ParkMe: System Design Rev1

Daniel Agostinho
agostd - 001414323
Group 5

Michael Bitzos
bitzosm - 001405050
Group 5

Kathryn Brownlee
brownlks - 001408416
Group 5

Anthony Chang
changa7 - 001413615
Group 5

Ben Petkovsek
petkovb - 001417104
Group 5

March 20, 2019

Revision	Date
Rev0	December 28, 2018
Rev1	March 20, 2019

Contents

List of Figures

List of Tables

1 Introduction

The purpose of this document is to record the decomposition of both the hardware and software systems into their constituent components. This decomposition and explanation of the system components will aid in developing and implementing the system later in the project's life cycle.

2 Scope

The system design document contains information about the breakdown of the components of the system. Specifically, this document will include a context diagram relating to how our components behave, a component diagram that details the interactions between the components, several sections which document our system variables, constants, and the general overview of system behaviour. Next, each component will be broken down and described in terms of input/output, behaviour, timing constraints/performance requirements, and initialization procedures. Finally, this document will detail normal operating states as well as undesired events and how they will be handled.

3 Context Diagram

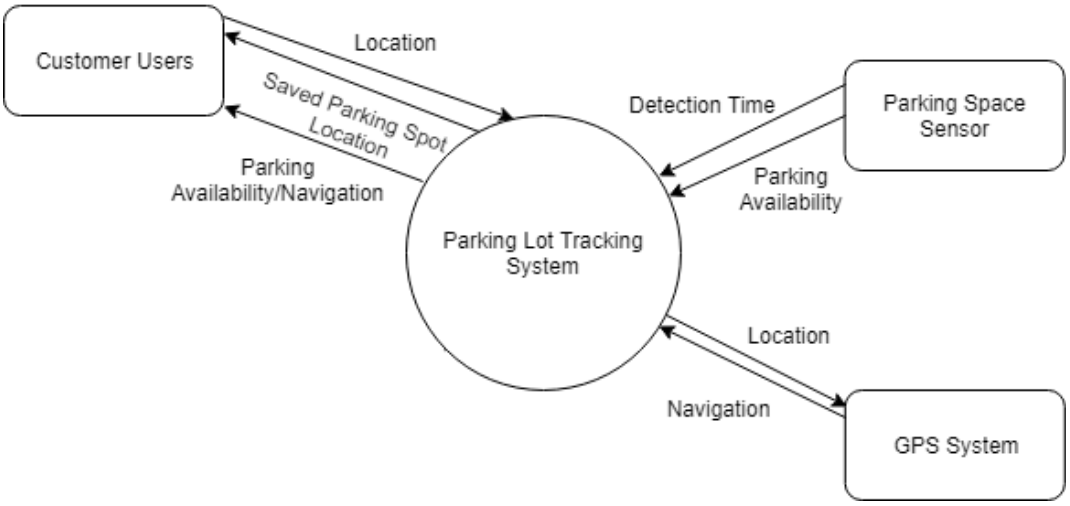

Figure 1: Context Diagram - Data Flow of the System.

