CS673 Software Engineering

Team 3 - PlanningJam 

Project Proposal and Planning

Team Member	Role(s)	Signature	Date
Ashley	Requirement Leader	*AshleySachdeva*	09/07/2025
David	Team Leader	*DavidMetraux*	09/07/2025
Donjay	Configuration	*Donjay Barit*	09/07/2025
Jason	QA	*JasonLee*	09/07/2025
Haolin	Design and Implementation Leader	*Haolin Yang*	09/07/2025
Donjay	Security		

X.Y.Z

X - MAJOR version when you make incompatible API changes
Y - MINOR version when you add functionality in a backward compatible manner
Z - PATCH version when you make backward compatible bug fixes

Revision history

Version	**Author**	**Date**	**Change**
1.0.0	Donjay Barit	9/16/2025	Added CI/CD pipeline diagram and updated process

1.0.1	David Metraux	9/20/2025	Making slight changes based off professor's comments
1.0.2	Donjay Barit	9/21/2025	Updates to the configuration management plan with new hosting provider and processes
1.1.2	David Metraux	9/22/2025	Pushed forward elements that we did not complete in iteration 1. Added an optional groups feature
1.2.0	David Metraux	10/6	Pushed forward elements that we did not complete in iteration 2.
1.2.0	David Metraux	10/13	Moved elements we did not complete in iteration 2 to iteration 3, got exact time for iteration 3

1. Overview

PlanningJam is a social app designed to make organizing hangouts simple and fun. It was created to help users coordinate plans efficiently, reducing the back-and-forth of messages while making it easy to discover activities that match their interests.

Users can securely log in, post plans, and invite friends, who can only respond **Yes**. Plans can be filtered by type, and users can send and accept friend requests to connect with others. The app is built using **React** for the frontend, **Django/Python** for the backend, and MongoDB for database management, with authentication handled via JWT.

2. Related Work

- Apple Invites
    - Apple Invites is a new event coordination app released by Apple. Its main focuses are around integration with the Apple ecosystem while allowing users to create

events, send out invitations and track RSVP. However, people not in the Apple ecosystem may encounter inconveniences when using the web interface.

- Partiful
    - Partiful is an event invitation platform that makes planning parties and hangouts feel fun and stylish. Hosts can create playful, visually engaging event pages and share them through links instead of emails. The app includes reminders, polls, and group chats to keep everyone coordinated.
- Paperless Post
    - Paperless Post is a website that allows users to design their invitations online. They offer many designs and templates for numerous occasions. The user also has access to RSVP and guest messaging after they design their invitations.
- Hobnob
    - Hobnob is an event invitation app designed for casual gatherings, parties, and small events. It allows users to create sleek digital invitations directly from their phone, complete with custom designs, photos, and GIFs along with provided templates. Hosts can send invites by text message, making it easy for guests to RSVP without downloading the app. Hobnob also includes group chat for each event, automatic reminders, ticketing services, and options to limit or manage guest lists.
- Sunshine Shine
    - Shine is an all-in-one event/gathering planning app for friends and families. Hosts can send invitations, track RSVPs, and use built-in messaging to chat with attendees. It also supports shared photo albums, so everyone can upload and relive memories after the event. The app focuses on keeping planning stress-free and community-oriented, with features designed for small, personal gatherings.
- Evite
    - Just like Paperless Post, Evite also offers a wide range of invitation free and premium templates and designs. It allows hosts to create event pages, manage guest lists, and track RSVPs with ease as well. The platform includes extra tools like polls, potluck lists, and reminders, making it suitable for parties, family gatherings, and more formal occasions.
- Howbout
    - Howbout is a social event calendar made for close friends. Once events are set, they are synced to everyone's calendar inside the app, giving groups a clear view of what's coming up. It also offers in-app messaging to send invitations and communicate with friends.
- PlanPop
    - PlanPop is a social calendar and event app aimed at friends and family groups. Users can create events, share them with their circles, and see what everyone is up to in one place. The app emphasizes private circles rather than broad public discovery, keeping things personal. Its goal is to help small groups plan casually without the clutter of big event platforms.

