Event Task Aid (ETA)
Project Plan & Software Design Specification (SDS)

Claire Cody, Luis Guzman-Cornejo, Ryan Kovatch, Evelyn Orozco, Clio Tsao

6-02-2025 - v2

Table of Contents

1. Revision History

Date	Author	Description
5-13-2025	Clio T.	Created the document outline, added Project Plan
5-13-2025	Ryan K.	Filled out 2. System Overview and added to 4. Software Architecture
5-20-2025	Clio T.	Completed additional sections for Project Plan and Software Modules
6-02-2025	Clio T.	Updated for final project implementation.

2. System Overview

The system will be a web application capable of creating events with various deadlines associated, such as for booking rooms, renting equipment, and requesting funds. The web application will be served by a Python backend using the popular Flask module. For our database of organizations and events, we will use MongoDB. The entire backend will be built, configured, and deployed using Docker.

3. Project Plan

Team and Management

Three members of the team will be working on the front-end user interface and logic, while two members of the team will be working on the backend MongoDB database management. Frontend modules are further divided between the members of the frontend team. Task assignments are designated to 1-2 team members to create and design according to their strengths. The frontend and backend teams will coordinate individual meeting times to make progress, with full group meetings at least once a week to coordinate integration between the two.

The team will communicate questions and updates via a group chat. This allows for live updates on progress, questions, or changes to the system. Upcoming meeting times are also confirmed in this group chat. Large-scale planning, decision-making, and work discussions will occur in weekly group meetings at a designated time. During these weekly group meetings, each team member shares what they have been working on, what they plan to work on next, and any sticking points they may need help with. These check-ins will ensure accountability with deadlines, allow for peer code review, and periodic quality assurance such that we can test and catch errors early. Meeting minutes are documented and tracked in a live document (Appendix A).

Version control will be managed with GitHub. Each member of the team will have their individual branch for development. For code review, testing, and collaboration, members can pull from each other's branches to see their changes. Finalized working versions of the system are pushed to the main branch for everyone to sync from.

Build Schedule

Our project timeline will be tracked according to a live spreadsheet Gantt chart (Appendix B). Our schedule is organized by days and weeks, with large deadlines such as project plan, initial prototype, and working system (1 week, 2 weeks, 3 weeks respectively) highlighted for ease of visibility. The various tasks before each deadline are identified and divided to team members in the weekly meetings before. Fine grain task assignments and bug fixes that occur are logged in the Project 2 Documentation for members to quickly reference.

Deadline and Milestones

Deliverables for the course include:

- Wednesday, May 14, 10 PM: Initial 3-page Proposed SRS/SDS/Project Plan.
- Wednesday, May 21, 10 PM: A complete SRS/SDS/Project Plan.
- Wednesday, June 4, 10 PM: Complete projects submitted.

We identify additional group milestones as:

- Friday, May 16, 11:59 PM: Set up Github branches for version control, establish version control practices for the group.
- Wednesday, May 21, 11:59 PM: Interview student leaders for possible use cases.
- Wednesday, May 21, 11:59 PM: Complete prototype dashboard view (frontend interface).
- Thursday, May 22, 11:59 PM: Complete setting up MongoDB schemas and backend endpoints (database).
- Tuesday, May 27, 11:59 PM: Finish connecting frontend interface to backend server.
- Friday, May 30, 11:59 PM: Complete working system on local development.
- Monday, June 2, 11:59 PM: Finalize system documentation.
- Thursday, June 5, 10 AM: Present final system.

Risk Mitigation

Use of GitHub version control with branches for each developer will allow for more fine grain control of what final changes are committed to the final branch. By testing individual changes early in personal branches, we can catch errors early in the process and avoid expensive mistakes further down the line.

Additionally, we planned out the necessary tasks for the system and assigned them to members, tracked in a Gantt chart. This allows for tracking of processes and a clear view of the critical path. Since the Gantt chart is in a shared live document, everyone has access to an up-to-date schedule and can adjust to account for any changes. By being aware of these timelines, we can mitigate delays and missing milestones.

