Rundown

Team #: 4

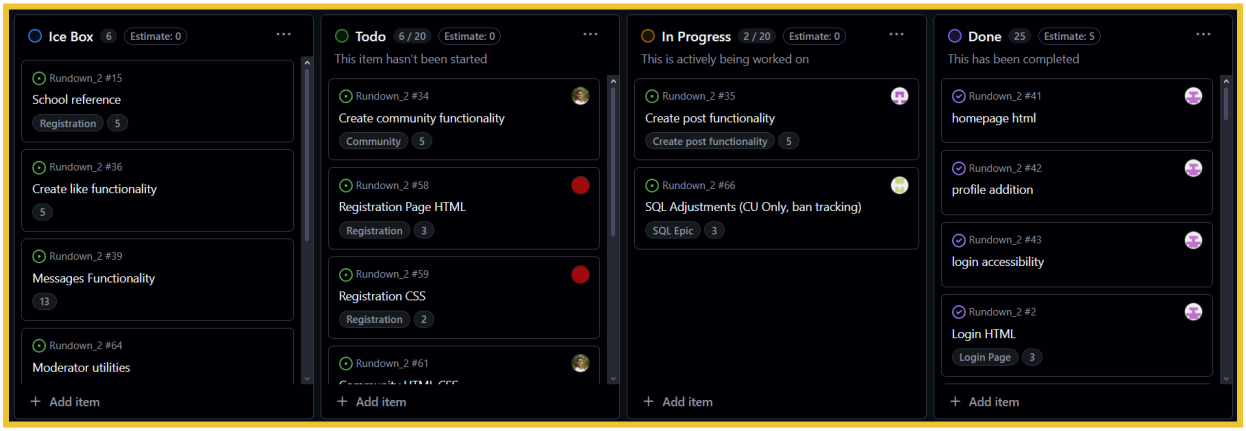
Team Name: CookedCoders

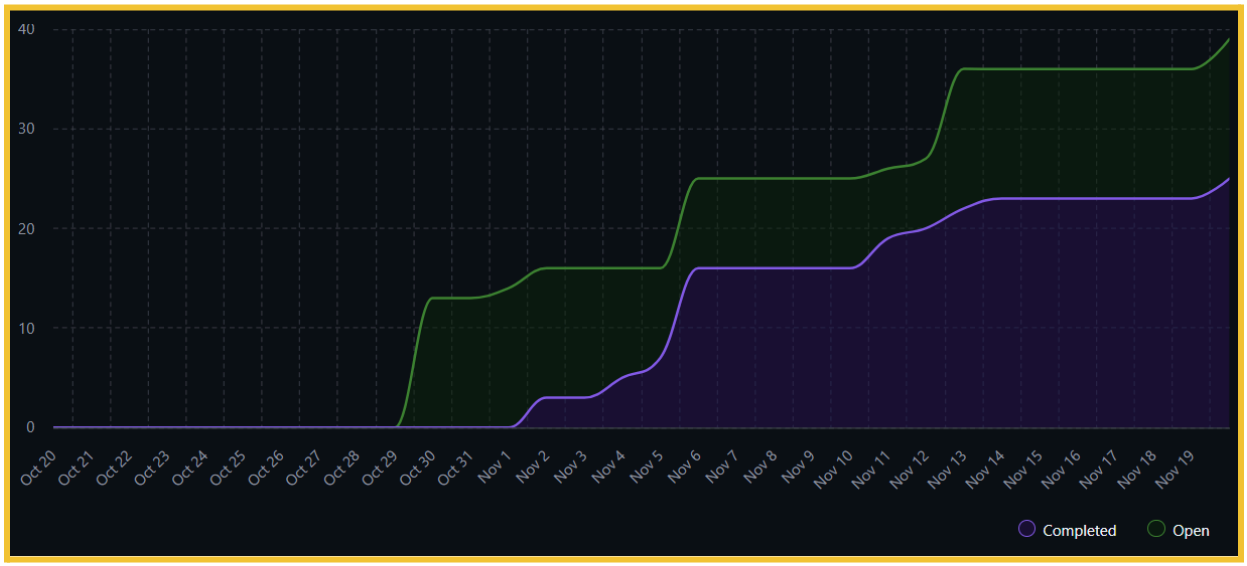
Discord: https://discord.gg/5cax3Fst

Team Members:

