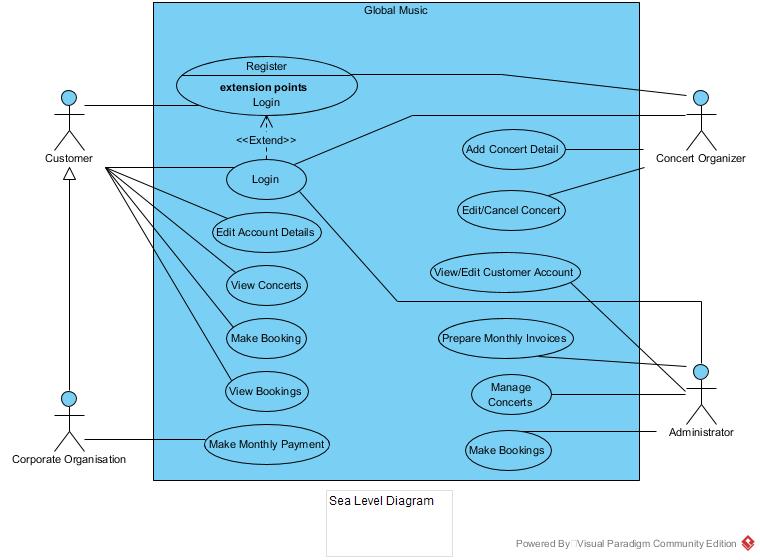
****

**Gl****obal Music**

**Group 2 Members:  
Samrat Bikram Shah, ID 1815674  
Amrit Raj Shilpakar, ID 1811037  
Praveen Raghubanshi, ID 1811043  
Zenith Maharjan, ID 1815684**

1. **Introduction:**

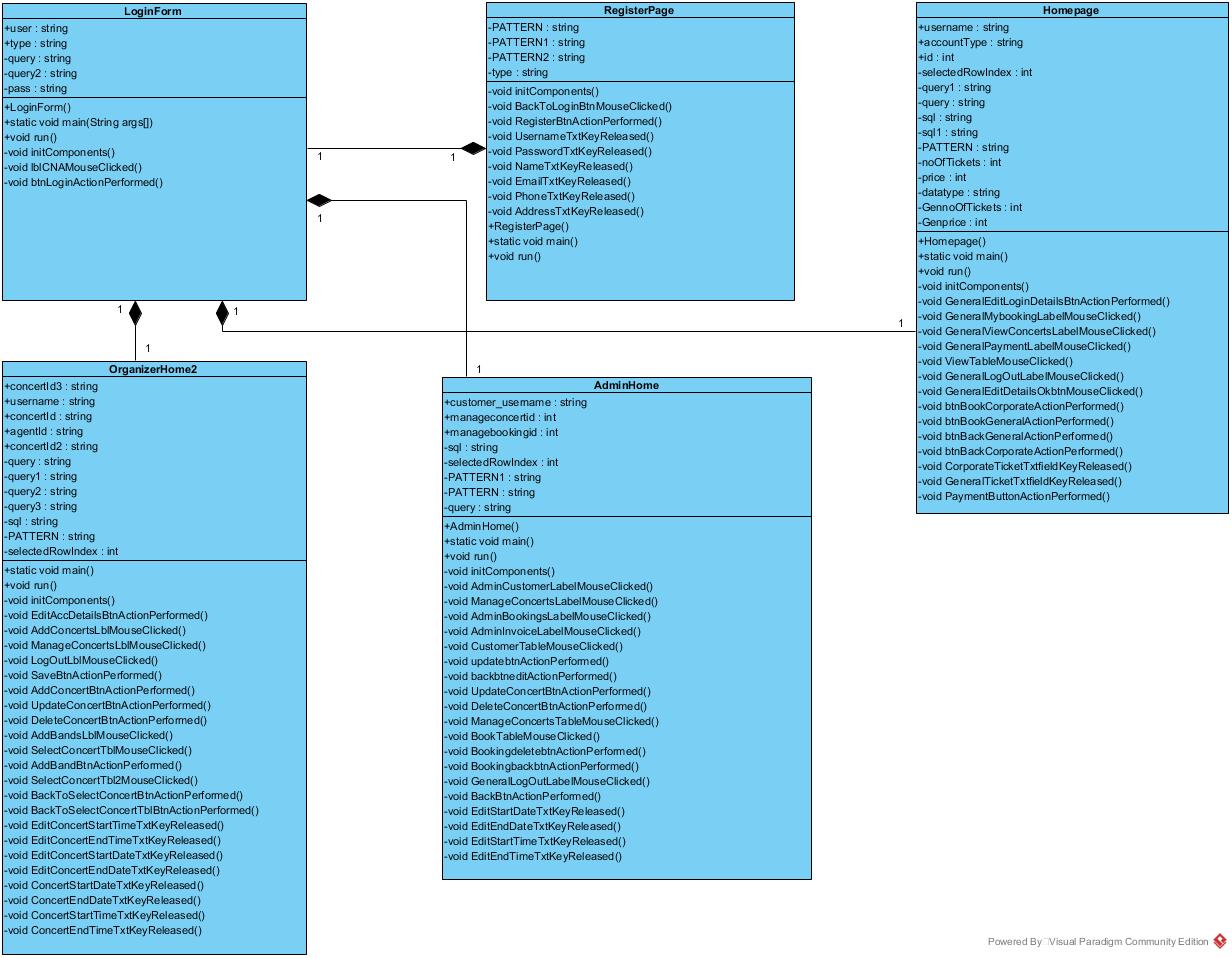
This project expected us to create an application named “Global Music” which allowed people to book tickets for various concerts added by organizers. This task really tested our problem-solving skills making us utilize all the things we learned so far. Besides the code itself, we were given the task of designing a proper database, normalize it, and implement it which would suit the need of our application. Likewise, other requirements were basic Use Case Diagrams, Class Diagram, and an Entity Relationship Diagram. Moreover, we also had to present a presentation for our project and every member of the group had to make an individual report about their work. Lastly, a client-server architecture was also expected.

