AlethioSphere

Product Design Document

Team 21

Saketh Ayyagari: 2023111008

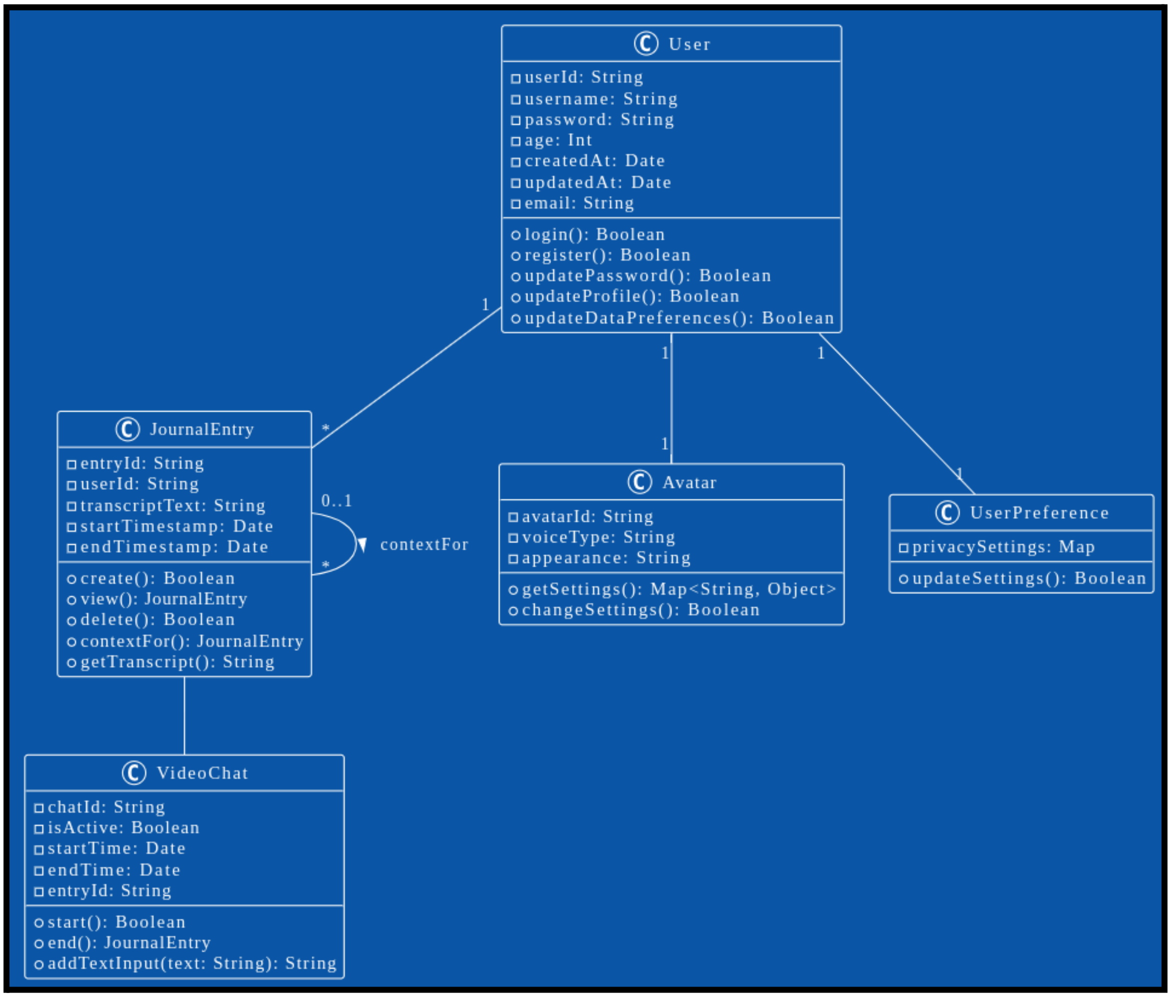
Raghav Doshi: 2023111036

Anshul Bhagwat: 2023101001

Ishaan Romil: 2023114011

Manasi Mundada: 2023101087

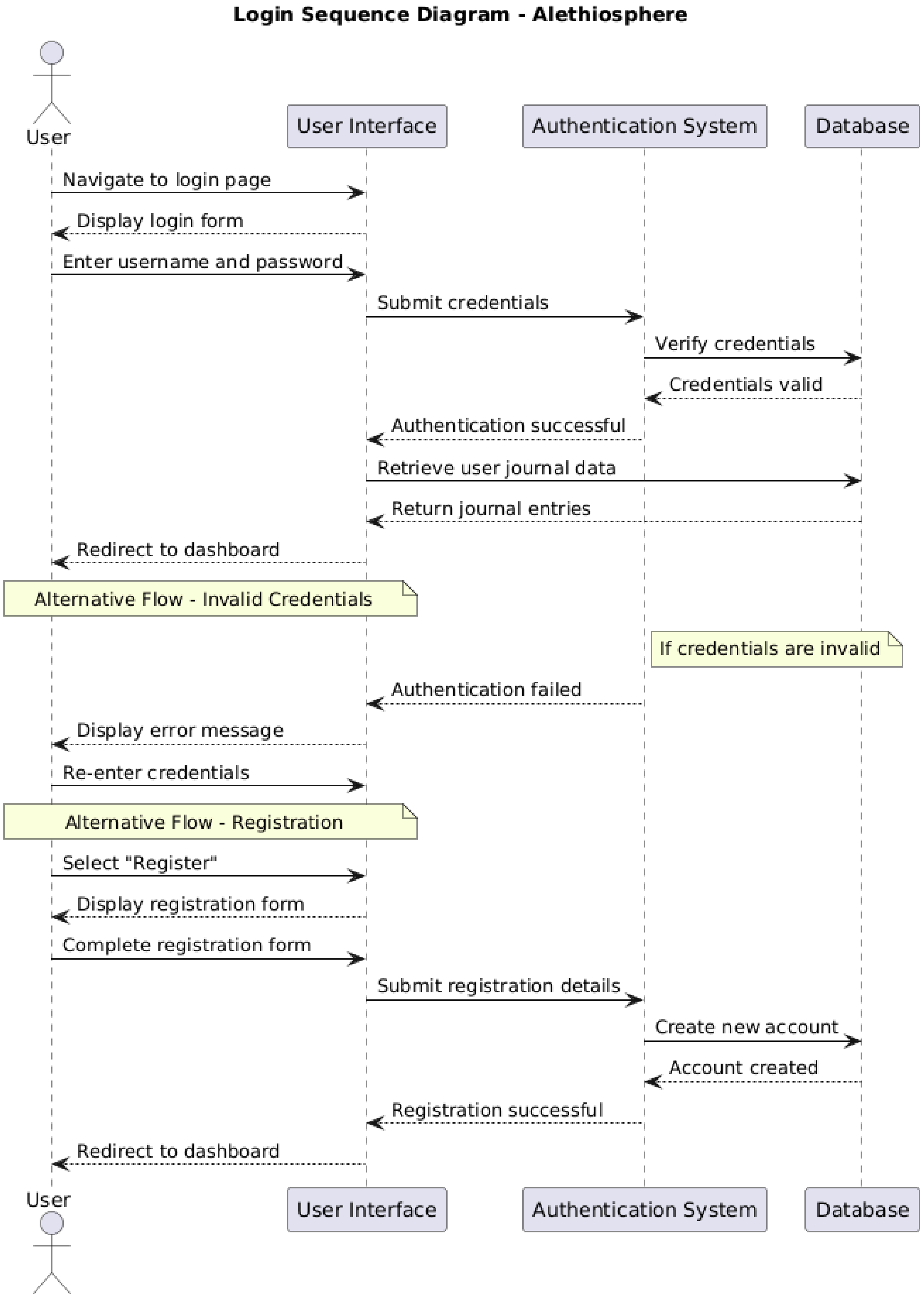
Design Model



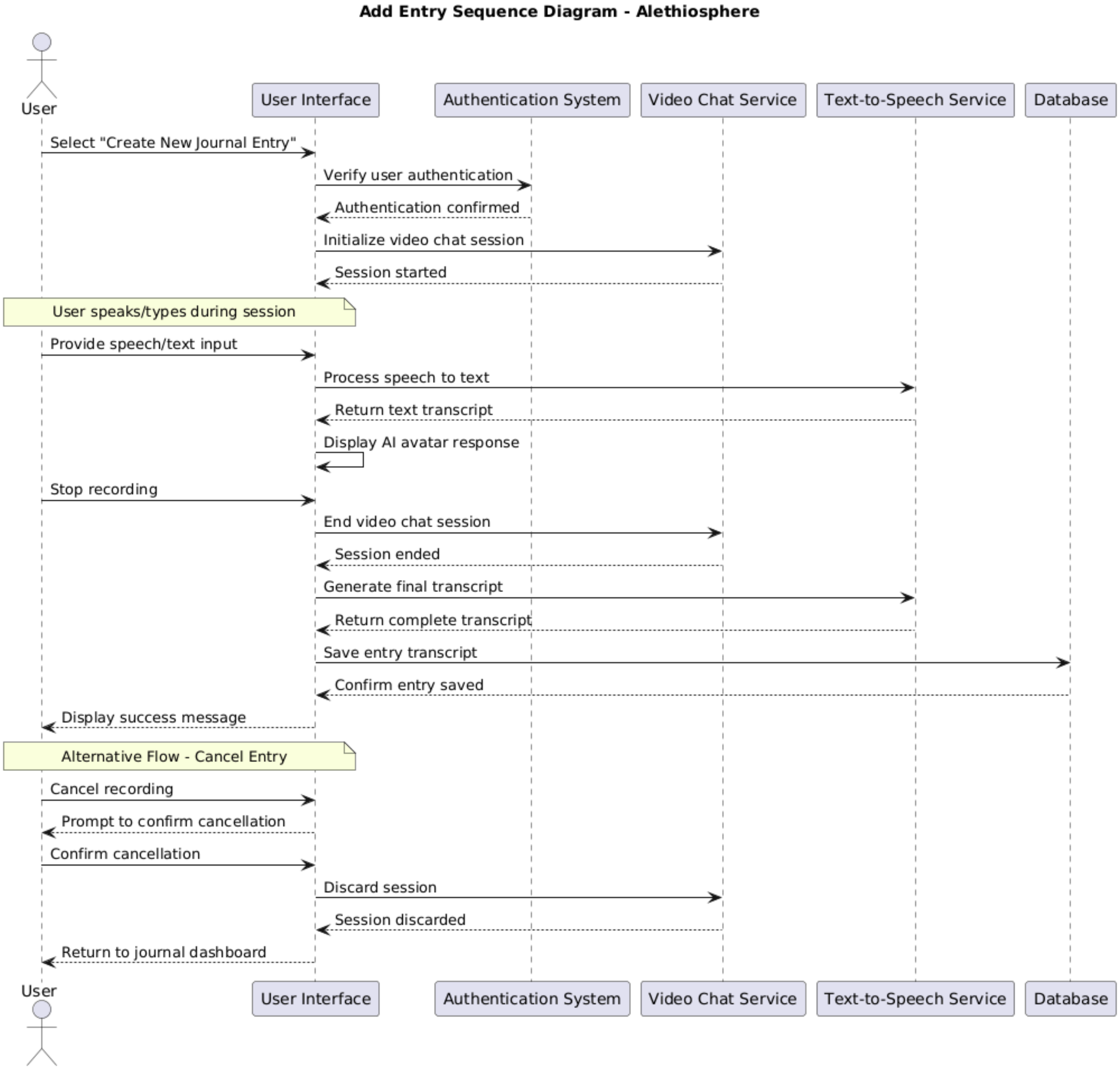
User	**Class State:** The `User` class is responsible for maintaining user information such as `userId`, `username`, `email` and `createdAt` **Class Behaviour:** Implements methods to update their data (`updatePassword()`, `updateProfile()`), update data preferences (`updateDataPreferences()`), and manage user authentication.
Avatar	**Class State:** Stores avatar-related properties such as `avatarId`, `voiceType`, `appearance`. **Class Behaviour:** Provides methods to retrieve settings (`getSettings()`), and update settings (`updateSettings()`).
JournalEntry	**Class State:** Holds journal entries with attributes like `entryId`, `userId`, `startTimestamp` and `endTimestamp` **Class Behaviour:** Implements methods to get the transcript (`getTranscript()`), view the entry object (`view()`), create a new entry (`create()`) and delete entries (`delete()`), along with the JournalEntry that this serves as context for.
VideoChat	**Class State:** Maintains data about video interactions, including `chatId`, `userId`, `startTime`, `endTime`, `isActive`, and `entryId` as a reference to the corresponding JournalEntry object. **Class Behaviour:** Implements methods to connect to journal entries (`toJournalEntry()`), cancel ongoing sessions (`cancel()`), and continue from a previous entry (`continueFromPrevious()`).
DataPreference	**Class State:** Maintains user-specific data and privacy preferences. **Class Behaviour:** Implements methods to update settings (`updateSettings()`) and retrieve settings (`getSettings()`).

