Team 4

arizona state university | ser 401 – senior project

Skoovy

Software requirements

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Version No. | Author | Revision Date | Approved By | Approval Date | Reason |
| 1.0 | Tyler Cole  Anthony Kowal  Lilith Matthews  Rudi Wever | 10-23-2016 |  |  | Initial Software Requirements Draft |
| 2.0 | Tyler Cole  Anthony Kowal  Lilith Matthews  Rudi Wever | 11-13-2016 |  |  | Revised based on comments |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

|  |
| --- |
| Table of Contents |
| 1 Introduction |
| **Scope:** ***Items that haven’t been specified yet are shown as questions in red font.***  The objectives of the Skoovy project are as follows:   * Allow users to create video (30sec. maximum) or photo * Allow users to add FX from market   + Free FX   + Buy FX   + Sell FX * Allow users to make posts Private or Public * Add Geolocation information to posts (are they really added, or are they included when content is generated?) * Allow users to add Hashtags to posts * Allow users the ability to search by:   + Location   + Hashtag   + Popular   + Notifications * Allow users the ability to turn search on/off * Allow users the ability to get notified on geolocation and hashtags * Profiles will be able to follow users, request location, by hashtag * Users will earn points for responding to requests. * Users will lose points for making a request. * Posting are categorized by type:   + Food   + Events   + Place   1.1 Product perspective |
| 1.2 Use case models of the system’s features |
| Use Case 1: User creates a video  Actors: User, Skoovy  Features: Camera access, network access, GPS  Use Case 2: User adds FX from Market  Actors: User, Skoovy  Features: Camera access, network access, GPS, Market  Use Case 3: Make posts private  Actors: User, Skoovy  Features: Camera access, network access, GPS  Use Case 4: Use Geolocation to Post  Actors: User, Skoovy  Features: Camera access, network access, GPS  Use Case 5: Add hashtags to a post  Actors: User, Skoovy  Features: Camera access, network access, GPS, word suggestions, tagging  Use Case 6: Search  Actors: User, Skoovy  Features: Camera access, network access, GPS, Search  Use Case 7: Notifications  Actors: User, Skoovy  Features: Camera access, network access, GPS, notifications, push notifications  Use Case 8: Following  Actors: User, Skoovy  Features: Camera access, network access, GPS, search, user profiles, privacy settings  Use Case 9: Requests  Actors: User, Skoovy  Features: Camera access, network access, GPS, search, user profiles, privacy settings, points  Use Case 10: Respond to requests  Actors: User, Skoovy Features: Camera access, network access, GPS, search, user profiles, privacy settings, points 1.3 Interaction design specification |
| The design of the app is intended to give the user easy access to five core features.  First, before the user gains access to the apps features, they must login. New and Existing user will be shown this screen upon the app’s first launch post installation. Existing users need only enter their credentials to gain access to the remainder of the app. New users will have the option to create an account from within the app. The user’s phone number will be required for sign-up with it being verified via a code sent via SMS text message. Users who forget their password will be able to retrieve their account via a confirmation email sent to the address on file.  Upon a successful login, the user will be shown the map screen. This is the primary screen of the app, showing the user their location, the location of other users and the location of geotagged content. The bulk of the content for the service is short videos, which are tagged with one or more of the following categories: Food, Event, or Location. Food videos appear as green pins on the map view, alongside yellow location pins and red event pins. Should the user wish to know more about a certain location, they can tap any user nearby and make a request of them at the cost of one point.    Points are shown on the profile page, which users can easily access by taping their profile picture in the lower left side of the map screen. Here the user history, ranging from videos to comments, is shown. Users all have a unique user name as well as the option to show their real name to their friends or the public. They also can write a short statement about themselves. The points a user has accumulated through making posts and contributing valuable content is shown here. They can also see the other users that they are following and that follow them. Other user information can be edited using the menu button in the top left. Individual posts can have their properties edited by tapping the “...” button on every post.  Tapping a video in the map, profile, or the grid view will take the user to video view mode. Here, the video content can be seen alongside its location, its category, the poster’s username, the number of views, its like/dislike ratio, and any comments made by other users. Here, a user can provide a rating by tapping the like or dislike button. This cannot be done for the user's own videos. The video player changes color based on the category of the video.    While in the map view, the user can select the grid button in the top left to enter grid view. Grid view shows videos in a grid with thumbnail previews with user information visible on top of the thumbnail. This is also the view users get upon searching for a topic using the search bar on the map screen. Users can sort the videos based on the top videos of the day, most recent videos, or videos near them.  Recording video is an important aspect of the service and the main form of content creation. As such, it has been made as simple as possible. The recording screen only shows what will be recorded, a bar indicating the time elapsed while recording, and a big red record button. To add a video overlay, a user must swipe on the recording window left or right to scroll through their purchased and free filters.  Filters are purchased in the filter marketplace that is accessed via the green shop button on the map screen. Here users can see previews of what the filter will look like on a video. They can freely play with them here, but in order to record video with them they must pay for the filter. There will also be free filters for users to get here.  1.4 User Characteristics |
| The user is anyone who wants to post or find out what is going on around them.  The user could be in any line of work considering this is not only for business application for promoting but mostly focusing on social media.  The same users of apps such as Waze, Uber and Facebook are in line for application of this product.  The design is influenced by the user due to a wide range of users and app understanding.  This will lead to a goal of extremely simplified design with ability to richen the experience with extra features if the user so chooses. 1.5 Constraints, assumptions and dependencies |
|  |
| What could limit the successful completion of the project include the lack of experience developing an app with a given deadline.  The client has expressed several constraints for the Skoovy project:  The Android version of the Skoovy app is to be written in JAVA.  The back-end system of the Skoovy app is to be developed with LAMP technology over cloud servers.  The ASU professor has designated a 4 member student team to complete the Skoovy project.  UI design work will be provided by employees of Hot Salsa Interactive.  Hot Salsa Interactive will incur any necessary server and hosting expenses. As dictated by the course instructor, on-campus teams and online teams are to work independent from each other.  Communication amongst the project members is crucial for team success. To facilitate project development, the team must meet weekly to discuss project goals.  The 2 main constraints are:  * Limited Time  Limited Personnel 2 Specific Requirements |
| 2.1 Use Case Specification |
| 2.2 Information Model |
| Entity Diagram for Skoovy Data.png 2.3 Performance requirements |
| System shall be available from all over the world at all times. Being a social network, any interruption in the sharing chain will cause people to give up on Skoovy; therefore it is essential that the system shall be available at all times.  System shall not be affected from the number of active users in the system until half of the registered users become active.  Being a worldwide network, assuming that half of the registered users are reaching to the website is a legitimate and necessary requirement.  The system should be capable of actively monitoring and analyzing the hashtags of at least 1,000 Skoovy users for location data. The application should load in less than 10 seconds with 99.9% uptime. The map image files shall be compressed to reduce the amount of bytes to under a megabyte each. The map image files shall load from the application itself in under 10 seconds. 2.4 Reliability |
| * Allow users to create video (30sec. maximum) or photo – MTBF – 300 Days * Allow users to add FX from market – MTBF – 180 Days * Allow users to posts – MTBF – 400 Days   2.5 Maintainability |
|  |

