StoryConnect

Design Document

Table of Contents

Executive Summary

Introduction

StoryConnect's primary purpose is to provide a platform for connecting writers, readers, and publishers. We plan to do this by leveraging a range of different technologies to make sure each of our users feels comfortable with any tools they decide to use.

Problem

Current solutions are a disjointed experience. While some platforms cater to providing writers with helpful tools, they cannot help writers launch their works into the limelight. However, the platforms that provide a community element lack writing tools that help elevate the author's work to the next level and have an unappealing interface.

Solution

StoryConnect will solve these issues by acting as a tool for readers and writers without cannibalizing others' experiences. We plan on letting each of these user types interact with each other and ultimately help writers better compose works and for readers to feel connected to the work they are reviewing.

Methodology

This platform is designed to connect two types of users: readers and writers. Readers will be able to easily find books they love in the platform's library through the use of suggestion algorithms to improve engagement. Readers will also provide feedback and comments for writers, creating a community of passionate readers. Publishers, a subtype of a reader, can additionally evaluate books and connect with authors through the platform, providing additional opportunities for writers.

The platform offers writers an opportunity to publish their books and receive ratings, reviews, and proposed edits from readers. Additionally, writers will have access to a suite of machine learning tools, including a 'Continuity Checker,' 'Road-Unblocker,' and data-visualization tools, which cater to the writing process. While this document will discuss specifics of technologies later, we aim to help writers create books that readers will love.

Background

Overview

StoryConnect is a platform for enjoying, authoring, and sharing amateur fiction. We provide a host of writing tools that allow authors to map out, organize, and bring to life their stories. Beyond our suite of writing tools, StoryConnect will connect you to a vibrant community of fellow readers, writers, and publishers, each taking on active and passive roles.

Users

We are a platform for story lovers of all kinds. We cater to those who love to get lost in another's world or create their own. By offering an extensive suite of professional writing tools, we encourage anyone to tell their story in the most streamlined way possible. Once a writer begins penning their story, we hope to give them a community of like-minded individuals who can engage with their work and help them along the way. Readers can then casually enjoy the new works or dive deeper with our annotation and feedback system. This system will allow readers to mark pages, highlight passages, and enjoy the story their way, allowing their contributions to be visible to the authors, as discussed previously.

Similar Ideas

Since our goal is to bridge feature gaps, we have taken inspiration from several existing platforms: Wattpad and Chapterly. Wattpad allows readers to browse for books they like and provide feedback through annotated comments throughout the book, subsequently increasing engagement. However, Wattpad only provides limited writing tools, hampering the overall quality of the works produced. Moreover, Wattpad's user interface doesn't cater to users' desires. Moving on to Chapterly, there is a stark contrast in its offerings compared to Wattpad. It offers a richer, more extensive writing toolbox to help create a more organized experience. Chapterly also provides features such as chapter sections, character sheets, roadmaps organization, and annotated notes on the story that helps writers better keep track of their writing process and progress.

Front-End

Our goal is to make StoryConnect available on both Mobile and Web platforms. Rather than developing separate front ends for each platform, which could be challenging and time-consuming, we'll use Dart (whose UI Framework is Flutter) to create a single codebase that works across all platforms. Doing so simplifies the development process and allows Front-End developers to focus on features rather than struggling with platform idiosyncrasies.

Back-End

Our Back-End will handle various tasks, such as receiving requests from the Front End, managing Machine Learning Functions, and querying the Database. We have chosen to use the Python/Django stacks to accomplish these tasks. Thanks to its wide range of data science libraries, Python is an excellent choice for handling machine learning tasks. Additionally, Django is a user-friendly framework for building web servers. We've also decided to use PostgreSQL for our DBMS due to its easy integration with Django. As for hosting our web application, We've opted to use DigitalOcean's Cloud Services. Finally, we'll utilize Docker to quickly deploy and test our web server while minimizing installation-based hangups.

Assets and Engines

Several of our functionalities incorporate machine learning-based algorithm features utilizing Natural Language Processing techniques, namely sentiment analysis, text classification, keyword extraction, and many more, in which we plan on making use of libraries from Python and integrating them into our program we built from scratch. In addition, one of our planned features is Road-UNBlocker which we plan to leverage tools similar to ChatGPT from their open-source API.

Software and Hardware Requirements

Our application will run seamlessly on various devices, including iOS and Android devices and Desktop platforms via a web browser. Specifically, our app will run on platforms supporting these minimum requirements: Android devices with API 16 (Android 4.1) and iOS devices running iOS 11 and above. For web browsers, we recommend using Chrome 84 and above,

Firefox 72.0 and above, Safari on El Capitan and above, and Edge 1.2.0 and above. For potentially more up-to-date information, visit:

https://docs.flutter.dev/development/tools/sdk/release-notes/supported-platforms

For our developers' toolchains, we will similarly be platform agnostic. Our team will use VS Code as our primary IDE, available on all major OSes, with accompanying installations of Python and Dart and supporting frameworks and extensions. Also abstracting away from any hardware requirements on the server side, we'll be using Docker on a cloud service provider (most likely digital ocean). This decision will make it simple to deploy our back-end on many platforms, so long as they support Docker itself.