Software Requirements Specification

for

EZ Tag System

Version 1.0 approved

Prepared by Isabel Almaguer, Carrie Dumit, Jason Jensen

<organization>

February 24, 2016

Table of Contents

Table of Contents ii

Revision History ii

1. Introduction 1

1.1 Purpose 1

1.2 Document Conventions 1

1.3 Intended Audience and Reading Suggestions 1

1.4 Product Scope 1

1.5 References 1

2. Overall Description 2

2.1 Product Perspective 2

2.2 Product Functions 2

2.3 User Classes and Characteristics 2

2.4 Operating Environment 2

2.5 Design and Implementation Constraints 2

2.6 User Documentation 2

2.7 Assumptions and Dependencies 3

3. External Interface Requirements 3

3.1 User Interfaces 3

3.2 Hardware Interfaces 3

3.3 Software Interfaces 3

3.4 Communications Interfaces 3

4. System Features 4

4.1 System Feature 1 4

4.2 System Feature 2 (and so on) 4

5. Other Nonfunctional Requirements 4

5.1 Performance Requirements 4

5.2 Safety Requirements 5

5.3 Security Requirements 5

5.4 Software Quality Attributes 5

5.5 Business Rules 5

6. Other Requirements 5

Appendix A: Glossary 5

Appendix B: Analysis Models 5

Appendix C: To Be Determined List 6

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
|  |  |  |  |
|  |  |  |  |

# Introduction

## Purpose

Our project is going to be based on the EZ tag system and its functionalities. The program that we create will assume, for simplicity, that an EZ tag account has already been created. Our program structure will contain classes for each phase of the process. The main classes that we have in mind are: a central management class, a class for lanes, and a class for accounts. The central management class will process information from the camera, such as if an EZ tag is present, the EZ tag ID, and the vehicle entrance and exit. The lanes class will set the direction and price for lanes based on time of day or other special events. The accounts class will contain the account and vehicle information, update information on accounts, determine validity of account, and generate transaction numbers.

## Document Conventions

This document uses the following conventions:

<Describe any standards or typographical conventions that were followed when writing this SRS, such as fonts or highlighting that have special significance. For example, state whether priorities for higher-level requirements are assumed to be inherited by detailed requirements, or whether every requirement statement is to have its own priority.>

## Intended Audience and Reading Suggestions

This document provides design details for our Toll System that may be read from front to back for a complete understanding of the project. Our documentation is intended to be a detailed guide for all readers that are interested in the back end development of an EZ Tag System, and all people involved in the primary functional areas of the system. The overall document is separated in sections, and designed to be read by specific topic to provide rapid access to material as needed. Toll management personnel and the developers in charge of maintenance of the product can read the specific sections that interest them as well as any reader that chooses to read only on areas of interest.

## Product Scope

This project is will be based on the EZ tag system and its functionalities. More specifically, we will be looking at the back end development of said system, which will include lane operations, cash management, toll schedule and central toll management systems. The project was conceived and will be developed by a team of three throughout the spring semester.

## References

Absolute Java 6th Edition by Walter Savitch.

https://www.hctra.org/ – Harris Country Toll Road Authority (EZ TAG) website.

Antonio Dumit – former employee who was in charge of hardware maintenance.

# Overall Description

## Product Perspective

The overall system architecture will be based on the functionality of the Houston tolling system. It will process the information of vehicles gathered from the suite of sensors on the toll lanes and allow users to log in to manage their accounts. The lane subsystem will consist of a combination of methods that will allow for the collection of revenue, and file transaction. And the whole system will be managed by a central system manager that will control the interaction of each subsystem. There will

## 

## Product Functions

* Create EZ Tag account.
* Add/Edit vehicles on EZ Tag account
* Add/Edit payment information.
* Apply charges/take payments..

## User Classes and Characteristics

Account, Central Processor, Lane Controller, Transactions.

Account: Name.

Address.

Vehicles.

Payment information.

Process transactions:

Process fees

File transaction

Create transaction number

Central Processor: Processes tag record.

Processes input data

Request status of account.

Verify valid tag.

\*If no tag is associated the toll can still be collected by license plate.

Lane: Determines fare

Lane status (Open/Closed/HOV/Toll/Other)

## Operating Environment

The end product will be run on a Linux server, however, because this will be created in Java, it will be a cross platform product.

## Design and Implementation Constraints

Major constraint is limited time given to complete project.

Each individual part of this project has a different level and understanding of Java.

The project requires us to put in some level of security measures that must be furthered researched.

## User Documentation

EZ Tag provides an online FAQ section that we will be using for the User Documentations. They can be found on: <https://www.hctra.org/about_faq/?CSRT=1683225753507280354>

## Assumptions and Dependencies

Everyone using the product has an active internet connection.

Everyone using the product has an operating system which can support it.

# External Interface Requirements

## User Interfaces

Our project will use a similar user interface to the existing EZ Tag log in screen. Both users and employees will be prompted to enter a username and password. There will be an option to have that information reset if it has been forgotten. If a user does not have an account, then they will be able to register a new account.

## Hardware Interfaces

<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>

## Software Interfaces

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>

## Communications Interfaces

<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>

# System Features

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

## System Feature 1

<Don’t really say “System Feature 1.” State the feature name in just a few words.>

4.1.1 Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

4.1.2 Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

4.1.3 Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use “TBD” as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1:

REQ-2:

## System Feature 2 (and so on)

# Other Nonfunctional Requirements

## Performance Requirements

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>

## Safety Requirements

Due to the nature of the information that will be on a user’s account, identity theft is a possibility. In order to combat this we are requiring users to log into their account before any information is provided.

For employees, an administrator will create their logins since they will have access to all accounts in the system.

## Security Requirements

To authenticate a user and allow access to the system a username and password will be required. A user with a regular account will only have access to their own account so will be able to create and modify their login information at will. An employee will have access to all accounts so an administrator will create an account and provide them with their username and a temporary password that will need to be changed.

## Software Quality Attributes

<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>

## Business Rules

No member is restricted to any specific role, however all actions/changes regarding the project must be agreed upon by each member of the team before being implemented.

# Other Requirements

Appendix A: Glossary

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>

EZ Tag: Name of the Harris County Toll Road Authority system

Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

Appendix C: To Be Determined List

<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>