# 1 Introduction

This is the software requirements specification document for the Skoovy project sponsored by Rick Brown, from Hot Salsa Interactive.

This project is being undertaken by the SER 401 - Team 4.  The team is comprised of undergraduate students majoring in software engineering at Arizona State University.  The team members are enrolled in a two-semester senior project course required of all undergraduate majors.  Successful delivery of the desired software product will fulfill the senior project requirement for the student team members.

### Project Sponsor:

Name: Rick Brown

Title: CEO

Company: Hot Salsa Interactive

Contact Information: [rbrown@hotsalsainteractive.com](mailto:rbrown@hotsalsainteractive.com)

### DEVELOPMENT TEAM**:**

|  |  |  |
| --- | --- | --- |
| **Team Member** | **Email** | **Phone** |
| Tyler Cole | tjcole2@asu.edu | 928-897-6711 |
| Anthony Kowal | anthony.kowal@asu.edu | 480-258-2794 |
| Lilith Matthews | ljmatth1@asu.edu | 435-650-5999 |
| Rudi Wever | rwever@asu.edu | 650-963-6749 |

The purpose of this document is to collect and analyze all assorted ideas that have been generated in defining the system, its requirements with respect to users.  This document provides a detailed overview of our software product, its parameters and goals.  This document describes the project’s target audience and its user interface and software requirements.  It describes how our client, our project team, and audience view the product and its functionality.  In this document we will try to outline concepts that may be developed later, and document ideas that are being considered, but may be discarded later as the product develops.

### **Definitions:**

***Baseline***. A baseline is a work product that has been formally reviewed and accepted by the involved parties. A baseline is changed only through formal configuration management procedures.

***Software Requirements Specification***. Documentation of the essential requirements (functions/features/uses, performance, design constraints, and attributes) of the software and its external interfaces.

***Protocol***. A set of conventions that govern the interaction of processes, devices, and other components within a system.

***Task***. The smallest unit of work subject to management accountability.

### **Acronyms:**

**ASU** is Arizona State University.

**FK** is Foreign Key.

**FX** is Filter Effects.

**GPS** is Global Positioning System.

**ID** is Identification.

**jpg** is a photo file format developed by Joint Photographic Experts Group.

**LAMP** is an archetypal model of web service [solution stacks](https://en.wikipedia.org/wiki/Solution_stack), named as an [acronym](https://en.wikipedia.org/wiki/Acronym) of the names of its original four [open-source](https://en.wikipedia.org/wiki/Open-source) components: the [Linux](https://en.wikipedia.org/wiki/Linux) [operating system](https://en.wikipedia.org/wiki/Operating_system), the [Apache HTTP Server](https://en.wikipedia.org/wiki/Apache_HTTP_Server), the [**MySQL**](https://en.wikipedia.org/wiki/MySQL) [relational database management system](https://en.wikipedia.org/wiki/Relational_database_management_system).

