleted will be stored in a recycle bin for 30 days before being permanently deleted |
| Layout and design are visually appealing | 9 | 90% of users report the layout of the application being positive in user feedback surveys |
| Application doesn’t take large amounts of battery | 10 | Application will take up less than 5% of the user’s battery after an hour of use. |
| Accessible and thoughtful accessibility options | 11 | Users report no problems or suggestions with the application’s current set of accessibility options |
| Calendar updates quickly in real time | 12 | When a habit is created or deleted by the user, it is instantaneously added or removed from the calendar |

## 1.4 — Performance Criteria

Specify the application's performance criteria, including speed, reliability, scalability, and user satisfaction benchmarks. Define how performance will be measured and the application's standards.

|  |  |  |  |
| --- | --- | --- | --- |
| **Category** | **Criteria** | **Measurement Tool** | **Application Standard** |
| Speed | Page Response Times | Chrome DevTools | Largest Contentful Paint is less than 3 seconds |
| Speed | API Response Times | Stopwatch | Requests should be filled within 1 second |
| Speed | Page Refresh Times | Stopwatch | Page can refresh all content within 1 second |
| Reliability | Logs are saved and backed up | Manual Checks | All user data saves over uses and remains unchanged after closing and reopening the application |
| Reliability | Webpages are encrypted | Chrome DevTools | Webpages are protected with HTTPS and authenticated with TLS 1.3 |
| Reliability | Notification accuracy | Manual Checks | Notifications are received within 10 seconds of the desired time |
| Scalability | Concurrent User Handling | Manually open multiple accounts in different tabs | Multiple accounts can use the app simultaneously with no detected lag or buffering |
| Scalability | Potential for User Growth | Manual checks | Account creation system does not limit the number of potential accounts to a finite number |
| Scalability | Page Resizing | Manual Check | The webpage can be resized with no buffering or lag |
| User Satisfaction | General Likeability | User Surveys | At least 90% of users report a positive experience using the app |
| User Satisfaction | Customisation from Files | Manual Check | User can upload files (pfp, backgrounds, etc) to the app and have them load within 4 seconds |
| User Satisfaction | Ease of Habit Creation | Stopwatch | All user data (habits, tasks, milestones) can be saved to their account within 2 seconds of publishing |

## 1.5 — IPO (Input-Process-Output)

Develop an IPO model that outlines the application's inputs, processes, and outputs.

|  |  |  |
| --- | --- | --- |
| **Input** | **Process** | **Output** |
| Email and Password | Send user details to the server and creates a new user account to save all future progress | User Dashboard / Home Page |
| Entered habit data including title, description, category | Create new habit on user’s profile and add this to their personal calendar | Updated calendar and user profile |
| Entered milestone data including title, description, progress | Create a milestone for the user, save it to their profile and attach this to the respective habit. Update their calendar to show a roadmap of their current milestones. | Updated calendar and display of milestone roadmap |
| Entered progress task data including title, description, amount | Create a progress task for the user, save it to their profile, and attach this to the respective habit. Allow the user to log progress tasks with the custom details they’ve entered | Updated user profile and progress task menu |
| New/edited habit, milestone, task data | Edit the details for any habit/milestone/task that has already been created on their profile and have the details updated upon saving. | Updated habit, milestone, task data |
| User’s chosen notification preference (daily, weekly, custom) | Update the user’s notification preferences and save this preference to their profile until changed. Send them a confirmation that alerts them of their change. | Confirmation alert and appropriate notifications |
| Selected UI colour choice | Allow the user to select their colour choice for UI colours and save this choice to their profile. Display their colour choice every time the user opens the app. | Updated application using selected UI colours |
| Entered text for chatbot | Send the user’s text to the AI chatbot, process their request, and generate an appropriate response message. Allow the AI creation permissions if authorised by the user. | Generative response and created habits allowed by the user |

## 1.6 — Gantt Chart

Create a Gantt chart to outline the project timeline. The chart should include all major milestones, tasks, and deadlines. It should also visually represent the project schedule, showing each task's start and end dates and the dependencies between tasks.