4. Software Architecture

4.1. High Level Overview

Event Task Aid: State Diagram

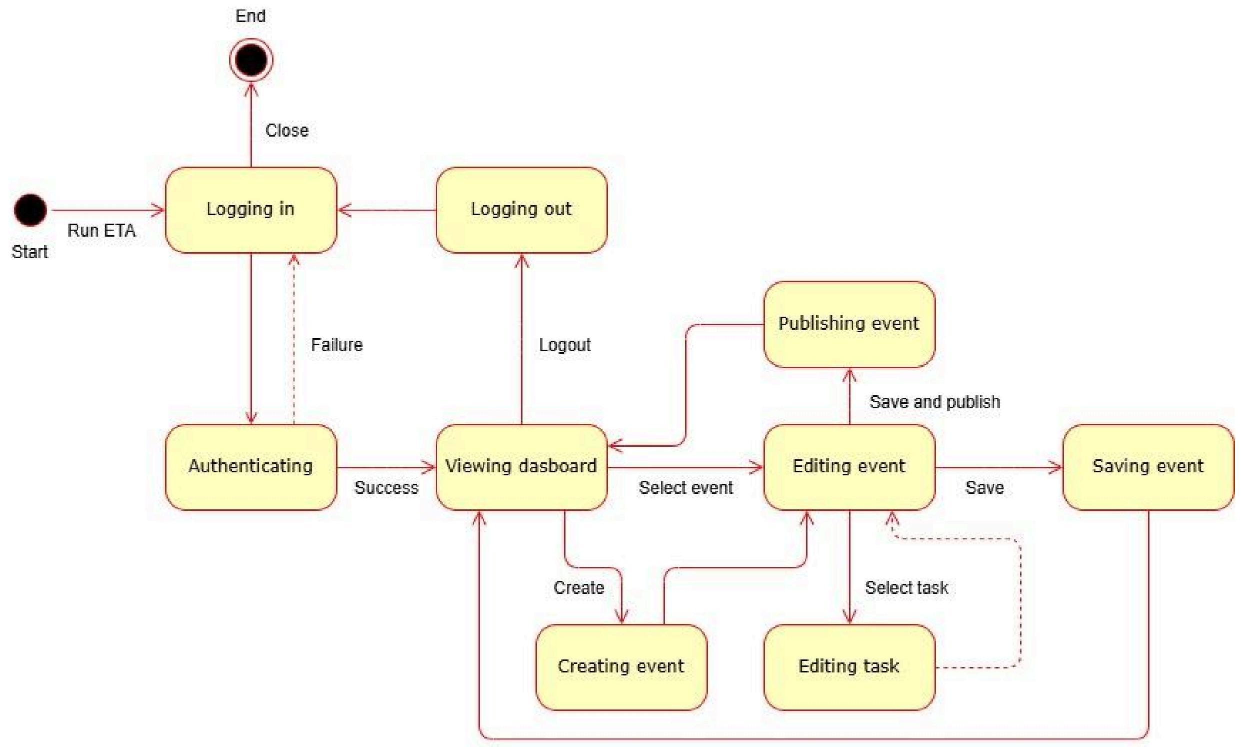


Figure 4.1 State diagram of overall ETA system architecture

4.2. Intended Technologies

Backend: A Docker Compose stack.

- Web server: Python + Flask for web server, PyMongo/MongoEngine for database access.
- Database: MongoDB. We download a preconfigured blank instance from Docker Hub during the build process.

Frontend: A collection of pure HTML + CSS + JavaScript documents. The documents may be *templates*, which include dynamic data and are rendered by Flask when served.

- We may use the Push and Notifications APIs to deliver notifications to users through their browsers when events are approaching/changes are made to events. (Need some way to check/send notifications periodically, perhaps a Cron job in one of our containers.)

4.3. Module Overview

Backend

- Web server: will serve the main web pages of the app + implement a REST API over HTTP. Communicates with the database to read/write objects.

- Database: will store objects representing organizations, users, and events. Exposed to the web server over a network/port.

Frontend

- The main UI implementation which is loaded by a browser and communicates with the backend over HTTP.
- Login page: allows org managers to log in to manage events, etc.
- Event view dashboard: shows list of events for an org. When logged in, it allows event creation and shows draft events.
- Event creation view: allows org managers to create an event and specify what services it will need (creating deadlines).
- Event edit view: similar to event creation view, allowing org managers to edit events that have already been created.
- Edit organization view: allows org managers to modify the details of their organization, including title, description, org colors, and members.

The functional frontend interfaces are further broken down into modules. The modules are detailed in Table 4.3 below.

MODULE	FUNCTIONALITY AND RATIONALE
Login	- Prompts user to enter login credentials which are then verified in the user database
	The login module ensures that only officers with organization access are privy to potentially sensitive event planning details. This also makes it easier for a single user to track events relevant to them. While we considered having users log in directly under an organization as an alternative design, we decided to have the system be accessible by user since that would allow for easier tracking of who was making edits to an event or task. Additionally, one user might have multiple organizations.
Dashboard	- Display the user's organization (can toggle between the user's different organizations from a drop down menu) - Allows user to view upcoming events - User can add upcoming events to their Google calendar or Apple calendar - View unpublished events in planning progress, as well as associated tasks for each event - Select events to edit - Create new events - Edit organization information - Publish events
	The dashboard module provides an "at-a-glance" overview of what events are coming up. It has further functionality that allow for easy tracking of which event tasks are for, and control over sharing details of upcoming events between members.
Event Creation	- Allows user to create a new event - Allows user to fill out fields with event details and add fields

	- Fields include: event start date, description, venue, budget, point of contact, expected attendees. - The event can be: - directly published from this interface - saved (unpublished) for additional editing - Tasks can also be created with an event, allowing for tracking of what needs to be accomplished - Tasks will have fields for: task title, description, date, and who it is assigned to
	The event creation view will be a form-type interface accessible from the dashboard. This is to make it easy to create new events. Initially, we considered having the event creation view only have a function to save a new event as unpublished, but we decided to add publish functionality as a field to this view in order to streamline the user experience a little more.
Event Edit	- Allows user to edit an existing event - Allows user to modify fields with event details and add fields - The event can be: - published from this interface - saved (unpublished) for later editing - Tasks be added to existing events
	The event edit view is nearly identical to the event creation view, serving the same core functionality of populating fields related to various information of an event and adding tasks. This view is accessible upon clicking on existing events from the dashboard.
Organization Edit	- Allows user to edit and save organization information, including title, description, org colors, and members
	The organization edit view will be accessible from the dashboard, allowing for organization officers to manage the details of their org. Since organizations often have specific information about goals, meetings, or other important updates in their description, we decided to make it easy for managers to edit and publish this information from their ETA dashboard.

Table 4.3: Modules of the ETA system, their functionalities, and rationale.