Key Differences

- People can only reply "Yes" on each event as they are not targeted but just simply made

aware of the event. This increases the efficiency of RSVP, including even spontaneous events.
- PlanningJam aims to eliminate awkwardness or pressure of saying no to an event.
- Instead of Phone based apps, PlanningJam is a Web-focused application, making it more flexible for people who don't necessarily want to download/update apps.
- Some of these websites send invitations to guests directly via email without friend functionalities while PlanningJam does.
- To summarize, PlanningJam is an in-between application that takes the best from the above phone-based applications and web applications while reducing social awkwardness and pressure for users.

3. Proposed High level Requirements

Functional Requirements

Essential Features

- **User Authentication**
    - **Description:** As a user, I want to log in and create an account, so that my plans and responses are secure and personalized.
    - **Estimated Person-Hours:** 5–10 hours

- **Post a Plan**
    - **Description:** As a user, I want to post a plan, so that all my friended users can see and respond to it.
    - **Estimated Person-Hours:** 5–10 hours

- **RSVP (Yes Response)**
    - **Description:** As a user, I want to respond to my friends' plans with Yes, so that I can indicate whether I will participate.
    - **Estimated Person-Hours:** 2-5 hours

- **Filter Plans by Type**
    - **Description:** As a user, I want to filter plans by type (e.g., food, sports, study), so that I can easily find plans I'm interested in.
    - **Estimated Person-Hours:** 1–3 hours

- **Friend Requests**
    - **Description:** As a user, I want to send and accept friend requests, so that I can connect with people I want to plan with.

Estimated Person-Hours: 10–15 hours

Desirable Features

- **Notifications for New Plans or RSVPs**
    - As a user, I want to receive notifications when a friend posts a plan or responds, so that I stay updated in real time.

- **Plan Editing and Deletion**
    - As a user, I want to edit or delete my own plans, so that I can manage them if details change.
- **User Profile Management**
    - As a user, I want to manage my profile (profile picture, bio, interests), so that my friends can learn more about me.
- **Dismissing Plans**
    - As a user I can X a plan if I do not want to see it on my feed. The event planner will not see this response.

Optional Features

1. **Calendar Integration**
    - As a user, I want to sync plans with my personal calendar, so that I can track my commitments.

2. **Plan Suggestions Based on Interests**
    - As a user, I want the app to suggest plans I may like, so that I can discover new events.
    - Could use ML to identify events that a user may not want to see in the future

Nonfunctional Requirements

Security Requirements

- **Authentication Security:** User login credentials should be encrypted and stored securely using Django Auth or JWT.

- **Data Privacy:** Users' personal information, plans, and responses must be protected and only accessible to authorized users.

- **Input Validation:** All user inputs (plan titles, descriptions, friend requests) must be validated to prevent injection attacks.

- **Session Management:** Implement secure session handling to prevent unauthorized access.

4. Management Plan

a. 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

b. Risk Management

We have identified there are some notable risks to this project. For instance, David Metraux might be moving during the term, and it would be wise to have him tell everyone when that is in advance. This could be the case for others as well, and it would make sense that any team member be prepared to take responsibility for the work of other team members in case of a similar situation.

One potential issue is that we're using a lot of technologies, and not all of us are familiar with each. We plan on mitigating the issue by studying each technology, but besides that, we are going to have to keep watch on the quality of the work done by each team member, and direct to tutorials if necessary.

Another potential issue with our project is server downtime, which could make coding and testing difficult. If this happens occasionally, we can hopefully shift our focus to other tasks that don't require the server that is down. However, if the downtime is frequent or long-term, we may need to consider switching to a new service provider.

In general, the best way to minimize risk will be to constantly communicate, and communicate early, using the pinging feature on Discord. This is the way most issues will be initially started, and once we get the attention of another person, it should not be too hard to come up with a solution.