**mp4** is a video file format developed Moving Pictures Experts Group.

**MTBF** is Mean Time Between Failures.

**PIN** is Personal Identification Number.

**PK** is Primary Key.

**SER** is Software Engineering.

**Skoovers** is the plural of a Skoovy user.

**SMS** is Short Message Service, also known as a text message.

**UI** is User Interface.

### **Abbreviations:**

None listed yet.

### **References:**

***Document Outline:*** <https://myasucourses.asu.edu/bbcswebdav/pid-14434850-dt-content-rid-85270256_1/courses/2016Fall-X-SER401-89620/SRS.pdf>

### **Problem** **Statement:**

The project needs to build an Android front-end system and a back-end server system to provide a social network where users can share their experience(s) of food, places, and events to accumulate points.

### **Vision:**

Want to know what is really going on around you?  Have the itch to share with everyone that tasty new frappe at the local coffee shop?  This is done every day with other social media’s such at Snapchat, Facebook and Twitter, but they lack the ease of seeing this all with a map.  This is Skoovy, a smart app that lets you know what is happening everywhere at any time, with short time-stamped videos/photos linked to that geolocation!

## 

### **Scope:**

***Items that haven’t been specified yet are shown as questions in red font.***

The objectives of the Skoovy project are as follows:

* Allow users to create video (30sec. maximum) or photo
* Allow users to add FX from market
  + Free FX
  + Buy FX
  + Sell FX
* Allow users to make posts Private or Public
* Add Geolocation information to posts (are they really added, or are they included when content is generated?)
* Allow users to add Hashtags to posts
* Allow users the ability to search by:
  + Location
  + Hashtag
  + Popular
  + Notifications
* Allow users the ability to turn search on/off
* Allow users the ability to get notified on geolocation and hashtags
* Profiles will be able to follow users, request location, by hashtag
* Users will earn points for responding to requests.
* Users will lose points for making a request.
* Posting are categorized by type:
  + Food
  + Events
  + Place

## 1.1 Product perspective

## Skoovy fits in line with the sponsor’s suite of products and tools as a unique offering.  The apps that have been previously developed were puzzle, trivia, 2D games or video effects overlay filters.  The element of this app that most relates to previous products is the video interaction, and the FX filters that may be applied when received from a marketplace.  The software is independent and totally self-contained for any interplay of the elements developed for this project.

## 1.2 Use case models of the system’s features

Use Case 1: User creates a video

Actors: User, Skoovy

Features: Camera access, network access, GPS

Use Case 2: User adds FX from Market

Actors: User, Skoovy

Features: Camera access, network access, GPS, Market

Use Case 3: Make posts private

Actors: User, Skoovy

Features: Camera access, network access, GPS

Use Case 4: Use Geolocation to Post

Actors: User, Skoovy

Features: Camera access, network access, GPS

Use Case 5: Add hashtags to a post

Actors: User, Skoovy

Features: Camera access, network access, GPS, word suggestions, tagging

Use Case 6: Search

Actors: User, Skoovy

Features: Camera access, network access, GPS, Search

Use Case 7: Notifications

Actors: User, Skoovy

Features: Camera access, network access, GPS, notifications, push notifications

Use Case 8: Following

Actors: User, Skoovy

Features: Camera access, network access, GPS, search, user profiles, privacy settings

Use Case 9: Requests

Actors: User, Skoovy

Features: Camera access, network access, GPS, search, user profiles, privacy settings, points

Use Case 10: Respond to requests

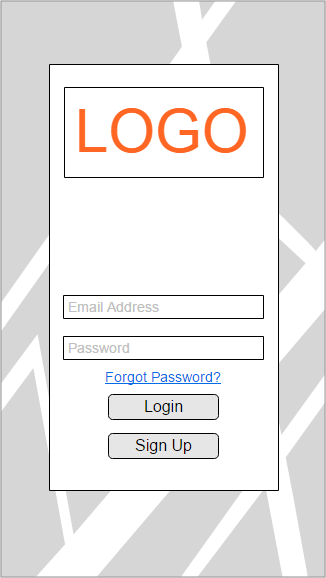
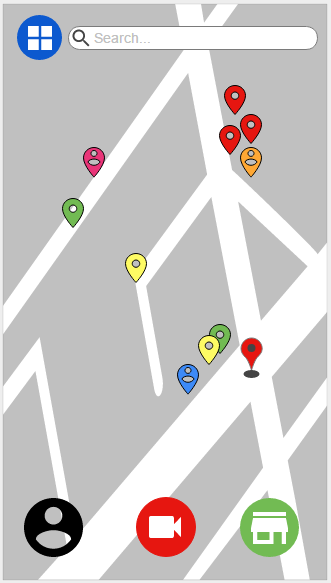
Actors: User, Skoovy

## Features: Camera access, network access, GPS, search, user profiles, privacy settings, points

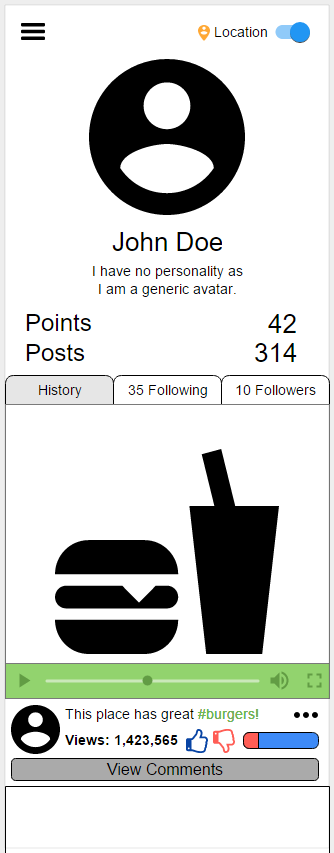
## 1.3 Interaction design specification

The design of the app is intended to give the user easy access to five core features.

