**Star Hotel Management System Designing Report**

**1. System Analyzing**

Star Hotel Management System is designed to meet the requirements of Star Hotel management, this system will include three main functions: Customer Management, Order Management, Report system. The manager and clerk of Star Hotel can use this system to manage customers and orders, booking rooms for customers, change or delete orders, generate reports based on multiple requirements, maintain customer information and manage bill payments.

The system is based on GUI and native database, the GUI is clear and easy to understand. The system is easy to learn and use, with this system, Star Hotel can easily maintain entire hotel with serval clerks.

**2. Prototype Introduction**

This prototype displays part of the whole system. Current prototype is focused on customer management system with GUI and a small database. Due to the time limitation, current prototype version can only manage the customer information, with a function of CRUD of customer information and basic GUI of the system.

**3. Method and Class**

Methods and classes in this prototype is designed to support customer management system. We designed methods and classes to call every page and control every controller on it. We also designed methods and classes to generate database of this system and meet the CRUD requirements.

Following are the 3 methods we implemented.

## 1. Add customer

**private void** saveData() {  
  
 CustomerModel newCustomer = **new** CustomerModel();  
  
 newCustomer.setFirstName(**firstNameTextField**.getText());  
 newCustomer.setSurname(**surnameTextField**.getText());  
 newCustomer.setGender(**maleRadioButton**.isSelected());  
 newCustomer.setContactNum(**contactNumTextField**.getText());  
 newCustomer.setAddress(**addressTextField**.getText());  
 newCustomer.setSuburb(**suburbTextField**.getText());  
 **if** (!**stateChoiceBox**.getValue().isEmpty()) {  
 newCustomer.setState(**stateChoiceBox**.getValue());  
 }  
 **if** (!**postalCodeTextField**.getText().isEmpty()) {  
 newCustomer.setPostalCode(Integer.*parseInt*(**postalCodeTextField**.getText()));  
 }  
 newCustomer.setFrequenter(**defaulterCheckBox**.isSelected());  
 newCustomer.setDefaulter(**frequenterCheckBox**.isSelected());  
  
 CustomerDAO dao = **new** CustomerDAOImpl();  
  
 **if** (**om**.equals(OpenMode.***ADD***)) {  
 dao.insert(newCustomer);  
 }

}

## 2. Show the Customer List

**public void** fillData() {  
 **firstNameTextField**.setText(**customerBuffer**.getFirstName());  
 **surnameTextField**.setText(**customerBuffer**.getSurname());  
 **femaleRadioButton**.setSelected(**customerBuffer**.getGender().equals(**"Female"**));  
 **maleRadioButton**.setSelected(**customerBuffer**.getGender().equals(**"Male"**));  
 **contactNumTextField**.setText(**customerBuffer**.getContactNum());  
 **addressTextField**.setText(**customerBuffer**.getAddress());  
 **suburbTextField**.setText(**customerBuffer**.getSuburb());  
 **stateChoiceBox**.setValue(**customerBuffer**.getState());  
 **postalCodeTextField**.setText(Integer.*toString*(**customerBuffer**.getPostalCode()));  
 **defaulterCheckBox**.setSelected(**customerBuffer**.isDefaulter());  
 **frequenterCheckBox**.setSelected(**customerBuffer**.isFrequenter());  
 **postalCodeTextField**.setText(Integer.*toString*(**customerBuffer**.getPostalCode()));  
}