Risk Management Sheet Link: CS673_SPPP_RiskManagement

c. Timeline

Iteration	Functional Requirements(Essential/ Desirable/Option)	Tasks (Cross requirements tasks)	Estimated/real total person hours
1	**Essential**: Application (front end and backend) up User Authentication **Desirable**: **Optional**: **MOVED PLANS FORWARD THAT WE DIDN'T FULFILL**	**Essential**: Host React front end on service Host Django back end on service Have basic UI in front end ● Website title ● Areas where plans can go Be able to create accounts and login: ● Have login page ● Have logout button Setup Q/A process on Jira **MOVED PLANS FORWARD THAT WE DIDN'T FULFILL**	71

2	**Essential**: Friend Requests Have people be able to create plans **MOVED PLANS FORWARD THAT WE DIDN'T FULFILL**	**Essential** Be able to create friend requests: ● Page where you can add users based on a code? ● List friends on same page ● Be able to remove friends from the same page, maybe with a button ● Setup database with this Make UI for plans ● Have ways to upload images ● Have plans be able to have title ● Have plans have tags ● Setup database with this in dropdown: ● Have plans be able to have description when created in text box ● Have plans be noted who made them **MOVED PLANS FORWARD THAT WE DIDN'T**	70/69.75

		FULFILL	
3	**Essential**: RSVP Filter plans by type **Desirable**: User Profile Management Plan Editing and Deletion Dismissing Plans	**Essential** ● Have a way to accept plans ● Have a way for each person to edit plan response ● Setup database with this **Desirable**: Page where you can add details about yourself ● Keep track of it in a database ● Page ui ● Have a button for each plan you made that lets you edit all contents of the plan ● Have a button for each plan you made that lets you delete a plan ● ● Have a notification page that lists all of friends' plans, including who made them ● Have these	70/43.5

		things affect the database ● Have a way to dismiss plans that you've been invited to: ● Add button to plans ● Keep track of dismissed plans in database ● created on google calendars	
Unachieved	**Desirable** Notifications for RSVPs Notifications for new plans **Optional**: Plan Suggestions Based on Interests Calendar Integration for own plans Calendar Integration for RSVP'd plans	Add to notification page a section that notes when a user has accepted the plan: ● Create table ● Create section of db that keeps track of this **Optional**: Integrate google calendars in a way that lists the calendar events you RSVP'd on google calendars ● Integrate google calendars in a way that shows what events you have that day ● Integrate google calendars in	N/A

		a way that lists the calendar events you	

5. Configuration Management Plan

a. Tools

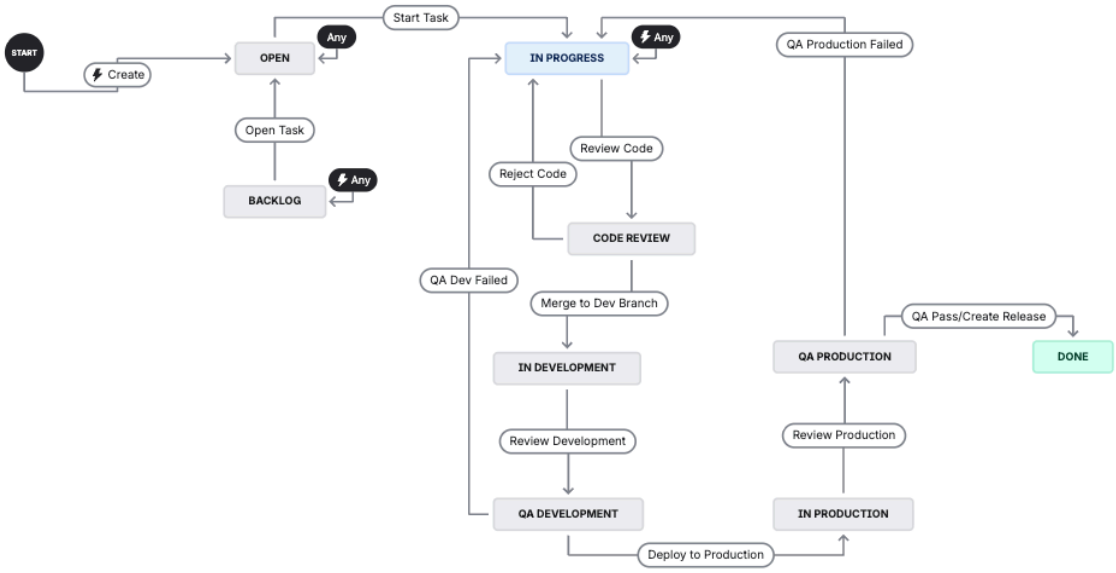
Github - Code repository, Version control
Github Action - CI/CD
Docker - Containers to build and deploy the application
Jira - Project management tool
Digital Ocean Droplets - Virtual private servers to host applications
MongoDB Atlas - database hosting
ChatGPT - AI help
Visual Studio Code - IDE
SourceTree - GUI for managing git repo
Postman - Tool for API development and testing

b. Code Commit Guideline and Git Branching Strategy

The Github repo will contain the following branches but not limited to:

