Dance Registration System

Final Report

Group Number: 2

CPSC 488 Section 01

**TABLE OF CONTENTS**

**1 REFERENCES ………………………..………………………………………………….……4**

**2 COMPLETION……..………..………………………………………………………………...5**

2.1 COMPLETE MODULES ………………………………………………………………..…...5

2.2 INCOMPLETE MODULES ……………………………………………………………..…...5

*2.2.1 Future Modules..*…………………..……………………………………………......*.*5

**3 PROBLEM EXPLANATION………………………………………………………………....6**

3.1 OVERVIEW OF THE SYSTEM……………………………………………………………...6

**4 DESCRIPTION OF THE SOLUTION.……………………………………………………....6**

4.1 OVERVIEW OF THE SOLUTION...………………………………………………………...6

**5 SPECIFICATION REQUIREMENTS……………………………………………………....6**

5.1 DESCRIPTION…………………......………………………………………………………...6

5.2 BROAD OBJECTIVES…………......………………………………………………………...6

*5.2.1 Functioning UI for Registration*…..……………………………………………......*.*6

*5.2.2 Scheduling Functionalities*……………………………...……………………….......6

*5.2.3 Parent-Child Visibilities*…………………………………..…………………….......7

**6 SCOPE………………………………….……………………………………………………....7**

6.1 PRODUCTS…………………......…………………………..………………………………...7

*6.1.1 User Interface*…..………………………………………………………………......*.*7

*6.1.2 Database*…..……………………………………………………………………......*.*7

6.2 ITEMS ADDRESSED…………......…………..……………………………………………...7

*6.2.1 Database Backup*.………………………………………………………………......*.*7

*6.2.2 Exporting Files*…………………………………………………………………......*.*7

*6.2.3 Registering Users*.………………………………………………………………......*.*7

*6.2.4 Assigning Students to Classes..*…………………………………………………......*.*7

*6.2.5 Removing Activities and Students*………………………………………………......*.8*

*6.2.6 Track Payments*…………………………………………………………………......*.*8

6.3 RECOMMENDED FEATURES......…………..……………………………………………...8

*6.3.1 Saving Logs of Deactivated Users…*……………………………………………......*.*8

*6.3.2 Calculating Class Charges..*……………………………………………………......*.*8

*6.3.3 Parent Payment...*.………………………………………………………………......*.*8

*6.3.4 Child Attendance*…………..……………………………………………………......*.*8

6.4 TERMS……………………….........…………..……………………………………………...8

*6.4.1 Student/Child/Children*…………..…………………..…………………………......*.*8

*6.4.2 Parent..*…………………………….……………………………………………......*.*8

*6.4.3 Class Charges.....*.………………………………………………………………......*.*8

*6.4.4 Exporting Files...*…………..……………………………………………………......*.*9

*6.4.5 Activities/Classes.*………………………………………………………………......*.*9

6.5 ASSUMPTIONS…..………….........…………..……………………………………………..9

**7 SYSTEM REQUIREMENTS...……….……………………………………………………...9**

7.1 DIAGRAMS...………………......…………………………..………………………………...9

*7.1.1 Use Cases………*..………………………………………………………………......*.*9

*7.1.2 ER Diagram*.……………………………………………………………………....*.*15

7.2 SOFTWARE REQUIREMENTS….......……....…………………………………………….15

7.3 SECURITY………………….…......…………..…………………………………………….16

*7.3.1 Passwords*…………..………………………………..………………………….....*16*

*7.3.2 Email Duplicates.*………………….…………………………………………….....16

*7.3.3 File Upload..*………..………………………………..………………………….....*16*

*7.3.4 Permissions*……..…………...…….…………………………………………….....16

7.4 PERFORMANCE SPECIFICATION....……....…………………………………………….16

7.5 REQUIRED CLASSES.…….…......…………..…………………………………………….17

*7.5.1 Configuration Classes*………………………………..………………………….....*17*

*7.5.2 Controller Classes*………………….………………..………………………….....*17*

*7.5.3 Domain Classes*………….…………………………..………………………….....18

*7.5.4 Model Classes*………………………………………..………………………….....19

*7.5.5 Repository Classes*………………….………………..………………………….....20

*7.5.6 Security Classes*………….…………………………..………………………….....*21*

7.6 CONTRIBUTION TO CLASSES…......……....…………………………………………….21

7.7 CAVEATS…………….…….…......…………..…………………………………………….21

7.8 CODE REUSABILITY……………......……....…………………………………………….21

**8 TESTING……………………………….……………………………………………………..21**

8.1 BOUNDARY CHECKS.……......…………………………..……………………………….21

*8.1.1 Excel Boundary Limits*…………………………………………………………......21

*8.1.2 Payment Boundary*………………..……………………………………………......*22*

8.2 UNIT TEST…………....……......…………………………..……………………………….22

