# 

# 

# **Business Intelligence Tool**

**Project Design and Status Report**

**4/9/2017**

CMSC 495-7980 Group 5

Jay Herford, Daniel Park, Thomas Barton, Jason Boone

Project Goals and Completion Percentages

|  |  |  |
| --- | --- | --- |
| Goal | Comp % | Notes |
| Basic design and feature list | 90% | Initial design phase complete. Final details of UI design in progress. |
| Application testing design | 100% | Testing documentation complete. Additional test may be added as needed as the development process proceeds. |
| Initial user interface mockup | 80% | Interface prototype is complete but does not yet contain elements for all phase 1 functions. |
| AWS database setup | 100% | Database created in AWS. All team members have verified connectivity using MySQL Workbench. |
| Database table design | 100% | The overall design of the tables is complete. |
| Database table implementation | 60% | Creation of all tables required to support a single reporting table has occurred. All constraints are in place. Additional reporting tables are next to be added. |
| Database table data population | 60% | Data for supporting tables has been loaded. Data for the first reporting table has been loaded. |
| Implementation of database connection from application | 50% | Database connection classes created and tested. Next step is integration into the application. |
| Functional UI returning data from the database | 20% | Designed methods to return desired data sets from database. |
| UI look and feel refinements | 50% | Initial design has been created. However, additional features and a more refined look are still to come. |
| Table view in application with no parameters set | 0% |  |
| Column ordering functionality within the UI | 0% |  |
| Table view in application with parameters set | 0% |  |
| Column view in application | 0% |  |
| Table statistics view in application | 0% |  |
| Graph view (Pie, Line, Scatter, Bar) in application | 0% |  |
| Final application refinements | 0% |  |
| Testing of completed application | 0% |  |
| Additional refactoring and bugfixes | 0% |  |