4 Component Diagram

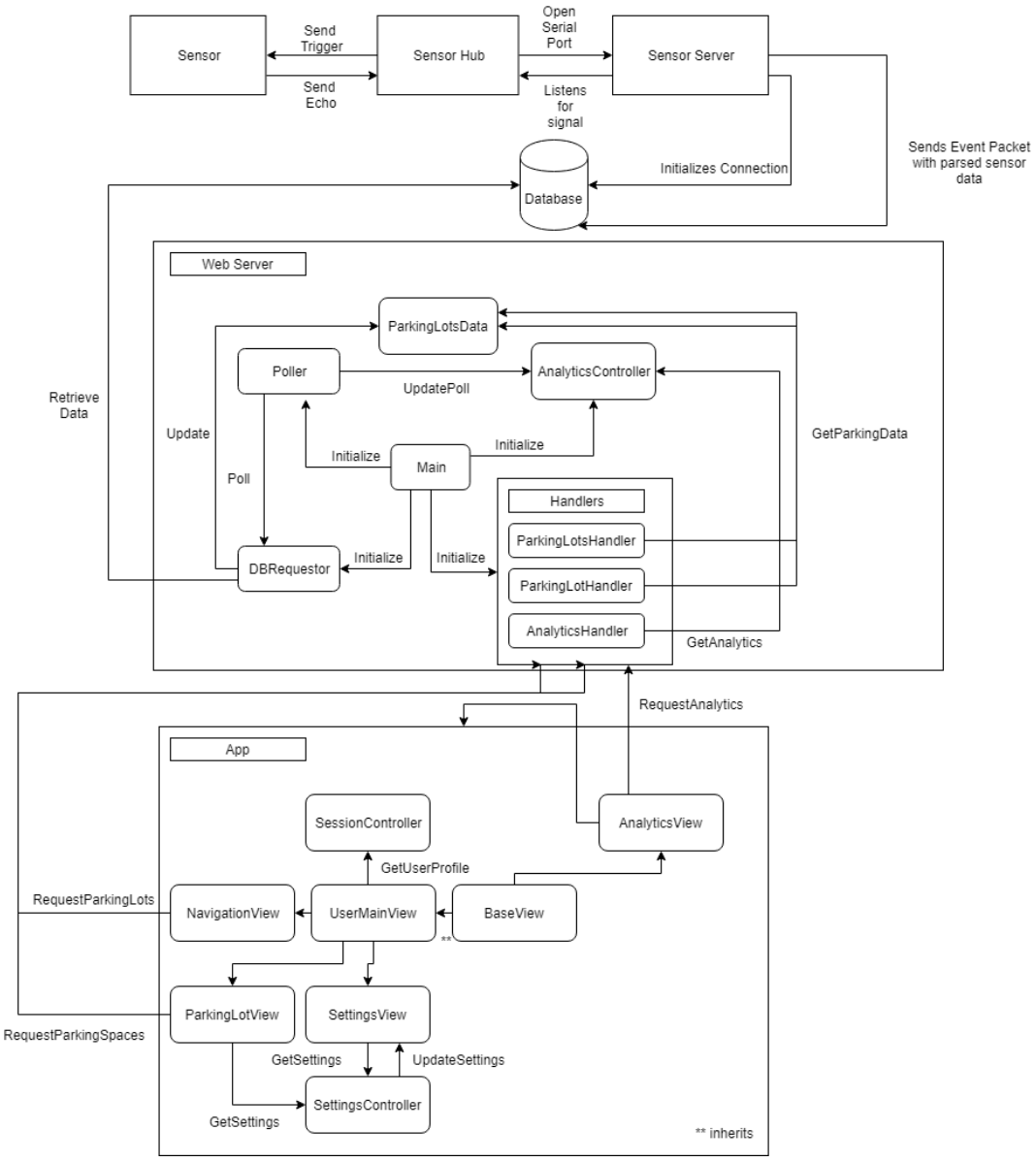

Figure 2: Component Diagram for ParkMe - Interactions Between Components.

