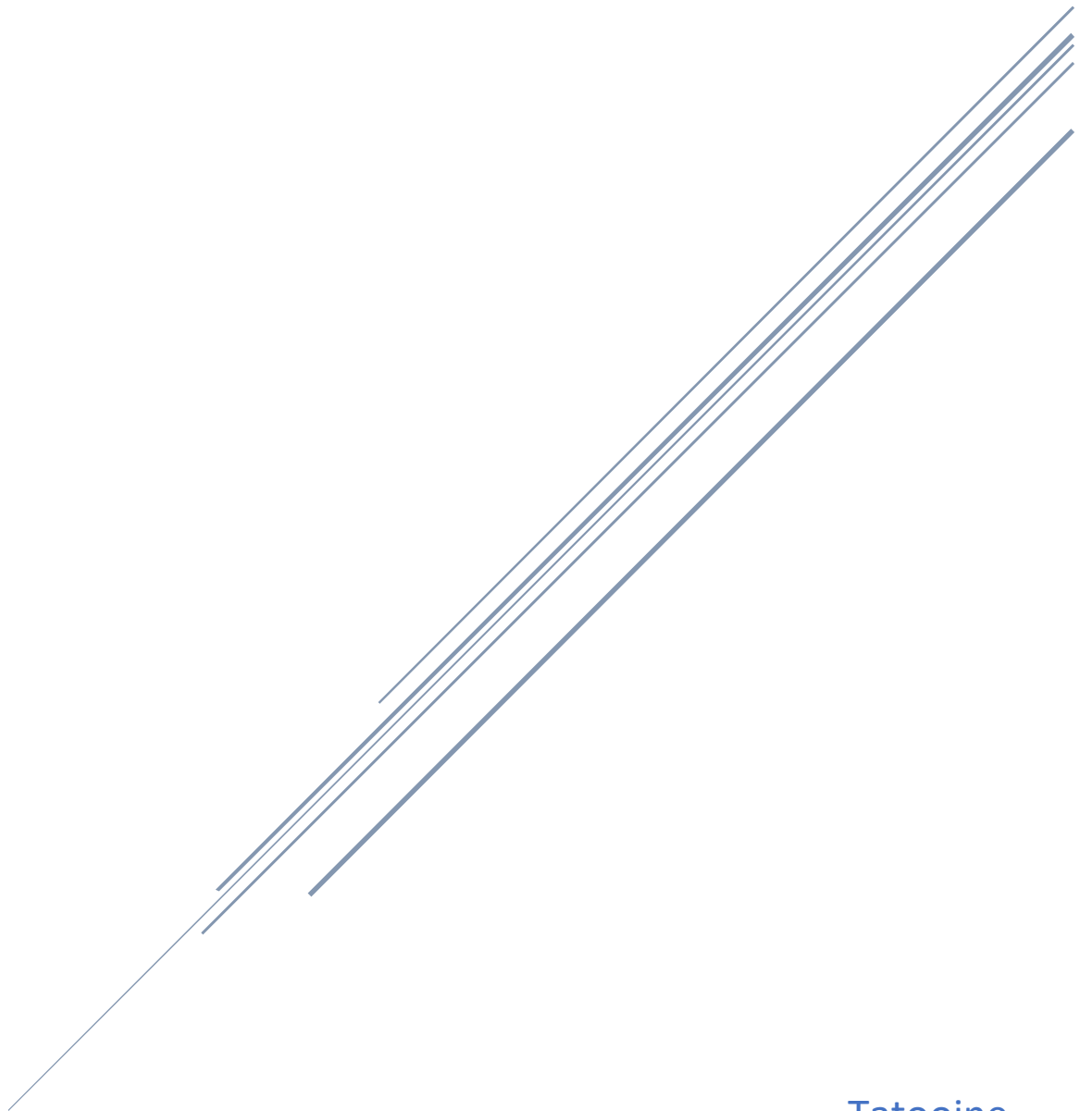


SKATEABLE

Dev Doc 2



Tatooine

Henry Ang, Josh Holt, Matt McCue, Stanley Tran

Table of Contents

Executive Summary	
Section 1: Project Description	1
1.1 System Description and Rationale	1
1.2. System Scope and Context Diagram	1
1.3. System and Development Constraint	2
1.4. Proposal and Request	3
Section 2: System Services	6
Introduction	6
2.1. Functional Requirements	6
2.2. Use-Case Diagram	8
2.3. Use-Case Descriptions	9
Section 3: Nonfunctional Requirements	16
Introduction	16
3.1 Operational Requirements	16
3.2 Performance Requirements	16
3.3 Security Requirements	16
3.4 Cultural and Political Requirements	16
3.5 Process Requirements/Project Constraints	16
3.6 Other Nonfunctional Requirements	17
Section 4: Development and Operating Environments	18
Introduction	18
4.1 System Architecture	18
4.2 Hardware Environment	19
4.3 Software Environment	19
4.4 System Security	19
4.5 Development Environment	19
Section 5: Data (structural) Design	20
Introduction	20
5.1 Class Diagram	20

5.2 Data Dictionary (Metadata)	21
Section 6: Behavioral Design	31
Introduction	31
6.1 System Services Implementation Model	31
6.2 Statechart Diagram	32
Section 7: User Interface Design Plan	33
7.1 User Interface Requirements and Constraints	33
7.2 Window Navigation Diagram	33
7.3 Screen Interface Design	34
Section 8: Development Plan	40

Executive Summary

The concept behind Skateable came to be one afternoon when our client, Chris Beebe, decided he was tired of going to the same skate park every time he wanted to skate. Chris is an avid skater and has been skating for more than 8 years. Once he moved up to Seattle, he needed something new. Something that would remind him once again why he fell in love with skating in the first place: adventure. His frustration was like electricity turning the gears inside his brain. What if there was a way that let skaters share their favorite and most unique skating locations with the rest of the skateboard community, giving people the chance to put adventure back into their skating as they explore new locations around their city with likeminded people.

Using map software (Google Maps), Skateable, a web application lets users create an account and log in to place pins on precise locations on a map to show that this is a great site to skate at. People who post a new site can write descriptions, notes about the location, and potentially upload pictures, while viewers have the ability to rate, comment, share, and arrange meet-ups at the location.

The Skateable application will be tackled by team Tatooine:

Henry Ang (B.S. Computer Science - Minor in Computer Engineering)

- Team Librarian - Henry is in charge of merging and managing files and documents that is created by all team members.
- Contact email: angh@spu.edu

Josh Holt (B.S. Computer Science - Minor in Computer Engineering and Business Admin)

- Team Recorder - Josh is in charge of taking notes when meeting up with our client, during team meetings, meetings with Dr. Arias, and any other information exchange meeting.
- Contact email: holtj3@spu.edu

Matt McCue (B.A. Computer Science - Minor in Psychology)

- Team Recorder/Librarian - Matt assists both Henry and Josh with their tasks and can take over a role in the event that any of the team members is absent/ill.
- Contact email: mhoward13@spu.edu

Stanley Tran (B.A. Computer Science - B.A. in Business Admin - Management)

- Team Leader - Stanley is in charge of setting up meeting times, making sure deliverables are met, and keeping the project moving.
- Contact email: stantran@spu.edu

As whole, the entire team will be taking part in the creation of the application when it comes to development during Spring quarter 2018.

Section 1: Project Description

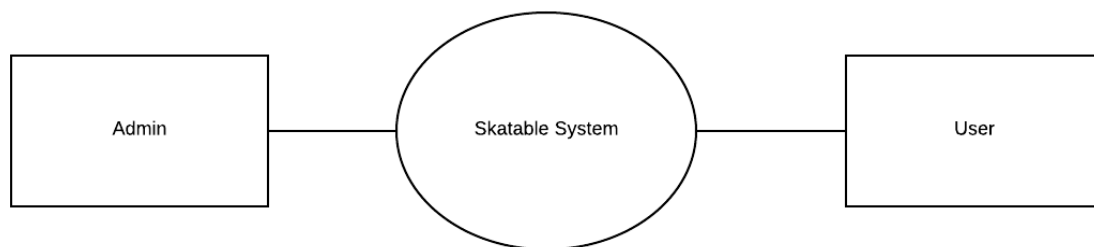
1.1. System Description and Rationale

Skateable is a web application that allows for users to enter a platform to connect with each other over skating locations around them.

The Skateable application will be built as a web application where it can be used across any platform. Chris envisioned that our target market would be skateboarders that live in the West Coast of the United States with ages ranging from 18-35 years old (most common age for skateboarders). The system will allow for the users to create an account, log in and be able to view or create “pins” on a map. If users choose to create a pin, they have options of labeling the pin with a description, adding notes to the location, and uploading pictures of the location. If the user chooses to view a location, they will have the options to rate the location, make comments about the location, and plan meet-ups at the location. The benefit from the app is to connect likeminded users over shared skate locations, to help users meet new people, and explore around the city more.

