CrowdLib Midpoint Report

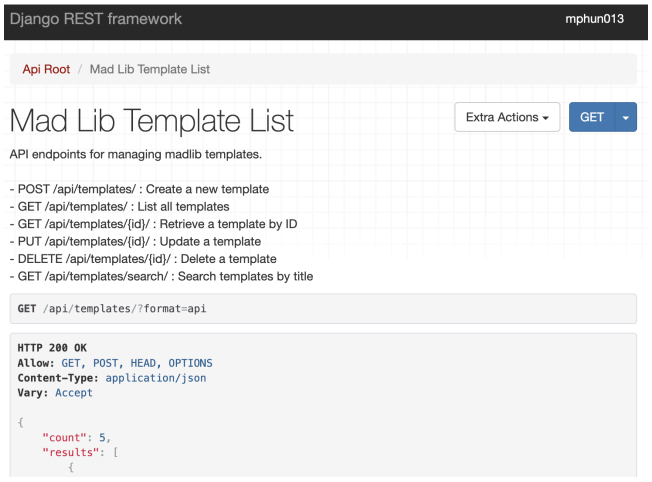
Github: apaya22/CrowdLib

Team: Team Blue

Members: Joshua Kibreab, Andy Payan, Max Phung, Shane Zimmerman

Functionalities, components, classes implemented:

- Implemented GoogleOAuth login, signup, and profile editing, such as profile picture, username, bio.
- API classes for madlibs, users:
  - Ability to create MadLibs
  - Ability to search for existing Madlibs
- Pages: Home, Explore, Create, Profile, Login, Signup



Mad Lib Template List

API endpoints for managing madlib templates.

- POST /api/templates/ : Create a new template
- GET /api/templates/ : List all templates
- GET /api/templates/{id}/ : Retrieve a template by ID
- PUT /api/templates/{id}/ : Update a template
- DELETE /api/templates/{id}/ : Delete a template
- GET /api/templates/search/ : Search templates by title

```
GET /api/templates/?format=api
```

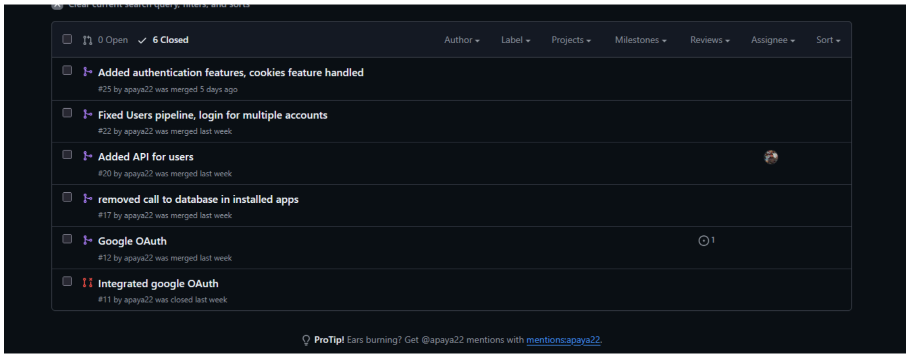
```
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "count": 5,
    "results": [
        {
```

- Reusable Components: NavBar, PostCard/FeatureCard, MadLibCard, and components for user input/forms.

Who did what:

Andy

Implemented Google OAuth login flow, created login redirect, added MongoDB sync pipeline (pipeline.py), built debug dashboard, created Users API foundation and URL routing for user endpoints, integrated environment variable system, and ensured Django + MongoDB user data stay synchronized. I will implement social APIs for storing likes and comments, as well as integrating an AWS bucket for storing AI-generated images for when we implement that feature. I will continue working on the backend, fixing any bugs relating to APIs, authentication, image storing, etc.. The backend is built in Python using Django and Django REST Framework, so I coded mostly using Python for the backend code I helped program.



