Software Engineering

Team X – Project Report

‘FishON’ – Inception Document

|  |  |
| --- | --- |
| **SOME GROUP INFORMATION HERE** |  |
| **Semester:** |  |
| **Instructor:** | Brad Armitage <[Bradford.Armitage@metrostate.edu](mailto:Bradford.Armitage@metrostate.edu)> |
|  |  |

**Table of Contents**

[Vision: 3](#_Toc516580011)

[Scope: 3](#_Toc516580012)

[Business Case: 4](#_Toc516580013)

[Risks: 5](#_Toc516580014)

[Candidate Architecture: 6](#_Toc516580015)

[FURPS+: 7](#_Toc516580016)

[User Stories: 10](#_Toc516580017)

[Use Cases: 10](#_Toc516580018)

[Glossary 22](#_Toc516580019)

# Vision:

***“To bring near real time fishing and social information to all anglers, enabling them to get the most out of their fishing experience.”***

# Scope:

The project ***high-level*** scope as follows:

1. Mobile app runs on both iOS and Android devices
2. Provide information on friends' activity feed, latest news, leaderboards.
3. Provides other fishing information such as:
   1. Weather information
   2. Lake information such as contour of lake, temperature, clarity, fish present.
   3. Which lakes seem to be ‘hot’ or ‘cold’
4. Ability to interact with other anglers
   1. Post pictures with fish
   2. Share hotspots (GPS coordinates)
   3. Lake/Regional competition
   4. Event creation/sharing/invitation
   5. Unique tips like ‘type of bait works here’, ‘fish here at this time’, ‘lots of this type of fish’
   6. Messaging between users
5. Monthly subscriptions with external credit cards vendor
6. Email notification and confirmation through external email service provider

Out of scope: browser access, annual subscriptions, subscription refunds, online chat

# Business Case:

There are several apps similar to our vision currently on the market.

**Fishbrain:**

Fishbrian is a social media fishing app on iOS and Android that connects anglers allowing them to post pictures with their fish, share fishing techniques, post spots where the caught fish, and shows lake info.

The app has premium and free accounts. Premium allows you to view lake contours and bait packages that work well with certain species of fish along with catch locations submitted by other users.

Fishbrain can link with Facebook to have friend lists synch up.

**Fishidy:**

Fishidy offers a contoured map that users can upload spot information along with pictures at the location of a fish caught.

Great for locations with high density of users but lacks a real public database beyond what users submit.

**Navionics:**

Navionics is GPS and lake contouring app that allows users to view and submit edits to contour of lake.

There are no social media aspects, but it does offer the most accurate mapping info to users and is widely used in the fishing community.

All of these apps offer aspects of what we are trying to create but all fall short of executing our vision up to the standards we have set. None of these apps work off publicly available data, and often fall short in their execution of key features. For Fishbrain the map feature is poorly executed. Fishidy has social media aspects but they are limited and not as well implemented as Fishbrain. Navionics has no social media aspect however it does offer the best lake maps on the market, at a steep price to the user. To execute our vision we believe that developing our own app is the way to go rather than buying one of the apps currently available on the market.

# Risks:

The majority of our risks comes from working with external software, databases, and companies.

**Public data set**:

A big part of our app will be pulled from public data sets: lake maps, lake contours, lake temperature, and weather information. The risk is we don’t have control if their servers go down or if/when they make updates to their system that could affect how our app function. Which can affect our user experience and our competitive edge over other existing apps.

**Credit card payments platform:**

We are working with Dolphin Credit to handle credit cards payment transactions. The risk will be relying on Dolphin Credit to handle all payments and working with them on license/contracts.

**Working with Apple AppStore and Google Play Store:**

Since our application will work with iOS and Android, a part of our risk will be making sure we fulfill the terms and requirements set forth by Apple and Google Play Store, as well as maintain the fulfillment.

**Ads revenue from vendors:**

Some of our revenue will come from ads we charge companies to host on our sites. Working to get new companies, maintain current ones, and update the ads is a risk because we would have to work out contracts set forth by both parties.

**Integrating with email servers:**

Since users will be using their email address to establish an account with us, reset password, receive notifications and confirmations. The risk will be to find a relying on BlueWhale to be dependable and reliably send out emails we need them to.

**GPS system on users’ phone:**

We must work with the user’s phone GPS system to determine their location and populate their information if they so choose. Since it’s not through us directly, the reliability is unstable and poses a risk we must work with carefully.

