Project Milestone 2

Requirements

CS 3704
Spring 2024

Prepared by
TimeTeam: Nicholas Hess, Yasir Hassan, Thomas Tran, Tim Vadney, Gio Romero-Ruiz

3/15/2024

Table of Contents

Requirements Workshop

0.a
TimeTeam

0.b

Present	**Absent**
Nick Hess (hessnt30)	Tim Vadney (vtim)
Yasir Hassan (yasirh)	
Thomas Tran (thomastran)	
Gio Romero (gioromeroruiz)	

0.c
Our project idea is called **TimeScape**. When trying to meet up with friends or other people, it can often be difficult to determine optimal times that work for everyone. As such, our project aims to take in the available times of all members, and visually display the windows that syncs up with the most amount of people (e.g. time slot X has ¾ members free, time slot Y has 4/4 members free, etc.). Doing so will make it more efficient to help arrange meetups.

Requirement Analysis

1. Provide an example of five hypothetical non-functional requirements for this system. Be sure to include the specific type of requirement discussed in class, with each requirement coming from a unique category.

- **Usability:** Have a 'Help' section to explain how to use the application
- **Reliability:** Should only have a downtime of about 5 seconds if it crashes
- **Performance**: Should be able to update near real-time with multiple members using/viewing it simultaneously
- **Supportability**: TimeScape should be implemented in a way that allows for the developers to continuously update and maintain the website.
- **Implementation/Constraints**: Must use React, JS, Django/Nodejs

2. Provide an example of five hypothetical functional requirements for this system.

- TimeScape must be able to **read user inputted data** as availability each day of the week
- TimeScape must be able to accurately **calculate** the best meeting times (time slots where the most amount of members are available) depending on the user input
- TimeScape must be able to **display** the best availability in an easy to understand format
- TimeScape must allow for a user to **invite others** to the planned meetup scheduler

- TimeScape must be able to allow the user to **create an account** and store their information regarding available meeting times

3. Think of a specific task required to complete each of the functional requirements and non-functional requirements mentioned above (10 total). Estimate the amount of effort needed to complete this task using function points (i.e., using the values here). Briefly explain your answer.

1. Usability: Have a 'Help' section to explain how to use the application
    - Create accurate documentation regarding how to use TimeScape
2. Reliability: Should only have a downtime of about 5 seconds if it crashes
    - Detect if the program has crashed and restore it to the most recent stable state
3. Performance: Should be able to update near real-time with multiple members using/viewing it simultaneously
    - Determine how to sync information from the database and front-end in a way that maintains performance
4. Supportability: TimeSkip should be implemented in a way that allows for the developers to continuously update and maintain the website.
    - Provide accurate documentation to ensure easy planning and structuring
5. Implementation/Constraints: Must use React, JS, Django/Nodejs
    - Group members learn these technologies depending on their role in the project
6. TimeScape must be able to read user inputted data as availability each day of the week
    - Parse input and add it to a data structure to track availability
7. TimeScape must be able to accurately calculate the best meeting times (time slots where the most amount of members are available) depending on the user input
    - Develop an algorithm to analyze user availability and calculate optimal meeting times
8. TimeScape must be able to display the best availability in an easy to understand format
    - Brainstorm different ways to display the visual data (ex. Bar graph, 3D plot, line graph)
9. TimeScape must allow for a user to invite others to the planned meetup scheduler
    - Create a unique invite link that adds you to a specific group
10. TimeScape must be able to allow the user to create an account and store their information regarding available meeting times
    - Develop user registration functionality and implement a secure storage mechanism for user data.

4. Write three user stories from the perspective of at least two different actors. Provide the acceptance criteria for these stories.

School club

- President
    - Bruno is the president of an extremely popular organization here at Virginia Tech. He recently has been struggling with gathering information on his organization's

        members to figure out when to host events at an optimal time. Recently Brad has only been able to get 15-20 people to come to his events and he realized that it was because the times he picks aren't optimal for his members. He does extensive research and he finds out about TimeScape which allows him to oversee all of his members' availability which can make the event planning for his club easier. Bruno creates an event schedule and using the club's email list he emails every single member of the club to fill out a form. Each member has to fill out their availability throughout the time frame Bruno asked for. Where Bruno can then decide to pick an optimal time to host club meetings.