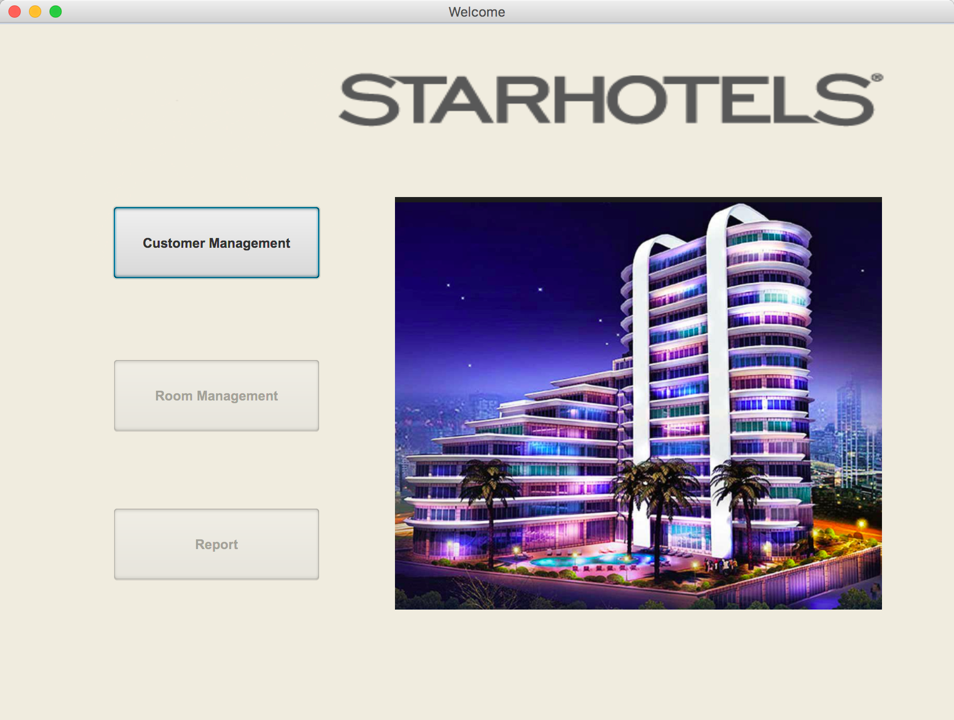
## 3. Delete the Customer

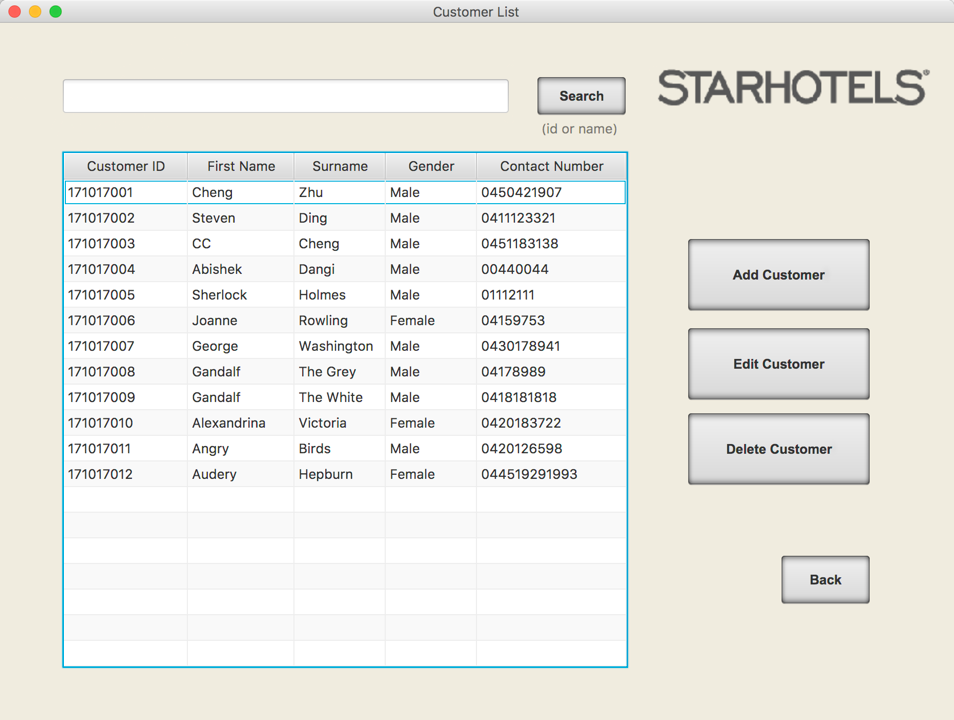
**deleteButton**.setOnAction(  
 event -> {  
 **int** selectedIndex = **customerTableView**.getSelectionModel().getSelectedIndex();  
 CustomerDAO dao = **new** CustomerDAOImpl();  
 dao.delete(**dataSource**.get(selectedIndex).getCustomerID());  
 **customerTableView**.getItems().remove(selectedIndex);  
 }  
);

