**Patient Information System**

Software Requirement Elicitation, Modelling and Design Document.

Client: Traffic Accident Reporting System (TARS)

Vendor: Pierce Kinghorn 12091381

Due Date: Week6 Friday 2020 11.45pm

**TABLE OF CONTENTS**

[1 SYSTEM REQUIREMENTS 2](#_Toc36470661)

[1.1 Functional requirements 2](#_Toc36470662)

[1.2 Non-Functional requirements 2](#_Toc36470663)

[2 SYSTEM ARCHITECTURE 3](#_Toc36470664)

[2.1 Architectural Design 3](#_Toc36470665)

[2.2 Use Case Diagram 4](#_Toc36470666)

[2.3 Class Diagram 5](#_Toc36470667)

[2.4 Sequence Diagrams 6](#_Toc36470668)

[3 DATA DESIGN 8](#_Toc36470669)

[4 INTERFACE DESIGN 8](#_Toc36470670)

[5 REQUIREMENTS MATRIX 12](#_Toc36470671)

# SYSTEM REQUIREMENTS

## Functional requirements

The functional requirements of the system will be split into four sections, these being interface requirements, business requirements, compliance requirements, and security requirements. The case study described involves all for sections and will be expanded on in further detail below.

For Interface requirements, users must be able to view, browse and manipulate data stored within the system. This pertains to both recorded QLD road accidents and vehicle and vehicle owner information. User interactions with the system is to be performed through a Java desktop application. The application must be able to run on a windows operating system. The system will be able to store information regarding vehicles and vehicle accidents. The information stored for vehicles will consist of id/plate number, model, year, owner name, owner address, and owner phone number. The information stored for vehicle accidents will consist of an autogenerated id, location, comments, and vehicle id, the vehicle id section is not limited to a single vehicle. When adding a vehicle to the system, the user will be able to check if the vehicle currently exists inside the database and in the case it does, offer the user to update the information. The application must auto-connect to the database on start-up.

To summarise.

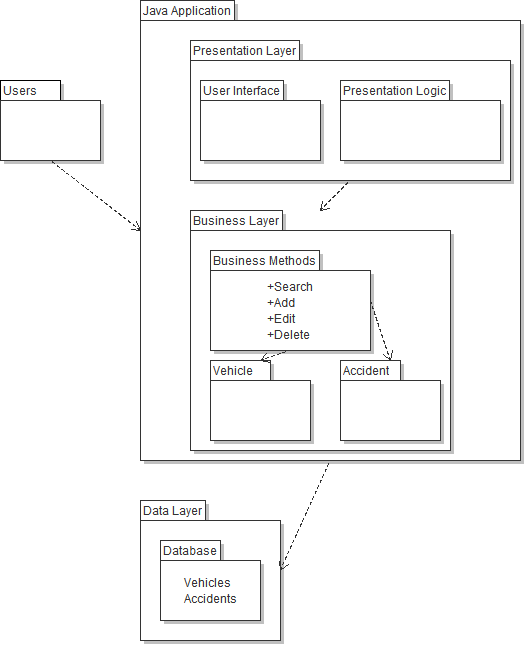
* Users must be able to view, browse and manipulate stored data.
* Users will interact with the system through a Java desktop application capable of running on windows operating systems.
* The System must be able to store information regarding vehicles and vehicle accidents.
* The System must allow ease of use of users allowing faster and more efficient work.
* The application must auto-connect to the database on start-up.

## Non-Functional requirements

The non-functional requirements of the system are listed as, performance, reliability, data integrity, usability, maintainability, scalability, availability, and manageability. Although there was no mention of any type of security for the system a recommendation can be made for the inclusion of a login type security for the system in order to ensure the verity of the data stored.