Refined Project Diagrams

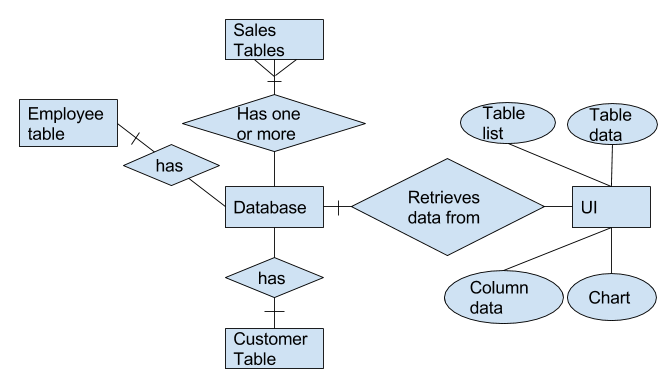


Figure 1: Program entity relationship diagram

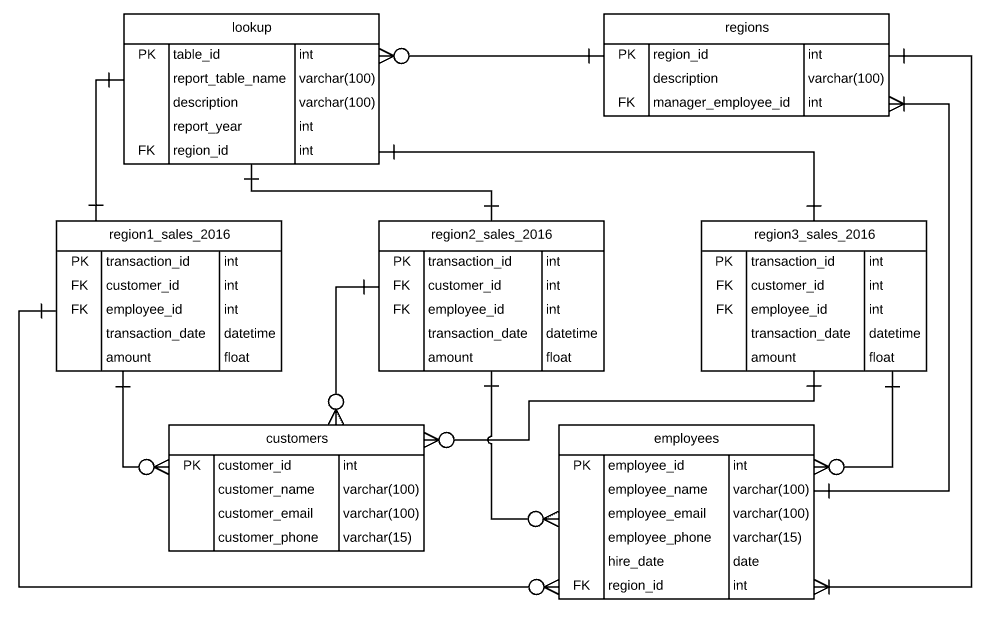
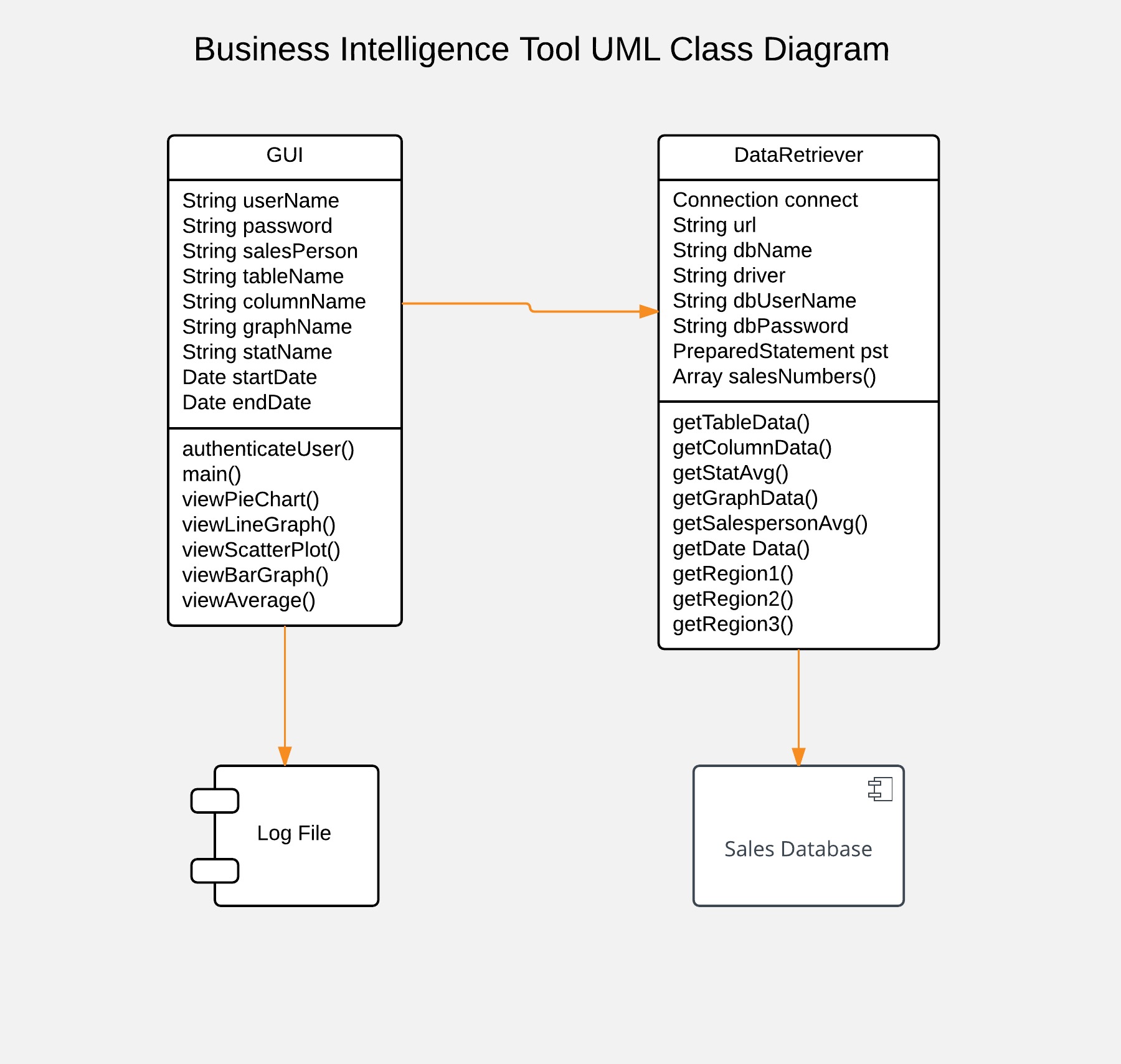


