PlanningJam: Iteration 0

David Metraux

Bachelor’s: Computer Science at Union College

Role: Team Leader

Reason: Was already doing some of the administrative work

Haolin Yang

Bachelor’s: Theoretical Mathematics at Ohio State University

Role: Design and Implementation Leader

Donjay Barit

Bachelor’s: Electrical and Computer Engineering at Seattle University

Role: Configuration Leader

Reason: Knows how to host backend, knows how to link Jira and Git

Ashley Sachdeva

Bachelor’s: Computer Science and Minor in Mathematics at Boston College

Role: Requirement Leader

Reason: Came up with project idea

Jason Lee

Bachelor’s: Film & TV Production, a Minor in Marketing, at University of Arizona

Role: QA Leader

Reason: Interested in the role

Project Overview

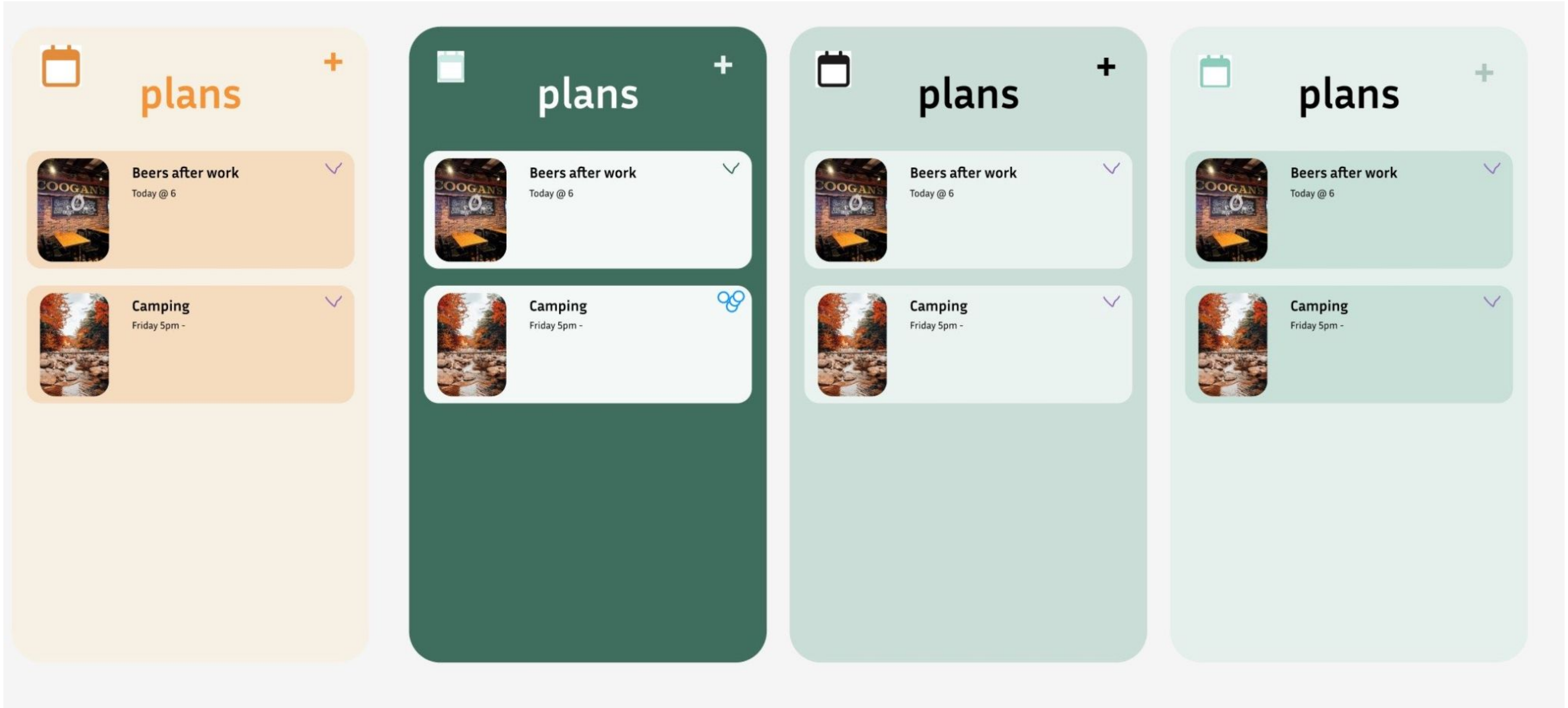
Description: PlanningJam is a low-commitment event planning web app where your friends can only say Yes.