First, before the user gains access to the apps features, they must login. New and Existing user will be shown this screen upon the app’s first launch post installation. Existing users need only enter their credentials to gain access to the remainder of the app. New users will have the option to create an account from within the app. The user’s phone number will be required for sign-up with it being verified via a code sent via SMS text message. Users who forget their password will be able to retrieve their account via a confirmation email sent to the address on file.



Upon a successful login, the user will be shown the map screen. This is the primary screen of the app, showing the user their location, the location of other users and the location of geotagged content. The bulk of the content for the service is short videos, which are tagged with one or more of the following categories: Food, Event, or Location. Food videos appear as green pins on the map view, alongside yellow location pins and red event pins. Should the user wish to know more about a certain location, they can tap any user nearby and make a request of them at the cost of one point.

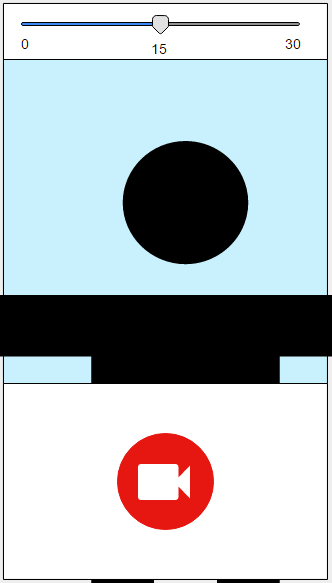
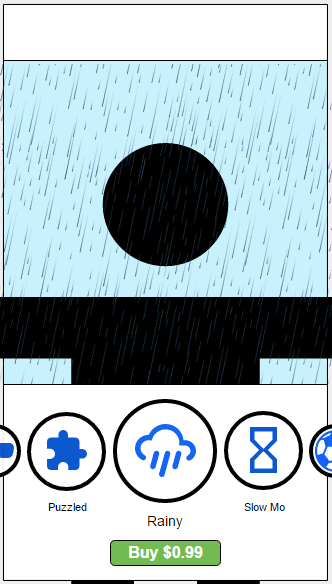
Points are shown on the profile page, which users can easily access by taping their profile picture in the lower left side of the map screen. Here the user history, ranging from videos to comments, is shown. Users all have a unique user name as well as the option to show their real name to their friends or the public. They also can write a short statement about themselves. The points a user has accumulated through making posts and contributing valuable content is shown here. They can also see the other users that they are following and that follow them. Other user information can be edited using the menu button in the top left. Individual posts can have their properties edited by tapping the “...” button on every post.



Tapping a video in the map, profile, or the grid view will take the user to video view mode. Here, the video content can be seen alongside its location, its category, the poster’s username, the number of views, its like/dislike ratio, and any comments made by other users. Here, a user can provide a rating by tapping the like or dislike button. This cannot be done for the user's own videos. The video player changes color based on the category of the video.



While in the map view, the user can select the grid button in the top left to enter grid view. Grid view shows videos in a grid with thumbnail previews with user information visible on top of the thumbnail. This is also the view users get upon searching for a topic using the search bar on the map screen. Users can sort the videos based on the top videos of the day, most recent videos, or videos near them.

Recording video is an important aspect of the service and the main form of content creation. As such, it has been made as simple as possible. The recording screen only shows what will be recorded, a bar indicating the time elapsed while recording, and a big red record button. To add a video overlay, a user must swipe on the recording window left or right to scroll through their purchased and free filters.

Filters are purchased in the filter marketplace that is accessed via the green shop button on the map screen. Here users can see previews of what the filter will look like on a video. They can freely play with them here, but in order to record video with them they must pay for the filter. There will also be free filters for users to get here.

## 1.4 User Characteristics

## The user is anyone who wants to post or find out what is going on around them.  The user could be in any line of work considering this is not only for business application for promoting but mostly focusing on social media.  The same users of apps such as Waze, Uber and Facebook are in line for application of this product.  The design is influenced by the user due to a wide range of users and app understanding.  This will lead to a goal of extremely simplified design with ability to richen the experience with extra features if the user so chooses.

## 1.5 Constraints, assumptions and dependencies

What could limit the successful completion of the project include the lack of experience developing an app with a given deadline.

The client has expressed several constraints for the Skoovy project:

The Android version of the Skoovy app is to be written in JAVA.

The back-end system of the Skoovy app is to be developed with LAMP technology over cloud servers.

The ASU professor has designated a 4 member student team to complete the Skoovy project.

