***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 real-time information regarding the availability of parking spots within the various lots around campus as well as auxiliary information such as traffic congestion, permit requirements, and expected times of peak congestion. The creation of this document succeeds a stakeholder interview with a company representative; the features, constraints, and priorities outlined herein intend to provide specification and address concerns elicited from this interview.

2. USER REQUIREMENTS

2.1: Features

2.1.1: Graphical Traffic 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. Per-lane traffic levels should be visible to assist a user in navigating within a lot.

2.1.2: Vacancy Querying

- Users shall be able to identify vacant spaces in a lot of their choosing.
- Any identified vacancies shall be displayed in-app via a focused Google Maps overhead view.

2.1.3: Account Creation

- 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.

2.1.4: "Soft" Reservation

- All users shall be able to select a vacant space they wish to occupy ahead of time.
- If multiple users contest for the same spot, priority should be given to whomever registers first, or, in cases where conflicting registrations are made simultaneously, whoever is anticipated to arrive the earliest while others are immediately notified about the conflict.
- Users with an account shall be able to additionally register a weekly schedule of their anticipated parking times and preferred parking lots.
- This data together shall be used to identify possible times of peak congestion and should serve users recommendations for better parking.

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 API

- The Google Maps API shall be used as the primary method of displaying map information to a user through built-in overlays.
- This API shall additionally be used to gather anonymous, localized traffic data to assist in congestion calculation and navigation.

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: registering a soft reservation, 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 tapping the slots they wish to occupy or free up. Adjacent used time slots shall be combined into a time block, and long pressing this time block shall prompt the user to select a lot they wish to occupy during the time block.

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 an available parking spot in real-time and navigate to it.
- **Actor(s):** Student, Staff, or Faculty
- **Steps:** 1. User selects desired vacant parking spot in a lot of their choosing.
  2. User navigates through parking lot using 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:** Users of the app 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 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 timesheet indicating the time they require parking.
  2. User selects desired parking lot for a given time block.
  3. App determines that their desired lots will be congested at the times they specify.
  4. User is served optional alternatives to tweak the spaces they choose in their schedule.
- **Expected Outcome:** User creates a weekly schedule that minimizes traffic issues they may face.

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 to 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.