- Member
    - Fred is a member of the club that is organized by Bruno. Fred is an extremely busy student taking 20 credits a semester as he is double majoring in Math and CS. Fred receives the email from Brad asking for his availability so he can get a grasp of what time members are free. Fred is able to successfully fill out the form as he had no issues navigating the application. This facilitates Fred's schedule because it helps him keep track of events going on around the club because this semester he wants to become an active member. Even with his stressful and busy schedule Fred is able to make time for Bruno's club.

Work

- Project manager
    - Bill is a project manager at a medium-sized software company. He wishes to schedule a department-wide meeting to get up-to-date on the status of all the teams he's overseeing. In the past, Bill would ask in the department's Slack channel for everyone's availability, and would get flooded with a multitude of messages. Furthermore, he would then have to sift through each one and manually determine a good meeting time to accommodate the most amount of people. As such, he uses TimeScape to visualize the availability windows of 15-20 employees to determine the most optimal meeting time. Ideally, he wishes to find a time slot where no employees are excluded from this meeting. Bill sends out the invitation link to the schedule, awaits his employees' responses, and selects an optimal meeting time.
- Software engineer
    - Brad is a software engineer working on a team overseen by Bill. Brad has multiple meetings during the day—within his team and with potential clients. Additionally, his schedule changes often, so he does not have a static window of availability week-to-week. Brad receives the invitation link from Bill to the TimeScape schedule and inputs his available time windows for specific days, changing them as needed as his obligations adjust throughout the week. In the past, he would have to keep messaging in the department's Slack channel to notify Bill, but he now has a centralized place to do it.

Fantasy Football Draft

- Organizer

    - John is trying to find the best time to host his yearly Fantasy Football Draft, usually this is easy as his friends and family always seemed to have the same week off in October. However, this year it seems that everyone is busy the entire month. John makes a group chat and asks everyone when they can all come to his house and his responses were unhelpful, either “A few hours on Monday” or “I'll let you know later”. John began to get annoyed and researched a way to help. He discovers TimeScape and sends an invite link to the group chat. The results start flooding in with about 30 submissions and the app seamlessly calculates the top five best meeting times to allow the most amount of people to attend.
- Attendee
    - Mike has a busy schedule and was unable to think about what times during the week he would be available. He receives the invite link from John and is relieved to see how user friendly and easy it is to go step-by-step each hour of each day of the week. Now he is happy that John was able to pick the perfect time for their favorite yearly tradition.

5. Provide two examples of risk that could potentially impact this project. Explain how you would mitigate these risks if you were implementing your project as a software system.

Risk 1:
Lots of requests to the database to update it frequently (real time) may cause buffering/latency issues for users. We want to avoid this, so maybe we could implement a refresh function that calls updates on the database every 5 seconds or so to keep latency low, but giving the effect of “real time” updates.

Risk 2:
Communication risks are always possible in projects, so to avoid detrimental communication risks we will make sure to lay out all of our expectations and requirements at the beginning of the project. We will also stay in contact with each other via group messages and (ideally) weekly meetings.

6. Describe which process your team would use for requirements elicitation from clients or customers, and explain why.

We would use either GUI/Flow Chart or we would interview the customers. Some things would require interviews, such as which is the best medium of displaying the availability (bar graph, line graph, etc.). This would give us feedback on which is the most accessible and efficient format. The flow chart would help users understand how the process of TimeScape is intended to work.

Requirements Analysis

Use Case 1: Finding Optimal Meetings with TimeScape

1 Preconditions

- Users have registered accounts on TimeScape.
- Users have inputted their availability data for each day of the week

2 Main Flow

- User logs in to the TimeScape application.
- User navigates to the "Schedule Meeting" feature within the application.
- User selects the option to create a new meeting and specifies the list of attendees for the meeting. [S1]
- TimeScape processes the request and retrieves the availability data of the listed attendees.
- TimeScape calculates the optimal meeting times based on the availability of the attendees.
- TimeScape displays the list of suggested meeting times to the user.
- User reviews the suggested meeting times and selects the preferred time slot for the meeting. [S2]
- TimeScape finalizes the meeting schedule and sends invitations to the selected attendees.

