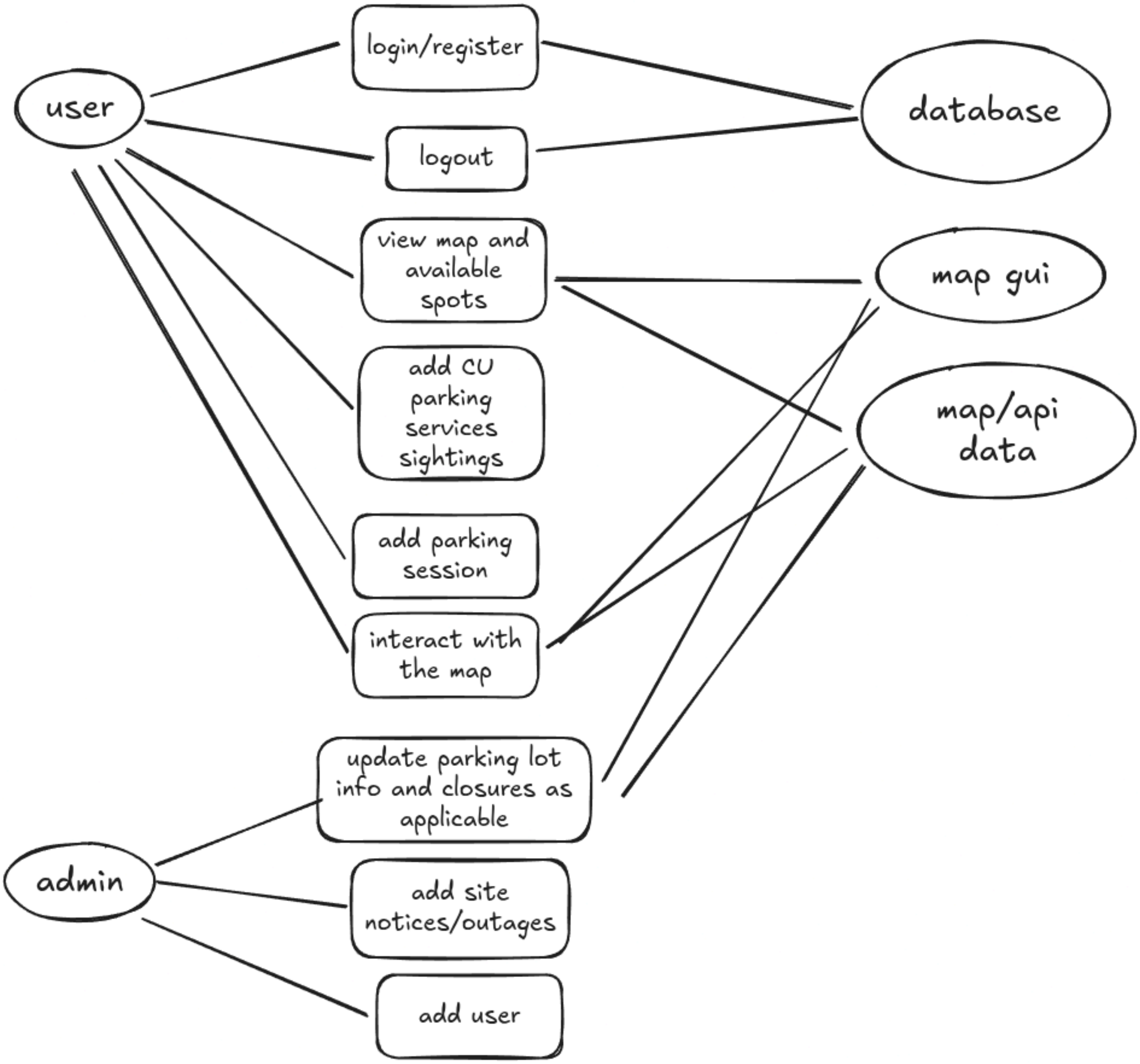
1. Team number: 4
2. Team name: Rest Them Wheels
3. Team members:

First	Last	GitHub Username	Email address
Sully	Harrer	harrly14	suha4835@colorado.edu
Dolgormaa	Sansarsaikhan	DolgormaaS	dosa9941@colorado.edu
Carys	Gardner	carysGard	caga2517@colorado.edu
Xander	DuBois	aldu9080	aldu9080@colorado.edu