Figure 2: Database entity relationship diagram

Figure 3: Program UML class diagram

Pseudo-Code

public class GUI extends Application {

static final LOGGER //create the logger

public void start (stage primary Stage) {

set Title("BI Intelligence Tool)

Text Box ("Username")

Text Box ("Password")

Button ("Login")

Action Event Handler for Login button

//Authenticated the user

setOnAction (ActionEvent Login) -> {

authenticate();//validates username and password

if (isvalid == false) {

"Try Again"

}

if (isvalid == true) {

Display new scene

"Welcome" + username; //welcome the user

Drop down menu with Salespeople

ToggleGroup group; //ensure only one view can be selected by grouping radio buttons

//create radio buttons for different data displays

Radio Button ("Table");

Radio Button ("Column");

Radio Button ("Graph");

Radio Button ("Chart");

//create a calendar to select start and end date ranges

DatePicker start;

DatePicker end:

//create a button to submit selections and view data

Button bt = new Button("Get Data");

}

Action Event Handler for Get Data button

//this action event handler will be triggered when the Get Data button is selected

setOnAction (ActionEvent Get Data) -> {

getStartDate(); //call a method to retrieve start date selection

getEndDate(); //call a method to retrieve the end date selected

getSalesPerson(); //call method to determine saleperson selected from drop down

getChartType(); //call a method to determine the data display type selected

//now we call the DataRetrieval class to populate our data displays

Call DataRetrieval Class

Pass the parameters from the aforementioned methods into the DataRetrieval Constructor

//Use switch statement to differentiate between data display selections

Switch Statement() {

case 1: Table

// new scene for data display

//display table

break;

case 2: Column

// new scene for data display

//display column

break;

case 3: Graph

// new scene for data display

//display table

break;

case 4: Chart

// new scene for data display

//display column

break;

}

main() {

//create and format log file to record username, password, data type selections

file handler

create log file

simple formatter

launch(args)

}

public boolean authenticate(String user, String pword) {

boolean isValid = false;

if (user.equalsIgnoreCase....

&& pword.equals... {

isValid = true;

}

return isValid;

}

//Separate class called DataRetrieval

class DataRetrieval {

//connect DB using JDBC

Connection conn = null;

String url = "jdbc:derby://localhost:1527/";

String dbName = "sample";

String driver = "org.apache.derby.jdbc.ClientDriver";

String userName = "\*\*\*\*\*\*";

String password = "\*\*\*\*\*\*";

try {

Class.forName(driver).newInstance();

conn = DriverManager.getConnection(url+dbName,userName,password);

System.out.println("Connected to the database");//for testing purposes

Constructor (saleperson, region, dates) {

}

//Use prepared statements to query db for result sets

PreparedStatement pst = conn.prepareStatement("SELECT \* FROM Sales WHERE (username = ? AND password = ?)");

pst.setString(1, username);

pst.setString(2, passWord);

ResultSet rs = pst.executeQuery();

}

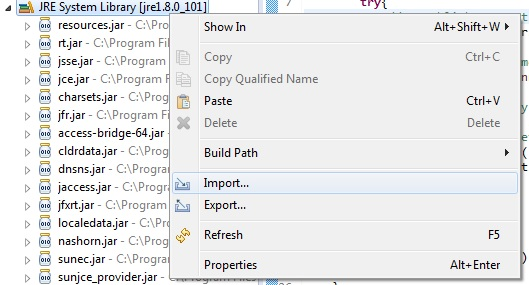
//more prepared statements depending on user input....