Sequence Diagram(s)

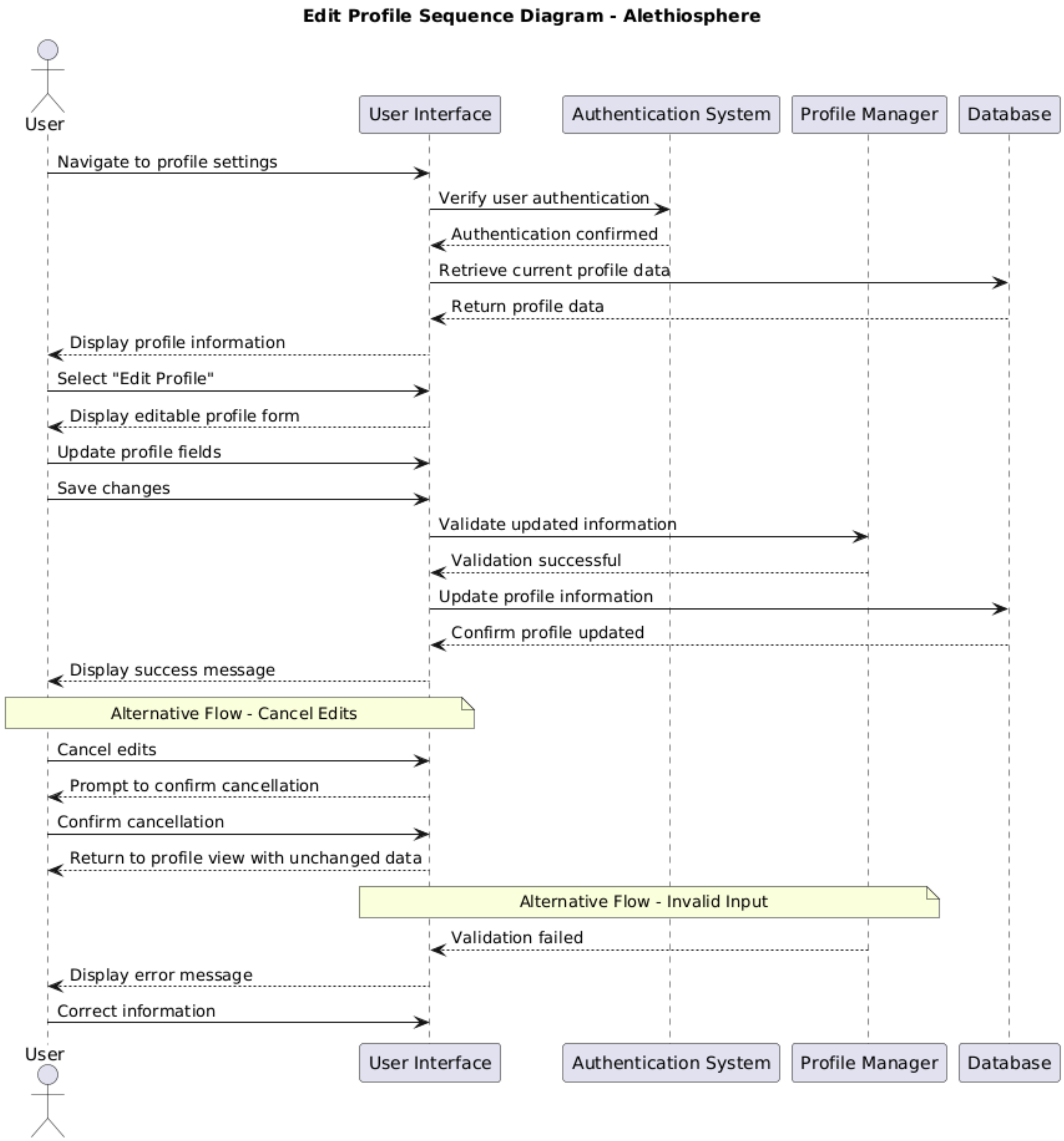
UseCase 1. Login



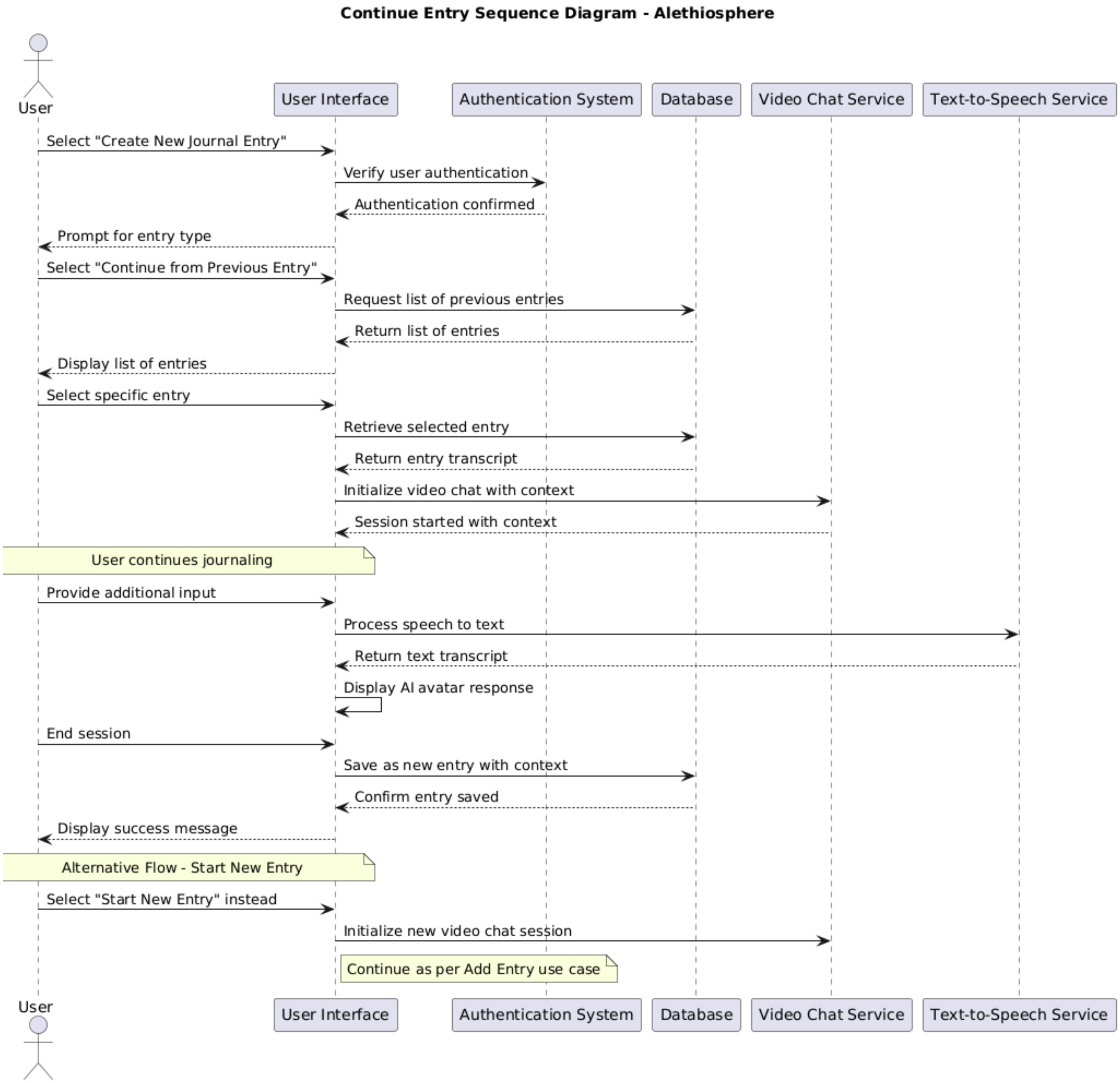
UseCase 2. Add Entry



UseCase 3. Edit Profile



UseCase 4. Continue Entry



Design Rationale

AI Pipeline Design Rationale

1) Used API calls rather than running the LLM locally
   a) This was to save on compute power and computational resources and since a network connection was guaranteed anyway (this being a web app), it seemed the logical choice
2) Used DeepSeek/ChatGPT instead of the other options
   a) This was to optimise on costs
   b) We thoroughly explored Claude and Gemini as well, finding that Gemini was too mechanical and Claude was considered not cost-efficient
   c) DeepSeek and ChatGPT provided good responses to initial prompting and are also considerably cheaper

Avatar Design Rationale

1) Used Live2D with Three.js instead of a 3D Avatar
   a) A 3D avatar would have required significantly more computational resources for real-time rendering and physics, increasing both performance overhead and code complexity, making it extremely challenging to finish within the given time-frame. And even if we finished, it would have been difficult to achieve the level of quality we are aiming for due to the aforementioned computational resources.