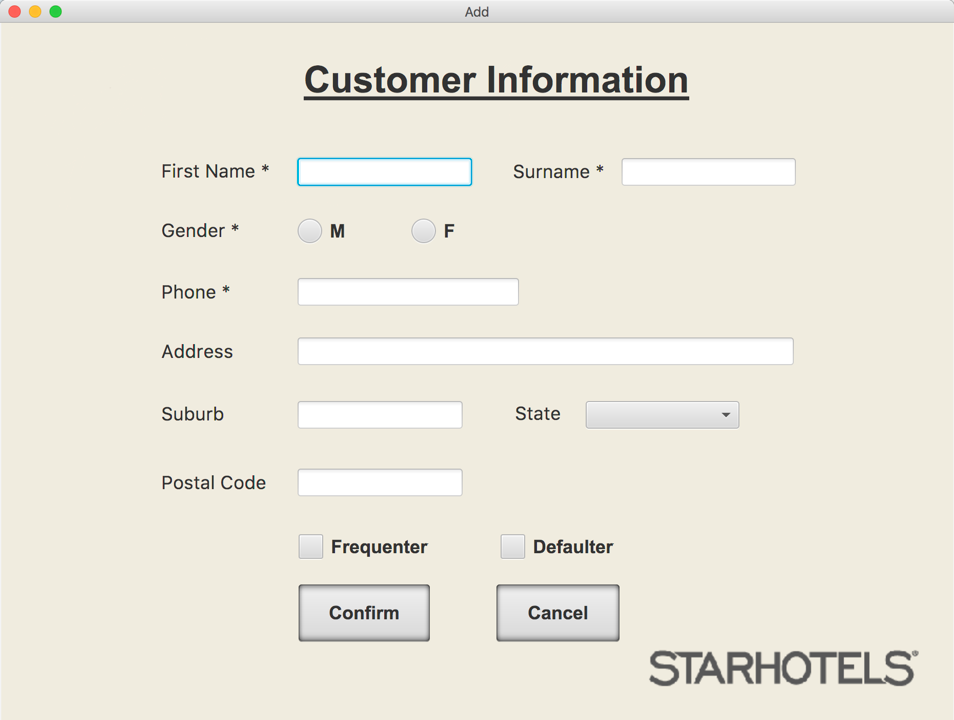
**4. GUI Designing**

The GUI of this prototype is designed as three pages. The first page is designed as the main page of entire management system, with a button to call customer management system page. In the customer list page, we can see customer information list can do have navigation function. We can also add customer, choose a customer to edit or delete. If user click add customer button or chose one term in the list and click edit button. The customer information page will display. On this page, user can type or edit every single information of customer, can with click confirm button, information will be transfer to database, if click cancel button, no change will be save.