5. Software Modules

5.1. Login Page

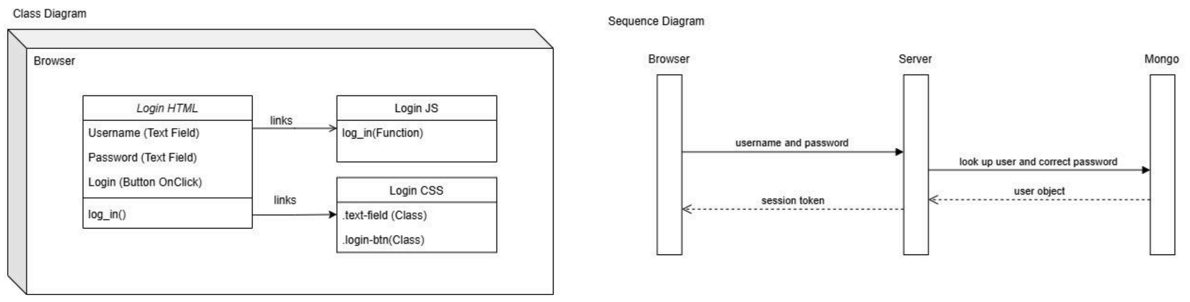
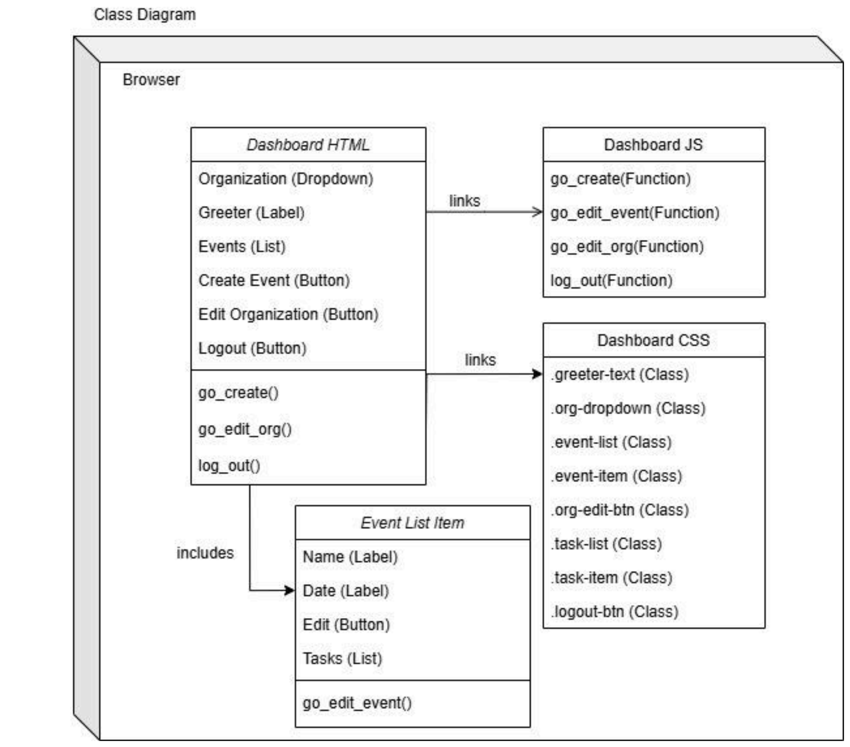


Figure 5.1: Login page class and sequencing diagram.

The login screen will have two text fields for username and password. Once both fields are filled out and the user clicks the “Login” button, the browser sends the username and password to the server, which looks up these field values in the database’s user collection. It then returns a session token, saved to the browser’s local storage and used for future API requests. Figure 5.1 shows the class and sequence diagrams for its operation.

5.2. Dashboard



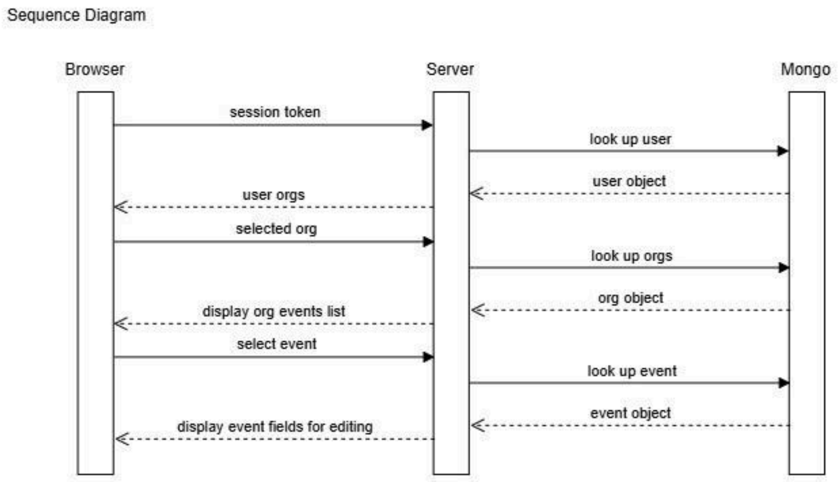
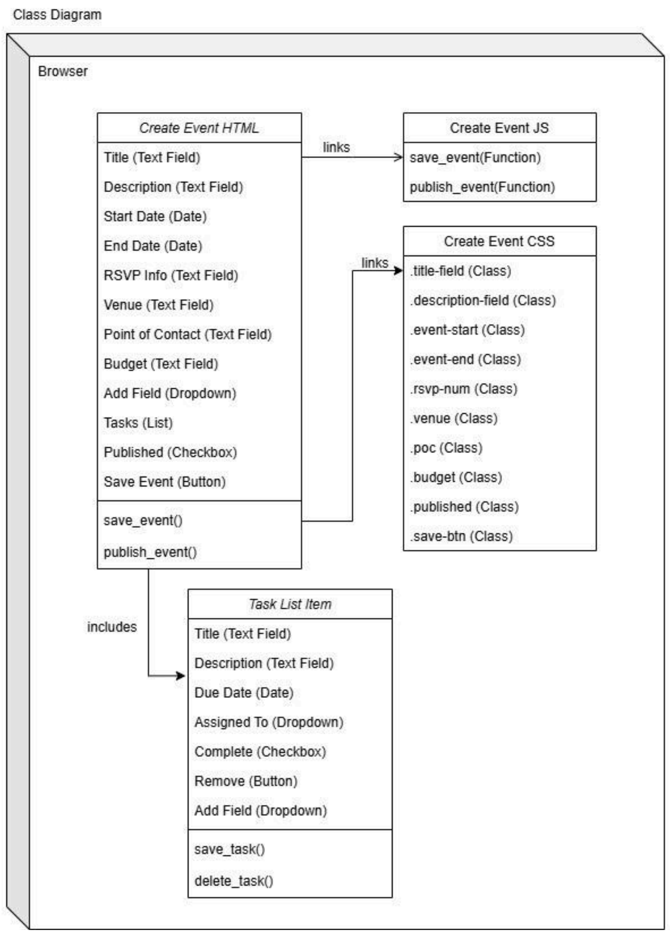


