Team [0] - Sprint 4 Deliverables

User Stories:

- As a student, I want to find the price of tickets for ski mountains near me so I can make sure I will be able to afford the tickets.
    - Notes:
        - We could allow users to get a list of nearby mountains along with the price of tickets for each mountain and then sort these mountains in increasing order based on price.
    - Tests:
        - Display information of nearby mountains. After a user provides the information of their current location, the app will come up with a list of nearby mountains with their prices.
        - Sorting the mountains by price in increasing order.

- As a father, I want to know the weather of my favorite mountain that way I can tell my family what clothes they should wear to keep them warm.
    - Notes:
        - We could build an interface for each mountain to include the weather, temperature, and humidity of each mountain.
    - Tests:
        - Display information for the weather of a certain mountain. When users click the button of one mountain, they are taken to an interface and can see the weather, temperature and humidity of this mountain.

- As a beginner, I want to learn how to ski and how to improve my skills, so I can have a good time and complete more advanced trails with ease.
    - Notes:
        - We can reference a list of tutorials with videos categorized based on different levels of skiing and put them in an interface.
    - Tests:
        - Displaying videos. Users can watch these videos without leaving our app/website.
        - Different skill levels of tutorials. These tutorials are categorized from beginner to master.

- As an avid skier, I want to know the specific conditions of trails on the mountains, so I can have a good time with professional equipment and service.
    - Notes:
        - We can include the condition of trails in the mountain interface.

    - Tests:
        - Displaying trail information. When users click the button of one mountain, they can see an interface and know the difficulty, length and condition of the trail (assuming others have commented that day as it would be a crowdsourced feature).

- As a skier, I want to know how others rate the mountain, so I know other skiers opinions and can decide whether a mountain is actually good.
    - Notes:
        - We can include a forum in the mountain interface and add a rating tool to allow users to give stars to mountains. The full mark will be 5 stars.
    - Tests:
        - Test rating mountains by stars.
        - Test leaving comments on the forum.

- As a skier, I want to search mountains based on name, so I can directly find the mountain I want to know.
    - Notes:
        - We can build a search tool and allow users to input the name as the keyword to search mountains.
    - Tests:
        - Test searching by full name.
        - Test searching by partial name.

- As an avid skier, I want to have my own personal account, so I can synchronize data across different devices and enjoy personalized service.
    - Notes:
        - We could build a database to allow users to register their accounts and synchronize information on different devices.
    - Tests:
        - Test sign up.
        - Test log in.
        - Test log out.
        - Test synchronizing information across devices (ie. pulling and pushing to/from the database).

- As a frequent skier, I want to save my favorite mountains, so I can find them directly next time.
    - Notes:

        - We could incorporate favorites into the database to store favorite mountains and allow users to add mountains into it or delete mountains from it.
    - Tests:
        - Test adding favorites.
        - Test deleting favorites.

- As an avid skier, I want to share my best moment with other skiers in this app. That way I can make friends and chat with others who share the same interests as me.
    - Notes:
        - We could build a forum to allow users to post their photos and comments.
    - Tests:
        - Test posting photos.
        - Test posting comments.

- As a skier, I want to know the precise location of my friends, so we can stay connected while skiing on different trails.
    - Notes:
        - We could build a tool to connect users to a group and display the location of them for users in this group.
    - Tests:
        - Test creating a group.
        - Test displaying locations.

Supplementary Specification:

Usability Requirements

1. The app should have an aesthetic UI and reflect that the main target users of this app are skiers.
    Tests:
        i. From surveys conducted among skiers, 7 out of 10 people should think the UI of this app is aesthetic and this app is for skiers.

2. The app could allow users to know how to use the most basic function of this app in a short time.
    Tests:
        i. Fresh users should be able to successfully open the interface of their target mountains within 5 minutes.

Reliability Requirements

1. The app won't run with many failures or bugs.
    Tests:
        i. When completing every 100 tasks in the app for the different devices, the valid bug reports received from the tests should be less than two.

Performance Requirements

1. The app could be able to respond to the requests of users in a short amount time.
    Tests:
        i. Run this app on iPhone X with the most updated iOS and LTE internet service, the longest response time for one click should be less than 5 seconds.

2. The app won't require excess amounts of RAM to be used.
    Tests:
        i. Run this app on iPhone X with the most updated iOS, it should not require more than 512MB of RAM.