|  |
| --- |
|  |

### Milestone 1: This should be completed by term 1, week 2.

# Section 2 - Research and Planning (10 Marks)

## 2.1 — Research

Conduct thorough research to gather information relevant to the project. This should include market analysis, user research, competitor analysis, and other pertinent data. Summarise the findings and explain how they will inform the project’s development.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Section 1: Similar Applications / Competitor Analysis**   1. Table found from <https://zapier.com/blog/best-habit-tracker-app/>     To best compete with these competitors, the application will to achieve the following (using the previous table’s layout):   |  |  |  |  | | --- | --- | --- | --- | |  | Best for | Standout Feature | Pricing | | TrackIt | Creating a consistent habit schedule into your future (and desktop uses) | Personalised calendar tracker + easy logging system | Free |  1. Youtube video recommending the best habit tracking apps <https://www.youtube.com/watch?v=dlysHPeH3Ck> (8min)   Key Takeaways:   * User interface must be intuitive and easy to follow * Accessibility is a big deal-breaker * Community features can motivate people to use the app more * Visualisation of growth is helpful (plant that grows as progress is made or another idea like that)  1. Summary of Competitor Analysis:   By researching other habit tracking apps, the development team has been notified of the importance of having an effective user interface for TrackIt. This is because the readability of the interface and the layout of visuals ultimately decides the user’s first impressions of the app. During the development process, user interface is a top priority along with functionality to ensure the app can sufficiently impact the market and compete with other applications. Additionally, other features like visualisations of progress, quick loading times, and others have been noted by the development team.  **Section 2: User Research**   1. Link to subreddit (online forum) where users discuss which habits they most commonly track <https://www.reddit.com/r/bulletjournal/comments/17ef29l/what_habits_do_you_track/>   Top Five Habits brought up:   1. Exercise and Movement (15%) 2. Water and Hydration (14%) 3. Teeth Brushing and Dental Care (12%) 4. Sleep (9%) 5. Reading and Hobbies (8%) 6. Other (42%)   When developing the application, I will be mindful of appealing towards these habits especially and clearly communicating to users that these categories of habits can be tracked with the app.   1. Link to article comparing habit tracking apps <https://clickup.com/blog/best-habit-tracker-app/>   Frequent advantages of habit tracking apps mentioned:   * Visual progress tracking * Note taking abilities / journaling features * In-app rewards / in-game store * Specific reminder features * Allows tracking negative habits to eliminate them * Exporting capabilities for habit data * Accountability partner system   Frequent limitations of habit tracking apps mentioned:   * Overwhelming UI * Small library of habit suggestions * Limited number of habits that can be set up on one account * Generic interface / uninspired visuals  1. Summary of User Research   Through researching the commonly habits tracked on habit tracking apps and their most commonly listed advantages and disadvantages, the research team has identified a popular pool of users who download these apps to track certain habit categories. During development, the team will design the application’s functionality to appeal to these categories. This way the application is more likely to be used and meet the needs of users. Similarly, the development team will incorporate the effectiveness of the user interface in their design process, prioritising this along with functionality and ensuring the design is engaging and easy to interpret. Furthermore, new ideas for features included in the app have been alerted to the development team including journaling features, negative habit tracking for elimination, accountability partner systems, etc.  **Section 3: Behavioural Psychology**   1. Link to research paper on habit tracking through app-based mediums <https://pmc.ncbi.nlm.nih.gov/articles/PMC7020232/>   The paper states that using an app to build habits is effective as habit repetition leads to increased autonomous behaviour. This is helpful in understanding the psychology of building habits as the application can be designed with this psychology to maximise functionality and effectiveness. Similarly, the paper also states that building habits reduces motivational conflicts and improves mood during completion of that habit. In the same way, this can be implemented into the design to incorporate this psychology. For example, the visualisation of growth could become happier over time to reflect the improve in mood the user will experience while completing their habit.  **Section 4: Hardware and Software**   1. Minimum Windows Server Hardware Requirements <https://learn.microsoft.com/en-us/windows-server/get-started/hardware-requirements?tabs=cpu&pivots=windows-server-2025>  * 1.4 GHz 64-bit Processor * 4GB RAM * 32GB Disk Space * Ethernet Adapter 1GB/s   The minimum requirements for running a windows server provides a rough idea on the requirements for running a web application on a desktop device. The environmental specifications listed above meets the requirements for a windows server, indicating that a basic desktop application should run efficiently on the desktop used for development.   1. Minimum Software Requirements for VS Code <https://code.visualstudio.com/docs/supporting/requirements>  * Windows 10 (64-bit)   The minimum software requirements to run VS Code would provide general insight into what software requirements must be achieved to run a basic web application. The development desktop meets the requirement of using a Windows 10 or later system, enabling the proficient running a web application on the desktop used for development.  **Section 5: Implementation Methods**   1. How To Build a Habit Tracker in VS Code Tutorial (HTML, CSS, JavaScript) <https://www.youtube.com/watch?v=-OcVVgDLI3U>   This youtube tutorial can provide an initial kickstart to the project, including ideas on how to create a calendar structure, styling, and updating the calendar. Although this will not be used to create the full calendar, it provides a helpful start for coding from scratch.   1. How To Build a Calendar in VS Code Tutorial (HTML, CSS, JavaScript) <https://www.youtube.com/watch?v=Z1BGAivZRlE>   This youtube tutorial creates a dynamic calendar that is aesthetically formatted and updates in real-time, dynamically changing the amount of days displayed to correspond with the appropriate month. This will provide the building blocks for programming the calendar into TrackIt.  Extra resources for building the app:   * Coding a nav bar for a webpage: <https://www.youtube.com/watch?v=f3uCSh6LIY0> * Drop down menu tutorial: <https://www.youtube.com/watch?v=dmFC1e_CUAQ>   \*hardware and software research  \*environment (network requirements, certains speeds, OS, etc)  \*implementation methods for new features (guide, tutorial, open source software, plugin) |

## 2.2 — Data Flow Diagram (DFD)

Design a DFD for each feature of your project to illustrate how data will flow through the system. The DFD should show the interactions between inputs, processes, and outputs. Provide detailed diagrams and explanations for each feature.

