CliffBuddy

Business Requirement Document

Team Name: Navite.io

Hunter Lewis (*Team Leader*)

Jay Park

Jerry Belmonte

Ji Park

California State University: Long Beach

CECS 491A SEC 05

10/6/21

Table of Contents

I. Executive Summary

1.1 Problem

There exists a lack of tooling to help climbers improve their skills given useful information to document, share, and receive feedback from others. Some major components that currently exist would be to retrospect, receive in-person feedback, and continuous practice. There doesn't exist an application that will provide ease of access for climbers, whether they are onsite or not. This problem unfortunately applies to climbers of all skill levels and experience. The product that Navite.io is creating would be solving such an issue.

1.2 Value Proposition

Cliffbuddy is a web application that will be planning on targeting:

- Indoor and Outdoor Climbers (Mixed and Bouldering)
- Mainly United States Based Climbers
- Novices and Veterans who want to share ideas within a social environment

We have spoken to various members who participate in the sport recreationally and have determined there is a lack of software that provides a platform that incentivizes collaboration. This software would be in a unique position to capture the revenue of this untapped market, and provide a unique solution to a currently unsolved problem.

1.3 Our Competitive Advantage

While there are currently 4 major businesses throughout the world that offer instructional information for climbers, only 2 of them provide resources for locating climbing routes and gyms, with only 1 being based in the United States. Not only that, none of them focus on climbing improvement.

CliffBuddy's marketing strategy is to emphasize the importance of collaboration to encourage the climbing community ("The adventure won't be lonely") and the ease of access. Indoor climbers, for instance, will open the application anytime with notifications of various events around the area and feedback from other users for their recent climb.

All feedback from other users will be considered "comments" and "suggestions" and not actual advice. The advice that will be suggested throughout the climb will be sourced information from the application itself. The advice and information provided from the web application will be based on AMGA certified Climbing Instructors and professionals within the field. This will coincide with information written by online sources that are ensured to be viable.

1.4 Financial Projections

Based on the size of the market and defined market area, the sales projections for the first year would be $10,800 given we have at least 150 active users for the first year. We project a 5-10% growth rate per year for the first three years.

The salary of the group members will be $0. To begin with, all resources used for the project will be free. Given that the system architecture becomes scalable, the system will project to have a monthly resource use of $25 a month.

Given the competitors are usually paying a large amount of money for employee salary, software licenses, server licenses, and training time, Navite.io's rates are aboundingly cheaper. There is no risk for investors trusting us to develop the product that has been envisioned.

The use of social media, online promotions, and word of mouth provides us a chance to aggressively build our user base before and after deployment of CliffBuddy.

II. Project Scope

2.1 Introduction

CliffBuddy will contain a design which contains a Front End user interface accessible by Google Chrome (within 2 months of stable release), along with a Back End containing various layers for data to travel. The project will be considered complete when given certain criteria established below.

2.2 Project Deliverables

All aspects of the project will be stored in the organization's public repository on GitHub.com. This will include all project documents such as but not limited to:

- A high/low level design document
- A technical specification document
- Business requirement document
- README Markdown file for source code use
- Site Map Diagram

The project will also contain the source code for the Front End, which includes the User Interface.

The Back End source code will contain:

- An entry point
- A service layer
- A data transfer object (DTO)
- Persistence layer

2.3 Project Acceptance Criteria

The project will be considered complete when:

- All the features are functional with given requirements with no system-breaking bugs
- The system infrastructure is successful with pertaining and responding with data
- User experience rating is considered "satisfactory" after beta tests

2.4 Project Exclusions

Anything that is considered an “extension point” or an “additional feature” will not be within the scope of the project, as those features are to be developed beyond project development. In addition, the project scope does not include development for mobile applications. It will be restricted to web development only, but mobile development could be a project to consider down the line in the future. Furthermore, this project will be developed using the Chrome browser, so if users visit the website using other browsers, such as Microsoft Edge or FireFox, then they might encounter uncertain errors. In our project, we will not be working to fine tune our product in the other browsers mentioned. Since our project scope is based in the US only, other foreign users may have troubles or difficulties using the site or accessing it in general.

2.5 Project Constraints

There are some constraints that Navite.io might encounter during the development of the project. One of the constraints is the cost constraint. The limitations with given technology is due to the given technology stack being free of charge. Since we are using technology that is free of charge, that could limit the potential, in which we could optimize our project. For example, using AWS might offer better performance, which would improve overall user experience, but that obviously comes at a cost. Ultimately, optimization limitations are the tradeoff in using free technology tools.