1.2. System Scope/Context Diagram

A function that our application will have, will be a login system where users can create an account and use the application. Users will be able to add or view pins on the map based on if they want to create a new pin or view a current pin. Another function is that users can create meet up events and have groups created for these types of events.



1.3. System and Development Constraints

The following will contain potential constraints that may come up during the development process and usage of Skateable.

- Skateable will need resources to be built such as implementing a database.
- Skateable will be starting out on the West Coast of the United States during early stages of usage until Skateable grows in user popularity.

1.4 Proposal and Request

CPE/CSC 415n – Team and Project Proposal

PART 1 TEAM

Team Name (temporary for now; will need to be settled by the end of January)

Team “Skateable and Dateable”

Team Members (name and Email address of all team members)

Leader: Stanley Tran - stantran@spu.edu

Recorder: Josh Holt - holtj3@spu.edu

Librarian: Matt Howard - mhoward13@spu.edu,

Henry Ang - angh@spu.edu

PART 2 PROJECT

Working Name of System (this can be changed later)

Skateable

Project Sponsor (name and complete contact information)

Christopher Beebe, Beebec@spu.edu, (971) 269 -8750

Project Scope

Skateable will be a web application that allows skaters to network and explore new places in the user surrounding area. We plan for the application to use a location-based system that allows for the users to mark or share locations through GPS that is a popular hangout and skate spot. Users should be able to select a location on the map and plan out times to meet and host events with other skaters within the area. Users can also write a short description of their marked locations, so that others know what to expect if they choose to visit a marked skate site. We are planning to start out with skateboarding sites but hope to expand to more as well (biking, hiking, rock climbing, etc).

The physical environment that we are going to be focusing on is cities specifically Seattle to start out with. If we continue on with the application, we would aim to expand down the west coast, major U.S. cities, and parts of Europe.

Why you have chosen this project -

Stanley took Entrepreneurship last quarter and as a part of the class, teams come up with a business plan with a proposed “product”. Stanley’s group came up with the Skateable application with his teammate from that class as our sponsor, we believed that Skateable would be a fun and challenging project to work on for Software Engineering.

Team Preparation - a statement of relevant previous coursework or practical experience in your chosen development environment.

Netcentric Computing

Application Design

Have taken/currently enrolled in Android Programming

Database Management

C++, Java

SYSTEM REQUEST – Skateable

1/11/18

Project Sponsor[1]

Name: Chris Beebe

Email: Beebec@spu.edu

Organization / Department:

Phone: (971) 269 - 8750

Opportunity Statement[2]: We aim to develop an application that lets users connect with one another over skateboarding. We hope that this app helps those who are new to cities be able to explore and learn about local skate sites.

Proposed Product

Background and Context[3]:

Christopher Beebe (our sponsor), was getting tired of going to the same old skate spots that he always goes to. As a solution to his problem, he thought of the Skateable application that would help him find new skate spots and meet other skaters close by to him. Christopher and his group (Stanley included) pitched this idea in their entrepreneurship class and this quarter we thought we could tackle the project and bring it to be.

Initial Vision and Scope[4]:

- GPS map that allows for users to mark locations that they deem as a good skate/hangout spot
- Ability to interact with other users on the application (Group chat)
- Notify user of upcoming meetups
- Possibly have advertisement
- Allow user to authenticate/report marked locations (Admin can remove bogus locations)
- Display map within a user defined mile radius
- Have a rating system of hangout spots
- Allow user to create a profile and join groups
- User can login/ create account using google for Facebook accounts

Stakeholders[5]:

- Skaters who use the application (Users)
- Potentially local skate/outdoors shops if we choose to partner with them to create ads
- Development Team (Admin)
- Potential Sponsor/Investors

Expected Benefits:

- Allow for users to network with likeminded people over activities that they share a common interest in
- Allow for users to explore new areas of the city that they may not yet be familiar with
- Advertisement for local Businesses and shops

Other Issues or Constraints to be considered (what else you would like us to know):

- Access to APIs (Google Location APIs)
- Budget - (maintaining database, web services)

Section 2: System Services

Introduction

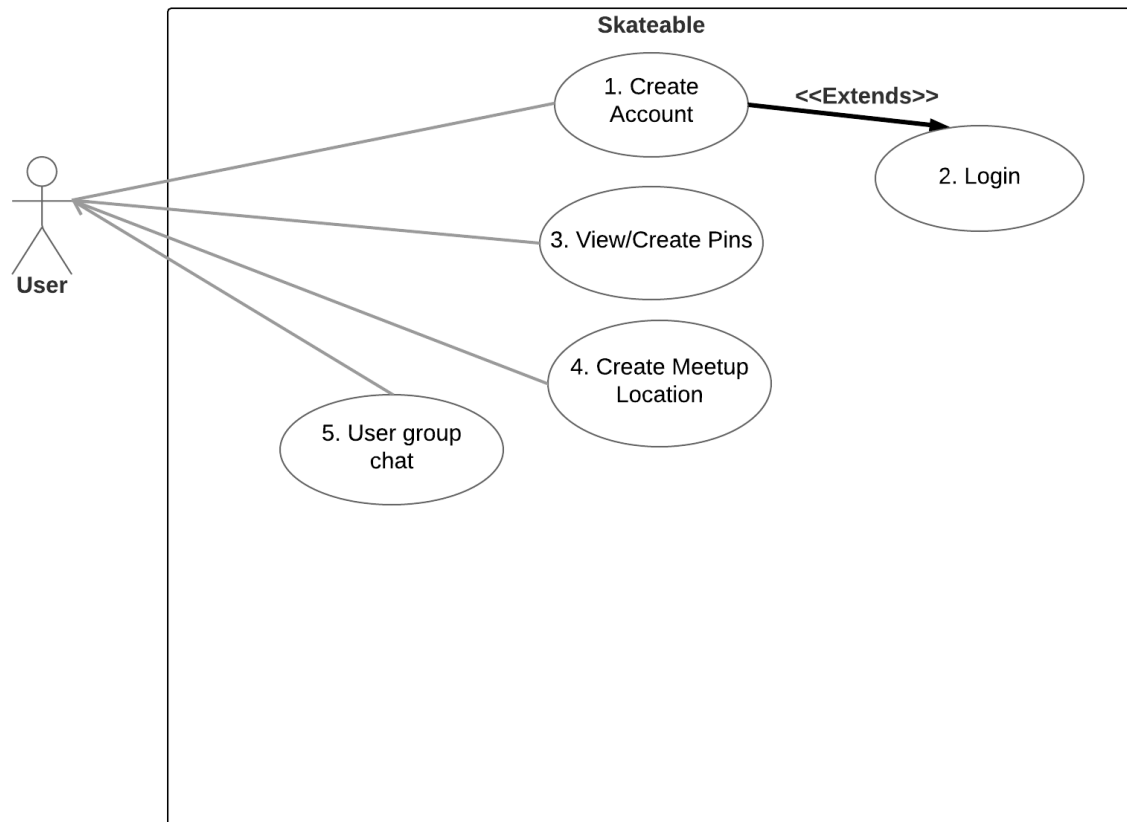
This section will detail the functional requirements and the use cases of the system. Functional requirements describe what services the system will provide a user. For each function requirement, this section will give a brief description, the intended use-cases for this service, and the initial acceptance tests that the service should be able to complete. This section will also provide a diagram of all the use-cases for the system. The use-case diagram will show which use-case is intended for each type of user. This section will also give a description of each use-case.

2.1 Functional Requirements

- **User creates an account**
 - User goes through the account creation process and then logs into the application through their created account information.
 - Use Case(s):
 - 1: Create Account
 - 2: Login
 - Initial Acceptance tests:
 - Make sure username is made with valid character
 - Check for existing username
 - Make sure password is made with valid characters
 - Set password to certain length
 - Weak password detector – should have at least numbers, uppercase, and lowercase letters
 - Make sure no existing username
- **Create a new pin/View existing pin**
 - User can create a pin on the map and with options of adding a description and adding pictures of the location.
 - User can also choose to view an existing location with options of rating the location and proposing a meet up at the location.
 - Use Case(s):
 - 3: View/Create Pins
 - 4: Create Meetup location
 - Initial Acceptance tests:
 - Checks to see if pin already exists
 - If so, only then can description be added
 - If so, then the pin can be viewed
- **Mark a location for a meetup**
 - User selects an existing location and sets the marker to show that this will be a meet up location at a certain day.
 - Use Case(s):
 - 4: Create Meetup location
 - Initial Acceptance tests:

- Marker Changes colors
 - Marker color marks a meet up
- **Give the user the ability to create a group with other users**
 - Users can select other users to create a group and give the group a name. This can act as group chat for planning out meetups.
 - Use Case(s):
 - 5: User group chat
 - Initial Acceptance tests:
 - Must select another user to create a chat
 - Once chat is created, must create a chat name
 - Name must meet word criteria

2.3. Use-Case Diagram



2.4. Use-Case Descriptions

Use-Case name: Create Account	ID: 1	Importance: High
Primary actor: User	Use-Case type: Essential	
Stakeholders and interests: User – wants to create new account to plan out meetup location.		
Brief description: This case describes the process of the user creating a new account for Skateable System.		
Trigger: User clicks “create new account” on web application Type (circle one): External Temporal		
Relationships: Association: User Include: --- Extend: --- Generalization: ---		
Normal flow of events: 1. User clicks “create new account” on web application. 2. User will fill out all required fields 3. Username will be validated with previous usernames. 4. An email will be sent to validate the user email address. 5. User will click link in email to confirm email and complete registration.		
Subflows:		
Alternate / exceptional flows: 3a: If username is taken, the user will be prompted to choose a new username. 4a: If email is not confirmed after 30 days, account information will be deleted.		