Figure 5.2: Dashboard page class and sequence diagram.

The Dashboard is the main page for the user to gain an at-a-glance overview of their organization's upcoming events, with the event title, description, and date visible. There will be a drop down menu with the organizations that the user is in, to select which one of them they wish to view. We plan for it to allow managerial users to select existing events and modify them, which will bring them to the Event Edit Page. Clicking on a button to create new events will bring them to the Event Creation Page. The user can also log out via a Logout button.

5.3. Event Creation Page



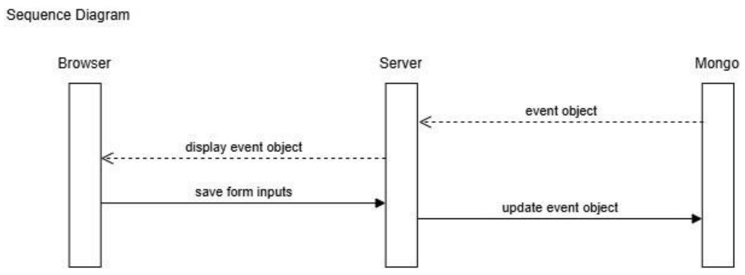
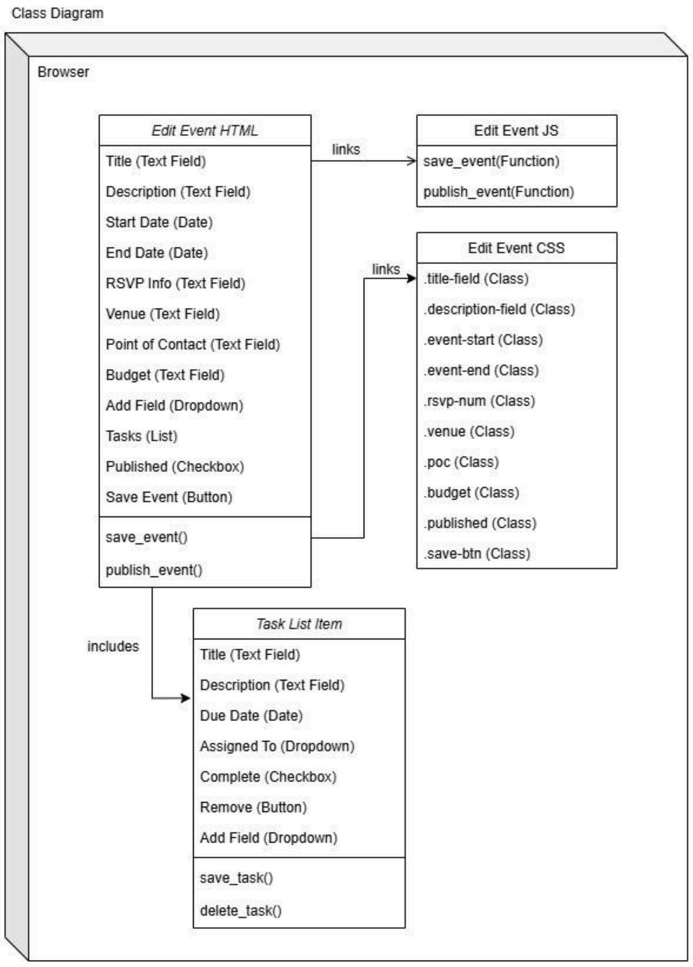


Figure 5.3: Event creation page class and sequence diagram.

The Event Creation Page for creating a new event contains many fields for the various information related to an event. There will also be a section for adding tasks to a list for further delegating various event-related tasks. The user can fill out these fields, add tasks, and then click save event to save the event (unpublished). This makes the event visible to other logged-in users to see and edit. If the Published field is checked, then the event will be saved and published for all users to view. Users can also continue to modify the event. These changes will appear back on the Dashboard page.

5.4. Event Edit Page



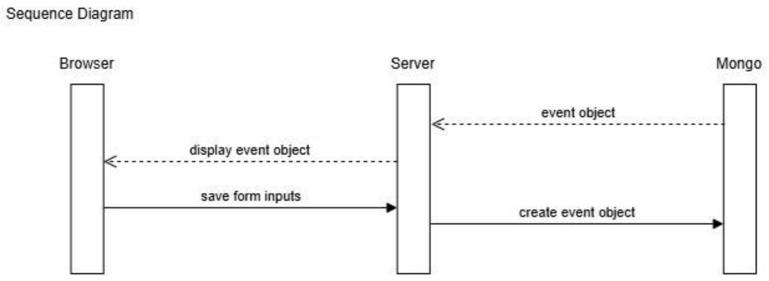


Figure 5.4: Edit event page class and sequence diagram.