Supportability Requirements

1. The app must be able to run on any devices on iOS 11.0.1 or higher.

Tests:

i. This app can indeed work on iPhone, iPad as long as they have installed the iOS 11.0.1 or more recent version.

2. The app should be able to display buttons and interface correctly on devices with different screen size.

   Tests:

   i. Run this app on varying iPhone and iPad models. Buttons will not overlap, disappear, or go out of bounds.

Functionality Requirements

1. The app must contain a function to get the current location of the user, find nearby mountains and sort them by ticket price.

   Tests:

   i. By clicking the nearby mountains button on the main menu. The app will display a list of near mountains which can be sorted by price.

2. The app must contain an interface to show the current weather for one mountain.

   Tests:

   i. In the interface of each mountain, it should contain the current information such as weather, temperature and humidity, and this kind of information should be update hourly.

3. The app should contain a tutorial section which provides a list of tutorials based on different skill levels.

   Tests:

   i. By clicking the "Tutorials" button on the main menu. The app will display a list of videos categorized by skill levels.

   ii. Upon clicking a single video, users will be redirected to the address of the target video. If the video is on YouTube or Vimeo, it will be directly embedded into the application or website.

4. The app should include trail conditions in the interface of each mountain.

   Tests:

   i. In the interface of each mountain, it should contain the trail information such as difficulty, length, and condition.

5. The app should contain a forum in the interface of each mountain and allow users to leave comments and ratings.

   Tests:

   i. Users can input words into a text box, give a rating, and submit the comment/review.

ii. Users can see other comments and rates.

6. The app must have a search function to support users searching for mountains based on a full name or partial name.
   Tests:
   i. By searching "Willard" as the keyword, the app should show "Willard Mountain" and other mountains whose names include this keyword.
   ii. By searching "lard" as the keyword, the app should also show "Willard Mountain" and other mountains whose names include this keyword.

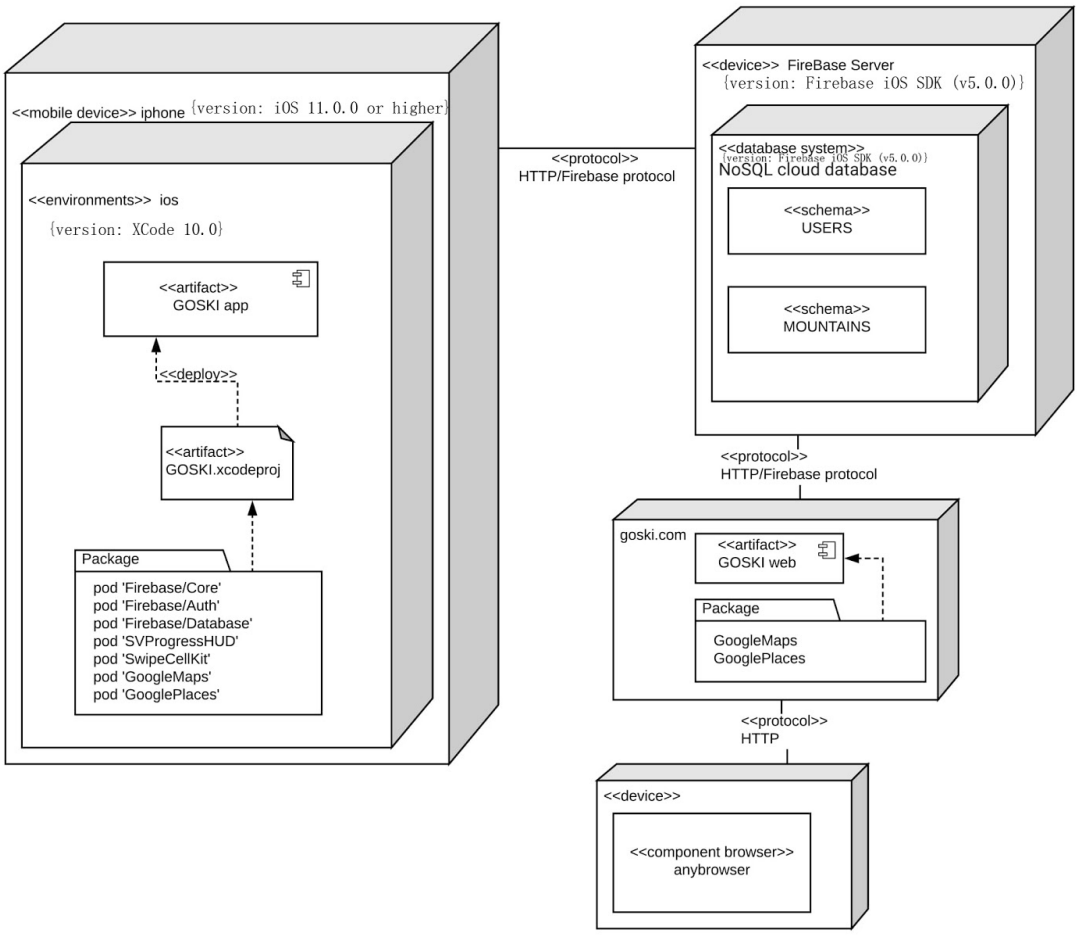
7. The app must contain functions to register an account, log in, and sign up.
   Tests:
   i. Users can register an account with an e-mail address.
   ii. Users can log in an account with an e-mail address and password.
   iii. Users can log out of an account.