Use-Case name: Login	ID: 2	Importance: High
Primary actor: User	Use-Case type: Essential	
Stakeholders and interests: User – wants to login to account to able to create a meetup location.		
Brief description: This use case describes the user logging in with a username and password into the account to use the system.		
Trigger: Enter valid username and password on CBC login in website Type (circle one): External Temporal		
Relationships: Association: User Include: --- Extend: 1 – If there is no account, perform 1 - Create an Account. Generalization: ---		
Normal flow of events: 1. User goes on Skateable web application login screen. 2. User enters username and password. 3. User clicks “login in” button. 4. After username and password validate user has access to system.		
Subflows:		
Alternate / exceptional flows: 2a. If username and password is not correct, notify user of invalid username or password and do not allow access to system.		

Use-Case name: View/Create Pins		ID: 3	Importance: High
Primary actor: User		Use-Case type: Essential	
Stakeholders and interests: User – wants to create or view pins of a location.			
Brief description: This case describes the process of the user viewing or creating a pin on a map of a location with a rating system that shows if a spot is viable or not.			
Trigger: User clicks “create new account” on web application Type (circle one): External Temporal			
Relationships: Association: User Include: --- Extend: --- Generalization: ---			
Normal flow of events: 1. User clicks “Drop pin” button on web application. 2. User will drag pin to desired location. 3. User will fill out pin information 4. User will confirm meetup information is correct and a pin location on a map.			
Subflows:			
Alternate / exceptional flows: 1a: If a pin is already created, user can click on pin and view meetup pin information.			

Use-Case name: Create Meetup location	ID: 4	Importance: High
Primary actor: User	Use-Case type: Essential	
Stakeholders and interests: User – wants to be able to create a meetup location to plan a meetup.		
Brief description: This use case describes the user creating a meetup location pin that allows notifies users in group of meetup location and time.		
Trigger: Enter valid username and password on CBC login in website Type (circle one): External Temporal		
Relationships: Association: User Include: --- Extend: 1 – Generalization: ---		
Normal flow of events: 1. User clicks “Drop pin” button on web application. 2. User will drag pin to desired location. 3. User will fill out meetup location information (Time, date, location name, group members) 4. User will confirm meetup information is correct and a new meet up location pin is created on a map.		
Subflows:		
Alternate / exceptional flows: 2a. If username and password is not correct, notify user of invalid username or password and do not allow access to system. Only users with an account will be able to access Skateable.		

Use-Case name: User group chat	ID: 5	Importance: High
Primary actor: User	Use-Case type: Essential	
Stakeholders and interests: User – wants to be able to talk to other users in a group.		
Brief description: This use case describes user talking to other users in a group chat.		
Trigger: User clicks on chat in a group meetup Type (circle one): External Temporal		
Relationships: Association: User Include: --- Extend: 1 – Generalization: ---		
Normal flow of events: 1. User will create or join a group meet up location. 2. User clicks “Connections” button in tab 3. User will select group to chat with		
Subflows:		
Alternate / exceptional flows: 2a. If username and password is not correct, notify user of invalid username or password and do not allow access to system.		

Section 3: Nonfunctional Requirements

Introduction

The following section will list out the nonfunctional requirements for the Skateable web application. This section is broken up into five parts as follows; Operational Requirements, Performance Requirements, Security Requirements, Cultural and Political Requirements, Process Requirements/Project Constraints, and any other Nonfunctional Requirement. Regarding Skateable, these requirements specify criteria where the system would be judged based upon.

3.1 Operational Requirements

- Must be connected to the internet via WIFI/Data plan to operate Skateable
- Operate on multiple web browsers
 - Google Chrome 49.0+
 - Safari 11.0 +
 - Firefox 59.0 +
 - Internet Explorer 11.0 +
- Skateable should be able to be easily serviced and patched when new a newer version is developed

3.2 Performance Requirements

- Skateable should take no longer than 2 seconds to load the home page
- Once users log into Skateable, it should take no longer than 3 seconds to load the home page with the Skateable map.
- All functions within the application should take no longer than 1 second to execute.
- Skateable's home page with the map should update every 15 seconds to ensure that all information is up to date. (new pins dropped/bad locations removed)

3.3 Security Requirements

- Anyone can create and account to be a user
- Must have an account to use Skateable
- Password will be encrypted

3.4 Cultural and Political Requirements

- Skateable will be in the English language when presented to the client
- Translations will be available upon expanding to new geographic areas

3.5 Process Requirements/Project Constraints

- Simple UI where users can easily navigate and understand where they are at in the system
- Scalable
- Project is currently starting out on the west coast but will scale as popularity and usage grows.
- Any system updates/System modifications must reflect on the previous versions before it. Changes interfaces must be addressed and explained so users can understand how to use updated version (patch notes).

3.6 Other Nonfunctional Requirements

- Complete product to be presented to client by June 1st, 2018 (last day of classes)
- Skateable must pass client's evaluation to make sure that all requirements are satisfied

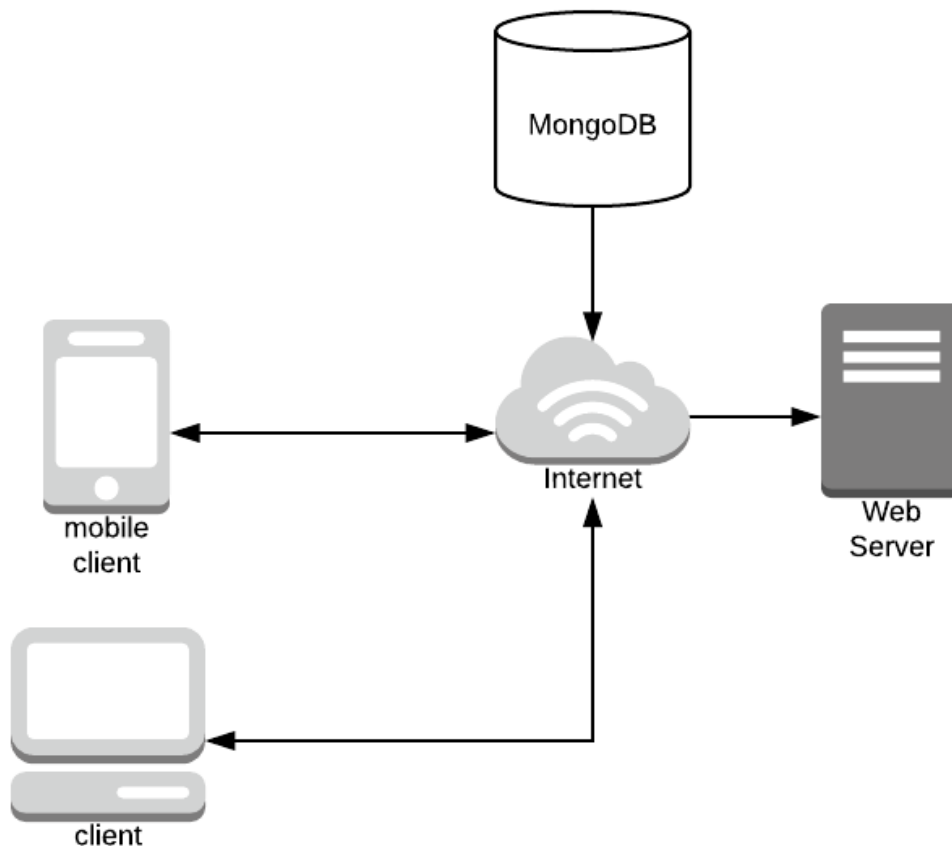
Section 4: Development and Operating Environments

Introduction

This section details the hardware and software environment of the system. The system architecture section describes the setup of the system for client to server. It will also have a system architecture diagram that shows how the system will be set up. The hardware environment section describes the minimum and recommended hardware requirements for the user, and the hardware requirements of the server. The software environment section will detail the operating system of the user devices and the server. It will also describe the system required software. The system security section describes the security implementation of the system. The development environment section details the tools used during the development of the system.