Motivation: Sometimes giving a full response can be pretty stressful.

Other features:

- Friend other users to be aware of their plans
- Filter plans by type

Mockup



plans

Beers after work
Today @ 6

Camping
Friday 5pm -

plans

Beers after work
Today @ 6

Camping
Friday 5pm -

plans

Beers after work
Today @ 6

Camping
Friday 5pm -

plans

Beers after work
Today @ 6

Camping
Friday 5pm -

Tech Stack

Front End: React

Back End: Django/Python

Database: MongoDB

Authentication: JWT

Related Works

- Howbout
- Partiful
- Hobnob
- We stand apart as a web-focused application with low commitment options compared to these other services.

Functional Requirements

Essential

- User Authentication
- Post a Plan
- RSVP (Yes Response)
- Filter Plans by Type
- Friend Requests

Desirable

- Notifications for New Plans or RSVPs
- Plan Editing and Deletion
- User Profile Management
- Dismissing Plans

Optional

- Calendar Integration
- Plan Suggestions Based on Interests

Non-Functional Requirements

- Authentication Security
- Data Privacy
- Input Validation
- Session Management

Objectives and Priorities

1. Complete features:
    a. Have project be properly hosted
    b. Have it so people can post plans to PlanningJam
    c. Have it so people can respond to plans on PlanningJam
    d. Have individual accounts that can be friends with each other
    e. Filter plans by type
2. Deploy the software with Q/A
3. Minimize bugs
4. Maintain high quality
5. Have all team members help equally

Risk Management

- Less availability during a week - Communicate in advance
- Less familiar with technologies - Help guide less familiar team members
- Server downtime - Work on other aspects or switch providers
- In general - Communicate!

Iteration 1 Plan

Essential:

Have front end and back end up

Have users be able to login

Be able to send friend requests

Desirable:

Have users be able to have a profile they can manage

Iteration 2 Plan

Essential:

Have users be able to create Plans

Be able to filter Plans by type

Desirable:

Let users be able to edit and delete their plans.

Give notifications for when friends make Plans

Optional:

Have calendar integration for your own Plans.

Iteration 3 Plan

Essential:

Let users RSVP to events

Desirable:

Let users dismiss Plans

Give Notifications for RSVPs

Optional:

Use ML to give Plan suggestions based on interests

Calendar integration for RSVP’d plans

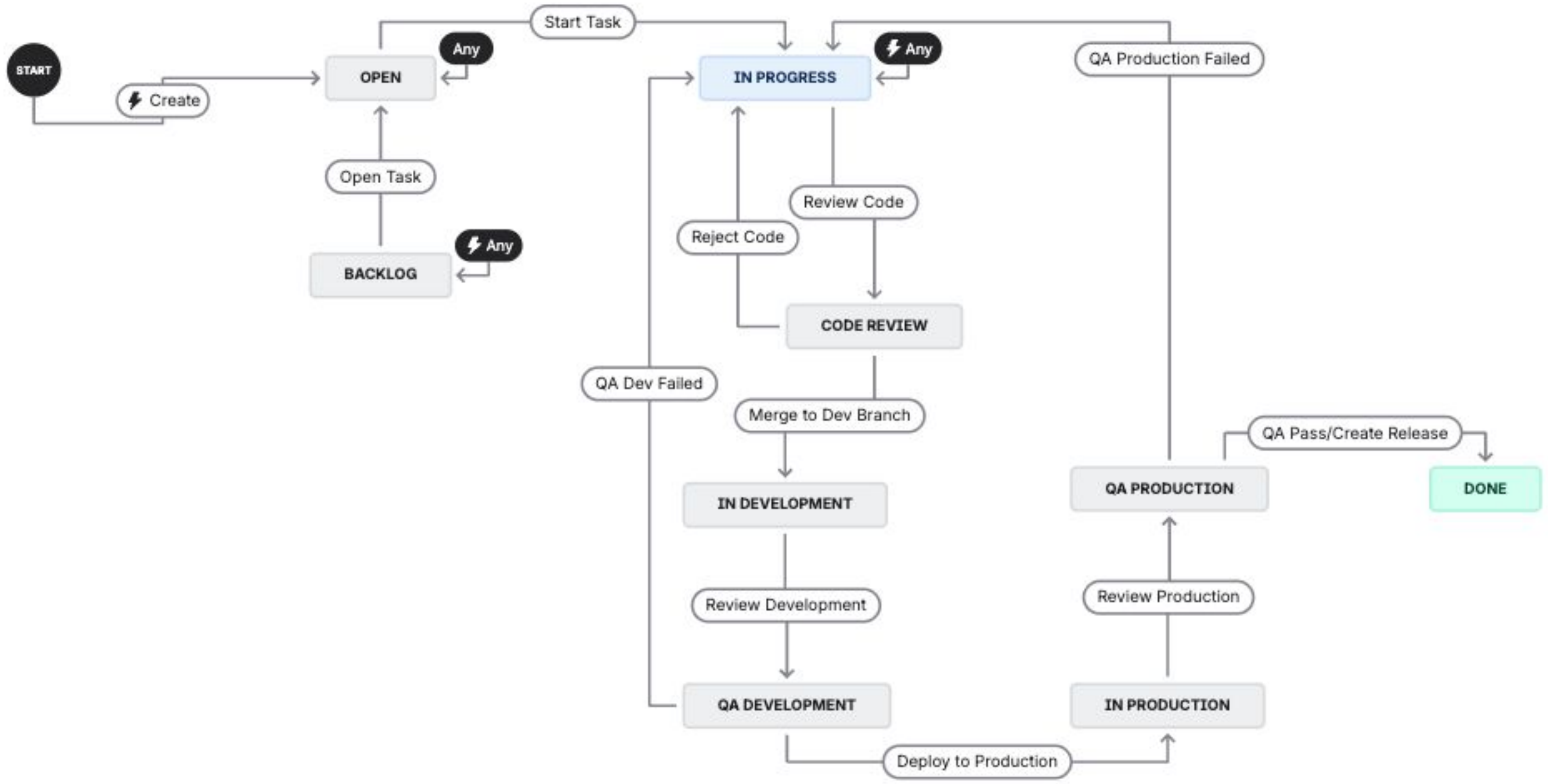
Tools

- Github - code repository, version control
- Github Action - CI/CD
- Docker - Containers to build and deploy the application
- Jira - Project management tool
- Render - frontend and backend hosting
- MongoDB Atlas - database hosting
- ChatGPT - AI help
- Visual Studio Code - IDE
- SourceTree - GUI for managing git repo
- Postman - Tool for API development and testing

Branches used

- development branch - a staging branch for testing updated codebase before deployment
- main branch- contains the full and most updated or recent release of the codebase
- backend branch - branch to be deployed to backend server
- frontend branch - branch to be deployed to frontend server
- iteration branches - contains each released iterations of the project
- feature branches - Jira tasks that team members are currently working on.