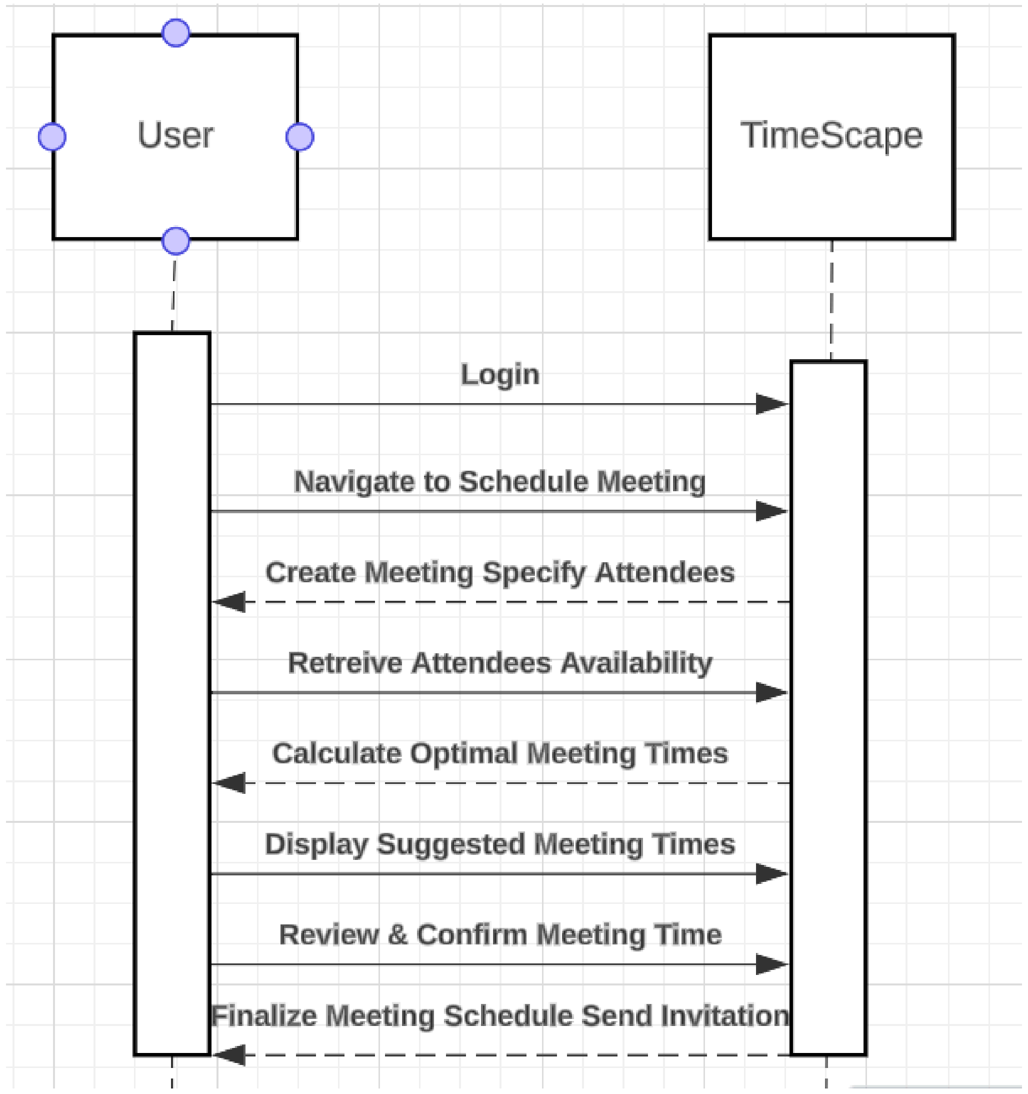
3 Subflows

- [S1] User provides the list of attendees for the meeting.
- [S2] User confirms the preferred meeting time from the list of suggested options.

4 Alternative Flows

- [E1] If none of the listed attendees have overlapping availability, TimeScape notifies the user that no suitable meeting times could be found. The user may choose to adjust the list of attendees or manually select a meeting time based on the individual availability of the attendees.

