Slopes Product Vision Statement

Ben Miller, Annalies Turley, Sean MacDonald, David Ma, Shavon Zhang, Jan Evangelista

Motivation/Opportunity:
To create an app which enables users to ski and snowboard with as much ease as possible.

Problem Statement:
We would like to solve to the problem of having to check multiple places for ski info. This app will be useful because people often make the decision on which ski hill to go to based on various conditions. These conditions are irritating to access since each small piece has its own website. A streamlined app with all useful information would be extremely helpful. A well designed app on multiple platforms would make skiers lives just a little bit easier.

The problem of	Having to check multiple sources for ski hill information.
affects	Any people planning on traveling to a ski hill near Vancouver
The impact of which	Makes it inconvenient for mountain-goers to view and process all available information and make their decision on where to go
A successful solution would be	A streamlined Android/iOS app that fetches all necessary ski hill information and displays it in an efficient, compact manner

Product Position Statement:
This app's target audience is mostly for passionate skier and snowboarders. Another large audience we would cater towards would be new skiers, who don't know what ski hill they want to go to. This project would be entirely software based, because we plan on making iOS and Android apps. In short, our app will pull data from multiple API's and put it all in one streamlined place. Data includes new snow, total snow, road conditions and weather. On top of good information, this app will also make recommendations based on the user's skiing preferences.

For	People planning on traveling to a ski hill near Vancouver
Who	Planning a trip to a ski hill near Vancouver
Our system	An Android/iOS app
That	Fetches ski hill data from myweather2 API and Google Maps API to present the user with necessary mountain data and present it to the user.
Unlike	Other snow report Apps
Because	Of the informed recommendation on what's best

Users:
Our app is for anyone who wants to have all the ski info and data they need all in one place. Anyone who is interested in ski and board will enjoy having this app.

Feature List:
Features for this app include easy access to all ski data, as well as a personalized recommendation system for deciding which hill to go to. Ski data would include:
-New snow
-Weather
-Driving distance

The user will have the ability to set one mountain as their favorite mountain, whose weather and slope information will be displayed on the home page. The user will also have the ability to change which mountain is set as their favourite. This makes the favourite mountain more easily accessible for the user, as he or she will no longer need to search for the mountain.

Depending on the skiers preferences we will implement an algorithm which would recommend the best possible ski hill. Skier's preference is determined through their input on desired size and difficulty of ski hills with the use of sliders. Size will be calculated based on the number of runs at the mountain, and communicated to the user

as such. The inputted size will be the maximum size of the mountains returned in results. Our mountains are ranked in difficulty on a scale of 1-5. When a user inputs a desired difficulty, we will return mountains with the inputted difficulty as well as those with a difficulty at one level lower and one level higher.

In the details page of a mountain, there will also be a button that will direct the user to their phone's native maps app, providing them with directions from their current location to the selected mountain.

Constraints:
Our app will be dependent on open source libraries to help us parse information on snow conditions from different websites of the different ski mountains.

Scopes and Limitations:
Our app will work with a variety of ski hills in BC, so we currently don't plan on extending the coverage to any ski hill in the world. Time permitting, we will implement coverage for as many ski hills around Canada as possible. This can be done by automating the task of filling the mountains database by using the OnTheSnow Web Services API to provide detailed resort information which can then be inserted into the database.

Assumptions and Dependencies:
We are assuming all snow conditions and data we are gathering will be accessible through API's such as myweather2. If these prove to not work out, our job will be much more difficult.

Additionally, we will be relying on the user device's internet connection. The database will only update when the device is connected, without a connection the last fetched data is displayed. Without this connection the app will not run properly, and should display an appropriate message.

Use Cases

1. **Display information**
   Identification: Displays relevant information for user
   1.1 User requests to look at data
   1.2 If user is connected to internet, data is loaded from API. If user is not connected to internet, cached data from previous API loads are loaded onto page.

