Haley Beauchamp, Ryan Seaman
CSCI 4063: Senior Capstone (Tashfeen)
10 October 2025

Proposal

1. Description:

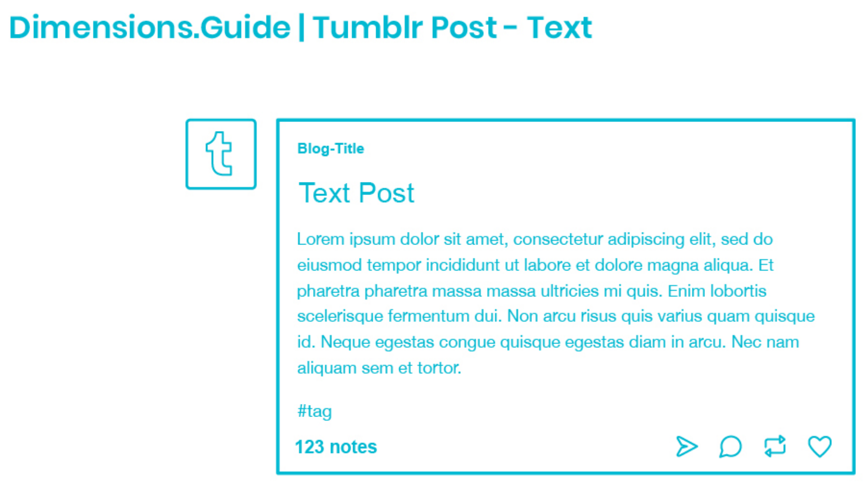
A cross-platform mobile app that recommends posts to users based on their progress in a tv series to avoid spoilers and promote a safe fandom experience. It will have a simple UI similar to Tumblr, where users can read and write text-based posts about fandoms they enjoy. We will implement Retrieval-Augmented Generation (Section 4) to flag spoilers and improve the user experience.

2. Libraries

- **Frontend:** Flutter, Dart
- **Backend:** Express, TypeScript, Node.js
- **LLMs:**
  - **Platform:** OpenAI API
    - **Models:** Text Embedding 3 Small, GPT-5 Nano
- **Database:** TypeORM, PostgreSQL, hosted by Neon
- **Containerization:** Docker

3. Features

1) User accounts and authentication
2) Text-based UI similar to Tumblr



Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Et pharetra pharetra massa massa ultricies mi quis. Enim lobortis scelerisque fermentum dui. Non arcu risus quis varius quam quisque id. Neque egestas congue quisque egestas diam in arcu. Nec nam aliquam sem et tortor.

Figure 1. Sample text based UI.

3) Ability to input your current episode for a given tv show
4) Recommendation of posts based on current episode in a series - spoiler control, tagging system
5) Post interactions like liking and commenting
6) User following system
7) User customization like profile pictures

4. Retrieval Augmented Generation

Retrieval-Augmented Generation is a method of allowing llms to respond to queries using supporting information beyond their training data.

Our implementation will involve retrieving relevant transcript data from a database to inform llm spoiler detection. The data is sourced from online transcript websites which are then chunked and stores as vectors in our PostGreSQL database hosted on neon. Posts are tagged with a show, and when a user's feed is being generated, each show is checked for spoilers. We pass the post in question to a chunking model, then we query the databse for context from transcripts for that show using cosine similarity. Then, the resulting context is used by the LLM to determine if the post is a spoiler for the current user.

Our database will be modeled, generally, as such. We will have users table, a posts table, and relevant join tables for likes and followers. Our vector database will have a shows table, an episodes table, and a chunks table so that we can store indexed chunks for every episode of every show. There will likely be other tables as well, but this is ultimately the largest functional piece of the database design.

5. Stretch Goals

- Support for multimedia posts (images, videos)
- Support for media types beyond tv shows (movies, games)
- Enhanced profile customization (themes, advanced layouts)

Oklahoma City University, Petree College of Arts & Sciences, Computer Science