Added authentication features, cookies feature handled
#25 by apaya22 was merged 5 days ago

Fixed Users pipeline, login for multiple accounts
#22 by apaya22 was merged last week

Added API for users
#20 by apaya22 was merged last week

removed call to database in installed apps
#17 by apaya22 was merged last week

Google OAuth
#12 by apaya22 was merged last week

Integrated google OAuth
#11 by apaya22 was closed last week

ProTip! Ears burning? Get @apaya22 mentions with mentions:apaya22.

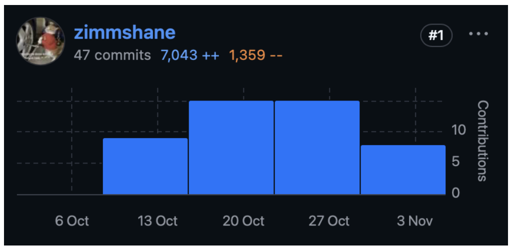
- found out Django handles session IDs, cookie saving. Make sure a logged-in user can only delete/see/edit their own profile. Added admin privileges for certain APIs to protect users' info.

- Made a home view for backend server, lists available endpoints

- Pipeline file creates a mongodb user when someone logs in with googleOAuth, as well as ensuring there isn't a duplicate account made by checking the database and comparing the accounts

- Made a foundation for views.py, lists debug endpoints for googleOAuth information, userviewset to list all users that have an account, create user, retrieve user info by userID, update a users profile for usernames, bios, profile picture, deleting a user, retrieve user public info by username. Each endpoint has its own security feature, separating privileges for authenticated users, admins, etc..

Shane Zimmerman

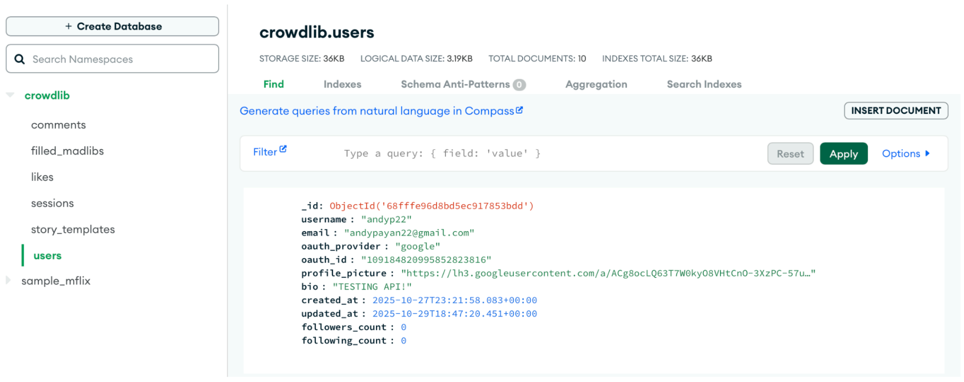


- Initialized project dependencies (Django, React, Pymongo)

- Initialized MongoDB Atlas for text storage.

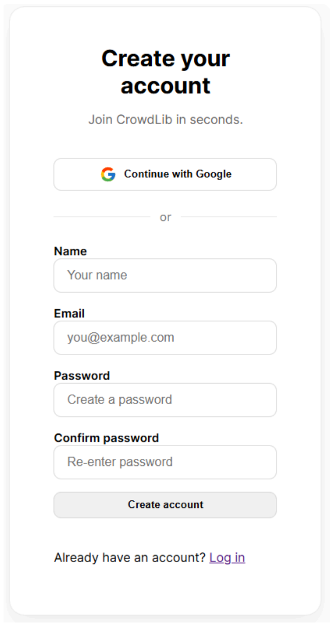
- Wrote custom database loading logic using Pymongo to make MongoDB working with Django.

