PlanningJam

Iteration 1

BU MET 673. Fall 2025 OL. Team 3

Team

David Metraux - Team Leader

Haolin Yang - Design and Implementation Leader

Donjay Barit - Configuration Leader

Ashley Sachdeva - Requirements Leader

Jason Lee - QA Leader

Table of Contents

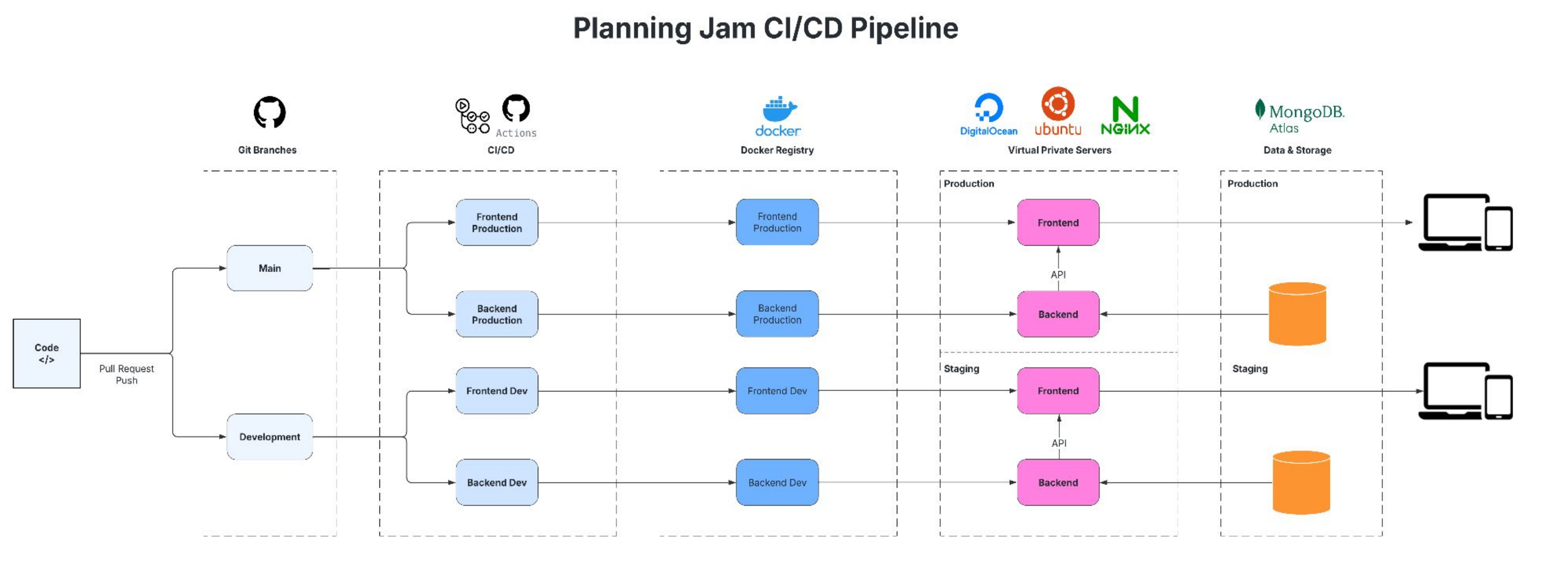
- Summary
- SDD Document
- Future Plans

Iteration 1 Summary

Iteration 1 Summary

- Backend: We implemented registration and login functionalities using JWT.
- Frontend: Implemented registration, login and a simple main page.
- Set up CI/CD Pipeline
- Set up Github Security

CI/CD Pipeline



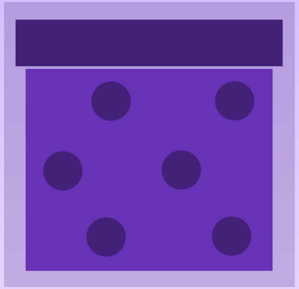
Moving from Render

We decided to move from Render to Digital Ocean due to constraints that the Render plan provides.

