Principles of Software Engineering Spring 2023  
*[FAU: CEN 4010]*

Task Hunter  
Gamified Task Manager

Group: 01

Ethan Fleming (Team Lead) – [eflemin6@fau.edu](mailto:eflemin6@fau.edu)  
 Meer Hossain – [hossainm2020@fau.edu](mailto:hossainm2020@fau.edu)  
 Marcus Watson – [watsonm2020@fau.edu](mailto:watsonm2020@fau.edu)  
 Brenden Martins – [bmartins2013@fau.edu](mailto:bmartins2013@fau.edu)  
 Brandon Rojas – [brojas2021@fau.edu](mailto:brojas2021@fau.edu)

Milestone 3  
03/24/2023  
  
Revision History

|  |  |
| --- | --- |
| Date | Comment |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

TABLE OF CONTENTS  
  
Executive Summary………………………………… 3

Data Definition………………………………… 4

Use Cases………………………………… 5

High-Level Functional Requirements………………………………… 11

List of Non-Functional Requirements………………………………… 15

High-Level System Architecture and Database Organization……… 18

High-Level UML Diagrams………………………………… 19

Competitive Analysis………………………………… 20

Team Roles………………………………… 22

Key Risks………………………………… 23

Executive Summary  
  
 We all have things we need to do, tasks that need to be taken care of. Many choose to simply keep a mental checklist of these tidbits of action. Yet, there is a substantial number of people for which that is not enough, and a written list of reminders is necessary. A written task list keeps these reminders clearly visible and helps to reduce the chance we’ll forget about them. However, finding the desire and resolve to work on these objectives can be a burden all its own. This is where a gamified task list comes in handy.  
  
 There is a wide array of general-purpose to-do lists and task managers out on the market. What they all fail to accomplish is giving the user a reason to accomplish their objectives. Humans are inherently lazy and love to procrastinate. In relation to the app, there’s no downside to pushing off today’s work until tomorrow and no benefit to completing it right now. There is a need for a program that incentives its users to accomplish their goals sooner rather than later, while not being so bulky and overcomplicated a new user is uninterested in starting. Task Hunter has been developed for exactly this niche.  
  
 The Task Hunter webapp was developed with the intent to give users a reason to accomplish tasks, without overcomplicating the process. Task Hunter provides what general-purpose task managers do not by creating a character profile that levels up as tasks are completed. Unlike existing gamified to-do lists, Task Hunter keeps the profile simple and focuses on the experience point and leveling system.   
  
 Task Hunter allows the user to generate custom tasks, for now or in the future. These tasks may be assigned several traits including name, date, difficulty, and notes. Created tasks may be found on the user’s bounty board, where they can see all active tasks or “bounties”. When the user completes their task they can mark the bounty as completed and will be rewarded with experience points based upon the difficulty of the bounty. These experience points are accumulated on the user’s profile and automatically level up the account if the next threshold is met.  
  
 What separates Task Hunter from its competitive gamified apps is the focus on simplification. It does not make the character profile overly complicated by bringing in too many aspects of games. Nor does it penalize the user for not completing their tasks in the same day. Something that is commonly seen in its competition. Task Hunter is a task manager, not a video game.  
  
Task Hunter – a task management webapp from your local adventurer’s guild.

Data Definitions

*Profile*: The profile of a user, including personal information, experience points, and level.

*Task*: An action or objective that needs to be completed or achieved.

*Task Traits*: Characteristics of a task that can be customized, such as name and difficulty.

*Difficulty*: The level of challenge or effort required to complete a task, which can be assigned a value in the app.

*Note(s)*: Text input within a task to provide extra information in regard to that task.

*Bounty*: A term used to refer to an active task or to-do item in Task Hunter.

*Bounty Board*: The section of the app where users can view all their active tasks or bounties.

*Completion*: The action of marking a task as finished or completed in the app.

*Experience Points*: Points awarded to users for completing tasks, which accumulate on their profile.

*Leveling System*: The mechanism by which a user's profile level increases as they accumulate more experience points.

*Adjustment*: The ability to modify a task after it has been created, including changing its name, date, difficulty, or notes.