- Wrote custom logic to get Django to use MongoDB for storing user sessions.
- Wrote model classes: UserOperations, MadlibTemplates and UserFilledMadlibs to handle controlling access and updates to the database.
- Wrote API view classes: MadlibTemplateViewSet, UserFilledMadlibsViewSet, UserViewSet.
- Created Architecture and class diagrams



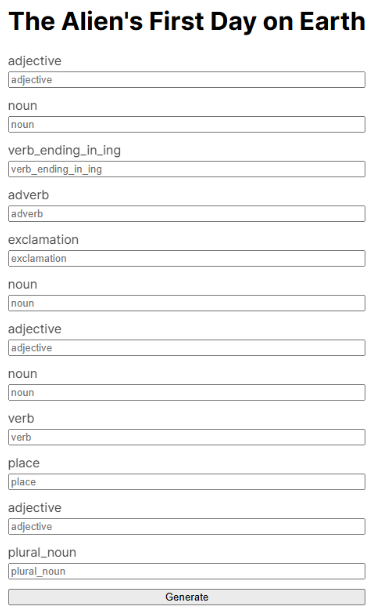
```
_id: ObjectId('68fffe96d8bd5ec917853bdd')
username : "andyp22"
email : "andypayan22@gmail.com"
oauth_provider : "google"
oauth_id : "109184820995852823816"
profile_picture : "https://lh3.googleusercontent.com/a/ACg8ocLQ63T7W0kyO8VHtCnO-3XzPC-57u…"
bio : "TESTING API!"
created_at : 2025-10-27T23:21:58.083+00:00
updated_at : 2025-10-29T18:47:20.451+00:00
followers_count : 0
following_count : 0
```

(View of database in MongoDB Atlas, showing an Andy's test account entry)

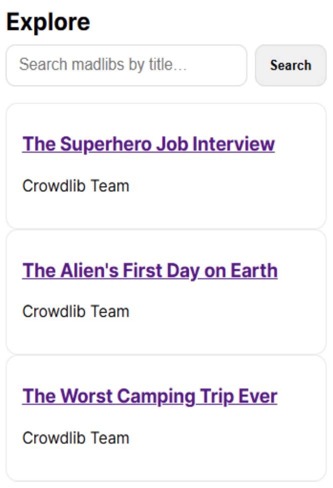


Joshua Kibreab

- Implemented the Explore page to dynamically fetch and display a list of all created Madlibs.
- Integrated MadlibPlay to render individual madlib templates and collect user input.
- Ensured madlib template could only be generated when user provides input where needed.
- Configured OAuth redirect handling for Google signin and signup.



- Designed signup page. Confirms if the user provided input.
- Called APIs from the frontend state to display the created madlibs.
- Updated Frontend class and Hierarchy diagram
- Planning to Generate a displayable Madlib when User provides proper input



Explore

Search madlibs by title...

Search

The Superhero Job Interview

Crowdlib Team

The Alien's First Day on Earth

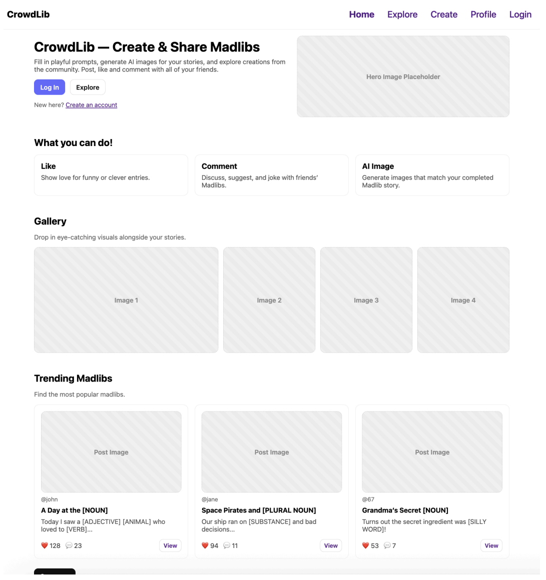
Crowdlib Team

The Worst Camping Trip Ever

Crowdlib Team

Max Phung:

- Built UI and frontend components designed through React+Vite.
- Wrote design for the Webapp and all of the pages with simple components such as a navbar and cards.
- Coding style includes use of function statements to design the interaction for the pages and what each component will do, such as button functions and search bars.