Primary Actor: User
Stakeholders: User, API
Preconditions: Info must be up to date, user is actively seeking out info
Postconditions: Must be correct
Main Success scenario: User is successfully able to view data
Extensions and Alternative Flows
Open Issues

2. **Change information**
   Identification: Allows user to change/input their information
   2.1 User requests to input information
   2.2 User inputs various fields of information using sliders
   2.3 App will save this information for later calculations
   Primary Actor: User
   Stakeholders: User
   Preconditions: User must request input
   Postconditions: Information is saved
   Main Success scenario: All fields successfully contain reasonable data
   Extensions and Alternative Flows
   Open Issues

3. **Update information**
   Identification: Updates based on if user is requesting more information
   3.1 Check if user has just opened the App, or if the user has requested an update of information.
   3.2 If so, pull data from API and update the current information
   Primary Actor: API
   Stakeholders: User
   Preconditions: Must have internet connection and an operating App
   Postconditions: Data displayed
   Main Success scenario: If all works as planned, correct data will be stored in device
   Extensions and Alternative Flows:
   If case occurs where the device has no internet connection, load cached data and display error message.
   Open Issues

4. **Choose Favourite Mountain**

Identification: Allow the user to choose a favourite mountain, this is the mountain that is loaded on the homepage when the app is started.

4.1 User presses the favourite button on the mountain detail screen

4.2 If there was no previous favourite mountain, the app saves this mountain as the favourite.

Primary Actor: User

Stakeholders: User

Preconditions: User is in the mountain detail page for the mountain in question

Postconditions: A mountain is saved a favourite.

Main Success Scenario: Selected mountain is saved by the app

Extension and Alternative Flows:

4.2a If there was a previously selected favourite mountain, the app communicates to the user via pop-up that only one mountain may be chosen, and prompts them to choose to overwrite. If the user selects yes, the new favourite is saved, if no is selected the popup closes and no action is taken.

Open Issues

5. **Provide Directions to Mountain**

Identification: Allow the user to see directions to the mountain by opening the Maps app

5.1 User presses the Get Directions button on the mountain detail screen

5.2 The app launches the phone's Map application with directions to the mountain in question.

Primary Actor: User

Stakeholders: User

Preconditions: User is in the mountain detail page for the mountain in question

Postconditions: The Maps app with the route to the mountain is opened

Main Success Scenario: The user can see the route to the mountain in the Maps app

Extension and Alternative Flows:

5.2a If the app cannot find the phone's Map app, an error message is displayed.

