DMH

Show Tracker
Software Development Plan
Version 1.0

Revision History

Date	Version	Description	Author
19/Feb/24	1.0	Initial draft of the Show Tracker Plan	Carlos Mbendera

Table of Contents

Software Development Plan

1. Introduction

This document outlines the comprehensive strategy for the development and deployment of "ShowTracker," an innovative iOS application designed to revolutionize the way users track and manage their TV show interests across various streaming platforms. The document serves as a central hub for project management, encapsulating the purpose, scope, methodologies, and organizational logistics required to successfully execute the project.

1.1 Purpose

The purpose of this Software Development Plan is to establish a clear and structured approach to the development of "ShowTracker." It aims to ensure that project objectives are met with efficiency, within the allocated time and resources. This document is intended for use by:

- The Project Manager, to orchestrate project scheduling, resource allocation, and progress monitoring.
- Project Team Members, to understand their roles, responsibilities, timelines, and interdependencies within the project.

1.2 Scope

This Software Development Plan pertains exclusively to the "ShowTracker" project, encompassing the development, testing, deployment, and initial release to external parties of the application. It outlines the methodologies and processes for the project lifecycle, from initial setup and development through to final testing and release.

1.3 Definitions, Acronyms, and Abbreviations

N/A

1.4 References

This section includes all documents and materials referenced throughout the Software Development Plan for "Show Tracker." These references provide additional context, guidelines, and frameworks that support the methodologies and processes adopted in this project. Key references include:

- **Iteration Plans:** Documents outlining the specific goals, tasks, and timelines for each phase of the project.
- **Development Case:** A detailed account of the development methodologies, tools, and practices to be employed throughout the project lifecycle.
- **Other Supporting Documentation:** Any additional plans, guidelines, or documentation relevant to the project, including quality assurance protocols and configuration management procedures.

These references are integral to the understanding and execution of the Software Development Plan and are accessible to all project team members for consultation and guidance.

1.5 Overview

The Software Development Plan for "Show Tracker" is structured to provide a comprehensive roadmap for the project's execution. It encompasses:

- **Project Overview:** Describes the purpose, scope, and objectives of "Show Tracker," detailing the deliverables and expected outcomes of the project.

- **Project Organization:** Outlines the organizational structure of the project team, including roles, responsibilities, and communication channels among team members and external interfaces.
- **Management Process:** Specifies the methodologies for project planning, monitoring, and control, including estimates, schedules, quality control measures, and risk management strategies.

This plan is designed to be a dynamic document, evolving as the project progresses to accommodate changes in scope, timelines, and methodologies while maintaining a clear path towards the successful completion of "Show Tracker."

2. Project Overview

2.1 Project Purpose, Scope, and Objectives

"ShowTracker" aims to offer a seamless and interactive way for users to track their TV show consumption across various platforms. The project's scope includes app development, database management, and API integration, with the objective of launching a user-friendly, efficient, and scalable iOS application.

2.2 Assumptions and Constraints

- Assumptions:
  - Access to reliable TV show data APIs.
  - Stable and scalable Firebase implementation.
- Constraints:
  - Development is limited to iOS platforms.
  - Timeline constraints for project milestones.

2.3 Project Deliverables

Deliverables for each project phase are identified in the Development Case. Deliverables are delivered towards the end of the iteration, as specified in section *4.1.3 Project Schedule*.

2.4 Evolution of the Software Development Plan

The *Software Development Plan* will be revised as needed when deemed necessary by the team.

3. Project Organization

3.1 Organizational Structure

The organizational structure of the "Show Tracker" project is designed to promote efficient collaboration and communication among team members. The structure is hierarchical, with clear lines of authority and responsibility:

- **Project Manager:** Oversees all aspects of the project, from planning to execution, ensuring that objectives are met on time and within scope, also acts as the liaison between academic staff and the team.

- **Assistant Project Manager:** Responsible for organizing and transcribing all team meetings, and will handle the duties of the Project Manager in a situation where the PM is unavailable due to unforeseen circumstances.

- **iOS Development:** Responsible for the design, development, and testing of the "Show Tracker" app on iOS platforms. This includes UI/UX design, coding, and integration with backend services.

- **Backend Development:** Manages the server-side logic and works with API integration, ensuring robust and scalable infrastructure for the app.

- **API Integration:** Specializes in integrating external APIs to fetch TV show data, ensuring

accurate and up-to-date information is available within the app via the Firebase Database.

- **Quality Assurance:** Conducts thorough testing and quality checks to identify and rectify bugs, ensuring the app meets all specified requirements and quality standards.

- **Configuration Management:** Manages versions, changes, and configurations of the project artifacts, ensuring consistency and traceability throughout the development process.

This structure ensures that all project facets are adequately covered by specialized personnel, facilitating efficient progress towards project milestones.

3.2 External Interfaces

TV Show Data Providers (e.g., IMDb API, TVMaze API)

- **Internal Contact:** API Integration Team Member
- **External Contact:** API Support Team
- **Responsibilities:** The API Integration Team Member is tasked with leveraging open data sources to fetch TV show information. This role encompasses understanding the provided API documentation, crafting and implementing data retrieval logic, and ensuring the fetched data is correctly parsed and stored within the project's database. This individual will also handle mapping the data structure to the application's needs, ensuring a seamless integration that feeds real-time TV show data into "ShowTracker."