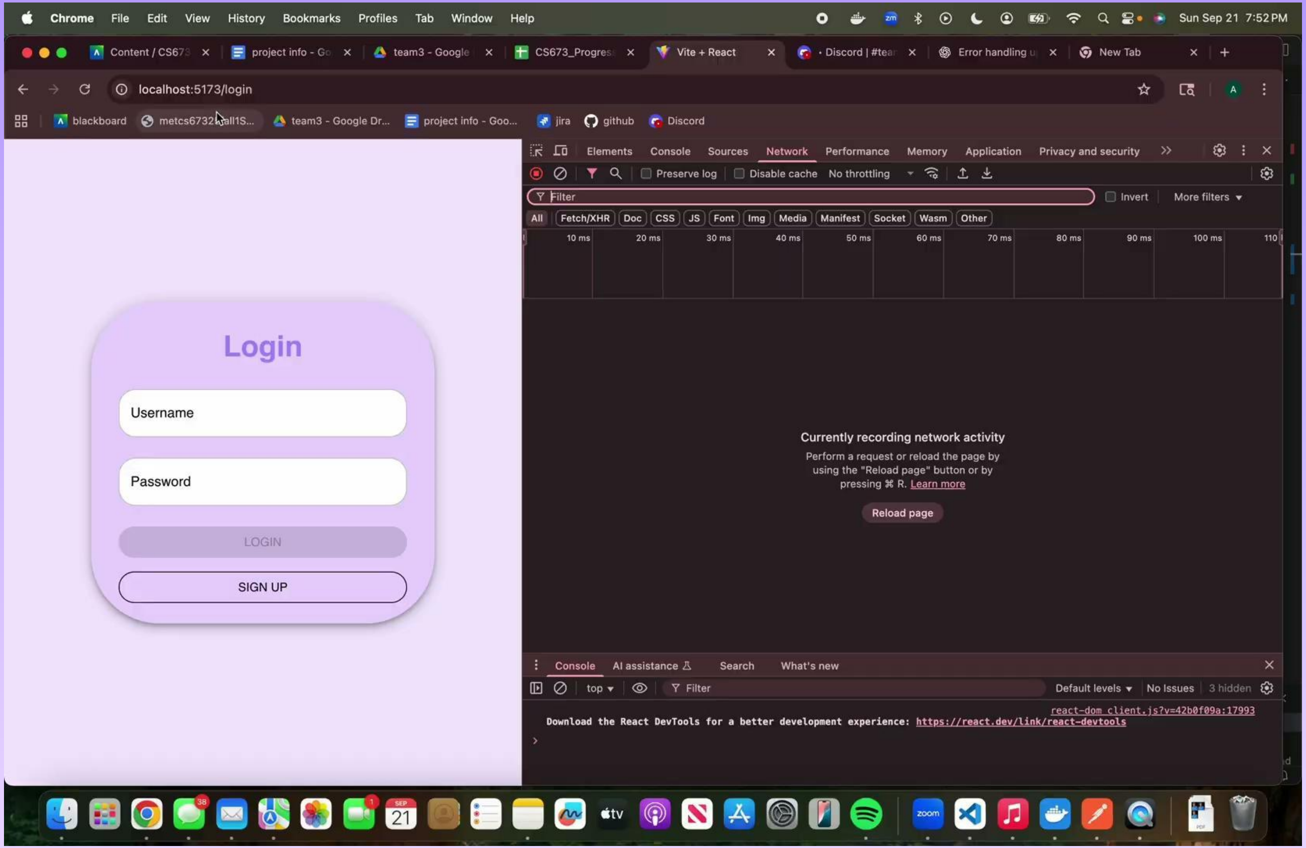
- **Why?**

  As our application and database grow with each iteration, Render no longer provides the sufficient hardware environment to run the entire application.

