***SpotMe*: Parking Space Finder App**
User-Requirements Documentation

1. INTRODUCTION

SpotMe is a mobile application system intended for use by the students, staff, and faculty of CSU Fresno. It allows its users to obtain live information regarding the availability of parking spots within the various lots around campus as well as auxiliary information such as permit requirements, and expected times of peak congestion while also giving users access to quality of life tools such as a current parked location marker and recommendations for planning their weekly parking habits. The creation of this document succeeds a stakeholder interview with a company representative; the features, constraints, and priorities outlined herein intend to provide specifications and address concerns elicited from this interview.

2. USER REQUIREMENTS

2.1: Features

2.1.1: Graphical Congestion and Occupancy Overview

- Users shall be able to, at a glance, see the parking lots around campus and an indication of their levels of congestion/occupation in the form of a color-coded marker. Additionally, users shall be able to view a color-coded occupancy status of individual spaces within each lot.

2.1.2: Per-spot Sensor Data Ingest

- The system shall eventually make use of hardware sensors installed in individual parking spaces to detect the presence of a vehicle and, subsequently, occupation status of each spot in real-time.
- NOTE: The acquisition and usage of hardware sensors to provide this data is an eventuality. For the purpose of furnishing a proof-of-concept, this data will be provided through virtual simulation.

2.1.3: Saving Parked Location

- Users shall be able to save the location of their parked vehicle which will remain in place as a persistent marker on their map view for the purpose of physically locating the car in a large lot. The location of this marker should be quickly retrievable, updateable, and removable.

2.1.4: User Accounts

- The app shall allow users to create an account with a simple username and a secure password.
- Accounts shall store only necessary information that will assist application functionality such as a user's parking schedule, which permits they possess, and which lots they prefer for a given time slot in their schedule.
- Pertinent information from these accounts as well as server-side data shall be used to augment the experience of the user, such as warning about permit conflicts in preferred parking lots/vacancy reservations and recommending parking lot preferences based on historical congestion data.

2.1.5: "Soft" Reservation

- All users shall be able to select a vacant space they wish to occupy ahead of time. The spot will be marked as reserved to other users of the app, preventing them from reserving it for themselves.
- Navigation to the reserved space will be provided to the reserver. If the current space becomes physically occupied en route, navigation should redirect to the closest parking space factoring in the user's known permits.
- If multiple users contest for the same spot simultaneously (i.e. before a user's app receives notice that the spot has been reserved by another user), the system shall first determine who to prioritize based on whose request arrives first. The other user(s) shall be notified of this conflict and prompted to reserve another space.

2.2: Integration Requirements

2.2.1: Google Cloud Computing Suite

- The Google Cloud Suite shall be used as a means of hosting our application to enable an always-on service.
- This service shall handle back-end computation, such as congestion levels and formulating recommendations; it shall also handle data management for user accounts.

2.2.2: Google Maps SDK API

- The Google Maps SDK API shall be used as the primary method of displaying the map to a user with built-in overlay customization. Physical interactions with the map itself shall be handled through this API.
- This API's Navigation SDK shall additionally be used to provide directions to the user in the necessary use cases, such as through the soft reservation feature (**2.1.5**).

2.3: User Interface

2.3.1: Primary Display

- The main UI shall consist of a Google Maps view of campus. Markers and traffic shall be overlaid atop with a color-coded indication of congestion; tapping the markers shall display additional information for a given lot such as permit requirements and a histogram of typical occupancy over time.
- Static widgets on the main display shall present the minimal, critical interactions for users: adjusting the map orientation, saving or locating their vehicle's location, updating map data, and an "Account" section detailed in **2.3.2**.

2.3.2: Account Menu

- The Account Menu shall consist of a simple popup navigation menu that presents the users with the options to register, log in, add permits, sign out, and create a weekly parking schedule as detailed in **2.3.3**.
- The option to add permits shall provide the user a checkbox list of permits; users need only check the ones which they use.