# Candidate Architecture:

The mobile app is intended to run natively on both iOS and Android devices. Its functionality is broadly classified into four categories: a profile management, the ability to browse information, social interactions or activities, and system administration. (See figure 1)

**FishON User**

**Staff Member**

**System Administration**

**Profile Management**

**Social Interaction/**

**Activities**

**View/Browse Information**

**Figure 1**: A depiction of the main user/system interactions

# FURPS+:

The following section describes the FishON project Function, Usability, Reliability and Security related non-functional requirements. These help define the architectural requirements and key support features to efficiently run and maintain the application.

**Functional Requirements:**

1. Licensing
   1. The application will be deployed and maintained in both Apple App store and Google Play store.
   2. There will be a onetime fee to register with App store and Google Play store
   3. Upon registration, two administrators or staff members of the FishON IT team will maintain app store Unique ID and code deployment responsibility. (One primary and another as shadow backup)
   4. Credit cards and Payment transactions will be outsourced to ‘Dolphin Credit’ vendor
   5. The licensing and contracts with ‘Dolphin Credit’ will be reviewed and renewed on an annual basis
   6. All email transactions will be outsourced to ‘BlueWhale’ email service provider
2. Localization
   1. The application will be maintained and available to users in ‘English’ language only
3. Email Servies
   1. The FishON application will interact with ‘BlueWhale’ email service provider. The email service will handle all the email transactions to
4. Security
   1. The app will support ‘https’ protocol
   2. All user interactions with system will be on ‘https’ protocol on 4G LTE or WiFi networks
   3. The database will be encrypted and protected by passwords
   4. The passwords will be regularly changed every 90 days by the System Administrators
   5. Minimum two system administrators will be authorized to access the protected infrastructures like Database, Licenses and app store unique IDs
   6. All software licenses will be maintained in electronic vault for auditing and maintenance purposes
5. Replication
   1. FishON IT team will maintain 3 backups of the system information including databases

**Usability Requirements:**

1. Accessibility
   1. The system will be designed and developed to follow the app store & Google Play store guidelines
2. Aesthetics
   1. The development team will consider using the best practices used in User interface designs
   2. The look and feel of the application will be native to iOS or Android devices in which it is running
   3. The user interface will have consistent behavior in both iOS and Android devices. (Example: The home screen, menu options)

**Reliability Requirements:**

1. Availability
   1. The app will be deployed in App store and Google Play store
   2. The availability will depend on the app store uptime and availability.
   3. The database and related information will be available up to 99.95% of time
2. Recoverability
   1. When system detects a failure, the Recovery Point Objective (RPO) will be set to 1 hr from the time of failure
   2. When system detects a failure , the Recovery Time Objective (RTO) will be set to 15 mins from the time of failure

**Performance Requirements:**

1. Response Time
   1. When system interacts with other external systems such as payment processing system and database, the transaction will be complete in maximum of 10ms
   2. An appropriate error message will be displayed to user in case of timeout delays
2. Throughput
   1. The system will have a load balancing mechanism
   2. System will support maximum of 100 ‘Concurrent user’ transactions at a time

**Supportability Requirements:**

1. Auditability
   1. System will maintain the audit trail of
      1. User payment transactions
      2. User account deletion
2. Compatibility
   1. The application will run on iOS 9+ versions
   2. The application will run on 5.0 + (Lollipop) version onwards
3. Testability
   1. The app will be testable on local developer machine with iOS and Android development environment
   2. The app will be testable on iOS and Google Play Store sandboxes before being deployed to production

**“+” Design Constraints:**

* + 1. Database

1. The transaction data will be maintained in ‘Oracle 11G Release 2’ version
2. The social interaction data such as friends and contacts will be maintained in graph database management system Neo4

**“+” Implementation Constraints:**

1. External System – BlueWhale email service provider
   1. System will interface with email server to process and the email transactions
2. External System – ‘Dolphin Credit’ payment processing system
   1. System will interface with payment processing system to handle all the credit card payments
3. Application platforms
   1. The application will be deployed and run in iOS and Android operating system compatible devices

# User Stories:

For a list of user stories view the accompanying artifact (FishON user stories.xlsx)

# Use Cases:

The following section is a preliminary selection of eight use cases for the proposed software.

