PULL-UP

San Jose State University

Department of Computer Science

Professor: Ahmad Yazdankhah

Team: Cool Kids

Team Members:
Calvin Truong
Gabriel Orellana
Borum Chhay
Steven Gonzalez

Fall 2016 - CS160

Table of Contents ……

1. **Introduction**

1.1. Purpose of the system

This application hopes to solve the problem of having a hard time finding parking spots in compact cities such as San Francisco or San Jose. Pull-Up intents to ease the frustration that comes with not being able to find parking in parking garages or parking lots. This application will allow users to rent their driveways or rent driveway parking spots.

1.2. Scope of the System

Due to a limited time constraint, this application will embody San Jose State University and its surrounding area. It's focus is to help SJSU students find parking when the three main garages are full and parking isn't available.

1.3. Acronyms, and Abbreviations

GUI: Graphical User Interface
SJSU: San Jose State University
UML: Unified Modeling Language

1.4. References

- Apache Tomcat: Open Source Web Server that will host our HTML, CSS and data files.
- MySQL: Will be used to store and access data in our database.
- HTML/CSS: Language to Build and style website.
- JavaScript(AngularJS): Structured scripting language for creating dynamic web pages.
- Java: concurrent, class-based, object oriented programing language.
- JDBC: Java API that can access tabular data stored in our database.
- SVN: Version Control for development, this will allow us to keep track of changes and our application versions.
- Java HTTP Servlets

2. **Current System**

Current System does not exist. We as a team are in the process of designing and building our Greenfield application from scratch. Since we are building the system from scratch, section 2 of this document will be Omitted.

3. **Proposed System**

3.1. Overview

Pull-Up is an interactive application which will give users the ability to search for parking around them. The homepage will show all recent listings in user's vicinity. Users will

create an account regardless if they're using the application for renting their own parking or if they're using Pull-Up to find parking. User accounts are crucial in determining user/parking location, availability, price, spot dimensions, available time slots, and visual images. When using the application, users will be able to search for parking based on location, they will be able to select from a variety of options, and directly request parking privilege if parking is "Available". Once both parties involved confirm the rental, Pull-Up will update the availability of the parking spot selected and the listing will show up as "Not Available" to other users. Users enter credit card/debit card info to pay, and the amount is accredited to the driveway owner's account, which they can then cash out or use for their parking intentions within the app (credit/debit transactions will be simulated for now).

3.2. Functional Requirements

1. Application should be able to display recent listings based on location.
2. Users should be able to Customize user accounts.
3. Application should display reviews for parking listings (In development)
4. Listings should provide pictures of the parking space.
5. The application should also accept debit/credit card payments.
6. Users should be able to filter parking listings by attributes: Parking dimensions, location, availability.

3.3. Nonfunctional requirements

1. System has to be functional (Availability 99.9%).
2. The application should be extensible, allowing features to be added in the future.
3. The application should not allow users to book more than 1 parking spot if there is a time conflict between rentals.

3.4. System's Models

3.4.1. Use-Case Model and UML

Use Case Name	Pull-Up Sign Up
Goal	
This use case describes how the users create a Pull-Up account	
Participating Actors	
Customer, Pull-Up System	

Nonfunctional Requirements

1. User inputs should be responsive and interactive within the browser
2. The Pull-Up system responds in a timely manner (no more than 2 seconds)
3. The user should use a web browser to access Pull-Up.
4. User inputs are valid.

Glossary

Customer: The main user of the application looking to rent parking.
Pull-Up System: The application and its database

Primary Flow of Events

Trigger

User clicks “Sign Up” on the top right hand corner of the application

Primary Preconditions

1. The user has access to the application
2. The user is on the homepage

Steps	Action	System Response
1	The user clicks “Sign Up”.	New HTML page is displayed with the sign up form.
2	User enter their information. This includes: name, address, car model, license plate, email, etc.	The system accepts text information through input text.
3	User creates a username and password. Password criteria: must include 1 lowercase letter, 1 uppercase letter, and a number.	System checks database to make sure username isn’t taken by other user and that the password matches the criteria.
4	User clicks “Submit”.	The systems calls the database to store user information for future usage.
5	User waits.	System redirects user to the main homepage.

Primary Postconditions
Database will be updated with users information.

Alternate Flow of Events

Alternate Trigger

User enters invalid username/password.

Steps	Action	System Response
1	User enters a taken username and clicks submit.	The system responds with message “This username is taken, please choose a different username”.
2	User enters a non-compliant password and clicks submit.	They system responds with “Your password does not match the criteria, please enter a different password”.

Alternate Postconditions

Database won’t add invalid information/inputs.

Use Case Name	Pull-Up Parking Query

Goal

This use case describes the process of querying for filtered parking spots.