Another constraint is the time. The time it might take for developers to learn and train their skills in new technology might be steep. As the developers are also students for various classes and part-time workers, the limited time to develop the skills necessary for the project is a constraint for the project. Furthermore, the developers being students, they have to allocate time to successfully perform in different classes, which

2.5 Project Constraints

could possibly take away from the time that is spent on project development. In addition, given our project proposal, 15 weeks might not be enough to fulfill everything within project scope. It's simply too early to tell at this stage, so in this aspect, time is yet again a constraint on our development process.

2.6 Avoiding Scope Creep

To combat undefined or unanticipated changes that occur after project development begins, Navite.io planned ways for developers to manage this issue.

- Identify the changes that are made to the requirements of the project
- Understand how this change will affect the system as a whole
- Gain unanimous approval with developers and stakeholders before proceeding change in requirements
- Implement the changes in a timely manner to reduce delays of project lifetime

All meetings will be held 3 times a week via scrum. This will make sure that developers are meeting stakeholder requirements and identify technical blocks.

III. Business Goals and Objectives

3.1 Mission Statement

Navite.io is a group that will build a product that encourages collaboration in a streamlined and easy-to-use manner. We will build a reputation among the United States as the leading, most reliable, and knowledgeable climbing resource for users.

3.2 Goals

Goal 1: Encourage Collaboration

The objectives to achieve said goal would be as follows:

- Develop a social space for users to share their thoughts on a puzzle solution
- Encourage users to post and comment often via scored values
- Allow users to define specific words and associate with other subterms

Goal 2: Increase Brand Awareness

The objectives to achieve said goal would be as follows:

- Create a social media manager role to make posts for advertising the product
- Create a user referral bonus program (after month 3 of beta release)
- Recruit various climbers to endorse the use of the product

Goal 3: Establish Expertise

The objectives to achieve said goal would be as follows:

- Cite all sources of information given within the list of terms in CliffBuddy
- Have CliffBuddy's user interaction be based on "suggestions" rather than actual advice, which will be displayed by the application
- Create credibility with contacts by showing that Navite.io has the ability to help people with their climbing skills

3.2 Goals (cont.)

Goal 4: Scale Project for Future Developments

The objectives to achieve said goal would be as follows:

- Plan with Tech Stack Lead for scaling system architecture
- Plan with CFO for financial decision analysis with scaling system
- Create modular code that will scale with hardware upgrades

Goal 5: Improve Customer Satisfaction

The objectives to achieve said goal would be as follows:

- Listen to feedback of beta testers, and understand the goals and struggles of users
- Manage expectations within marketing

Goal 6: Monetize Crucial Features

The objectives to achieve said goal would be as follows:

- Create a payment program for CliffBuddy, with Beta users given a discount before official deployment
- Develop features that will be exclusive to paid users, but will not take away experience for users who are receiving input from the main feature

Goal 7: Find New Markets for Future Projects

The objectives to achieve said goal would be as follows:

- Contact people within the climbing industry for any new developments
- Ask for customer thoughts of the product, which might lead to a potential market

3.2 Goals (cont.)

Goal 8: Increase Management Communication

The objectives to achieve said goal would be as follows:

- Train for soft skills in employees, such as conflict resolution and time management
- Develop a business workflow for successful code delivery
- Take sprint retrospectives seriously and develop skills based on retrospects

3.3 How Project Ties with Business Goals

CliffBuddy as a product will also be considered the public debut for Navite.io. This will mean that the success of the company as a whole will rest on the success of the project. The project, and in tandem the product, reflects the mission statement that Navite.io represents. By developing this project, the company's bottom line will lead to success in the long run based on the development's life cycle. The goals that the company would like to achieve would only be possible if a working product is tailored for the market.

Various objectives will be completed for Navite.io's success during the software development life cycle. This will mean that the developers would have to work on building the business before deployment. All objectives that are to be completed before deployment will be addressed within the business timeline.

