FLEET OF TAXI SERVICE MANAGEMENT SYSTEM - LABORATORY EXERCISE 2: SOFTWARE REQUIREMENTS 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

A software database system for a Taxi Fleet Management is proposed in this Software Requirements Specification. The objective of this document is to describe in detail the user requirements as well as the specifics of what the system requires and how it will be developed.

The aim of the system is to respond to a customer request for transport to a location. It will then find an available driver for the job and deploy their taxi for service. It will also allow for users to manage profiles and administrators to manage diver profiles.

1.1 Definitions

1.1.1 Customer: The person who falls part of the target market of the proposed product. This is the person who is utilizing the services provided by the system.

1.1.2 Dispatch Agent: The person, or person who are required to operate the dispatch control center. They are responsible for answering customer phone calls and capturing data.

1.2 Scope

The taxi fleet management system is a tool that provides a service where a customer can request a taxi via a phone call to the dispatch control center and an appropriate driver will be routed to them.

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 dispatching agent to input the customer's current location and destination. The web interface must also provide a means for the taxi driver to access information about the job that has been allocated to them.

2. Overall Project Description

2.1 User Requirements

A brief description of the different users and their basic requirements is given. Furthermore, potential users were surveyed for additional requirements. The user requirements are specified based on the MOSCOW and FURPS methodologies [1].

2.1.1 Basic User Requirements

- Passengers
  - Lift Requests
- Drivers
  - Receive allocated job card
  - View relevant customer details
  - Respond to the job card
- Dispatch Control Center Agents
  - Capture customer details
  - View active job cards

2.1.2 Detailed User Requirements

Passenger

- Lift Request
  - Passengers should be able to request a taxi driver through making a phone call to the dispatch control center.
  - Passengers must provide their current location and their intended destination.

Driver

- Receiving Allocated Job Card
  - Only the assigned customer must be displayed. This prioritized by the distance from the driver.

- Viewing relevant customer details
  - The driver must be able to view the customers current location.
  - The client's destination must be displayed.
  - The client's contact details must be displayed.
  - The calculated fare for the trip must be available.

- Responding to allocated job card
  - In response to an allocated job card a driver must be able to indicate when he is on the way to the customer.
  - A driver must be able to indicate when the job has been completed.

- Capturing customer details.
  - A dispatch control center agent must be able to capture client data by inputting it into the provide web interface.
  - Agents must be able to view information on currently active jobs. including currently assigned driver and the job specifics.

2.2 Testing Plan

To achieve a quality product the test driven development(TDD) paradigm will be adopted. TDD calls for the writing of tests before writing the code to be used in the implementation. This encourages simple modular code.

The TDD workflow follows the sequence of writing a test, validating that the new test fails, writing the application code, running all the tests together, and then refactoring. This sequence allows for the project to move from getting code working to refactoring the code into simple understandable modules.

2.3 Implementation Plan

Following the Scrum SDLC methodology, will provide shippable incrementally improved products. A minimum of required features will be needed for a product that can meet the project scope as defined above. Additional functionality can be incrementally provided over the life span of the product.

2.4 Constraints

2.5 Assumptions and Dependencies

REFERENCES