Climb! Documentation

Winter 2020

1 Specification

The idea of a digital scorecard like Climb! was proposed climbers on our team. The initial discussion amongst the members of our team demonstrated a legitimate need among competitive climbers for a better climbing experience. Traditional climbing competitions with paper scorecards are frustrating for both climbers and event supervisors. From a logistics standpoint, logging routes and verifying scores is a tedious, error-prone process. Shown in Figure 1 is a problem scenario following a hypothetical climber. Therefore, we hypothesized that a digital scorecard could streamline a climbing competition while maintaining its integrity.
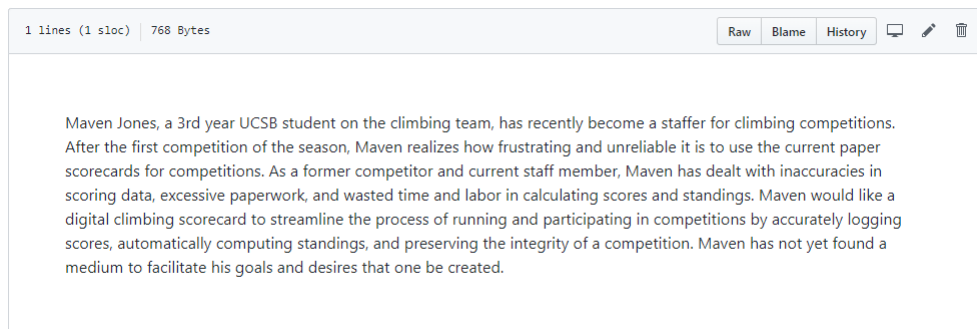

Figure 1: Problem Scenario

To validate our hypothesis, some of our teammates asked fellow climbing team members for their opinions regarding the Climb! idea through a verbal questionnaire. Our questions were the following:
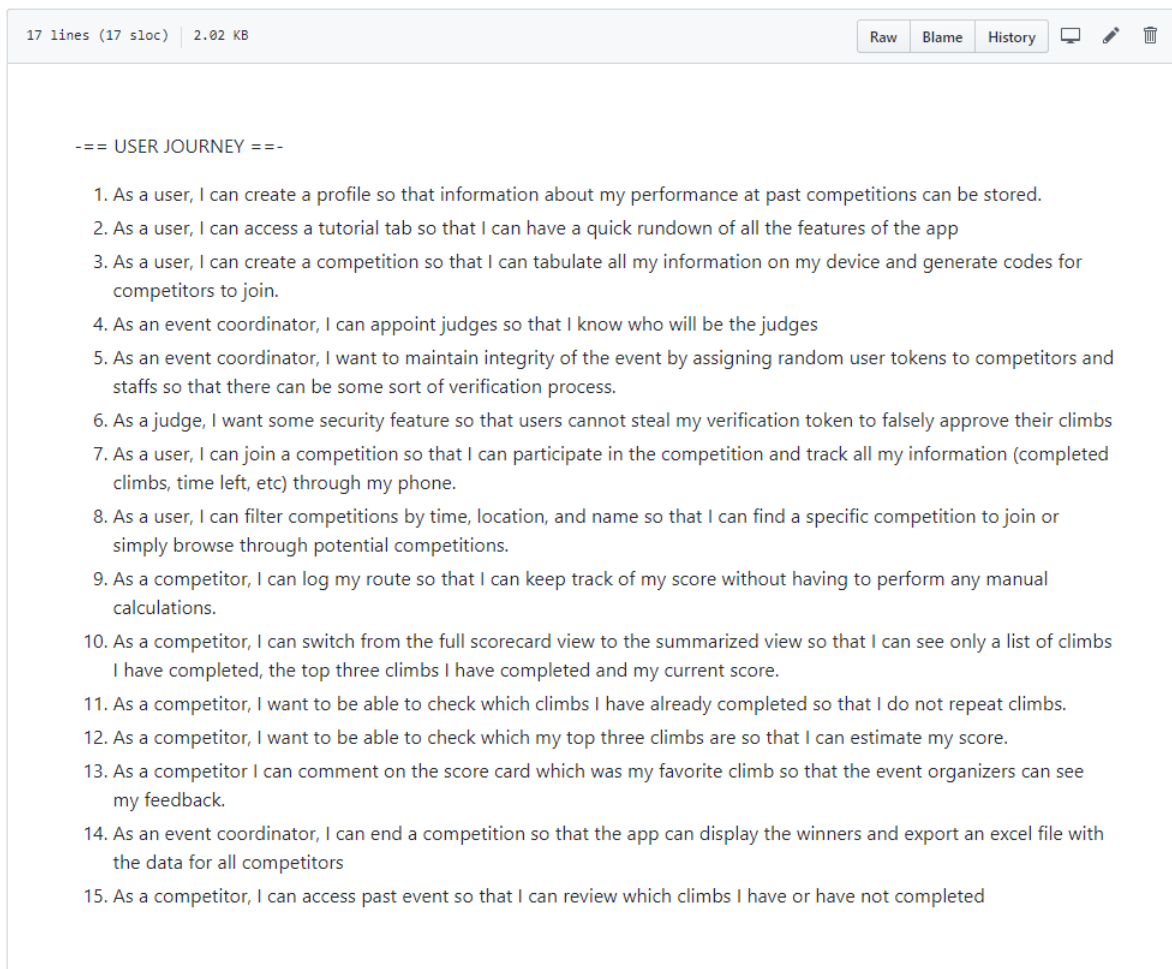
- Would you prefer a paper or digital scorecard for climbing competitions?