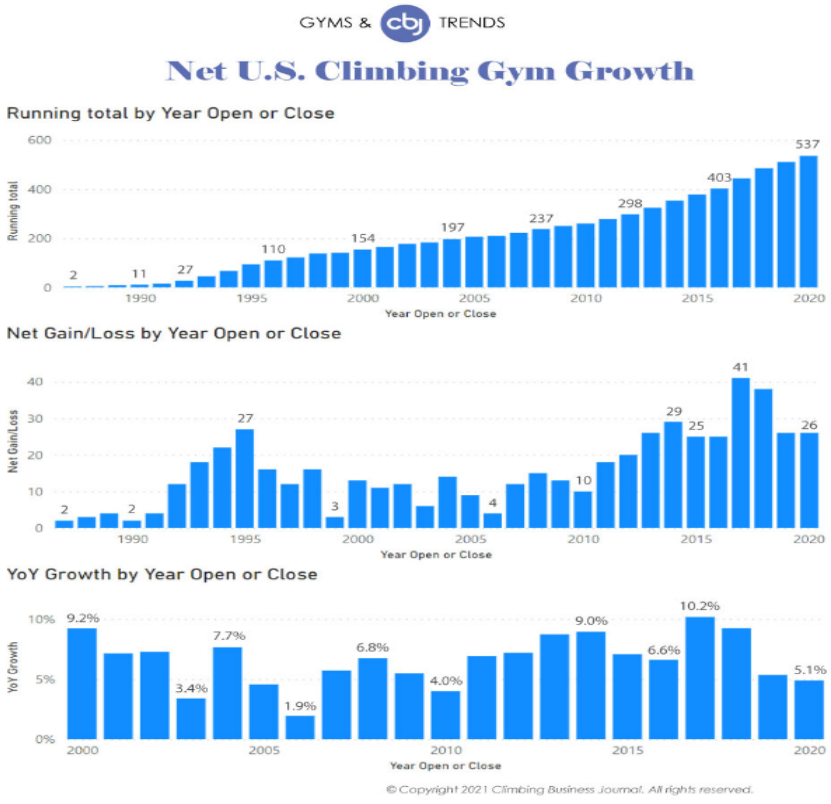
IV. Project Value

4.1 The Rising Popularity of Climbing

The use of the product will coincide with the ever-rising popularity of the sport. The inclusion of the sport within the 2020 Olympics boosted the popularity to a greater degree. Navite.io has decided to analyze the data that is available to track --and possibly predict-- the growth of the sport.

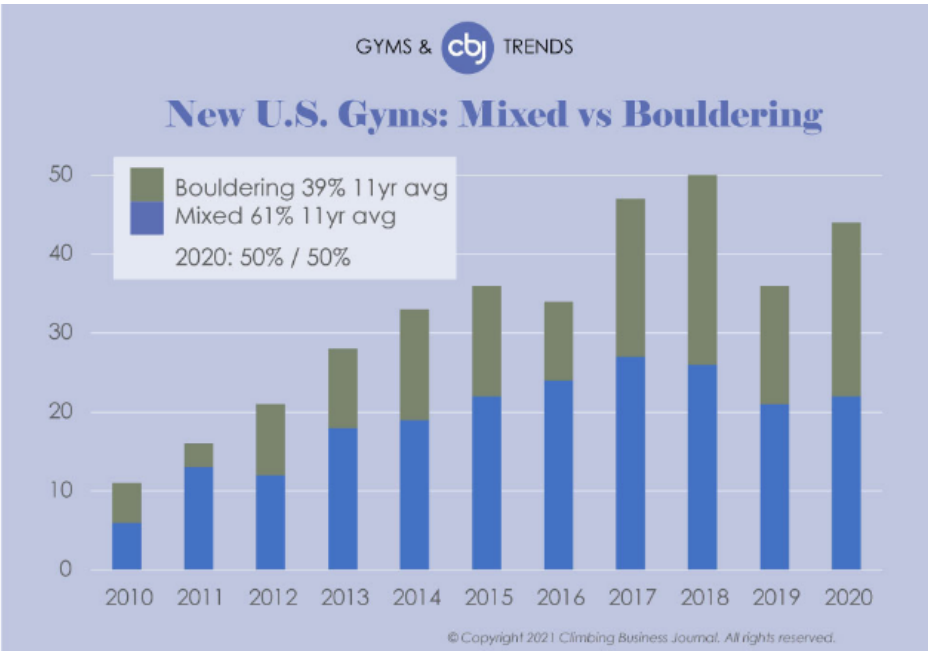
By the nature of outdoor climbing, it was quite difficult to find active outdoor climber statistics. However, statistics for indoor climbing were quite easier to aggregate.

The following data was conducted by Climbing Business Journal, which studied the net United States Indoor Climbing Gym Growth.