Participating Actors

User, Database

Nonfunctional Requirements

1. User must be accessing the website through web browser
2. System responses no longer than few seconds due to small amount of initial data

Glossary

User: The person who is making the search
System: Pull-Up Application and database

Primary Flow of Events

Trigger

User hits “search” button after choosing filters

Primary Preconditions

User must be logged in

Steps	Action	System Response
1	User navigates to search page	System sends response for the request to the page
2	User fills in appropriate text fields and checkboxes for their parking needs, then hits search button	System creates a SQL query for parking locations that meet the needs of user
3	User waits	System creates a grid of parking location objects based on result of query
4	User clicks on any of these objects to retrieve more info about parking location	System provides full page of info regarding the parking location

Primary Postconditions

User will be able to view all parking locations that respect their query

Alternate Flow of Events

Alternate Trigger

Database cannot find any parking locations that match needs

Steps	Action	System Response
1	User fills in filters that do not match any parking locations in database	System gives error message and asks to change filters if

		possible
Alternate Postconditions		
User is redirected to search page again		

Use Case Name	Pull-Up Request Parking Spot
Goal	
This is the process a user takes after deciding on a parking spot he/she likes	
Participating Actors	
User, Database	
Nonfunctional Requirements	
1. Safe and secure checkout process required 2. System responds no longer than a few seconds to make appropriate changes to database	
Glossary	
User: The person who is making the search System: Pull-Up Application and database Listing: Page where a parking spot's info is stored	
Primary Flow of Events	
Trigger	
User clicks "Request Button" on someone's parking spot listing	
Primary Preconditions	
User must be logged in	
Steps	**Action**	**System Response**

1	User clicks request on the parking spot page	System opens a new page for user to fill out info
2	User fills in info such as credit card number and security code	Systems validates information
3	User waits	System approves info
4	User sees receipt	System makes appropriate changes to the database and the parking spot's availability

Primary Postconditions

User has successfully rented a parking spot

Alternate Flow of Events

Alternate Trigger

Credit card info is incorrect or declined

Steps	Action	System Response
1	User enters invalid credit card info or is declined	System generates appropriate error message
2	User can either re-enter correct credit card info or clear up issues with bank, then try again	System tries the process again

Alternate Postconditions

User will be approved after one or more unsuccessful attempts

Use Case Name	Pull-Up Parking Listing Creation

Goal

The process of creating a parking listing

Participating Actors

User, Database

Nonfunctional Requirements

1. Process should prevent creation spamming
2. System responses no longer than 3 seconds.

Glossary

User: Person trying to create listing
System: Pull-Up Application and Database

Primary Flow of Events

Trigger

User clicks “Create Listing” button

Primary Preconditions

User is logged in

Steps	Action	System Response
1	User clicks “Create Listing” button	System sends a form response
2	User fills out info such as Listing name, Image, Price, availability, etc. and then hits Submit	System performs input validation
3	User waits	System updates database by inserting new listing into table and shows “success” message