A diagram of a personalization

AI-generated content may be incorrect.

TrackIt’s personalisation feature creates a unique personal experience for the user by looking at their past history of logs/habits and their account history to spontaneously create new content. The first process *Recommend Habit* allows the user to ask the application what habits they should consider tracking. The app will then check their history of past logs and habits to find the most ideal habit to potentially take up. Similarly, *Suggest Friends* matches different accounts that are related with their history of habits and logs. Accounts with similar habits are recommended to each other. *Update Display Info* allows the user to upload files off their computer to customise their public profile. This helps personalise accounts and enables users to express their other interests and hobbies outside of habits. Finally, the process *Update UI Colours* takes the user’s chosen UI colour preference and updates their application to display the a new interface incorporating those colours. The app will be built with a default colour in the UI and this can be changed by user request.

A diagram of a chatbot

AI-generated content may be incorrect.

The chatbot feature in TrackIt provides an AI model to the user which can respond to any textual input from the user. The model will include information on habit building and the application, assisting users in creating habits or navigating the app. The process *Generate chabot response* is the user’s main way of accessing the chatbot, allowing them to ask any question or input a textual statement. This will then be sent to the AI model to generate the most appropriate response to the user’s request. Additionally, the user is allowed to save multiple conversations with the chatbot, enabling them to revisit previous conversations to remind themselves of the response they received. This is depicted through the *Create new conversation* process. Furthermore, the user can search through their history of conversations to help them navigate a build up of different chatbot conversations. This works by inputting a key word, e.g. “reading”, and scanning the user’s cache for conversations that include the word “reading”. These conversations are then returned to the user, as shown in the *Search Conversations* process.

A diagram of a computer program

AI-generated content may be incorrect.

The gamification of TrackIt incentivises users to create more habits and log more frequently to receive in-game points which can be spent on virtual items and cosmetics. After logging a task, the user is rewarded with a corresponding point value that adds to their total. Lengthier logs will be awarded more points, encouraging the user to pursue their habits more intentionally. The in-game shop will offer various cosmetics that can change the aesthetic of the user’s application or their profile. For example, a background pattern or profile picture border. Additionally, the user can bookmark items they wish to purchase in their future. The system will place a reminder on their dashboard of how many points they must receive before they can afford to purchase that item, as shown in *Bookmark Items*. Moreover, TrackIt also includes a streak counter that incentivises the user to log daily to build up their streak counter and receive a great in-game points multiplier to all points earned.

## 2.3 — Interface Design (Wireframe)

Develop wireframes for the application's user interface. These wireframes should visually represent the application's layout and design. Include annotations and explanations for each element of the interface.

|  |
| --- |
| The community page for TrackIt features a grid of user profiles which are recommended to the logged in user. These profiles are sorted by number of habits in common and the grid is vertically scrollable to load more new profiles. By clicking on a profile the user is took to their profile page.    Profile pages in TrackIt contain the user’s current tracked habits, their best streak attained throughout the account’s history, the streak they are currently holding, a bio written by that user, and a banner for the app’s social component that, upon being clicked, redirects the user back to the community page. There is also a statistic that displays that profile’s current in-game points located below their display name and profile picture which are chosen by the user.    The dashboard page features the user’s personal calendar where they track their habits and milestones. Below it is a list of their current habits they’re tracking, colour coded to match the category they’ve inputted them into. There is also a preview of the in-game shop which the user is redirected to upon clicking.    The in-game shop displays cosmetics available for purchase, sorted by the user’s bookmarked items at the top and featured / popular community items below it. There is also a search bar where the user can search through all available items to find a specific item they are looking for.    The help page features a list of the admin for TrackIt and their details if the user wishes to contact them for help in the app. Below it, there is a system manual which is free for all users to read for an in depth explanation of how to fix common problems and how to configure the application, such as accessibility options, etc. The user can also enter their email to enter the app’s news letter to receive weekly news about the app and habit tracking.    The expanded view of the personal calendar enlarges the calendar for the user, allowing better visibility. Above it there is a list of the habits they are currently tracking and below it there is the option to create a new habit. By clicking this button the user is redirected to the create new habit page.    When creating a new habit, this layout will appear in-front of the calendar view with the background faded to increase visibility. The user is given two input fields where they can enter the title for their habit (“read more”), the name of the progress task (“read 15min”, “read 30min”) and the value of the progress task that adds onto their calendar (“15”, “30”). There is also a list of the user’s current habits which can be dragged to reorder them in priority and options to edit pre-existing habits. |