# SYSTEM ARCHITECTURE

## Architectural Design

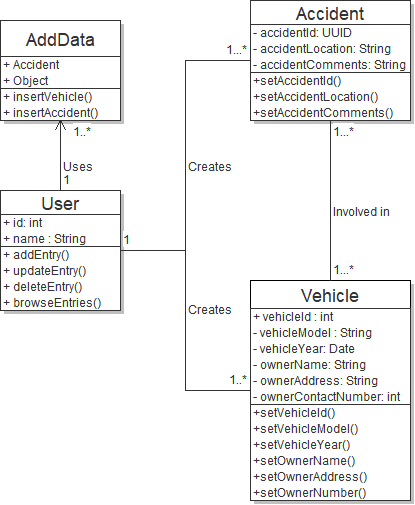


## Use Case Diagram



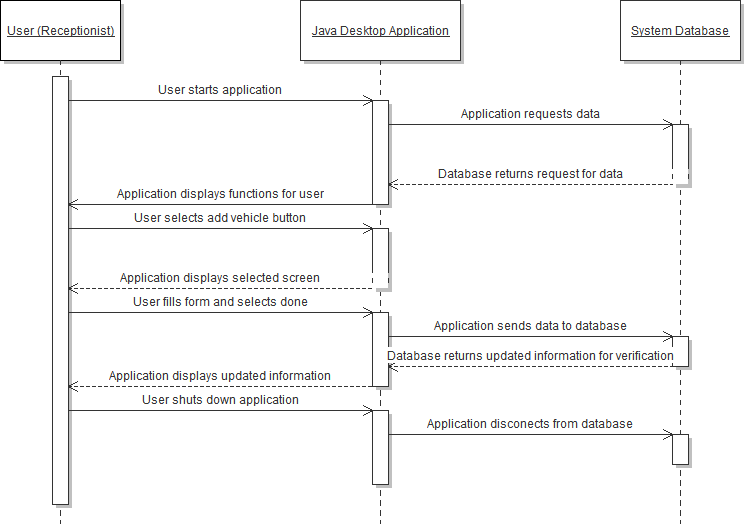
This use case diagram shows all the basic functions that the application will perform for users. This functions primarily involve either searching or adding data to the TARS database. This also shows the relationship between verifying information and displaying error messages.

## Class Diagram

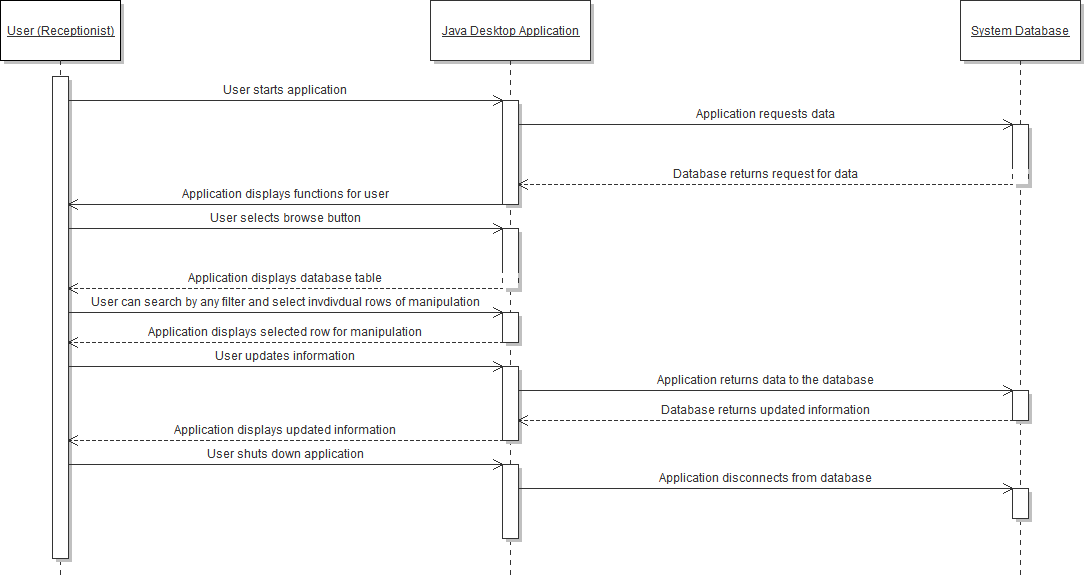


## Sequence Diagrams

Activity sequence diagram of user adding a vehicle to the database.



Activity sequence diagram of user browser database and updating selected information.



# DATA DESIGN

Database Table Design Example.

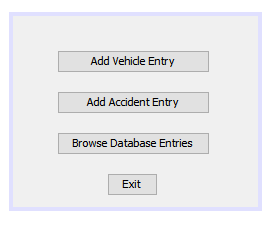
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Vehicle Id (String) | Vehicle Model (String) | Vehicle Year (Int) | Owner Name (String) | Owner Address (String) | Owner Contact Number (Int) | Accident ID (Int) | Accident Location (String) | Accident Comments (String) |
| 526-UMA | TESLA | 2077 | John Smith | 32 Castle Rd Hill | 0420682586 | 2865 | Vettenburgh Street | Was going twice the speed limit |

Further notes to be made are that all stored data is not unique except for Vehicle Id and can be reused in additional entries for any of the columns listed above. This is made with the assumption that there are no two vehicles with the same Id.

# INTERFACE DESIGN

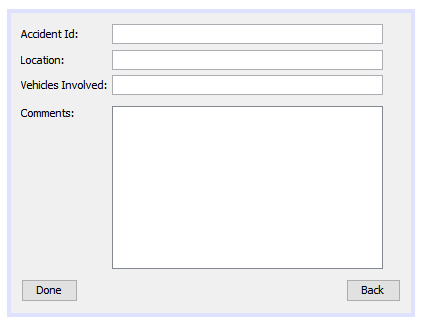
The below images are a basic guideline of the Java Applications UI and are subject to change.