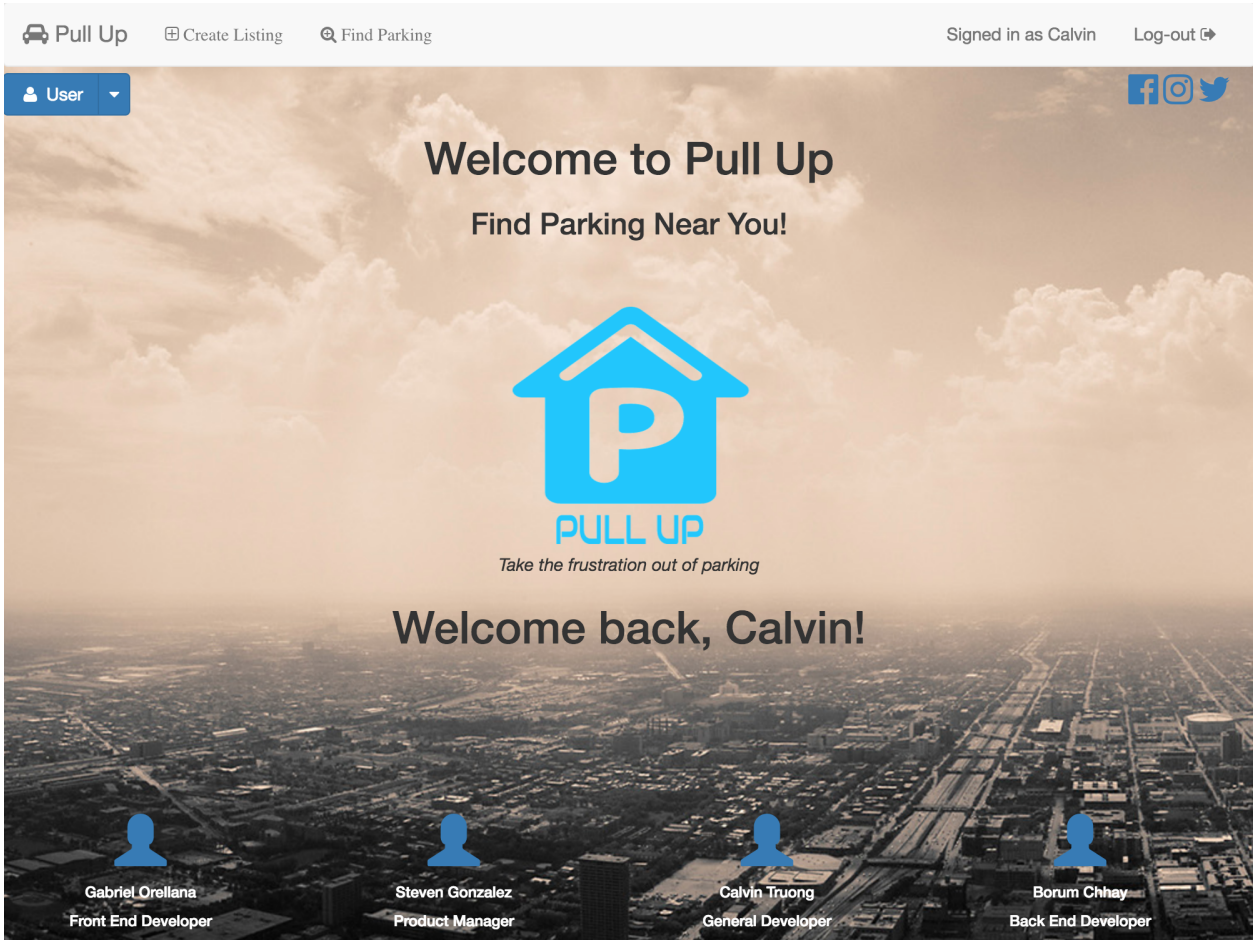
Primary Postconditions

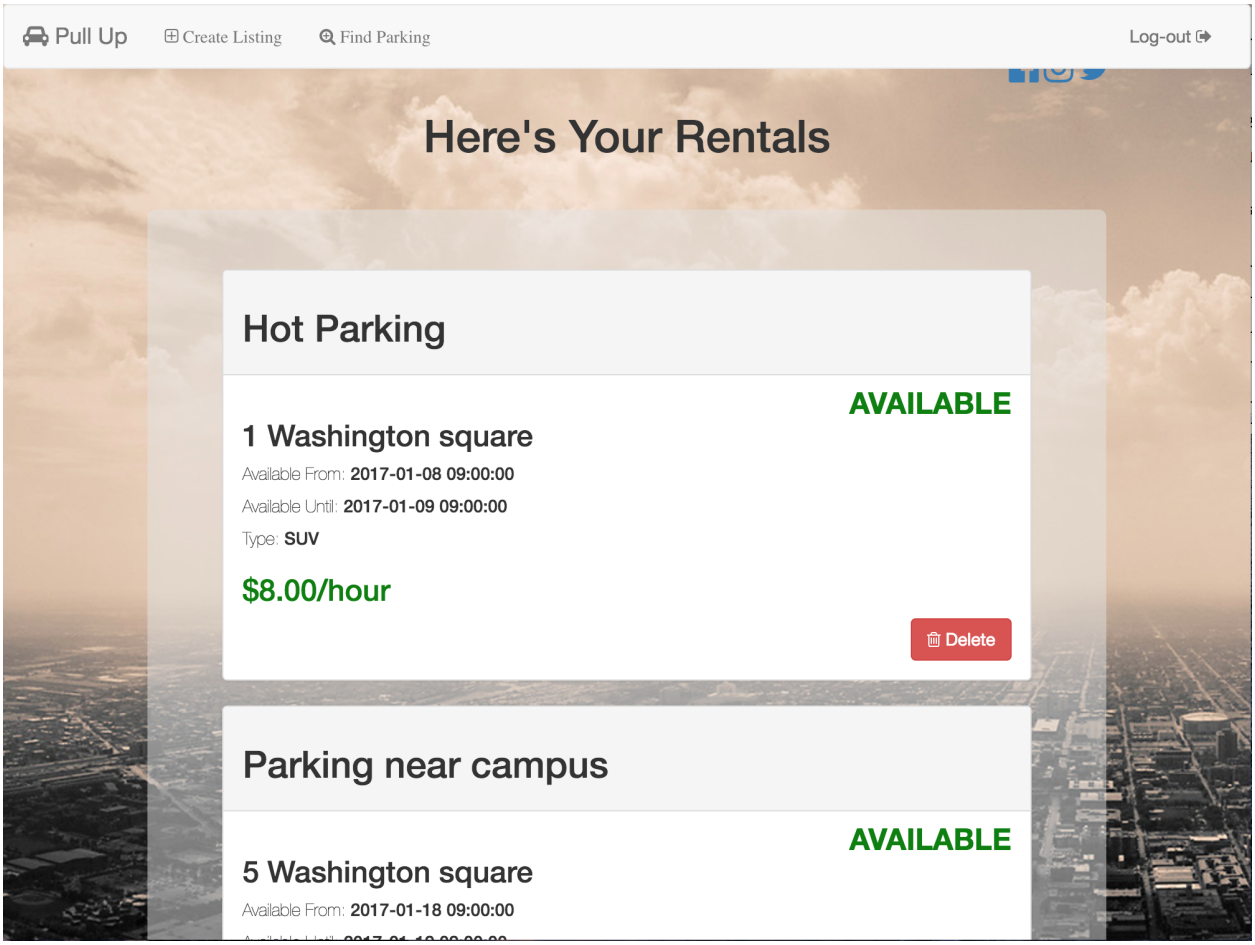
User will have a new listing available for rent

