Requirements and Design

1. Change History

Change Date	Modified Sections	Rationale
Nothing to show		

2. Project Description

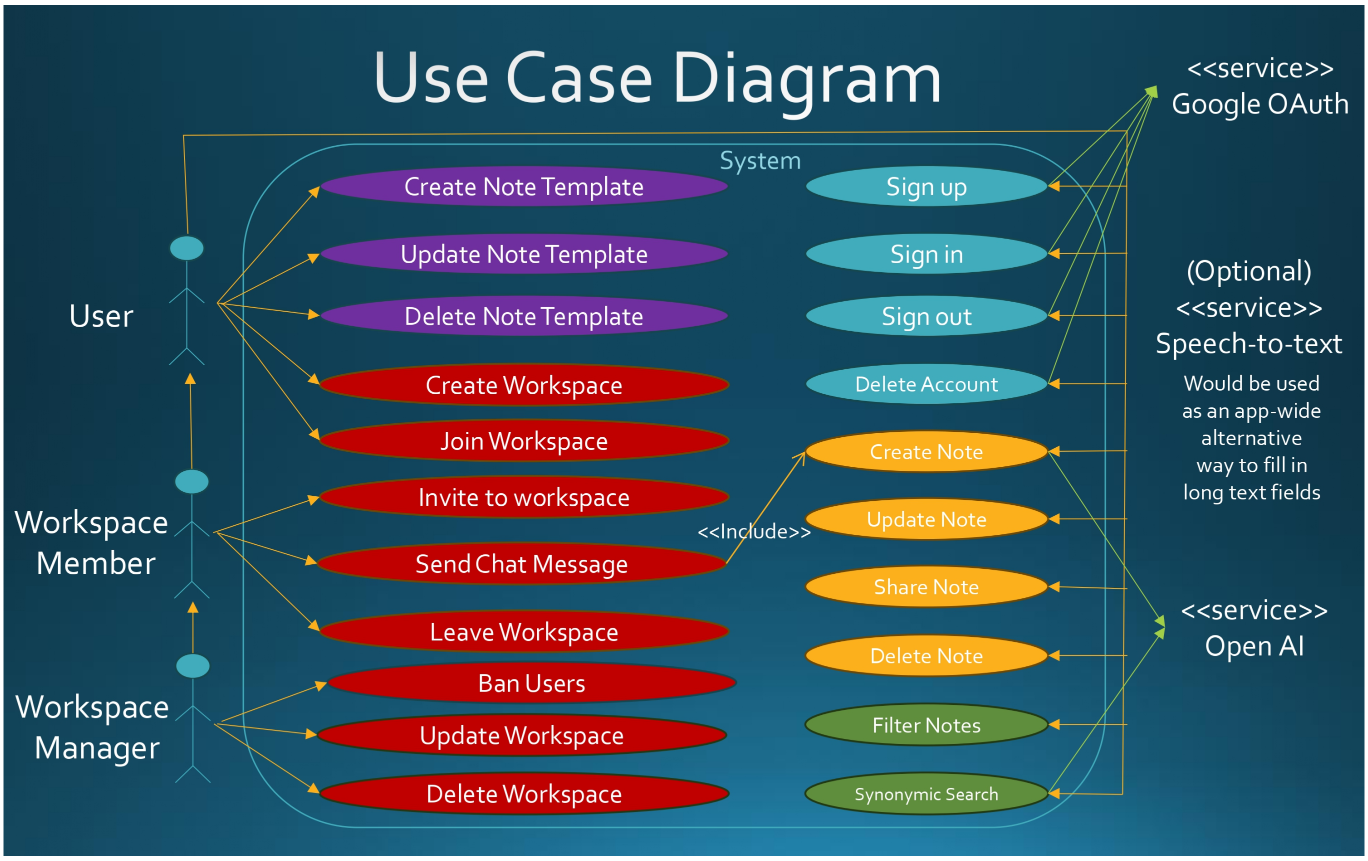
In short, we are aiming to create an application that would allow users to store all their (non-sensitive) information in one place andin a way that is easy to update, access and search. The app itself would not incorporate dedicated features such as notes, todo-lists etc. Instead one would be able to store information in a general note, which would be composed of numerous input fields: text, datetime etc. Users will be able to create templates for their own formats of notes and join workspaces where they can share their notes with other users. The target audience is the general public. If one has a large amount of different kinds of notes, todos or saved locations, they can use this app as a "general memory system" providing them one place to retrieve and update their entire collection of notes.