## 2.4 — Storyboard

Create a storyboard to outline the user journey through the application. The storyboard should include key screens, user actions, and interactions. Provide a narrative to explain how users will navigate the application and achieve their goals.

|  |
| --- |
| **Narrative**  Upon opening the application, the user is shown the **dashboard**. From the dashboard they click on their **personal calendar** to enlarge the view. On their calendar they decide to **create a new habit** for themselves and click the button and enter the appropriate details. They decide to return to the dashboard by using the header at the top of every screen. On the dashboard they click on their **profile** to view all their current stats. Once satisfied with their profile, they decide they want to find some friends on the app and go to the **social** page. On the social page they send friend requests to accounts they’ve chosen and open up their individual **profile** page to view their best streaks all-time. After this they decide to view the **in-game shop** by using the dropdown menu on page headers to be redirected straight to the page. After buying a few in-game cosmetics they decide the app would be better for them if they toggled colour-blind options to *on*. They click on the **help** button in the bottom left of every page and are redirected to the help page where they are given all the specific instructions to enabling this setting. After this they finish up for the day and close down the application. |

## 2.5 — Pseudocode

Write five pseudocode algorithms for critical functions and algorithms in the web application's backend. These pseudocode algorithms should provide a clear foundation for coding the application. Include detailed explanations and comments for each algorithm.

|  |
| --- |
| Critical Functions:  1. Create habit/milestone/task  2. Create new user  3. Community Screen  4. In-game shop  5. Log task  1. Create habit:  BEGIN New Habit  GET user  LET title = INPUT String  IF title = None THEN  DISPLAY "Title cannot be nothing"  ENDIF  LET description = INPUT String  LET category = INPUT FROM List of categories  LET selection = INPUT FROM user's mouse  IF selection IS Create Task Button THEN  Create Task  ELSE IF selection = Publish Button THEN  Commit habit to user account and calendar  DISPLAY "Habit created successfully!"  ENDIF  END New Habit  To create a new habit for the user, the function starts by getting the user's account to retrieve all their information and data. The user then inputs the title and description of their habit. The habit title cannot be empty. They then pick a category to place their habit into. The create task button will create sub-tasks for the habit which runs another subfunction. When the user clicks on publish the habit is added to their account and an appropriate message is displayed.  2. Create new user:  BEGIN Sign Up  Get Special Characters  LET username = INPUT String  IF len(username) < 3 THEN  Clear username  DISPLAY "Username must be longer than three characters"  ELSEIF username = None THEN  DISPLAY "Username cannot be nothing"  ENDIF  LET password = INPUT String  IF len(password) < 7 THEN  Clear password  DISPLAY "Password must be at least 8 characters long"  ELSEIF password DOES NOT CONTAIN Special Character THEN  Clear password  DISPLAY "Password must contain a special character" + special characters  ELSEIF password = None THEN  DISPLAY "Password cannot be nothing"  ENDIF  LET selection = INPUT FROM user's mouse  IF selection IS Sign Up Button THEN  commit user session to database  END Sign Up  In the sign up process, the algorithm begins by defining a list of special characters that will later be used for validation. The user can then input a username they want to identify their account with. The algorithm will validate this username, ensuring it's longer than three characters and not empty. The same process happens for the password however, the password must be at least eight characters long and contain a special character, (taken from the predefined list of special characters). After these two fields are filled out properly, the algorithm will wait for the user to click on the sign up button to confirm their details and send these details to the database. The user's account is then saved for future login to the system.  3. Community Page:  BEGIN Social  GET user  GET accounts  FOR i=0 TO 30  IF account.habit IN user.habits THEN  DISPLAY account  STEP 1  ENDIF  ENDFOR  LET selection = INPUT FROM user's mouse  IF selection IS account THEN  Redirect to account profile page  ELSEIF selection IS friend\_request THEN  Send friend request to account's inbox  ENDIF  END Social  The application's community page contains an algorithm which determines which accounts to suggest. It begins by defining the user's account and accessing the database of all other user accounts. It then retrieves 30 accounts randomly out of the entire user database. If the retrieved accounts are tracking habits that are tracked by the user then they are displayed on the screen in a grid. This repeats until 30 suitable accounts are found that match the user's habits. Once all accounts are displayed, if the user clicks on an account that is suggested they will have the option to send a friend request to that account. The friend request will be displayed in the other user's inbox.  4. In-game shop  BEGIN Shop  GET user  GET items  FOR i=0 TO len(AvailableItems Dictionary) STEP 1  IF item status IS "featured" THEN  DISPLAY item  ELSEIF item status IS "bookmarked" THEN  DISPLAY item  ENDIF  LET selection = INPUT FROM user's mouse  IF item IS selected THEN  IF user.points >= item.value THEN  Add item to user's inventory  user.points =- item.value  ELSEIF user.points < item.value THEN  DISPLAY "You do not have enough points to buy this item"  ENDIF  ENDIF  DISPLAY search bar  LET prompt = INPUT String  IF prompt != NONE THEN  DISPLAY items that match the prompt  ENDIF  END Shop  The in-game shop algorithm starts by getting the user’s items. It then uses a FOR loop to check if an item is bookmarked or featured. If so, these items are displayed in the shop. The user’s mouse input is tracked and if they click to purchase an item then the appropriate number of points is subtracted from their account. If they do not have enough points to purchase the item, the system will display the relevant message. Depending on what prompt the user puts in the search bar, items that match the prompt will be displayed in the shop.  5. Log Task  BEGIN Log Task  GET user  finish = len(user.tasks)  FOR i=0 TO finish STEP 1  DISPLAY task  NEXT LET task\_selection = INPUT task selection from user  IF task\_selection != None THEN  Update calendar (task\_selection.value)  ENDIF  IF milestone IS completed THEN  print "Milestone Reached! You've earnt " + task\_selection.value  user.points += task\_selection.value  ENDIF  END Log Task  To log a task, the app will get the user’s tasks and display them on the screen. If the user makes a selection to log the task then this log is made and the calendar is updated with the task value. If a milestone is reached by accumulating value through logging a task, then a message is displayed to the user that they’ve reached a milestone and they receive a certain number of points. |

