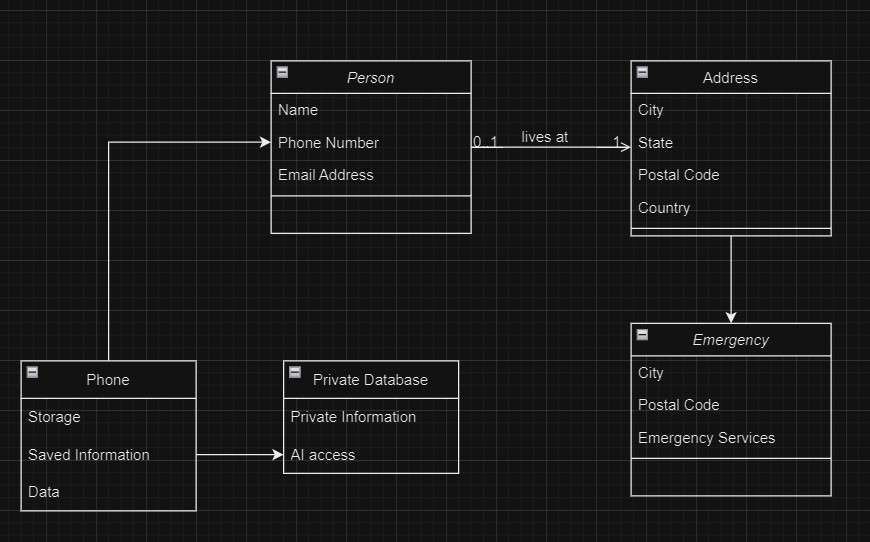
**Product Design**

|  |  |  |  |
| --- | --- | --- | --- |
| **Team:**  **Members:** | | Ctrl-Alt-Elite  Austin Hoffman,  Akash Patel,  Hadia Bilal,  Shawana Tahseen,  Timmy Akindunni | |
| ***Revision Number*** | ***Revision Date*** | | ***Summary of Changes*** | | ***Author(s)*** |
| *Initial* | *09/25/2023* | | *Initial Creation* | | Austin,  Akash,  Hadia,  Shawana,  Timmy |

# Class Diagram(s)

Class diagram relating to user information, and their emergency contacts (CD1):



Class diagram relating to the UI screens and interactions (CD2):

A computer screen shot of a computer

Description automatically generated

# ER Diagram

A diagram of a computer

Description automatically generated

# User Interface Wireframe(s)

A screenshot of a login screen

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a login

Description automatically generated

A screenshot of a computer screen

Description automatically generated

# Design Summary

Our current focus is on the layout of UI. This includes our screens and database layout.  
We have a variety of screens, each with their own unique features. Those are pictured in CD2. The linkage between them, and the visual presentation are shown in the wireframes.

We have chosen a palette design that is bright which creates an open and calm atmosphere.

For the database design, we structured it with entities like user, profile, conversation, feeling log, user pattern, emergency contact, and emotion, all of which are interconnected through well defined relationships.

For UI implementation, we plan to use a base UI class containing common variables and shared functions, then use inheritance to implement the more unique screens.

For AI implementation, we plan to research, collect data, and train the AI model. The app will be structured in such a way that the user can send requests to the AI and get responses. While the AI is in training, we will be using stubs and premade responses in order to realize the functionality of the app before full AI integration.

# Design Rationale

**Considered Designs:**

**Cloud-based AI**: We initially consider leveraging cloud-based AI systems for the chatbot feature. However, this was ruled out in favor of local AI execution, emphasizing user privacy and data security.

**Application Development Environments**: We briefly considered using Unity or Xamarin Forms to create the application because of some of the features each tool possessed. We disregarded these different tools though because there exist better and more focused tools for our use case.

**Database:**

We had considered creating our own system to control the conversation history and storage, however, we decided to go with SQLite due to its built-in functionality and ingrained features.

**Selected Design and Rationale:**

**Flutter Mobile Application:** Flutter was chosen for its cross-platform capabilities, allowing a single codebase to cater to both iOS and Android users. This optimizes development resources and ensures consistent user experience across devices. Flutter also runs on Dart, which contains syntax and design very similar to C#. This is good because C# is a language that simplifies and improves upon C, a language the entire team is familiar with.

Our decision to use flutter was also in part to Flutters ability to simplify UI creation. With Flutter, making intuitive and easy to use UI is much easier.

**Local AI execution:** Keeping the AI operations local ensures data privacy. Users will be more inclined to be honest and open if they know their sensitive data isn't leaving their device. Keeping the AI local also makes it so that our application does not rely on external servers, so the project will continue to work even without future support.