3. Requirements Specification

3.1. List of Features

1. **[Manage Notes]**: A user can create, update, read and delete their own notes as well as share it with other users and workspaces they are in.
2. **[Retrieve Notes]**: A user can search through their notes and filter through the notes they see by their tags and creation/last edit dates. The searching will be synonymic, which means cases when the user does not type exactly the content of the note would be handled and the note will still be displayed.
3. **[Collaborate]**: A User can create workspaces in which case they become their managers. Inside a workspace, notes can be posted regularly or with a “chat” option that would send notifications to the users in the workspace. Notes in the workspace are visible to all Users who accepted the invitation. Naturally, the manager can update their workspace, or even delete it, as well as ban certain users.
4. **[Customize Format]**: Users would be able create and manage their own formats of notes, known as templates. These can include any combination of text, location and date fields, with the condition of at least one field. When creating a note, they would be able to start from the template of their own making.

3.2. Use Case Diagram



3.3. Actors Description

1. **[User]**: The general user of the application. Can fully manage (CRUD + search + template creation)their own notes. Can join and create workspaces.
2. **[Workspace Member]** - can contribute to workspaces they are in by sending notes and chat messages, as well as inviting new members.
3. **[Workspace Manager]**: Inherits from User. Has additional options to update and delete the workspaces they own. Can also ban users from their workspace(s).
4. **[Google OAuth API]**: External service used for authentication.
5. **[Open AI API]**: Or equivalent. The role of the API would be to create vector embeddings for notes and search querries so that vector-based search algorithms, such as KNN, can be used. This is to enchance the search quality so that the user does not have to exactly match the text of the note in their prompt.
6. **[Text-to-Speech API]**: Optional. Would be used as an alternative way to fill in any text field inside the application. If T2S gets implemented, then there will be a microphone icon next to every text field and the user would be able to press it and say, what they want filled in, instead of actually filling it in.

3.4. Use Case Description

- Use cases for feature 1: Manage Notes

1. **Create Note**: The user can create notes by filling in a chosen note template, adding or removing fields if necessary. They can then store this note in a chosen workspace.
2. **Update Note**: Users can update their notes and change the title, description, and other data.
3. **Share Note**: Users can share their note to a selected workspace.
4. **Delete Note**: Users can delete their selected note.

- Use cases for feature 2: Retrieve Notes

5. **Search Notes**: A user can search for notes matching a given prompt, and is provided with a list of notes.
6. **Filter Notes**: After retrieving search results, a user can filter the results by certain tags and creation/last edit dates.

- Use cases for feature 3: Collaborate

7. **Create Workspace**: A user can create a workspace and become the manager of it.
8. **Join Workspace**: A user can join the workspace they are invited to or reject the invitation
9. **Invite to Workspace**: Any user that is part of a workspace can invite other users to the workspace
10. **Send a Chat Message**: A user can send chat messages to other users or the workspaces that they are part of. A chat message is a new note that is sent with notification to the users involved.
11. **Update Workspace**: The workspace manager can update workspace metadata, like title, descriptions, etc.
12. **Leave Workspace**: A user can leave any workspace that they are part of.
13. **Delete Workspace**: The workspace manager can delete the workspace and all associated data
14. **Ban users**: The workspace owner can ban a user, kicking them out and preventing them from joining in the future.

- Use cases for feature 4: Customize Format

15. **Create Template**: A user can create a note template, consisting of components like title, tags, description(s), and custom fields like "Due date" for a note template. A note template can be created from an existing note or directly.
16. **Update Template**: A user can update their templates, editing the components.
17. **Delete Template**: A user can delete their templates, and will not be able to use it for future notes.

...

3.5. Formal Use Case Specifications (5 Most Major Use Cases)

Use Case 1: [Create a Note]

Description: App user is logged in and creates notes by filling in a chosen note template, adding or removing fields if necessary. After filling in the fields, the user can choose which workspace to place it in.

Primary actor(s): User

Main success scenario:

1. User clicks the “Create Note” button
2. System displays a default empty template (a text field plus tags), the user has an option to switch to other templates via a template menu (if such templates are available)
3. User inputs all details of the note into the fields
4. User can add additional fields or remove fields from the template. User is physically unable to remove the tag field, or the only remaining content field in the template.
5. User clicks “Create” button
6. The system creates the note with the filled in data and stores it in the database, and displays a confirmation message.