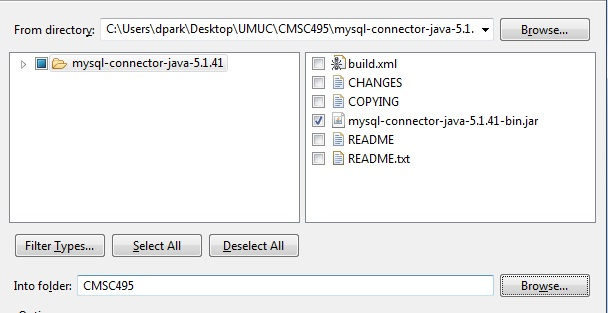
//return data to GUI class

JDBC Connectivity

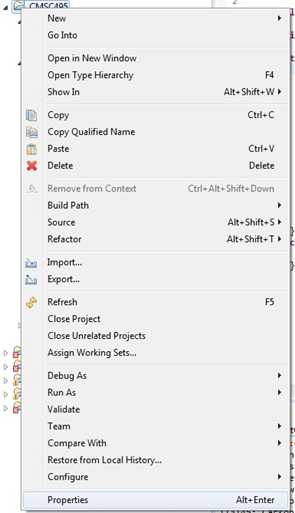
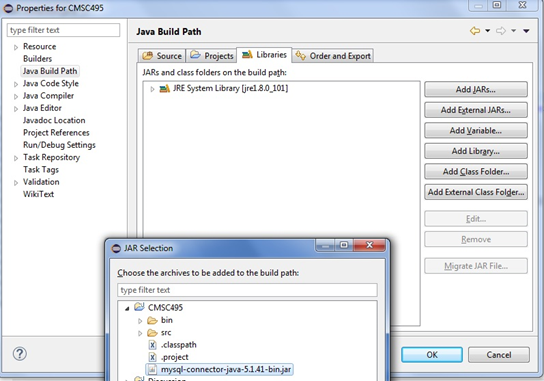
The business intelligence software will use JDBC connection to the Amazon Web Services Relational Database System MySQL instance. In order to utilize JDBC connectivity, the JDBC driver must be configured at the time the code is compiled. The program has been created using JDBC MySQL driver version 5.1.41. Previous versions of the JDBC driver may work but have not been tested. The JDBC driver can be downloaded at <https://dev.mysql.com/downloads/connector/j/5.1.html>. The following are the steps to configure JDBC in both Eclipse and NetBeans.

**Eclipse**

1. Unzip the downloaded JDBC driver.
2. Add the JDBC driver jar file to project folder
   1. Right click JRE System Library -> Import
   2. Choose File System
   3. Locate the directory the JDBC jar file has been unzipped

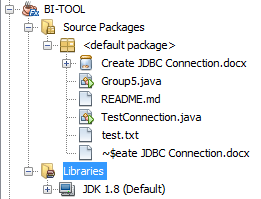


3) The jar file must now be added to the project build path

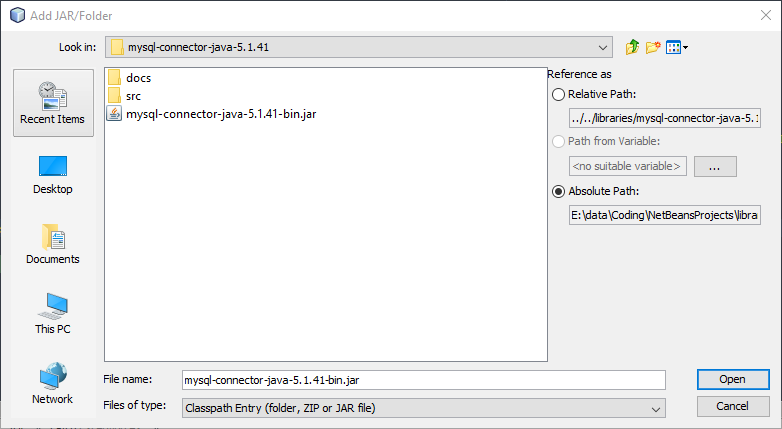
1. Right click project folder -> Properties
2. Go to Java Build Path -> Add JARS and find the JDBC jar file that has been imported to the project folder

**NetBeans**

1. Unzip the downloaded JDBC driver.
2. Add the JDBC driver jar file to project folder
3. Right click Libraries -> Add/Jar Folder