*8.2.1 Available Test*……………………….…………………………………………......22

8.3 INTEGRATION TEST………….....………………………..……………………………….22

*8.3.1 Successful Mappings..*……………….…………………………………………......22

8.4 BLACK BOX TEST………….....…………………………..……………………………….22

*8.4.1 Quality Control…..*………………….…………………………………………......*22*

8.5 WHITE BOX TEST…….……….....………………………..……………………………….23

*8.5.1 User Interface..*…………..………….…………………………………………......*2*3

*8.5.2 Input Files..*………..……..………….…………………………………………......23

*8.5.3 Application Connection*..……...…….…………………………………………......23

8.6 POST-MORTEM ANALYSIS….....………………………..……………………………….23

**1 REFERENCES**

baeldung, W. by: (2021, January 1). *Upload and display Excel files with Spring MVC*. Baeldung. https://www.baeldung.com/spring-mvc-excel-files

baeldung, W. by: (2023, November 19). *One-to-one relationship in JPA*. Baeldung. https://www.baeldung.com/jpa-one-to-one

devs5003. (2023, November 11). *How to upload Excel Data into database using Spring Boot ?*. JavaTechOnline. https://javatechonline.com/how-to-upload-excel-data-into-database-using-spring-boot/#Coding\_Steps\_on\_%E2%80%98How\_to\_Upload\_Excel\_data\_into\_Database\_Using\_Spring\_Boot\_%E2%80%99

Fadatare, R. (2023, March 27). *Spring Boot Commandlinerunner example tutorial*. Java Guides. https://www.javaguides.net/2020/02/spring-boot-commandlinerunner-example.html

Fejér, W. by: A. (2023, October 6). *Many-to-many relationship in JPA*. Baeldung. https://www.baeldung.com/jpa-many-to-many

Heisenberg. (1962, December 1). *Model.addAttribute() parameters*. Stack Overflow. https://stackoverflow.com/questions/40303916/model-addattribute-parameters

*How to - tabs*. How To Create Tabs. (n.d.). https://www.w3schools.com/howto/howto\_js\_tabs.asp

*Interface commandlinerunner*. CommandLineRunner (Spring Boot 3.2.0 API). (n.d.). https://docs.spring.io/spring-boot/docs/current/api/org/springframework/boot/CommandLineRunner.html

Kargopolov, S. (2022, September 25). *Spring Boot Rest and JPA - save to a database*. Apps Developer Blog. https://www.appsdeveloperblog.com/spring-mvc-and-jpa-save-to-a-database/

Mark Otto, J. T. (n.d.-a). *Buttons*. · Bootstrap. https://getbootstrap.com/docs/4.0/components/buttons/

MDBootstrap snippets - material design for Bootstrap. (n.d.). https://mdbootstrap.com/snippets/standard/mdbootstrap/2920201?view=side

Muhammad Salman FarooqMuhammad Salman Faroo, Bourkadi, Alex R., & Khal DrogoKhal Drogo. (1958, November 1). *Exporting mysql database to excel using Java Code*. Stack Overflow. https://stackoverflow.com/questions/12283850/exporting-mysql-database-to-excel-using-java-code

*Sequence diagram syntax and features*. PlantUML.com. (n.d.). https://plantuml.com/sequence-diagram

Sirgur, B. (2021, August 21). *Spring boot thymeleaf display list of records from database*. B2 Tech. https://bushansirgur.in/spring-boot-thymeleaf-display-list-of-records-from-database/

*Spring Boot Jpa example with Eclipse and Maven*. LearnJava. (2021, March 28). https://learnjava.co.in/springboot-jpa-example-with-eclipse-and-maven/

*Spring Boot Web Application Upload File*. Simple Solution. (n.d.). https://simplesolution.dev/spring-boot-web-application-upload-file/

*Spring Data JPA3.2.0*. Spring Data JPA. (n.d.). https://spring.io/projects/spring-data-jpa

Spring Security - how to fix websecurityconfigureradapter deprecated. (n.d.). https://www.codejava.net/frameworks/spring-boot/fix-websecurityconfigureradapter-deprecated

*Window history.back()*. History back() Method. (n.d.). https://www.w3schools.com/jsref/met\_his\_back.asp

**2 COMPLETION**

**Completion Status: Incomplete**

**2.1 Complete Modules**

The following modules have been completed in this version:

* Class List Structure
* Scheduling Structure
* Assignment Structure
* Payment/Transaction Structure
* Exporting Files
* Database Backup

**2.2 Incomplete Modules**

2.2.1 Future Implementation

The following modules have not been completed.