*Adventurers Guild*: Fictional organization used for lore within Task Hunter. Supports hunters by providing the bounty board for their tasks.

*Competitive Gamified Apps*: Other task management apps with gamification features that Task Hunter aims to differentiate from.

*Home Page*: The initial page the user is shown upon logging in.

*Bounty Board Page*: Page that displays active tasks with summary of details on each.

*Calendar Page*: Page that displays a calendar (in weekly or monthly view) with markings for dates where a task is set to start.

*Profile Page*: Page that displays user’s profile and information pertaining to it, including name, experience points, and level.

Use Cases

**Use Case - Add Task**

The user comes to the page and wants to add a task to their bounty list. The user arrives at the Home Page and uses the add task button to enter the desired tasks and criteria. The system displays suggested tasks and customization options.

1. **Description:**
   1. The use case describes the process of how the user will utilize the “Add Task” feature of the system.
2. **Actors:**
   1. User
   2. System
3. **Pre-conditions:**
   1. User has an active internet connection
   2. System is available
   3. User has an active account
   4. User is logged into system
4. **Primary Flow of Events:**
   1. User logs into their account
   2. User arrives at the home page
   3. User arrives at the bounty board.
   4. User enters information about the task based on the system form: name, priority level(experience), due date, comments, reminders, etc.
   5. User confirms their input and presses the “Add Task” button.
   6. System stores and saves the added task to the UI and database.
   7. Terminate Use Case: Add task
5. **Alternate Flows:**
   1. User enters prohibited characters/format into form.
      1. Web site notifies user that the characters they used are prohibited
      2. Return to 4.3

**Use Case - Edit task**

The user visits the home page and wants to modify a previously made task. The user arrives at the task page and selects the desired task. Users can change any of the features of the task (ex: priority). The system will read the change and update the task as necessary.

1. **Description:**
   1. Use case describes the process of modifying a task from the bounty list.
2. **Actors:**
   1. User
   2. System
3. **Pre-conditions:**
   1. User has an active internet connection
   2. System is available
   3. User has an active account
   4. User is logged into system
   5. User has created a task
4. **Primary Flow of Events:**
   1. User logs into the system
   2. User arrives at the home page
   3. User visits the bounty board
   4. User attempts to click on the edit task button
   5. User changes a feature of task
   6. User confirms and presses the save button
   7. System saves and updates the task on the UI and database
   8. Terminate Use Case: Edit task
5. **Alternate Flows:**
   1. User enters prohibited characters/format into form.
      1. Web site notifies user that the characters they used are prohibited
      2. Return to 4.4

**Use Case - Completed task**

The user visits the webpage and wishes to quickly mark a specific task as completed. The user will arrive on the bounty board page and attempt to change the status of tasks. The system will ask the user to confirm their option and mark the task as completed.

1. **Description:**
   1. The use case demonstrates how the process of marking a task completed by the user. See Use Case: Add task to see the creation process.
2. **Actors:**
   1. User
   2. System
3. **Pre-conditions:**
   1. User has an active internet connection
   2. System is available
   3. User has an active account
   4. User is logged into system
   5. User has created a task
4. **Primary Flow of Events:**
   1. User logs into their account
   2. User arrives at the home page
   3. User arrives at the bounty board
   4. User attempts to click on the completion button
   5. User confirms that the task is completed
   6. System saves and updates the task on the UI and database
   7. Terminate Use Case: Completed task
5. **Alternate Flows:**
   1. User attempts to add tasks
      1. System refers to Use Case: Add task
   2. User attempts to edit task from bounty board
      1. System refers to Use Case: Edit task

**Use Case - View Bounty Board**

The user visits the application and wishes to view the bounty board. The user will arrive at the home page and view the bounty board page. The system displays the bounty board and shows the progress of the user as they complete the bounty tasks.