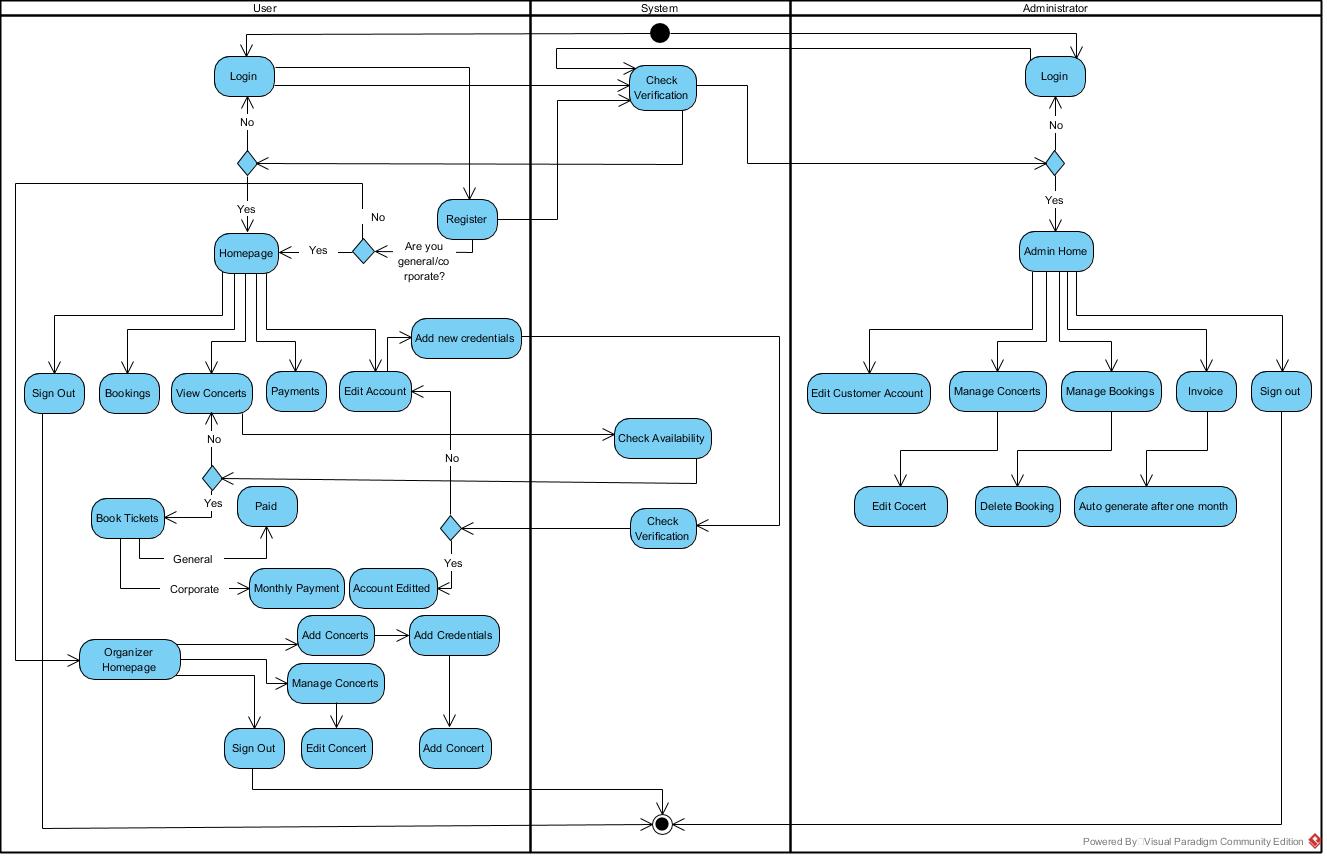
1. **System Design:**
   1. **UML Use Case Diagrams (Sea Level):**