The Event Edit page is nearly identical to the Event Creation page, except that it is accessed when a user chooses to edit an existing event from their dashboard. After the existing event object is retrieved from the database, the information for each field will be loaded in for the user to edit. Once editing is complete, the user may choose to save or publish as with event creation. These changes also appear on the Dashboard Page.

5.5. Organization Edit Page

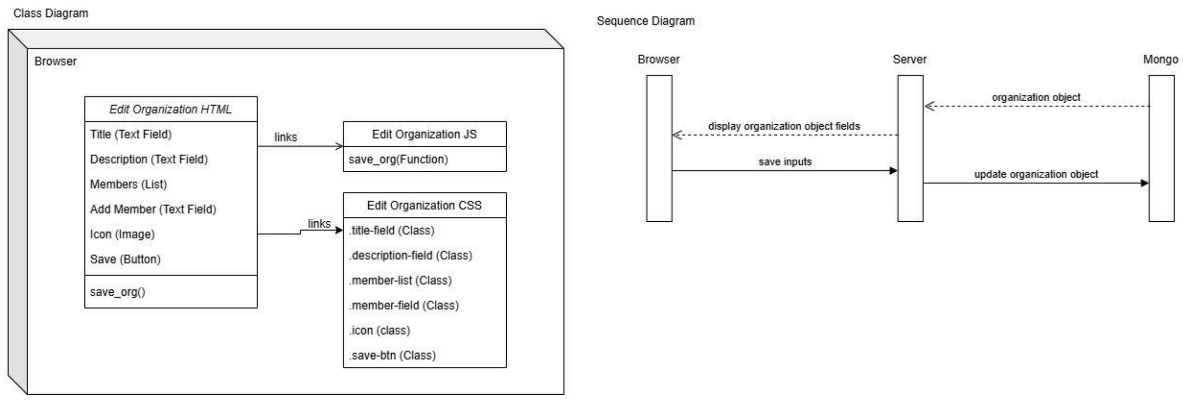


Figure 5.5: Organization edit page class and sequence diagram.

The Organization Edit Page is accessed when a user chooses to edit information for their org from the dashboard. This page will allow them to modify fields for their organization title, description, and members. Once complete, the organization object has its fields updated in the database.

5.6. Alternative Designs

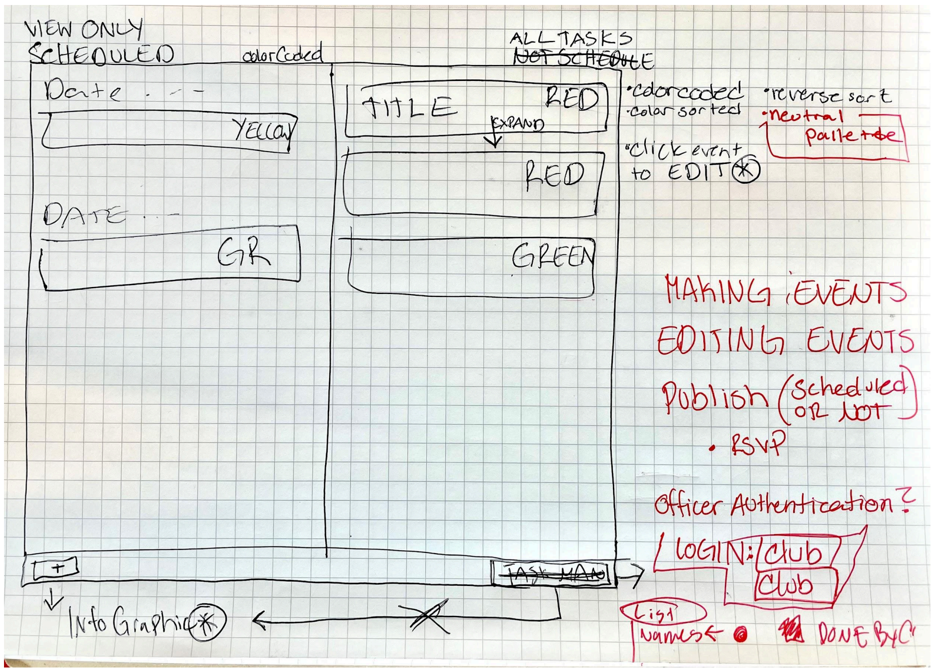


Figure 5.6: An early design of the dashboard.

Figure 5.6 shows an early sketch, where officer authentication would have been done by club instead of by user. We decided to keep the general design of the layout for the dashboard view, but added additional pages to modularize functionalities.

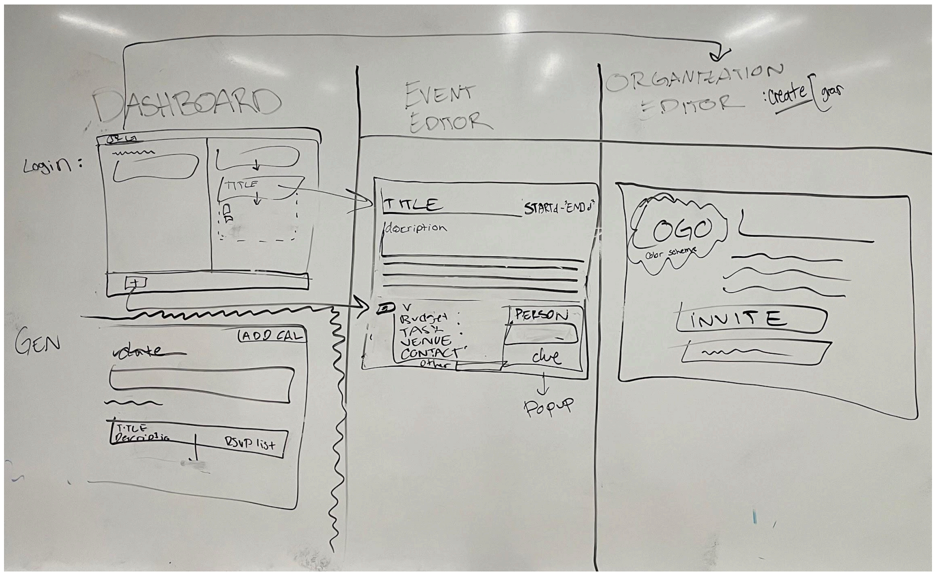


Figure 5.7: A more detailed design sketch of the ETA system.