**List of use cases**

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID | Use Case Name | Description | Page |
| UC\_SI\_01 | Send Friend Request | This use case describes to send a friend request on the app | 11 |
| UC\_SI\_02 | Accept friend request | This use case describes accepting a friend request on the app | 12 |
| UC\_SI\_10 | Post Photo to profile repository | This use case describes posting a photo to the user’s profile | 13 |
| UC\_PM\_01 | User Login | This use case describes how user logs in to the app using an email address and password | 14 |
| UC\_PM\_05 | Reset forgotten password | User can reset password through pressing Forgot Password on sign in page | 15 |
| UC\_PM\_10 | Upgrade to pro | This use case describes how user can upgrade to a Pro membership by paying with credit card | 17 |
| UC\_SA\_01 | Delete User Account | This use case describes how a system administrator deletes an account from the system | 19 |
| UC\_BI\_02 | View weather information | This use case describes how a user will be able to view weather information for a specific address or zip code | 20 |

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC\_SI\_01 | |
| Use Case Name | Send Friend Request | |
| Description | This use case describes to send a friend request on the app | |
| Actors | User A (Tier 2 or 3), User B (Tier 2 or 3) | |
| Trigger | User A selects “add friend” button on User B’s profile | |
| Pre-conditions | User A and User B are not currently friends, User A is on User B’s profile | |
| Post conditions | User B receives friend request | |
| Process Flow / Data Flow | | |
| <Place for flow diagrams> | | |
| Basic Flow | | |
| Step | **Description** | **Alternate Flow/ Error Flow #** |
| 1 | User A presses “add friend” button |  |
| 2 | System notifies User B that User A has added them as a friend |  |
| Alternate Flows | | |
| Step | **Description** | |
| 1 |  | |
| 2 |  | |
| Exception / Error Flows | | |
| Step | **Description** | |
|  |  | |
|  |  | |
| Extension Points | | |
|  | | |

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC\_SI\_02 | |
| Use Case Name | Accept friend request | |
| Description | This use case describes accepting a friend request on the app | |
| Actors | User A (Tier 2 or 3), User B (Tier 2 or 3) | |
| Trigger | User A selects “add friend” button on User B’s profile | |
| Pre-conditions | User A and User B are not currently friends, User A has sent a friend request. Refer to UC\_SI\_01 | |
| Post conditions | User B accepts or denies the User A friend request | |
| Process Flow / Data Flow | | |
| <Place for flow diagrams> | | |
| Basic Flow | | |
| Step | **Description** | **Alternate Flow/ Error Flow #** |
| 1 | System sends User B a notification User A has added them as a friend |  |
| 2 | User B presses “accept” button | A1. User B presses “deny button” |
| 3 | System correlates User A and User B as friends in their contact list |  |
| 4 | System removes friend request |  |
| Alternate Flows | | |
| A1 | **User B denies the friend request** | |
| Step | **Description** | |
| 1 | User B presses “deny button” | |
| 2 | System removes the notification from User B | |
| Exception / Error Flows | | |
| Step | **Description** | |
|  |  | |
|  |  | |
| Extension Points | | |
|  | | |

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC\_SI\_10 | |
| Use Case Name | Post Photo to profile repository | |
| Description | This use case describes posting a photo to the user’s profile | |
| Actors | User (Tier 2 or 3) | |
| Trigger | User selects “add photo” button | |
| Pre-conditions | User is currently on their | |
| Post conditions | User successfully uploads photos to user profile repository | |
| Process Flow / Data Flow | | |
| <Place for flow diagrams> | | |
| Basic Flow | | |
| Step | **Description** | **Alternate Flow/ Error Flow #** |
| 1 | User presses “add photo” button |  |
| 2 | System accesses the phone’s gallery | E1. There is no gallery |
| 3 | System prompts user to select a photo |  |
| 4 | User selects a photo from their gallery | E2. User exits gallery |
| 5 | System uploads photo to user Profile photo repository. |  |
| 6 | System prompts the user to make the photo ‘Private’ or ‘Public’ | A1. The user selects to make the photo remain as private in their collection |
| 7 | System makes the recently uploaded photo visible to all friends |  |
| Alternate Flows | | |
|  | | |
| Step | **Description** | |
| A1 | The user selects to make the photo remain as private in their collection | |
| 1 | The uploaded photo remains as private. | |
| Exception / Error Flows | | |
| Step | **Description** | |
| E1. There is no pictures in the phone gallery | | |
| 1 | System alerts user that gallery is missing | |
| 2 | System displays a message to close the app and restart | |
| E2. User Exits Gallery | | |
| 1 | System aborts process | |
| 2 | User is returned to their profile | |
| Extension Points | | |
|  | | |