UI design work will be provided by employees of Hot Salsa Interactive.

Hot Salsa Interactive will incur any necessary server and hosting expenses.

# As dictated by the course instructor, on-campus teams and online teams are to work independent from each other.  Communication amongst the project members is crucial for team success. To facilitate project development, the team must meet weekly to discuss project goals.  The 2 main constraints are:

* Limited Time

# Limited Personnel

# 2 Specific Requirements

**Specific Requirements:**

***Items that haven’t been specified yet are shown as questions in red font.***

1.0 User Account:

1.1 Verification:

1.1.1 A new user can register with the Skoovy system by providing:

1.1.1.1 An email

1.1.1.2 A unique Skoovy username

[Once a 4-digit PIN is generated where does it go? Phone/email/app on phone?]

1.1.2 The Skoovy system will send a 4-digit PIN code to the supplied email address.

1.1.3 A new user can be verified using their Skoovy username and 4-digit PIN code.

[Is there a user login? Logout?]

1.1.4 Once a new user has been validated, a profile will be created.

1.2 Profile:

1.2.1 Users have a profile.

1.2.2 Upon creation of a profile, a user’s point balance is initialized to zero.

1.2.3 Only content marked public is visible on a user’s profile.

[Can a user see his/her private content?]

1.2.4 A content owner can view references/descriptions to his public content from his/her profile.

[What content is display on the user’s profile?]

[Define history]

[Define what is included in content when displayed on user profile]

1.2.5 A content owner can delete his/her public content from their profile.

1.2.6 A content owner must confirm the deletion of the content from their profile.

[Can a content owner delete private content?]

1.2.7 A user can accumulate a point by:

1.2.7.1 Posting content that answers a request for content

1.2.7.2 When his/her published content receives a comment and/or like

1.2.8 A user has a point deducted from their point balance when they make a request for content.

1.2.9 An active user has the ability to turn his/her geolocation on/off.

1.2.10 If a user’s location is turned off, this user will not be indicated on any maps for any users.

1.2.11 Any user can view all users that are following him/her.

[What does following someone generate?]

1.2.12 A user can remove themselves from following another user.

1.2.13 A user can view all the users they are following.

1.2.14 A user can view all comments/likes for content they own.

1.2.15 The content owner can reply to the comments individually or collectively for a single content.

2.0 Skoovy User:

2.1 A user can create content.

2.2 Content can be a movie.

2.3 Movie must be in mp4 format.

2.4 Movie must include geolocation information.

2.5 Content can be a picture.

2.6 Picture must be in jpg format.

2.7 Picture must include geolocation information.

2.8 Users can select to add a filter to their content before posting.

2.9 Users must add a description to content before posting.

2.10 Users can edit the description of any content they own.

2.11 Minimum length of content description is 1 non-space character.

2.12 Users can include zero or more hashtags with a content’s description by preceding the selected word(s) or unspaced phrase with the “#” character (also known as [number sign](https://en.wikipedia.org/wiki/Number_sign) or [octothorpe](https://en.wikipedia.org/wiki/Octothorpe)).

[How do hashtags get displayed from a search?]

[How does a user make a request? And what is the process? Currently not handled by information model because no description was supplied.]

2.13 Users can make a request from another user.

[Does the user making a request have to indicate a category and comment?]

[Are replies to requests always public or can they be designated public vs. private?]

[Does a reply category type have to match the request category type?]

2.14 Users can create a user group with a description.

2.15 Maximum number of user groups is 25 per user.

2.16 The owner of a group can add a new member to a specific group.

2.17 The owner of a group can delete a member of a specific group.

2.18 The owner of a group can edit the specific group’s description.

2.19 The owner of a group can delete a specific group.

3.0 Media/Content:

3.1 Content must have 1 or more categories associate with it.

[How to display on pin color if content has more than 1 category associated? Multiple pins? Priority order of pins?]

3.2 The Skoovy system generates a unique ID for the filename for content.

3.3 The unique ID will be made up of the content owner’s user ID + timestamp at posting.

3.4 Content timestamp is the time of posting, not the time when the content was generated.

3.5 Any user can add comment(s) to visible content.

3.6 Content owner must select content to be public or private.

3.7 Content owner can add groups to privately marked content.

[Again, how does content owner select his/her private content (if not displayed on profile) to share with a new group?]

3.8 Content owner can delete groups from privately marked content.

3.9 A user can request content based on:

3.9.1 Geolocation

3.9.2 Category

3.9.3 Hashtag

3.10 The results of a user’s search include content that is marked as public and content that they are a recipient of.

3.11 A user can post comments to the displayed content.

3.12 The content owner can add zero or more replies to a content’s comment.

3.13 The content owner can delete a content.

3.14 Deleting content will delete comments and replies associated with the content.

[Does a user still retain points for deleted content?]

[How to add a comment to private content if it’s no longer visible after viewing?]