* Credit/Debit Payment
* Parents making payments
* Logging deactivated users
* Calculating Class Charges
* Child Attendence

**3 PROBLEM EXPLANANTION**

**3.1 Overview of the System**

The Dance Registration System is a beginning software that is targeted towards dance studios and supplying a proficient system allowing registration of users, tracking users, payments, and activities. The system provides a button-based UI to ensure it is user friendly. The process allows for five different types of users: Admins, Managers, Instructors, Parents, and Children. All users have respective dashboards allowing for different functionalities based on the user role.

**4 DESCRIPTION SOLUTION**

**4.1 Overview of the Solution**

The solution, at this time, has not been finalized. Currently, users can be created, assigned to classes, and displayed on the UI for specific users to see. Payments can be made by the manager and balance and transactions can be displayed to the parents.

**5 SPECIFICATION REQUIREMENTS**

**5.1 Description**

The Dance Registration System is a beginning software that is targeted towards dance studios and supplying a proficient system allowing registration of users, tracking users, payments, and activities. The system provides a button-based UI to ensure it is user friendly. The process allows for five different types of users: Admins, Managers, Instructors, Parents, and Children. All users have respective dashboards allowing for different functionalities based on the user role.

**5.2 Broad Objectives**

5.2.1 Functioning UI for Registration

The UI allows the manager users to create/register new users. They can then assign them their user-roles.

5.2.2 Scheduling Functionalities

Once a child is enrolled in the studio, the Manager can then assign them to specific classes/activities. They can determine the type of class, class level, and studio where the child will participate in the class.

5.2.3 Parent-Child Visibilities

The parent has the capabilities to see all of their child/children’s activities. They can see the classes they are enrolled in and the specific time and place said class will occur.

**6 SCOPE**

**6.1 Products**

6.1.1 User Interface

The UI runs on the localhost server. It allows all the functionalities to be performed.

6.1.2 Database

The database is populated by loading Excel files into the program, through the UI. Once populated, the data stored can be displayed on the UI.

**6.2 Items Addressed**

6.2.1 Database Backup

The database backup can be performed on the UI. This can only be preformed by the administrative users.

6.2.2 Exporting Files

There are five excel files to be loaded into the program. Each of the files can also be exported. They can be found in: /fall2023registrationsystem/ProgramDocuments/Export Files

6.2.3 Registering Users

Managers can register all of the users. At the time of registration, they will assign the user-role, in turn granting the new user specific privileges.

6.2.4 Assigning Students to Classes

After the manager registers the new children users, they can then assign them to the requested classes. They will assign the class, class level, and studio where the class will be taught.

6.2.5 Removing Activities and Students

Managers have the privileges to remove students from activites and to remove activities from the studio.

6.2.6 Track Payments

The managers can view all of the balances for every parent. The parent can view their balance and all of their transaction history.

**6.3 Recommended Features**

6.3.1 Saving Logs of Deactivated Users

The database should save a log of all the users once enrolled in classes at the studios.

6.3.2 Calculating Class Charges

The balance should be calculated by accounting for all the classes he children attended per month, whilst factoring in discounts.

6.3.3 Parent Payment

The parent users should be able to make payments on their respective accounts.

6.3.4Child Attendance

The instructors should be able to take attendance of every class they teach and that attendance should be logged in the database.

**6.4 Terms**

6.4.1 Student/Child/Children

The users who will be enrolled in classes.

6.4.2 Parent

Parent users of the Children.

6.4.3 Class Charges

The cost per taking a class. These differ based on level and discounts are offered for multi-classes and siblings.

6.4.4 Exporting Files

For each excel file uploaded, they can also be exported from the database.

6.4.5 Activities/Classes

The courses offered at the studios for students to be enrolled in and instructors to teach.

**6.5 Assumptions**

The program is assumed to be working up to date.