### Milestone 2: This should be completed by term 1, week 5.

# Section 3.1 - Development (10 Marks)

## 3.1.2 — Logbook

Develop a logbook for tracking changes and implementing significant features and changes/additions to your web application. The logbook should follow the styling guide provided in the course specs. Include entries for each significant development activity, with dates, descriptions, and outcomes.

|  |
| --- |
| Located in repo/sharepoint |

### Milestone 3.1: This will be checked at various points each term.

# Section 4 - Testing and Maintaining (10 Marks)

## 4.1 — Acceptance Testing

Apply methodologies to test and evaluate whether the application meets all outlined requirements and objectives. Describe the acceptance testing process, including test cases, scenarios, and criteria. Document the results and any necessary actions.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| The purpose of this acceptance test is to run thorough testing on the developed app, ensuring all core features and aspects of the app run as intended. This includes abstracting the app into different areas, each with their own unique tests that focus on different criteria, revealing distinct insights into the app’s functionality. The number of detected failures in the acceptance testing will provide an ample gauge of the app’s readiness for deployment to the client.  To runt eh acceptance test, black box testing will be undertaken, testing the app without reviewing the system’s internal code. Each test will contain test cases to measure the amount of testing, scenarios that simulate real world usage, and criteria which determine whether a test is a success or failure.   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | Test Case # | Scenario | Steps Taken | Intended Result | Actual Result | Pass or Fail | Necessary actions | | 1 | Register new user account | Click sign up, enter user details | New account saved in database, redirected to dashboard | Worked as expected | Pass | None | | 2 | Log in with incorrect details | Enter incorrect details into login form, click login | App denies access, alerts user of incorrect details, redirects user back to login form | Error: Did not redirect back to login form | Fail | Add redirection to login form | | 3 | Create and log new habit | Click create habit, enter habit details, click save, log on calendar | New habit saved in database, log appears on calendar | Worked as expected | Pass | None | | 4 | Reset habit progress | Click habit button, click reset progress button | All logs for habit removed, XP set to zero | Worked as expected | Pass | None | | 5 | Edit habit info | Click habit, click edit, change details, click save | Habit updates saved and shown in habit list | Worked as expected | Pass | None | | 6 | Delete habit | Click habit, click delete, confirm action | Habit permanently removed from habit list | Worked as expected | Pass | None | | 7 | Sort and filter habits | User sort/filter options | Habit list changes order based on selected criteria | Worked as expected | Pass | None | | 8 | Create habit with invalid details | Click add habit, enter invalid details, click save | Invalid details corrected | Worked as expected | Pass | None | | 9 | Log habit in past / future days | Click habit calendar, click past date | Log saved to selected date | Worked as expected | Pass | None | | 10 | XP bar reflects current progress for category | Log multiple habits in same category | XP bar fills up and updates according to logs | Worked as expected | Pass | None | | 11 | Purchase item | Click shop, click purchase on item, confirm | Coins deducted, item added to inventory | Worked as expected | Pass | None | | 12 | Use effect item | Go to inventory, click use, | Item removed from inventory, effect applied (e.g. bonus XP), effect appears in active effects | Worked as expected | Pass | None | | 13 | Edit profile | Click profile, click edit, change user details, click save | Updated info saved to database, shown on profile page | Worked as expected | Pass | None | | 14 | Send friend requests | Search for user, click send request | Friend request status updated, sent to other user | Worked as expected | Pass | None | | 15 | Attempt to send duplicate friend requests | Send friend request to same user twice | Error shown or second request blocked | Second request blocked | Pass | None | | 16 | Admin account can delete user accounts | Login as admin, go to user list, click delete | Selected user account removed from database | Error thrown | Fail | Fix delete user account route | | 17 | Navigate app only using keyboard | Use tab, arrow keys, and enter to navigate all pages | All key features accessible without mouse | Worked as expected | Pass | None | | 18 | Input with XXS malicious script | Enter script tag (e.g. <script>) into form input | Script is sanitised or blocked, no malicious action triggered | Script is sanitised | Pass | None | | 19 | Create habit from recommended habits | Go to recommendations, click add on suggested habit | Habit details filled out in create habit form | Worked as expected | Pass | None | | 20 | Converse with AI chatbot | Open chat, ask question | AI responds appropriately and stays within scope of app | Worked as expected | Pass | None |     Summary of Results   |  |  |  | | --- | --- | --- | | Total Test Cases | Passed | Failed | | 20 | 18 | 2 |   Actions Taken  Two errors were detected in logging in with incorrect details and deleting user accounts from the admin dashboard. For the first problem, the user is denied access from the app and are notified of their incorrect details however, are not redirected back to the login form. In the login route, a redirection back to the login form will be added. The second problem, detected an issue with deleting user accounts as these are linked to other data models like habits, logs, etc. To solve this problem, cascading delete will be added to the user model, deleting all orphan when the user account is deleted.  Conclusion  From the acceptance testing, the app meets most requirements and will be ready for deployment to the client and real-world implementation once the two identified errors are fixed and solutions are implemented. |

