Table of Contents

1. [Project Overview](#_Project_Overview)
2. [Tech Stack](#_Tech_Stack)
3. [Feature Breakdown](#_Feature_Breakdown)
4. [Use Cases](#_Use_Cases)
5. [Database Design](#_Database_Design)
6. [Backend Endpoints](#_Backend_Endpoints)
7. [Tools & Libraries](#_Tools_&_Libraries)
8. [Challenges and Considerations](#_Challenges_and_Considerations)

# Project Overview

Tarkov Guru is intended to be a comprehensive tracker for “Escape From Tarkov”. The project will involve multiple features that allow users to manage and track their in-game progress, including tasks, hideout upgrades, and required items that are needed to progress.

The web application will provide the following features:

* User Authentication: Allow users to create accounts, log in, and manage their data.
* Task Tracking: Users can keep track of quests/tasks they have and mark them as completed.
* Hideout Upgrade Tracking: Users can select the current level of each hideout component and see what items are required for future upgrades.
* Required Items: Display the items needed for uncomplete tasks and remaining hideout upgrades.

# Tech Stack

Frontend: React (with TypeScript), Redux, React Router

Backend: Node.js with Express.js

Database: PostgreSQL

Authentication: Auth0

# Feature Breakdown

User Authentication

* Sign Up / Login: Allow users to create accounts using email and password or third-party providers like google.
* Profile Management: Users can update their profile username, email, settings, and password.

Task Tracker

* Task List: Display a list of all available tasks fetched from API
* Task Status: Allow users to mark task as “completed”
* Filtering and Sorting: Users can filter task by trader (ex. Prapor, Therapist) and sort them by status.

Hideout Upgrade Tracker

* Component Levels: Allow users to manually set the current level of each hideout component (ex. Workbench, Security).
* Upgrade Requirements: Display the required items and resources for the next upgrade level.
* Progress Visualization: Show a checklist or progress bar for each hideout component.

Item Tracker

* Required Items: Display a list of items needed for tasks and hideout upgrades while specifying which items need to be found in raid status.

# Use Cases

# User Authentication Use Cases

1. Sign up with email and password
   * Actor: New User
   * Precondition: User is on the sign-up page.
   * Trigger: User decides to create a new account.
   * Path:
     1. User enters username, email, and password.
     2. User clicks “Sign Up” button.
     3. System validates email format, username uniqueness, and password strength.
     4. If valid and not registered, the system creates a new user account.
     5. System sends a confirmation email.
     6. User confirms email through link provided.
   * Postcondition: User account is created and activated.

# Task Tracker Use Cases

1. View Task List
   * Actor: User
   * Precondition: User is logged in and on the task tracking page.
   * Trigger: User wants to see all available tasks.
   * Path:
2. The user navigates to the task tracking page.
3. The system fetches all tasks from the API filtering out any completed tasks found stored in database and/or filtering based on trader and displays them.
4. The user views the list of tasks
   * Postcondition: The user sees all available tasks
5. Mark Task as Completed
   * Actor: User
   * Precondition: User is logged in and viewing the tasks list.
   * Trigger: User decides to mark a task as “completed”.
   * Path:
6. The user clicks on the completed button next to a task
7. The system updates the task status to “completed” in the database.
8. The task is visibly removed from the list and if requirements are met, the next task is displayed in list.
   * Postcondition: The task is marked as "completed" for the user, and the updated status is stored in the database.
9. Filter and sort tasks
   * Actor: User
   * Precondition: User is logged in and viewing a list of tasks.
   * Trigger: User wants to filter by trader or sort tasks by XP gained.
   * Path:
10. The user selects a filter option (trader) or a sort option (XP gain).
11. The system applies the filter or sort order to the task list.
12. The user views the filtered and/or sorted list of tasks.
    * Postcondition: The task list is displayed according to the selected filters or sorting criteria.

# Hideout Upgrade Tracker Use Cases

1. View current hideout status
   * Actor: User
   * Precondition: User is logged in and on the hideout tracking page.
   * Trigger: User wants to view the status of their hideout.
   * Path:
2. The user navigates to the hideout tracking page.
3. The system retrieves the user’s current hideout component levels from the database.
4. The system displays the current levels of each hideout component.
   * Postcondition: The users sees the current levels of all hideout components.
5. Update hideout status
   * Actor: User
   * Precondition: User is logged in and viewing the hideout tracking page.
   * Trigger: User wants to update the level of a hideout component.
   * Path:
6. The user selects a hideout component and chooses a new level.
7. The system updates the user’s hideout component level in the database.
8. The system updates the display to show the new component level.
   * Postcondition: The updated component level is stored in database and reflected on the user’s interface.
9. View required items for hideout upgrades
   * Actor: User
   * Precondition: User is logged in and viewing the hideout tracking page.
   * Trigger: User wants to see what items are required for the next upgrade.
   * Path:
10. The user clicks on a hideout component.
11. The system displays the required items for the next level of that component.
    * Postcondition: The user sees the required items for the next upgrade.

# Database Design

# Backend Endpoints

# Challenges and Considerations

\* TAKE INTO ACCOUNT THE ITEMS NEEDED REQUIREMENTS FOR COMPLETING AND UNMARKING TASKS \*

Task Related Database:

* The tasks will be kept in three categories. Completed, Active, and Locked.
* All tasks in each category will be stored by id only.
* The active tasks must meet requirements.

1. Level requirement
2. Previous task requirement (if applicable)

* Any task that is not completed or active is locked.

Completing a task:

* Once a task is marked as completed the task will be appended to the completed list. If there is a task in the locked list that requires the now completed task to be completed AND meets the level requirement, the locked task will now be unlocked and active.
* If a player levels up there needs to be a check to see if any tasks now have the prerequisite task(s) done AND now meet the level requirement to be active.

Unmarking a task as completed:

* When a task is unmarked as completed, the task following (if applicable) needs to be locked instead of active. The now unmarked task should be set to active again.