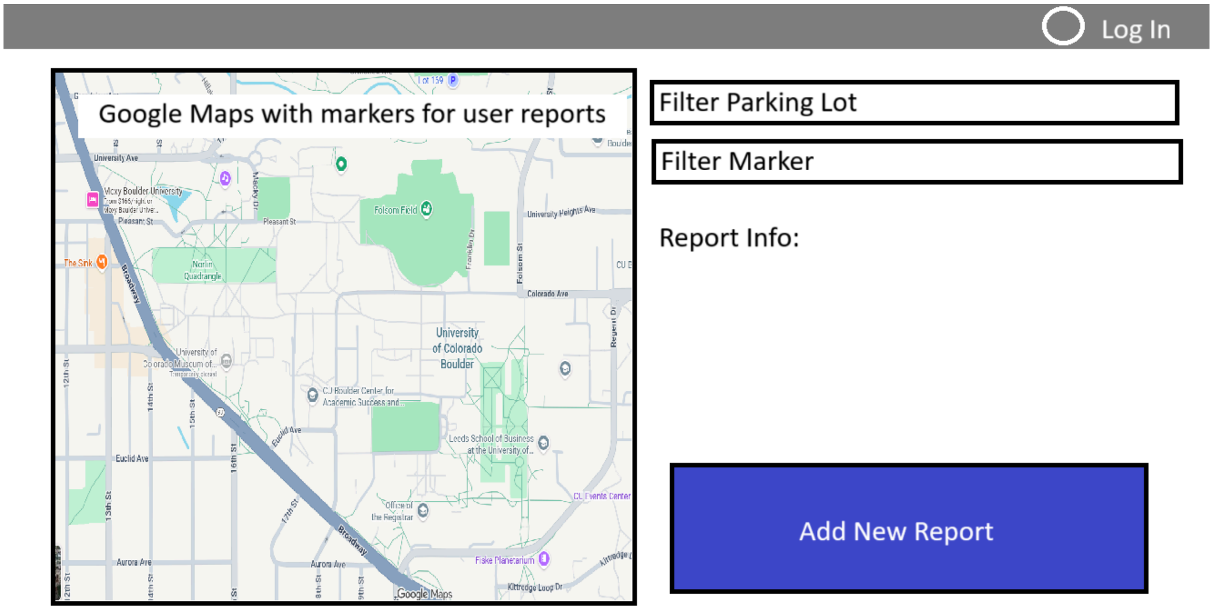
4. Application name: Campus Parking Helper
5. Application description: A user reported map displaying available parking spots on campus. Users will be able to navigate around the campus map to choose their desired parking spot to log their session. Users will be required to provide their email to be able to use the app to login.

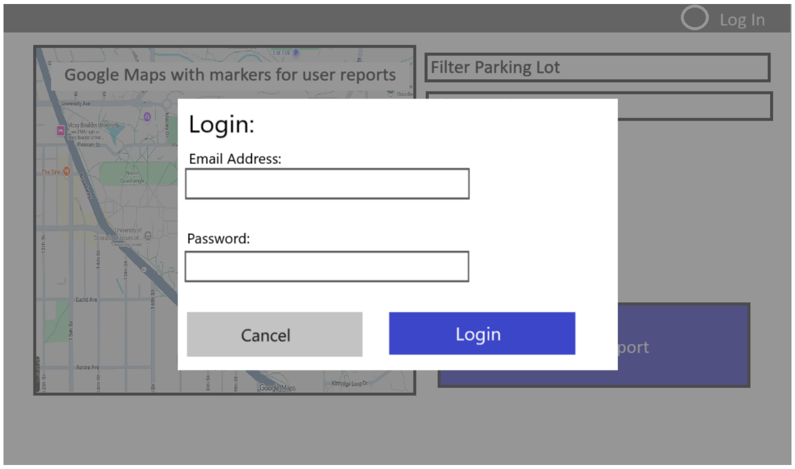
6. Audience: CU students, staffs, and visitors
7. Vision statement: For CU students, staff, and visitors, who seek to avoid getting parking violations and tickets from CU parking service, as well as find open spots on campus and avoid lots that are completely full. Unlike CU Parking Services, our product has 24/7 availability, more fine-grained accuracy, and compatibility with pay-to-park lots.
8. Project repo
9. Development plan:We will use a Kanaban board to organize to-do items in order of priority and to divide tasks between team members. We will meet as needed rather than at a planned date and time. Jira Kanban board
10. Communication plan: We will communicate over Discord.
11. Meeting plan:
    - Team meetings: As needed
    - First TA meeting: 6:30pm on Wednesday