## 2.1 Use Case Specification

|  |  |
| --- | --- |
| **Use Case ID** | **Use Case Description** |
| **UC01** | **Use Case 1: User Creates Video** |
| **UC02** | **Use Case 2: User Adds FX from Market** |
| **UC03** | **Use Case 3: Make Posts Private** |
| **UC04** | **Use Case 4: Use Geolocation to Post** |
| **UC05** | **Use Case 5: Add Hashtags to a Post** |
| **UC06** | **Use Case 6: Search** |
| **UC07** | **Use Case 7: Notifications** |
| **UC08** | **Use Case 8: Following** |
| **UC09** | **Use Case 9: Requests** |
| **UC10** | **Use Case 10: Respond to requests** |

### **Use Case 1: User Creates Content**

|  |  |
| --- | --- |
| **Objective:** | The user creates Content |
| **Primary Actor:** | User |
| **Dependencies** | Skoovy user must have already completed registeration |
| **Trigger:** | User selects option to post a content from app main menu |
| **Secondary Actors:** | Skoovy |
| **Preconditions:** | 1. Skoovy app must have started successfully 2. User must have completed registeration 3. Device must have a camera |
| **Post Condition(s):** | 1. User posts content to Skoovy |

**Main Success Scenario**

1. Skoovy opens user’s camera
2. User films a video
3. Skoovy asks to post or add FX
4. User selects to post content
5. Skoovy asks how to post the content respective to privacy settings
6. User selects privacy setting of public or private
7. Skoovy asks to confirm the posting with selected privacy setting
8. User selects to confirm the post
9. Skoovy posts the content
10. Use case ends

**Variations**

Variation ID: *1.0 User Posts Photo*

If, in step 3, the user takes a photo,

* 1. Use case continues at step 3

Variation ID: *2.0 User adds free FX from Market*

If, in step 5, User selects “Add FX” option after generating content

2.1 Skoovy ask what FX to add

2.2 User selects the FX

2.3 Use case continues at step 3

Variation ID: *3.0 User Buys FX*

If, in step 2.2, user selects FX that is not free

3.1    Market displays FX and prices

3.2   User selects FX to purchase

3.3 Android/Apple asks user to enter payment info

3.4   User buys FX

3.5. Use case continues at step 3

**Failure Variations**

Variation ID: 4.0 *Skoovy app detects no data connection*

If, in step 10, Skoovy app has no data connection

4.1 Skoovy displays ‘no connection’ message to user for 3 seconds.

4.2 Use case ends

**Business Rules**

Users must have a registered Skoovy account

**Use Case 2: Change a content’s public/private setting**

|  |  |
| --- | --- |
| **Objective:** | The user changes posted content’s public/private setting |
| **Primary Actor:** | User |
| **Dependencies:** | UC 01 : User Creates Content |
| **Trigger:** | User selects post from user prolife |
| **Secondary Actors:** | Skoovy |
| **Preconditions:** | 1. Skoovy app must have started successfully 2. User must have completed registration 3. User must have already posted content |
| **Post Condition(s):** | 1. Posted content is set to user choice of public or private settings |

**Main Success Scenario**

1. Skoovy displays selected content and prompts user to toggle public/private button
2. User selects to toggle button to private

3.1 Skoovy saves the content’s settings.

3.2 Skoovy displays a “content is now private’ message for 3 seconds

3.3 Skoovy updates the system to reflect settings change.

3.4 Skoovy display is returned to user profile.

4. Use case ends

**Variations**

Variation ID: *1.0 User Changes content from private to public*

If, in step 2, user selects to toggle button to public

1.1 Skoovy saves the content’s settings.

1.2 Skoovy displays a ‘content is now public’ message for 3 seconds.

1.3 Use case resumes at step 5.

**Failure Variations**

Variation ID: 2.0 *Skoovy app detects no data connection*

If, in step 3.1, Skoovy app has no data connection

2.1 Skoovy displays ‘no connection’ message to user for 3 seconds.

2.2 Use case ends

**Business Rules**

Users must have a registered Skoovy account

### **Use Case 3: Edit Post**

|  |  |
| --- | --- |
| **Objective:** | The user modifies the descriptive data of their post |
| **Primary Actor:** | User |
| **Dependencies:** | UC 01 – User Creates Content |
| **Trigger:** | User selects “Edit” on one of their posts |
| **Secondary Actors:** | Skoovy |
| **Preconditions:** | 1. Skoovy app must have started successfully 2. User must have completed registration 3. Device has an internet connection that the app is permitted to use. |
| **Post Condition(s):** | 1. The descriptive data of the User’s post has been updated. |

**Main Success Scenario**

1. Skoovy displays the “Edit Post” screen.
2. User modifies the data within the Description text box and/or the hashtag list. When finished, User hits “Submit Changes”

3.1 Skoovy replaces the existing post data with the new post data and makes the changes visible to the public immediately.

3.2 Skoovy returns the user to their post.

4. Use case ends

**Variations**

Variation ID: *1.0 Delete Post*

If, during step 2, the User selects the “Delete Post” button.

1.1 Skoovy asks if the user is sure they wish to delete the post, suggesting to instead make the post private.

1.2 User confirms their desire to delete the post.

1.3 Skoovy deletes the posts data from the server and removes it from public view.

1.4 Use case resumes at step 3.2

**Failure Variations**

Variation ID:2.0 *Unable to make changes*

If, during step 3.1, for any reason, Skoovy is unable to make the changes to the post.