8. The app should contain a favorites feature to allow users to add favorite mountains to their account and remove mountains from it.
   Tests:
   i. Users can add mountains to their favorites.
   ii. Users can remove mountains from their favorites.

9. The app must contain a forum on the main menu to allow users to share posts, photos, and can comment on other peoples posts.
   Tests:
   i. Users who have logged in with a valid account can post text and photos on this forum.
   ii. Other users who have logged in with a valid account can leave comments on the posts.

10. The app must contain a function for users to create a group and show each person's location within the group.
    Tests:
    i. Users can create a group and invite other people.
    ii. Users can join a group.
    iii. Users can see other group members' location within the app.

Deployment Diagram



Use Cases:

Use Case: Register	
Identifier:	UC1
Description:	The Register use case models a user registering an account in our system.
Actors:	User
Preconditions:	The database has available space. The email has not been registered before.
Flow of events:	1. The use case starts when the users select Register option. 2. The system prompts users to input email and passwords. 3. Users specify the email and passwords. 4. The system registers an account for the users and automatically logs in for them. The system redirects the user to the main menu.
Postconditions:	Users have registered an account and are redirected to the main menu.

Use Case: Login	
Identifier:	UC2
Description:	The Login use case models a user logging in an account in our system.
Actors:	User
Preconditions:	This account has been registered via Register use case.

Flow of events:	1. The use case starts when the users select Login option. 2. The system prompts users to input email and passwords. 3. Users specify the email and password. 4. If the email matches the password 4.1 The system connects the account with this user. 4.2 The system redirects users to the main menu. else 4.1 The system tells the user the password is incorrect.
Postconditions:	Users log in the software and are redirected to the main menu.

Use Case: GuestLogin	
Identifier:	UC3
Description:	The GuestLogin use case models a user logging in as a guest without an account.
Actors:	User
Preconditions:	The system is installed correctly.
Flow of events:	1. The use case starts when the users select Guest Login option. 2. The system prompts users they are logging in as a guest and redirects them to the main menu.
Postconditions:	Users are redirected to the main menu with a guest tip.

Use Case: NearbyMountain	
Identifier:	UC4
Description:	The NearbyMountains use case models a user finding mountains based on the user's location.

Actors:	User
Preconditions:	The location service in the users' device is enabled.
Flow of events:	1. The use case starts when the users select nearby mountains option. 2. The system displays nearby mountains and the user's current location on a map.
Postconditions:	Nearby mountains are found and displayed in a map.

Use Case: MountainDistance	
Identifier:	UC5
Description:	The MoutainPrice use case models a user finding nearby mountains based on users' location.
Actors:	User
Preconditions:	The location service in users' devices is enabled.
Flow of events:	1. The use case starts when users indicate they want to a list of mountains by distance. 2. The system displays a list of mountains sorted by distance in an increasing order.
Postconditions:	Nearby mountains are found and displayed by distance in an increasing order.