4.2 Addressing the COVID-19 Pandemic

The major viable concern that stakeholders might prose would be the drastic change the climbing industry faced during the COVID-19 pandemic. This is obviously present with the study on the Net Gain and Loss by Year Open of Close. While the major effects are still challenging the climbing industry 1 year later, the steady opening of indoor gyms have allowed the rise of climbing to continue once more. This is shown with the overall increase of new climbing gyms, as 44 gyms were open in the U.S. in 2020.



The trend of bouldering-focused gyms in North America was observable at the start of the 2010s and continued into 2020. This will provide a potential target audience for CliffBuddy, as there will be a section for climbers that are interested in taking on the subsection of bouldering. Of the 61 new climbing gyms that are planned to open in the U.S. in 2021, 70 percent are mixed facilities (43 gyms) of both mixed climbing and bouldering.

4.3 CliffBuddy's Value Within the Market Today

Even with the challenges of 2020 and the low U.S. growth rate of the indoor rock gym, the outlook for Navite.io's target audience remains positive.

As stated previously, the U.S. industry has only 3.5 percent of indoor gyms permanently close after 2020. Not only that, the vast number of new climbing gyms are planning to open in 2021 or later. Gym developers and brands remain to have an optimistic view despite the year of economic failure.

This is valuable for Navite.io's introduction of CliffBuddy. The use of the application through new local climbing gyms will pique the interest of new climbers and veterans who want to learn or polish their skills. The absence of climbing for veterans might leave them wanting to track their performance since the last time they've climbed.

Most importantly, there is a severe lack of tooling that helps climbers document their climbs accurately for analysis. Videos and pictures are not the right tool for the job as they omit information that our website will be sure to include. Having a centralized area for the growth is essential in high skill ceiling sports such as rock climbing.

V. Project Requirements

The following is an in-depth analysis of all the project requirements that CliffBuddy developers will adhere to. All features will be listed and contain user stories which have the following:

- Functional Requirements
- Preconditions
- Any Required User Input
- Successful Postconditions
- Nonfunctional Requirements

Features

- **5.1 Climbing Problem and Solution Mapping (Main Feature)**
    - **5.1.1 Create Problem Foundation**
        - (A problem foundation is the basic components needed to create a problem)
        - Functional Requirement
            - Users enter a valid name for the map that is within thirty characters
            - Users upload an image of a climbing wall that is not greater than 10mb in size
            - Users must specify whether the problem was completed or not completed

- **5.1.1 Create Problem Foundation (cont.)**
  - Functional Requirements (cont.)
    - Users select a valid word from the “climbing type” category
      - The climbing types that are available by default:
        - Bouldering
        - Mountaineering
        - Top Rope
        - Free Solo
    - Users must specify the difficulty and enjoyment on a scale of 0 to 5
    - Users can cancel the process of creating a problem foundation
    - Users can finish the process and create the problem foundation
  - Preconditions
    - User is logged in
    - User is within the limit of 10 maps per account
    - User is on the “Your Solutions” or proper page
  - Successful Postconditions
    - The user will transition to page to select whether they will edit the problem or edit the solution

- **5.1.1 Create Problem Foundation (cont.)**
  - Successful Postconditions (cont.)
    - The created map will be visible from the “Your Solutions” page as an image with the name, difficulty, and enjoyment visible under the image if the user did not cancel operation.
  - Unsuccessful Postconditions
    - If a user fails to include the name, image, climbing type, or completion the user is notified of the missing fields with an error message on the same page
    - If a user fails to follow the constraints for the fields, the user is given information on the constraints
  - Nonfunctional Requirements
    - The user will be able to view the problem after at most 3 seconds after it is created

- **5.1.2 Selection of Solution or Problem Editing When Viewing Map**
  - Functional Requirements
    - The user can select to edit the problem or the solution
  - Preconditions
    - A user is currently viewing a valid problem that they have created
  - Successful Postconditions
    - The user transitions to the page where they can edit the problem or the solution
  - Unsuccessful Postconditions
    - If the user has not added any nodes to their problem, then they will be given an error when they choose to edit the solution
  - Nonfunctional Requirements
    - The transition to the next page will take at most 1 second to occur

- **5.1.3 Add Nodes to Problem**
  - Functional Requirements
    - Users can add nodes to the problem to identify holds
      - Nodes are circles that are placed on the image
    - The node can be placed within the boundaries of the uploaded image
  - Preconditions
    - A user is editing a problem that has a foundation created.
    - The user is within the 100 node placement limit
  - Successful Postconditions
    - A node is added onto the image
  - Nonfunctional Requirements
    - The node should be placed onto the image no later than 1 second after the user adds a node