This sea level diagram of the system “Global Music” describes the use cases for our actors who are customer, concert organizer, and administrator. The customer is generalized into corporate organizer. As per the use case, the customer, corporate organization and concert organizer all have the ability to create an account and login with their credentials. However, the administrator possesses a fixed username and password for login purposes. Here, a customer be it General or Corporate can access all the available concerts and book a ticket for it. Meanwhile, for the payments, corporate does monthly payments while a General user pays during booking. Similarly, an Organizer can add, edit and delete its concerts and add bands to the concerts. Lastly, an admin has permission to delete all bookings, manage users and concerts as well as generating monthly invoice.

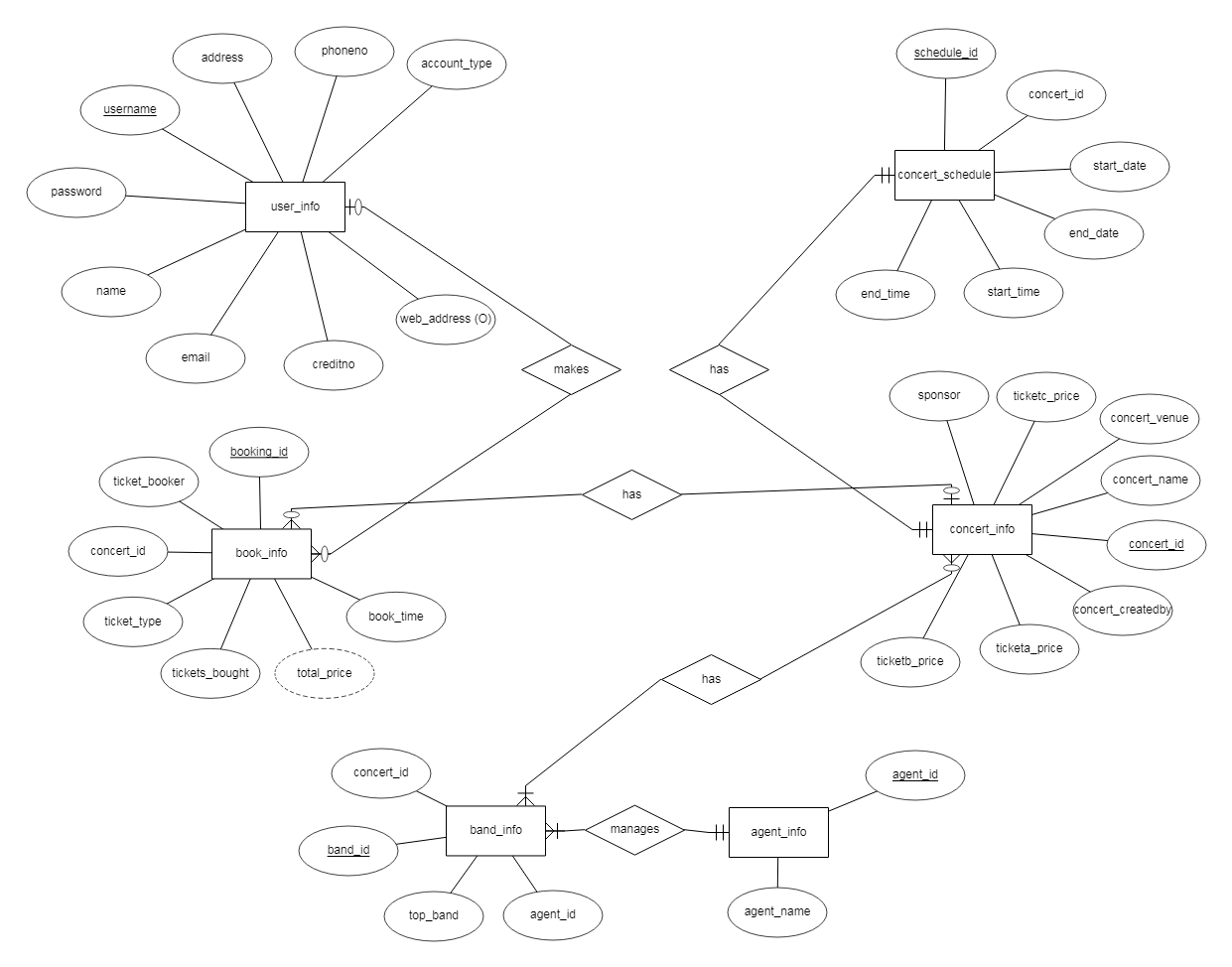
* 1. **UML Class Diagram:**

For this class diagram, here are five different classes used. Here, the first class that will be accessed is the LoginForm class. It has one to one relationship with every other class. Saying this, one cannot log in without registering so RegisterPage class is essential in order for this application to function. After logging in it can access any of the three Homepage classes according to the user who has logged in. It needs General or Corporate login id to go to Homepage class while it needs Organizer login id to go to OrganizerHome2 class. Lastly, it requires a special admin id and password to go to the AdminHome class.

****

* 1. **UML Activity Diagram:** This activity diagram describes the flow of the program from logging in to accessing homepages to logging out. It has three partitions for User, System, and administrator which describes the activities that take place within them.
  2. **Database Design:**

Our database is divided into 6 tables and normalized to 3NF form. The Entity Relationship diagram of its Entities and Attributes can be seen below:

****

1. **Implementation:**

To implement our system, we used Visual Paradigm to design our Use Case diagram, Class diagram, and Activity diagram while we used ERDPlus, which is an online database modeling tool to design the E-R diagram. As for our Java coding, we used NetBeans IDE version 8.1. One core reason for using these software platforms is familiarity since our lecturer used them in the past and we merely got used to it. Another reason for using NetBeans was that it was faster and easier to use than other IDEs.