4.1 System Architecture

The system makes use of a client-server architecture design. The system will consist of three parts: a user application, a MongoDB database, and a web server hosted in the cloud. The user's web browser will get the webpage from the web server and retrieve personal, group, pin, and chat information from the MongoDB database via the internet. Users will communicate in their group chats through the server.



4.2 Hardware Environment

The hardware requirement for mobile devices is to be able to run a web browser. The hardware requirement for laptop devices is to be able to run a web browser.

There is no hardware requirement for the database as it is run through mongoDB and optimized for us.

The web server will be in a DigitalOcean so there is no hardware requirement for it.

4.3 Software Environment

The system will be run through a web browser, so it will support Google Chrome, Firefox, I.E, and Safari.

The database server will run on MongoDB. MongoDB is a free NoSQL database service in the cloud that can scale with our demands.

We will use a DigitalOcean to host the web server.

4.4 System Security

The system will need authorization and mongoDB built in security. The database will make use of MongoDB authorization. The users will have credentials to query their own data but will not have access to other user's data. There will be a database admin that will be able to add/delete data. We will use MongoDB's built-in security features to protect the database. The web server will be protected by the built-in security of the cloud service.

4.5 Development Environment

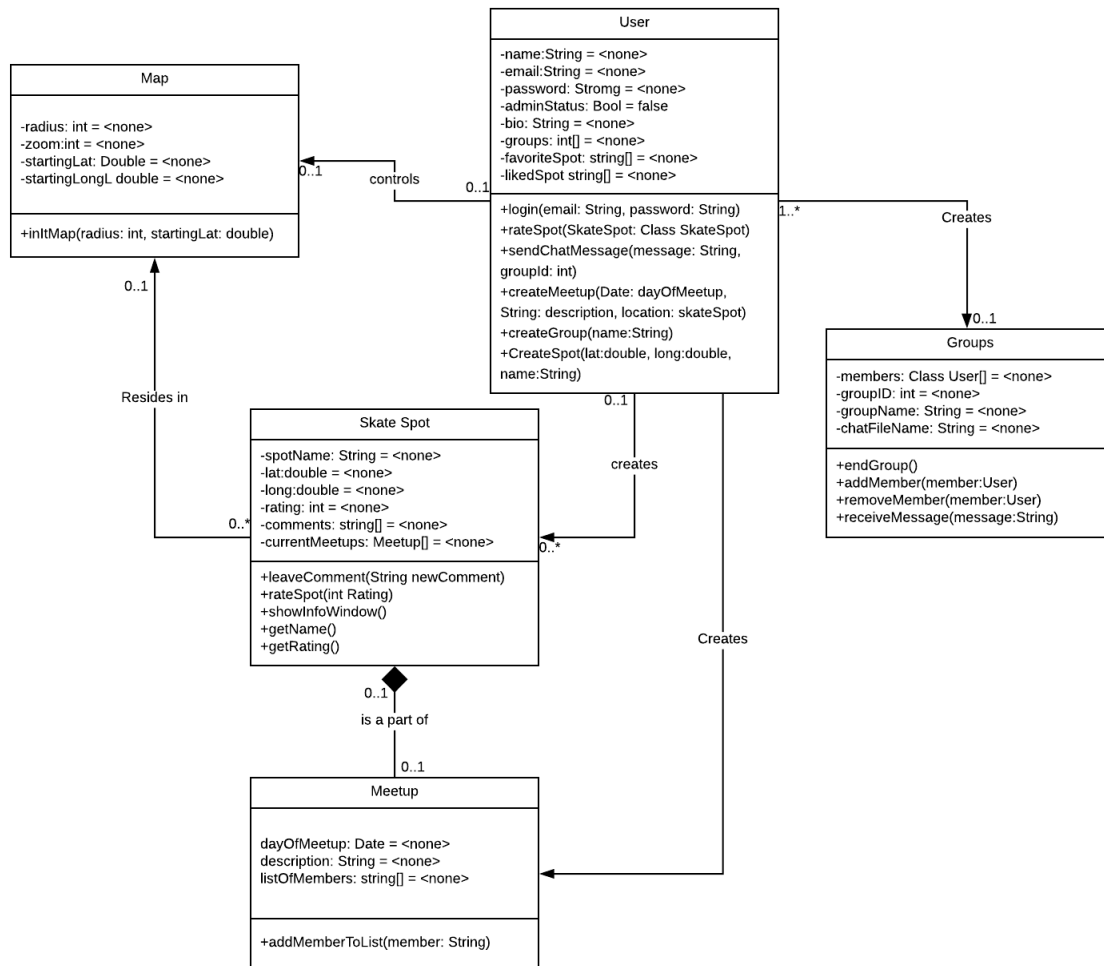
The front-end part of the application will be developed in JavaScript and HTML/CSS. These tools were selected because we have previous experience using them to create web apps. We will be using the node.js loopback framework to create the back-end operations. We chose this tool because it has integration with NoSql, and we have experience with this framework. We will use various code editors such as intelij, brackets, and notepad++. We will use these code editors based on each member's preference so that each member does not have to learn a new type of editor. Our team will make use of lucidCharts to create our diagrams. The reason is that it is free, and we have all used it in previous classes.

Section 5: Data (Structural) Design

Introduction

This section describes the data design of Skateable. It will contain a class diagram to show the classes of the system. The diagram will show each classes' attributes, operations, and relationship to other classes. This section will also have a data dictionary. The data dictionary will list each class in alphabetical order and give more detail about each class. Each class will have their name, visibility, and description shown. Each classes' attributes will have their name, type, visibility, multiplicity, and initial value shown. Each classes' operations will have their name, return type, visibility, scope, and a description shown.

5.1 Class Diagram



5.2 Data Dictionary (Metadata)

Groups
-members: Class User[] = <none> -groupID: int = <none> -groupName: String = <none> -chatFileName: String = <none>
+endGroup() +addMember(member:User) +removeMember(member:User) +receiveMessage(message:String)

Class Name: Groups

Visibility: Public

Description: This class controls the group created by a user. It holds the members of the group, the ID, the group name. It allowed users to create a group, add a member, send a message, and receive messages.

Attributes:

Name	Type	Visibility	Multiplicity	Initial Value
members	User[]	Private	1..*	<none>
groupID	Int	Private	1	<none>
groupName	String	Private	1	<none>
chatFileName	String	Private	1	<none>

Operations:

Name	Return Type	Visibility	Scope	Parameters	Description
endGroup	void	public	instance	<none>	This operation is called whenever removeMember is used. This operation checks if the user array becomes empty, and if

					it does then it deletes the group.
addMember	void	public	Instance	·Member: User = <none>	This operation takes the selected user and adds them to the group list.
removeMember	void	public	Instance	·Member: User = <none>	This operation takes the selected user and removes them from the group list.
receiveMessage	void	public	Instance		This operation takes the currently selected group, and queries for updates to the group chat. It then updates the group chat if it is not up to date.

Map
-radius: int = <none> -zoom:int = <none> -startingLat: Double = <none> -startingLongL double = <none>
+inItMap(radius: int, startingLat: double)

Class Name: Map

Visibility: Public

Description: This class holds the current location on the map that the user is looking at. It also holds the radius and zoom the user selected.

Attributes:

Name	Type	Visibility	Multiplicity	Initial Value
radius	int	private	1	<none>
zoom	int	private	1	<none>
startingLat	double	private	1	<none>
startingLat	double	private	1	<none>

Operations:

Name	Return Type	Visibility	Scope	Parameters	Description
inItMap	void	public	instance	<ul style="list-style-type: none"> · Radius:int = <none> · startingLat:Double = <none> · startingLong:double = <none> 	This operation takes in the user entered radius, and the currently selected latitude and longitude to search for nearby skate spots in the radius.

Meetup
dayOfMeetup: Date = <none> description: String = <none> listOfMembers: string[] = <none>
+addMemberToList(member: String)

Class Name: Meetup

Visibility: Public

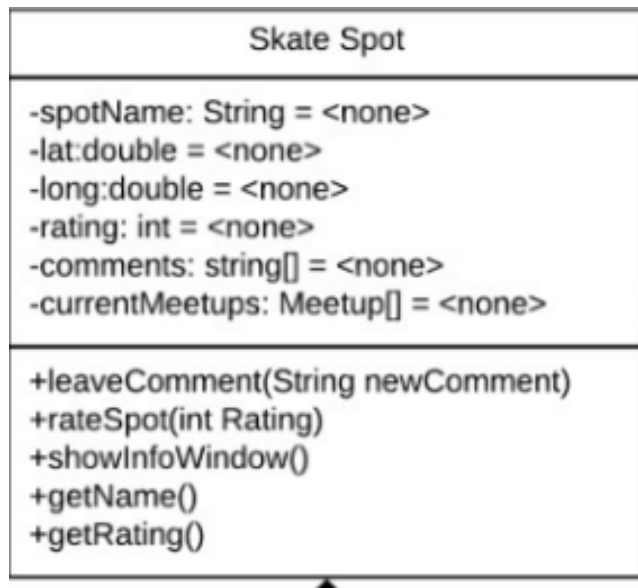
Description: This class holds the date, description, and list of members for a meetup event at a specific skate spot.