- **5.1.4 Add Information to Nodes When Editing Problem**
  - Functional Requirements
    - Once a node is placed, selecting the node once more allows the user to add optional information to the node
      - The user can specify the type of hold by searching the glossary for the hold type
      - The user can save or cancel the addition
  - Preconditions
    - A node exists on the image
  - Successful Postconditions
    - The user is back on the page viewing the entire problem
  - Nonfunctional Requirements
    - The user will return to the problem view page after at most 1 second after adding the information

- **5.1.5 Delete Nodes from Problem When Editing Problem**
  - Functional Requirements
    - The user can delete a node
  - Preconditions
    - A node exists on the image
  - Successful Postconditions
    - The user is back on the page viewing the entire problem but the node that was selected for deletion is removed from the display
  - Nonfunctional Requirements
    - The deletion of the node will take at most 1 second to occur

- **5.1.6 Select Nodes for a Solution When Editing Solution**
  - Functional Requirements
    - The user can select a node as a part of their solution for the current step
    - The user can include optional information for the node
      - Users can specify whether the node was difficult or not
      - Users can specify what appendage they used by selecting RH, LF, RF, LF
    - Once a node has been selected, they can save or cancel the selection
  - Preconditions
    - A node exists on the image that is not already part of the current step of the solution
  - Successful Postconditions
    - The node color changes to red if the user has chosen the node as part of the solution
    - If the user cancels then no change occurs and the user is returned to the editing solution page
  - Nonfunctional Requirements
    - The color change of the node will take at most 1 second to occur

- **5.1.7 Change Between Steps When Editing Solution**
  - Functional Requirements
    - The user can switch to the next step or switch to a different step in the solution
    - The nodes that were marked as part of the solution for that step will be saved
  - Preconditions
    - The user is editing a solution
  - Successful Postconditions
    - The user transitions to another step in the solution and the nodes that were marked as a solution from that step are saved.
  - Nonfunctional Requirements
    - The switch will take at most 1 second to occur

- **5.1.8 Finalize the Solution**
  - Functional Requirements
    - The user can select to finish the solution
  - Preconditions
    - The user is editing a solution
  - Successful Postconditions
    - The user transitions to the page where they can choose to edit their solution or problem
  - Nonfunctional Requirements
    - The switch will take at most 1 second to occur

- ## 5.2 Glossary of User Defined Terms
    - ### 5.2.1 Create Word
        - Functional Requirements
            - The user must name the word and must not exceed 30 characters
            - The user must define the word and must not exceed 600 characters for the definition
            - The user can associate the word with an optional image with a size limit of 5 MB
            - A word must be selected to be part of at least one category.
                - Categories are not user-defined but we can add more should the need arise.
                - Included Categories
                    - Default
                        - includes default words
                    - Holds/Rocks
                    - Equipment
                    - Grips
                    - Climbing Type
                    - Other
            - The user can save or cancel
        - Preconditions
            - The user has not exceeded the 50 word limit per account

- **5.2.1 Create Word (Cont.)**
  - Successful Postconditions
    - The user transitions to the page where they can view the word and its contents
    - The word will appear in the glossary as the name of the word along with the username of the user who created it
  - Nonfunctional Requirements
    - The transition to the next page will take at most 1 second to occur

- **5.2.2 Create Subword**
  - Functional Requirements
    - The user must name the subword and must not exceed 30 characters
    - The user must define the subword and must not exceed 600 characters for the definition
    - The user can save or cancel
  - Preconditions
    - The user has not exceeded the 15 subword limit
  - Successful Postconditions
    - The subword appears next to the definition of the word along with its own definition
  - Nonfunctional Requirements
    - After creating the subword, the subword should appear after at most 1 second after creating

- **5.2.3 Delete Subword**
  - Functional Requirements
    - The user can delete a subword
  - Preconditions
    - A subword exists
    - The user is on the page where the contents of a word is shown
  - Successful Postconditions
    - The subword is removed from view
  - Nonfunctional Requirements
    - The deletion of the subword should occur after at most 1 second

- ### 5.2.4 Delete Word
  - Functional Requirements
    - The user can delete a word by selecting the words that they have created
  - Preconditions
    - A word the user created exists
  - Successful Postconditions
    - The word is removed from view
  - Nonfunctional Requirements
    - The deletion of the word should occur after at most 1 second