1. **Description:**
   1. The use case shows how the user can view the currently active tasks on the bounty board.
2. **Actors:**
   1. User
   2. System
3. **Pre-conditions:**
   1. User has an active internet connection
   2. System is available
   3. User has an active account
   4. User is logged into system
   5. User has created a task
4. **Primary Flow of Events:**
   1. User logs into their account
   2. User arrives at the home page
   3. User arrives at the bounty board
   4. User views all the available tasks on the board
   5. User may click the tasks to view more details
   6. Terminate User Case: View Bounty Board
5. **Alternate Flows:**
   1. User attempts to add tasks
      1. System refers to Use Case: Add task
   2. User attempts to edit task from bounty board
      1. System refers to Use Case: Edit task
   3. User attempts to mark task as completed
      1. System refers to Use Case: Completed task

**Use Case - View Calendar**

The user arrives at the webpage and wishes to view their tasks in a calendar format. The user will arrive at the calendar page. The system displays the calendars with their corresponding tasks and specifications.

1. **Description:**

The use case describes how the calendar feature can be viewed and used by the user. Refer to Use case: Add task for creation.

1. **Actors:**
   1. User
   2. System
2. **Pre-conditions:**
   1. User has an active internet connection
   2. System is available
   3. User has an active account
   4. User is logged into system
   5. User has created a task
3. **Primary Flow of Events:**
   1. User logs into their account
   2. User arrives at the home page
   3. User arrives at calendar page
   4. User views the calendar
   5. User clicks on a specific date or task to see more details
   6. Terminate Use Case: View Calendar
4. **Alternate Flows:**
   1. User attempts to add tasks
      1. System refers to Use Case: Add task
   2. User attempts to edit task from bounty board
      1. System refers to Use Case: Edit task
   3. User attempts to mark task as completed
      1. System refers to Use Case: Completed task

**Use Case - View Profile**

The user arrives at the webpage and wishes to view their personal profile. The user will arrive at the profile page. The system displays the profile of the user with their personal information and completed tasks.

1. **Description:**

The use case describes how the profile page can be viewed and used by the user.

1. **Actors:**
   1. User
   2. System
2. **Pre-conditions:**
   1. User has an active internet connection
   2. System is available
   3. User has an active account
   4. User is logged into system
3. **Primary Flow of Events:**
   1. User logs into their account
   2. User arrives at the home page
   3. User arrives at the profile page
   4. User views the profile
   5. User may click on completed tasks to see more details
   6. Terminate Use Case: View Profile
4. **Alternate Flows:**
   1. User is not logged in
      1. System requests user to enter credentials
      2. User enters credentials
      3. System validates credentials

**Use Case - View completed tasks**

The user wants to view their previously completed tasks. The user will arrive at the home page and visit either the profile or calendar pages. The system will display previously made tasks and on click, display the specifics of the tasks.

1. **Description:**
   1. The use case describes how the user can view their previous tasks in two different ways. Refer to Use Case: View Calendar and View Profile.
2. **Actors:**
   1. User
   2. System
3. **Pre-conditions:**
   1. User has an active internet connection
   2. System is available
   3. User has an active account
   4. User is logged into system
   5. User has created a task
   6. User has completed a task
4. **Primary Flow of Events:**
   1. User logs into their account
   2. User arrives at the home page
      1. User arrives at the profile page
         1. User clicks on completed tasks button
         2. User views all previously completed tasks
         3. User may click on individual tasks for more details
      2. User arrives at the calendar page
         1. User views completed tasks shown calendar view
         2. User may click on individual tasks for more details
   3. Terminate Use Case: View completed tasks
5. **Alternate Flows:**
   1. User attempts to add tasks
      1. System refers to Use Case: Add task
   2. User attempts to edit task from bounty board
      1. System refers to Use Case: Edit task
   3. User attempts to mark task as completed
      1. System refers to Use Case: Completed task

High-Level Functional Requirements

**Non-Member expectation**

1. **Creating Account**
   1. User will need to make a Username and a Password for the account. Username must be unique and not in the database already. Users must re-enter the password two times correctly when making the account. The system shall prevent the user from creating an account if the User’s chosen password does not match the re-enter password field.
   2. **Stimulus/Response Sequence**
      1. User enters a Username
      2. User enters a Password
      3. User re-enters Password for confirmation
      4. System shall check if Username is available
      5. System shall validate Password
      6. System shall confirm that the account was created to the User
   3. **Function requirement label**
      1. Requirement 1.1 Creating Account

