SW Engineering CSC648/848

Rently

Tool and Equipment Rentals

Section 01 - Team 01

Chase Alexander - Team Lead, full-stack

Benjamin McCullough - Github Master, frontend

Lauren Barer - Scrum master, database, cloud, security

Yu Hang Lee - Frontend lead

Chu Cheng Situ - Backend lead

Milestone Two

21 October 2021

Revision History

Revision ID	Revision Date	Revised By

1.Data Definitions V2

1. Equipment Category:
   The type of equipment the user is looking to rent/lend (e.g camping, hiking, tools, gym equipment, etc). The categories will be broken up using the following outdoor, indoor, home.
   a. equipmentCategory_ID: Primary Key : int
   b. description: varchar (255)

   Rows for the Equipment Category table:

   c. Outdoor Equipment (String)
   d. Indoor Equipment (String)
   e. Home Equipment (String)
2. Pricing_Penalties:
   The cost associated with a rental, and penalty for any damage including dents, breaks, and cracks
   a. Pricing_ID: Primary Key : int
   b. pricing: int(11)
   c. penalties: varchar(50)
3. Rental (Who, When, What, Time):
   The time that the equipment will be rented out for. A deadline for the return time.
   a. Rental_ID: Primary Key: int
   b. startTime: datetime
   c. endTime: datetime
   d. RegisteredUserRenter_ID: int(11)
   e. RegisteredUserRentee_ID: int(11)
   f. Equipment_ID: int(11)
   g. Pricing_ID: int(11)
   h. delivery: tinyint(4) - bool, yes or no
4. Registered_User:
   A person who signed up on the application must fill in their name, date of birth, address, payment method. This is a way to login either under renter or owner, to clearly understand who the user is.
   a. RegisteredUser_ID: Private Key: int
   b. userName: varchar(45)
   c. email: varchar(45)
   d. password: varchar(50)
   e. dob: datetime
   f. address: varchar(255)
   g. zipCode: varchar(9) which is a map API

5. Private_Chat:
   Direct messaging to renter or owner. Being able to have clear communication regarding any issues.
   - PrivateChat_ID: Private Key: int
   - RegisteredUserTo_ID: int(11)
   - RegisteredUserFrom_ID: int(11)
   - chat: varchar(255)
6. Equipment:
   To instore the location for renter and owner, To be able to filter results within a certain distance of the renter
   a. Equipment_ID: Private Key: int
   b. RegisteredUser_ID: int(11)
   c. EquipmentCategory_ID: int(11)
   d. Pricing_ID: int(11)
   e. description: varchar(45)

2. Functional Requirement V2

1. User Accounts (1)
   a. First-time users can create an ID. (1)
      - Users must fill in the username, password, confirm password, and full name.
   b. Users can log in to their ID. (1)
      - If the user has an account, then the next time the user would require to enter their username and password to login into their account.
   c. Users can log out. (1)
      - After browsing through the application, users can log out their account.
2. Result Filtering (1)
   a. Users can filter the equipment they want to rent by category. (1)
      - For example, categories would be outdoor, indoor or gym equipment.
   b. Users can filter the distance that they can go to pick up the equipment. (2)
      - For convenience, users can choose the distance that they are willing to pick up the equipment.
      - For example, User_1 wants to pick up indoor equipment in a radius of 3 miles.
3. Price and Date (1)
   a. Users can see the rental price of the equipment (1)
      - When users are looking to rent an equipment, users are able to see the price per day.
   b. Users will get a penalty if they return equipment late or damaged (1)

- If the users damage the equipment or return it late, users will need to pay a small amount of penalty to the owner when they return the equipment.