- development branch - a staging branch for testing updated codebase before deployment to production
- main branch- contains the full and most updated or recent release of the codebase
- feature branches - Open jira tasks that team members are currently working on.

Each team member will be responsible for implementing tasks assigned from the project's Kanban/Scrum board in Jira. The project's Jira board is configured in a development transition stages based on the following work flow diagram:



Each Jira issue is a child or subtask of an epic or story. Once a code implementation task is done, the code author will perform a pull request and a code review will be conducted by a peer.

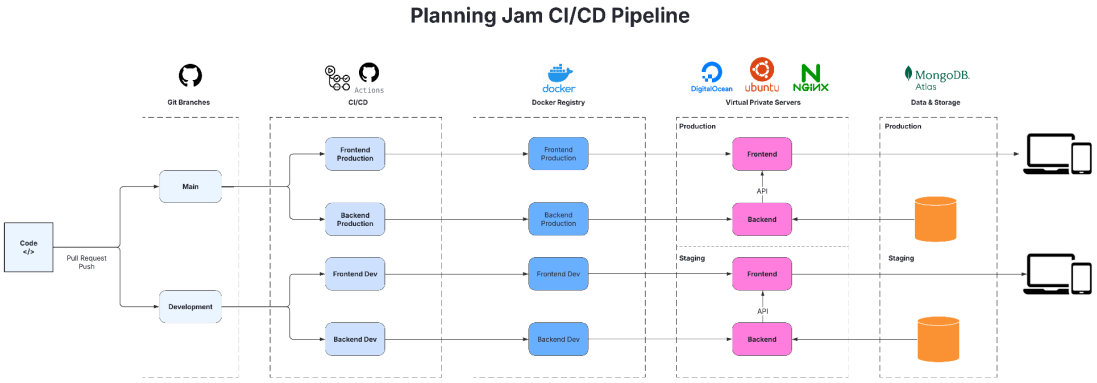
When reviewing a code for a pull request, the code reviewer should identify the following, but not limited to, before approving any pull requests:

- Identify potential bugs, edge cases, or other issues that could arise
- Identify if code can be refactored I.e. redundancy of codes, can be refactored to implement design patterns
- All print or console log statements are not present
- Identify any potential security issues in the code or pull request comments 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

If the code review fails, the implementation task is sent back to the code author to make improvements as needed. If the code review passes, then the pull request is approved and merged into the development branch. A quality assurance is performed, likely from the QA leader, to make sure that the merge is successful and that no issues arise from any updates made to the development branch. If everything looks good and there are no issues, then the development branch can be merged into the production branch and CI/CD can be initiated to the production servers. A final QA is performed to ensure no bugs or issues are present from a UI/UX perspective such as the UI is presentable, buttons are clickable, fields are being updated, etc. Once all tasks are completed within the Epic tasks (iteration), a release

will then be created.

c. CI/CD Plan



This project will have two hosting environments, staging and production with both a backend and frontend services. When a push or a pull request is performed and is merged into either the development or production branches, the CI/CD pipeline gets automatically initiated for the respective environments. The CI/CD is done by Github action and a script is run to build separate applications for the frontend and backend services. Automated unit tests are performed to validate the codes. Should issues arise during the build or tests, the team is notified of failures. If successful, the CI/CD performs a deployment where it builds the frontend and backend into dockerized containers that will be pushed to a docker registry provided by Digital Ocean. The CI/CD will then SSH into the hosting environments, a virtual private server (VPS), where it then pulls the docker images and runs the containers. A final UI/UX should be performed to ensure all user experience with the UI is working.

