

Team[0] - Final Release & Best Practices

Final Release

Installable application

Because publishing an application on the App Store will cost money and a long time of verification, we installed our app locally. Our application is installable and finished. The professor has tried our application on iOS devices after Thursday's class.

Installation Guide

You need Xcode to install this app.

To work on this project.

1. (Download) Download this skeleton project.
2. (Firebase stuff) Ask h-h-r for GoogleService-Info.plist and drag it into Xcode navigation bar next to Info.plist.
3. (Pods you need)
 - 'Firebase/Core' 'Firebase/Auth' 'Firebase/Database' 'SVProgressHUD'
 - 'SwipeCellKit' 'RealmSwift' 'ChameleonFramework' 'GoogleMaps' 'GooglePlaces'
 - Try to build project, if Xcode complains about: "no such module ...", type "pod update" in the terminal and hit enter in the terminal; wait till you see "Pod installation complete!"

Final Test Results

Test Cases

TC1: Register [Successful Registration]

Preconditions: App is installed in an iOS device with 10.0.0 or higher version.

The database(Firebase) has available space.

Sequence of Actions and Specific Expected Results:

1. Click Register button on the welcome interface.
GoSki should show an interface to allow user input email and password.
2. Input an email address, passwords for twice.
GoSki should display the email explicitly and display the password as dots.
3. Click Register button
GoSki should register an account for the user and redirect the user to the main

menu.

TC2: Register [Duplicate email]

Preconditions: App is installed in an iOS device with 10.0.0 or higher version.

The database(Firebase) has available space.

Sequence of Actions and Specific Expected Results:

1. Click Register button on the welcome interface.
GoSki should show an interface to allow user input email and password.
2. Input an existed email account, the password and confirmed password.
GoSki should display the email explicitly and display the passwords as dots.
3. Click Register button
GoSki should display a window to tell the user this email has been registered.

TC3: Login [Successful Login]

Preconditions: App is installed in an iOS device with 10.0.0 or higher version.

The database(Firebase) is connected

Sequence of Actions and Specific Expected Results:

1. Click Login button on the welcome interface.
GoSki should show an interface to allow user input email and password.
2. Input an existed email account and the password.
GoSki should display the email explicitly and display the passwords as dots.
3. Click Login button
GoSki should display the main menu.

TC4: Login [Email and password do not match]

Preconditions: App is installed in an iOS device with 10.0.0 or higher version.

The database(Firebase) is connected

Sequence of Actions and Specific Expected Results:

1. Click Login button on the welcome interface.

GoSki should show an interface to allow user input email and password.

2. Input an existed email account and password.

GoSki should display the email explicitly and display the passwords as dots.

3. Click Login button

GoSki should show the Login Failed window to indicate the password and the account do not match.

TC5: Guest [Successful Guest Login]

Preconditions: App is installed in an iOS device with 10.0.0 or higher version.

The database(Firebase) is connected

Sequence of Actions and Specific Expected Results:

1. Click Guest button on the welcome interface.

GoSki should show a window including tips for guests and display the main menu. The Find Friends and Discussion Forum should not be able to used by guests.

TC6: DisplayMountainsOnMap

Preconditions: App is installed in an iOS device with 10.0.0 or higher version.

The database(Firebase) is connected.

Location function is allowed.

Sequence of Actions and Specific Expected Results:

1. Click FindMountains on the main menu.

GoSki should display near mountains on a map and the user's current location.

TC7: ViewMountainsInformation

Preconditions: App is installed in an iOS device with 10.0.0 or higher version.

The database(Firebase) is connected.

Location function is allowed.

Sequence of Actions and Specific Expected Results:

1. Click ViewMountains on a mountain map interface

GoSki should display a list of near mountains.

2. Click a mountain entry.

GoSki should display a interface of the mountain including an address, price, and weather information.

TC8: FindFriends

Preconditions: App is installed in an iOS device with 10.0.0 or higher version.
The database(Firebase) is connected.
Location function is allowed.

Sequence of Actions and Specific Expected Results:

1. Click FindFriends on main menu.
GoSki should display an interface including map.
2. Click Friends button on this interface.
GoSki should display a list of this account's current friends.
3. Click AddFriends button.
GoSki should show a window to allow the user to input an email address of a target friend.
4. Input the email address of target friends.
GoSki should show a window on the target friend's interface.
5. Click Accept.
GoSki should update the friend list to include each other in both users' friend list.
6. Click Share Location
GoSki should display each other's location on the map.

TC9: SendMessageInDiscussionForum

Preconditions: App is installed in an iOS device with 10.0.0 or higher version.
The database(Firebase) is connected.
Users have logged in GoSki with an account.