**Members expectations**

1. **Edit Profile**
   1. User shall be able to edit their profile by providing a name. The System shall store name when the user clicks the save button. The System shall prevent any changes to the User’s profile if any of the fields are left blank. The user has to type the information in a valid format in order for the system to store the information.
   2. **Stimulus/Responsive Sequence**
      1. User will navigate to Profile Page
      2. User will click on edit profile
      3. User will input their name and tasks
      4. User shall click save
      5. System shall store their name
      6. System shall refresh to user profile with updated information as a confirmation
      7. System will have a button to redirect the user back to the bounty page at will
   3. **2.3 Function requirement label**

i. REQ 2.1 Edit Profile

1. **User viewing the Home page**
   1. User will navigate to the home page after logging in. The system provides three different options of pages to navigate from the home page.
   2. **Stimulus/Responsive Sequence**
      1. User will view the home page
      2. User will be able to navigate the profile page
      3. User will be able to navigate the bounty board page
      4. User will be able to navigate the calendar page
      5. System will show the corresponding pages

**3.3 Function requirement label**

i. REQ 3.1 View Home Page

1. **User viewing the Bounty Board**
   1. User will be able to view the available tasks on the bounty board. User may create a new task. User may mark tasks as completed or edit them from this page.
   2. **Stimulus/Responsive Sequence**
      1. User will view the Bounty Board page.
      2. User will be able to select to make a new task.
      3. User will be able to mark a task as complete.
      4. User will be able to select to edit an individual task.

**4.3 Function requirement label**

i. REQ 4.1 View Bounty Board

1. **User creating a Task**
   1. User will be able to create a task from the Bounty Board page. The user will be able to input task traits, including name, date, difficulty, and notes.
   2. **Stimulus/Responsive Sequence**
      1. User will be viewing the create task window.
      2. All information in the window will be set to blank or default.
      3. User may enter text for name of task.
      4. User may select date by clicking into date box and via a date selection pop-up.
      5. User may select difficulty from 3 available options.
      6. User may add notes in text format in the available notes textbox.
      7. System saves this information in the database.

**5.3 Function requirement label**

i. REQ 5.1 View Home Page

1. **User editing a Task**
   1. User will be able to edit a task by selecting an edit button on the task itself. This function will allow the user to adjust the traits of the task, including name, date, difficulty, and notes.
   2. **Stimulus/Responsive Sequence**
      1. User will be viewing the edit task window.
      2. User may adjust name by selecting name textbox and changing text value.
      3. User may adjust date by selecting date box and choosing a new date from date pop-up.
      4. User may adjust difficulty by selecting a non-selected difficulty option.
      5. User may adjust notes by selecting the notes textbox and changing the text values.

**6.3 Function requirement label**

i. REQ 6.1 Edit Task

1. **User completing a Task**
   1. User will be able to mark a task as completed from the Bounty Board page.
   2. **Stimulus/Responsive Sequence**
      1. User will view the Bounty Board page
      2. User will be able to select the complete button for a task
      3. System updates the task to completed
      4. User will be able to view completed task in Step 10

**7.3 Function requirement label**

i. REQ 7.1 Complete Task

1. **User viewing the profile page**
   1. User will be able to view the profile page by clicking on the profile picture icon in the top right corner of the screen. There will be relevant information to the tasks in progress, already done, and stats on those tasks as well.
   2. **Stimulus/Responsive Sequence**
      1. User will be viewing the Profile Page.
      2. User may edit profile by selecting the edit profile button.
      3. User will be able to see profile name and counts for total tasks completed, including a subset of counts for tasks completed for each difficulty.

**8.3 Function requirement label**

i. REQ 8.1 View Profile Page

1. **User viewing the calendar page**
   1. The user will arrive at the calendar page. The system displays the calendars with their corresponding tasks and specifications.
   2. **Stimulus/Responsive Sequence**
      1. User will be viewing Calendar Page
      2. Any dates with tasks assigned to start on that date will have a marking.
      3. User may select into any visible date to see tasks assigned to start on that date.
      4. User may alternate view to weekly or monthly.

