# **Introduction and Context**

This requirements definition document will aid in learning the goals our group aims to achieve with the Big Blue Parking Genie web application project. The problem being addressed by this web app is the streamlining of reserved parking for college events.

As of now, there are only a few established ways of attendees parking for school sponsored events, these include designated parking areas with toll booths, the side of a public street, and parking somewhere, possibly faraway, that is legal. This end goal of this project is to have a working app that enables a college (USU) to create agreements with local property owners that’ll lend their parking spots to event attendees for a fee. This app will efficiently handle many scenarios that could happen with this underlying context.

Throughout this document the following will be specified in more detail: Users and their goals, the functional and non-functional requirements, and the future features. Users and goals describe the entities that will use and be used in this software system. The requirements tell of planned features, the functional specifies what the software will be capable of and non-functional gives details on the development process, development tools being used, and group rules. The future features section is self explanatory but will not be full of monstrous ideas due to our group focusing on the bare essentials of what does and does not need to be included.

# **Users and their Goals**

There are three differing groups that the users of this software system are categorized into, human users, front-end software, and back-end software. Human users include customer, parking-spot owner, and college supervisor/representative. The front-end user is the user interface that acts as the means to reserve a spot for a customer and as an interface to the back-end. The back-end users only communicate with the front-end users and other back-end users.

The customer human users goals are to browse parking spots available for a certain event and/or rent said spot. In our system, the parking-spot owner gets paid for rental and use of the parking spot(s) agreed upon with the college. Lastly, the college supervisor/representative is the human user that is hired by the college to manage all of the parking spot agreements and college event creation/handling.

There is only one front-end user in this system which is the user interface. The UI will be the only component that the customer interacts with. This will include an easy to navigate menu on the website and will be the tool used to fulfill the customers goal of renting a parking spot. The back-end users are the server and the database. The server will respond to client (front-end) requests and also communicate with the database to read and write data. The database will hold all of the data necessary for the application to function properly.

# **Functional Requirements**

## **User Authentication**

### 3.1.1 Description:

### User will access their account by logging in with their username (email) and password.

### 3.1.2 Stimulus/Response Sequences:

When the user hits the login button, their credentials will be authenticated and they will be redirected to their account homepage. Their homepage will be generated based on their role/permissions.

### 3.1.3 Functional Requirements:

### UA-1: Set up database permissions

### UA-2: Create login portal

UA-3: Create login/authentication API endpoint

## **User Registration**

### 3.2.1 Description:

### Those interested in renting out their parking space or renting those spaces should be able to create an account to do so.

### 3.2.2 Stimulus/Response Sequences:

### When the user hits “Register” they’ll be redirected to a page where they can input their information used to identify their account.

### 3.2.3 Functional Requirements:

### UR-1: Set up registration form

### UR-2: create registration endpoint in API

UR-3: create entries in database

## **Event Scheduling**

### 3.3.1 Description:

### Supervisors should be able to create and notify people of events so that renters and rentees can schedule parking spaces accordingly.

### 3.3.2 Stimulus/Response Sequences:

### When a supervisor wants to create an event, they can add it to the calendar and be presented with a form to enter in the applicable information such as date, time, location, etc.

### 3.3.3 Functional Requirements:

### ES-1: Set up event creation form

### ES-2: create event endpoint in API

ES-3: create event entries in database

## **Space Rental**

### 3.4.1 Description:

### Renters should be able to make their spot available during events that they choose so that others may reserve it.

### 3.4.2 Stimulus/Response Sequences:

### Parking space owners may select to rent out their space for particular events on the calendar by selecting the option on the event in the calendar or in the event info.

### 3.4.3 Functional Requirements:

### SR-1: Set up button to list rental space as available

### SR-2: List spot as available in database for that date/time

SR-3: API endpoint to manage space availability

## **Space Selection**

### 3.5.1 Description:

### Users should be able to select/deselect and reserve an available parking space during an event.

### 3.5.2 Stimulus/Response Sequences:

### Users select an event they would like a space for in the feed/calendar which is then reserved for them.

### 3.5.3 Functional Requirements:

### SS-1: Button to reserve space

### SS-2: reservation API endpoint

SS-3: update entry in DB

# **Non-functional Requirements**

All non-functional requirements have not been solidified, this is a constant W.I.P.

1. The software development model adopts some Agile development guidelines such as
   1. Welcome changing requirements, even late in development
   2. Business people and developers work together twice a week (in-class)
   3. Build projects around motivated individuals, trust them to get the job done
   4. Simplicity
   5. Regular intervals where the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly
2. The technologies used are Vue, Django, SQLite and GNU/Linux
3. The back-end APIs will be test with Postman
4. Members will communicate efficiently at least once a week in order to not have the rest of the group consult Dr. Watson for possible removal from the group. (unless specified to all member before-hand)

# **Future Features**

1. The web application will be hosted on AWS

**6. Glossary**

Agile development model: More of a set of guidelines, it includes a set of lightweight principles that are an alternative to other methodologies such as Waterfall and Spiral.

Big Blue Parking Genie: The web application that our group is developing. Its most used service will let customers rent a parking spot from an established parking lot or a local property owner who has agreed to let their parking spot be rented.

Supervisor: A person hired by the college to manage the parking spot agreements and the school event calendar.

Web application: Software that can be used via an app on a smartphone or through a website on a browser.