Figure 5.7 includes the additional modules of the system. The General/Manager view split for non-managerial users of the organization was an initial idea, though not implemented.

6. Dynamic Models of Operational Scenarios (Use Cases)

The Event Task Aid can be adapted for the task tracking for various event needs. It is intended for student leaders of campus clubs, non-profit organizers, and other groups that may need to coordinate community events of a larger scale. The following sections provide two example use cases of planning and tracking tasks for an event using the system.

6.1. Community Food Drive

A user planning a food drive could use the ETA as such:

1. Log in under their organization group.
2. Create an event and fill out relevant fields.
3. Assign event tasks.
4. Mark completed tasks.
5. Publish the event.

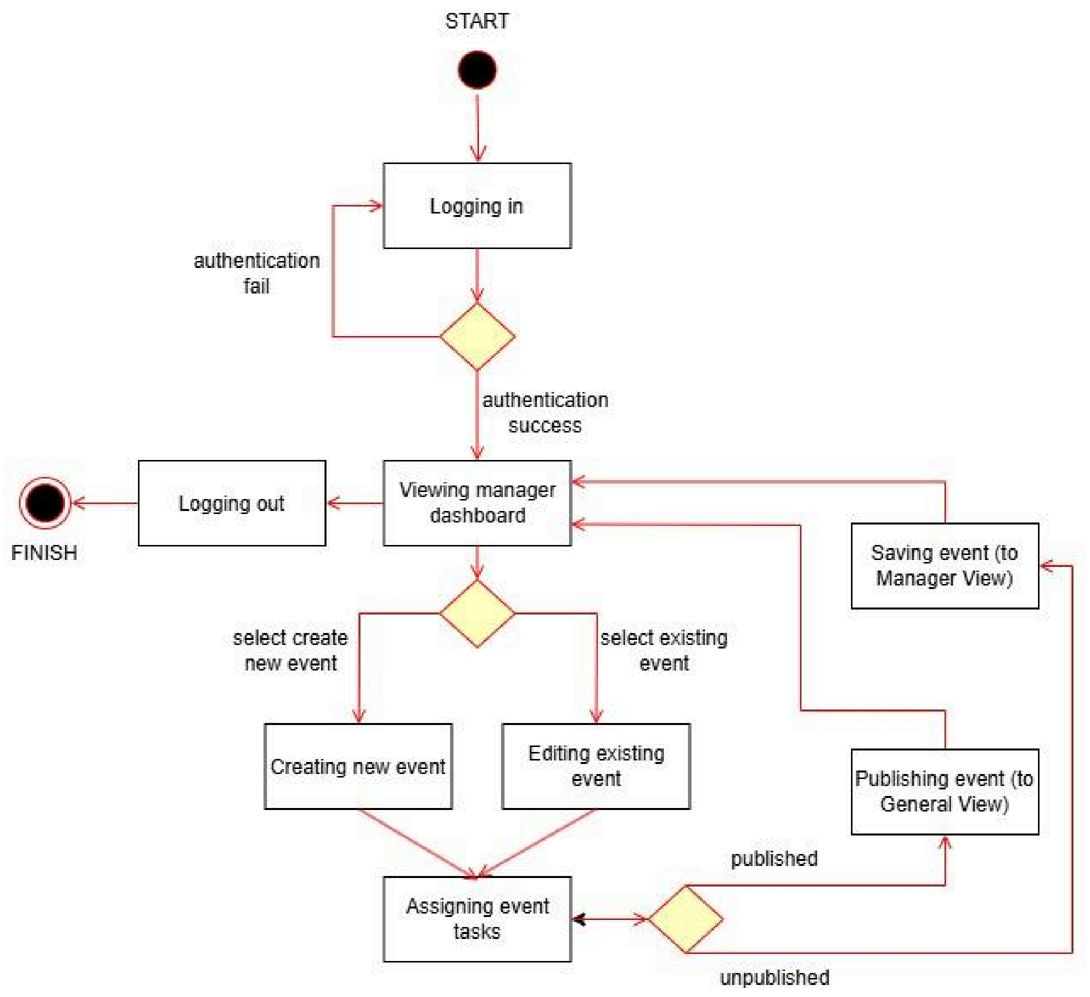


Figure 6.1: An activity diagram of the steps the user may make to make an event.

6.2. Club General Meeting

The ETA could also be used for scheduling a general meeting for a UO student organization. A club officer may:

1. Log in under their organization group.
2. Create an event and fill out relevant fields.
3. Publish the event.
4. Update the organization description.