5 Database Design

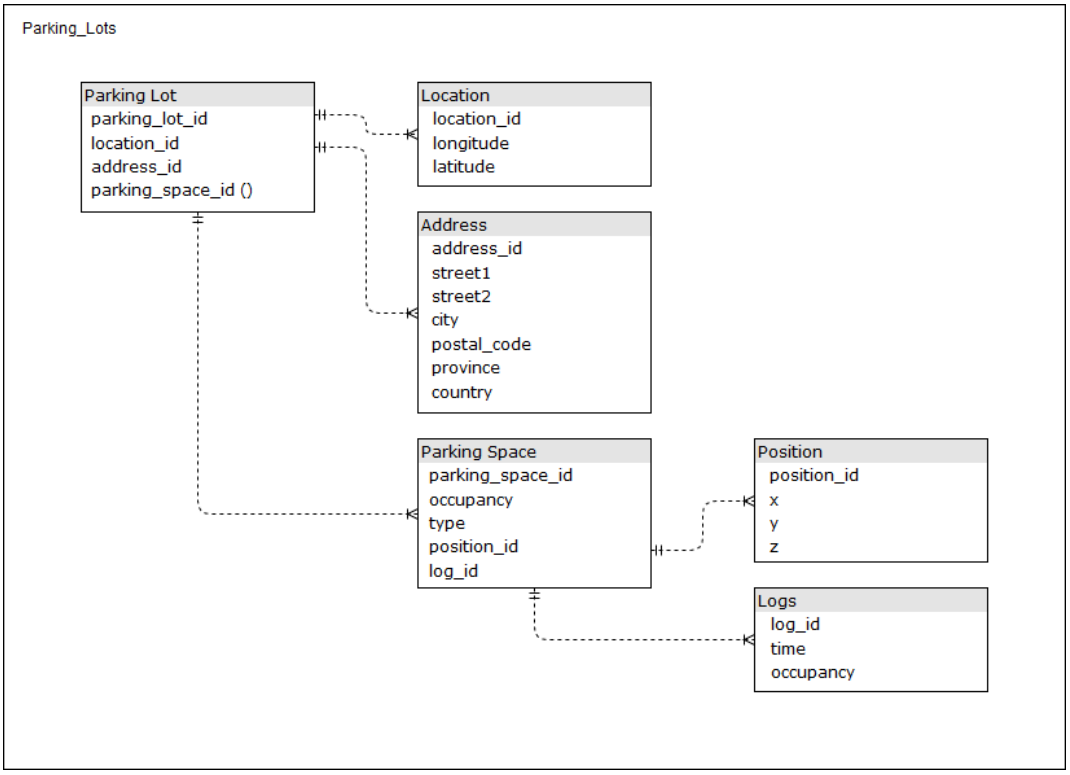

Figure 3: Database Schema - Design of the Database.

6 Monitored and Controlled Variables

ParkMe only contains two systems where it interacts with the environment around it: The mobile application where the user interacts with our system directly and the system responds back. The parking identifier is the second system where the users only interact with it and it does not directly interact with it.

6.1 Monitored Variables

Name	Type	Range	Units	Physical Interpretation
SPOT_OCCUPATION	Status	"true"/"false"	Unitless	If the Parking Spot Identifier has detected a vehicle or obstacle in the way.
USER_GPS_DATA	GPS	Longitude: -180 ,180 Latitude: -90, 90	Longitude, Latitude in Degrees	The GPS location data of the users mobile device.
USER_DEV_ROTATION	Vector2 (Number, Number)	Does not have a range	Unitless	The rotation of the mobile device relative to the normal of the device.
USER_APP_INPUT	Action	Does not have a range	Unitless	The user's varied input to the mobile application**

Table 1: Monitored Variables for ParkMe.

** - User inputs are described in the Software Requirements Specification document, this is just the general idea of inputs being registered from the environment from the user that all falls under potential application actions.

6.2 Controlled Variables

Name	Type	Range	Units	Physical Interpretation
USER_NAVIGATION	–	–	–	The navigation the application uses to guide the user to the parking lot.
SPOT_AVAILABILITY	Status	Occupied / Full	Unitless	If the parking spot is occupied.
SPOT_TYPE	Status	Accessible / N/A	Unitless	If the parking spot is an accessible type.
PARKING_LOT_STATUS	List of Statuses	List of True/False	Unitless	A mapping of the parking lot that the user requested with their availabilities.

Table 2: Controlled Variables for ParkMe.

** - User inputs are described in the Software Requirements Specification document, this is just the general idea of inputs being registered from the environment from the user that all falls under potential application actions.

7 Constants

- FLOAT - PARKING_SPOT_RESPONSE_TIME = 500ms
- FLOAT - PARKING_SPOT_EXPIRATION_TIME_WARNING = 10 minutes
- INT - MAX_CONCURRENT_USERS = 1000 users
- FLOAT - DETECTION_ERROR_RATE = 0.01%
- INT - NEARBY_DISTANCE = 20 km
- INT - MINIMUM_ALLOWED_CAPACITY = 150 MB

8 Behaviour Overview

The following is taken from the Software Requirement Specifications Rev. 1 document.