CrowdLib

Home Explore Create Profile Login

CrowdLib — Create & Share Madlibs

Fill in playful prompts, generate AI images for your stories, and explore creations from the community. Post, like and comment with all of your friends.

Log In Explore

New here? Create an account

Hero Image Placeholder

What you can do!

Like
Show love for funny or clever entries.

Comment
Discuss, suggest, and joke with friends' Madlibs.

AI Image
Generate images that match your completed Madlib story.

Gallery

Drop in eye-catching visuals alongside your stories.

Image 1 Image 2 Image 3 Image 4

Trending Madlibs

Find the most popular madlibs.

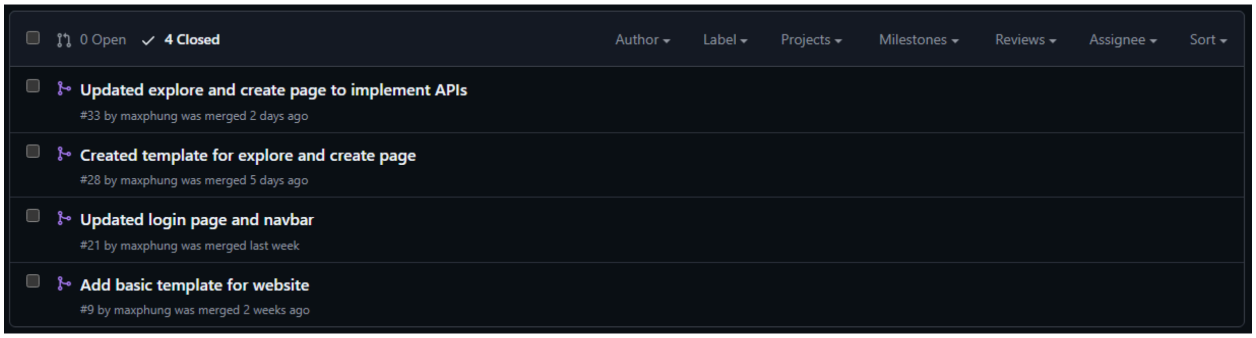
Post Image
@john
A Day at the [NOUN]
Today I saw a [ADJECTIVE] [ANIMAL] who loved to [VERB]...
❤️ 128 💬 23 View

Post Image
@jane
Space Pirates and [PLURAL NOUN]
Our ship ran on [SUBSTANCE] and bad decisions...
❤️ 94 💬 11 View

Post Image
@67
Grandma's Secret [NOUN]
Turns out the secret ingredient was [SILLY WORD]!
❤️ 53 💬 7 View

- Then, the layout of the pages are composed of html-style code within each .jsx file.
- Design/style of each page are completed through .css styling, defining all of the components with sizing, color, positioning, etc.
- Wrote calls to our api inside the .jsx files for the "Explore" and "Create" page to display backend information such as templates and madlibs.
- Currently working on functionality for the api calls to allow users to make madlibs and search for madlibs using GET and POST calls to the frontend.

- Will continue working on the UI side, getting our features to work on the frontend and communicate properly with the backend and api.



0 Open ✓ 4 Closed

Updated explore and create page to implement APIs
#33 by maxphung was merged 2 days ago

Created template for explore and create page
#28 by maxphung was merged 5 days ago

Updated login page and navbar
#21 by maxphung was merged last week

Add basic template for website
#9 by maxphung was merged 2 weeks ago

Plan to complete project

Our plan to complete the project is to finish implementing the rest of the core functionality, including supporting likes/comments, an explore page, sharing completed Madlibs entries and AI generated images, as well as actually implementing the AI image generation feature. We will be adamant about debugging the core features we have completed and using tests to verify the correctness of each feature. We will be polishing the frontend to make the web-app look clean while doing what we can to boost user experience.