Use Case: MountainInfomation	
Identifier:	UC6

Description:	The MountainWeather use case models a user learning the information for a specific mountain.
Actors:	User
Preconditions:	The information of the mountain has been correctly updated with scripts.
Flow of events:	1. The use case starts when users indicate they want to see specific information of mountains. 2. The system displays a list of nearby mountains and prompts users to specify a mountain. 3. They user specifies a mountain they want to look at. 4. The system displays information of the mountain including location, price, weather and trails.
Postconditions:	The information of one mountain is displayed.

Use Case: Tutorials	
Identifier:	UC7
Description:	The Tutorials use case models a user choosing tutorial videos based upon skill levels and proceeding to watch the tutorial
Actors:	User
Preconditions:	Internet/Wireless Service connection is good (~5 mbps or more).
Flow of events:	1. The use case starts when users select tutorials option. 2. The system displays a list of videos categorized by skill levels. 3. The user specifies one video they want to view. 4. The system redirects the user to the URL of the specified video.
Postconditions:	The videos are categorized by skill levels. The video is displayed.

Use Case: MountainReview

Identifier:	UC8
Description:	The MountainReview use case models a user reading or adding reviews for mountains.
Actors:	User
Preconditions:	Users have logged into our application with an account via use case Login.
Flow of events:	1. The use case starts when users indicate they want to give comments to a mountain by selecting nearby mountain option. 2. The system shows a list of mountain. 3. Users specify a mountain they want to give comments on. 4. The system displays a list of mountain. 5. Users selects the forum of this mountain. 6. The system prompts users to input comments and displays existed comments. 7. Users input comments and choose to send. 8. The system display this comment to others.
Postconditions:	Comments of a mountain are displayed. New comments are added into the screen.

Use Case: SearchMountain	
Identifier:	UC9
Description:	The SearchMountains use case models a user searching for mountains based on the name of a mountain.
Actors:	User
Preconditions:	Users must give a keyword.
Flow of events:	1. The use case starts when users indicate they want search mountains. 2. The system prompts the users to input keywords. 3. Users specify a keyword. 4. If the keyword matches some of the mountain's names. 3.1 The system displays all of them.

	else 3.1 The system prompts that no mountain with this name exists in our database.
Postconditions:	The mountains whose names contain this keyword are displayed or a not found message is prompted.

Use Case: SaveFavorites	
Identifier:	UC10
Description:	The SaveFavorites use case models a user saving a mountain as their favorites and accessing it next time.
Actors:	User
Preconditions:	The user has logged into our application with an account via Login use case.
Flow of events:	1. The use case starts when the users select favorites option for a specific mountain. 2. The system saves the mountain into the user's favorites. 3. Users indicate they want to see the saved favorites. 4. The system displays a list of favorite mountains.
Postconditions:	The designated mountain is saved in the user's favorites. A list of favorites is displayed.

Use Case: DiscussionForum	
Identifier:	UC11
Description:	The DiscussionForum use case models a user reading or adding messages in a forum shared by themselves and other users.
Actors:	User
Preconditions:	Users have logged in our application with an account via Login use case.

Flow of events:	1. The use case starts when users select discussion forum option. 2. The system displays existed messages and prompts the user to input message. 3. Users specify the message and select send option. 4. The system displays this new message to other users.
Postconditions:	Messages sent by other users are displayed.

Use Case: Find Friend	
Identifier:	UC12
Description:	The Find Friend use case models a user finding the precise location of their friends who is skiing on different trails.
Actors:	User
Preconditions:	Users have logged in our application with an account via Login use case. Location function is on. Users and their friends have been connected through our app.
Flow of events:	1. The use case starts when the User select Find Friend option. 2. The system displays a map and prompts users to add friends. 3. Users specify the friend they want to add. 4. The system prompts the target friend that there is a user want to add you. 5. Users specify whether they want to accept this invitation. 6. The system prompts users to share location, if the user accepted. 7. User select share location option 8. The system displays the user and friends location on a map.
Postconditions:	Users locations are displayed to each other and updated timely.

Work Breakdown Structure:

Goski

1. Account
    - 1.1 Sign Up
        - 1.1.1 Connect to the database.
        - 1.1.2 Check duplication account.
        - 1.1.3 Create UI to accept email address and password.
        - 1.1.4 Create the welcome menu and the Sign Up button.
    - 1.2 Login
        - 1.2.1 Connect to the database.
        - 1.2.2 Check correctness of the password. Create a window to show error information if they do not match.
        - 1.2.3 Create UI to accept email address and password.
        - 1.2.4 Create Login button.
    - 1.3 Guest Login
        - 1.3.1 Create Guest Login button.
        - 1.3.2 Create a window to tell the users they are logging in as guests.
        - 1.3.3 Disable functions that can not be used as a guest.