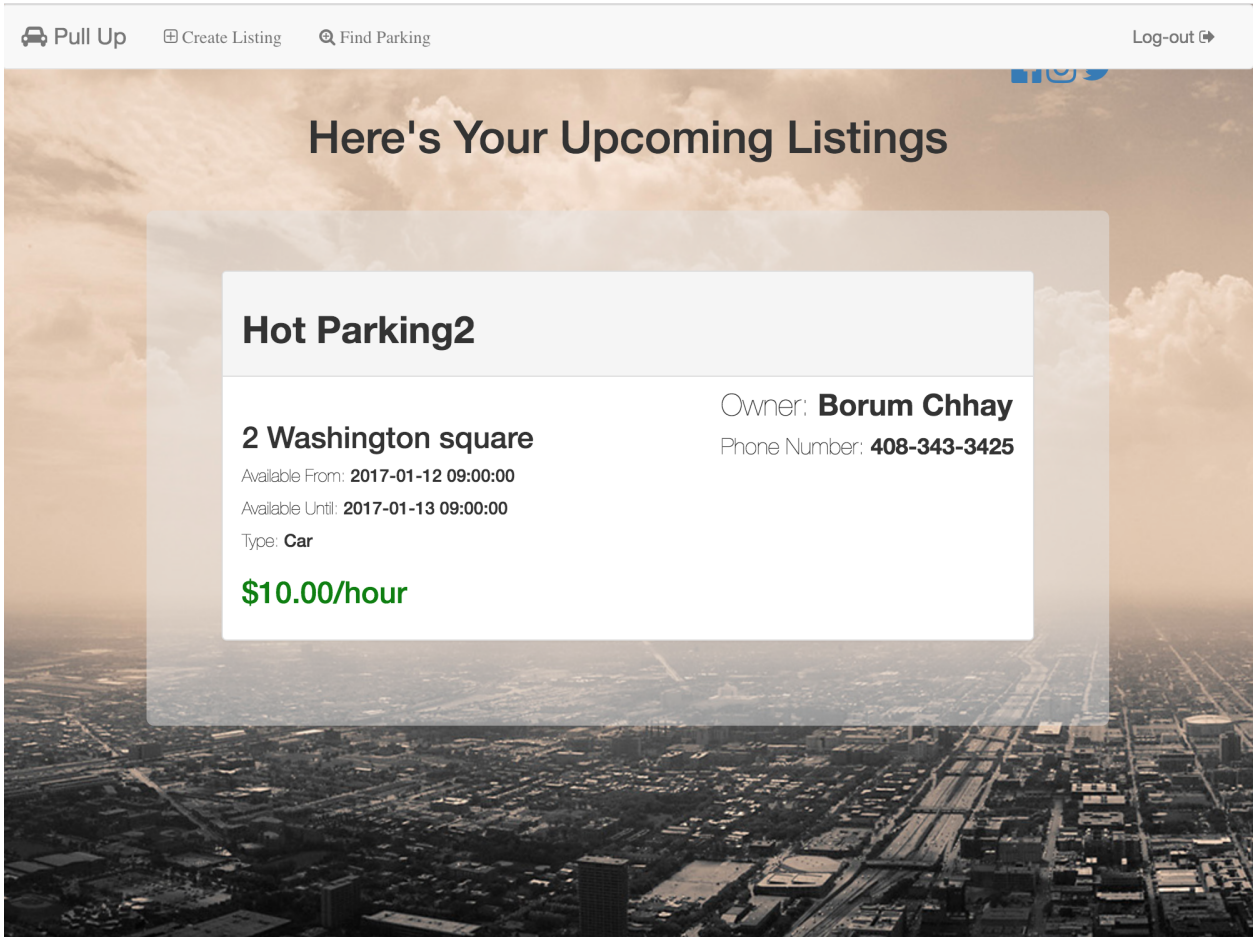
Alternate Flow of Events

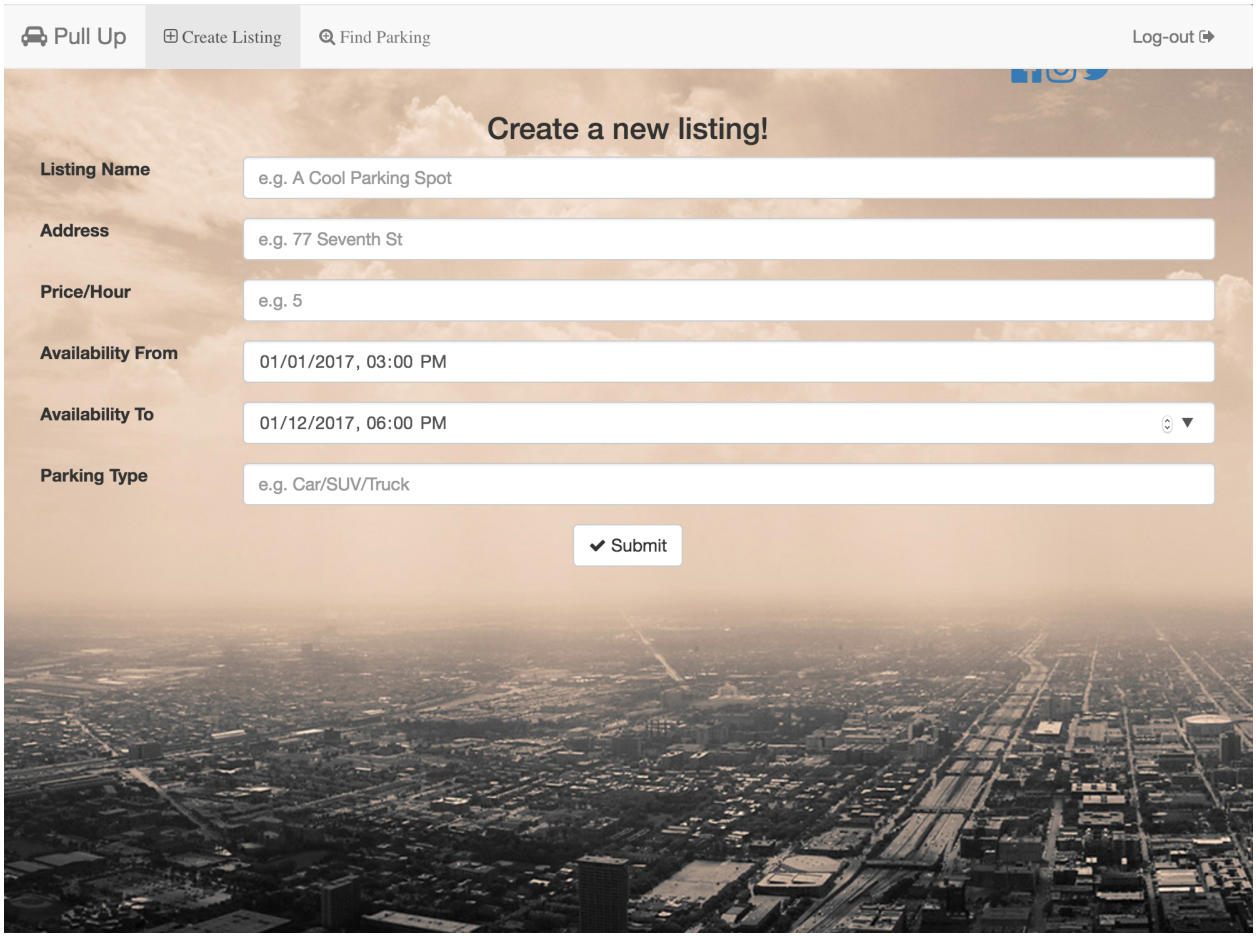
Alternate Trigger		
User enters invalid info		
Steps	**Action**	**System Response**
1	User enters invalid info	System creates an error message and re-prompts info
2	User sees error message and re-inputs info	System does input validation again and repeats until success
Alternate Postconditions		
User will have a new listing for rent after one or more unsuccessful creation attempts		