- Do you have access to devices during climbing competitions?

- Would you prefer a mobile application or a web application for a climbing scorecard?

The overwhelming response to to this simple questionnaire was that a digital scorecard was preferred. Although most of our participants said that they would prefer to use a scorecard on their phones, they mentioned that they would prefer a web application because they didn't want to download an app onto their phones.

Based on our initial user analysis, we determined that our users need the following in a digital scorecard:

- Ease of logging

- Ease of verification

- Accessibility of results

These user needs influenced the first set of user stories for Climb! in the form of the user journey.


```
17 lines (17 sloc)   2.02 KB                                    Raw   Blame   History   💻  ✏  🗑
```

-== USER JOURNEY ==-

1. As a user, I can create a profile so that information about my performance at past competitions can be stored.
2. As a user, I can access a tutorial tab so that I can have a quick rundown of all the features of the app
3. As a user, I can create a competition so that I can tabulate all my information on my device and generate codes for competitors to join.
4. As an event coordinator, I can appoint judges so that I know who will be the judges
5. As an event coordinator, I want to maintain integrity of the event by assigning random user tokens to competitors and staffs so that there can be some sort of verification process.
6. As a judge, I want some security feature so that users cannot steal my verification token to falsely approve their climbs
7. As a user, I can join a competition so that I can participate in the competition and track all my information (completed climbs, time left, etc) through my phone.
8. As a user, I can filter competitions by time, location, and name so that I can find a specific competition to join or simply browse through potential competitions.
9. As a competitor, I can log my route so that I can keep track of my score without having to perform any manual calculations.
10. As a competitor, I can switch from the full scorecard view to the summarized view so that I can see only a list of climbs I have completed, the top three climbs I have completed and my current score.
11. As a competitor, I want to be able to check which climbs I have already completed so that I do not repeat climbs.
12. As a competitor, I want to be able to check which my top three climbs are so that I can estimate my score.
13. As a competitor I can comment on the score card which was my favorite climb so that the event organizers can see my feedback.
14. As an event coordinator, I can end a competition so that the app can display the winners and export an excel file with the data for all competitors
15. As a competitor, I can access past event so that I can review which climbs I have or have not completed

Figure 2: User Journey

Therefore, the features we ended up committing to for Climb! were the following:

- Competitors can create a competition

- Competitors can join a competition

- Competitors can log routes

- Competitors can have their logged routes verified

- Competitors can view their past logged routes for a competition

- Competitors can view top climbs

- Competitors can view their tentative final score

With these users, stories, and features, we began the design phase of our product.

2 Design

2.1 First Ideas

Early in the quarter, we wanted to make sure that every team member would have the opportunity to contribute code; as a result, we wanted to split our team into two subgroups, one of which would be focusing on developing the frontend, including

- Designing the user interface

- Creating the forms

- Constructing the API requests

- Handling the API responses,

and the other of which would be focusing on developing the backend, including

- Creating the endpoints

- Handling the API requests

- Updating databases

- Constructing the API responses.

We hoped that we could develop in parallel, although we knew that there would be significant overhead in communication between subgroups.