Failure scenario(s):

- 5a. All fields of the to-be-created notes are empty
    - 5a1. system displays a warning message asking the user to confirm the note creation
- 6a. The note could not be created
    - 6a1. System displays error message stating that the note could not be created as well as the reason for the failure (e.g. connection lost)

Use Case 2: [Search for Notes]

Description: The user searches for a note that matches their inputted prompt, returning a list of matching notes

Primary actor(s): User

Main success scenario:

1. User clicks the "Search for a Note" button
2. System displays a text input field
3. User types in their query and clicks the "Search" button
4. System fetches a first page of notes with matching keywords from the query and displays them. In the background, the other pages of search result are completed

Failure scenario(s):

- 4a. No matching notes
    - 4a1. System displays an error message stating there were no notes that matched the query provided.
- 4b. Available notes not fetched
    - 4b1. System displays an error message stating the error code and reason the fetch failed, such as connection lost.

Use Case 3: [Create A Note Template]

Description: The user creates a note template by adding and deleting components to their desire (e.g. headers, text fields, attachments, links to other notes, datetimes).

Primary actor(s): User

Main success scenario:

1. User clicks the "Create Note Template" button
2. System displays a default note template, along with buttons to create new fields or delete fields
3. User customizes the note template to their desire by adding/removing/setting default values/moving the input fields around the space available. The tag field is not removable and there must be at least one content field in the message, else the user cannot remove fields.
4. User clicks the "Create" confirmation button
5. System saves the note template and displays a confirmation message

Failure scenario(s):

- 5a. Creation of note template failed (server-side)
  - 5a1. System displays error message stating the reason and tells the user to retry
- 5b. Creation of note template failed (user-error)
  - 5b1. System displays error message, pointing to areas in the template that the user may need to fix

Use Case 4: [Create Workspace]

Description: User creates a workspace, effectively becoming the workspace manager

Primary actor(s): User

Main success scenario:

1. User clicks the "Create Workspace" button
2. System displays input fields needed to create a new workspace (e.g. name)
3. User fills in the required field: name
4. A create button becomes available to the user
5. User can add optional fields, such as description, initially invited accounts or profile picture
6. User clicks the "Create" confirmation button
7. System receives the input and creates the corresponding workspace
8. System displays the newly created workspace to the user and sets the user as the Workspace Manager

Failure scenario(s):

- 3a. Workspace name entered by the user is already taken by another workspace
  - 3a1. Front End system displays error message containing possible suggestions of similar but not claimed yet names and highlights the name field in red.
- 5a. One of the invited email addresses is invalid.
  - 5a1. The system highlights it in red with an error message and the user cannot create the workspace prior to removing or correcting the address
- 5b. Could not upload the profile picture.
  - 5b1. The system displays an error message stating the reason for failure and the profile picture is not updated.

Use Case 5: [Send a Chat Message]

Description: User sends a chat message in the workspace's chat forum.

Primary actor(s): User or Workspace Manager

Main success scenario:

1. User clicks into any of the workspaces they are in or a previous chat with another account

- 1.1. A user can also start a conversation with a new account for which they will need to enter its associated email address in a separate form