2. Main Menu
    - 2.1 Create the whole layout.
    - 2.2 FindMountains
        - 2.2.1 Create a method to get the user's current location.
        - 2.2.2 Create a method to get the nearby mountains' information.
        - 2.2.3 Draw a map and put mountains and the user's locations on the map.
        - 2.2.4 Create a button and link it to a list of mountains.

2.2.5 Create a button for each mountain in the list and link each of them to a mountain interface.

2.2.6 Create a mountain interface for each mountain including address, weather, and ticket price.

2.3 Search Bar

2.3.1 Create a box to accept keyword.

2.3.2 Implement a method to find mountains based on a keyword.

2.3.3 Pass the results to the mountain list.

2.4 Precise location

2.4.1 Create a button on the main menu and link it to a map.

2.4.2 Create a button and link it to add friends.

2.4.3 Implement a method to get information of users.

2.4.4 Implement a method to display the locations of friends on the same map.

2.5 Instructional Videos (Tutorials)

2.5.1 Create a button on the main menu and link it another interface.

2.5.2 Create three buttons on this interface to represent different skill levels.

2.5.3 Link a list of videos to each of the buttons.

2.6 Add Friends

2.6.1 Implement a method to send friends request.

2.6.2 Implement a window to prompt users the friend invitation.

2.6.3 Connect to the database to store friend lists.

2.7 Favorites

2.7.1 Add a favorites button in the mountain information interface.

2.7.2 Connect to the database and push favorite mountain information as well as user's information into the database.

2.7.3 Create a button on the main menu and link it to a method which gets favorite mountain list from the database.

2.8 Discussion Forum

2.8.1 Create a button on the main menu.

2.8.2 Connect to the database and implement a method to create needed structures in the database.

2.8.3 Implement a box to get the input string.

2.8.4 Implement a method to push the string and the user's name to the database.

2.8.5 Implement a method to get existed message from the database.

3 Specific Mountain Interface

3.1 Basic Information

3.1.1 Display the mountain address passed from Google API.

3.2 Weather

3.2.1 Write scripts to get weather information from ski mountains and store it in the Firebase.

3.2.2 Implement a function to display the weather information in the database on the mountain information interface.

3.3 Price

3.3.1 Write scripts to get ticket prices from ski mountains and store them in the Firebase.

3.3.2 Create an interface to contain price information and display the ticket prices in the database on the app.