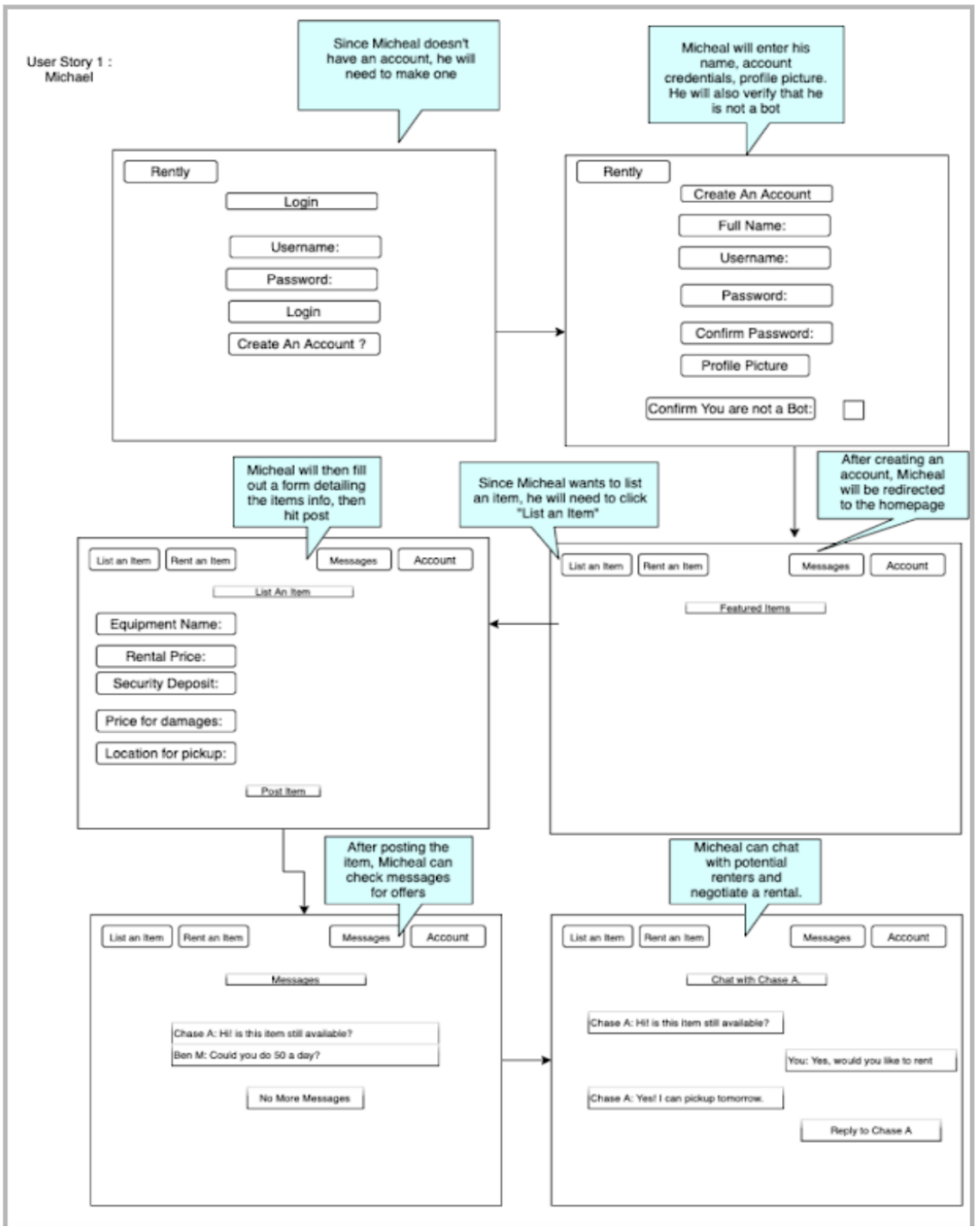
c. Users can choose how many days they want to rent it (2)
   - For example, User_2 wants to rent a kettlebell for 7 days.