* 1. **Some code snippets and description:**

*import java.sql. \*;*

*import net.proteanit.sql.DbUtils;*

*Connection con;*

*PreparedStatement ps;*

*ResultSet rs;*

*try*

*{con = DriverManager.getConnection("jdbc: mysql://localhost/musicfestival”,” root","");*

*String sql = "select \* from user\_info;";*

*ps = con.prepareStatement(sql);*

*rs = ps.executeQuery();*

*CustomerTable.setModel(DbUtils.resultSetToTableModel(rs));*

*}*

*catch(Exception e)*

*{*

*JOptionPane.showMessageDialog(null, e.getMessage());*

*}*

One of the codes that will be seen throughout the whole project is these where a Connection is established with the musicfestival database in the server and a certain SQL query is executed in java so it can be viewed in the table. We only did this and not the complete client-server architecture for the fact it is easier to implement and use. For this, we imported the MySQL-connector-java-8.0.15.jar file to establish the database connectivity. After importing the SQL package, the Connection, ResultSet, and Prepared Statement classes were initialized. These all commands are conducted in a try-catch block. Followingly, we imported DbUtils class using the rx2xml.jar file to display our table’s data in the GUI which can be seen throughout the program. Mostly all the codes in the project are the light or heavy modification of the code above. For instance:

*String query = "update user\_info set username=?" + "where username=?";*

*ps = con.prepareStatement(query);*

*ps.setString(1, UsernameTxt.getText());*

*ps.setString(2,customer\_username );*

*ps.executeUpdate();*

*ps.addBatch();*

All this code does is update data existing in the database rather than viewing it. For this we don’t require the ResultSet class.

**As for the GUI part:**

*AdminCustomerLabel.setBackground(new java.awt.Color(255,255,255));*

*AdminInvoiceLabel.setBackground(new java.awt.Color(34,36,49));*

*AdminBookingsLabel.setBackground(new java.awt.Color(34,36,49));*

*ManageConcertsLabel.setBackground(new java.awt.Color(34,36,49));*

*AdminBotPanel.removeAll();*

*AdminBotPanel.repaint();*

*AdminBotPanel.revalidate();*

*AdminBotPanel.add(AdminCusEditPanel);*

*AdminBotPanel.repaint();*

*AdminBotPanel.revalidate();*

ErrorDateLabel.setText(null);

ErrorDateLabel.setText(null);

These are also seen throughout the code which are just front-end codes allowing you to change panels in the homepage according to the requirement while adding color to the label you clicked to access the panel.

1. **Testing**

For this, the errors were easily identified since most of the stuff was done in the IDE. It pointed out some syntax error easily enabling us to identify measly errors. Meanwhile, to test connectivity, database errors and SQL errors we used exception statements block in a try-catch block to identify any errors while running the program. Likewise, while passing values between classes we tested the variable’s value using System.out.println statement.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.N. | Java class name | Action | Expected | Result |
| 1. | RegisterPage | Left Blank | Error registering | Pass |
| 2. | RegisterPage | Existing username | “Username exists” error | Pass |
| 3. | RegisterPage | Wrong format | “Incorrect format” error | Pass |
| 4. | RegisterPage | Unique information in correct format | Successful Register | Pass |
| 5. | LoginForm | Wrong information | “Login failed” error | Pass |
| 6 | LoginForm | Correct information | Successful Login per account type | Pass |
| 7. | Homepage | Click My Booking tab | Display all booked concert | Pass |
| 8. | Homepage | Click View concerts tab | Display all concerts | Pass |
| 9. | Homepage | Click Payments Tab | Display payments left | Pass |
| 10. | OrganizerHome2 | Click Add Concerts | Adds concert to database | Pass |
| 11. | OrganizerHome2 | Enter data in incorrect format while adding | “Incorrect Format” error | Pass |
| 12. | OrganizerHome2 | Click Manage Concerts | Display all added concerts | Pass |
| 13. | OrganizerHome2 | Click Add Bands | Add bands into database | Pass |
| 14. | AdminHome | Click View Customer accounts | Display all customer accounts | Pass |
| 15. | AdminHome | Click Manage Concerts | Display all concerts | Pass |
| 16. | AdminHome | Click Manage Bookings | Display all bookings | Pass |
| 17. | AdminHome | Click Invoice Tab | Displays monthly invoice | Pass |