- ## 5.3 Word Search

5.3.1 Node Association Dropdown Menu

- Functional Requirements
  - Users can search for user-defined words by inputting the characters of the word or using a filter
- Preconditions
  - User must be logged into CliffBuddy
  - User should either be on a climbing problem or glossary page
- Required User Input
  - User selects a node and is redirected to a dropdown menu
  - User searches for specific words associated with node
- Successful Postconditions
  - Node is associated with a specific word others can view
  - Example: Types of holds, what equipment, what exercise is useful for that technique
- Nonfunctional Requirements
  - Dropdown menu should show all user suggested words in a manner of seconds
  - 5 most relevant words will be displayed immediately in dropdown menu

- **5.4 Performance Visualization**
    - **5.4.1 Map Summary**
        - Functional Requirements
            - Users can click on a button to view the Summary report of a map.
            - The report shows the number of different rocks and its level of difficulty.
            - The report shows the number of different grips and its level of difficulty.
        - Preconditions
            - There is a stable wifi connection to allow server communication.
            - Users must be logged into CliffBuddy
            - Only valid maps can be visualized.
            - Valid maps are maps with at least one node with a defined hold, and at least one sol
        - Required User Input
            - Users can click on a button to view the report.
        - Successful Postconditions
            - The feature will show the metrics to help summarize what components are included on a map. These metrics include:
                - Number of steps needed to complete each problem
                - Number of different rock types

- ### 5.4.1 Map Summary (cont.)
  - Successful Postconditions (cont.)
    - Number of rock types that users struggled with and a list of them along with their difficulty
    - Number of different grips that users struggled with and a list of them along with their difficulty
  - Nonfunctional Requirements
    - This feature will group the maps together and will become the source for the gathered datasets.
    - For scalability, visualization should handle other glossary terms that could be added on.
    - The time it takes to gather all necessary data shouldn't take too long, possibly at most 5 seconds.

- ### 5.4.2 Graph Visualization
  - Functional Requirements
    - User can click on a button to view Graph representation of their improvement
    - The graph will measure user improvement measuring difficulty vs time
    - One dependent variable can be chosen to display at a time, this variable being a word from the glossary
    - Independent variable denotes the maps that were selected
  - Preconditions
    - Requires that multiple maps of different dates are selected
    - Maps should be valid with at least one node with a defined hold
  - Required User Input
    - User clicks on the button to view graph
  - Successful Postconditions
    - The feature will present to the User with the graph that shows their progress or lack of progress.
  - Nonfunctional Requirements
    - The graph should be completed and displayed to the user in less than 4 seconds after the time it's queried (the user presses the button to view the graph).
    - At least 2 maps should be selected to show progression

- ## 5.5 Problem Search
    - ### 5.5.1 Search for Problems
        - Functional Requirements:
            - Users can search for specific Problems that are posted on CliffBuddy
            - Users can search Problems on a dedicated search bar that is dedicated for a single category.
            - One single “search” button will be used to accept different search criterias at once.
            - All searches made will be returned in alphabetical order based on the Map name.
        - Preconditions:
            - There is a stable wifi connection to allow server communication.
            - The user must be logged into CliffBuddy.
            - The user must be on the page for Problem Search.
        - Required User Input:
            - The user can use all, none, or a combination of these criterias to narrow their search.

- ### 5.5.1 Search for Problems (cont.)
  - Required User Input (cont.)
    - The parameters are: Uplaoder username, Map name, and Location (address, city, state)
      - Precondition to Location: the uploader has the option to input location, so in order to search with location, the uploader must have included the location at the point of creation.
  - Successful Postconditions:
    - The search returns all of the Problems that match the user-desired combination of inputs.
    - If there are no Problems that match the input criteria, then shows a message that says “No matches were found.”.
    - If the User does not put any search criterias, then it will return all datasets.
  - Nonfunctional Requirements:
    - The search should return datasets ideally in less than 3-4 seconds.
    - Considering search inputs and existing datasets are the same, the search should return the same datasets for reliability.