## 4.2 — Load Testing

Apply methodologies to test and evaluate the application’s stability and performance under different levels of usage and load. Detail the load testing procedures, tools, and metrics. Provide a report on the findings and recommendations for improvement.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| The purpose of this load testing is to test the app under expected and rigorous load amounts, giving understanding towards the app’s stability, performance, and responsiveness in different cases. This will prevent the app from crashing during high traffic usage and will allow the application to better adapt to expected usage amounts and overestimates for additional precaution.  Load testing methodologies were applied by simulating concurrent users and automating HTTP requests and API calls simultaneously. Various cases including different numbers of concurrent users were conducted against the app, monitoring the application’s response to heavy traffic.  A locust test plan will be used to conduct the load test. The locust plan procedure is as follows:   1. Login 2. Create a habit 3. Log habit 4. Visit the shop 5. Purchase different items 6. Use and Item 7. Visit community page 8. Send friend request 9. Check inbox 10. Accept friend request   These 10 steps encompass the main functions of the app and should provide a detailed understanding of the app’s performance at different load levels. The levels that will be ran are as follows: Low load – 10 users, Normal load – 100 users, Heavy load – 500 users, Extreme load – 1000 users.  Results   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | Low Load 10 Users | | | | | | | | Type | Name | # Requests | # Fails | Average (ms) | Max (ms) | RPS | | GET | / | 251 | 0 | 9.25 | 51 | 0.55 | | POST | /buy\_item/1 | 81 | 0 | 9.46 | 32 | 0.18 | | POST | /buy\_item/2 | 92 | 0 | 8.75 | 19 | 0.2 | | POST | /buy\_item/3 | 76 | 0 | 9.43 | 23 | 0.17 | | GET | /community | 246 | 0 | 9.03 | 44 | 0.54 | | POST | /create\_habit | 250 | 0 | 10.56 | 36 | 0.55 | | GET | /inbox | 244 | 0 | 9.44 | 37 | 0.53 | | POST | /login | 10 | 0 | 12.99 | 24 | 0.02 | | POST | /respond\_friend\_request/100 | 244 | 0 | 10.12 | 37 | 0.53 | | POST | /send\_friend\_request/5 | 120 | 0 | 8.65 | 23 | 0.26 | | POST | /send\_friend\_request/6 | 125 | 0 | 9.72 | 34 | 0.27 | | GET | /shop | 249 | 0 | 8.93 | 36 | 0.55 | | POST | /use\_item/1 | 88 | 0 | 9.38 | 38 | 0.19 | | POST | /use\_item/2 | 73 | 0 | 9.52 | 25 | 0.16 | | POST | /use\_item/3 | 85 | 0 | 9.09 | 26 | 0.19 | |  | Aggregated | 2234 | 0 | 9.46 | 51 | 4.89 |   Analysis:   * No failures on any action * All routes had average response times below 13ms * Max response time was 51ms   From the low load test, all routes are performing exceptionally well and have fast response times. Load testing will continue into the normal load test to find limits.   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | Normal Load 100 Users | | | | | | | | Type | Name | # Requests | # Fails | Average (ms) | Max (ms) | RPS | | GET | / | 757 | 0 | 15.14 | 70 | 5.68 | | POST | /buy\_item/1 | 226 | 0 | 11.87 | 48 | 1.7 | | POST | /buy\_item/2 | 258 | 0 | 11.61 | 58 | 1.94 | | POST | /buy\_item/3 | 243 | 0 | 11.42 | 72 | 1.82 | | GET | /community | 702 | 0 | 11.36 | 57 | 5.27 | | POST | /create\_habit | 748 | 0 | 11.9 | 69 | 5.62 | | GET | /inbox | 686 | 0 | 11.55 | 61 | 5.15 | | POST | /login | 100 | 0 | 59.78 | 89 | 0.75 | | POST | /respond\_friend\_request/100 | 674 | 0 | 11.96 | 41 | 5.06 | | POST | /send\_friend\_request/5 | 339 | 0 | 11.68 | 62 | 2.55 | | POST | /send\_friend\_request/6 | 356 | 0 | 11.4 | 47 | 2.67 | | GET | /shop | 735 | 0 | 11.44 | 71 | 5.52 | | POST | /use\_item/1 | 248 | 0 | 11.71 | 73 | 1.86 | | POST | /use\_item/2 | 228 | 0 | 11.54 | 55 | 1.71 | | POST | /use\_item/3 | 234 | 0 | 11.59 | 66 | 1.76 | |  | Aggregated | 6534 | 0 | 12.77 | 89 | 49.06 |   Analysis:   * Still no failures from any request * Login recorded an exceptionally high average response time of 59.78ms * Yet its highest response time was 89ms   From the normal load test, the app continues to perform well under normal traffic however, the login route should be monitored. It is likely that either on the high or extreme load test that the login route may break down. Further testing is required.   