1. **Reflection and Responsibilities:**

Looking at the requirements and quota that were to be met in this group project, we firstly decided to make a proper database design, normalize it and implement it. We thought this was a significant task to do in order to start our project. Besides this, we discussed the basic UML diagrams and decided to make an Activity Diagram, a rough sketch of a Class Diagram and an   
E-R diagram. As a sea level diagram sample was provided to us in the assignment brief, things were going smoothly till now. Then we started making our database and had problems with normalization. It took a while to realize it but we asked our lecturer about the issue who advised about the proper database structure. Since it was a group project, we decided to stay in one of our member’s house and divide the tasks while doing the project. To fix the database structure, we also referred to some movie booking database relationship schema using online sources (vertabelo, 2019). After the diagrams, our next motive was to implement a basic GUI and establish connectivity with the database using Java programming. Till now we had little to no problems but finally, we faced a problem. We couldn’t display our data from the database into the JTable. We looked up many things found some cryptic stuff about arrays, etcetera. Finally, as a solution, we found the rs2xml.jar file which solved this problem very easily and hence our GUI and basic connectivity was done. A few days later, our lecturer told us about how an application is built and how it has a GUI layer, business layer for logic and a backend layer for all connectivity. Thinking we had time, we tried implementing it but couldn’t succeed in doing so, which eventually cost us some time. Likewise, for the database, we had a fair share of problems. We forgot to add bands, agents, sponsors, and basic information so we had to adjust some changes in the database again while adding more GUI Panels for it to work properly. This pushed the works near the deadline

* 1. **Roles and Responsibilities:**

|  |  |
| --- | --- |
| **Name and ID** | **Role** |
| Samrat Shah, 1815674 | GUI, Backend and Validation of Homepage (General, Corporate, Booking), Basic Sea Level Diagram and Database designer. |
| Amrit Raj Shilpakar, 1811037 | GUI, Backend and Validation of Login, Organizer Homepage backend and backend validations along with Entity Relationship diagram. |
| Praveen Raghubanshi, 1811043 | GUI, Backend and Validation of Registration Page, GUI and front-end validation of Organizer Homepage along with a class diagram. |
| Zenith Maharjan, 1815684 | GUI, Backend and Validation (front-end and backend) of Admin Homepage along with an Activity diagram. |

1. **Conclusion:**

In conclusion, we were able to separate our work evenly, and every member of the group contributed to carry out the project smoothly. In this report, we described the way we started our project, how we divided them, the problems faced and how we were able to overcome it. Likewise, we also described the diagrams for our code which assisted us to find out the way we should build the application. Lastly, we also discussed some of our important code snippets and how we used it in most parts of the program also showing how we tested our application to work as per our need.

1. **References:**

Drkušić, N. (2015) ‘How to Design a Database Model for a Movie Theatre Reservation System.’ *vertabelo blogs* 31st July. Available at: <https://www.vertabelo.com/blog/technical-articles/a-database-model-for-a-movie-theater-reservation-system> (Accessed:08/05/2019)