Name	GitHub	email
Alfred Whitemore	awhitemore	alwh8829@colorado.edu
Nikhil Pudtha	nipu345	nipu6271@colorado.edu
Dhilon Prasad	dhilon	dhpr4013@colorado.edu
Ryan Murray	ryanmm123	rymu8585@colorado.edu
Matvey Bubalo	MatveyBu	mabu6218@colorado.edu
Baden Miles	Bami6878	bami6878@colorado.edu
RJ Wiebe	RJ-Wiebe	rowi2520@colorado.edu

Github Repository: Rundown
Project Tracker: Project Board





Deployment: Our Site

Demo: Demo_Video

Application Description:

We are intending to build a college forum page that allows college students within their universities to connect with fellow students in a range of communities. This could range from finding housing, club/sports pages, or even general class and campus-wide communities. The page will allow students to find a secure, reliable, and widely used forum that connects them with students with similar interests.

Admins will control the page UI and moderator access for a given university, allowing them to choose colors and create certain communities. Moderators will control the specific functions within certain subtopics and moderate those channels, similar to Discord. The general user base (students) will then be able to join/get invited to specific channels that fit their needs. Some pages will be publicly available to the community like the housing page, lost id page, etc.

Audience:

The audience of the app will be college students. An ideal user would be an active student, with a desire to connect online and interact with communities on campus. The app would be built with the consideration of being accessible for that type of user including intuitive interactions while accessing multiple communities.

Vision Statement:

For university students who want to connect and expand. Rundown is a web-based forum that allows users to find lost student ID cards, look for subleases around campus, and

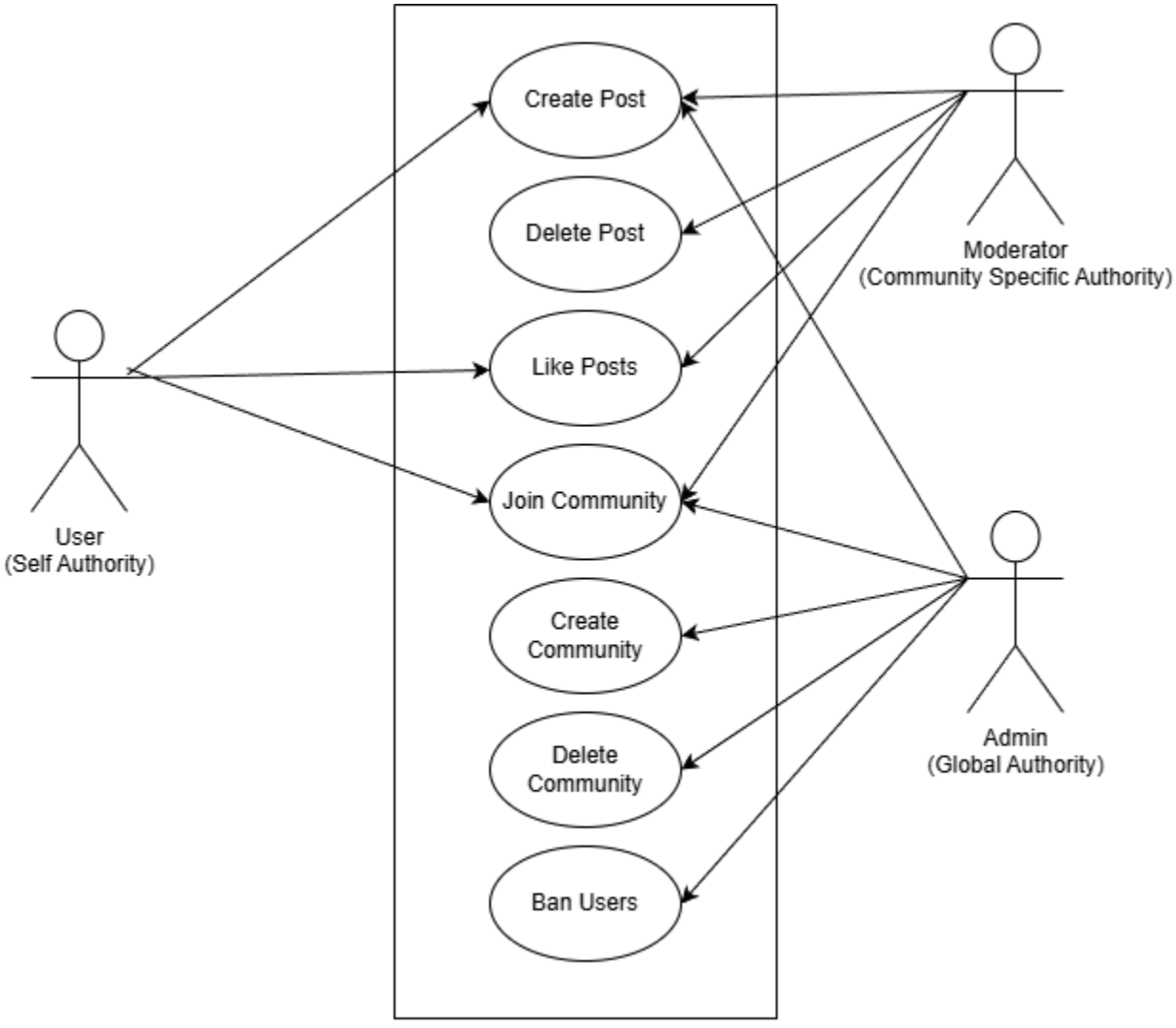
easily communicate with peers. College settings and students made unique.
Development methodology: We plan on using the Agile methodology