Sequence of Actions and Specific Expected Results:

1. Click Discussion Forum
GoSki should show a forum.
2. Input a message and click "Send"
GoSki should display this new message which can be viewed in other users' devices. This message should include the email address and the content.

TC10: Tutorials

Preconditions: App is installed in an iOS device with 10.0.0 or higher version.
The database(Firebase) is connected.

Sequence of Actions and Specific Expected Results:

1. Click "Tutorials"
GoSki should a list of videos with links categorized by skill levels.
2. Click a video.
GoSki should redirect the user to the video website.

Test Result

Testing Date: 04/16/2019 20:16

Tester: Haoran

Used Testing Methodologies: Black Box Testing, Unit Testing

Test Case	Result	Comments
TC1	Pass	Unit Testing
TC2	Pass	Unit Testing
TC3	Pass	Unit Testing
TC4	Pass	Unit Testing
TC5	Pass	Unit Testing

Testing Date: 04/17/2019 17:35

Tester: Eric

Used Testing Methodologies: Black Box Testing, Unit Testing

Test Case	Result	Comments
TC6	Pass	Unit Testing
TC7	Pass	Unit Testing

Testing Date: 04/18/2019 21:41

Tester: Haoran

Used Testing Methodologies: Black Box Testing, Unit Testing

Test Case	Result	Comments
TC8	Pass	Unit Testing

Testing Date: 04/19/2019 12:51

Tester: Damin

Used Testing Methodologies: Black Box Testing, Unit Testing

Test Case	Result	Comments
TC10	Fail	Display image of videos successfully. But fail to redirect users to the website.

Testing Date: 04/20/2019 09:05

Tester: Huiming

Used Testing Methodologies: Black Box Testing, Unit Testing

Test Case	Result	Comments
TC10	Pass	Unit Testing

Testing Date: 04/21/2019 21:30

Tester: Shuze

Used Testing Methodologies: Black Box Testing, Integration Testing

Test Case	Result	Comments
TC1	Pass	Integration Testing
TC2	Pass	Integration Testing
TC3	Pass	Integration Testing
TC4	Pass	Integration Testing
TC5	Pass	Integration Testing
TC6	Pass	Integration Testing
TC7	Pass	Integration Testing
TC8	Pass	Integration Testing
TC9	Pass	Integration Testing
TC10	Pass	Integration Testing The bug mentioned on 04/19 has been fixed.

Testing Date: 04/23/2019 17:20

Tester: Shuze

Used Testing Methodologies: Black Box Testing, Integration Testing

Test Case	Result	Comments
TC1	Pass	After improvements of source code and fixing of bugs, our application passes all the test cases in integration testing.
TC2	Pass	
TC3	Pass	
TC4	Pass	
TC5	Pass	
TC6	Pass	
TC7	Pass	
TC8	Pass	
TC9	Pass	
TC10	Pass	

Test Conclusion:

Based on the above results, we think our application is ready for production use.

Contribution Summary

Stakeholders' Name	Contributions
Eric Partridge	<ol style="list-style-type: none">1. Further improved the source code. Fixed bugs.2. Prepared for the group presentation.3. Tested our application.
Shuze Liu	<ol style="list-style-type: none">1. Wrote the final release document.2. Prepared for the group presentation.3. Tested our application.
Haoran Hu	<ol style="list-style-type: none">1. Further improved the source code. Fixed bugs.2. Prepared for the group presentation.3. Tested our application.4. Wrote a document for best practices and application installation guide.
Damin Xu	<ol style="list-style-type: none">1. Further improved the source code.2. Prepared for the group presentation.3. Tested our application.
Huiming Chen	<ol style="list-style-type: none">1. Prepared for the group presentation.2. Tested our application.

Status Report

We followed our test plan and finished tests of our software. We detected one bug and have fixed it. After unit testings, we have two rounds of integration testings. Also, between two integration testings, we improved the quality of our source code and used the integration testing to ensure the reliability of the improvements. We wrote corresponding documents for the testing results and an installation guide. Furthermore, we summarized the techniques used in our project in the "Best Practice". We have prepared slides and presentations for the Final Release. Every group member has a great attitude to the whole project and contributes a lot. We showed our demo in class and successfully finished our project!

Best Practices:

1. Project Management Web Site

We are using google docs and Github as our project management web sites. Google docs is where we wrote sprint reports and project related information.

Github:

<https://github.com/h-h-r/GOSKI>

Bug Tracking

<https://github.com/h-h-r/GOSKI/issues?q=is%3Aissue+is%3Aclosed>

We used bug tracking tools provided by Github to track our bugs and marked them as "Closed" when we fixed them.

