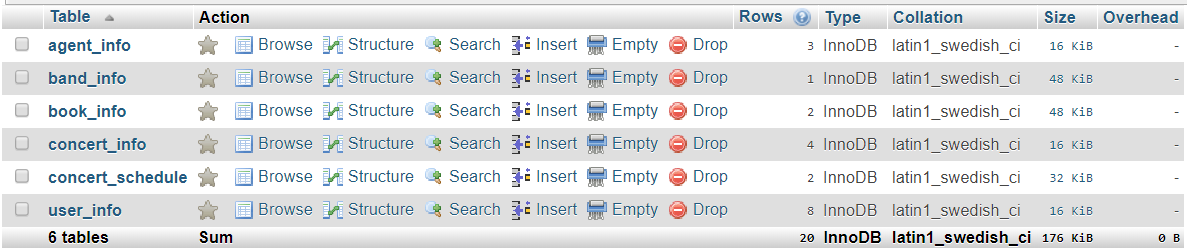
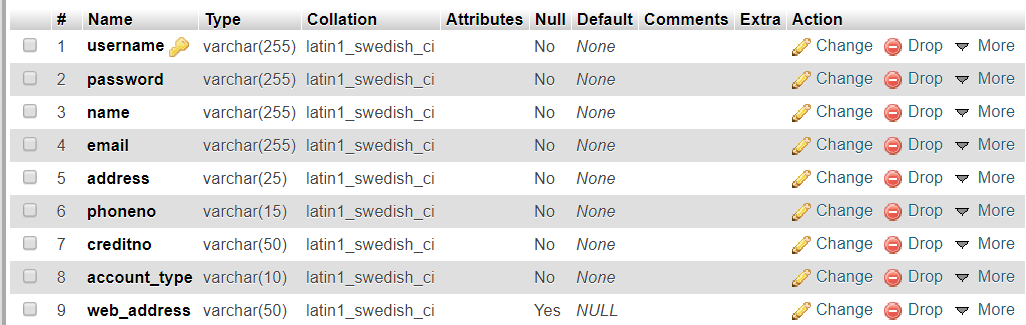
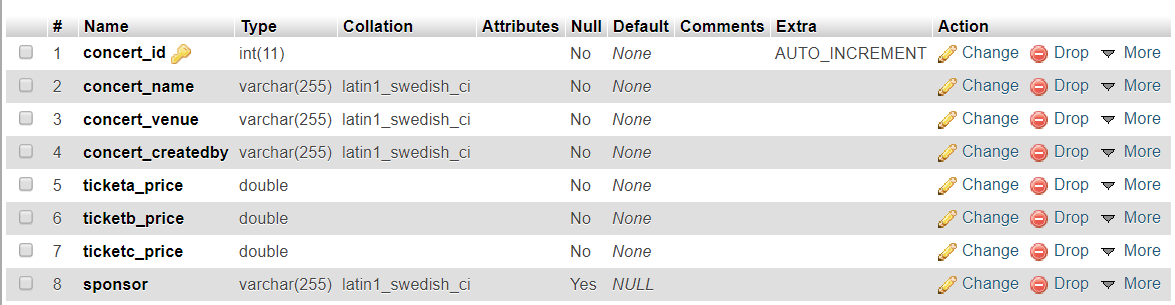
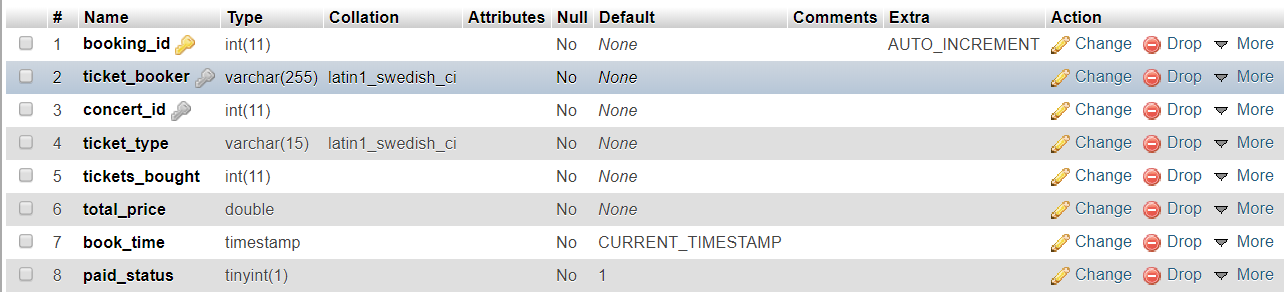