2.1 Skoovy informs the user that an error has occurred during posting and that no changes have been made to their post. Skoovy also informs the user that their changes will be saved as a draft for them to attempt the changes later. Skoovy saves the changes in a separate file.

2.2 User selects “OK” on the error window.

2.3 Skoovy returns User to main screen.

2.4 Use Case Ends

**Business Rules**

Users must have a registered Skoovy account

### **Use Case 4: Search**

|  |  |
| --- | --- |
| **Objective:** | The user searches for posts |
| **Primary Actor:** | User |
| **Dependencies:** |  |
| **Trigger:** | User selects the search icon |
| **Secondary Actors:** | Skoovy |
| **Preconditions:** | 1. Skoovy app must have started successfully 2. User must have completed registration 3. Device has an internet connection that the app is permitted to use. |
| **Post Condition(s):** | 1. Search content results have been returned to the user |

**Main Success Scenario**

1. Skoovy displays options of how to search
2. User selects “Search by location”
3. Skoovy requests input
4. User inputs a location they would like to search
5. Skoovy displays results
6. Use case ends

**Variations**

Variation ID: *1.0 Search by Hashtag*

If, in step 2, the user selects to search by “Hashtag”

1.1    Skoovy requests input for hashtag

1.2    User inputs a hashtag they would like to search

1.3    Use case resumes at step 5

Variation ID: *2.0 Search by Popular*

If, in step 2, the user selects to search by “Popular”

2.1    Skoovy indicates the video with most hits and asks if the user would like to filter further

2.2    Use case ends

**Failure Variations**

Variation ID: 3.0 *Skoovy app detects no data connection*

If, in step 5, Skoovy app has no data connection

3.1 Skoovy displays ‘no connection’ message to user for 3 seconds.

3.2 Use case ends

**Business Rules**

Users must have a registered Skoovy account

### **Use Case 5: Notifications**

|  |  |
| --- | --- |
| **Objective:** | The user wants to change notifications setting |
| **Primary Actor:** | User |
| **Dependencies:** |  |
| **Trigger:** | User selects ‘notifications’ from the app’s settings menu |
| **Secondary Actors:** | Skoovy |
| **Preconditions:** | 1. Skoovy app must have started successfully 2. User must have completed registration 3. Device has an internet connection that the app is permitted to use. |
| **Post Condition(s):** | 1. User profile is updated for notifications |

**Main Success Scenario**

1. Skoovy displays Notifications on/off for locations
2. Skoovy displays Notifications on/off for hashtags
3. User toggles location button to ‘on’ state
4. Skoovy prompts user to select a location.
5. User enters location(s)
6. User toggles hashtag button to ‘on’ state
7. Skoovy prompts user to enter a hashtag.
8. User enters hashtag(s)
9. Skoovy prompts user to save changes
10. User confirms changes
11. Skoovy displays settings menu
12. Use case ends

**Variations**

Variation ID: *1.0 User toggles location notifications button to ‘off’ state*

If, in step 3, user *toggles location notifications button to ‘off’ state*

1.1    Use case resumes at step 10

Variation ID: *2.0 User toggles hashtag notifications button to ‘off’ state*

If, in step 6, user *toggles hashtag notifications button to ‘off’ state*

2.1    Use case resumes at step 10

**Failure Variations**

Variation ID: 3.0 *Skoovy app detects no data connection*

If, in step 10, After the user confirms changes, the Skoovy app has no data connection

3.1 Skoovy displays ‘no connection’ message to user for 3 seconds.

3.2 Use case ends

**Business Rules**

Users must have a registered Skoovy account

### **Use Case 6: Following**

|  |  |
| --- | --- |
| **Objective:** | The user follows another user |
| **Primary Actor:** | User |
| **Dependencies:** | Having more than 1 Skoovy user in the system |
| **Trigger:** | User selects to view a content item of another user. |
| **Secondary Actors:** | Skoovy |
| **Preconditions:** | 1. Skoovy app must have started successfully 2. User must have selected content from another user 3. Device has an internet connection that the app is permitted to use. |
| **Post Condition(s):** | 1. User receives notification 2. Users profiles are updated 3. Skoovy display is returned to display of content. |

**Main Success Scenario**

1. After the user selects to view content from another Skoovy user, the user is presented with a toggle button that will allow them to ‘follow’ that other user’s postings.
2. User toggles button to ‘follow’ state.

3.1 Skoovy saves the request to ‘follow’ another user.

3.2 Skoovy displays a ‘following’ message to the user (primary actor) for 3 seconds.

3.3 Skoovy updates both user’s profiles to indicate ‘following’ and ‘followed by’ for the respective users.

3.4 Skoovy display is returned to display of content.

4. Use case ends.

**Variations**

Variation ID: *1.0 User is selects to ‘un-follow’ the selected user.*

*If in step 2 the user toggles the button to the ‘un-follow’ state,*

1.1 Skoovy saves the request to ‘un-follow’ another user.

1.1.1 Skoovy displays a ‘un-following’ message to the user (primary actor) for 3 seconds.

1.1.2Skoovy updates both user’s profiles to remove ‘following’ and ‘followed by’ for the respective users.