**7 SYSTEM REQUIRMENTS**

**7.1 Diagrams**

7.1.1 Use Cases

Admin Management

Access Dashboard

**Preconditions**: admin is logged into the Dance Registration System

**Activity**: system validates admin’s credentials

**Postconditions**: system returns Admin Dashboard, has access to admin functionalities

Create Manager/Admin

**Preconditions**: admin is logged into the Dance Registration System, selects Create User

**Activity**: admin inputs required information and submits the form

**Postconditions**: system validates information, creates profile, user can access system

Export Data

**Preconditions**: admin is logged into the Dance Registration System, selects Export Data

**Activity**: admin chooses desired export file and initiates the process

**Postconditions**: system generates file containing selected data in Excel format

Delete User

**Preconditions**: admin is logged into the Dance Registration System, selects Delete

**Activity**: admin selects target user and confirms the deletion

**Postconditions**: system removes profile and associated data, user no longer has access

Reset Password

**Preconditions**: admin is logged into the Dance Registration System, selects Reset

**Activity**: admin selects target user and initiates password reset

**Postconditions**: system generates temporary password for user to use to reset password

Parent Interaction

Access Dashboard

**Preconditions**: parent is logged into the Dance Registration System

**Activity**: system validates parent’s credentials

**Postconditions**: system returns Parent Dashboard, has access to parent functionalities

Select Child Account

**Preconditions**: parent is logged into the Dance Registration System, selects Child

**Activity**: system displays list of linked child accounts, parent chooses which child

**Postconditions**: system loads information about child and their associated activities

View Child Activities

**Preconditions**: parent selects a specific child account

**Activity**: system display summary of child’s information as well as their activities

**Postconditions:** parent gains insight into their child’s scheduled activities

Instructor Interaction

Access Dashboard

**Preconditions**: instructor is logged into the Dance Registration System

**Activity**: system validates instructor’s credentials

**Postconditions**: system returns Instructor Dashboard, has access to functionalities

See Studio Schedule

**Preconditions**: instructor is logged into the Dance Registration System, selects Schedule

**Activity**: system displays schedule of all activities and information within the studio

**Postconditions**: instructor gains insight on when/where activities are occurring

See Class List

**Preconditions**: instructor is logged into the Dance Registration System, selects Class List

**Activity**: system displays list of students enrolled in each class and relevant information

**Postconditions:** instructor can see what students are in what classes, including their own

See Associated Activities

**Preconditions**: instructor is logged into the Dance Registration System

**Activity**: system displays list of activities the instructor teaches

**Postconditions:** instructor gains insight into activities they teach, including location

|  |  |
| --- | --- |
| **Admin Interactions** | A diagram of a system  Description automatically generated |
| **Instructor Interactions** | A diagram of a instructor  Description automatically generated |
| **Parent Interactions** | A diagram of a parent  Description automatically generated |
| **Manager Interactions** | A diagram of a manager  Description automatically generated |
| A diagram with text on it  Description automatically generated with medium confidence | |
| **Transactions** | A diagram of a person's process  Description automatically generated |
| A diagram of a payment process  Description automatically generated |
| **User Registration** | A diagram of a software process  Description automatically generated |
|  |  |

7.1.2 ER Diagram

A diagram of a data flow

Description automatically generated

**7.2 Software Requirements**

* Intel® Core™ Processors 8th Generation or newer
* Microsoft Windows 11 or Windows Server 2022 or newer
* Eclipse IDE for Enterprise Java and Web Developers v2023-09 R
* MySQL Workbench v8.0.34

**7.3 Security**

7.3.1 Passwords

Users are required to change their passwords upon the first logon. every account is set with a default password of “1234,” which is an easily hackable password. After attempting to log in with the default password, the system will prompt the user to create a new password. The account cannot be logged into until the password is changed from the default. Not only does the system require users to create a new password, but the password must follow specific security constraints. Passwords must be at least eight characters long, have an upper and lowercase letter, at least one number, and at least one symbol. Requiring a password with those constraints is a good practice to reduce the likelihood of an attack.

7.3.2 Email Duplicates

Duplicate email addresses are an easy way for an attacker to gain unauthorized access to someone else’s account. Because of this concern, the system does not allow two users with the same email address to register. A check is implemented that throws an error if a user tries to register with an email that already has an account associated with it. By implementing this check, the chance of account hijacking and data exposure is lessened.