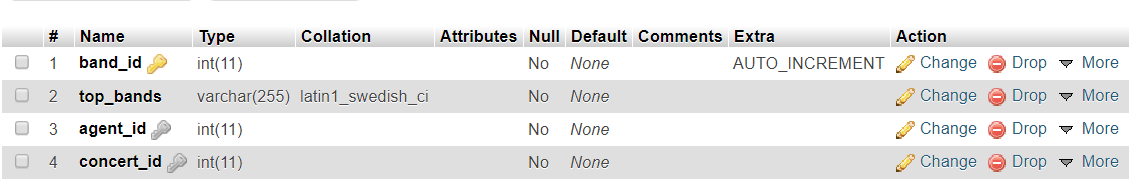
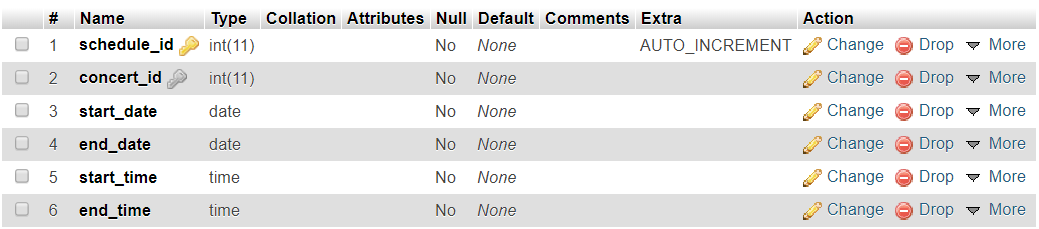
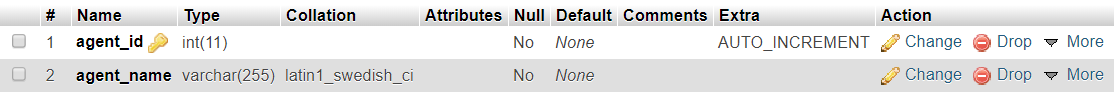
1. **Appendix:**
   1. **Database tables:**