Model 1: Finding Optimal Meeting Times


User
TimeScape
Login
Navigate to Schedule Meeting
Create Meeting Specify Attendees
Retreive Attendees Availability
Calculate Optimal Meeting Times
Display Suggested Meeting Times
Review & Confirm Meeting Time
Finalize Meeting Schedule Send Invitation

Use Case 2: User Views Meeting Schedule

1 Preconditions

- User has registered an account on TimeScape
- User is logged into the TimeScape Web App
- User has previously scheduled meetings or has been invited to meetings by others

2 Main Flow

- User navigates to the “Meeting Schedule” section within the TimeScape application
- System retrieves the user’s meeting schedule from the database [S1]
- System displays the user’s upcoming meetings in chronological order
- User reviews the details of each meeting, including the date, time, attendees, and meeting topic
- User selects a specific meeting to view more details or to make modifications [S2]

- System provides options for the user to edit the meeting details, cancel the meeting, or join the meeting

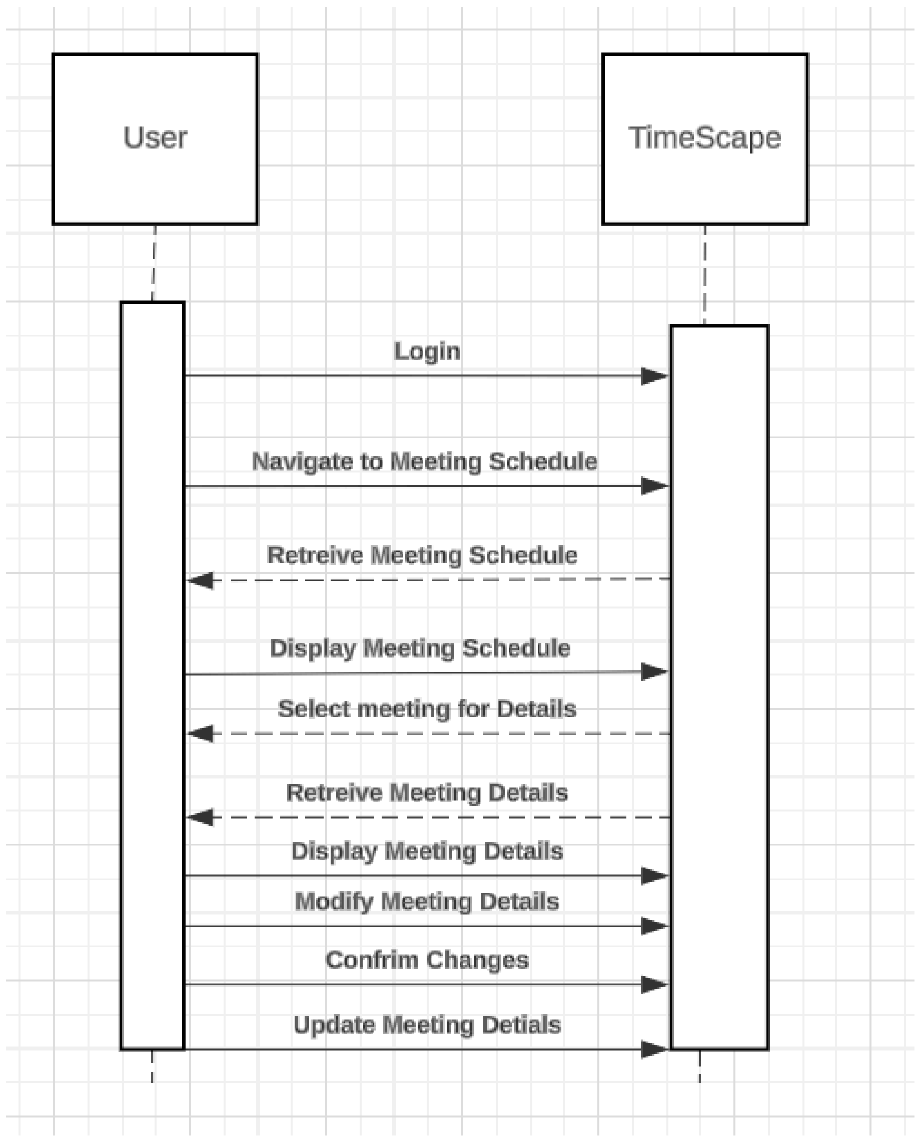
3 Subflows

- [S1] System retrieves the user's meeting schedule from the database
- [S2] User selects a specific meeting from the schedule to view more details

4 Alternative Flows

- [E1] If the user has no upcoming meetings scheduled, the system displays a message indicating that there are no meetings scheduled at this time
- [E2] If the user encounters any issues while viewing the meeting schedule, the system notifies the user and suggests retrying the action later