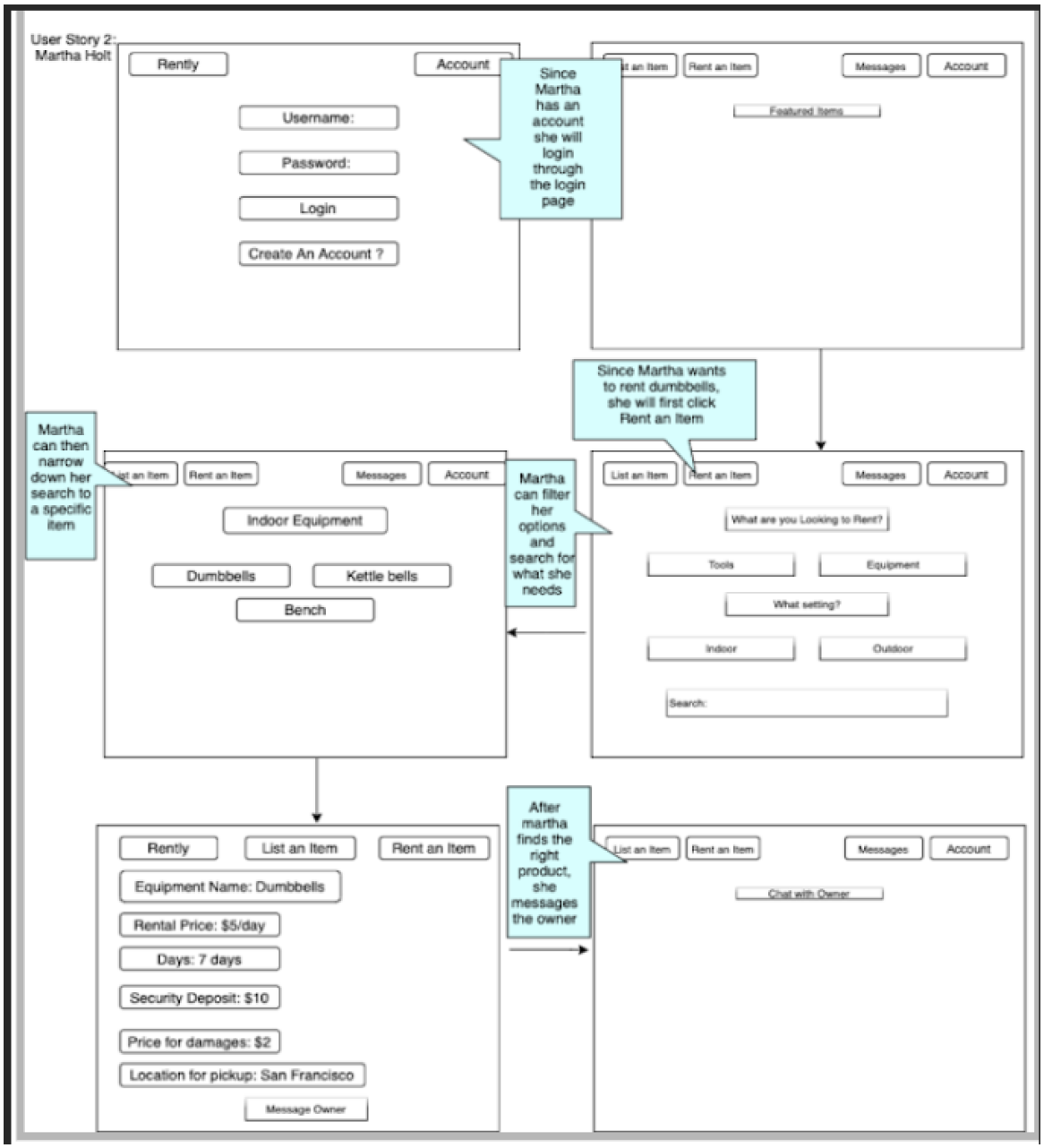
4. Posting
   a. Users can rent out their equipment with the information of pickup location (1)
      - Users that are going to list the equipment would need to provide the pickup location in order for users to see how far is the distance to pick it up.
   b. Users can set the prices for certain kinds of damage (1)
      - For example, User_3 lists an item and sets the price for damage as $2 for broken pieces, $4 for missing parts.
   c. Users can select the amount for the security deposit (1)
      - For example, after User_4 lists the item, User_4 selects $10 as a security deposit.
   d. Users can decide if they want to offer delivery for an extra fee (3)

5. Chat Box
   a. Users can create a new conversation with the owner of the equipment to confirm the rental (2)
      - Chat Box is used to communicate between the owner of the equipment and the users that are going to rent the equipment to make sure that when the user is going to pick up the equipment.
6. In-app payment processing (3)
   a. Debit / Credit Card Payment
      - Visa, Master,American Express and Discover card payment.
   b. Bank Transfer

(1- must have; 2 - desired; 3 - opportunistic)

3. UI Mockups and Storyboards





UX Validation Meeting Notes:

Useful: Our website is very useful because it's very easy to operate in regards to renting items. Useful because people can find what they are looking for in the categories using our search function. It's easier to narrow things down to what you would like specifically.

Usable: Very slim and clean getting straight to the point of trying to get you to rent whatever you need in a fast past. By using fast and reliable messaging gets you in contact with the renters as quickly as needed.

Findable: Easy navigation through the website finding exactly what you are looking for in specific categories. Not messy, easy to read and understand what is going on in the page.

Credible: Renters have to verify their address and emails to get credibility. Users have to go through a recaptcha process during login/account creation to verify credibility. Adding profile pictures will make it feel more user friendly.