Testing

We plan to test our project using both automated tests (using Django REST Framework's built-in APITestCase) and manual API testing curl commands. Shane and Andy will focus on backend testing, while Max and Josh will work with frontend testing.

Each module (Users, Madlibs, Social) has its own test plan:

- **Users API:** We have test cases for testing the users API, including creating an test creating a new user with OAuth, updating a user profile including their bio and profile picture, deleting a user by creating a dummy account and having that fake user use the delete profile functionality, and testing an edge case for duplicate usernames, where we create two dummy users with the same username and checking to see if it allows the creation of two users with the same name.

- **Madlibs API**: We test creating new Madlib templates, filling out templates as users, retrieving completed entries, and validating that unauthorized users cannot submit or delete entries.

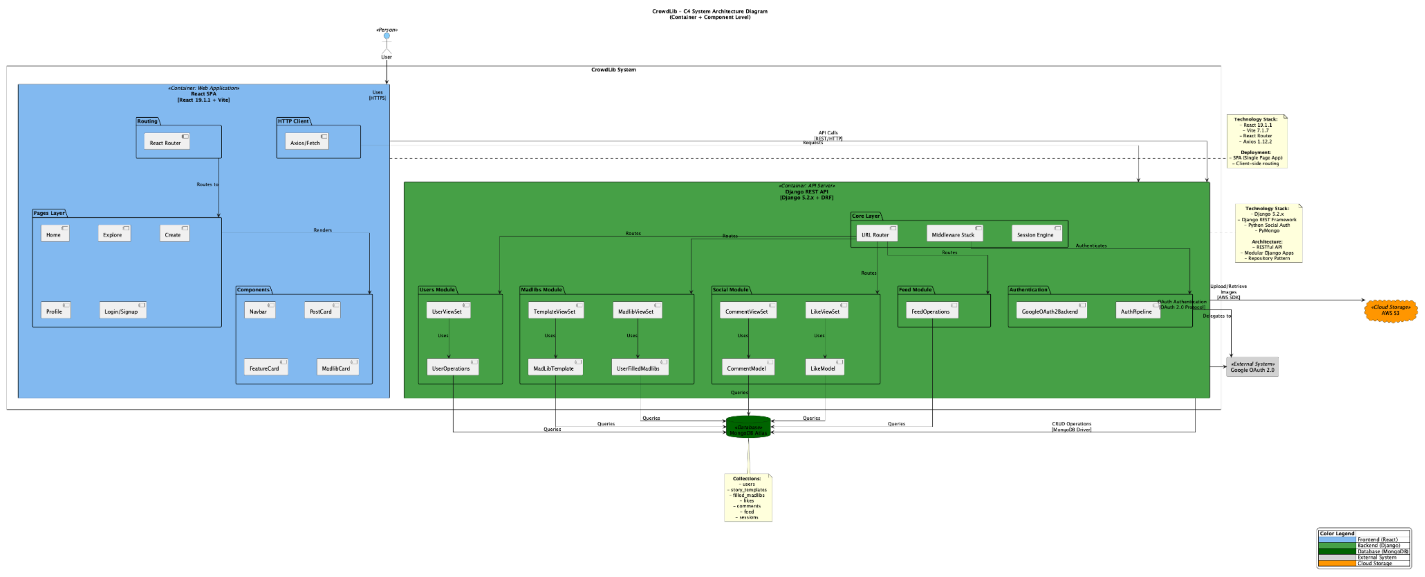
- **Social API** (Likes & Comments): We will test adding likes and comments to posts and ensure that each user can only interact once per post or delete their own comments.

- **Authentication:** We verify that only authenticated users can modify data, while public users can view shared content.

- **Frontend:** The frontend is made to just display all the information from the backend, so our focus for testing the backend will be to manually verify each page is up to our standards in terms of aesthetics, proper displays for the information, and responsiveness/performance. We will make sure that the frontend makes proper calls to the backend with no major bugs.