- ### 5.5.2 Filter Searches
  - Functional Requirements
    - Users can filter their search, and when the search is queried, it will return the set of Problem(s) that satisfy the search filter.
    - The filter bar will be vertical on the left side of the web page with tabs dividing the different filter criterias.
    - Hitting the “apply filter” button will implement those filters and return the correct datasets.
  - Preconditions
    - There is a stable wifi connection to allow server communication.
    - The user must be logged into CliffBuddy.
    - The user must be on the page for Problem Search
  - Required User Input
    - Users can search using a single or combination of multiple parameters. These parameters are:
      - Difficulty (the user will be able to decide an exact value or a range)
      - Enjoyment (the user will be able to decide an exact value or a range)
      - Glossary Terms

- ### 5.5.2 Filter Searches (cont.)
  - Successful Postconditions
    - The filter returns all of the Problems that match the user-desired combination of inputs.
    - If there are no Problems that match the input criteria, then shows a message that says “No matches were found.”.
    - If the user doesn’t specify any filters, it will return datasets based on the searches.
  - Nonfunctional Requirements
    - The filter should return the results ideally, in less than 3 seconds.
    - Considering, filter inputs and existing datasets are the same, the search should return the same sets of Problems for reliability.
    - For scalability, this feature should be able to implement other filter criterias on top of the existing one.

- **5.6 Climbing Scheduler**
    - **5.6.1 Name Schedule**
        - Functional Requirements
            - User can start the process of creating a schedule
            - The user must name the schedule with at most 30 characters
            - The user can save or cancel the process
        - Preconditions
            - The user has not exceeded the 10 schedule limit
            - The user is logged in
        - Required User Input
            - User input for name
            - Selection of save or cancel
        - Successful Postconditions
            - The user starts the process of creating a schedule and transitions to a page where they can select a map and date to add to the schedule if the user saved the schedule name
        - Nonfunctional Requirements
            - The transition between pages takes at most 3 seconds

- **5.6.2 Select Map**
  - Functional Requirements
    - Users can pick a public map or private map and add it to their schedule
    - The user must select a start date and end date for each schedule
    - The user can save or cancel the selection
  - Preconditions
    - The user has not exceeded the 50 map limit per schedule
  - Required User Input
    - User searches and selects for map
    - User select start and end date
  - Successful Postconditions
    - The user can see a map on their schedule with the image of the map, name, start date, and end date visible.
    - Start dates should have the status of “current”
    - End dates should have the status of “past” or “completed”
  - Nonfunctional Requirements
    - The addition of the map takes at most 1 second

- ### 5.6.3 Delete Schedule
  - Functional Requirements
    - User can select a schedule within their established list and delete it when necessary
  - Preconditions
    - The user has not exceeded the 10 schedules limit
    - The user must be logged in
    - User must be within the schedule site
    - User must have existing schedule created
  - Required User Input
    - User selects schedule via checkmark
    - Selects an option deletion of schedule
    - User will be prompted if they want to delete schedule
    - User deletes schedule
  - Successful Postconditions
    - Schedule that was once displayed on the site will be deleted
    - All dates that were associated with the schedule will now be considered empty
    - Schedule dates associated with a map will be deleted
  - Nonfunctional Requirements
    - Deletion process should take 2 seconds maximum
    - Prompt for deletion should have a floating selection box with user not allowed to access other content

- **5.6.4 Share Schedule**
  - Functional Requirements
    - Users can share a schedule by posting their schedule on their profile for other users to see
    - Users can click on share schedule button, which will prompt them to select from the list of schedules they have
  - Preconditions
    - The user is logged in to their account
    - The user has a created schedule for sharing
  - Required User Input
    - User must go to the options to edit their profile and add to share schedule
    - Once they choose to share schedule, they must choose schedules they want to share on their profile
  - Successful Postconditions
    - The schedule is shared on the user profile for other users to see
  - Nonfunctional Requirements
    - The profile should successfully update in less than 3 seconds.

- ### 5.6.5 Remove Shared Schedule
  - Functional Requirements
    - Users can remove a shared schedule from their profile
    - Users can click on their shared schedule to see the list of shared schedules.
    - Clicking on “unshare” will prompt the user to choose the schedules they want to unshare(can choose multiple at once).
    - Clicking on “apply” will ensure the changes are made.
  - Preconditions
    - User is logged in to their account.
    - Users have at least one shared schedule on their profile.
  - Required User Input
    - The choice of schedules they would like to unshare
    - Also requires unshare confirmation by clicking “apply”
  - Successful Postconditions
    - The desired schedules are unshared and removed from their profile.
    - The profile correctly reflects those changes
  - Nonfunctional Requirements
    - The changes should be made in under 3 seconds.