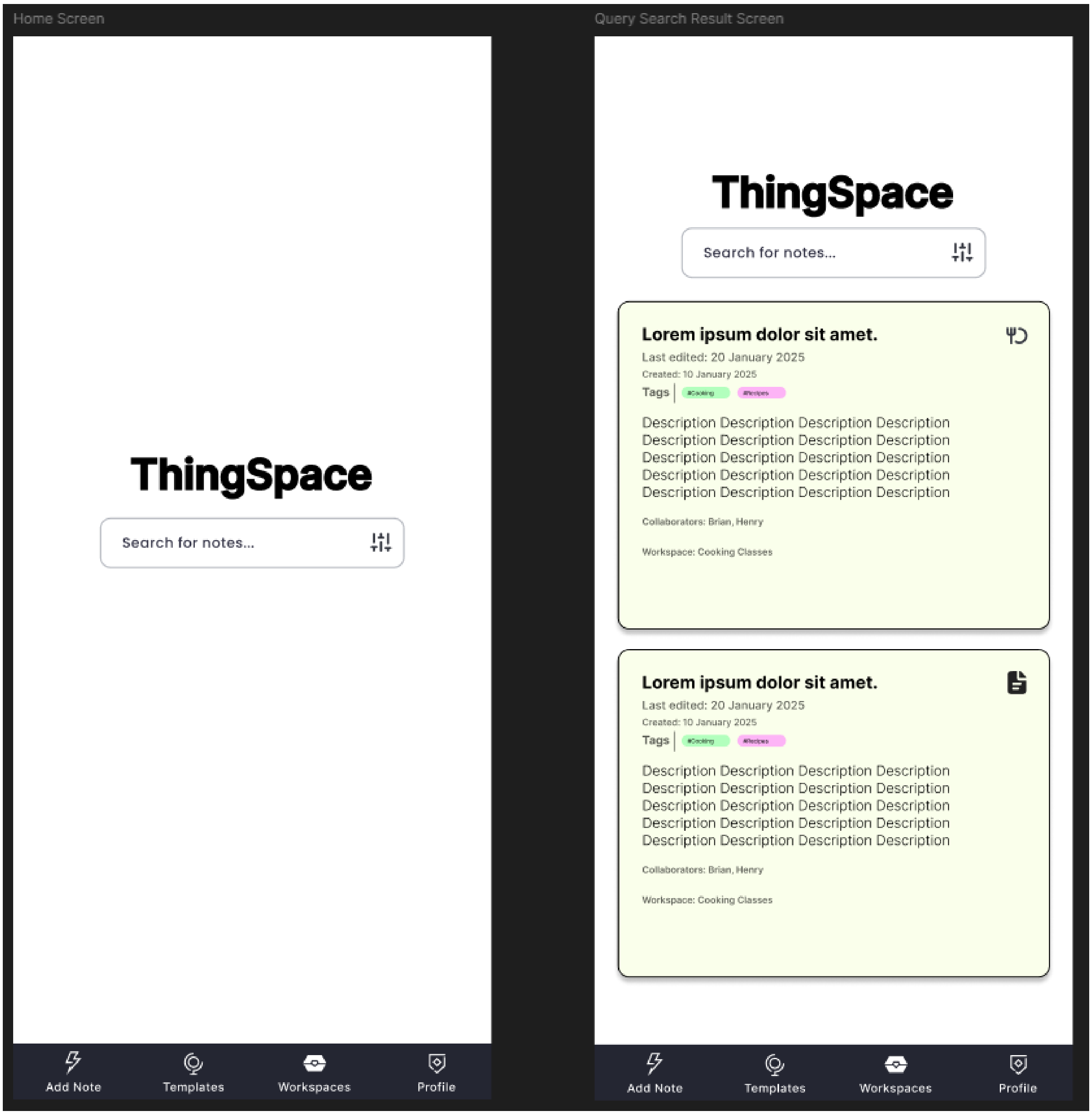
2. System displays the workspace/conversation
3. User clicks the "Chat" button

4. User is taken to an altered version of note creation screen with the default text field template that spans the lower portion of the screen, while the upper part displays previous messages on the chat. User can expand and contract the note creation part, as well as change the template.
5. User performs a note creation with the template of their liking
6. System adds the note to the workspace/conversation and displays the notes with their content expanded and sorted as latest notes at the bottom of the page. The chat notes have an additional tag "chat@<workspace_name>"
7. Users in the workspace or the other user in the conversation receive a push notification that the message is sent.

Failure scenario(s):

- 1.1a. The email entered is invalid -1.1a1. The system displays an appropriate error message and cannot proceed with creating the conversation before the email is corrected.
- 2a. User does not belong to any workspace, nor has any conversations
  - 2a1. System displays “Create a workspace” and "start a conversation" buttons, allowing for a user to create a workspace which is needed to send chats
- 6a. The message couldn’t be sent
  - 6a1. System displays error message indicating the error code and reason
  - 6a2. User tries to send the message again after fixing the error (ex. connection error)

3.6. Screen Mock-ups



Lorem ipsum dolor sit amet.
Last edited: 20 January 2025
Created: 10 January 2025
Tags | #Cooking #Recipes

Description Description Description Description
Description Description Description Description
Description Description Description Description
Description Description Description Description
Description Description Description Description