3.4 Condition of trails

3.4.1 Write scripts to get the availability of trails from ski mountains and store it in the Firebase.

3.4.2 Implement a function to display the availability of trails in the database on the mountain information interface.

3.5 Rating

3.5.1 Create a button on the mountain information interface and connect it to a forum.

3.5.2 Connect to the database and implement a method to create needed structures in the database.

3.5.3 Implement a box to get the input string.

3.5.4 Implement a method to push the string and the user's name to the database.

3.5.5 Implement a method to get existed message from the database.

4.Other

4.1 Sprint 1 Deliverables

4.1.1 Vision Statement

4.1.2 User Scenarios

4.1.3 Project Schedule

4.1.4 Contribution Summary

4.1.5 Status Report

4.2 Sprint 4 Deliverables

4.2.1 User Stories

4.2.2 Supplemental Specifications

4.2.3 Deployment Diagram

4.2.4 Use Cases

4.2.5 Work Breakdown Structure

4.2.6 Updated Schedule

4.2.7 Contribution Summary and Status Report

4.3 Interim Release

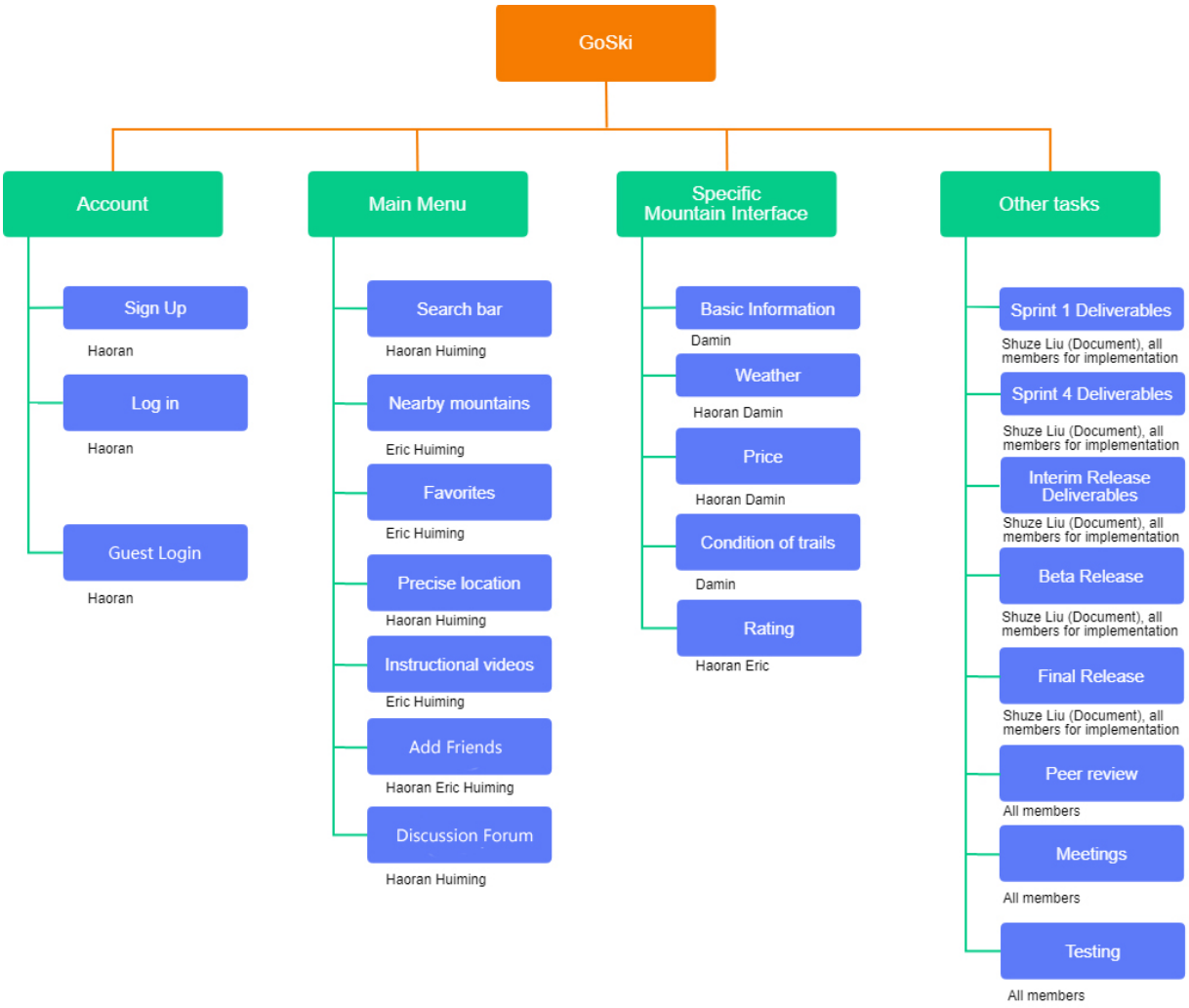
4.3.1 First Demo

4.3.2 Sequence Diagram

4.3.3 Static Class Diagrams

4.3.4 CRC Cards

WBS Overview and Assignment of tasks for each team member.



Updated Schedule

High-Level Project Schedule:

	Goals
Sprint 1 (01/28 - 02/11) Sprint 1 Deliverables on 02/11	Decide the website's layouts. Implement sign up and log in. Implement the function to get locations of Users. Write a document.
Sprint 2: (02/11 - 02/19)	Implement the main menu of our iOS application. Implement scripts to get altitude and weather information of mountains.
Sprint 3: (02/19 - 02/25)	Display mountains sorted by the distance. Implement basic information and weather feature in iOS and website.
Sprint 4: (02/25 - 03/11) Sprint 4 Deliverables on 03/11	Implement the "Nearby Skiers" feature. Summarize and connect each other's work. Write a document.
Sprint 5: (03/11 - 03/18)	Implement the "Search" feature. Implement scripts to get conditions of trails information.
Sprint 6: (03/18 - 03/25) Interim Release on 03/25	Implement the "Comment Forum". Implement the "Price" features. Implement scripts to get conditions of trails information. Release a demo.
Sprint 7: (03/25 - 04/01)	Implement the "Rating" feature. Implement the ability to favorite a mountain to save it
Sprint 8: (04/01 - 04/08)	Implement the "Precise Location" feature. Implement tutorial recommendation feature.
Sprint 9: (04/08 - 04/15) Beta Release on 04/15	Implement the "Best Moments" feature. Test and release a demo.
Final Release (04/15 - 04/25) Final Release on 04/25	Fix bugs and any further improvements necessary.