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | High Load 500 Users | | | | | | | | Type | Name | # Requests | # Fails | Average (ms) | Max (ms) | RPS | | GET | / | 1819 | 173 | 577.83 | 11928 | 21.81 | | POST | /buy\_item/1 | 489 | 67 | 642.46 | 8695 | 5.86 | | POST | /buy\_item/2 | 558 | 56 | 626.18 | 8723 | 6.69 | | POST | /buy\_item/3 | 545 | 58 | 568.91 | 8703 | 6.53 | | GET | /community | 1485 | 210 | 629.58 | 10745 | 17.81 | | POST | /create\_habit | 1753 | 171 | 537.17 | 12170 | 21.02 | | GET | /inbox | 1433 | 234 | 750.89 | 11591 | 17.18 | | GET | /login | 127 | 127 | 4415.29 | 11927 | 1.52 | | POST | /login | 678 | 178 | 1146.73 | 11197 | 8.13 | | POST | /respond\_friend\_request/100 | 1376 | 199 | 698.55 | 8950 | 16.5 | | POST | /send\_friend\_request/5 | 743 | 117 | 675.09 | 8692 | 8.91 | | POST | /send\_friend\_request/6 | 723 | 122 | 677.67 | 11994 | 8.67 | | GET | /shop | 1680 | 190 | 578.12 | 11257 | 20.14 | | POST | /use\_item/1 | 493 | 69 | 603.07 | 8706 | 5.91 | | POST | /use\_item/2 | 507 | 76 | 569.27 | 8721 | 6.08 | | POST | /use\_item/3 | 532 | 69 | 632.83 | 8630 | 6.38 | |  | Aggregated | 14941 | 2116 | 680.15 | 12170 | 179.15 |   Analysis:   * High load test reported 2116 aggregated failures * All average response times were above 500ms * Login route still records the highest response times   From the high load test, the app is most suitable for low to normal traffic usages and should most likely be deployed in a low-normal user setting. The login page is the app’s weakest point where most failures are recorded. The app was holding up but around one minute started to record failures. Overall, the app could be used in high usage setting but for only short periods of time. After around one minute of high usage, the app fails, making it most suitable for low-normal usage.   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | Extreme Load 1000 users | | | | | | | | Type | Name | # Requests | # Fails | Average (ms) | Max (ms) | RPS | | GET | / | 2220 | 343 | 2143.34 | 29711 | 26.14 | | POST | /buy\_item/1 | 621 | 125 | 2448 | 35203 | 7.31 | | POST | /buy\_item/2 | 618 | 116 | 2416.9 | 25926 | 7.28 | | POST | /buy\_item/3 | 667 | 136 | 2160.94 | 26686 | 7.85 | | GET | /community | 1677 | 371 | 2243.68 | 35203 | 19.74 | | POST | /create\_habit | 2104 | 333 | 1891 | 26147 | 24.77 | | GET | /inbox | 1448 | 389 | 2289.92 | 26493 | 17.05 | | GET | /login | 49 | 49 | 22426.82 | 35287 | 0.58 | | POST | /login | 1068 | 68 | 2471.13 | 42121 | 12.57 | | POST | /respond\_friend\_request/100 | 1327 | 355 | 2122.21 | 26028 | 15.62 | | POST | /send\_friend\_request/5 | 772 | 174 | 2322.91 | 29534 | 9.09 | | POST | /send\_friend\_request/6 | 790 | 205 | 2479.83 | 29735 | 9.3 | | GET | /shop | 2014 | 365 | 2258.5 | 27509 | 23.71 | | POST | /use\_item/1 | 565 | 117 | 2147.94 | 25876 | 6.65 | | POST | /use\_item/2 | 638 | 133 | 2026.49 | 26093 | 7.51 | | POST | /use\_item/3 | 590 | 126 | 1984.37 | 26661 | 6.95 | |  | Aggregated | 17168 | 3405 | 2260.19 | 42121 | 202.13 |   Analysis:   * 3405 reported aggregated failures * All average response times are above 1800ms * Login route holds up as the route experiencing longest response times   From the extreme load test, the app’s performance didn’t change too drastically despite having the user amount doubled. It is possible to maybe deploy the app in high-extreme user settings however, this is not recommended as the app can only hold for around a minute before experiencing failures.  Recommendations  The load tests showed that the app is most comfortable in a low-normal user ranger. Once the user range reaches around 500 users, the app begins to fail. To improve the app’s performance on higher user numbers, the application should be deployed to a production server which contains load balancing. This would allow the app to function better in large user scale settings. Further, options around caching and asynchronous processing should be explored as this can improve app response times.  Conclusion  The app performs well under low-normal usage. It is recommended that the app, in its current state, should be deployed for smaller scale settings. Once the app receives 500 concurrent users, performance begins to degrade and response times increase significantly. The load test had spotlighted areas that could be improved for future scalability for the app. Adding solutions to the app will enable it to be scaled into higher user settings as the number of users on the app grows. |