To solve the problems mentioned above, Park Me will implement several general systems. The first and most important of these is the ability to monitor the occupancy of parking spots in a given parking area through a series of physical sensors. This data will then be used by the mobile application to update the customer on the number of available spots in the lot, as well as the location of those spots. Specific data on information such as accessible parking, expectant mothers, paid parking...etc will also be displayed, and will utilize preferences set by the user.

Ideally, the application will also include functionality to navigate the user to a parking spot/row/area, but this integration will depend on the accuracy and availability of GPS systems and the complexity of those systems.

9 Component Breakdown

9.1 Sensor Component (Module-HW-S)

- **Input/Output:** Each sensor will have a voltage, ground, and trigger input, as well as a data output.
- **Behaviour Description:** Each sensor will take in power inputs, and a trigger input that instructs it to take in its sensor data. The collected data will then be output through the hardwired connection to the sensor hub.
- **Timing Constraints/Performance Requirements:** Each sensor must continually maintain a data connection and provide constant real-time information to the sensor hub such that its error detection rate is less than DETECTION_ERROR_RATE.
- **Initialization:** Not applicable for purely hardware peripherals.

9.2 Sensor Hub Component (Module-HW-A)

- **Input/Output:** The Sensor Hub Component will take in sensor data as input, and output final sensor information to the serial port.
- **Behaviour Description:** The sensor component will input data to the Sensor Hub Component, which will then clean the data and format a data packet with time, sensor reading, and sensor identifier. This formatted packet will be sent through serial port to the sensor server component.
- **Timing Constraints:** The sensor component must continually receive data in real time and push the formatted packets to the serial port immediately.

- **Initialization:** The sensor component will be initialized with the sensor pin assignments, sensor thresholds, as well as the serial port opening.

9.3 Sensor Server Component (Module-HW-P)

- **Input/Output:** The Sensor Server component will receive serial port data and will output server data updates with the new information.
- **Behaviour Description:** The component will take in serial port data which consists of formatted packets of sensor information, and will format the data further to fit the database schema. The component will then update the database entry for the sensor with the new data.
- **Timing Constraints:** The Sensor Server component will need to maintain a real-time connection to the serial port and format the data in real time as well. In addition to this, it will need to generate the database update requests and send them in real time.
- **Initialization:** The Sensor Server component will be initialized with a serial port connection, as well as a connection to the database server.

9.4 Database Component (Module-DB)

- **Input/Output:** The database component is a database instance that receives and sends data to the server.
- **Behaviour Description:** The database receives data inputs from the server and updates the document belonging to the parking lot collection. Each document contains information about a parking spots availability and status and sends this information back to the server when requested.
- **Timing Constraints:** The database component should contain the most up to date status for each parking spot. The database should also always be running and ready to receive updates in real time.
- **Initialization:** The database component will be initialized with a predetermined schema where each collection and document pairs will use the same data types.

9.5 Web Server

- **Input/Output:** The web server will receive API RESTful requests from the mobile application and will be responsible for returning RESTful responses back.

- **Behaviour Description:** The server will be responsible for supplying the parsed database data to the mobile application. This server will act as a middle ground between the mobile application and the database to ensure security and force a single point entry for the database. It will also be responsible for computing the real-time analytics as well as supplying this data to the mobile application.
- **Timing Constraints:** The server should supply data that is update to date with reference to the requirements laid out in the Software Requirement Specifications document Rev 1. Additionally the server should not ever stop running so that all API requests are successfully returned.
- **Initialization:** The server will be started up/initialized once to allow for its subcomponents to initialize as well.

Sub-Components:

9.5.1 Main (Module-SW-M)

Starts up the server and initializes all connected components.