Accessible: Delivery of the item is designed to help people who are not able to come and get it themselves. The website would also be accessible through any browser and web page on any mobile or computer device.

Desirable: Yes, because it allows you to reach a bigger audience to rent your item. It is also desirable because you don't have to deal with store policies and its easier to rent items. Gives you access to items closer to you and within your desired area.

Valuable: It's valuable because it lets everyday people make income with items that they have just lying around. Users save money and gas by renting close by and getting a better budget deal.

4. High Level Architecture, Database Organization

DB Organization:

Our database is going to be organized as following:

- 2 types of individuals using our site, user who will rent the objects, and renter who will rent out the objects
- User and Renter will need to create a user account, there will be a login for their account.
- Data Types Include: Outdoor Equipment, Indoor Equipment, Home Equipment
- The individual when coming upon the page can choose if he/she would like to be the user or renter.
- Once they choose they will be prompted to the correct login page.
- The user page will have 3 icons with our equipment options to choose from depending on what they desire to rent.
- Once the user chooses a category they will be prompted with a search bar which would filter all items posted.
- Once an item is chosen and location is determined, the person who has the item and is the closest will pop up.
- The user can then fill out the renter form, which would hold the option of pick up or delivery, info about the user, duration of the rent, and security deposit. If the user agrees with the form's information they can proceed.
- The renters page will have an option to manage current rentals, add new rentals, be able to provide images for the user, change pricing, and message with the user through our chat icon.

Tables:

Table: Register_User

Columns:

RegisteredUser_ID	int(11) PK
userName	varchar(45)
email	varchar(45)
password	varchar(50)
dob	date
address	varchar(255)
zipCode	varchar(9)

Table: Equipment

Columns:

Equipment_ID	int(11) PK
RegisteredUser_ID	int(11)
EquipmentCategory_ID	int(11)
Pricing_ID	int(11)
description	varchar(45)

Table: Equipment_Category

Columns:

equipmentCategory_ID	int(11) PK
description	varchar(255)

Table: Pricing_Penities

Columns:

Pricing_ID	int(11) PK
penities	varchar(50)
price	int(11)

Table: Private_Chat

Columns:

PrivateChat_ID	int(11) PK
RegisteredUserTo_ID	int(11)
RegisteredUserFrom_ID	int(11)
chat	varchar(255)

Table: Rental

Columns:

Rental_ID	int(11) PK
startTime	datetime
endTime	datetime
RegisteredUserRenter_ID	int(11)
RegisteredUserRentee_ID	int(11)
Equipment_ID	int(11)
Pricing_ID	int(11)
delivery	tinyint(4)

Media Storage:

In our case the renter would have to upload photos of the item that he is renting. Also, the renter and user will need to upload their profile picture. Our website will upload these images as a jpeg with an image resize of 40 mb. Our database would store these images as a path to the item which is being rented and the profile picture that the renter and user uploaded, for instance a tent will be stored under outdoor equipment. The user would be able to look at these images when filtering and on each renters page and all users profile images.

Search and Filter:

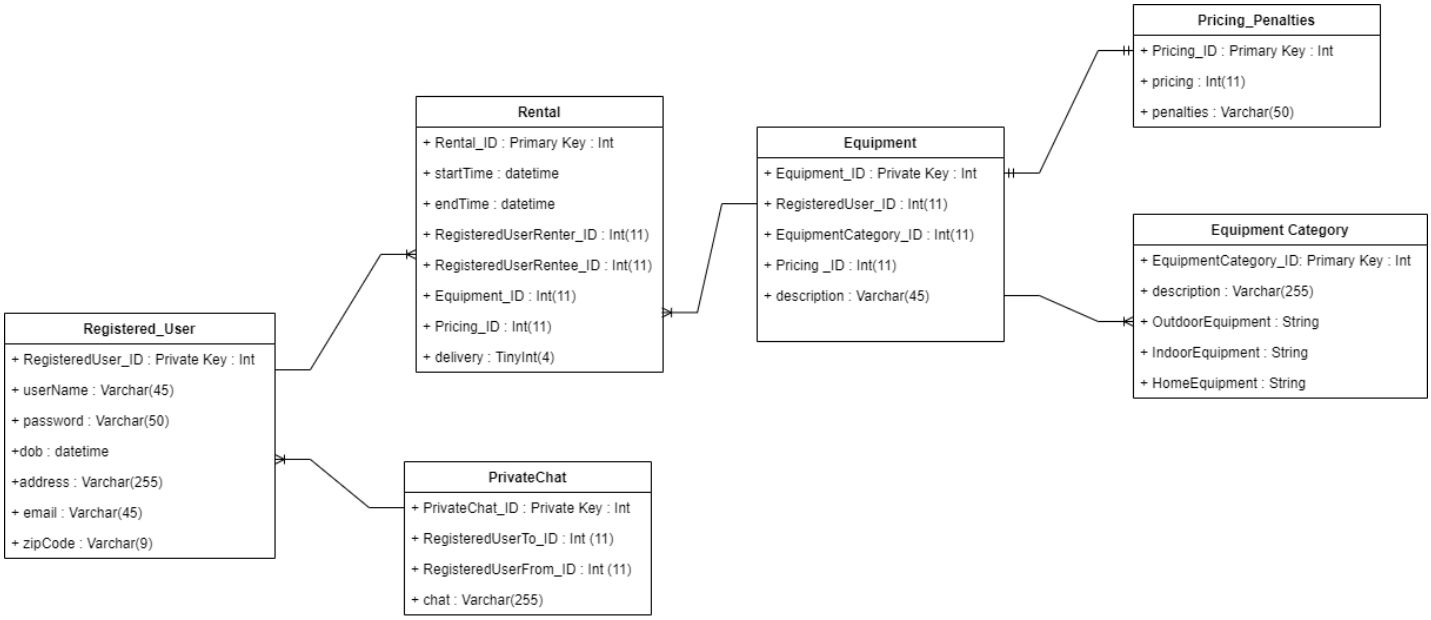
At first the user is prompted with 3 categories, when a specific category is chosen they can then search within that category. We will use the SQL search query and table to store all of the possible options in that category and what is available with the renter. Our search will run through and find the exact item the person is looking for. Once they find the item our filter can filter out the location to see which is in their boundaries. For example I as the user need a treadmill. On the main page I select indoor equipment which then will bring me to that page. Once I'm on the indoor equipment page I type in the search bar treadmill. That search will run through the database and only return treadmills. Once I am prompted with 4 treadmills, I go to the filter and select within 20 miles. The filter will use the map API to get the radius of my zip code for 20 miles, which then filters my options down to 2 treadmills in my area.

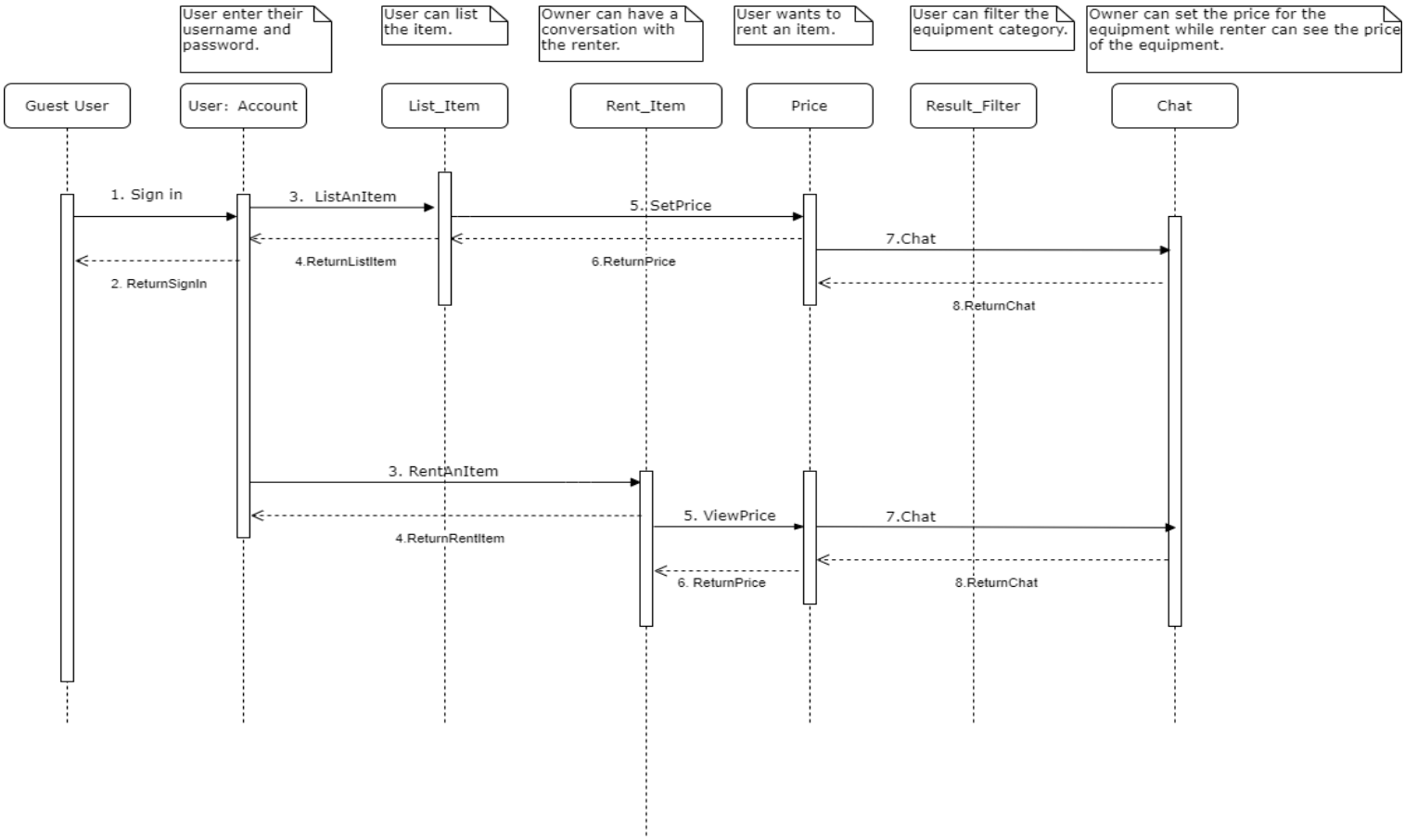
Add/Delete/Search Architecture:	Functional Requirement:
Add/Delete/Search for Users	When users register
Search/Display for Equipments	When users search on the equipments they want to rent
Add/Delete to Equipments	When users add or delete the equipments
Search for nearby store using map api	When users want to search nearby store
Display to Private Chats	When users decide to direct message to the owner