Hence, we are moving the hosting of the application to Digital Ocean instead.



Demo



Currently recording network activity

Perform a request or reload the page by using the "Reload page" button or by pressing ⌘ R. Learn more

```
react-dom_client.js?v=42b0f09a:17993
Download the React DevTools for a better development experience: https://react.dev/link/react-devtools
```

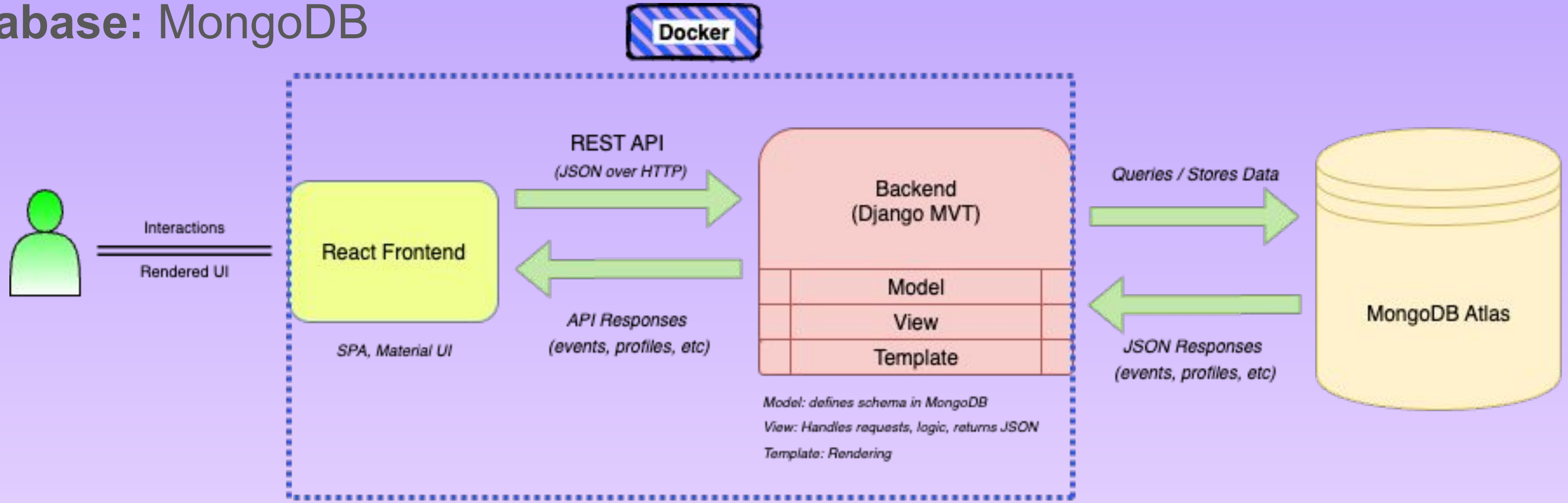
SDD Document

Software Architecture

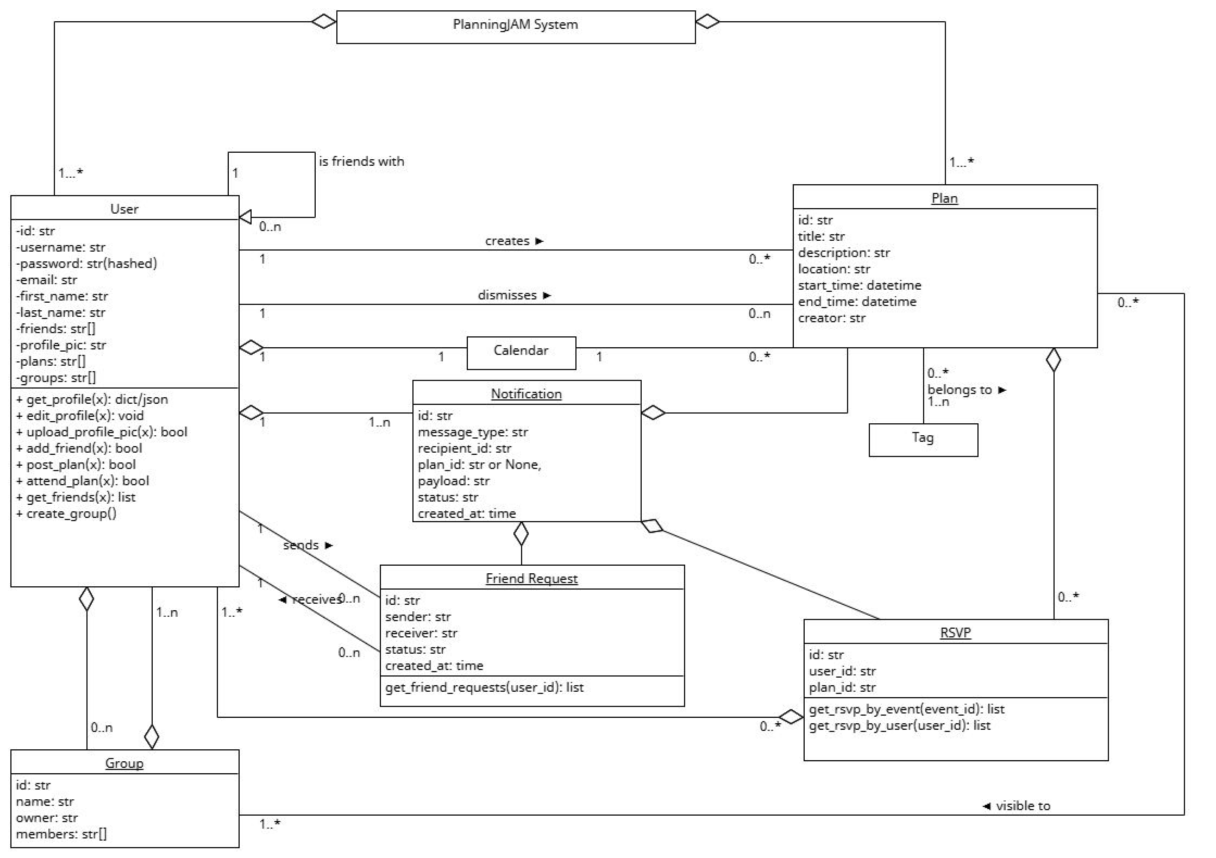
Front End: React/Vite

Back End: Django/Python

Database: MongoDB



Class Diagram



PlanningJAM System

User

-id: str
-username: str
-password: str(hashed)
-email: str
-first_name: str
-last_name: str
-friends: str[]
-profile_pic: str
-plans: str[]
-groups: str[]

+ get_profile(x): dict/json
+ edit_profile(x): void
+ upload_profile_pic(x): bool
+ add_friend(x): bool
+ post_plan(x): bool
+ attend_plan(x): bool
+ get_friends(x): list
+ create_group()

Plan

id: str
title: str
description: str
location: str
start_time: datetime
end_time: datetime
creator: str

Calendar

Notification

id: str
message_type: str
recipient_id: str
plan_id: str or None,
payload: str
status: str
created_at: time