2. Use case resumes at step 3.4

**Failure Variations**

Failure Variation ID: *2.0 Save data failure*

If in step 3.1 the Skoovy system fails to properly save data generated from the request to follow another user,

2.1 Skoovy displays ‘request failed’ message to the user (primary actor) for 3 seconds.

2.1.1Skoovy display is returned to display of content.

2.2 Use case ends.

Failure Variation ID: *3.0 No data connection at request time*

If in step 3.1 the Skoovy system detects that the user has no data connection,

3.1 Skoovy displays ‘request failed’ message to the user (primary actor) for 3 seconds.

3.1.1Skoovy display is returned to display of content.

3.2 Use case ends.

**Business Rules**

Skoovy must have at a minimum, 2 registered users in the system in order for one user to follow another user.

Users must have a registered Skoovy account

### **Use Case 9: Requests**

|  |  |
| --- | --- |
| **Objective:** | The user requests content |
| **Primary Actor:** | User |
| **Dependencies:** | User’s point balance must be greater than 0. |
| **Trigger:** | Request Content is selected from the Main menu |
| **Secondary Actors:** | Skoovy |
| **Preconditions:** | 1. Skoovy app must have started successfully 2. User must have completed registration 3. Device has an internet connection that the app is permitted to use. |
| **Post Condition(s):** | 1. User receives notification |

**Main Success Scenario**

1. Skoovy displays options of how to request
2. User selects what they would like to request
3. Skoovy requests input
4. User inputs a location/content they want
5. Skoovy sends request to nearby Skoovers with their receive requests on, subtracts points from requesting user
6. User selects the option to receive notifications when their request is fulfilled
7. Skoovy will send push notifications to the users mobile device upon activity for the request
8. Use case ends

**Variations**

Variation ID: *1.0 Notification for hashtags*

If, in step 2, user selects “Hashtag”

2.1    Skoovy requests input for hashtag

2.2    User inputs a hashtag

2.3    Skoovy displays results

2.4    Use case ends

**Failure Variations**

Failure Variation ID: *2.0 No data connection at request time*

If in step 3.1 the Skoovy system detects that the user has no data connection,

2.1 Skoovy displays ‘request failed’ message to the user (primary actor) for 3 seconds.

2.1.1Skoovy display is returned to display of content.

2.2 Use case ends.

**Business Rules**

User Users must have a registered Skoovy account

### **Use Case 10: Respond to requests**

|  |  |
| --- | --- |
| **Objective:** | The user responds to requests |
| **Primary Actor:** | User |
| **Dependencies:** | UC 01 User Creates Content |
| **Trigger:** | User selects the respond to request from Main menu |
| **Secondary Actors:** | Skoovy |
| **Preconditions:** | 1. Skoovy app must have started successfully 2. User must have completed registration 3. Device has an internet connection that the app is permitted to use. |
| **Post Condition(s):** | 1. User receives notification |

**Main Success Scenario**

1. User selects the “Respond to Request” item
2. Skoovy loads and displays main menu
3. Skoovy displays new requests received by the user based on geographical location
4. User selects a request
5. Skoovy displays the request
6. User selects respond via video
7. UC 01 User Creates Content is completed successfully
8. Skoovy posts video and awards user points for responding to request
9. Use case ends

**Variations**

Variation ID: *10.0 Denies Request*

If, in step 6, User denies request

1.1    User selects to cancel request for content

1.2    Use case ends

**Failure Variations**

Failure Variation ID: *2.0 No data connection at request time*

If in step 3 the Skoovy system detects that the user has no data connection,

2.1 Skoovy displays ‘request failed’ message to the user (primary actor) for 3 seconds.

2.1.1Skoovy display is returned to display of content.

2.2 Use case ends.

**Business Rules**

User Users must have a registered Skoovy account

## 

## 2.2 Information Model

## Entity Diagram for Skoovy Data.png

## 2.3 Performance requirements

System shall be available from all over the world at all times. Being a social network, any interruption in the sharing chain will cause people to give up on Skoovy; therefore it is essential that the system shall be available at all times.

System shall not be affected from the number of active users in the system until half of the registered users become active.  Being a worldwide network, assuming that half of the registered users are reaching to the website is a legitimate and necessary requirement.  The system should be capable of actively monitoring and analyzing the hashtags of at least 1,000 Skoovy users for location data.

## The application should load in less than 10 seconds with 99.9% uptime. The map image files shall be compressed to reduce the amount of bytes to under a megabyte each. The map image files shall load from the application itself in under 10 seconds.

## 2.4 Reliability

The specific accepted mean times between failures for the features developed:

* Allow users to create video (30sec. maximum) or photo – MTBF – 300 Days
* Allow users to add FX from market – MTBF – 180 Days
* Allow users to posts – MTBF – 400 Days

## 2.5 Maintainability

Design and construction standards in a way to facilitate the work in modifying the software once the product is delivered and the project team is no longer available. In addition, future work on the software should be significantly shorten the time needed to repair defects that occur.