Model 2: User Views Meeting Schedule



Use Case 3: Participant accepts invite and indicates availability

1 Preconditions

- Participant must have received a valid invitation from the organizer
- Participant must have a registered TimeScape account

2 Main Flow

[S1] Participant logs in and accepts the invitation sent by the organizer to join the meeting
[S2] Participant indicates their available time slot(s) within the organizer's created meeting plan
[S3] TimeScape processes the request and updates the display to show the participant's indicated slots

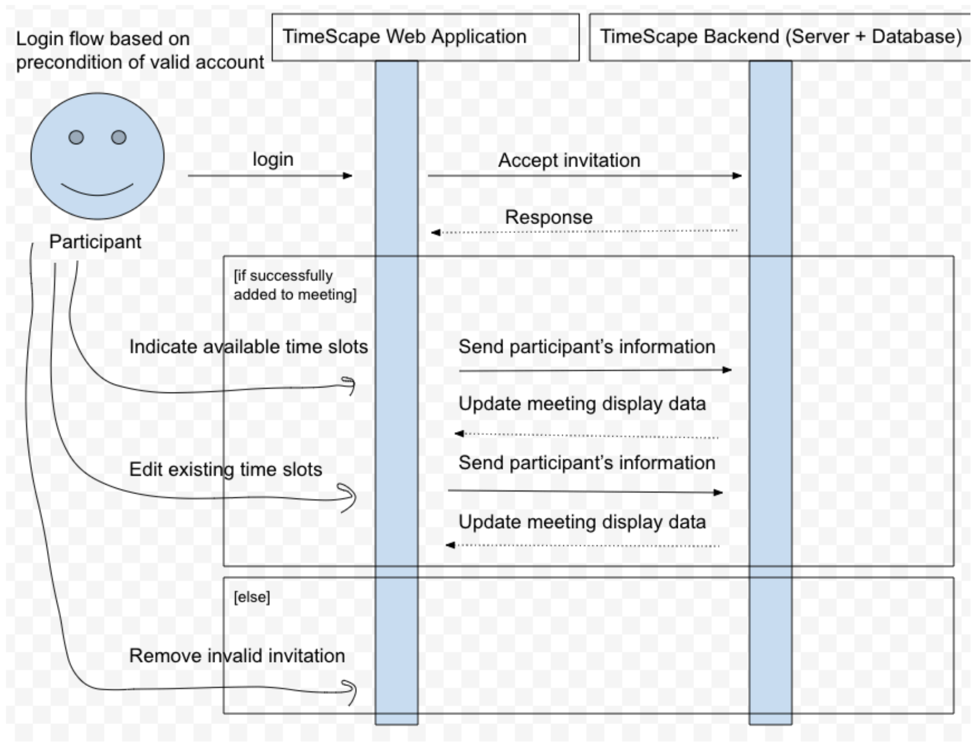
3 Subflows

[S1] Participant is either logged into their TimeScape account within the web application to accept, or clicks on the invite within their email to accept
[S2] Participant inputs their available time slot that falls within the meeting time range. Can edit or change as needed.
[S3] TimeScape's database for the meeting logs the participant's time, and uses it to update the display for the organizer and other participants to see

4 Alternative Flows

[E1] Participant has no available time slots within the meeting range set by the organizer
[E2] Not a valid invitation