2.3.3: Weekly Parking Scheduler

- The weekly parking scheduler shall consist of a tabular timesheet. The columns shall correspond to days of the week, and the rows shall correspond to 30-minute time slots.
- Users can create a weekly schedule by long pressing and swiping across the empty slots they plan to park during to create a time block. Tapping a user-created time block shall prompt the user to select a lot they wish to occupy during the time block; long pressing this block shall allow users to resize the block or delete it.
- Advisory color-coding and selection warnings should notify the user if the lot they prefer during a time block has a typically high level of congestion or requires a permit they do not have listed in their account.

2.4: Constraints

2.4.1: API Usage Budget

- Due to the use of the Google Cloud Computing and Google Maps APIs, there may be an associated cost for our application if we require the use of many requests to these APIs. This can be mitigated with conservative API call handling and optimization efforts (see **4. Priorities and Milestones**).

2.4.2: Prototype Timeline

- The timeline anticipated for this application slates the demonstration of a prototype before the end of the year. A suggested timeline with milestones for the prototype is specified in section **4. Priorities and Milestones**.

3. USE CASES

3.1: Locating Empty Spots

- **Description:** Users of the app can find and navigate to an available parking spot in real-time.
- **Actor(s):** Student, Staff, or Faculty
- **Steps:**
  1. User selects a desired vacant parking spot in a lot of their choosing.
  2. User navigates through the parking lot using an in-app Google Maps overview.
  3. User successfully parks.
- **Expected Outcome:** User successfully navigates to an available parking spot and parks.

3.2: Spreading Parking Lot Usage

- **Description:** App users can quickly determine highly congested lots and navigate to alternate lots.
- **Actor(s):** Student, Staff, or Faculty
- **Steps:** 1. User views the color-coded markers in the map UI.
  2. User determines congestion based on color-coded markers.
  3. User travels to an area of least congestion.
- **Expected Outcome:** Occupation is balanced within the different parking lots.

3.3: Weekly Parking Scheduler Recommendations

- **Description:** Users of the app can create a weekly schedule and be served suggestions to alternate spaces if their initial choices may be inoptimal
- **Actor(s):** Student, Staff, or Faculty
- **Steps:** 1. User selects time slots on the timesheet indicating the time they require parking.
  2. User selects the desired parking lot for a given time block.
  3. The app determines that their desired lots will be congested at the times they specify.
  4. User can choose optional alternatives to tweak the spaces they choose in their schedule.
- **Expected Outcome:** User creates a schedule that potentially minimizes the time spent searching for vacancies.

4. PRIORITIES AND MILESTONES

4.1: Priorities (in Order of Precedence)

4.1.1 User Interface

- Design and develop a simple, intuitive UI framework to serve as the backbone for further development. Prioritize ease of use for quick information retrieval and easy navigation.

4.1.2 Further API Integration

- Develop further API integrations to achieve core functionality.
- Investigate methods to get accurate wireframes or layouts of individual parking spaces (via API, mapping tools, or manually created layouts).

4.1.3 Initial Testing and Validation

- Within our suggested timeframe, focus implementation on two parking lots: Recreation Center and Business East. This will allow for streamlined testing and optimization in a controlled environment before expanding to additional lots.
- Conduct real-world testing with the team's personal vehicles. Collect feedback on accuracy, performance, and user experience during this phase.

4.1.4 Performance and Resource Optimization

- After validating basic functionality, ensure our system can handle a high volume of requests, especially during peak parking hours.

4.2: Milestones

4.2.1: **Base Functionality**

- Expected Timeframe: 1 Month
- Implement high-priority features to achieve core application functionality for use in testing.

4.2.2: Further Feature Development, Prototype Demonstration

- Expected Timeframe: 1 ½ Months after achieving base functionality
- Further develop lower priority feature set and tweak core functionality based on internal testing results; prepare a prototype for demonstration.