Collaborators: Brian, Henry

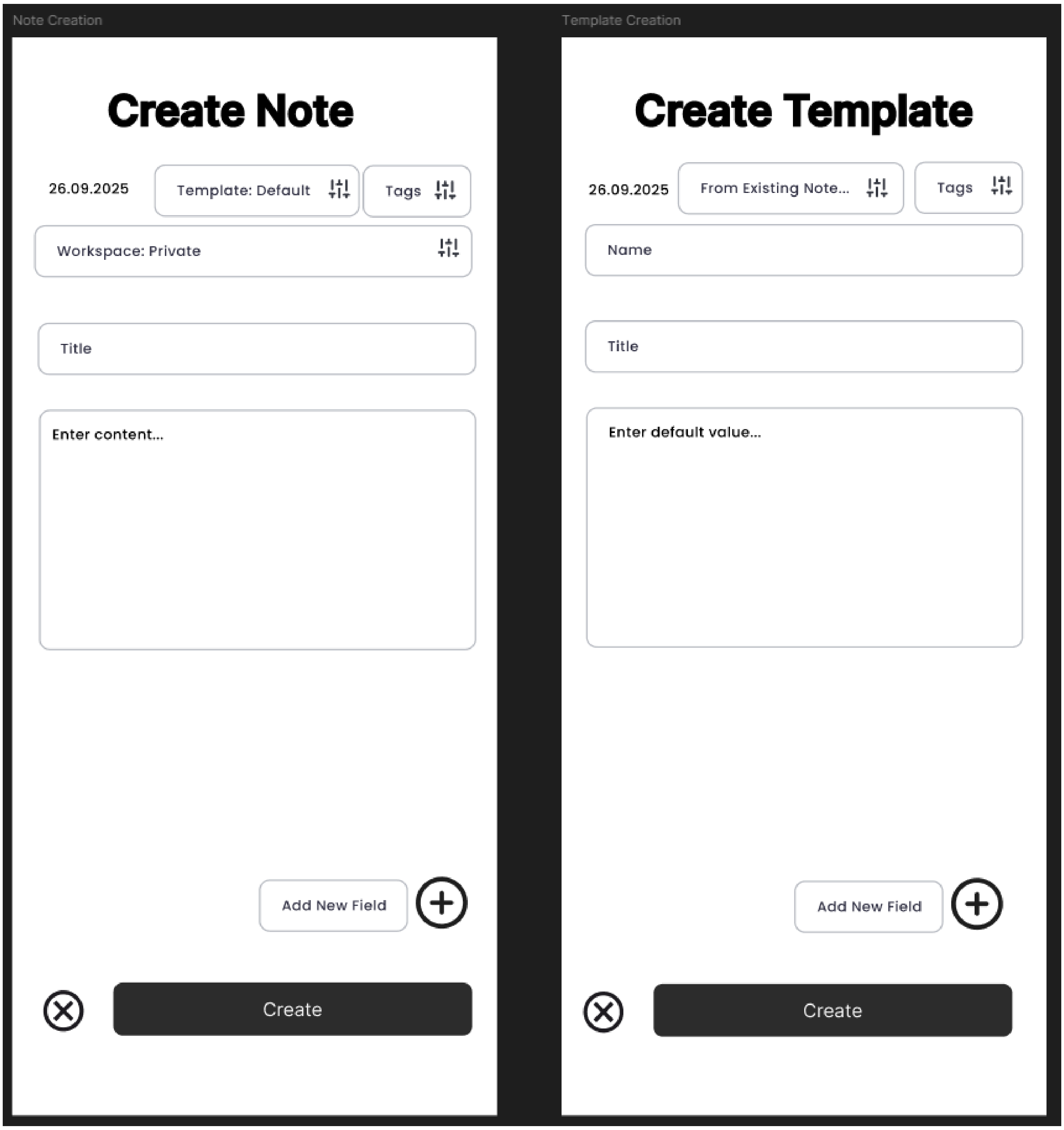
Workspace: Cooking Classes

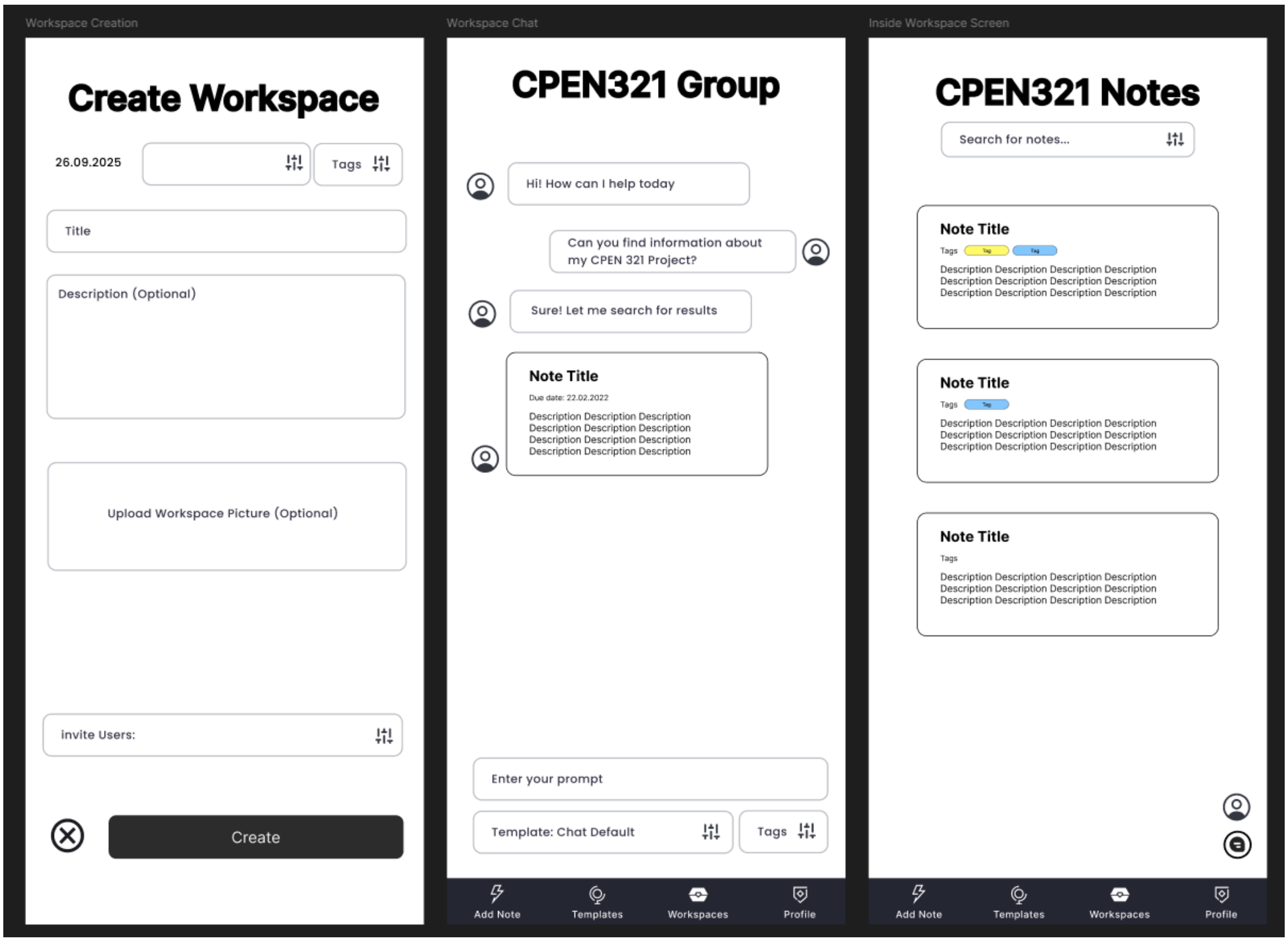
Lorem ipsum dolor sit amet.
Last edited: 20 January 2025
Created: 10 January 2025
Tags | #Cooking #Recipes

Description Description Description Description
Description Description Description Description
Description Description Description Description
Description Description Description Description
Description Description Description Description

Collaborators: Brian, Henry

Workspace: Cooking Classes


Note Creation
Create Note
26.09.2025
Template: Default
Tags
Workspace: Private
Title
Enter content...
Add New Field
Create
Template Creation
Create Template
26.09.2025
From Existing Note...
Tags
Name
Title
Enter default value...
Add New Field
Create



Hi! How can I help today

Can you find information about my CPEN 321 Project?