2.2 Architectural Design

We knew that we would need modules for users and competitions because users would be registered in a competition and would have a score for that competition.
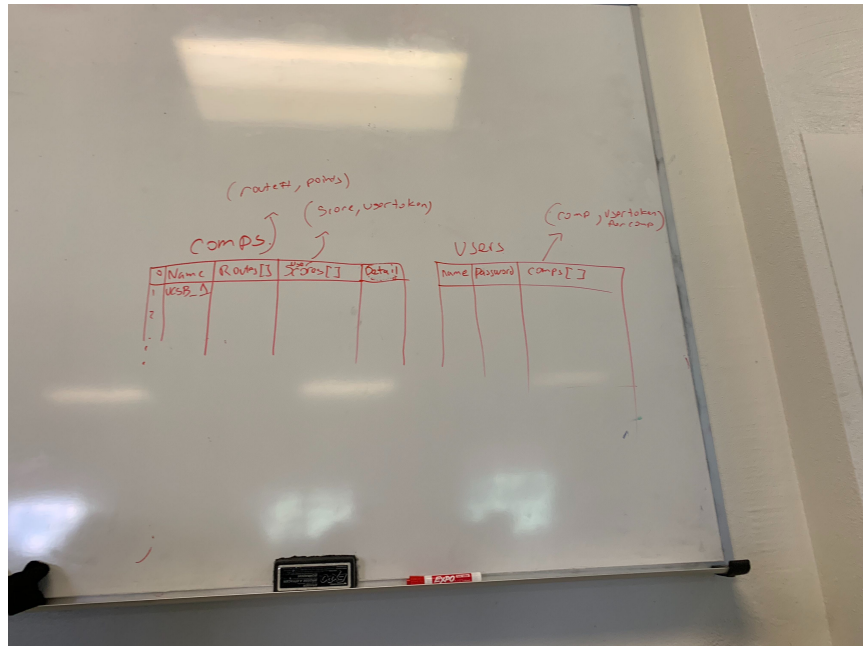
2.3 Detailed Design

Figure 3: Early Design of Competition and User Modules

As shown in Figure 3, we originally designed tables for competitions and users, where the competitions table consisted of the fields

- ID

- Name

- List of Problems

- List of Tuples (User, Score)

- Details

and the users table consisted of the fields

- ID

- Username
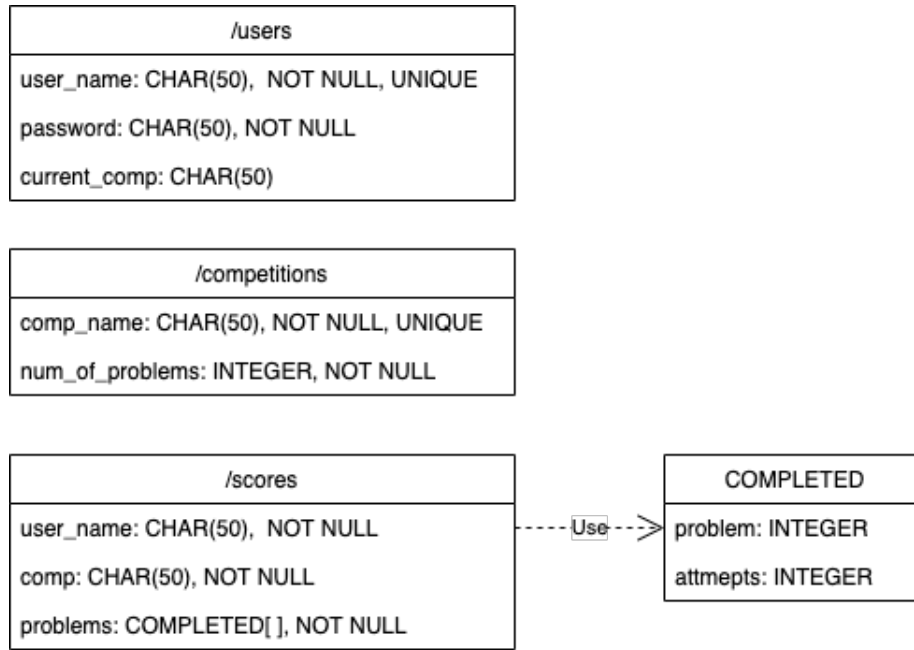
- Password

- List of Competitions.

Figure 4: Later Design of User, Competition, and Score Modules

However, during our implementation, we realized that this would not be easy to maintain, especially the relation between users and scores. Therefore, we designed yet another table for scores, as can be seen in Figure 4. Each score would be represented by the fields

- Username

- Competition

- List of Completed Problems (Number, Attempts)

This module design was used throughout our minimum viable product (MVP) and our final. The only difference was that we fixed all the relations in our implementation after our MVP demo.