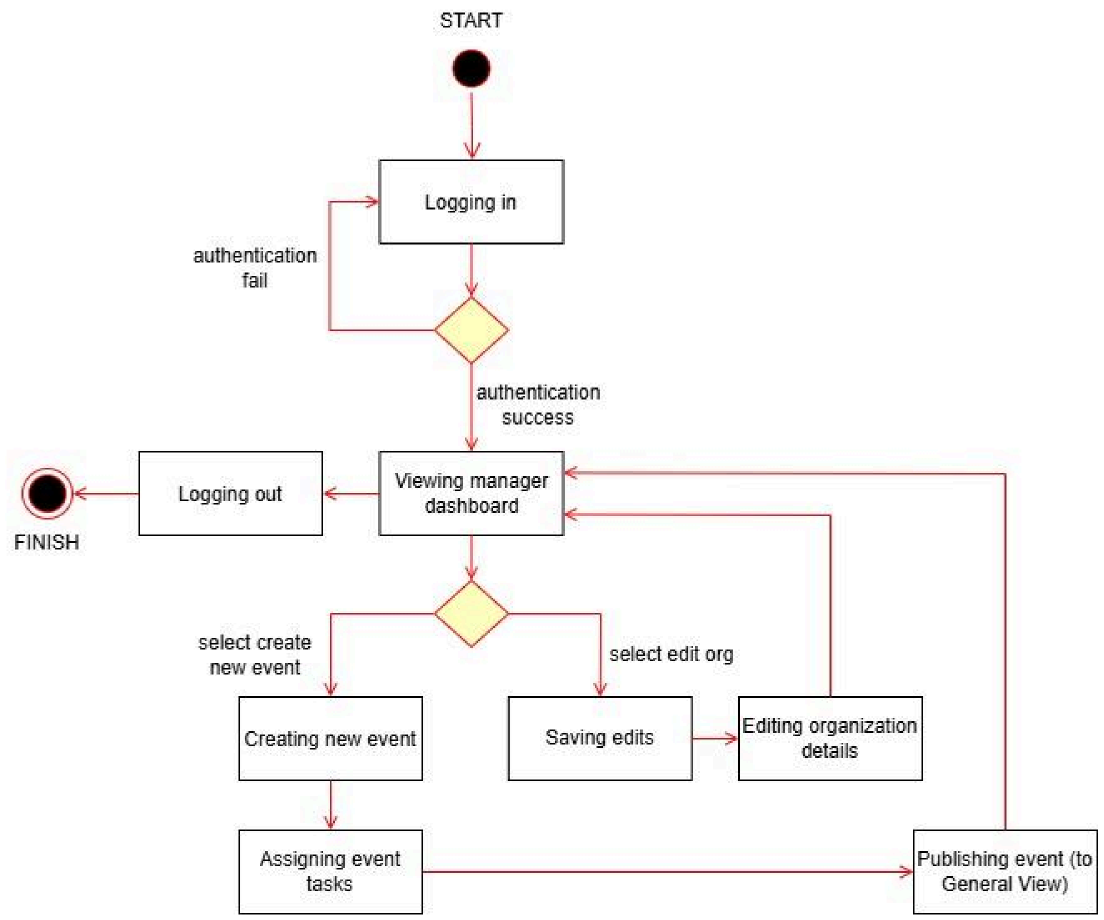


Figure 6.2: An activity diagram of the steps a user may take to create an event and modify org details.

6.3. General Member Event View

In addition to organizational officers using the ETA, it is also accessible to general members to view published events as a simpler use case. A general member may:

1. Log in under their organization group.
2. View published events from their dashboard.
3. Select an event and view it.

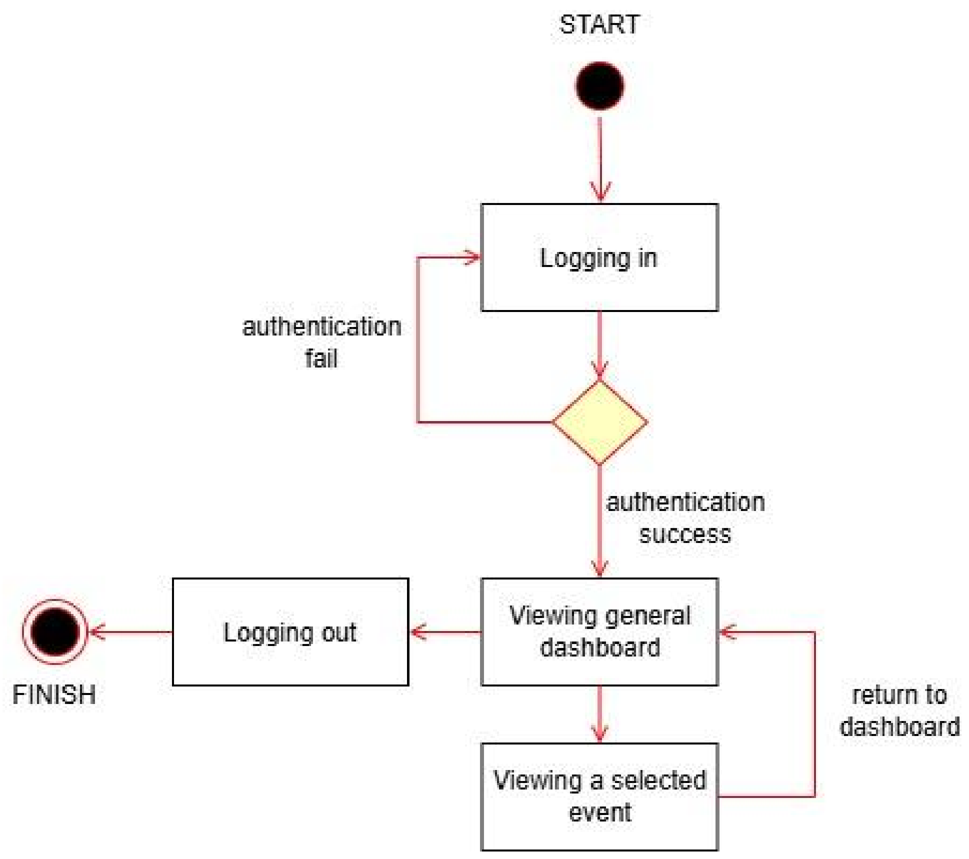


Figure 6.3: An activity diagram of the steps a general user may take to view an org event.