Diagrams

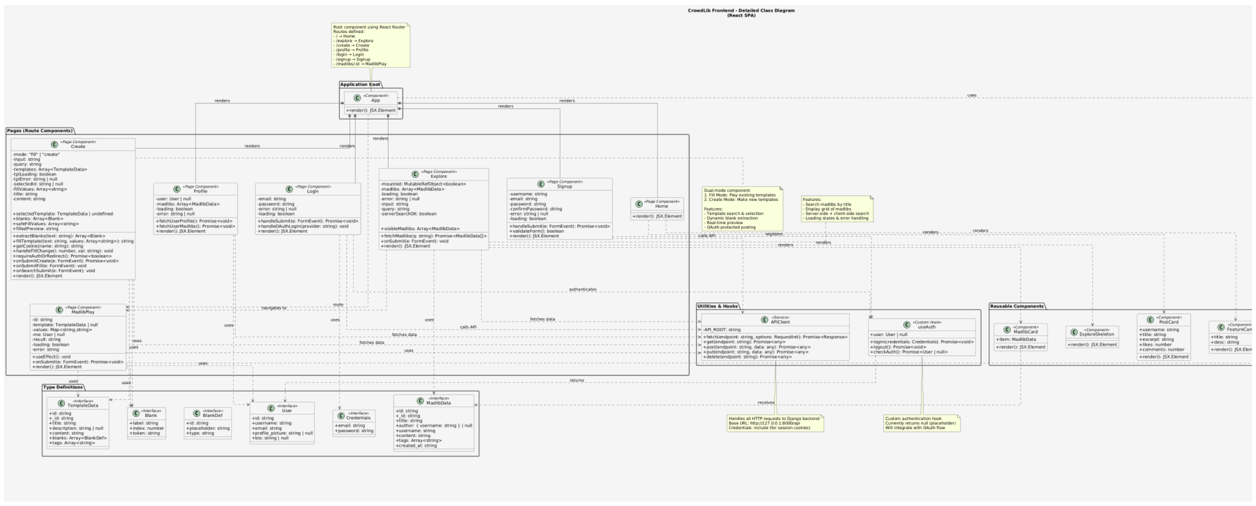
Architecture



This diagram shows how the Frontend User interacts with the Django REST API. The API routes requests (Core & Routing) through authentication to specific modules (User, MadLib, Social), which then use a Data Layer to interact with MongoDB to collect the needed data.