Attributes:

Name	Type	Visibility	Multiplicity	Initial Value
dayOfMeetup	Date	private	1	<none>
description	String	private	1	<none>
listOfMembers	String[]	private	1..*	<none>

Operations:

Name	Return Type	Visibility	Scope	Parameters	Description
addMemberToList	void	public	instance	Member:String = <none>	This operation starts when a user selects the attending “button” for the meetup and adds their name to the list of users attending the meetup.



Class Name: Skate Spot

Visibility: Public

Description: This class controls the information of a skate spot. A skate spot is found by its latitude and longitude. User's interact with this class by creating, rating, and commenting on the skate spot.

Attributes:

Name	Type	Visibility	Multiplicity	Initial Value
spotName	string	private	1	<none>
lat	double	private	1	<none>
long	double	private	1	<none>
rating	int	private	1	<none>
comments	String[]	private	0...*	<none>
currentMeetups	Meetup[]	private	0...*	<none>

Operations:

Name	Return Type	Visibility	Scope	Parameters	Description
inItInfoWindow	void	public	instance	lat:double = <none> long:double = <none>	This operation takes in the current latitude and longitude to check for a skate spot at this location and opens the info window on screen for the skate spot.
LeaveComment	void	public	instance	comment: string = <none>	This operation takes in a user entered string and then enters in the string into the skate spot comments attribute.
rateSpot	void	public	instance	rating: int = <none>	This operation takes the user input from the “user rating” widget for the selected skate spot and then adds the rating to the total user rating for that spot.

showInfoWindow	void	public	instance		this operation takes in the current latitude and longitude to check for a skate spot at this location and opens the info window on screen for the skate spot.
getName	string	public	instance		This operation returns the name of the group.
getRating	int	public	instance		this operation returns the rating of the skate spot.

User
-name:String = <none> -email:String = <none> -password: String = <none> -adminStatus: Bool = false -bio: String = <none> -groups: int[] = <none> -favoriteSpot: string[] = <none> -likedSpot string[] = <none>
+login(email: String, password: String) +rateSpot(SkateSpot: Class SkateSpot) +sendChatMessage(message: String, groupId: int) +createMeetup(Date: dayOfMeetup, String: description, location: skateSpot) +createGroup(name:String) +CreateSpot(lat:double, long:double, name:String)

Class Name: User

Visibility: Public

Description: This class holds the user information, and the operations users can perform. If they are an admin they will have an admin status denoted by the attribute adminStatus.

Attributes:

Name	Type	Visibility	Multiplicity	Initial Value
name	string	private	1	<none>
phone	int	private	1	<none>
email	int	private	1	<none>
password	string	private	1	<none>
adminStatus	boolean	private	1	false
bio	string	private	1	<none>
groups	Int[]	private	1..	<none>
favoriteSpot	String[]	private	1..	<none>
likedSpot	String[]	private	1..	<none>

Operations:

Name	Return Type	Visibility	Scope	Parameters	Description
login	void	public	instance	email: string= <none> password: string = <none>	This operation takes the user email and password as parameters and logs them to the service. It then fetches their information from the DB and updates the program with the information.
rateSpot	void	public	instance	spot:SkateSpot = <none>	This operation takes the user input from the rating widget for the selected skate spot and then adds the rating to the total user rating for that spot.
sendChatMessage	void	public	instance	message:string = <none> groupID: int = <none>	This operation connects a user to the user selected group chat, and sends the user entered text to the group chat. The message is then added to the

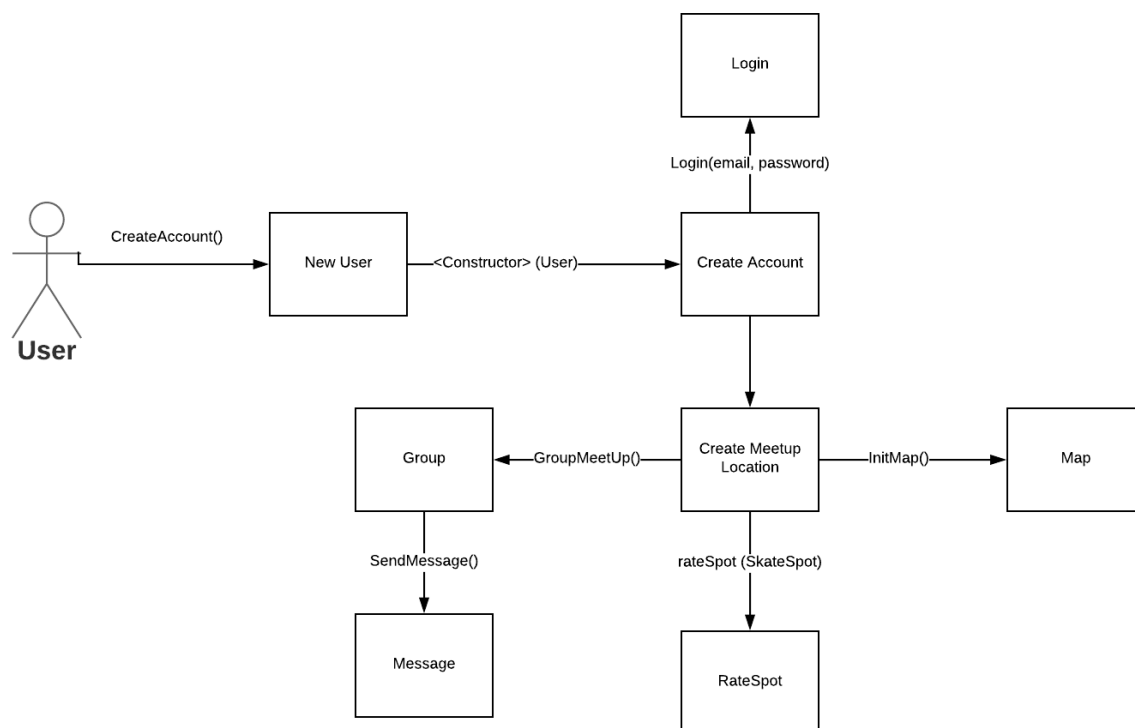
					group chat history file.
createMeetup	int	public	class	dayOfMeetup: Date = <none> description:String = <none> location: skateSpot	This operation takes the currently selected skate spot, gets the user entered date, and description to create a meetup at that skate spot for all users to see.
createSpot	int	public	class	Lat:double = <none> Long: double = <none>	The operation takes the current coordinates selected on the map by the user and the user entered name. It uses that information to create a skate spot.

Section 6: Behavioral Design

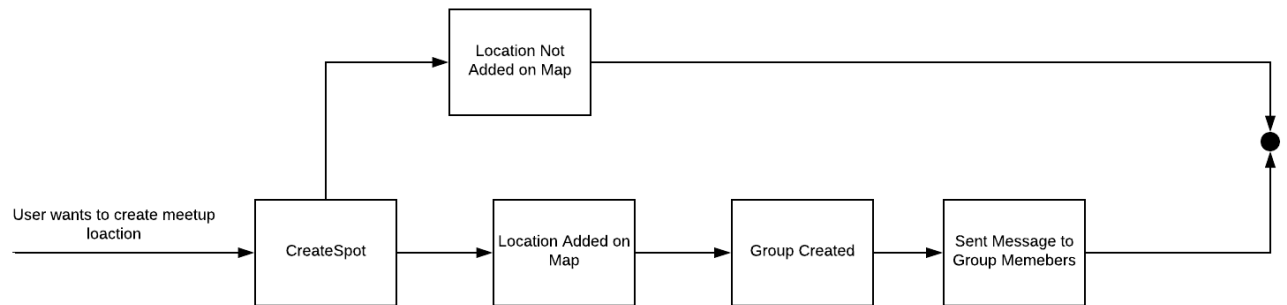
Introduction

This section addresses the user behavioral design for Skateable, which includes the System Services Implementation Model and Statechart Diagram. The collaboration diagram shows how each use-case is implemented and the relationship between the stallholders with the flow of each use-case. Class diagrams operations are included in the behavior model. Below the System Services Implementation Model is the state chart diagram that shows the state of each object in our system. The create a meetup location model is shown in the system of Skateable.