**User Log Feature:** Allows users to actively record their feelings, fostering a sense of engagement and responsibility towards their mental health journey. We decided on this because it further encourages the use and appreciation of the apps correlation abilities. Beyond being able to just converse with the AI, you can also use the app to directly store your emotions. Using this feature in tandem with chatting with the AI, allows the user to see possible correlations between events from their day and how they are feeling.

**Simple AI Chatbot Design:** Instead of a complex, multifunctional chatbot, we opted for simplicity to ensure it's approachable, friendly, and doesn't overwhelm the user. Its primary function is listening and drawing insights, in line with our objective to provide users a virtual companion. This design choice decreases the broadness of the AI’s knowledge, but allows the AI to run on simpler devices, and to keep its responsibilities strict so it performs as best it can with its resources.

**AI vs Non-AI Parts:** For our development, we have decided to break down our tasks by the involvement of the AI. Tasks will be categorized by how much they rely on the AI, features that require no AI involvement will be more prevalent in sprints, and features regarding AI, like the chat bot, or conversation correlation will be less prevalent in sprints. This will let us slowly build the AI and to instead focus on the underlying features, so that in the later stretches of development the AI is our sole focus.

**UI Visual Design:**

We chose the bright palette design because colors influence moods and emotions, so we need to create a sense of safety for users. The bright and open atmosphere of the app should also help put users in the right mindset. The wireframe is a demonstration of the initial design of the app, the final look however is designed to promote openness, but also the feeling that the app is secure, and the private things they share are kept only in this one place.

**Database Design Rationale:**

The database design demonstrated by the ERD supports the core functionalities of the app, allows users to start conversations, log in the emotion they are feeling and to also provide emergency contacts in case anything goes wrong. We structured it this way to be able to simply and efficiently support the features presented to the user.

**Security:**The users security is very important. The users information will be stored securely on the app, and it will not be exported away from it. We plan to implement this by allowing the user to create a password, or way to uniquely identify them so only they have access to their information.

Another way we plan to keep their secrets private, is in our use of local AI and not making any external calls. Our users do not need to fear a third party using their private information for marketing, or any other purpose. They also have the assurance that should they want something gone forever, all they need to do is delete it in the app.

**Emergency Response:**  
We plan to implement a simple emergency detection and response system. It monitors for certain audio inputs, like sobbing, screaming, etc. Should it detect such a thing, it will try to initiate a conversation, if it is unsuccessful it will contact an emergency contact. To implement the audio detection, we plan to use a very simple and small AI trained on a small dataset of such audio.

**Emergency Contact:**  
Our database contains fields to store information regarding emergency contacts. Our screens also permit this functionality. Should the user not give an emergency contact, the contact default is set to the suicide hotline. This system is tied into the emergency response system and will connect with the phones call functionality.

**Progress and Goals:**

The user is able to interact with the database through the screens to record their goals. They will also see their progress displayed in a manageable way. This implementation is separate from the AI, but ties into the log history and correlation features of the application.

**TTS and STT:**

We will integrate the user’s device’s ingrained TTS and STT functionalities in the application in order to provide simple TTS and STT functionality. This will be an option the user can enable in the settings and will provide accessibility to the user. It will also provide a simple alternative to typing. The applications text box will allow such input.

**Pattern Correlation:**  
The users log history and conversations will be used in our correlation algorithm to try and map certain experiences the user notes with how they feel. This will be displayed on a screen to provide the user with easy-to-understand information about possible correlations.

**Check-in Functionality:**  
The app will work with the devices notification API to periodically send check ins to the user. These will be attempts to start conversations with the user, allowing them to discuss how they feel, or vent about their day. This feature will remind the user to interact with the application and will also provide more data for the correlation system.

**User Profile:**  
The application will implement the user profile through the database. Fields have been delegated in the database to be able to store the user information and key entries to that user. The user profile will also be what allows access to view such information. Only the user who owns such information will be able to view it. This ties into our security features.

**Trade-offs and Considerations:**

Local AI execution might limit the computational capabilities compared to cloud-based solutions. However, we believe the privacy advantage far outweighs the potential drawbacks. Users will be sharing sensitive information, and we want to make sure they are absolutely assured their secrets are secret.

The simplicity of the AI may sometimes not capture all complexities of a conversation. We accepted this trade-off for the sake of user-friendliness, preventing misinterpretations that can arise from information hallucinations, and resource usage. By creating a simpler, more focused AI, older devices may be able to run the application because of the lowered device resource requirements. Making our AI model simpler also allows us to personally curate training data due to the smaller dataset needed for training, which will keep responses more helpful and reduce the chance of harmful responses.

**Database**:

We have elected to use SQLite, which may add more complication than designing our own database system due to the nature of integrating it into Flutter. However, we believe the simplicity of use once it is implemented outweigh the initial costs of integration.