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC\_PM\_01 | |
| Use Case Name | User Login | |
| Description | This use case describes how user logs in to the app using an email address and password | |
| Actors | User (Tier 2 or 3) | |
| Trigger | User wants to access account | |
| Pre-conditions | User already created a profile | |
| Post conditions | User is logged into system and can view homepage. | |
| Process Flow / Data Flow | | |
| <Place for flow diagrams> | | |
| Basic Flow | | |
| Step | **Description** | **Alternate Flow/ Error Flow #** |
| 1 | User goes to sign in page in app |  |
| 2 | System prompts user for email address and password |  |
| 3 | User enters email address and password |  |
| 4 | User press “Sign In” | E1 Missing required fields  E2 User enters incorrect email address and or password |
| 5 | System validates the entered email address and password |  |
| 6 | System logs user in |  |
| Alternate Flows | | |
| Step | **Description** | |
|  |  | |
| Exception / Error Flows | | |
| Step | **Description** | |
| E1 | **Missing required fields** | |
| 1 | System prompts for missing fields | |
| 2 | Go to basic flow 3 | |
| E2 | **User enters incorrect email address and or password** | |
| 1 | System displays error message | |
| 2 | System prompts user to reenter email address and or password | |
| 3 | Go to basic flow 3 | |
| Extension Points | | |
|  | | |
|  | | |

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC\_PM\_05 | |
| Use Case Name | Reset forgotten password | |
| Description | User can reset password through pressing Forgot Password on sign in page | |
| Actors | User (Tier 2 or 3) | |
| Trigger | User selects the ‘Forgot Password’ option on Sign in page. | |
| Pre-conditions | User is on the sign-in page | |
| Post conditions | User has a new password and is able to log in | |
| Process Flow / Data Flow | | |
| <Place for flow diagrams> | | |
| Basic Flow | | |
| Step | **Description** | **Alternate Flow/ Error Flow #** |
| 1 | User presses on “Forgot Password” from the sign in page. |  |
| 2 | System displays message verifying that user wants an email to be sent to their email address with options to reset password, “Continue” or “Cancel” |  |
| 3 | User presses “Continue” | A1 User presses “Cancel” |
| 4 | System prompts user to enter the user’s email address |  |
| 5 | User enters email address and press “Enter” |  |
| 6 | System verifies the email address is in database | E1 Missing or wrong email address field |
| 7 | System sends the email to the specified email address with a link to change password |  |
| 8 | User click on link from email (External event outside of the app) |  |
| 9 | System takes user to Change Password page |  |
| 10 | System prompts the user to enter new password and reenter new password |  |
| 11 | User provides required fields and press enter | E2 Missing fields or non-matching fields |
| 12 | System validates that the two password fields are the same |  |
| 13 | System updates user password |  |
| 14 | System display message to user that their password is updated |  |
| 15 | System takes user to sign in page |  |
| Alternate Flows | | |
| Step | **Description** | |
| A1 | **User cancels Forgot Password flow** | |
| 1 | User is taken back to the sign in page | |
| 2 | User can then enter correct email address and password to go to Homepage or go back to Basic Flow 1 | |
|  |  | |
| Exception / Error Flows | | |
| Step | **Description** | |
| E1 | **User provides the wrong email address or doesn’t provide one** | |
| 1 | System displays error message | |
| 2 | Go to Basic Flow 4 | |
| E2 | **User provides non-matching fields or doesn’t provide both fields** | |
| 1 | System displays error message | |
| 2 | Go to Basic Flow 4 | |
| Extension Points | | |
|  | | |