List of Bugs:

Descriptions	Status
Video icons cannot redirect users to Youtube websites.	Fixed
App crashes when press the button in friends list.	Fixed
Need to reenter the friends list view to update friends table.	Fixed
Unable to process friend request.	Fixed
Friends location won't update.	Fixed
Unable to see friends' location in the map.	Fixed
Unable to turn off sharing location with friends functionality.	Fixed
Merge conflict about GOSKI.xcworkspace.	Fixed
Unable to read/write to firebase.	Fixed
Warnings in Xcode not resolved.	Fixed
Tabel view cell and destinationVC unmatched.	Fixed
Corrupted data of lift ticket price.	Fixed
Search bar layout issue.	Fixed

Google docs:

<https://drive.google.com/drive/folders/1gJIR6XrRbtylZbYxvq45vqRGOOnExVtPk?usp=sharing>

We used google docs to achieve collaboration on documents and created a folder for our documents.

My Drive > Team[0] Reports ▾			🗒	ⓘ
Name ↑		Owner	Last modified	
 Team[0] - Best Practices 		Haoran	12:08 PM me	
 Team[0] - Beta Release 		me	12:10 PM me	
 Team[0] - Final Release		me	12:07 PM me	
 Team[0] - Guide for Source Code		me	12:10 PM me	
 Team[0] - Interim Release Deliverables 		me	12:10 PM me	
 Team[0] - Peer Reviews 		me	11:10 AM me	
 Team[0] - Sprint 1 Deliverables 		me	11:11 AM me	
 Team[0] - Sprint 4 Deliverables 		me	12:08 PM me	
 Team[0] - Status Report 		me	12:05 PM me	

2.Code repository

We're using GitHub as the code repository. We achieved code merging and code backtracking with Github. And our coding activities have been recorded in our Github.

Github:

<https://github.com/h-h-r/GOSKI>

h-h-r / GOSKI

Watch 0 Star 2 Fork 0

Code Issues 1 Pull requests 0 Projects 0 Wiki Insights Settings

No description, website, or topics provided. [Edit](#)

[Manage topics](#)

74 commits 1 branch 0 releases 2 contributors

Branch: master New pull request Create new file Upload files Clone or download Find File

h-h-r Update README.md	Latest commit c75ac0a 27 minutes ago
GOSKI.xcodeproj	fixed issue with scrolling and watching tutorial videos 3 days ago
GOSKI.xcworkspace	minor ui tweaks 2 days ago
GOSKI	minor ui tweaks 2 days ago
Pods	tutorial and price formountains implemented 10 days ago
.DS_Store	added temperature, weather and trails to mountain info 10 days ago
.gitignore	added .DS_Store to gitignore a month ago
Podfile	finished main forum 11 days ago
Podfile.lock	finished main forum 11 days ago
README.md	Update README.md 27 minutes ago

README.md

GOSKI

3.Documented Coding Standards

We are following the “Swift Style Guide” for coding standards. This style guide is based on Apple’s excellent Swift standard library style and also incorporates feedback from users across multiple Swift projects within Google. It is a living document and the basis upon which the Formatter is implemented.

<https://google.github.io/swift/>

4.Use the basic concepts behind object oriented design

Polymorphism:

One of the benefits of using Xcode is a number of libraries and classes that are made available during development. This came in handy when we could inherit the classes and override their functions. For example, one of the most common base classes that we used was UIViewController. Several of our classes are subclasses of this class so that we don’t need to dive into the detail of initialization or termination of a view that is going to appear or leave the screen.