Sure! Let me search for results

Note Title
Due date: 22.02.2022
Description Description Description Description Description Description Description Description Description Description Description Description

Note Title
Tags
Description Description Description Description Description Description Description Description Description Description Description Description

Note Title
Tags
Description Description Description Description Description Description Description Description Description Description Description Description

Note Title
Tags
Description Description Description Description Description Description Description Description Description Description Description Description

3.7. Non-Functional Requirements

1. **[Feature Accessibility]**
   - **[Description]**: Any workspace of conversation the user is a member of has to be accessible to the user within two clicks from the main screen.
   - **Justification**: Several messaging/file sharing applications, such as Discord, have every conversation reachable with maximum of 2 clicks (Discord server - specific channel, or discord dms - particular chat; WhatsApp community - particular chat). The app will be feature-rich, however it should still be competitive wrt. usability.

2. **[Searching Speed]**
   - **Description**: Producing a page of synonymic search result should last no longer than 5 seconds
   - **Justification**: Synonymic searching includes calling an API to check for synonyms. While this process can take time and fitting below a second of response time is unlikely (at least not guaranteeable before actual tests with the API), we should not get close to the 10 seconds response limit mentioned in https://www.nngroup.com/articles/response-times-3-important-limits/ The 10 seconds is the user attention limit, i.e. the time the user is said to be willing to wait without attempting to focus on other tasks. As we envision synonymic search being used frequently, getting close to this limit on regular basis would mean straining the user attention, hence we impose a safety factor of 2. One might point out that while the note database gets larger, there is more notes to search and more matches, hence the response time shall increase. This is why the requirement only concerns itself with one page of search results. While the note

base increases, we would get more direct matches, and the first page could get populated with those ones while the app is looking for less direct matches in the background.

3. **[Filtering Speed]**

    - **Description**: Updating the display with a page of filtering (by tag or creation/last edit) results should take no longer than 1 second.
    - **Justification**: This is so that the user is not significantly disturbed by waiting for the resonse, as mentioned in https://www.nngroup.com/articles/response-times-3-important-limits/ Again, while with increasing number of notes, the response time shall increase as well, filtering done by this app is an O(n) operation, and does not disturb the sorting of the data. More importantly, only a part of results that fit the filter have to be created. This is again, because the requirement only concerns itself with one page of results at a time, which is what the user will see.

4. Designs Specification

4.1. Main Components

1. **Users**

    - **Purpose**: Manages all functionality relating to users, including creation, tracking metadata, etc. Users are a good component as each user must store some of their own data.

2. **Notes**

    - **Purpose**: The notes component manages all note items. This includes creation, processing, and search retrieval. This functionality can be effectively bucketed together, and other components can interact with these notes, maintaining a separation of concerns.

3. **Workspaces**

    - **Purpose**: The workspace contains its own general information, as well as a reference to member users and included notes. This component would be responsible for forwarding push notifications to member users, and the banning functionality.

4. **Templates**

    - **Purpose**: Contains Template data plus handles the automatic generation of note creation forms from the template.

4.2. Databases

1. **[MongoDB]**
    - **Purpose**: Storing user data, their notes and the workspaces they are in. Since some of the note data can be customised, it does not have to follow exactly the same format. As such, a more flexible non-relational database like MongoDB is preferred over relational ones like MySQL.
2. **[Vector database like FAISS or Pinecone (Optional)]**
    - **Purpose**: We convert notes to vector embeddings in order to do vector similarity for search. For performance reasons, we may want to use a vector database. This is optional, however, and will depend on the performance of our app.

4.3. External Modules

1. **[OpenAI or equivalent]**
   - **Purpose**: We will be using the OpenAI client to create vector embeddings of our notes. These embeddings will be used for KNN or similar for matching queries. This can be used alongside lexical matching algorithms like BM25. Note: we may need to use a vector database like FAISS or Pinecone, but that's probably not necessary for a small scale application like ours.
2. **[OpenAI Whisper or equivalent (Optional)]**
   - **Purpose**: As an optional functionality, we want to allow speech input for creating notes. We will use OpenAI Whisper or other speech to text models to do this.