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC\_PM\_10 | |
| Use Case Name | Upgrade to pro | |
| Description | This use case describes how user can upgrade to a Pro membership by paying with credit card | |
| Actors | User (Tier 2) | |
| Trigger | User wants to upgrade to Pro membership | |
| Pre-conditions | User already created a profile | |
| Post conditions | User becomes a Pro member with access to enhanced features | |
| Process Flow / Data Flow | | |
| <Place for flow diagrams> | | |
| Basic Flow | | |
| Step | **Description** | **Alternate Flow/ Error Flow #** |
| 1 | User presses on Membership icon |  |
| 2 | System takes user to Membership page |  |
| 3 | User chooses “Upgrade to Pro membership” |  |
| 4 | System displays monthly subscription price and asks user to confirm to proceed |  |
| 5 | User presses “Confirm” | A1 User presses “Cancel” button |
| 6 | System populates the following fields for user   * Name * Address * City * State * Zip Code * Phone Number |  |
| 7 | System prompts the user to enter billing information with following fields   * Credit card number * Name on card * Card expiration date * Card security code * Billing Address Line 1 * Billing Address Line 2 * Billing City * Billing State * Billing Zip Code |  |
| 8 | User verifies auto-populated fields and provides required fields and presses “Check-out” | E1 Missing required field(s) |
| 9 | System authorizes credit card information with credit card system data | E2 User enter invalid credit card information |
| 10 | System informs user their card will be charged this day next month if membership is monthly unless membership is cancelled and prompts user to complete transaction |  |
| 11 | User presses “Complete transaction” |  |
| 12 | System takes user to the thank you and congratulations page. |  |
|  |  |  |
| Alternate Flows | | |
| Step | **Description** | |
| A1 | User presses the “Cancel” button | |
| 1 | Returns the user to the membership page | |
| Exception / Error Flows | | |
| Step | **Description** | |
| E1 | **Missing required field(s)** | |
| 1 | System displays error message with missing information highlighted | |
| 2 | Go to Basic Flow 8 | |
| E2 | **User enter invalid credit card information** | |
| 1 | System displays error message | |
| 2 | System takes user back to Check Out screen as in Basic Flow 8, with missing required fields highlighted | |
| 3 | Go to Basic Flow 9 | |
| Extension Points | | |
|  | | |

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC\_SA\_01 | |
| Use Case Name | Delete User Account | |
| Description | This use case describes how a system administrator deletes an account from the system | |
| Actors | Systems Administrator | |
| Trigger | A staff member wants to delete an inactive user account | |
| Pre-conditions | * Systems Administrator has permission to access all user accounts * An inactive account exists | |
| Post conditions | * Inactive user membership as pro-user is terminated * User cannot access the account | |
| Process Flow / Data Flow | | |
| <Place for flow diagrams> | | |
| Basic Flow | | |
| Step | **Description** | **Alternate Flow/ Error Flow #** |
| 1 | Administrator logs in with systems administrator capabilities |  |
| 2 | Administrator selects the account to delete |  |
| 3 | System prompts the administrator to confirm delete | A1. Administrator selects “Cancel” option |
| 4 | System deletes the account |  |
| Alternate Flows | | |
| Step | **Description** | |
| A1 | Administrator selects “Cancel” option | |
| 1 | System returns the user to user management page | |
| Exception / Error Flows | | |
| Step | **Description** | |
|  |  | |
|  |  | |
| Extension Points | | |
|  | | |

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC\_BI\_02 | |
| Use Case Name | View weather information | |
| Description | This use case describes how a user will be able to view weather information for a specific address or zip code | |
| Actors | User (Tier 1, 2 & 3) | |
| Trigger | User selects weather icon/option from menu to view weather information | |
| Pre-conditions | 1. User is viewing lake information 2. User selects ‘Weather’ option from the main menu | |
| Post conditions | User views the weather information for the specific address or zip code | |
| Process Flow / Data Flow | | |
| <Place for flow diagrams> | | |
| Basic Flow | | |
| Step | **Description** | **Alternate Flow/ Error Flow #** |
| 1 | User selects weather icon available on a lake view |  |
| 2 | System fetches weather information of current lake | E1. Unable to determine the current location information |
| 3 | System displays weather information in the overlay | E2. System unable to fetch the weather information |
| 4 | User views the following weather details on the overlay screen   * Current temperature * Day’s High and Low temperature * Days forecast (Sunny, Cloudy Rainy) * Precipitation % * Wind (MPH) * Pressure * 5 Day forecast (Date – [day-date], Forecast, High | Low) |  |
| Alternate Flows | | |
| Step | **Description** | |
|  |  | |
| Exception / Error Flows | | |
| Step | **Description** | |
| E1 | **Unable to determine the current location information** | |
| 1 | System displays an error message indicating that there is no GPS info available. | |
| 2 | System returns the user to Lake view page | |
| E2 | System unable to fetch the weather information | |
| 1 | System displays an error message indicating that unable to fetch weather information. | |
| 2 | System returns the user to Lake view page | |
| Extension Points | | |
|  | | |

# Glossary

***Angler*:** A person who fishes with a hook and line.

***Android*** an open-source operating system used for smartphones and tablet computers. Developed by Google & Open Handset Alliance.

***Bait*** food used to entice fish or other animals as prey.

***GPS*** global positioning system.

***GPS coordinates*** are a unique identifier of a precise geographic location on the earth, usually expressed in alphanumeric characters. Coo

***iOS*** an operating system used for mobile devices manufactured by Apple Inc.