7. References

Faulk, Stuart. (2011-2017). CIS 422 Document Template. Downloaded from https://uocis.assembla.com/spaces/cis-f17-template/wiki in 2018. It appears as if some of the material in this document was written by Michal Young.

IEEE Std 1016-2009. (2009). IEEE Standard for Information Technology—Systems Design—Software Design Descriptions. https://ieeexplore.ieee.org/document/5167255

Parnas, D. L. (1972). On the criteria to be used in decomposing systems into modules. *Commun. ACM, 15*(12), 1053-1058.

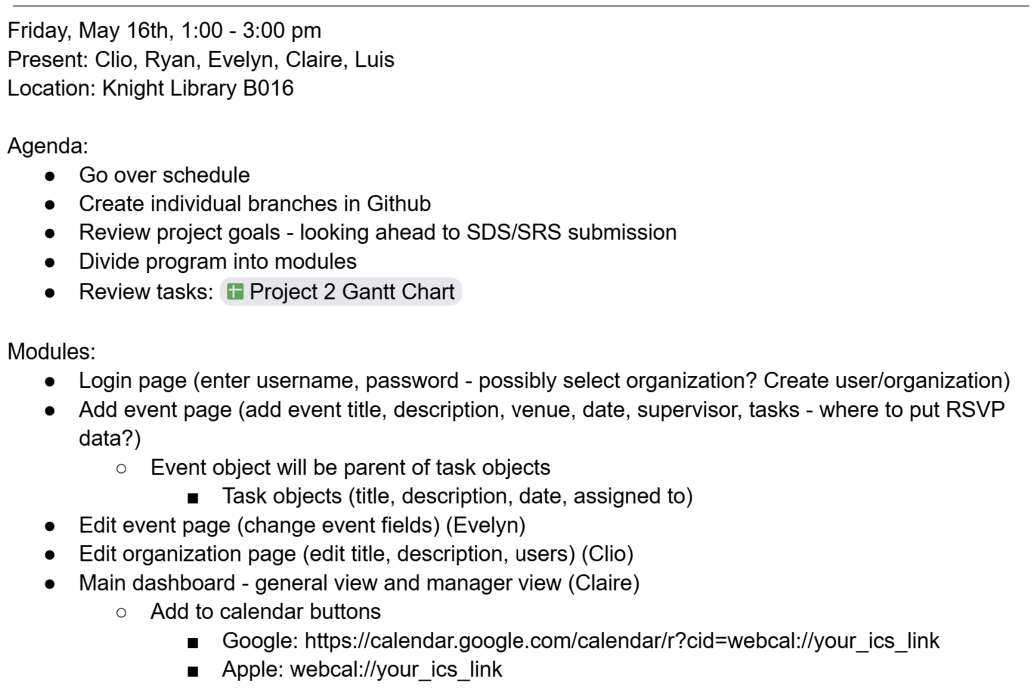
8. Acknowledgements

The template of the Project Plan and SDS was compiled and provided by Prof. Anthony Hornof which also builds slightly on a similar document produced by Stuart Faulk in 2017, and heavily on the publications cited within the document, such as IEEE Std 1016-2009. This document also draws on what we learned over the course of Project 1 for CS 422 while putting together its Software Design Specification (SDS).

9. Appendix

9.1. Meeting Minutes

Meetings are documented as in Figure 9.1. All meetings are documented in a live document linked here: CS422 Project 2 Documentation



Friday, May 16th, 1:00 - 3:00 pm
Present: Clio, Ryan, Evelyn, Claire, Luis
Location: Knight Library B016

Agenda:

- Go over schedule
- Create individual branches in Github
- Review project goals - looking ahead to SDS/SRS submission
- Divide program into modules
- Review tasks: Project 2 Gantt Chart

Modules:

- Login page (enter username, password - possibly select organization? Create user/organization)
- Add event page (add event title, description, venue, date, supervisor, tasks - where to put RSVP data?)
    - Event object will be parent of task objects
        - Task objects (title, description, date, assigned to)
- Edit event page (change event fields) (Evelyn)
- Edit organization page (edit title, description, users) (Clio)
- Main dashboard - general view and manager view (Claire)
    - Add to calendar buttons
        - Google: https://calendar.google.com/calendar/r?cid=webcal://your_ics_link
        - Apple: webcal://your_ics_link

Figure 9.1: Sample meeting agenda and notes.

9.2. Gantt Chart

The project timeline is tracked with a live Gantt chart linked here: Project 2 Gantt Chart
Table 9.2. is a sample portion of the Gantt chart.

		May												
Tasks and Milestones	**Member(s)**	9	10	11	12	13	14	15	16	17	18	19	20	21
Initial meeting, discuss project plan	all	■												
Determine front-end implementation	Clio, Claire, Evelyn	■	■	■	■									
Determine back-end implementation	Ryan, Luis	■	■	■	■	■								
3 page summary	all				■	■	🟥							
Gather use cases from student leaders	Ryan, Evelyn, Clio						░	░	░	■	■	■	░	░
Set up Github branches for version control	all								■					
Refine project specifications (SDS/SRS compliant)	Clio									■	■	■	░	
Submit SRS/SDS/Project Plan	all													🟥
Implement prototype dashboard view/layout	Clio, Claire, Evelyn								■	■	■	■	░	░
MongoDB server setup	Ryan, Luis								■	■	■			
MongoDB schemas	Ryan, Luis											■	░	░
Backend endpoints	Ryan, Luis												░	░

Table 9.2: A sample portion of the project timeline Gantt chart.