7.3.3 File Upload

The program checks the excel files prior to reading them into the database. The first value in the first cell, A1, must match the hard-coded value defined in the program. If matched, then the program will proceed to read in the data.

7.3.4 Permissions

To ensure that the qualified personnel are the only users granted access to the given information, the WebSecurityConfig File contains rules that only permit certain users with their granted roles upon login. This is implemented by restricting access to specific logons. Once an administrator is logged in, all mapping that follows is only accessible by the administrator. The same logic follows for Managers. These are the only two roles that have the permissions allowing them to edit, modify, and delete data and information. Users that try to access webpages that are not granted access to receive a 403 permission denied error in return.

**7.4 Performance Specification**

At this time, the program does not require much optimization. One major speed/time optimization is found when loading the larger excel files. These will take approximately 2-3 seconds longer than smaller files. The charges excel file is the smallest, for reference.

**7.5 Required Classes**

7.5.1 Configuration Classes

AdminConfig.java

Configures the initial setup for admin and users by checking and creating an admin if none exists.  It utilizes the admin and user repositories to interact with the database and ensures presence of both Admin and User entities with predefined email and password.

ContextConfig.java

Configures and provides beans for Apache HTTP client and request configuration, allowing for customized settings for HTTP communication within the system.

WebConfig.java

Enables Cross-Origin Resource Sharing (CORS) by specifying allowed origins, methods, headers, and allowing credentials, ensuring secure communication with client origin. 

WebSecurityConfig.java

Configures security settings that define authentication providers, password encoding, and role-based access controls.  Specifies access permissions for URLs based on user roles, handles login/logout functionality, and manages session creation policies.

7.5.2 Controller Classes

AccountController.java

Handles user account-related operations, including viewing account details, updating user information, and managing password changes with validation.

AdminController.java

Manages administrative functionalities, including displaying user role breakdowns, creating users with various roles, handling user deletions, resetting passwords, and performing database backups.

AssignChildController.java

Manages assignment of children to specific activities, handles the display of assignment forms, processing forms, and submissions, as well as updates the child assignment details.

DisplayController.java

Manages display of various information, including activities, children, and parent balances, as well as displaying information in regard to the admin, managers, and instructors.

ErrorController.java

Handles custom 403 errors, providing mapping to specific HTML page when access to resource is forbidden.

ExportController.java

Handles export requests, providing methods to export the Excel files for admin, instructor, charges, parent/child, and activity data.

LogonController.java

Handles user registration, login, and redirects user to their respective dashboard depending on their roles.  Manages password change for first-time logins and provides additional functionalities such as determining user roles and checking email existence in specific tables

ManagerController.java

Handles manager dashboard, creating and managing instructors, parents, and children, exporting data, selecting parents and children, and handling class-related actions such as deleting and updating activities.

ParentController.java

Handles parent dashboard, selecting child to display activities, and displaying activities for selected child.

TransactionController.java

Manages payment-related operations, including displaying payment form, processing cash and check payments, and updating parent and transaction details accordingly.

UploadController.java

Manages uploading and processing of Excel files containing data related to activities, admins, managers, charges, instructors, and parent/children, as well as populating the corresponding repositories and database tables.

7.5.3 Domain Classes

Activities.java

Entity class storing information about activities, including but not limited to activityID, start/end times, location, and instructor.

Admin.java

Entity class storing information about admin, including but not limited to adminID, first/last name, email, and password.

Charges.java

Entity class storing information about charges, including but not limited to classLevel, flat rates, and multi-child discounts.

ChildId.java

Defines a composite key for specific child information, consisting of the childID, activityID, and activityLevel.

Children.java

Entity class storing information about children, including but not limited to childID, age, parentID, and contains a mapping to the composite key defined in the “ChildId” class.

Instructor.java

Entity class storing information about instructors, including but not limited to instructorID, first/last name, classes taught, and password.

Location.java

Entity class storing information about studio locations, including but not limited to studioID, address, and phone number.

Manager.java

Entity class storing information about managers, including but not limited to managerID, first/last name, email, and password.

Parent.java

Entity class storing information about parents, including but not limited to parentID, secondary contact, and account balance.

Transactions.java

Entity class storing information about transactions, including but not limited to transactionID, amount, and payment type.

7.5.4. Model Classes

CustomUserDetails.java

Implements *UserDetails* interface to provide custom user details, including authorities and information associated with the *User* class.