Firebase (Google Cloud Platform)

- **Internal Contact:** Backend Development Lead
- **External Contact:** Firebase Support Team
- **Responsibilities:** The Backend Development Lead ensures that the API data structures and the iOS app's data requirements are aligned, facilitating smooth data transactions between the app and its backend. This role involves setting up the Firebase project to handle the app's data storage, retrieval, and synchronization needs effectively. The Backend Development Lead will work closely with the API Integration Team Member to ensure that data fetched from TV show data providers is correctly formatted, stored, and accessible within the Firebase database, ensuring consistency and reliability in the data presented to users.

Apple App Store (TestFlight Public Beta Testing)

- **Internal Contact:** iOS Development Lead
- **External Contact:** Apple App Store Support Team
- **Responsibilities:** The iOS Developer is tasked with leveraging Apple's TestFlight for distributing "ShowTracker" to public beta testers. This role encompasses preparing the app for beta release, which includes ensuring the app's stability and usability are up to standards that provide a meaningful testing experience. The iOS Developer will manage the beta testing process by uploading the app to TestFlight, configuring beta testing settings, inviting testers through email or a public link, and gathering feedback for subsequent iterations of the app.

3.3 Roles and Responsibilities

Person	Unified Process for EDUcation Role
Carlos Mbendera	Project Lead, iOS Developer, Writes iOS App and Oversees the entire project, Liaison between faculty and team.
John Tran	Assistant Project Lead, Backend Engineer, Organizes and Records Meetings, Write API Code to get and parse TV Shows
Cole DuBois	Quality Assurance Engineer
Blake Ebner	Configuration Management Engineer, GitHub Repo Management
Kenji Craig	API and Firebase Integration, Makes Sure that iOS App and API Code can access the same data via Firebase

- **Rotating Among Team Members** Technical Documentation Production

4. Management Process

4.1 Project Plan

The project plan includes a detailed schedule for achieving the project milestones. Key deliverables are as follows:

- **Week 2 -** Project setup documentation, Firebase configuration guide, GitHub repository setup.
- **Week 4 -** API data retrieval script with documentation.
- **Week 6 -** Initial iOS app design prototype.
- **Week 8 -** Integration of the iOS app with the Firebase backend.
- **Week 9 -** Final project report, including testing, debugging logs and writing other technical documentation

4.1.1 Iteration Objectives

- **Requirements Iteration -** Team Members meet and discuss project goals and roles.
- **Design -** Very short phase where the team agrees on features and design of the project
- **Implementation -** In this phase, the API Backend, Firebase Database and iOS App will be completed
- **Testing -** Quality Assurance will work with the entire team to ensure that all features are work well
- **Deployment -** Release of iOS App to Beta Testers and Users

4.1.2 Releases

- **Week 4 -** Backend and Database
- **Week 6 -** Prototype iOS App with a successful communication between the Backend and Database
- **Week 8 -** Beta iOS with all features done and awaiting quality control

4.1.3 Project Schedule

- **Weeks 1-2:** Requirements
- **Weeks 3-4:** Design
- **Weeks 5-6:** Implementation
- **Weeks 7-8:** Testing
- **Week 9:** Deployment and Project Presentation

4.2 Project Monitoring and Control

4.2.1 Quality Control

Quality control involves regular walkthroughs and reviews to ensure deliverables meet predefined criteria and guidelines. Defects identified are tracked as Change Requests on GitHub.

All deliverables are required to go through the appropriate review process, as described in the Development Case. The review is required to ensure that each deliverable is of acceptable quality, using guidelines and checklists.

Any defects found during review which are not corrected prior to releasing for integration must be captured as Change Requests so that they are not forgotten.

4.2.2 Risk Management

Risks are initially identified during the Inception Phase and reassessed at each iteration. This proactive approach allows the team to prioritize, monitor, and mitigate risks effectively, ensuring project stability and progress.

Risks will be identified in the Inception Phase using the steps identified in the RUP for Small Projects activity “Identify and Assess Risks”. Project risk is evaluated at least once per iteration and documented in this table.

4.2.3 Configuration Management

A version-controlled repository (e.g., GitHub) houses all project artifacts, including source code, documentation, and executables. Change Requests are managed to ensure all modifications are documented, reviewed, and approved, maintaining project integrity and traceability.

Blake will be overseeing the lifecycle of this project and watching for the consistency of the code and our project.

5. Annexes

The project will follow the UPEDU process.

- **Programming Guidelines:**
  - **Language Specification:** State the specific programming languages being used for the project, limiting codebase to these languages for consistency and expertise focus.
  - **Minimal and Efficient Code:** Strive for the minimal code that accomplishes the task efficiently to ease maintainability and debugging.
  - **Code Commenting:** All code must be properly commented to explain functionality and logic for future reference and team understanding.
  - **Code Components Explanation:** Clearly document the purpose and functionality of code components, e.g., "API Fetcher: Responsible for retrieving and processing data from designated APIs."
  - **Code Compression:** Aim for conciseness in code to prevent excessive length, making it easier to identify errors and understand logic.
- **Design Guidelines:**
  - **Modular Design:** Avoid monolithic structures by developing self-contained modules that interact through well-defined interfaces.