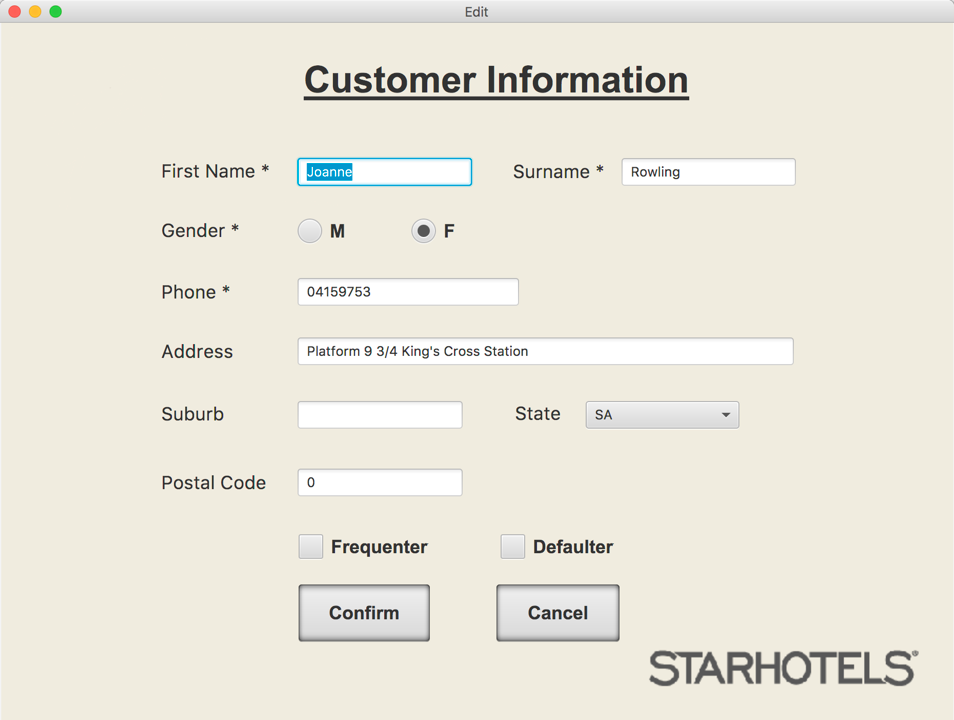
GUI screen shoots:

(Welcome Page)

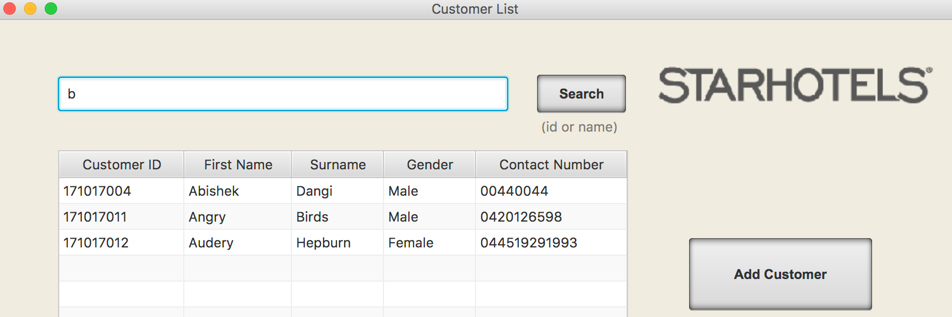
(Customer List)



(Add Customer)



(Edit Customer)



(Search)

**5. Database**

In database part, we designed a native database with SQL, it contains all information of customers such as name, gender, address, numbers, etc. Java code is used to connect database with GUI. With interaction with the GUI, user can find typical terms in the database, create new terms to database, change existed information, and delete customer information from database.

CRUD codes are shown as below:

### Create

**public void** insert(CustomerModel ctm) {  
 String insertRow = **"INSERT INTO Customer(first\_name, surname, gender, contact\_num, address, suburb, state, postal\_code, defaulter, frequenter) \n"** + **"VALUES(?,?,?,?,?,?,?,?,?,?)"**;  
 **try** (Connection conn = ConnectDB.*connect*();  
 PreparedStatement pstmt = conn.prepareStatement(insertRow)) {  
 pstmt.setString(1, ctm.getFirstName());  
 pstmt.setString(2, ctm.getSurname());  
 pstmt.setBoolean(3, ctm.getGender() == **"Male"**);  
 pstmt.setString(4, ctm.getContactNum());  
 pstmt.setString(5, ctm.getAddress());  
 pstmt.setString(6, ctm.getSuburb());  
 pstmt.setString(7, ctm.getState());  
 pstmt.setInt(8, ctm.getPostalCode());  
 pstmt.setBoolean(9, ctm.isDefaulter());  
 pstmt.setBoolean(10, ctm.isFrequenter());  
 pstmt.executeUpdate();  
 } **catch** (SQLException e) {  
 System.***out***.println(e.getMessage());  
 }  
}