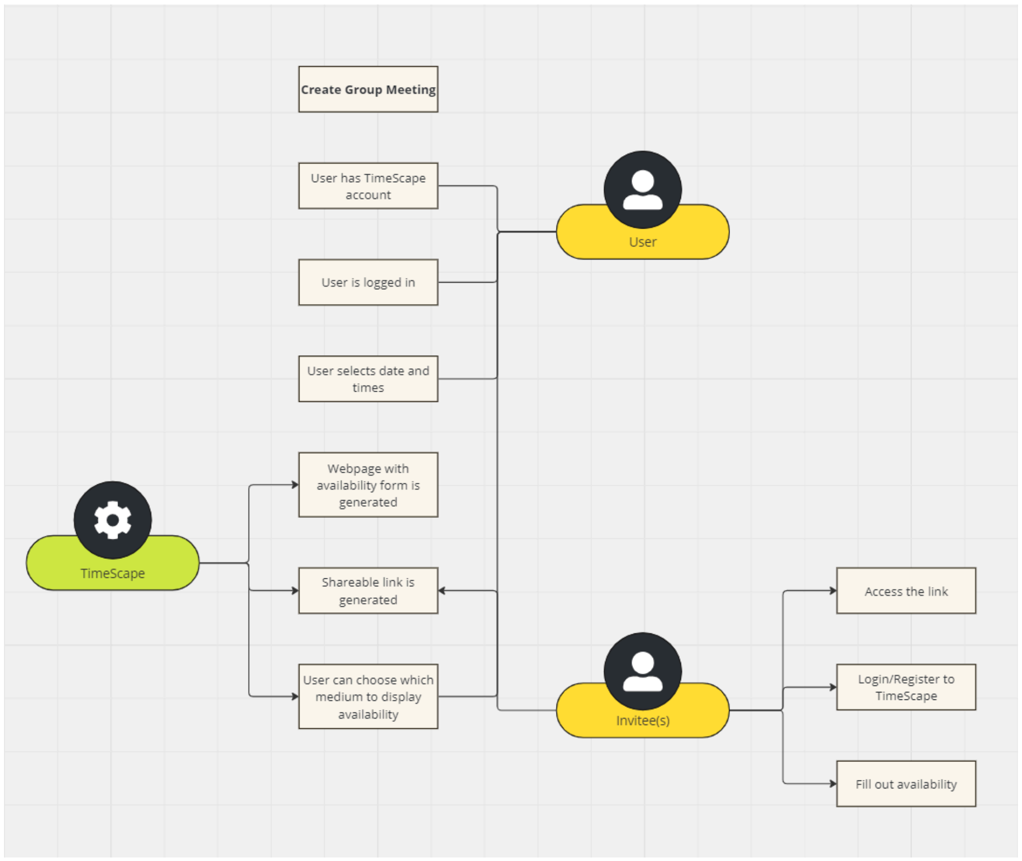
Model 3: Participant accepts invite and indicates availability


Login flow based on precondition of valid account
TimeScape Web Application
TimeScape Backend (Server + Database)
Participant
login
Accept invitation
Response
[if successfully added to meeting]
Indicate available time slots
Send participant's information
Update meeting display data
Edit existing time slots
Send participant's information
Update meeting display data
[else]
Remove invalid invitation

Use Case 4: Create an informal group meeting and share with friends

1. Preconditions
    a. User must have TimeScape account
    b. User is logged into the TimeScape Web Application
2. Main Flow
    a. User will select the day of the week they wish to find the best times for[S1].
    b. A page is generated based on the user's requirements with a visual table of available times [S2].
    c. Link is generated to share this page with user's friends [S3].
3. Subflows
    a. [S1] User will also select a window of hours (i.e. 9am-5pm) for invitees to input their availability.
    b. [S2] User can choose which medium to display availability (bar graph, table, etc.)
    c. [S3] Invitees will click the link and will be prompted to fill out their availability
4. Alternative Flows
    a. [E1] Invitee doesn't have an account, so they have to register first, then fill out availability.