Features that will be included in interim release:

Account	Sign up
	Log in
Main Menu	Search bar
	Nearby mountains
	Near skiers
Specific Mountain Interface	Basic Information
	Weather
	Price
	Condition of trails

Updated Project Schedule:

	Eric	Shuze	Haoran	Damin	Huiming
Sprint 1 (01/28 - 02/11) 02/11 Sprint 1 Deliverables	Determine the user's current location.	Write documents. Connect to each member.	Implement Sign up and Login Functions.	Write scripts to connect mountains' websites.	Decide the layouts of the website.

Sprint 2: (02/11 - 02/19)	Find ski mountains near users.	Design domain model, user stories.	Implement the main menu.	Get altitude, temperatures, humidity, and weather.	Finish the basic design of the website.
Sprint 3: (02/19 - 02/25)	Display mountains by sorted distance.	Design deployment diagram, use cases.	Implement the “Weather” feature.	Get altitude, temperatures, humidity and weather.	Learn how to use the Google Places API to locate the users’ locations and find nearby mountains.
Sprint 4: (02/25 - 03/11) Sprint 4 Deliverables on 03/11	Summarize and connect.	Summarize and connect.	Implement the “Near Skiers” feature. Summarize and connect.	Summarize and connect data to the database.	Learn how to make the website interact with firebase so that the users can register and post comments to the community.
Sprint 5: (03/11 - 03/18)	Implement the “Comment Forum”.	Draw diagrams.	Implement the “Search” feature.	Get conditions of trails and price information.	Implement the QR code system for the users to add friends more easily.
Sprint 6: (03/18 - 03/25) Interim Release at 03/25	Implement the “Comment Forum”.	Prepare CRC cards, design approach.	Implement the “Price” feature.	Get conditions of trails and price information.	Implement the “Forums”.
Sprint 7: (03/25 - 04/01)	Implement “Favorites” feature.	Summarize and connect.	Implement the “Rating” part.	Summarize and connect data to the database.	Summarize and connect.
Sprint 8: (04/01 - 04/08)	Implement the “Tutorial recommendation” feature.	Prepare testing documents	Implement the “Precise Location” feature.	Summarize and connect to the database.	Implement other parts.
Sprint 9: (Beta Release) (04/08 -	Prepare for the demo.	Connect each part. Prepare for release.	Implement the “Best Moment” feature.	Prepare for release. Improve scripts.	Prepare for release. Finalize a demo

04/15) Beta Release on 04/15					
Final Release (04/15 - 04/25) Final Release on 04/25	Fix potential defects. Do further improvement.	Fix potential defects. Do further improvement.	Fix potential defects. Do further improvement.	Fix potential defects. Do further improvement.	Fix potential defects. Do further improvement.

Contribution Summary

Stakeholders' Name	Contributions
Eric Partridge	1. Implemented function to get the location of users. 2. Implemented function to get the information of mountains and sort them by distance.
Shuze Liu	1. Wrote the document for Sprint 4. 2. Designed user stories, user cases. Updated schedules.
Haoran Hu	1. Further developed the application for iOS devices. 2. Drew deployment diagram.
Damin Xu	1. Implemented scripts to get information of ski mountains.
Huiming Chen	1. Implemented the framework of our website.

Status Report

We are following our schedule and have implemented scripts to get information from mountains' website. Also, the weather function in our iOS application is now available. The framework for our website have been built. We have also built a database in firebase to support our iOS application. At the end of Sprint 4, we finished a document including, user stories, supplemental specifications, deployment diagram, use cases and work breakdown structure. Our next step is to build the search function and comment forum. Throughout the past 4 sprints, we have stayed on schedule and finished all other specific tasks on time.