Tag

Friend Request

id: str
sender: str
receiver: str
status: str
created_at: time

get_friend_requests(user_id): list

RSVP

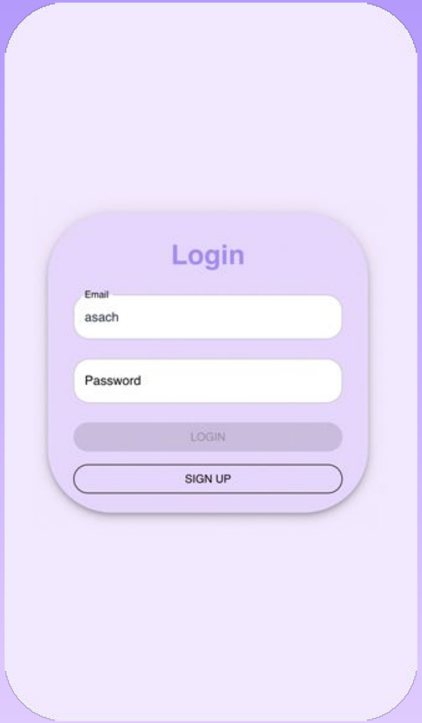
id: str
user_id: str
plan_id: str

get_rsvp_by_event(event_id): list
get_rsvp_by_user(user_id): list

Group

id: str
name: str
owner: str
members: str[]

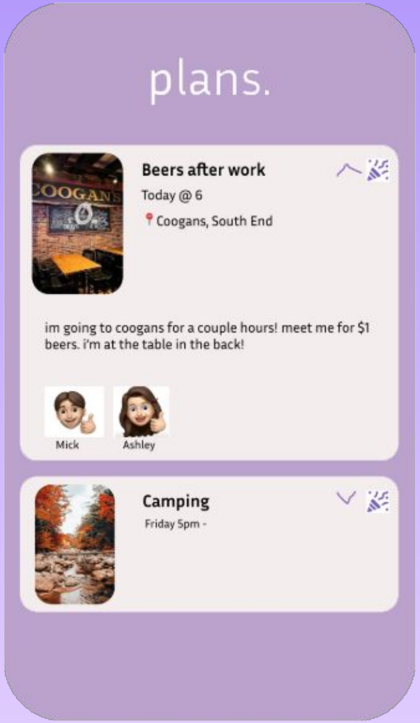
UI Design



Login

Email: asach

Password



plans.

Beers after work

Today @ 6

📍Coogans, South End

im going to coogans for a couple hours! meet me for $1 beers. i'm at the table in the back!

Camping

Friday 5pm -

A balance of simplicity with functionality

Approachable and effective

Database Design

MongoDB

Database Collections

- Users
- Plans
- Groups

Security Design

Code Security - vulnerability and exposure scanning

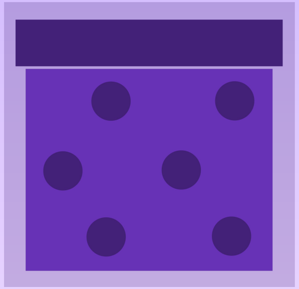
Application Security - JWT session tokens, HTTPS, TLS, SSL

Deployment Security - Securing secret keys and environment variables

Infrastructure Security - SSH key authentications, backups, security softwares

DNS Security - Proxied IPs, Strict SSL/TLS, DDoS defense

Database Security - Encrypted data, TLS, Whitelist IPs



Design Patterns

Frontend

- Builder Pattern
- Observer Pattern
- Container/Presentational Pattern

Backend

- Factory Pattern
- Model-View-Template Pattern
- Repository Pattern (with MongoDB)

Rest API

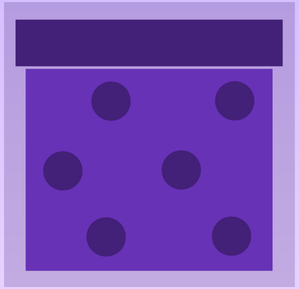
Authentications with JWT - user registration and logins, get logged in user

Users and Friends - update profile, send/accept friend requests, get list of friends, get another user details

Plans - create/edit/delete plans, get plan details

RSVP - rsvp to plan, see who’s also rsvp’d

Notifications - retrieve plans notifications



Future Plans

Moving Forward

For the next iteration, we plan to:

- **Backend:** Add more core functionality such as friends and plan.
- **Frontend:** Add corresponding web pages.
- Securing the infrastructure
- Optimize our database by removing some of the auto-generated Django admin collections if deemed necessary.

Thanks for Watching