6.1 System Services Implementation Model



6.2 Statechart Diagram



Section 7: User Interface Design Plan

Introduction

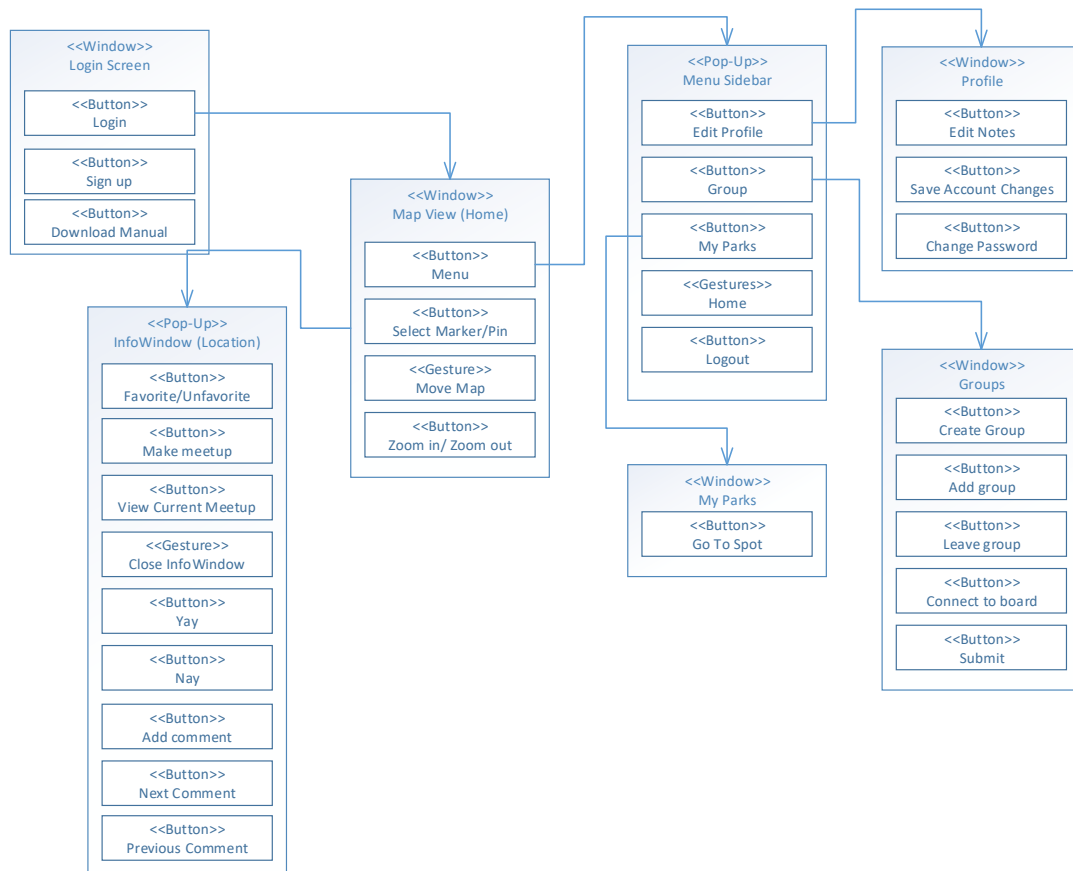
This section addresses the user interface layout for Skateable, as well as the user-driven navigation from page to page.

7.1 User Interface Requirements and Constraints

Our application, Skateable, will be used by a variety of individuals from many walks of life. However, given that Skateable will be available as a web application, it is expected that the user has some understanding of images and icons for various actions; ie, 'X' for exit, menu icon, settings icon, etc. Other than these given icons, the app will be self-explanatory. In the event a user needs additional tutorials on how to navigate/use Skateable, a 'help' section will be provided.

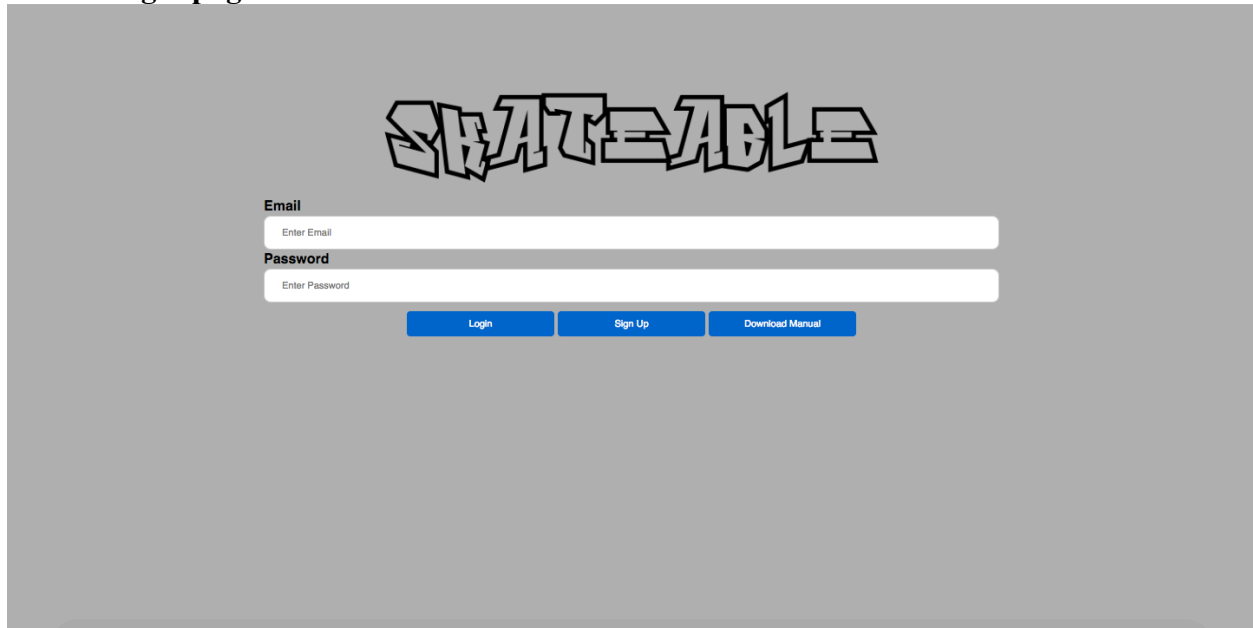
In the **Window Navigation Diagram** section, the series of diagrams will demonstrate how the user will interact and navigate through the system. For the **Screen Interface Design** section, there will be screenshots and designs representing the current template for the user interface.