**9.3 Function requirement label**

i. REQ 9.1 View Calendar Page

1. **User viewing a completed task**
   1. User will view the completed tasks in two different ways. Either through the profile page or calendar page.
   2. **Stimulus/Responsive Sequence**
      1. User will view the the home page
      2. User will view the profile page
      3. User will scroll down and press on the view completed tasks button.
      4. All completed tasks can be viewed in list format
      5. System will show the details for the tasks if clicked
      6. User may go back the home page to view through calendar view
      7. User will view the entire calendar
      8. User will scroll back to view previous tasks on each date
      9. System will show the details for the tasks if clicked

**10.3 Function requirement label** i. REQ 3.1 View Home Page

List of Non-Functional Requirements  
  
**Performance Requirements:** 1. *Responsiveness*: Responsiveness will be covered over various monitor sizes, refresh rates, etc. Ranging from mobile device screens to screens as big as 48’’ tv screens. Resolutions ranging from 1024 x 600 to 1900 x 1440.  
 2. *Cycle Time*: The cycle time at expected performance will be 1.0. With this in mind, the system will operate between 1.0 - 1.2 with a load of 10-20 concurrent users or a slight lag. The system will operate with a 1.21 - 1.30 with a load of 21-50 concurrent users or a moderate lag. The system will operate with a 1.31-1.50 with a load of 51-90 concurrent users or a heavy lag. Finally, the system will operate with a 1.51-1.70 with a load of 90-100 concurrent users or a very heavy lag. Any number of concurrent users over 50 will cause the system's performance to halt briefly until a user finishes.  
 3. *Speed Per Transaction*: Reaching for a speed per transaction between 20-70 milliseconds based on the cycle length. This will allow for the system to process 30-100 transactions an hour.  
 4. *Test Requirements*: Our test requirements for performance will include, testing the program with incorrect or invalid values to check the ability of the program to run into errors without incident and see how it handles errors, testing the limits of the program by using values at the extreme ends of the range ensuring the software can handle the expected range of values, and test the user-system interactions to ensure that the system is user-friendly and the UI is cohesive and appealing to the eye.   
 5. *Reliability*: Downtime due to system failures must be less than 1 hour in a total of 6 months. The system must be maintained for operation to be working 100% of the time for the first calendar year of its operation   
 6. *Maximum Bug Counts*:

* We strive to have little to no bugs but at most 3 bugs during testing and integration.
* No more than 2 bugs can be allowed after system delivery.

7. *Execution Speed*: On a 300mb internet connection, the main page should be loaded within 150 milliseconds. With a >100mb connection taking at most 500 milliseconds.

8. *Storage Utilization*: Storage utilization is of major importance to us. It should be within 30-50% at most of all storage to be easily used along-side other programs, when running in the background will go as low as 10% of storage. This gives us a large margin of storage in case of any emergency situations.   
 9: *Robustness*: We want to make sure our program is never down for more than 15 minutes due to failure. The percent of failure causing errors will be under 0.05%. The probability of losing data due to corruption must be under 0.3%.

**Ease of Use**:

1. *Training Time:* The app will be very straightforward and easy to use, thus training time should be little to none.

**Interoperability Requirements**:

1. *Browser Compatibility*: The browsers we will focus on are Google Chrome, as the main browser, and Safari as a secondary browser. We will seek to expand to other browsers in the future such as Firefox, Opera, and Microsoft Edge.
2. *Computer and OS Compatibility*: The main operating systems we will be focusing on are Windows and Mac OS. As long as the computer can run a supported browser, the computer won’t be a factor.

**Expected Load**:

1. We will be ready for 100 concurrent users and our load testing will be done to keep track of high traffic moments and keep an eye on larger servers if necessary.

**Security Requirements**:

1. *Login/Password System*: The login/password system will be standard with Wi-Fi connectivity necessary for login. Requiring a captcha verification to cull the chance of bots overloading the system.
2. *Encryption*: Since there are no current plans for purchasing or exchanging of valuable information, encryption will not be necessary for now. We reserve the right to change that in the future.
3. *Access Control*: The development team will all be granted access to edit front and back-end code as well as the databases being provided to all of us. Users and visitors will have a much more limited access based on the UI.
4. *Spam Protection*: Captcha verification will be used to cover any spam issues.
5. *Resource Utilization*: Resources such as the MySQL database on the lamp.cse.fau.edu server will be accessed through the PHP code using the usernames and passwords therein. All access to the LAMP servers and their resources will be obtained with the usernames and passwords given.