Jira Board Configuration



Code Review Checklist

- Identify potential bugs or issues that could arise
- Identify if code can be refactored I.e. redundancy of codes, can be refactored to implement design patterns
- Identify any potential security issues i.e. sensitive information such as API token keys
- Code is readable and has appropriate comments where needed/necessary
- Code can be merged with the merging branch with no conflicts

CI/CD Plan

When development branch is merged into Front/Back End branches:

1. CI builds application
2. Performs automated tests
3. Packages into dockerized container
4. Deploys to production environment
5. UI/UX should be manually checked

Metrics

Metric Name	Description
Defect Density	The # of defects per KLOC (thousand lines of code). Helps track code reliability
Code Coverage	Percentage of source code executed by automated tests. Goal > 60%
Cyclomatic Complexity	Using tools to ensure no function is too complex
Code Review Coverage	Percentage of Pull Requests reviewed before merging. Goal: 100%
Defect Resolution Time	Time taken to resolve logged defects
User Stories Completed	# of user stories completed vs # planned per iteration

Coding Standard

- Frontend (React): Airbnb Style Guide, enforced with ESLint + Prettier
- Backend (Django/Python): PEP 8
- Database (MongoDB): Consistent Schema naming conventions (camelCase for field names, plural collection names, normalize strategically, ER Diagram)
- General: Comments should be kept concise but clear, 1-2 lines in general. For more complex functions, explaining intent & reasoning could help

Testing

- Unit Testing
    - Backend (Python/Django): Django’s testing framework & PyTest
    - Frontend (React): Jest & React Testing Library
- Integration Testing
    - API Testing: Integration between the Django backend and MongoDB Atlas can be tested using Postman (manual) and Newman (automated)
    - End-to-End (E2E) Testing: Frontend/backend interactions can be tested with Cypress to simulate user flows

- Manual Testing
    - Manual exploratory testing will supplement automation to catch edge cases and usability issues. Overall just executing test cases to simulate end-user interactions

Types of Defects

- **Functional Defects:** Features not working as expected or failing business logic
- **Integration Defects:** Failures in communication between the React frontend, Django backend, and MongoDB Atlas
- **UI/UX Defects:** Layout inconsistencies, navigation issues, accessibility gaps, or poor responsiveness
- **Performance Defects:** Slow response times, memory usage concerns, or inefficient queries
- **Security Defects:** Exposure of sensitive data (API tokens, misconfigured database access)

Workflow and Personnel

- **Discovery:** Defects may be identified during code reviews, automated testing, manual QA, or by developers themselves
- **Logging:** The QA leader or the team member discovering the defect will create a Jira issue with clear reproduction steps, screenshots, and severity level
- **Assignment:** Jira will assign the defect to the responsible developer, based on the affected codebase (frontend vs. backend)
- **Resolution:** The developer addresses the issue and submits a new pull request, which undergoes peer review and regression testing
- **Verification:** QA retests the fix to confirm resolution and checks for regressions in other areas
- **Closure:** Once verified, the Jira issue is marked as closed and the defect is documented in the release notes if applicable

Defect Prioritization

Defects will be prioritized based on severity and impact:

- Critical: Blocks major functionality or production use → immediate resolution
- High: Impacts a core feature but has a workaround → addressed within the sprint
- Medium: Minor bug that does not block core workflows → fixed when capacity allows
- Low: Cosmetic/UI issues or minor enhancements → addressed in future sprints

AI Usage

Tools	Who	Tasks	helpful	Evaluation/modification
chatgpt	Ashley	Coming up with a name	Yes, although it recommended a name that already exists.	Came up with a list of names and we picked one! PlanJAM, but there is already one called that. So we will go with PlanningJam instead.
ChatGPT	Haolin	Market Research	Yes	Used the AI to research related applications and come up with differences. The evaluation criteria is based on how well it summarized each application/website. It is also verified for those whose functions are harder to get into without signing up.
ChatGPT	David	Rewording for clarity	Yes	I took a paragraph I had trouble wording in a concise way and edited the result to include more context