The first Screen that appears on start-up which allows the user to add either a vehicle entry or an accident entry as well as browse all database entries. By pressing any of these buttons the user will navigate to the appropriate screen for the function. The exit button will prompt the user if they are sure if confirmation and if ‘Yes’ is selected the application will discontent from the database and shutdown.



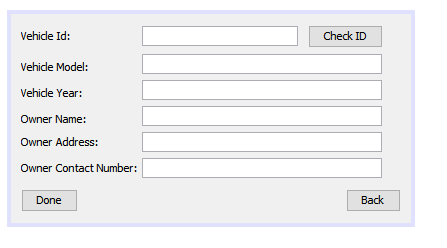
(Main Screen)

The user is sent to this screen on clicking add accident entry on the Main Screen. This screen allows the users to fill out the listed fields below before pressing the ‘Done’ button to send the information to the database and return the screen to the Main Screen. Note that the accident id field will be un-editable and will be filled with an autogenerated number. Pressing the back button will send users back to the main page and discard any changes made.



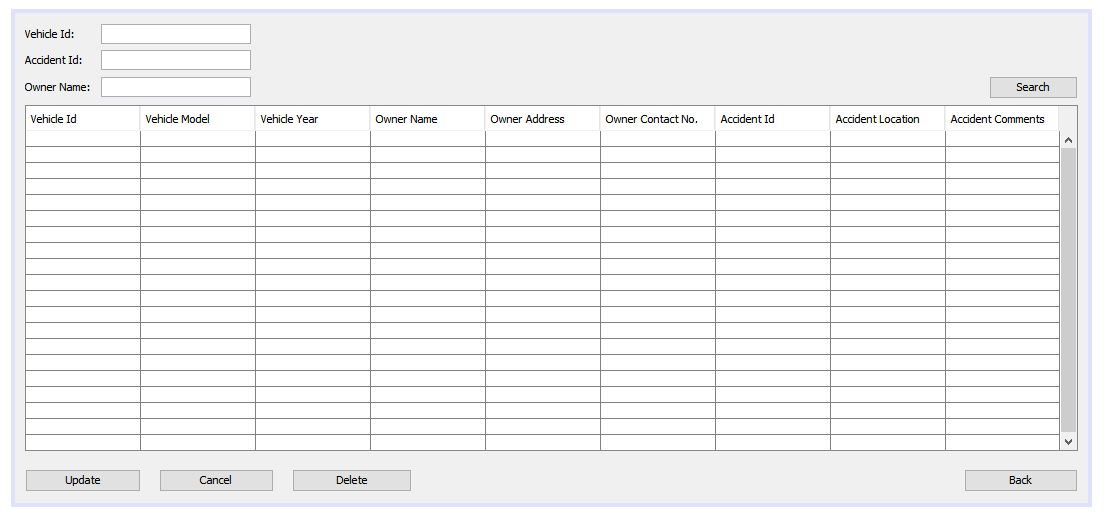
(Add Accident Entry Screen)

The user is sent to this screen on clicking add vehicle entry on the Main Screen. This screen lets users add vehicles to the database by filling out the fields below and pressing the button labelled ‘Done’. The back button will return users to the main screen and discard any changes made. The ‘Check ID’ button will only function when the vehicle id field has been done and will check the database for any existing vehicle entries with a matching Id. If a positive match is found the application will prompt the users if they wish to update the existing entry. In the case they select yes, the fields are auto completed with the existing information.



(Add Vehicle Entry Screen)

Users are sent to this screen upon selecting the ‘Browse Database Entries’ button on the main screen. This screen allows users to view all currently existing entries. Users will also be able to filter the entries by using vehicle Id, accident Id, or owner name and the clicking search. This will update the table to display the relevant entries. Although there are only three filters now more can be added for each individual property. Users can select there edit there chosen entry by clicking on the cell and changing the information. The user can then select Update Entry to send the information back to the database or Cancel which discard any changes made. The ‘Back’ button will return users back to the main page and discard any changes made. The ‘Delete’ button will remove the selected entry from the database.



(View Database Entries Screen)

# REQUIREMENTS MATRIX

|  |  |  |
| --- | --- | --- |
| **Requirement** | **Button** | **Inputs Required** |
| Browse Database | Browse Database Entries, Search | Click button on main screen, type in filters, search. |
| Check existing vehicles | Check | Click check on add vehicle screen. |
| Add vehicle entries | Add Vehicle Entry | Button on main screen, fill required fields, click done. |
| Add accident entries | Add Accident Entry | Button on main screen, fill required fields, click done. |
| Update entries | Update | On database entries screen click update. |
| Delete entries | Delete | On database entries screen click delete. |