7.2 Window Navigation Diagram



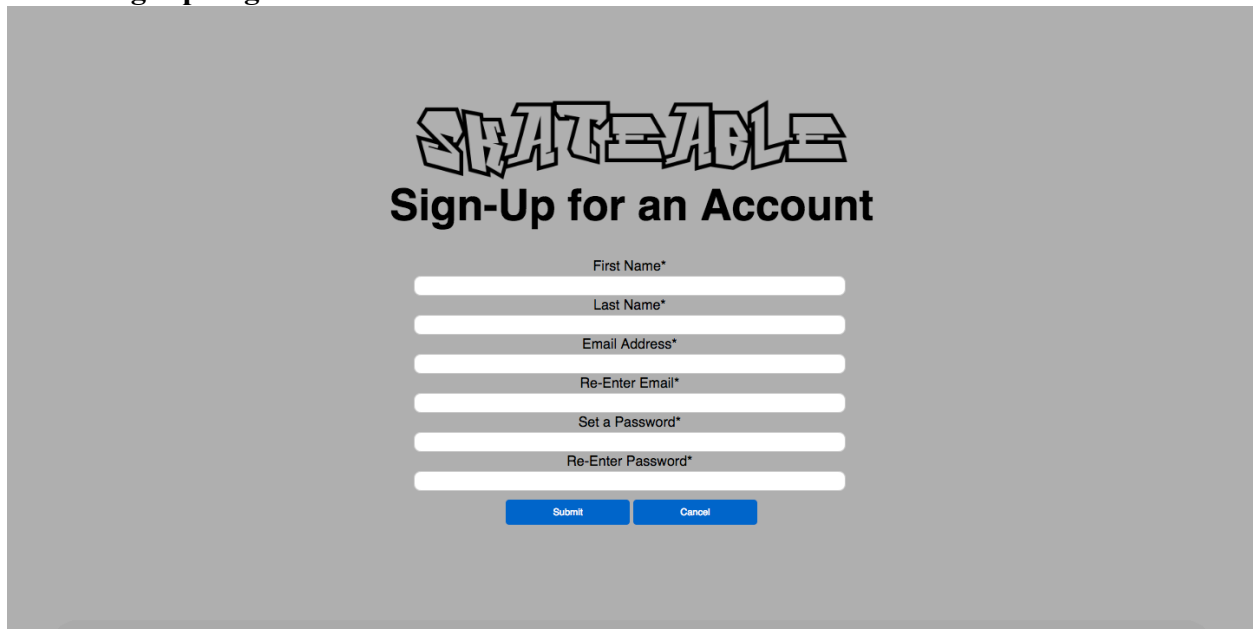
7.3 Screen Interface Design

1. Login page



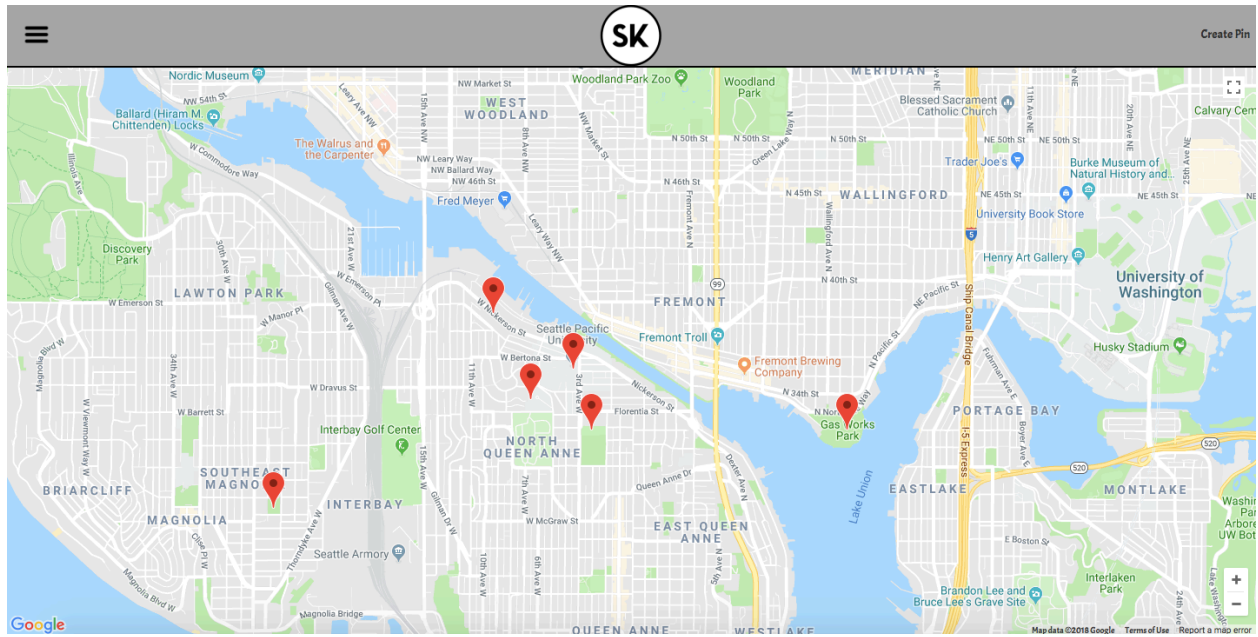
The login page features a dark gray background. At the top center is the 'SEATEABLE' logo in a stylized, outlined font. Below the logo are two input fields: 'Email' with a placeholder 'Enter Email' and 'Password' with a placeholder 'Enter Password'. Below these fields are three blue buttons: 'Login', 'Sign Up', and 'Download Manual'.

2. Signup Page

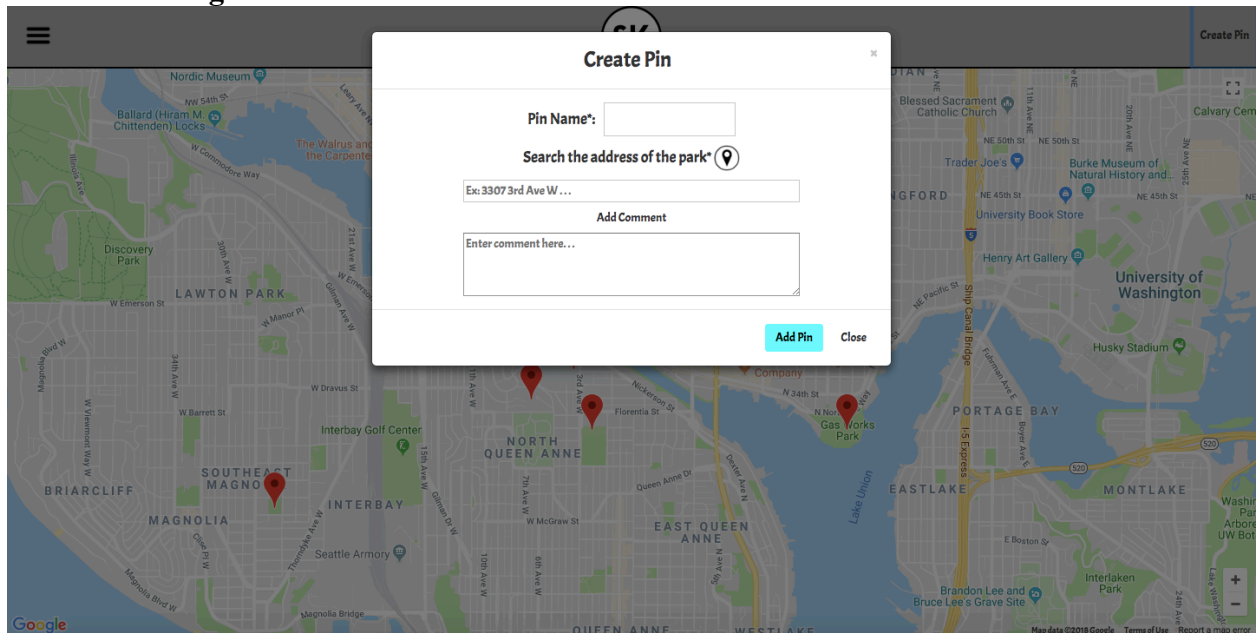


The signup page features a dark gray background. At the top center is the 'SEATEABLE' logo in a stylized, outlined font. Below the logo is the heading 'Sign-Up for an Account'. Below the heading are seven input fields: 'First Name*', 'Last Name*', 'Email Address*', 'Re-Enter Email*', 'Set a Password*', and 'Re-Enter Password*'. Below these fields are two blue buttons: 'Submit' and 'Cancel'.