Fig: Database tables

* 1. **Database attributes:**
  2. **Used Code examples:**

*import java.sql.\*;*

*import javax.swing.\*;*

*import net.proteanit.sql.DbUtils;*

*import javax.swing.table.DefaultTableModel;*

*private void GeneralMybookingLabelMouseClicked(java.awt.event.MouseEvent evt) {*

*GeneralMybookingLabel.setBackground(new java.awt.Color(255,255,255));*

*GeneralViewConcertsLabel.setBackground(new java.awt.Color(34,36,49));*

*GeneralPaymentLabel.setBackground(new java.awt.Color(34,36,49));*

*GeneralBotPanel.removeAll();*

*GeneralBotPanel.repaint();*

*GeneralBotPanel.revalidate();*

*GeneralBotPanel.add(MyBookingPanel);*

*GeneralBotPanel.repaint();*

*GeneralBotPanel.revalidate();*

*try*

*{*

*con = DriverManager.getConnection("jdbc: mysql://localhost/musicfestival”,” root","");*

*String query1 = "select book\_info.concert\_id, concert\_name, booking\_id, tickets\_bought, total\_price from concert\_info, book\_info where ticket\_booker =? AND concert\_info.concert\_id = book\_info.concert\_id;";*

*ps = con.prepareStatement(query1);*

*ps.setString(1, username);*

*rs = ps.executeQuery();*

*BookTable.setModel(DbUtils.resultSetToTableModel(rs));*

*}*

*catch(SQLException e){*

*JOptionPane.showMessageDialog(null, "Not Connected");*

*}*

*}*

This code is an event handling for which a certain panel will be viewed after clicking the label and a connection is established executing the SQL command to view contents from database executed from the query.