2.4 UI Design

Climb! is a digital scorecard; as a result, we wanted to design a clean, familiar user interface to acquire and retain as many users as possible. The actual scorecard would look like Figure 5 and the results like Figure 6.

First Last
Auth Token

Log a Route

Scorecard			
Route	# Attempts	Initial 1	Initial 2
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			
18			
19			
20			
21			
22			
23			
24			
25			
26			
27			
28			
29			
30			

Log a Route

Figure 5: Early Design of Digital Scorecard

First Last
Auth Token

List of Climbs Completed	
Route	# Attempts
2	1
3	1
4	1
7	1
11	2
22	2
24	4
25	7
27	6

Top Three Climbs		
Route	# Attempts	Points
24	4	$24 - \dfrac{4}{100} = 23.96$
25	7	$25 - \dfrac{7}{100} = 24.93$
27	6	$27 - \dfrac{6}{100} = 25.94$

Total Score = 23.96 + 24.93 + 25.94 = 75.83

Figure 6: Early Design of Results

We ended up modifying these initial sketches in order to hide unnecessary information and provide the user with exactly what he or she wants. For example, we showed only the "List of Climbs Completed" because there would be no need to fill an entire scorecard (we wanted to use a form to log one problem at a time). Furthermore, we wanted to hide the work needed to calculate results because the final score was all that mattered to the user.

With these design ideas in mind, we proceeded to implement Climb!

3   Implementation

3.1   Framework

We decided that our frontend team would use React and that our backend team would develop with Node.js. Our frontend and backend web applications can be found in their own directories in our repo. In hindsight, JavaScript frameworks were probably not the best choice because there was a steep learning curve for many of us which distracted us from our goal of meeting user needs.

3.2   Challenges

We wanted to dedicate an entire subsection to our technical difficulties with JavaScript frameworks. Our roadblocks included the following:

- Simultaneously running frontend and backend on localhost: we circumvented the problem of running two web applications on the same port by pushing each iteration of the frontend and backend to Heroku. Until we wrote unit tests, it was difficult to spot errors quickly because the deployment process would take a couple of minutes. We also had to use the `git subtree` command to push the frontend and backend directories in our repo, which was quite tedious.

- Routing: we had never worked with multi-page web apps in React before. We used react-router and react-router-dom to create a multi-page web application.

- Maintaining session state: At first, we couldn't keep track of whether or not a user was logged in to provide the appropriate user experience. We sought Ekta's advice as well as scoured online documentation. We settled on using `localStorage`, which was beneficial because we knew whether or not a user was logged in as well as knew what the username and current competition were to make it easier for the users to fill out forms.

- Making API calls: We did not know how to prevent user-inputted variables from being reset to `null`. We used `preventDefault()`, which resolved our issue. We could finally make API calls.

- Working with tuples in SQL: We often had to parse and split strings to pass the appropriate JSON response back to the frontend.

4 Testing

4.1 Unit Tests

The frequent deployment of our frontend and our backend to Heroku caused many errors to fall through the cracks, many of which were only noticed during our MVP demonstration (when our entire backend crashed). We decided to write and automate unit tests for the backend to augment the process of handling errors. We used the Mocha.js testing framework to test several API endpoints that were accessed by the frontend. Our unit tests allowed us to write code to handle `null` values, invalid problem numbers, invalid witness usernames, and invalid credentials.

We then automated the testing process using GitHub actions. We set up a custom workflow that would automatically install dependencies, start the web server, and run the unit tests.

4.2 User Acceptance Tests

Our first and foremost goal throughout the software development process to provide meaning to our users. After gaining the seal of approval from our teammates who climbed, we made a simple questionnaire for other members of UCSB's climbing team. The questionnaire consisted of the following prompts:

- On a scale from 1 to 5, how would you rate the look and feel of our UI?

- On a scale from 1 to 5, how familiar is the UI to you?

- On a scale from 1 to 5, how likely are you to use Climb! during climbing competitions if you had a choice?

- Any other thoughts?