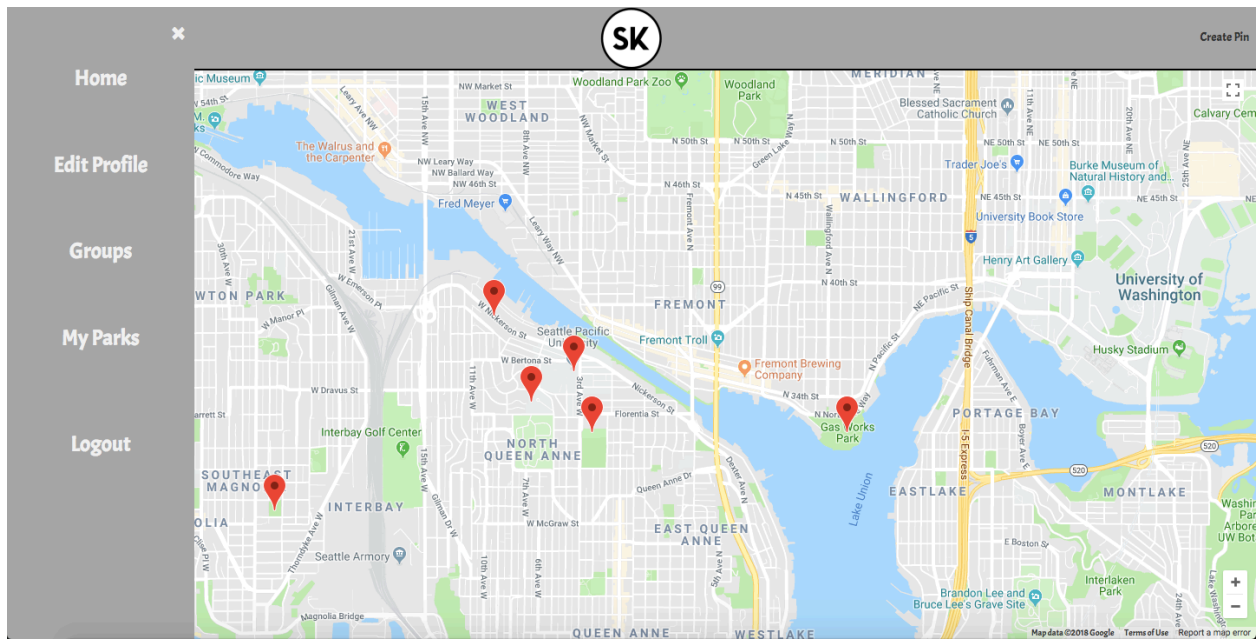
3. Home Page



4. Creating a Pin



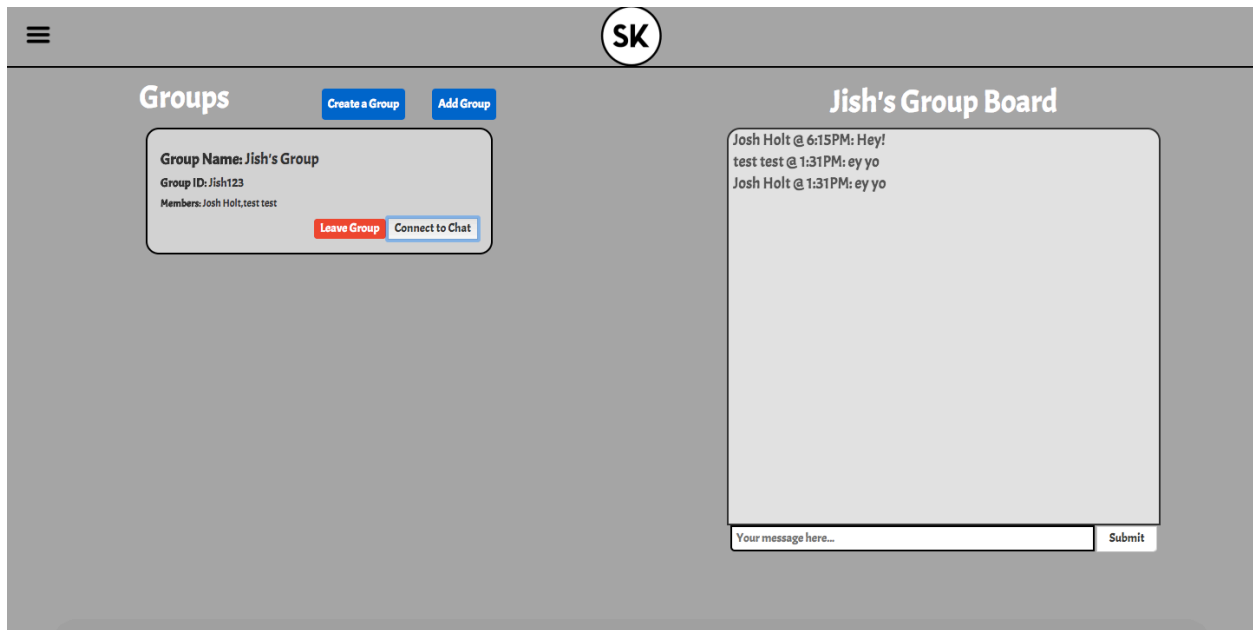
5. Sidebar view



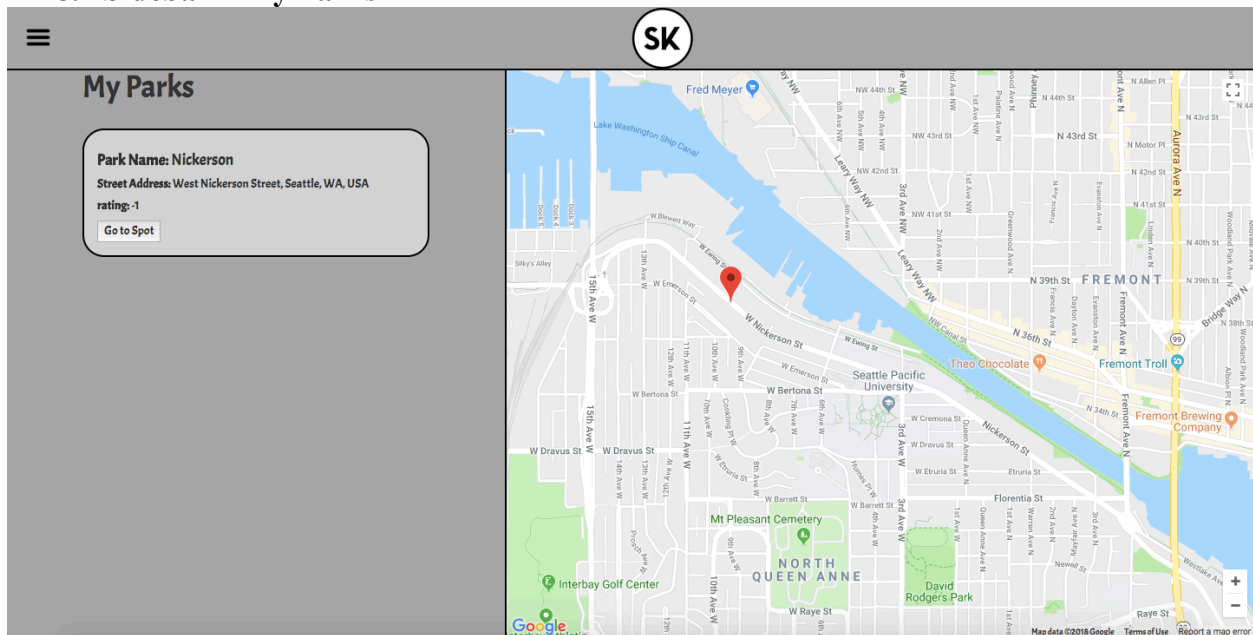
6. Sidebar – Profile



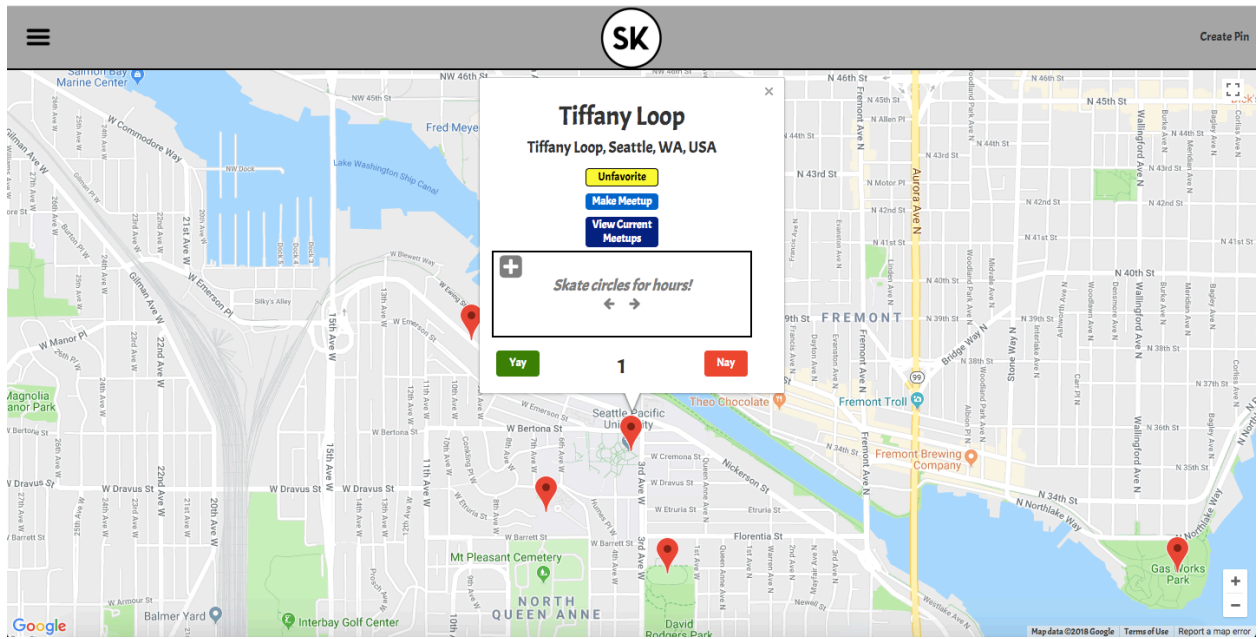
7. Sidebar – Groups View



8. Sidebar – My Parks



9. Marking a Meet up



Section 8: Development plan

Intro

This section will outline our development plan by week for spring quarter development.

Week 1 - (3/27)

- Frontend
 - Login/sign up pages

Week 2 - (4/3)

- Frontend
 - Framework for home page

Week 3 - (4/10)

- Frontend
 - Send out surveys for skaters to test the UI prototype

Week 4 - (4/17)

- Receive feedback from surveys and finalize UI.

Week 5 - (4/24)

- Backend work

Week 6 - (5/1)

- Backend work

Week 7 - (5/8)

- Combine frontend and backend
- Software basically done (right before Erickson conference May 11th)

Week 8 - (5/15)

- Quality Assurance for another team
- Send out surveys to skaters to test the beta version application by end of week 9

Week 9 - (5/22)

- Get feedback through an outside group's quality assurance on our project and make changes accordingly
- Receive feedback from surveys

Week 10 - (5/29)

- Make changes based on feedback from surveys

Finals Week - (6/6)

- Software ready to be delivered to client