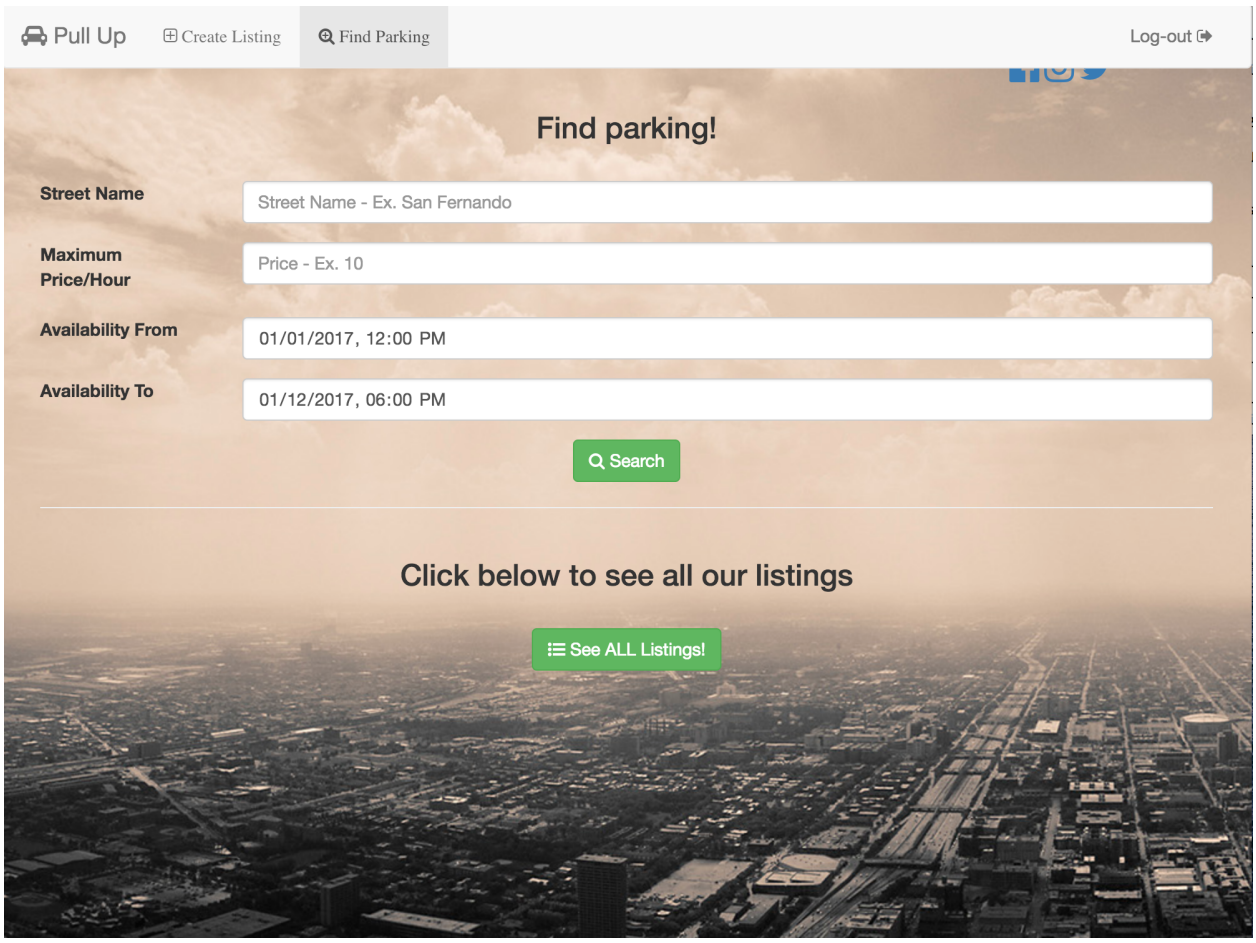
3.4.2. User Interface

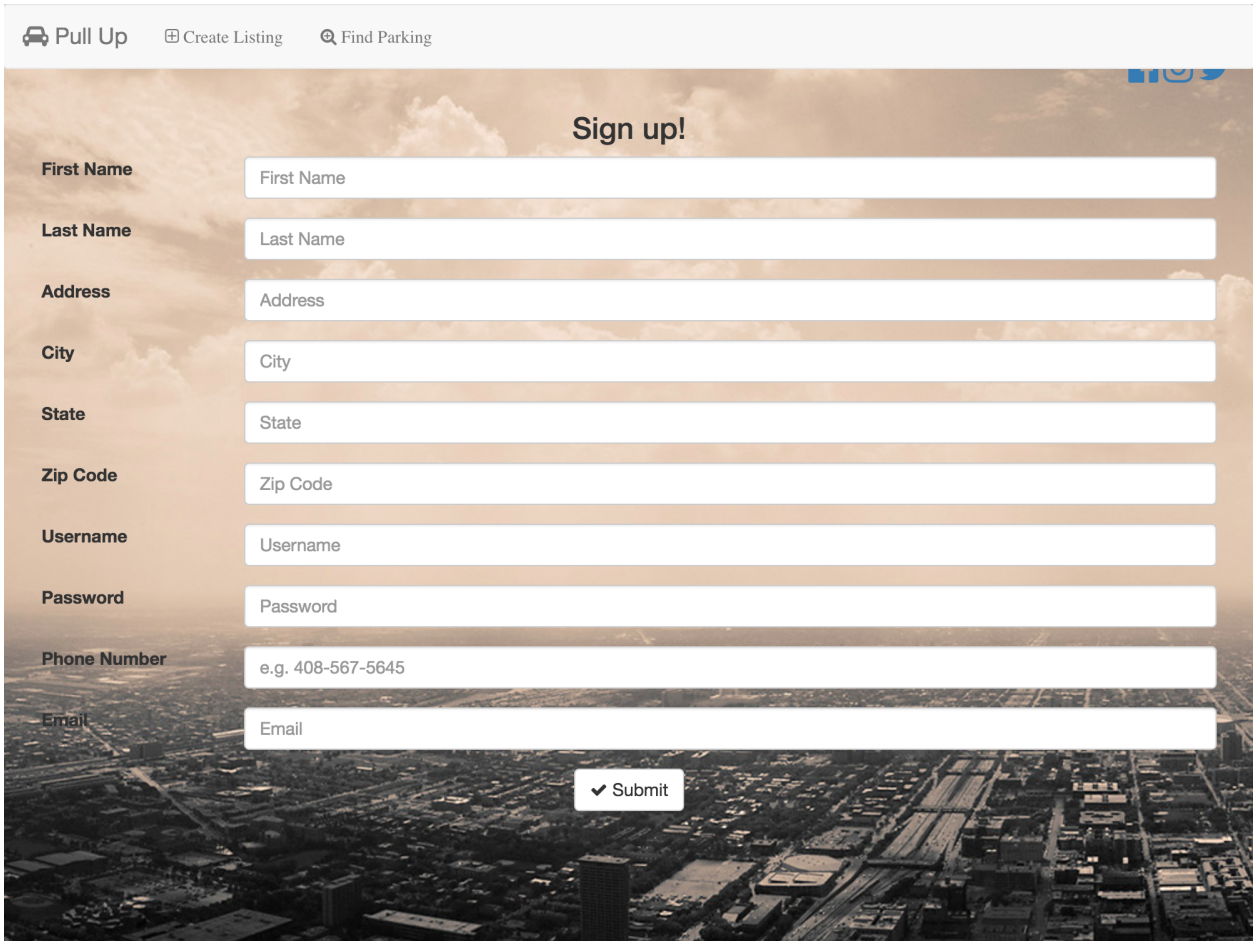

Pull Up
Create Listing
Find Parking
Signed in as Calvin
Log-out
User
Welcome to Pull Up
Find Parking Near You!
PULL UP
Take the frustration out of parking
Welcome back, Calvin!
Gabriel Orellana
Front End Developer
Steven Gonzalez
Product Manager
Calvin Truong
General Developer
Borum Chhay
Back End Developer