9.5.2 Poller (Module-SW-P)

- **Input/Output:** Gets initialized from the Main component, and updates the AnalyticsController on poll event.
- **Behaviour Description:** Continually retrieves calls database to update local server data storage as well as updates the AnalyticsController with update to date data.
- **Timing Constraints:** The poller runs at a frequency determined the Software Requirement Specifications document.
- **Initialization:** Initializes the poller frequency based on requirement specifications.

9.5.3 DBRequestor (Module-SW-DB)

- **Input/Output:** Gets called from the Poller component, and updates the ParkingLotsData component with requested data from database.
- **Behaviour Description:** Requests the updated data from the database and sends to ParkingLotsData component to perform analytics on.
- **Timing Constraints:** The poller runs at a frequency determined the Software Requirement Specifications document.
- **Initialization:** Initializes a connection to the database.

9.5.4 AnalyticsController (Module-SW-AC)

- **Input/Output:** Retrieves data from the ParkingLotsData component and gets requested analytics from both ParkingLotsHandler and ParkingLotHander.
- **Behaviour Description:** Performs analytic computations on data storage held in ParkingLotsData.
- **Timing Constraints:** The AnalyticsController should update its analytics whenever the ParkingLotsData component is updated with new data.

9.5.5 ParkingLotsHandler (Module-SW-PLSH)

- **Input/Output:** Retrieves API requests from mobile application and returns the appropriated RESTful response.
- **Behaviour Description:** Handles all API that has to do with all the parking lots in the system.
- **Timing Constraints:** The requests should respond as fast as the Software Requirement Specifications document has specified.
- **Initialization:** The handler will be initialized and hook into API endpoints on start up.

9.5.6 ParkingLotHandler (Module-SW-PLH)

- **Input/Output:** Retrieves API requests from mobile application and returns the appropriated RESTful response.
- **Behaviour Description:** Handles all API that has to do with a specific parking lot.
- **Timing Constraints:** The requests should respond as fast as the Software Requirement Specifications document has specified.
- **Initialization:** The handler will be initialized and hook into API endpoints on start up.

9.5.7 AnalyticsHandler(Module-SW-AH)

- **Input/Output:** Retrieves API requests from mobile application and returns the appropriated RESTful response.
- **Behaviour Description:** Handles all API that has to do with any analytics provided in the system.

- **Timing Constraints:** The requests should respond as fast as the Software Requirement Specifications document has specified.
- **Initialization:** The handler will be initialized and hook into API endpoints on start up.

9.5.8 ParkingLotsData (Module-SW-PLD)

- **Input/Output:** Gets updated by the DBRequestor component and provides parsed data to the AnalyticsController as well as the ParkingLotHandler and ParkingLotsHandler.
- **Behaviour Description:** Handles all manipulation (getters/setters) of the data copied from the database into the server.
- **Timing Constraints:** The requests should respond as fast as the Software Requirement Specifications document has specified.
- **Initialization:** It will be initialized on startup and continually updated with the polling DBRequestor component.

9.6 App

- The app will be responsible for interpreting the parsed database data and displaying it in a way understandable and informative to users. This app will act as an interface between users and the server to aid in information retrieval and understanding. It will also be responsible for displaying the real-time analytics, navigation, and parking lot mappings to the user so they can interactively but abstractedly send requests to the server.
- **Timing Constraints:** The app should supply data that is update to date with reference to the requirements laid out in the Requirements Document Rev 1.
- **Initialization:** The app will be started up/initialized upon each new session (app has been fully exited previously) to allow for its session controller and base component to initialize as well.

Sub-Components:

9.6.1 Base (Module-SW-S-BV)

- **Input/Output:** Starts up application initialization.

- **Behaviour Description:** Initializes base application function. Displays landing page for application, and user mode selection.
- **Timing Constraints:** The startup should respond as fast as the Software Requirement Specifications document has specified.
- **Initialization:** Initializes a connection with the server.