Communication Plan: We plan on using a discord server for the main communication source.

Meeting Plan:

Meeting w/ TA at 1:00pm Thursdays
Group meeting Fridays at 2:30 via discord

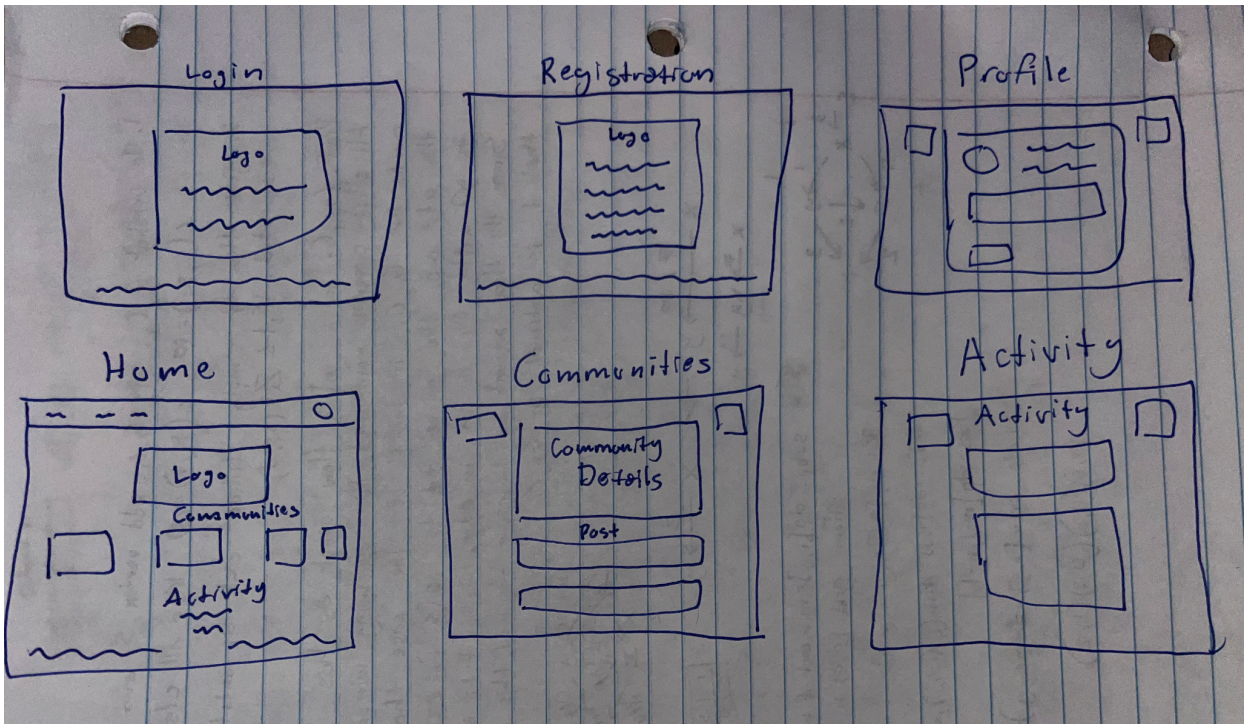
Use Case Diagram:



1. User (Self authority)

    - Username: user1
    - Password: user123
    - Role: Basic user with self authority
2. Moderator (Community specific authority)
    - Username: moderator1
    - Password: mod123
    - Role: Moderator for Housing community
3. Admin (Global authority)
    - Username: admin1
    - Password: admin123
    - Role: Administrator with global authority

Wireframes:



Extra Credit Work

1. Losing the Scope of the College-Focused Aspect - Critical
Risk: As Rundown scales, the platform may drift from its original goal of serving college communities.
Severity: Critical - loss of focus could alienate the primary user base.
Mitigation: Maintain feature review checkpoints to ensure all new ideas align with the college-community vision. Regularly gather student feedback to stay relevant to campus life.

2. Applying the App to Multiple Colleges - Moderate
Risk: Expanding to new campuses might make the app too generic or cause compatibility issues.
Severity: Moderate - risk of diluting community identity or facing technical challenges.
Mitigation: Design the backend to support multiple universities while keeping localized content features for each campus.

3. User Data Security - Critical
Risk: Exposure or misuse of private information from user profiles or posts.
Severity: Critical - breaches would severely harm user trust and platform credibility.
Mitigation: Implement strong password hashing, encrypted data storage, and strict access controls. Regularly conduct security audits.

4. Chat Moderation - Critical
Risk: Unmoderated or toxic discussions could damage the app's community environment.
Severity: Critical - poor moderation directly impacts user experience and growth.
Mitigation: Integrate automated moderation tools (e.g., flagging/reporting) and designate verified moderators for each college community.

5. UI Scalability and Accessibility - Moderate
Risk: Interface may not adapt well to different devices or accessibility needs as the app grows.
Severity: Moderate - may affect usability and inclusivity.
Mitigation: Follow accessibility standards, conduct usability testing, and design responsive UI components.

Assignments:

Front end:
- RJ Wiebe
- Baden
- Nikhil Pudtha

Back end:
- Ryan

Full stack:
- Dhilon
- Alfie
- Matvey

Contributions:

Alfie: I picked up a more full-stack role during this project. My main priority was to complete the registration page and its functionality including email verification. In addition, I worked alongside Dhilon

to create the test cases and ensure they worked as we developed more features. With this, I mainly worked on connecting our front-end to the back-end, using technologies like Node.js, chai, mocha, and SQL when adding new users to the database.

Nikhil: I focused primarily on front-end development throughout the project. I created the HTML layout for the admin privileges page, which allows administrators to add and delete communities. This included structuring the page, ensuring intuitive navigation, and preparing it for integration with backend logic. I also built the HTML layout for the post pages within the app, designing the structure for how posts are displayed to users.