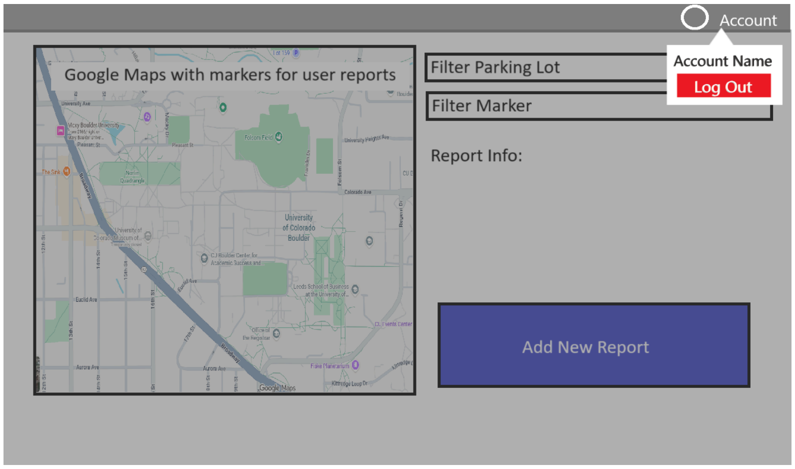
12. Use case diagram

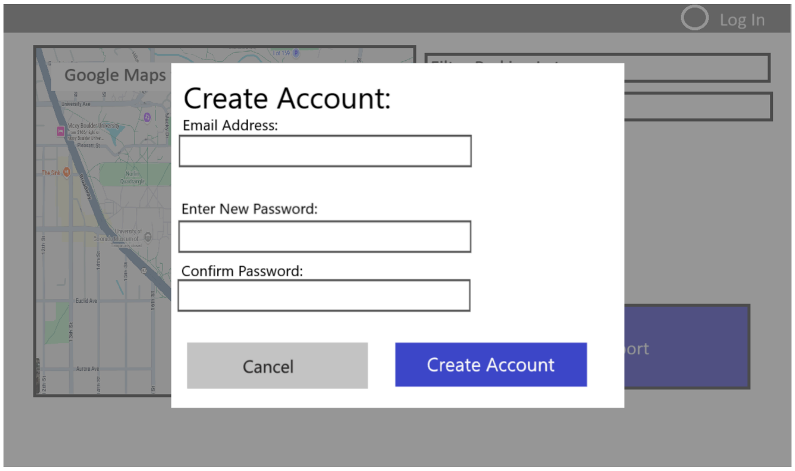


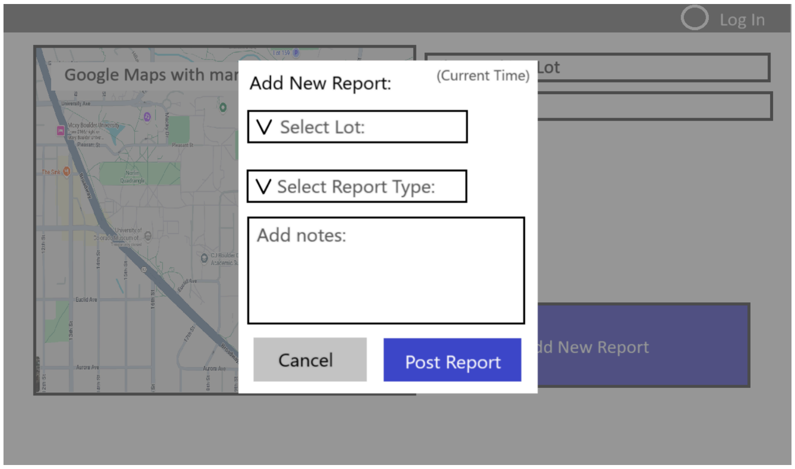
13. Wireframe:











Extra credit:

Risks:

1. Risk: Denial of Service attack
   Severity: Moderate
   Mitigation strategies: Require users to login with their email and verify with their phone number, so that the attacker can't create alternative accounts over and over again to attack.
2. Risk: Data breach
   Severity: High
   Mitigation strategies: MFA authentication, require strong password, encryption.
3. Risk: False report
   Severity: Low
   Mitigation strategies: Only require the user to log one session at a time.
4. Risk: API goes down
   Severity: Low
   Mitigation strategies: Display a screen that informs the user that API is down and that they have to wait. This is beyond our control.
5. Risk: SQL injection
   Severity: High
   Mitigation strategies: User input shouldn't be able to execute SQL queries. We don't want the attacker to obtain user information.