6. Quality Assurance Plan

a. Metrics

Metric Name	Description
Defect Density or LOC	The # of defects per KLOC. Helps track code reliability. If unable to calculate, LOC is fine as is
Code Coverage	Percentage of source code executed by automated tests. Goal > 60%
Cyclomatic	Using tools to ensure no function is too complex

Complexity	
Code Review Coverage	Percentage of Pull Requests reviewed before merging. Goal: 100%
User Stories Completed	# of user stories completed vs # planned per iteration

b. 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

c. Code Review Process

When a developer completes a task, they can open a pull request. A fellow team member will then review the code before it can be merged. During code review, reviewers are expected to:

- ☐ Identify bugs or logic issues
- ☐ Check for possible security risks (exposed API keys, sensitive data, etc)
- ☐ Spot Redundancy or Design Pattern Improvements
- ☐ Ensure code readability, and sufficient commenting when necessary
- ☐ Ensure any print or console log statements are not present
- ☐ Confirm that the code can be merged without conflicts

If a review fails, the task is returned to the author for updates. If the review passes, the pull request is merged into the development branch.

d. Testing

Testing will be conducted through a combination of manual and automated methods. It will be performed continuously during development during each iteration:

- Unit Testing
  - Each developer is responsible for writing and executing their own code

        - Backend (Python/Django): Django's testing framework & PyTest
        - Frontend (React): Jest & React Testing Library
- Integration Testing
    - QA leader will oversee overall weekly integration testing to validate that the different components (APIs, Databases, UI/UX) of the system interact correctly
        - 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

- Testing Objectives
    - Detect bugs and regressions early in development
    - Verify that new features meet functional and design requirements
    - Ensure system components work together
    - Engage from the end-user perspective

e. Defect Management

Defects will be tracked and managed through Jira, where each issue is linked to the corresponding Epic and sprint iteration

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:

1. **Discovery:** Defects may be identified during code reviews, automated testing, manual QA, or by developers themselves
2. **Logging:** The QA leader or the team member discovering the defect will create a Jira issue with clear reproduction steps, screenshots, and severity level
3. **Assignment:** Jira will assign the defect to the responsible developer, based on the affected codebase (frontend vs. backend)
4. **Resolution:** The developer addresses the issue and submits a new pull request, which undergoes peer review and regression testing
5. **Verification:** QA retests the fix to confirm resolution and checks for regressions in other areas
6. **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

7. AI usage Log

Tools	Who	Tasks	helpful	Evaluation/modification	links
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.	https://chatgpt.com/share/68bddd0e-9834-8006-b6cb-d42e8fdc8318

ChatGPT	Haolin Yang	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.	https://chatgpt.com/share/68bf59e3-ae8c-8008-95db-d02009eb6647
ChatGPT	David Metraux	Rewording for clarity	Yes	I took a paragraph I had trouble wording in a concise way and edited the result to include more context	https://chatgpt.com/share/68c78d17-04ec-8012-bf0e-10c54dc127fb
ChatGPT	Donjay Barit	Dockerfile configuration help	Yes	Had issues with setting up Dockerfile for the frontend using vite and react.	https://chatgpt.com/share/68c611ed-284c-8008-8ef3-e63d88a63500
ChatGPT	David Metraux	Basic React Component	Yes	Removed aspects that were unnecessary, like the various classNames	https://chatgpt.com/share/68c9929a-eed0-8012-95ef-714bfafbe0fa
ChatGPT	Donjay Barit	Github Action CI with Environment variables	Yes it provided the information I needed to work with the CI workflows	Determine how to use sensitive environment variables in Github CI.	https://chatgpt.com/share/68cb8514-28b0-8000-a27a-4acd24230b75
ChatGPT	Ashley	Github	Yes	Needed help with git command for rebasing branch	https://chatgpt.com/share/68d09c4e-74f4-8006-8ee9-15b3cae90

					02d
ChatGPT	Ashley	Github	Yes	Asked for clarity on how refresh tokens work	https://chatgpt.com/share/68d09dec-c340-8006-9a7c-f01fc142297b

8. References

MongoDB. (n.d.). *Django MongoDB backend.* MongoDB Docs. Retrieved September 13, 2025, from https://www.mongodb.com/docs/languages/python/django-mongodb/current/#django-mongodb-backend

9. Glossary