Out of 7 participants, the average responses to the questions in respective order were around 3, 4, and 4. We used this feedback to modify our original scorecard design to make it more closely resemble the paper scorecard and make our UI more mobile-friendly (after all, it wouldn't make sense for our users to bring laptops to competitions for scoring).
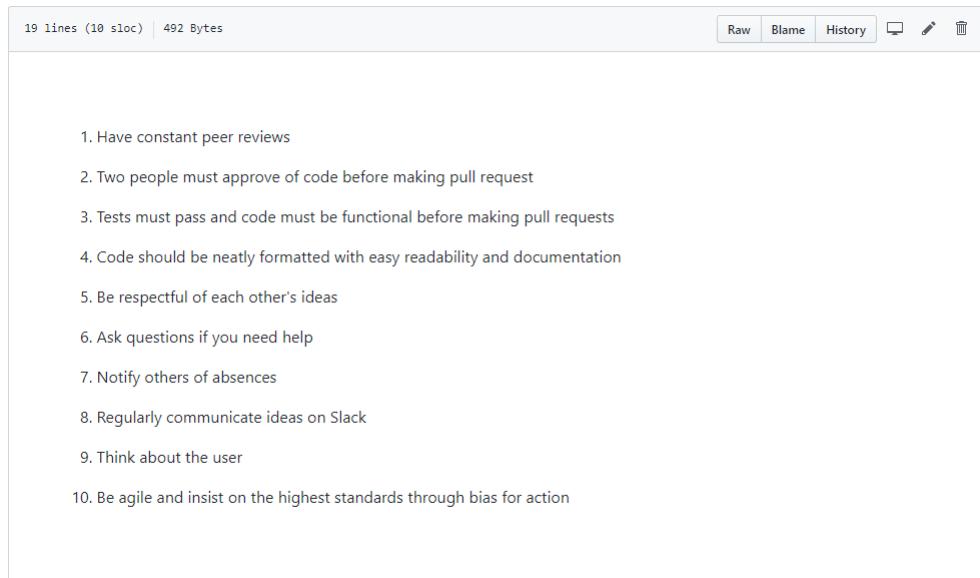
5 Team

5.1 Norms

Figure 7: Initial Team Norms

5.2 Roles

Figure 8: Current Team Roles

5.3 Sprints

This quarter consisted of six sprints, one of which was the final sprint.

5.3.1 Sprint 1

The goal of this sprint was to develop a "Hello, World!" web application using our stack (see **Section 3**) as well as to shortlist user stories leading up to our MVP. Our major development activities included

- Becoming acquainted with React and Node.js

- Creating "Hello, World!" web applications using React and Node.js (we ended up making two web apps because our team was large)

- Prioritizing user stories in the product backlog

5.3.2 Sprint 2

The goal of this sprint was to deploy our "Hello, World!" web applications to Heroku. Our major development activities included

- Deploying web applications to Heroku using `git subtree`

- Preparing for our first retro

We held our first retro after this sprint, where we decided to implement code documentation and reviews to ship better code and coordinate development between the frontend and backend teams.

5.3.3 Sprint 3

The goal of this sprint was to turn our "Hello, World!" web application into Climb!. Our major development activities included

- Creating a homepage and navigation bar for Climb!

- Creating database tables for the components shown in Figure 3

- Creating endpoints to perform basic CRUD operations on data

5.3.4 Sprint 4

The goal of this sprint was to implement all MVP user stories and demo our MVP. During our sprint planning meeting, we committed to the following set of features:

- Signing up

- Logging in

- Registering for a competition

- Logging routes for a competition

- Viewing scores for a competition

Our major development activities included

- Implementing forms for each of our MVP stories

- Creating endpoints for each of our MVP stories

- Demoing our MVP to our peers and mentors

- Reviewing feedback from our peers and mentors

After some hiccups with our MVP demonstration, we held our next retro, during which we decided that our experiment was only marginally beneficial. We decided to switch up the roles going forward to facilitate the development effort.

5.3.5 Sprint 5

The goal of this sprint was to refine our web application based on feedback from our MVP. Our major development activities included

- Handling errors in the backend

- Implementing unit tests

- Speaking to climbers for user acceptance tests

5.3.6   Final Sprint

The goal of this sprint was to implement our final stories and prepare for our product presentation. During our sprint planning meeting, we committed to the following set of features:

- Maintaining session state

- Verifying logged scores

- Viewing historical routes for a competition

- Viewing top climbs and scores for a competition

As of writing this documentation, we have not yet held our final retro nor have we presented our product.

6　Acknowledgements

We used the following technologies and referred to their documentation over the course of the project:

- React and React Router

- Axios

- Bootstrap

- Node.js

- PostgreSQL

- Mocha.js

- GitHub and GitHub Actions

- Heroku