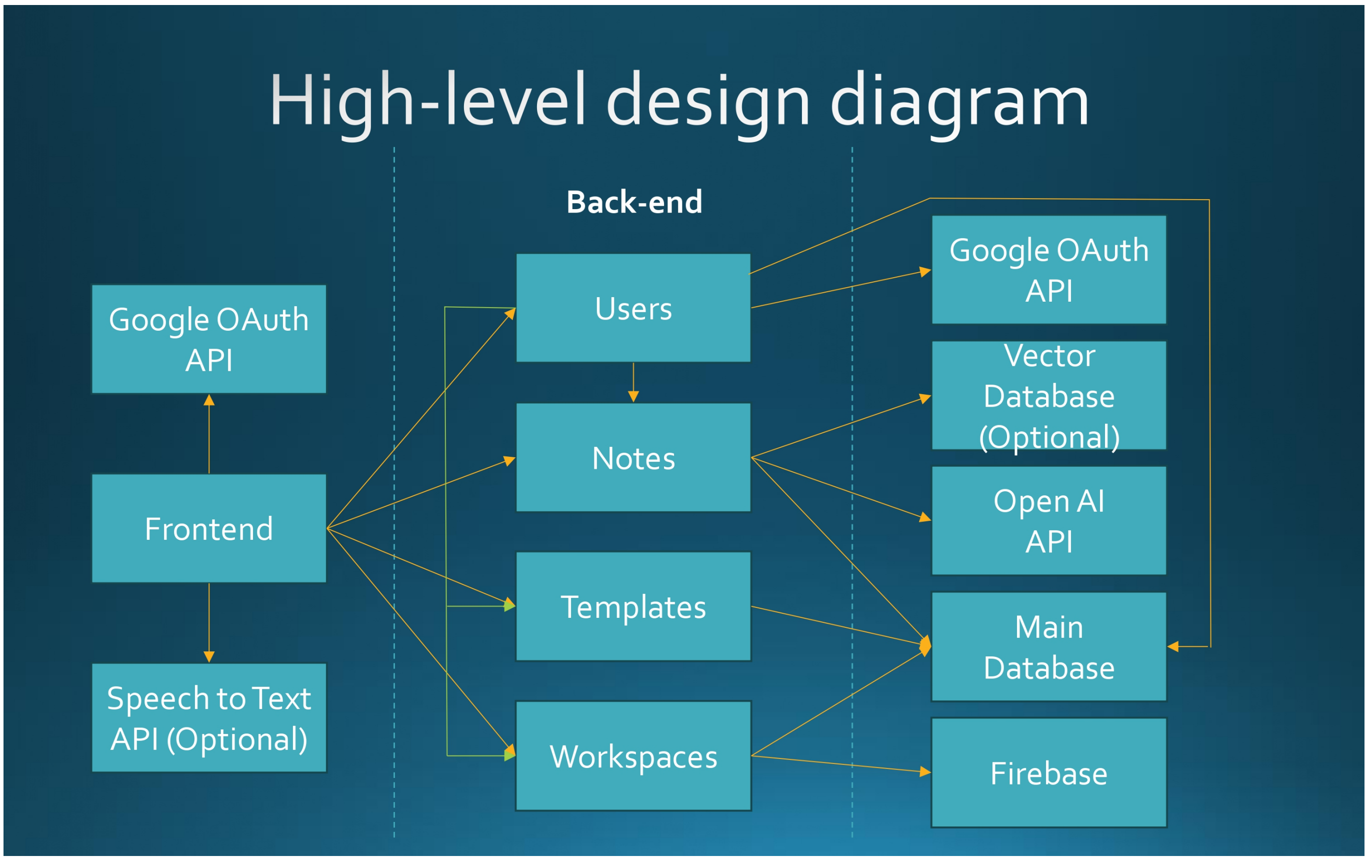
4.4. Frameworks

1. **[Firebase]**
   - **Purpose**: Push notification support
   - **Reason**: Firebase is allowed for push notifications, and can significantly decrease complexity of implementing them.

Libraries:

1. **[Retrofit]**
   - **Purpose**: Managing API calls and frontend-backend connection
   - **Reason**: Retrofit is already implemented in M1, and it is much neater to use interfaces created by it than making direct HTTP calls from the frontend.

4.5. Dependencies Diagram



The dependency of Users on interfaces from other components is because user deletion. When a user gets deleted, Users have to notify all other modules to remove all notes, templates and workspaces associated only with the user being deleted and make the user no longer an active member of any workspace they were in.

4.6. Use Case Sequence Diagram (5 Most Major Use Cases)

1. **[WRITE_NAME_HERE]**
   [SEQUENCE_DIAGRAM_HERE]
2. ...

4.7. Design and Ways to Test Non-Functional Requirements

1. **[WRITE_NAME_HERE]**
    - **Validation**: ...
2. ...