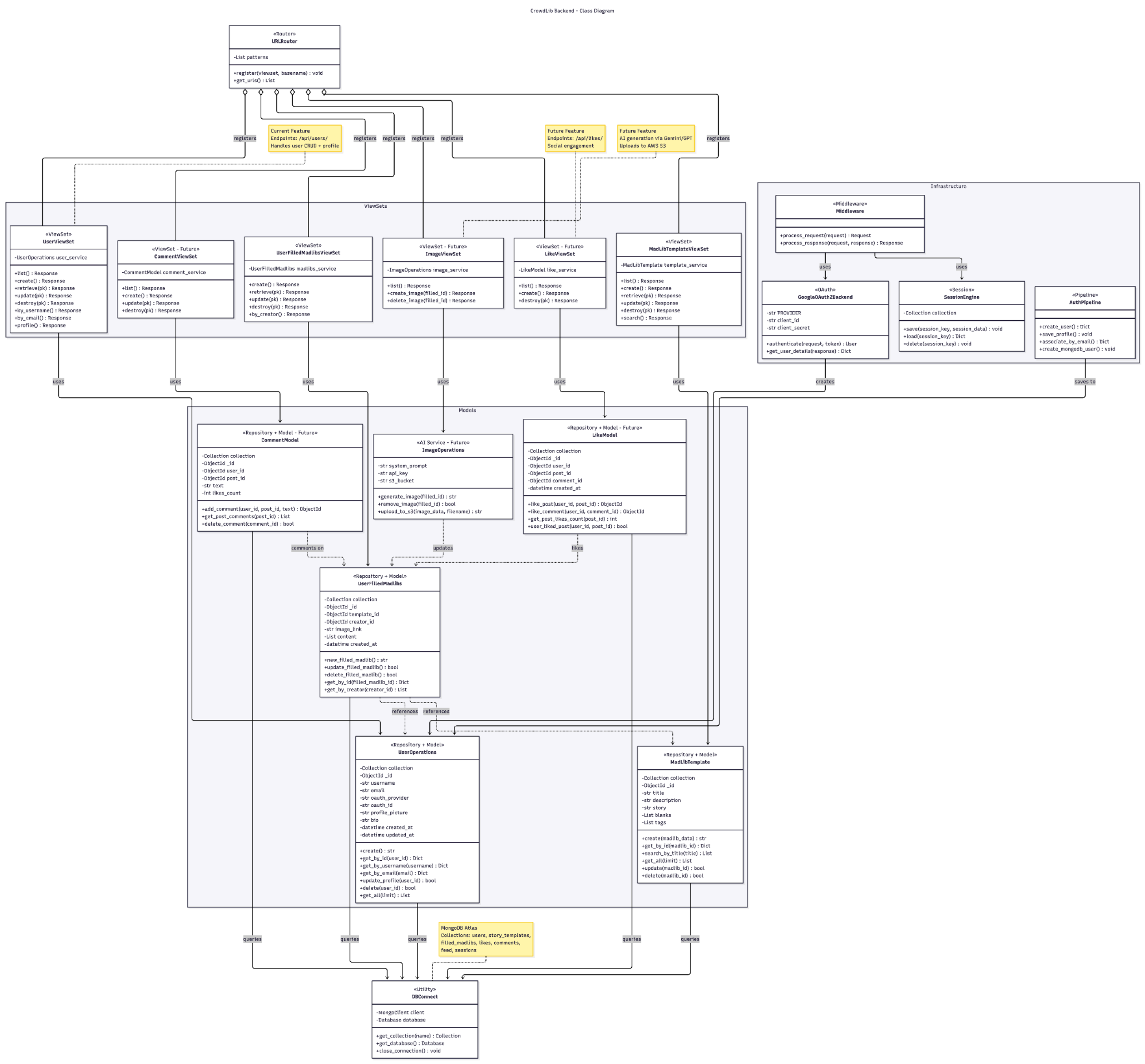
Class

Frontend



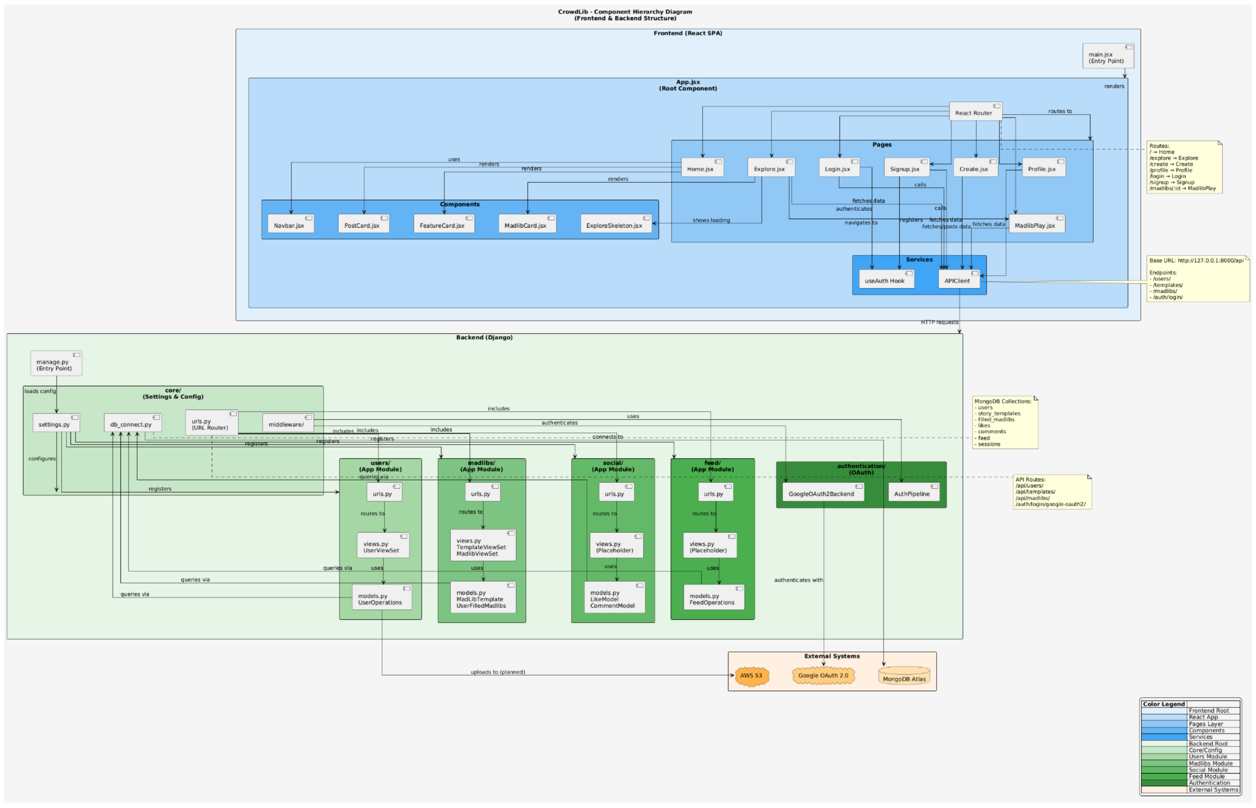
This is a React frontend class diagram for CrowdLib. It illustrates the component hierarchy, starting from the root App component which renders various Pages (Home, Profile, Create, Explore). These pages are built using Reusable Components (like NavBar, Card) and Utilities & Hooks for API calls and auth. The diagram also specifies the props, state, and interfaces (Type Definitions) for the application.

Backend



The CrowdLib backend uses Django REST Framework and MongoDB in a modular architecture that separates API logic, database operations, and authentication. ViewSets like UserViewSet and MadlibTemplateViewSet handle users and madlibs API endpoints, while classes like UserOperations and MadlibTemplates manage MongoDB interactions. The system also uses Google OAuth authentication, session management, and is designed to support future features like AI-generated image uploads and social interactions (likes/comments).

Component Hierarchy



This diagram shows how the file and module structure for both the CrowdLib frontend (React) and backend (Django) are composed. The frontend half shows the App.jsx root routing to Pages, which are built from Components and use Services. The backend half shows how the main urls.py requests to different Django apps such as users, madlibs, which each app then has its own models and views. The information is all stored in the mongodb database, in which there is communication between the DB and the backend to display the needed information requested from the frontend.

Extra Credit:

Project Design:

- Our design flow allows us to work collaboratively because it takes two major parts of web development: frontend and backend. Therefore, there is a lot of communication between our members to work on functionality between our database and user interface. Splitting features between us allows us to also work on different parts of our project independently. For instance, one member works on the log-in feature with GoogleOAuth, and another member can work on other features such as user Madlib creation. Github is used allowing us to track all changes and work collaboratively, minimizing conflicts between each member. Using React + Vite in the frontend allows a structure for different members to use when working on the UI. Each .jsx file for the frontend are all structured the same with function calls and html styling. So when changes are to be made, it is easy to navigate through all changes.

Crowdlib Extra Credit Midpoint Demo:

https://youtu.be/UrGbdqUhySY?si=nosMMJF8xrRGJHKJ