### Read

**public** ObservableList<CustomerModel> showAll() {  
 String selectrow = **"SELECT customer\_id, first\_name, surname, gender, contact\_num, address, suburb, state, postal\_code, defaulter, frequenter FROM Customer"**;  
  
 ObservableList<CustomerModel> resultList = FXCollections.*observableArrayList*();  
 **try** (Connection conn = ConnectDB.*connect*();  
 PreparedStatement ps = conn.prepareStatement(selectrow)) {  
 ResultSet rs = ps.executeQuery();  
 *// loop through the result set* **while** (rs.next()) {  
 CustomerModel ctm = **new** CustomerModel();  
 ctm.setCustomerID(rs.getInt(**"customer\_id"**));  
 ctm.setFirstName(rs.getString(**"first\_name"**));  
 ctm.setSurname(rs.getString(**"surname"**));  
 ctm.setGender(rs.getBoolean(**"gender"**));  
 ctm.setContactNum(rs.getString(**"contact\_num"**));  
 ctm.setAddress(rs.getString(**"address"**));  
 ctm.setSuburb(rs.getString(**"suburb"**));  
 ctm.setState(rs.getString(**"state"**));  
 ctm.setPostalCode(rs.getInt(**"postal\_code"**));  
 ctm.setDefaulter(rs.getBoolean(**"defaulter"**));  
 ctm.setFrequenter(rs.getBoolean(**"frequenter"**));  
  
 resultList.add(ctm);  
 }  
 } **catch** (SQLException e) {  
 System.***out***.println(e.getMessage());  
 }  
 **return** resultList;  
}

### Update

**public void** update(CustomerModel ctm) {  
 String updateRow = **"UPDATE Customer SET first\_name = ?, surname = ?, gender = ?, contact\_num = ?, address = ?, suburb = ?, state = ?, postal\_code = ?, defaulter = ?, frequenter = ? WHERE customer\_id = ?"**;  
 **try** (Connection conn = ConnectDB.*connect*();  
 PreparedStatement pstmt = conn.prepareStatement(updateRow)) {  
 pstmt.setString(1, ctm.getFirstName());  
 pstmt.setString(2, ctm.getSurname());  
 pstmt.setBoolean(3, ctm.getGender() == **"Male"**);  
 pstmt.setString(4, ctm.getContactNum());  
 pstmt.setString(5, ctm.getAddress());  
 pstmt.setString(6, ctm.getSuburb());  
 pstmt.setString(7, ctm.getState());  
 pstmt.setInt(8, ctm.getPostalCode());  
 pstmt.setBoolean(9, ctm.isDefaulter());  
 pstmt.setBoolean(10, ctm.isFrequenter());  
 pstmt.setInt(11, ctm.getCustomerID());  
 pstmt.executeUpdate();  
 } **catch** (SQLException e) {  
 System.***out***.println(e.getMessage());  
 }  
}

### Delete

**public void** delete(**int** id) {  
 String deleterow = **"DELETE FROM Customer WHERE customer\_id = ?"**;  
  
 **try** (Connection conn = ConnectDB.*connect*();  
 PreparedStatement ps = conn.prepareStatement(deleterow)) {  
 *// set the corresponding param* ps.setInt(1, id);  
 *// execute the delete statement* ps.executeUpdate();  
  
 } **catch** (SQLException e) {  
 System.***out***.println(e.getMessage());  
 }  
}