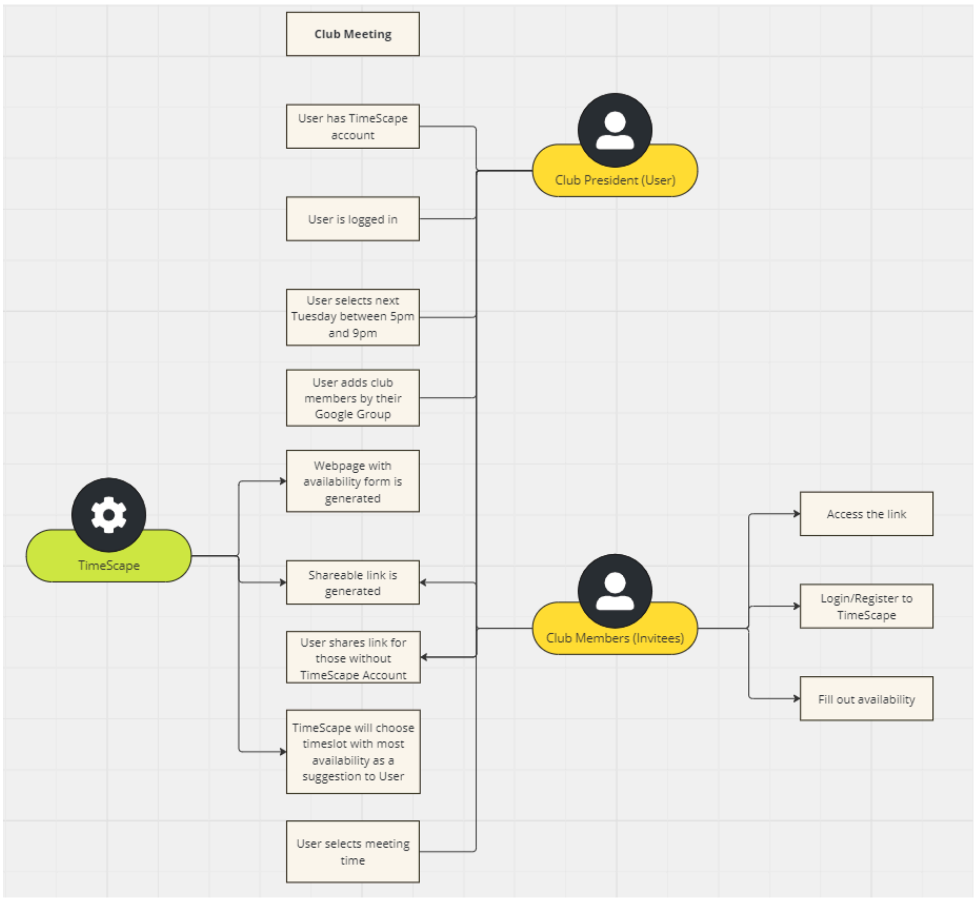
Model 4: Create an informal group meeting and share with friends


Create Group Meeting
User has TimeScape account
User is logged in
User selects date and times
Webpage with availability form is generated
Shareable link is generated
User can choose which medium to display availability
User
TimeScape
Invitee(s)
Access the link
Login/Register to TimeScape
Fill out availability

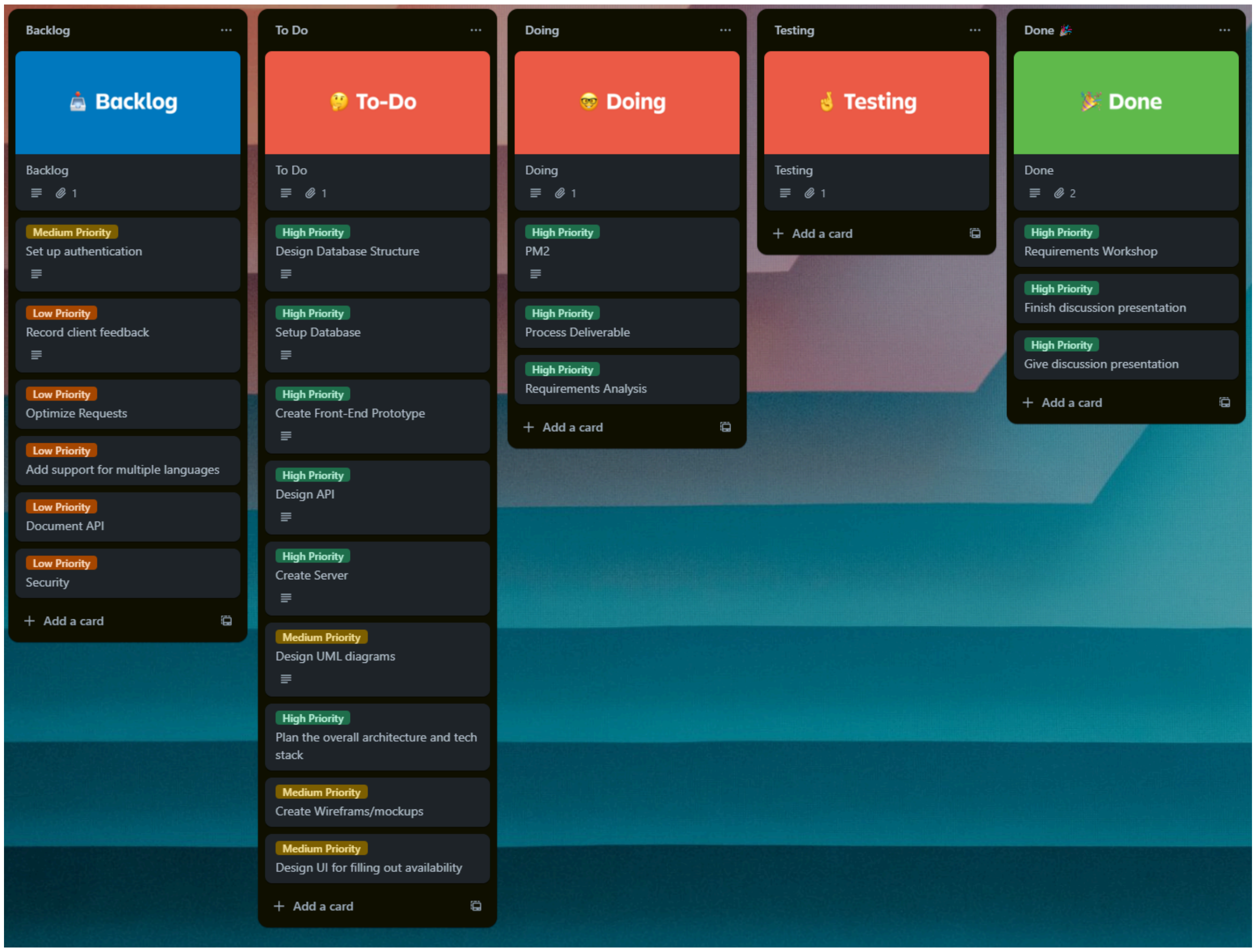
Use Case 5: Club president needs to organize meeting for next Tuesday evening