- **5.7 SMS Reminders**
    - **5.7.1 Safety SMS Reminders**
        - Functional Requirements
            - User will receive safety notifications that are sent to their phone after creating reminder on CliffBuddy
        - Preconditions
            - User must be logged in
            - User must have a registered phone number on account
            - User should be in “reminders” site
        - Required User Input
            - Select “Safety” reminder
            - User will create custom safety message sent to their phone
            - Message cannot exceed 600 characters
            - Specify date and time for message to be sent (must be in future)
            - Maximum number of messages is 10
        - Successful Postconditions
            - Notification sends at the correct time
            - Displays how long until notification sends
            - Displays a map to climbing location
            - Nonfunctional Requirements
            - Should be modular with climbing scheduler
            - User notifications must be sent at the immediate time scheduled

- **5.7.2 Equipment SMS Reminders**
  - Functional Requirements
    - Users can receive reminders on their phones for specific equipment they need to bring before a climb
    - Users can take pictures of selected equipment and have picture sent to user as a reminder
  - Preconditions
    - User must be logged in
    - User must have a registered phone number on account
    - User should be in “reminders” site
  - Required User Input
    - Select “Equipment” Reminder
    - User will create custom equipment reminder message sent to their phone
    - Message cannot exceed 600 characters
    - Specify date and time for message to be sent (must be in future)
    - Maximum number of messages is 10
  - Successful Postconditions
    - User can be able to share their equipment as a post
    - Notification sends at the correct time
    - Displays how long until notification sends
    - Displays a map to climbing location

- **5.7.2 Equipment SMS Reminders (cont.)**
  - Nonfunctional Requirements
    - Should be modular with climbing scheduler
    - User notifications must be sent at the immediate time scheduled
- **5.7.3 Extension Point**
  - SMS reminders for exercise recommendations
  - Links to video tutorials that are recommended by the community that contains good safety practices

Future Features

This feature will not be within the scope of our project, but contains user stories for future development purposes.

- **5.8 Event Sharing**
    - Create an event
        - Specify the name, time period, and location
        - Name must not exceed 30 characters
        - Time period must be in the future
        - Location requires address, city, state
        - Must specify equipments using words from the glossary and money/fees (if required)
        - Can include a max of 5 optional maps to the event
    - Remove created event
        - Once the date of the event is passed, the event will only be visible to the user who created it
        - The creator can ultimately remove their event whenever desired
        - Requires user to click on “remove this event”, and will prompt confirmation message
    - Share an event
        - View created event and promote on profile
        - Add event to schedule

VI. Key Stakeholders

These individuals will be driving Navite.io forward as the project development life cycle takes place. The roles that each team member has represents their leadership within a section of the project.

- Project Client: Vatanak Vong
- Project Manager: Vatanak Vong
- Project Leader: Hunter Lewis
    - Performs and leads the scrum meetings and delegates tasks for sprints
    - Making sure project development time is following set schedule
    - Manages high level and low level designs, aligning it with all project requirements
    - Assists front end and back end team from a full stack perspective
- Front End Lead: Jay Park
    - Delegates tasks for ensuring a high quality user experience
    - Manages site map and trains team members on front end technologies
- Back End Lead: Jerry Belmonte
    - Delegates tasks regarding the transport of data through the back end architecture
    - Manages 3rd party API's and all hardware associated on the service layer
- Tech Stack Lead: Ji Park
    - Researches and implements various technologies necessary for project development
    - Manages Technical Specifications and updates regularly, making sure all requirements can be brought to technology's optimal potential

Key Stakeholders (cont.)

- Director of Marketing: Hunter Lewis
    - Develops and implements marketing plans, including promotional programs
    - Manage marketing budget and outreach to contacts to progress brand
- CFO: Jerry Belmonte
    - Track flow of finances
    - Analyzes and manages resources placed into development cycle
    - Manages financial actions of a company and suggest financially optimal solutions to project scaling

VII. Product Limitations

A number of limitations are brought to CliffBuddy due to being a web application. The security of the application can be easily compromised if the development team neglects applying regular security patches. By using methods such as SSL enforcement, the problem of data breaches will be mitigated. The internet dependence of the application can be a major drawback for the product, as it's considered compulsory. All data is transferred to the server via HTTP, which means without a reliable internet connection, one cannot access CliffBuddy offline. This is quite an issue if a user were to use the app in an outdoor site where internet connection can be nonexistent at times. We will attempt to solve this later by developing a mobile application for CliffBuddy, to allow for offline usage.

The geographical location of our target audience will be focused within the United States, which means most of the support users will receive will be given within indoor and outdoor locations within the United States. While there is a possibility for expansion to become worldwide, Navite.io would like to start within the country of origin before extending their roots to other potential markets.

IX. Resources

https://www.99boulders.com/the-growth-of-climbing

https://www.climbingbusinessjournal.com/climbing-gyms-and-trends-2020/