**Portability Requirements**:

1. *Platform Compatibility*: The application will be developed with the focus split between PC and Mobile devices with tablet support coming shortly after.
2. *Percentage of Target-Dependent Statements*: In the initial version of the app, 0% of statements will be target-dependent. For mobile and the eventual tablets, 20% of statements will be target-dependent.

**Supportability Requirements**:

1. *Coding Standards*: Our app will be coded in a range of 75-80% of coding standards for PHP, HTML 5 and CSS3. The code will be produced, reviewed, tested, reviewed again and finalized by yet another developer.
2. *Naming Conventions*: Naming conventions will follow the set standard conventions each language generally follows.

**Storage Requirements**:

1. Since lamp.cse.edu will be our main storage, the capacity is unknown.

**Survivability**:

1. The storage for our app will be held on lamp.cse.fau.edu and GitHub. Our work will be backed up on google drive and a main google doc that all team members will have access to at all times.

**Availability Requirements**:

1. *Accessible Times*: As long as the LAMP server is available, our app should be available 24/7.
2. *Downtime Impact*: A maintenance splash page will be displayed during the scarce times that there is downtime. Besides the case of emergency maintenance, there will be scheduled downtime announced prior to the downtime.
3. *Support*: Support will be provided on a FCFS basis through email, with all emails being responded to within 24 hours, unless there is an unmanageable influx. During times of unmanageable influx, there will be an automated reply to let users know of the backup.

**Fault Tolerance**:

1. *Exception Handling*: Any situation where an exception can occur, will be covered with exception handling. The users will be given an explanation to explain why the exception occurs. They will then be given the opportunity to input the correct data or receive an error message.

High-Level System Architecture and Database Organization

**High-Level Architecture**

Graphical user interface, application

Description automatically generated

**Database Organization**

USER

* ID (INT)
* Name (VARCHAR(20))
* Password (VARCHAR(20))
* Experience (INT)
* Level (INT)

BOUNTY

* ID (INT)
* UID (INT)
* Name (VARCHAR(20))
* Date (DATE)
* Difficulty (INT)
* Notes (TEXT)
* Complete (BOOLEAN)

**Media Storage**

All images will be stored in files under Assets folders. There will be very few images required outside of site design, including mascot and 1-3 star ratings. A single gif will be used for Level Up event, which will also be stored within Assets folder. No video or audio shall be utilized for Task Hunter. All images, including gif, shall be limited to 1mb to increase performance.

**Search/Filter Architecture and Implementation**

Task Hunter will not require a general search bar. However, on the Bounty Board page the bounties will be sorted by difficulty. In the future we may provide a sort-by dropdown for the user to select an attribute to sort by. The attributes that may be selected for this sort would include name, date, and difficulty.

**APIs**

Task Hunter does not require any APIs for its execution.

**Significant Non-Trivial Algorithm or Process**

Task Hunter does utilize a rating system, referred to as difficulty, for bounties. However, this is an exclusively localized rating for the individual user. As far as prioritization, the default sort for bounties on a user’s bounty board will be in ascending difficulty, which is handled by the SQL query upon getting the user’s active bounties. There will be no need to develop an algorithm to compare this rating across users.

High-Level UML Diagrams

Competitive Analysis  
  
 The analysis of competitors’ web apps will focus on five main features (homepage, design, navigation, content, simplicity) and two additional features (tasks, profile). The competitive analysis will utilize a numerical scale (1=bad, 2=poor, 3=fair, 4=good, 5=outstanding) and consists of five web apps chosen for their focus on task management, to-do lists, and gamification.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Task Hunter** | **Bounty Tasker** | **Habitica** | **To-Do Adventure** | **Habit Hunter** | **Quest Log** |
| **Homepage:** | 5 | 3 | 5 | 3 | 4 | 1 |
| **Design:** | 5 | 5 | 5 | 5 | 5 | 3 |
| **Navigation:** | 5 | 4 | 5 | 4 | 4 | 4 |
| **Content:** | 4 | 4 | 4 | 4 | 5 | 3 |
| **Simplicity:** | 4 | 3 | 4 | 2 | 1 | 5 |
| **Tasks:** | 5 | 5 | 4 | 5 | 5 | 3 |
| **Profile:** | 4 | 5 | 4 | 4 | 5 | 4 |
| **Mean:** | **4.57** | **4.14** | **4.43** | **3.86** | **4.14** | **3.29** |

