Dorian Kavadlo

Completionist Tracker – Project Proposal

Problem Description

What is the project?
The Completionist Tracker is a web application that helps gamers track their progress toward 100% completion in video games. It allows users to create custom checklists for any game, monitor their progress over time, and explore community-curated achievement lists for games that lack built-in tracking systems.

What specifically is the business logic?
This project includes business logic because it goes beyond simply creating and displaying data. The app calculates each game's completion percentage in real time based on how many checklist items the user has completed. It updates this progress dynamically as the user checks off tasks. It also supports importing community-created checklists, which the system adapts to each user's personal progress. These calculations, progress tracking, and checklist integrations are performed automatically, not just actions triggered by users.

What problem does it solve?
Many video games, especially older titles, lack robust tracking for optional objectives, side quests, or personal challenge runs. Players often use spreadsheets or online forums to track progress manually, which can be disorganized and disconnected from gameplay. This app provides a structured and visual solution for progress tracking that adapts to any kind of game or goal.

Who are the user personas?
Target users include completionist gamers, retro game players using emulators, creators who do challenge runs, and players who enjoy tracking achievements beyond what's officially provided by platforms like Steam, PlayStation or Xbox.

What is the value of solving this problem?
Solving this problem improves the user's overall gaming experience. The app enables users to define what "100% complete" means for them, and helps them track it clearly.

Community checklists also create a collaborative element that supports replayability and engagement. This offers real user value across a wide range of games and playstyles.

How do users solve their problem with the app?
Users create or import a checklist for a specific game, then mark off items as they progress. The app calculates completion percentages and displays visual summaries of progress. Community-curated lists offer additional goals and challenges, allowing users to expand the way they interact with familiar games.

How do users interact with the application?
Users log in to a dashboard where they can add games, build or import checklists, and mark items complete. They can browse community lists, manage their own game library, and view completion stats that update automatically as they make progress.

Minimum Viable Product (MVP)

Overview of Features and User Flow

Users begin by signing up or logging in. Once authenticated, they can add a game to their library and either create a custom checklist or import one from a community list. Each game's checklist includes items the user can mark as complete. The system calculates the percentage of completed tasks and displays the user's progress in a visual dashboard. Users can view a list of all tracked games and optionally explore or contribute community-curated achievement lists.

Minimal Feature Set

Feature	Description
User Authentication	Users can securely sign up, log in, and manage their own game library
Game Management	Add games with title, platform, and genre
Checklist Creation	Users can define goals/tasks for each game
Progress Tracking	System calculates real-time % based on task completion
Community Templates	Users can browse and import community-created checklists
Visual Dashboard	Displays overall progress per game and across all games

High-Level Architecture

The application will consist of three main components:

- **Frontend**: Built with HTML/CSS/JS or a framework like React; it handles user interaction and displays checklists, dashboards, and game data.
- **Backend**: A Flask or Django REST API will handle routing, authentication, data validation, and business logic (e.g. progress calculations).
- **Database**: A relational database (PostgreSQL or SQLite) will store user data, game info, checklist items, and shared community templates.

Data Description

The application will manage the following data structures:

- **User**
    - Fields: id, username, email, password_hash
- **Game**
    - Fields: id, user_id (FK), title, platform, genre
- **ChecklistItem**
    - Fields: id, game_id (FK), description, completed (bool), order
- **CommunityChecklist**
    - Fields: id, title, description, list_items[]
- **UserProgress**
    - Not stored directly; calculated from the ratio of completed tasks to total tasks in a checklist

This structure allows users to track their game progress independently while also sharing or importing community-built goal sets.