   b) Live2D allows for expressive animation while maintaining a lightweight web-friendly implementation, essentially addressing the performance issue of a 3D avatar
   c) The disadvantage with using Live2D (and a 2D avatar in general) is the lack of realism. However, we believe this is offset by the significant expected increase in quality
2) Used Rhubarb Lip Sync instead of AI-based phoneme prediction
   a) AI-based phoneme models (like OpenAI's Whisper for viseme extraction) could have provided more accurate mouth movements but would require real-time processing, increasing latency and decreasing performance.
   b) Rhubarb is lightweight, provides precomputed lip-syncing, and integrates easily with Live2D by providing a .json file which is directly compatible with Live2D's .moc3 output file.
   c) The disadvantage of using Rhubarb is a decrease in accuracy but we expect it to not be a significant issue as Rhubarb will still be able to approximate to a very large degree.
3) Frontend Implementation: Used Three.js for animation instead of a game engine like Unity
   a) Unity WebGL export is heavy and has long load times, making it unsuitable for a browser-based application like ours.
   b) Three.js is lightweight, integrates seamlessly with Live2D, and allows for direct JS control over the avatar.
   c) The disadvantage of using Three.js is that it looks slightly less realistic and may not be able to simulate more complex movements which we may want in future versions. However, we expect this to not be too detrimental to the overall project.

UI/UX Design Rationale

1. Using Figma for designing screens:
   a. To provide a good idea of the design, layout and colour palette of the screens.
   b. Very easy learning process and very powerful to create designs quickly.

2. Using NextJs for frontend development.
    a. Already familiar with the usage and features of it
    b. There are good routing features and fast loading of web pages.
3. Using tailwind css for styling web pages..
    a. Mainly due to familiarity of usage.

Backend Design Rationale:

1. Using MongoDB (NoSQL) instead of SQL databases:
    a. NoSQL permits us the flexibility of storing different kinds of data
    b. Since we are already familiar with the mechanics and syntax of MongoDB, adopting it would help streamline the integration of persistent data storage with the backend