1. Preconditions
    a. User must have TimeScape account
    b. User is logged into the TimeScape Web Application
2. Main Flow
    a. User will select the date and time [S1].
    b. Webpage generated with availability form [S2].
    c. Invitees fill out their availability [S3].
3. Subflows
    a. [S1] User will specifically choose next Tuesday between 5pm and 9pm
    b. [S2] User adds all club members to the form using their Google group.
    c. [S3] TimeScape will give suggestions on best times to have the meeting (times with most availability).
4. Alternative Flows
    a. [E1] User sends out a shareable link to those without a TimeScape account

Model 5: Club president needs to organize meeting for Tuesday evening


Club Meeting
User has TimeScape account
Club President (User)
User is logged in
User selects next Tuesday between 5pm and 9pm
User adds club members by their Google Group
Webpage with availability form is generated
TimeScape
Access the link
Shareable link is generated
Club Members (Invitees)
Login/Register to TimeScape
User shares link for those without TimeScape Account
Fill out availability
TimeScape will choose timeslot with most availability as a suggestion to User
User selects meeting time

Process Deliverable



Backlog
- Backlog
- 1
- Medium Priority: Set up authentication
- Low Priority: Record client feedback
- Low Priority: Optimize Requests
- Low Priority: Add support for multiple languages
- Low Priority: Document API
- Low Priority: Security
- + Add a card

To Do
- To-Do
- To Do
- 1
- High Priority: Design Database Structure
- High Priority: Setup Database
- High Priority: Create Front-End Prototype
- High Priority: Design API
- High Priority: Create Server
- Medium Priority: Design UML diagrams
- High Priority: Plan the overall architecture and tech stack
- Medium Priority: Create Wireframs/mockups
- Medium Priority: Design UI for filling out availability
- + Add a card

Doing
- Doing
- Doing
- 1
- High Priority: PM2
- High Priority: Process Deliverable
- High Priority: Requirements Analysis
- + Add a card

Testing
- Testing
- Testing
- 1
- + Add a card

Done
- Done
- Done
- 2
- High Priority: Requirements Workshop
- High Priority: Finish discussion presentation
- High Priority: Give discussion presentation
- + Add a card

High priority tasks are labeled as such due to time restraints. When designing our project we are limited by the scope of the course, so we are prioritizing upcoming deadlines as high priority. These high priority tasks are also integral parts of our application so they require the most forethought and planning.

The medium priority tasks are the ones we know that come after the high priority. We need Login/Authentication, and a good design once we get our backend up and running, so it is important for us to think about those tasks while we are working on the ones before.

Low priority tasks are more so tasks that our application could live without. Ideally we would have those features, but they are not of utmost importance. These features would likely be added later in the project's lifetime.