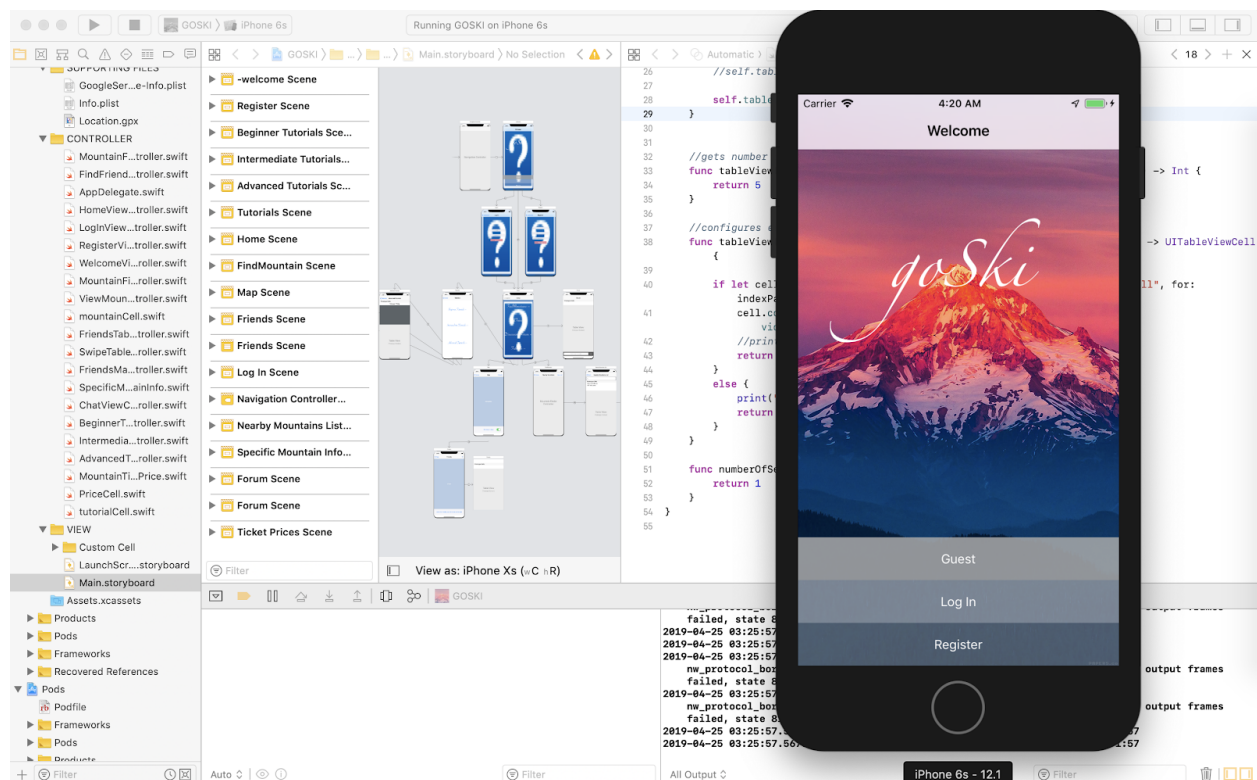
High cohesion, low coupling and data encapsulation:

According to the class diagrams and CRC cards shown in the previous sprint reports, all classes are created according to abstraction, low coupling, high cohesion and data encapsulation principles. To be more specific, all of the view controller and table view controllers inherit swift built-in classes and almost every class acquires and processes data locally without collaborating with other classes. There are only a few cases when a class needs to use another class. For example, when FriendListViewController needs to store a user's friend's datagram using FriendItem class, which still follows the abstraction and data encapsulation principle.

5. Build tools

Our project was built using Xcode. It allows us to build our application and test it either on the real mobile device or an iPhone emulator with kinds of versions.

<https://developer.apple.com/xcode/>



6. Third Party Component or Tool

We're using CocoaPods which is a dependency manager for Swift and Objective-C Cocoa projects. It has over 60 thousand libraries and is used in over 3 million apps. CocoaPods can help us scale projects elegantly. In the profile, we have used following pods in our project: 'Firebase/Core', 'Firebase/Auth', 'Firebase/Database', 'SVProgressHUD', 'SwipeCellKit', 'RealmSwift', 'ChameleonFramework', 'GoogleMaps', 'GooglePlaces'

<https://cocoapods.org>

7.Design Patterns

In the iOS application, one of the design patterns we are using is the model, view, and controller design pattern also known as the **MVC pattern**. Since we have numerous different views that the user can see but a single set of data for ski mountains we needed it to be easily accessible. The model consists of a list of ski mountains objects that store information about each mountain such as the name of the mountain, latitude, and longitude, address and distance from the user. The views are each screen the user can see from the welcome screen to the find nearby mountains screen to the find my friends screen. The controller deals with communication between the model and the view. It can access the list of ski mountain objects and make a marker on the map for each mountain. It can also adjust the map camera and move it so it is centered as the users current location as well as adjusting how zoomed into the map you are and what type of map you are looking at.

We also incorporated the **singleton pattern** into our class managing the list of ski mountains. We did this as it is not necessary to create multiple instances of the same group of data. This way after initially creating the list of ski mountains nearby and you want to display a list of them to the user, you don't need to recreate the list and calculate all the distances again.

In addition, we have used the **observer pattern** in multiple places in our iOS application. For example, in the GUI aspect, whenever we put a UIButton in the user interface and link that UIButton to the method usually called "UIButtonPressed", this means we have created an observer. As long as the button is present on the current view controller, the button is observing whether it gets pressed by someone. If somebody does this, the method linked to it will be triggered to perform corresponding events, ie. go to the next view controller, present UIAlert. The other case worth using the observer pattern is in the find friends feature since we need to update friends' locations on the map in real time. This was done by using the powerful API in Firebase, and providing the observer methods with the appropriate database path, we're able to retrieve the other users' longitude and latitude data from firebase in real time.