FLEET OF TAXI SERVICE MANAGEMENT SYSTEM - SOFTWARE DESIGN SPECIFICATION

Danielle Winter (563795), Frederick Nieuwoudt (386372), Stephen Friedman (360938) & Sello Molele (0604606X)

School of Electrical & Information Engineering, University of the Witwatersrand, Private Bag 3, 2050, Johannesburg, South Africa

1. INTRODUCTION

1.1 Purpose of this Document

This Software Design Specification (SDS) details the design for the WitsCABS application. The back- and front-end interfaces are described separately. The front-end and back-end integration is also described. The remainder of Section 1 details the project scope and abbreviations and definitions used in the remainder of the SDS. Section 2 addresses the basic system architecture and Section 3 gives a more detailed description of how the system will be implemented.

1.2 Project Scope

The taxi fleet management system is a tool that provides a service where a user can request a taxi and the system will dispatch the closest available driver.

The system back-end will be required to store information about drivers, customers, and the current pricing scheme. The system will keep historical data regarding trips that have been taken for auditing and dispute resolution.

The system front end will be presented in the form of a web interface. This will provide a means for Dispatch Control Centre agents to input new customer's details and for drivers to manage their current job requests. The web interface will also provide a quoted price for the client.

For the purposes of this project, the entire system functionality is described in the SDS. However, only a few modules and components of the code are implemented as a prototype of the system in order to show basic functionality.

1.3 Definitions

1.3.1 Definitions

- WitsCABS: The name of the software application including front- and back-ends
- Driver: The taxi-cab drivers employed by the WitsCABS company to pick-up and deliver customers
- Customer: Clients of the company who request lifts

1.3.2 Abbreviations

- DCC: Dispatch Control Centre

2. SYSTEM ARCHITECTURE

2.1 Front-end Views

The front-end of the WitsCABS application is presented as two separate views in an HTML web-application. The front end is coded in angular.js using Twitter Bootstrap for a unified visual appearance and pop-up functionality. The respective interfaces are the DCC view and the Driver view. These views are linked through a common log-in screen view where the respective users select and log in to their interface.

2.1.1 DCC View The DCC view includes panels for a new customer form and an active job list. The "new customer form" panel has text boxes for the customer's name, current location, destination and cellphone number. A submit button allows the DCC agent to submit the entry to the back-end database, at which point, the distance and price for the job is calculated. The new job will be added to the "active jobs" panel in the DCC interface and the DCC will be able to see the price calculated for the customer.

2.1.2 Driver View The driver view allows the driver to see their assigned job, the customer name and contact details and their travel destination. In addition to this, two buttons are presented to the driver - one for "On the Way" when they leave the taxi rank and one for "Job Complete" when the customer has been delivered.

2.1.3 Log-in View The log-in page gives the user the ability to select whether they are a driver or a DCC agent and allows them to log in to their respective view. The log-in page should prompt users for a username and password and their entry should be authenticated through the back-end.

2.2 Back-end Implementation

3. DETAILED ARCHITECTURE AND FEATURES

3.1 Front-end Architecture

The front-end architecture is detailed below for the angular.js and HTML files. The front-end and back-end communicate through button pushes - firstly with the log-in page user authentication and then in the respective views.

3.1.1 angular.js File

- angular module - WitsCABS
- angular controller - JobController
- customer directive - jobs
- array of customers
  - customer name (string)
  - customer destination (string)
  - customer current location (string)
  - customer phone number (string)
  - distance (integer)
  - price (integer)
  - job active (boolean)
  - driver assigned (boolean)

3.1.2 HTML File: When DCC agents submit a new customer in the form, the back-end database is updated and a taxi-driver is assigned to the new job. The driver interface will be updated. Drivers can state when a job is complete, at which point, the database will once again be updated and the respective views refreshed. Upon log-in details being entered, authentication is implemented through the back-end framework before users can be redirected to their respective pages.

- DCC View
  - Headings

  - Active job panel
    * active customers list
    * driver assigned (boolean)
    * customer location, destination and phone number (paragraphs under each customer)

  - New customer form panel
    * customer name field
    * customer destination field
    * customer current location field
    * customer phone number field

    * submit button

- Driver View
  - Headings

  - Assigned job panel
    * customer details
    * "On the way" button
    * "Job complete" button

- Log-in Page
  - Log-in modal
    * driver log-in button - redirects to driver interface
    * DCC log-in button - redirects to DCC interface
    * close button

3.2 Back-end Architecture

*for Stephen and Sello to describe