Open Issues

Use Case Diagram

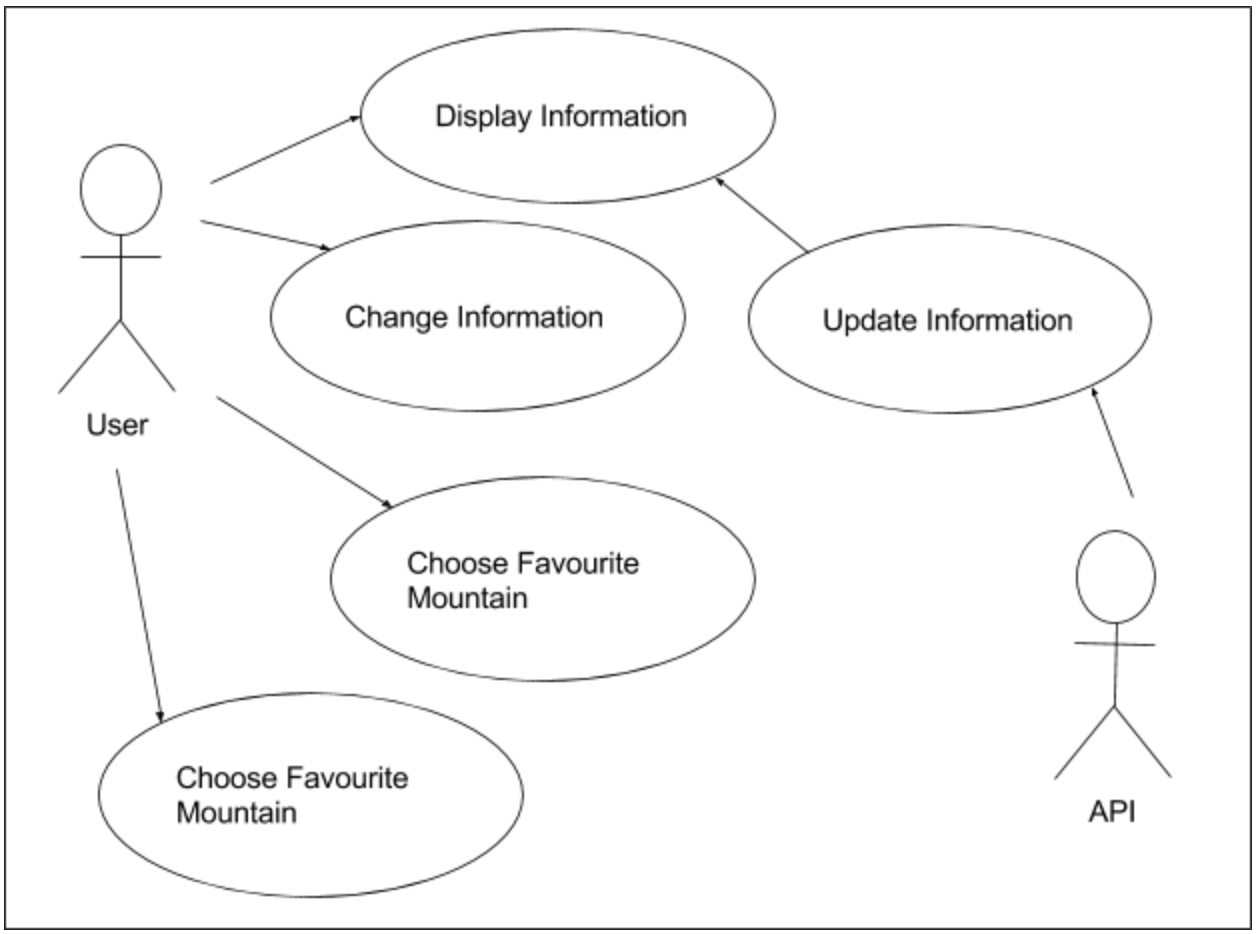

Figure 1: Use Case Diagram for the Slopes system

Non-Functional Requirements

Performance Requirements

The main performance requirement for the app is that it should update reasonably quickly, such that using the app is significantly more efficient than manually checking the websites. Additionally, the information displayed on the app should be as accurate as permitted by the APIs. If the APIs are unresponsive, or the device cannot connect to them, the most recent data should be displayed, along with a warning message to communicate the the data is not completely up to date.

Safety Requirements

The main safety concern of our app is to display the correct snow and weather conditions. If a mountain is currently experiencing extreme and dangerous weather conditions (such as strong winds), our app should provide clear warnings.

Security Requirements

Our main security concern is making sure that the location data we retrieve from our users is not accessible by anybody else and is not used for anything other than providing mountain information. Additionally, we ask permission before making use of a user's GPS data.

Software Quality Attributes

The success of this app will depend on the correctness and the reliability of the information that we provide to our customers. Our goal is to have skiers and snowboarders use our app over the websites of the ski mountains, to retrieve snow conditions. For this to be the case, our app must provide the equivalently correct information in a more time-efficient and convenient manner than having to access each website individually. It must also have a pleasing and easy to use user interface and flow.

For the sakes of the developers, the maintainability of Slopes should be made simple. As time progresses, we may want to expand to mountains outside of those surrounding Metro-Vancouver, therefore the addition or removal of ski mountains should be made easy for developers.

Changelog

- Mentioned cached data in Assumptions and Dependencies
- Added information to the Security section
- Added use of sliders in Change Information use case, to ensure that the user can only enter allowed values.
- Removed features:
    - The showing of road conditions on route to each mountain. Removed due to this information not being accessible using our current APIs.
    - Details on price of skiing at each mountain. Removed due to potential users not seeing this as necessary information.
- Added features:
    - A favourite mountain feature for the user to have a favourite mountain displayed on the home page. Added as requested by potential users.
        - Added use case for this feature.

        - The ability for users to get directions to their desired mountain by clicking a button that will direct them to their native maps app which the current location and the location of the mountain already inputted.
            - Added use case for this feature.
- Updated use cases and requirements to reflect behaviour with no internet connection.