User.java

Entity class storing information about users, including but not limited to id, email, password, and first/last name.

7.5.5 Repository Classes

ActivitiesRepository.java

Repository for *Activities* entity, providing methods to interact with the database.

AdminRepository.java

Repository for *Admin* entity, providing methods to interact with the database.

ChargesRepository.java

Repository for *Charges* entity, providing methods to interact with the database.

ChildRepository.java

Repository for *Children* entity, providing methods to interact with the database

InstructorRepository.java

Repository for *Instructor* entity, providing methods to interact with the database.

LocationRepository.java

Repository for *Location* entity, providing methods to interact with the database.

ManagerRepository.java

Repository for *Manager* entity, providing methods to interact with the database.

MapCustomUserRepository.java

Custom implementation of the *UserRepository* to store and retrieve user information.

ParentRepository.java

Repository for *Parent* entity, providing methods to interact with the database.

TransactionRepository.java

Repository for *Transaction* entity, providing methods to interact with the database.

UserRepository.java

Repository for *User* entity, providing methods to interact with the database

7.5.6 Security Classes

PasswordConstraintValidator.java

Custom implementation of the *ConstraintValidator* to validate password constraints using the Passay library.

ValidPassword.java

Marks fields or types that need to be validated against specific password constraints.  Associated with the *PasswordConstraintValidator* class for specifying the rules.

**7.6 Contribution to Classes**

Estimating the percentages of code contributed from open-source, 75% open-source, whilst 25% of code was raw coded.

**7.7 Caveats**

It is important to note that the excel file containing the Admin, Manager, and Location, must be uploaded into the program first. This is crucial because the manager is the only user with the capabilities to upload all other files.

Once the Admin file is uploaded, the following order of uploading files is not specific.

**7.8 Code Reusability**

The reusability of the code throughout the program is quite extensive. Most actions had been preformed multiple times, however they needed different mappings. The logic on the excel file upload, for example, is reused for all the file uploads. The deletion logic is used for both activities and children.

**8 TESTING**

**8.1 Boundary Checks**

8.1.1 Excel Boundary Limits

As any business would aspire to continually grow, the size of the excel files uploaded has been dynamically assigned. It has been assumed that the number of children, parents, instructors, etc. will grow, therefore the sizes are found by the number of rows and columns.

8.1.2 Payment Boundary

When making a payment on a parent’s account, there is the possible to overpay. To accommodate this, the parent’s account is credited the overpayment. This will show as a negative balance, meaning they do not owe that amount. The calculations of balances should take this into account in future work.

**8.2 Unit Test**

8.2.1 Available Tests

The Unit Tests can be found for their respective files in the following packages accordingly:

src/test/java…

* edu.sru.thangiah.group2.fall2023registrationsystem.controller.test;
* edu.sru.thangiah.group2.fall2023registrationsystem.domain.test;
* edu.sru.thangiah.group2.fall2023registrationsystem.service.test;
* edu.sru.thangiah.group2.fall2023registrationsystem.model.test;
* edu.sru.thangiah.group2.fall2023registrationsystem.LoginPage.test;
* edu.sru.thangiah.group2.fall2023registrationsystem.test
* edu.sru.thangiah.group2.fall2023registrationsystem.config.test
* edu.sru.thangiah.group2.fall2023registrationsystem.security.test

**8.3 Integration Test**

8.3.1 Successful Mappings

To navigate throughout the UI, the user must click through buttons. All of the buttons correctly and successfully correspond to their mappings at this time. The data passed through these are successfully carried.

**8.4 Black Box Test**

8.4.1 Quality Control

The excel files are parsed and loaded into the program. Once loaded they make the correct associations to display the data on the HTML. Through the mappings the data carried through correctly at this time.

**8.5 White Box Test**

8.5.1 User Interface

Extensive testing has and will continue to be executed on the program to ensure successful functionality across all areas of the UI.

8.5.2 Input Files

The Admin’s file to upload is the first required upload. This uploads enrolls managers into the program. Managers can then upload all other files in any order.

8.5.3 Application Connection

At this time, all applications connect successfully when the program is executed. The database connection is automated successfully and the localhost connection is permitted while the eclipse project is steadily running.

**8.6 Post-Mortem Analysis**

After reflecting on the implementation of this project, we agree that we could have worked more vigorously, and thoroughly. Additionally, we agree that delegation of tasks could have been better managed, along with timeliness of the completion of those tasks.