**Task Hunter (4.57)** [Link to be added upon creation]  
 The homepage will be simple with eye-catching colors. If the user has already logged in on their device, they will be taken to their personal home screen, otherwise the option to log will be prominently displayed. Navigation is consistent across the site utilizing a clearly visible menu drop-down in the corner. Content is focused on tasks but does not go as in-depth as other sites. Simplicity is clear and notable throughout the site, but not as overly simplified as Quest Log. Tasks are clear with all necessary functionalities. Profile is concise but missing some in-depth features.

**Bounty Tasker (4.14)** [Bounty Tasker App Page](https://apps.apple.com/us/app/bounty-tasker-to-do-list-rpg/id1229566216)  
 Bounty Tasker is mobile-only. The homepage is quick to auto-load into account creation but is slightly confusing in what you must do to make an account. Design is excellent with pronounced colors drawing attention to key features. Navigation is not perfect, as a couple features are hidden within other tabs. Content is full, but slightly lacking. Simplicity is fair, but some aspects of the app are not clear from the start (such as what happens if you don’t complete your tasks). Tasks are complete with all that is necessary to set up. Profile is perfect with everything necessary but nothing too complicated.

**Habitica (4.43)** <https://habitica.com/static/home>  
 Habitica offers both web and mobile options. The homepage is perfect, with options to create an account and further details of what it offers. The design is beautiful with a preference towards shades of purple. Navigation is clear and concise. Content feels a tiny bit lacking compared to other options. Simplicity is not perfect as its attention to detail goes a bit beyond what one might want. Tasks are a bit overcomplicated. Profile is also a bit overcomplicated.

**To-Do Adventure (3.86)** [To-Do Adventure App Page](https://apps.apple.com/us/app/to-do-adventure-habit-tracker/id1521792040)  
 To-Do Adventure is mobile-only. The homepage quickly forces the user to log in or create an account without providing much information otherwise. The design is exceptional with a focus on oceanic themes. Navigation is a bit clunky as it is not clear where everything is. Content isn’t perfect as some of it isn’t clearly shown. There’s not much simplicity. Even the overall progression of the profile is complicated. Tasks themselves are well structured. Profile isn’t perfect as the progression tied to it is quite complicated.

**Habit Hunter (4.14)** <https://habithunter.activeuser.co/>  
 Habit Hunter is mobile-only, but has a landing web page that provides information and links to which app stores to download it from, however it’s a bit bare in style. The design of the app is phenomenal, and probably my favorite of all competitors. Navigation is a bit clunky due to the wealth of information and options available to the user. Content is full and overwhelming. There is absolutely no simplicity to this app. Tasks are perfect. Profile is exactly what one would want.

**Quest Log (3.29)** [Quest Log App Page](https://apps.apple.com/us/app/quest-log/id1625568037)  
 Quest Log is mobile-only. This app is the definition of simple. The homepage is nearly bare except for options to log in and then your list. The design is incredibly simple, which can be a blessing, but the dichotomy of an exclusively black/white design leaves much to be desired. There is not much content except for the user’s list. A hallmark of simplicity. Tasks are missing some aspects one might want. Profile is a bit better than tasks, but not perfect.

**Team Roles:**

**Team Leader, Back End Lead:** ● Ethan Fleming

**Front End Developers:**  
 ● **Lead -** Meer Hossain  
 ● Marcus Watson  
 ● Brandon Rojas

**Back End Developers:**  
 ● Brenden Martins

**Scrum Master:** ● Ethan Fleming

**Product Owner:** ● Meer Hossain

**Developers:** ● Marcus Watson ● Brandon Rojas ● Brenden Martins

**Key Risks:**