## 4.3 — User Survey

Collect feedback through a survey to analyse and respond to feedback on the application’s usability, functionality, and overall satisfaction. Design a user survey with relevant questions and distribute it to a representative sample of users. Summarise the survey results and outline the changes made based on user feedback.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| The purpose of the user survey is to collect feedback from real users on the app’s functionality, usability, and overall experience. The results of the survey will help identify the app’s strongest and most satisfying areas as well as recorded what flaws and problems users encounter the most with the app.  Survey Design:  Form: Microsoft Forms  Survey Length: 7 Questions  Number of Responses: 3  Types of Questions:   * Rating (out of 5 stars) for general understanding of usability * Ranking to discover common favourites and least favourites * Short Answer for detailed feedback   Survey Questions:   |  |  |  |  | | --- | --- | --- | --- | | Question # | Question | Type | Purpose | | 1 | How easy was it to create an account and start using the app? | Rating (out of 5) | Gain an understanding of how easy it is to initially get into the app | | 2 | How useful were the habit categories given? | Rating (out of 5) | Insight into the scope of categories and their flexibility as there is no ‘custom’ option | | 3 | How effective was the XP bar system in motivating you to log habits? | Rating (out of 5) | Gain insight into the main motivating feature of the app’s purpose (to build healthy habits) | | 4 | How appealing was the app’s visual interface and theme colours? | Rating (out of 5) | Record opinions on the app’s CSS design and cornflower blue theme colour | | 5 | Rank the app’s core features based on enjoyability | Ranking 6 options | Understand the favourite features and least favourite features | | 6 | Were any of the app’s features confusing or difficult to use? | Short answer | Detailed insight into most difficult or confusing aspects of the app for future development | | 7 | What improvements or new features would you recommend? | Short answer | Identify new ideas for the app that weren’t previously considered |   Link: <https://forms.cloud.microsoft/Pages/ResponsePage.aspx?id=U2-ogWHTs0qC_dmPrAJSDZNw068dhsZMi6MOyNmXi3ZUQ1hQVVFWRFY4TlhXSE9JWU9ESUdRUThWUy4u>  Summary of Results   |  |  | | --- | --- | | Question | Average Score / Common Responses | | Ease of getting started | 4/5 – App is generally easy to start | | Usefulness of categories given | 4.67/5 – Habit categories are well liked | | Effectiveness of XP bar for motivating | 4.5/5 – XP bar system is helpful | | Appeal of UI | 3.67/5 – App UI could be improved | | Rank core features | Profile customisation and levelling up were the most favoured features | | Which features were confusing | App needs instructions to start off | | Improvements | More pictures and visuals for all areas |   Feedback Analysis and Action Plan   |  |  | | --- | --- | | Common Feedback Point | Planned Action | | Instructions to use app are unclear | Create an instructions/tutorial page for the user to read so they can understand the app | | Shop and items are not enjoyable | Add more initial items to the shop and make earning coins easier | | Message that shop item cannot be purchased is unclear | Change error to flash for visual indicator of error in shop without redirecting user |   Further, from the feedback gathered, a list of additional features to add to the app have been recorded. However, these features are time consuming and may not be available before the app’s deployment:   * Recording of previous months * Graphs that chart habit progress   Conclusion  The app is generally useful and enjoyable for users however all survey respondents expressed a want for instructions to start the app. Further the visual interface could be improved with more visuals or pictures. Additionally, extra features could be added such as a recording of previous months or a visual chart that tracks progress. Overall, the app’s core features can be easily used by users to start tracking habit progress but instructions are required. |