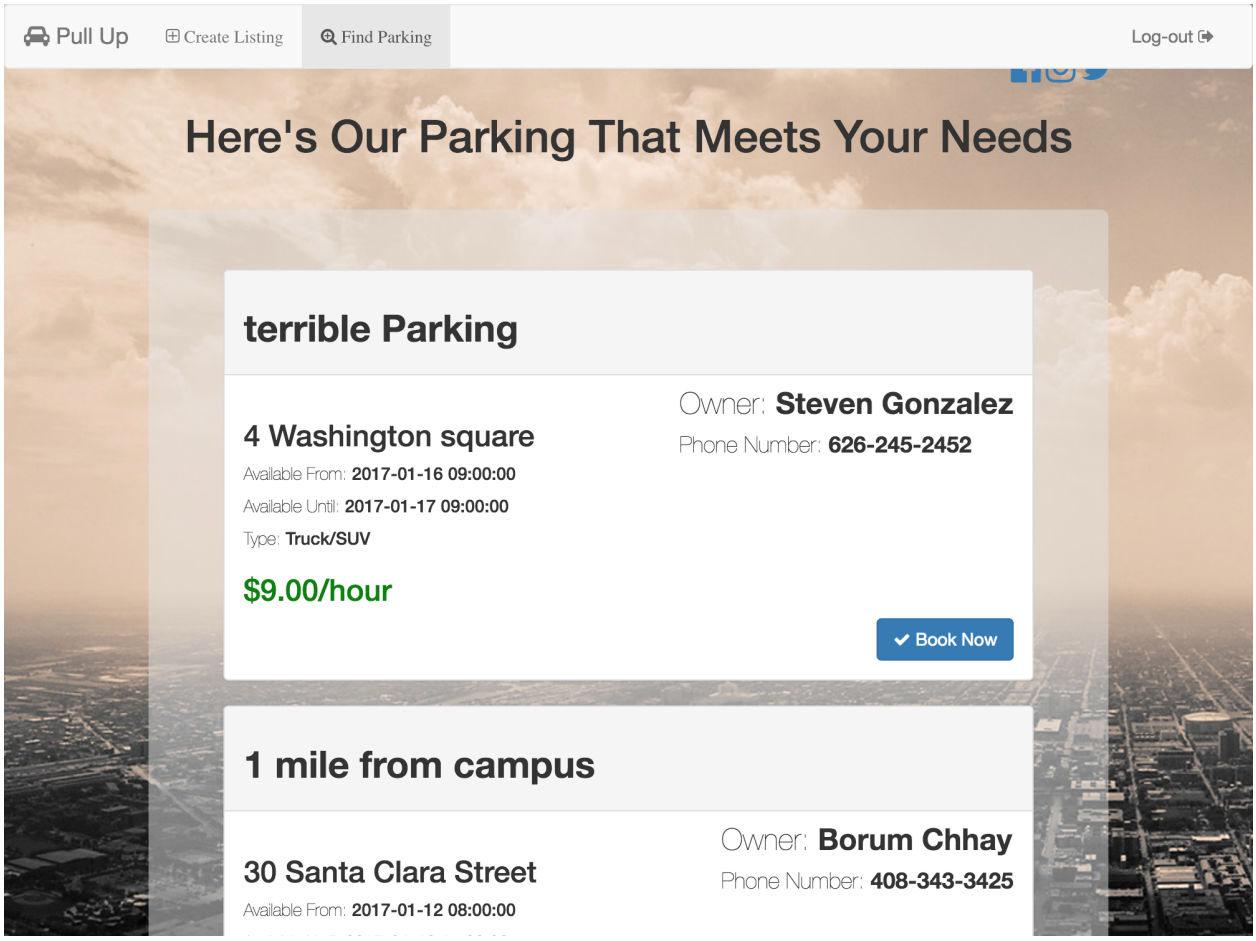

Pull Up
Create Listing
Find Parking
Log-out
Here's Your Rentals
Hot Parking
AVAILABLE
1 Washington square
Available From: 2017-01-08 09:00:00
Available Until: 2017-01-09 09:00:00
Type: SUV
$8.00/hour
Delete
Parking near campus
AVAILABLE
5 Washington square
Available From: 2017-01-18 09:00:00


Pull Up
Create Listing
Find Parking
Log-out
Here's Your Upcoming Listings
Hot Parking2
2 Washington square
Available From: 2017-01-12 09:00:00
Available Until: 2017-01-13 09:00:00
Type: Car
$10.00/hour
Owner: Borum Chhay
Phone Number: 408-343-3425


Pull Up
Create Listing
Find Parking
Log-out
Create a new listing!
Listing Name
e.g. A Cool Parking Spot
Address
e.g. 77 Seventh St
Price/Hour
e.g. 5
Availability From
01/01/2017, 03:00 PM
Availability To
01/12/2017, 06:00 PM
Parking Type
e.g. Car/SUV/Truck
Submit


Pull Up
Create Listing
Find Parking
Log-out
Find parking!
Street Name
Street Name - Ex. San Fernando
Maximum Price/Hour
Price - Ex. 10
Availability From
01/01/2017, 12:00 PM
Availability To
01/12/2017, 06:00 PM
Search
Click below to see all our listings
See ALL Listings!


Pull Up
Create Listing
Find Parking
Sign up!
First Name
First Name
Last Name
Last Name
Address
Address
City
City
State
State
Zip Code
Zip Code
Username
Username
Password
Password
Phone Number
e.g. 408-567-5645
Email
Email
Submit



4. **Glossary**

User: The person who is making the search
System: Pull-Up Application and database
Listing: Page where a parking spot's info is stored
Customer: The main user of the application looking to rent parking.
Pull-Up System: The application and its database
GUI: Graphical User Interface
SJSU: San Jose State University
UML: Unified Modeling Language
Apache Tomcat: Open Source Web Server that will host our HTML,CSS and data files.
MySQL: Will be used to store and access data in our database.
HTML/CSS: Language to Build and style website.
JavaScript(AngularJS): Structured scripting language for creating dynamic web pages.
Java: concurrent, class-based, object oriented programing language.
JDBC: Java API that can access tabular data stored in our database.

SVN: Version Control for development, this will allow us to keep track of changes and our application versions.