1. Locate the unzipped JDBC jar driver



1. Select Open

Updated Project Schedule

Added more details to product deliverables based on the project goal breakdown. Changes are marked in blue. Responsibilities now list deliverables that have been completed as several have been completed ahead of schedule.

|  |  |  |
| --- | --- | --- |
| **Deliverable** | **Due Date** | **Responsibility** |
| **Project Plan** | Mar 26 | Team Collaboration (Effort Lead: Daniel) - **Completed** |
| **Test Plan** | Apr 2 | Team Collaboration (Effort Lead: Thomas and Jason) - **Completed** |
| **Project Design and Status Report** | Apr 9 | Team Collaboration (Effort Lead: Jason) - **Completed** as of the submission of this document. |
| **Phase 1 Source** | Apr 16 |  |
| Create AWS Database and establish connections | Apr 10 | Daniel- **Completed**. All team members verify connectivity |
| Set up GitHub Project | Apr 10 | Jay - **Completed.** |
| Table design | Apr 10 | Jason - **Completed**. |
| Setup JDBC | Apr 10 | Daniel - **Completed**. |
| Basic Core UI | Apr 12 | Jay - **Completed** Initial draft UI |
| Produce result sets | Apr 12 | Daniel |
| Creation of database reporting table 1 and supporting tables | Apr 10 | Jason - **Completed**. |
| Test data population of reporting table 1 and supporting tables | Apr 10 | Jason - **Completed**. |
| Display list of tables | Apr 12 | Thomas |
| Table display functionality with no parameters set | Apr 12 | Thomas, Jay |
| Column ordering functionality within the UI | Apr 12 | Thomas, Jay |
| Group review | Apr 13 | Team |
| Final Revisions | Apr 15 | Team |
| **Phase 2 Source** | Apr 23 |  |
| Reporting table 2 creation and test data population | Apr 17 | Daniel, Jason |
| Produce result sets | Apr 19 | Daniel |
| Table display functionality with parameters set | Apr 19 | Thomas, Jay |
| Column display functionality | Apr 19 | Jay |
| Table Statistics view functionality | Apr 20 | Jay |
| Group review | Apr 20 | Team |
| Final Revisions | Apr 22 | Team |
| **Phase 3 source** | Apr 30 |  |
| Reporting table 3 creation and test data population | Apr 24 | Daniel, Jason |
| Produce result sets | Apr 24 | Daniel |
| UI look and feel refinements | Apr 26 | Jay |
| Graph display functionality (Pie, Line, Scatter, Bar) | Apr 26 | Thomas, Jay |
| Group review | Apr 27 | Team |
| Final Revisions | Apr 29 | Team |
| **Final Submission** | May 7 |  |
| Full testing | May 1 | Team |
| Final revisions | May 6 | Team |

Group Member Contributions

**Jay Herford**

* Established communicative channel for team and collaborated with team on application ideas and technologies to be used.
* Created Github group and project.
* Consulted with team members on prior experience of establishing an RDS in AWS.
* Refined documentation and contributed section to Project Plan. Finalized UML in Project Design.
* Created Pseudo-code for Java classes.
* Created initial GUI core with primary stage, assorted buttons, and date picker.

**Daniel Park**

* Created the initial project plan documentation.
* Contributed to the test plan documentation and the project design and status report.
* Setup and configured AWS RDS instance.
* Documented and distributed steps to configure MySQL Workbench connection to the AWS RDS instance.
* Documented and distributed steps to configure JDBC connection to the AWS RDS and created a sample Java class for testing.

**Thomas Barton**

* Gave feedback on the original plan document.
* Contributed to the test plan documentation test procedures for the GUI.
* Contributed the program ERD for the project design and status report.
* Contributed minutely to the database design.
* Contributed to GUI design functionality.

**Jason Boone**

* Contributed sections to the project plan document.
* Created the initial test plan documentation layout and populated some sections.
* Contributed sections to the project design and status report.
* Created Architectural Context, Data Flow and Entity Relationship diagrams for documentation.
* Finalized the database design based on collaborative team ideas.
* Set up database tables with necessary constraints and populated them with test data.
* Supplied some short code examples while brainstorming about database connection and the process of returning data.