API's:

For our application we will use Express to create a REST API, and we will use json as our data format. We will use two API's which are third party. The first is a map API which will hold the map and filter it based on zip code and desired miles to zip code(which is zip code: varchar 9, which is the google maps api). The second API will be a messenger API which we will use for users and renters to communicate. We will also use our own API's that are created using express.

5. High-level UML Diagrams





6. Identifying Key Risks

We do have some level of risk when it comes to skill. A few of our team members were having a hard time understanding how to set up ExpressJS and use it to send data to the frontend. Another group member had never used React and wanted to get caught up on the basics. To address these risks we have set up a proper study plan for everybody so that they can get up to speed with the technologies that they are least familiar with. Each group member is spending a set amount of time every week studying and practicing their assigned technologies.

Any risks relating to our scheduling have been managed. Everybody knows what assignments they have due when, and every week we go over them during our group meetings. Our group is using Trello to keep each other up to date on our task progress. If any change happens in a certain person's progress we are all notified and able to see the change on our Trello workspace.

So far we do not have any teamwork risks. Everybody regularly attends the meetings. If a member has to miss a meeting for some reason they let the team know in advance, and they are able to catch up on what they missed by reviewing the meeting minutes and asking questions. Currently, everybody has stayed on top of their tasks and gotten them completed by the specified deadlines.

Regarding legal issues, we do have some level of risk that still needs to be managed. We have no issues with software licensing, but there is another company that exists with the same name as our application (Rently). To manage this risk we will use Rently only as a class/resume project and will not launch it for public use or monetize it without changing the name.

7. Project Management

For the milestone 2 tasks, I broke up the pieces of the assignment and assigned them to the necessary group members. The frontend team was assigned tasks related to the frontend, the backend team was assigned tasks related to the backend, and the team lead was assigned tasks relating to project management. I also divided the two major parts of the assignment (the M2 document and the vertical prototype) evenly between the first 2 weeks and saved the third week as extra time for us to handle incomplete work or unforeseen errors.

At our scrum meetings, we have meeting agendas and each agenda has at least one item relating to each group member so that they can catch the team up on their progress since our previous meeting. Our team's progress is transparently shared in a way that everyone knows the status of the whole project. Oftentimes our team members will use screen share to show us where they are at with their assignment. Outside of the scrum meetings we use Trello. Trello allows everyone in the team to see all of the current tasks, who is responsible for them when they are due, and at what stage each task is at in its completion.