9.6.2 UserMain (Module-SW-S-UMV)

- **Input/Output:** Gets initialized from BaseView. Initializes and updates SessionController and SettingController, updates interface for all functionality available to the user when in UserMain mode.
- **Behaviour Description:** Displays functionality of user mode as described in the Software Requirement Specifications document. Allows user to input a location and returns nearby parking lots. Makes requests to the server for possible parking lots.
- **Timing Constraints:** The requests should respond as fast as the Software Requirement Specifications document has specified.
- **Initialization:** Initializes the SettingsController and SessionController.

9.6.3 Settings (Module-SW-S-SV)

- **Input/Output:** Gets initialized by request from UserMainView. Sends settings API requests.
- **Behaviour Description:** Handles all settings API requests (getters/setters) to modify current user settings and preferences.
- **Timing Constraints:** The requests should respond as fast as the Software Requirement Specifications document has specified.
- **Initialization:** Hooks into the session and settings controllers.

9.6.4 SessionController (Module-SW-S-SSC)

- **Input/Output:** Gets initialized from the UserMainView. Allows access and change of current user parking status, and location.
- **Behaviour Description:** Handles user parking lot space state for later retrieval and navigation.
- **Timing Constraints:** The requests should respond as fast as the Software Requirement Specifications document has specified.

- **Initialization:** Initialized on UserMainView startup and hooks into UserMainMenu.

9.6.5 SettingsController (Module-SW-S-SEC)

- **Input/Output:** Gets initialized from the UserMainView. Allows access and change of current user preferences for spot location, type, etc.
- **Behaviour Description:** Handles user preferences for the session, allowing for modification and retrieval via API requests.
- **Timing Constraints:** The requests should respond as fast as the Software Requirement Specifications document has specified.
- **Initialization:** The controller will initialize and hooks into SettingsView and UserMainMenu.

9.6.6 NavigationView (Module-SW-S-NV)

- **Input/Output:** Gets initialized from UserMainMenu. Sends maps API request constructed from UserMainMenu, once user has chosen a parking lot location. Displays the returned navigation info, alongside verbal instructions.
- **Behaviour Description:** Provides navigation instructions to the user's previously selected parking lot, both visually and verbally, using pre-existing APIs.
- **Timing Constraints:** The poller runs at a frequency determined the Software Requirement Specifications document.

9.6.7 ParkingLotView (Module-SW-S-PLV)

- **Input/Output:** Gets initialized from UserMainMenu. Sends/receives request to server to get ParkingLot info, once user has chosen a parking lot location from the UserMainMenu. Requests preferences from SettingsController and sends spot info to UserMain.
- **Behaviour Description:** Suggests available parking spots based on UserSettings and displays location and approximate user location, for navigation, once the user has reached the parking lot.
- **Timing Constraints:** The requests should respond as fast as the Software Requirement Specifications document has specified.

9.6.8 AnalyticsView (Module-SW-S-AV)

- **Input/Output:** Gets initialized from the baseView. Displays graph data via API usage, sends requests to the server.
- **Behaviour Description:** Displays graphs and other analytics as available.Makes requests to the server for required parsed database data then displays it as specified.
- **Timing Constraints:** The requests should respond as fast as the Software Requirement Specifications document has specified.

10 Normal Operation

The normal operation sequence of the system is as follows.

- The sensors take in data from parking spaces.
- The sensor component receives the sensor data, formats it, and then sends it through the serial port.
- The Sensor Server component continually reads the serial port for update packets, formats them to follow database schema, and then sends the update request to the database.
- The database updates the respective tables.
- The mobile application pulls database data and provides an interface for users.

11 Undesired Event Handling

The following is taken from the Software Requirement Specifications Rev. 1 document.

In the event of a undesired event that is not a result of a previously stated assumption, the system will be designed in such a way that it will self-correct any discrepancies. This means that any desynchronization issues between the application and the servers will be fixed with the use of multiple users supplying data to the database. Additionally this will be solved by having periodic updates so that the most recent data is used. Finally, in the case of an undesired event that is not covered, the use of error handling and error messages will be utilized to prevent any confusion for the user and preventing any disastrous errors within the system.