Dhilon: Along with Alfie, I contributed more in a full-stack sort of way. Primarily, I helped build the tests and make sure all of the lab work was completed, along with passing these test cases and returning HTML versus JSON in some cases versus others. Part of that was connecting the register process with an email using nodemailer, an exterior module that I researched and implemented using my gmail and passkey, which included updating the SQL to include tokens. I also built the full “your communities” page (HTML + functionality), the “explore communities” page, the individual “community” pages complete with descriptions and like/post functionality, along with linking our four major home page communities. Additionally, I created our logo/favicon and footer and header partials.

Ryan: I designed the original navbar using HTML, CSS, and JavaScript. My version had Home, Communities, Notifications, Resources, a profile icon, a search bar, and a logout button. While merging, conflicts arose with the original navbar, so parts of mine were incorporated into the final version. From my design, we kept the profile icon, the logout button, and the communities tab. I also created the add/delete communities functionality, allowing admins to create new communities and delete those that violate guidelines. I added a shield that redirects admins to an existing community when they create one with a duplicate name.

Matvey: I handled most of the SQL work: I built/edited the table setup for users, communities, posts, joins (membership), and likes, and I spent time making sure the schema actually matched the role/ban rules the app needed. I continuously adjusted the schema to match the current scope of the application. I also populated it with sample data using JavaScript due to the need of hashing the passwords for user security. I kept up documentation in the form of an ERD diagram in the same folder as the SQL setup. I handled the cloud deployment of our application on Render as well. I deployed the repo as a web service and a database and set it to redeploy on each commit to the main branch to keep it always up to date.

Baden: I began with the foundational implementation of both Handlebars and Node.js. After this, I focused on the login page and profile page. I created the HTML code and CSS for both, while adding functionality to the profile page to allow users to change their profile picture, bio, username, name, and password. I also ensured a connection to the database for permanent changes and ensured that they were reflected to the user. Along with this, I spent time organizing the project report and presentation to ensure we met all the requirements.

RJ:

For each of the three features below:

- The test plans should include specific test cases (user acceptance test cases) that describe the data and the user activity that will be executed in order to verify proper functionality of the feature.
- The test plans should include a description of the test data that will be used to test the feature.
- The test plans should include a description of the test environment (localhost / cloud) that will be used to test the feature.
- The test plans should include a description of the test results that will be used to test the feature.
- The test plan should include information about the user acceptance testers.
- The actual test results are based on observations after executing the tests.

1. User should be able to login with correct credentials.
    a. The user will submit their username/email and password. Scenario is user is trying to log back into their profile
    b. We test the feature with the database and whether the password matches the hash
    c. Environment: local host for now until cloud is discussed
    d. Expected results: Pass verification and get back to home screen with fully updated profile
    e. Testers are any users regardless of role but must have registered previously, we created fake profiles to simulate this (they are able to log in and their role status is displayed) and we also created accounts through the frontend with our own gmails (which logged in correctly) to test this.
2. User registration fails when the user provides invalid credentials.
    a. User will submit username, password, and school email into a form that submits to try to create a new user object
    b. Use an api to verify school email is real, verify username is unique.
    c. Environment: local host for now until cloud is discussed
    d. The results should be a newly created user within their corresponding university college rundown page
    e. We create fake users in our test cases to register with a username or email that's already taken, and we return a JSON error, and in the frontend, when we try to register with incorrect information, an error message displays in red at the top of the page

3. If a user is banned, the user is prompted with a message and there are no missing components.
    a. A user has been banned from a community by a moderator or admin for profane language in any sense or cyberbullying
    b. Test data: banned user's username, community, banned message
    c. Environment: local host for now until cloud is discussed
    d. Expected results are the banned user getting a message that they have been banned and they don't have access to certain areas
    e. We tried banning a user only through the frontend from our moderator account, and then we logged out and logged in with the fake user that we banned with member access and when they tried to access the community, a page popped up that said they have been banned and gave them a redirect to home. We also saw the data mentioned above appear in the database when we queried information from the backend (added functionality of the frontend button). Banning from the whole app is a future implementation