Dance Registration System

Technical Manual

Group Number: 2

CPSC 488 Section 01

**TABLE OF CONTENTS**

**1 OVERVIEW………………………..…………………………………..………………………4**

**2 SYSTEM REQUIREMENTS………...………………………………..……………………...4**

**3 ASSUMPTIONS…………………..…………………………………………………………....4**

**4 INSTALLING PROGRAMS………..……...………………………………………………....4**

4.1 INSTALLING ECLIPSE IDE………………………………………………………………...4

4.2 INSTALLING MYSQL WORKBENCH…...…………...……………………………………5

**5 COPYING EXCEL FILES...………..……...………………………………………………....5**

**6 RUNNING DANCE REGISTRATION SYSTEM...………………………………………....6**

6.1 USING ECLIPSE IDE………………………………………………………………………...6

**7 DANCE REGISTRATION CLASSES....…………………….……………………………....7**

7.1 CONFIGURATION CLASSES……………………………………………………….……...7

*7.1.1 AdminConfig.java*………………....……………………………………………......*.*7

7*.1.2 ContextConfig.java*…………………………………………………………….........7

7*.1.3 WebConfig.java*………………….…………………………………………….........7

7*.1.4 WebSecurityConfig.java*……………………………………………………….........8

7.2 CONTROLLER CLASSES…………………………………………………………………...8

*7.2.1 AccountController.java*…………....……………………………………………......*.*8

*7.2.2 AdminController.java*………………....………………………………………........*.*8

*7.2.3 AssignChildController.java*……………….……………………………………......*.*8

*7.2.4 DisplayController.java*………………....………………………………………......*.*8

*7.2.5 ErrorController.java*………………....…………………………………………......*.*8

*7.2.6 ExportController.java*………………....………………………………..………......*.*8

*7.2.7 LogonController.java*………………....………………………………..………......*.*8

*7.2.8 ManagerController.java*………………....…………………………..…………......*.*9

*7.2.9 ParentController.java*………………....…………………………..……………......*.*9

*7.2.10 TransactionController.java*………………..………………….………………......*.*9

*7.2.11 UploadController.java*………………....………..…………….………………......*.*9

7.3 DOMAIN CLASSES……………………………………………………………………….....9

*7.3.1 Activities.java*…………………………....………..…………….………………......*.*9

*7.3.2 Admin.java*……………………………....………..…………….………………......*.*9

*7.3.3 Charges.java*………………………….....………..…………….………………......*.*9

*7.3.4 ChildId.java*…………..………………....………..…………….………………......*.*9

*7.3.5 Children.java*…………………………....………..…………….……………….....10

*7.3.6 Instructor.java*………………...………....………..…………….……………….....10

*7.3.7 Location.java*…………………………....………..…………….……………….....10

*7.3.8 Manager.java*…………………………....………..…………….……………….....10

*7.3.9 Parent.java*……………………………....………..…………….……………….....10

*7.3.10 Transactions.java*……………………....………..…………….……………….....10

7.4 MODEL CLASSES………………………………………………………………………….10

*7.4.1 CustomUserDetails.java…*……………....………..…………….……………….....10

*7.4.2 User.java*………………...……………....………..…………….……………….....10

7.5 REPOSITORY CLASSES……………………………………………………………..…….11

*7.5.1 ActivitiesRepository.java*……………......………..…………….……………….....11

*7.5.2 AdminRepository.java*……………...........………..…………….……………….....11

*7.5.3 ChargeRepositorys.java*……………........………..…………….……………….....11

*7.5.4 ChildRepository.java*……………......………..…..…………….……………….....11

*7.5.5 InstructorRepository.java*……………......………...…..……….……………….....11

*7.5.6 LocationRepository.java*……………......………..….………….……………….....11

*7.5.7 ManagerRepository.java*……………......………..….………….……………….....11

*7.5.8 MapCustomUserRepository.java*……......………..…………….……………….....11

*7.5.9 ParentRepository.java*……………......………..……..……….…………..…….....11

*7.5.10 TransactionsRepository.java*……………......………….…………….…………..11

*7.5.11 UserRepository.java*……………......………..…………………….….………….11

7.6 SECURITY CLASSES………………………………………………………………………12

*7.6.1 PasswordConstraintValidator.java*……..………..…………….……………….....12

*7.6.2 ValidPassword.java*………...……...........………..…………….……………….....12

**8 INTERACTIONS BETWEEN CLASSES....…..…………….……………………………...12**

8.1 ADMIN TO MANAGER INTERACTIONS……………………………………….……….12

8.2 MANAGER TO PARENT INTERACTIONS……………………………………...……….12

8.3 MANAGER TO INSTRUCTOR INTERACTIONS……………………………….………..12

8.4 MANAGER TO CHILD INTERACTIONS………………………………………..………..12

8.5 PARENT TO CHILD INTERACTIONS……………………………………...…………….12

8.6 INSTRUCTOR TO CHILD INTERACTIONS…………………………………….………..13

**9 UML DIAGRAMS ON FUNCTIONALITY………………………………………………..13**

9.1 USE CASE CONDITIONS..…………………………………………………………..…….13

*9.1.1 Admin Management*……………..............………..…………….……………….....13

*9.1.2 Parent Interaction*…………….................………..…………….……………….....14

*9.1.3 Instructor Interaction…*……………........………..…………….……………….....14

1 OVERVIEW

This document provides a comprehensive description of the necessary setup for the successful execution of the Dance Registration System.

2 SYSTEM 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

3 ASSUMPTIONS

* It is assumed that all system requirements have been met except for the installation of the MySQL Workbench and the Eclipse IDE.
* It is also assumed that the Dance Registration System has been pre-packaged on a storage medium or downloaded from GitHub.

4 INSTALLING PROGRAMS

**4.1 Installing Eclipse IDE 2023-09 R**

Eclipse IDE for Enterprise Java and Web Developers can be downloaded freely online. In order to get Eclipse, follow the steps:

1. Download Eclipse IDE 2023-09 R from <https://eclipse.org/downloads>
2. Run the Eclipse Installer
3. Select the “Eclipse IDE for Java Developers” package to install
   1. Select your installation folder
   2. The default folder will be in your User directory
4. Installer may request that the computer be rebooted

**4.2 Installing MySQL Workbench v8.0.34**

MySQL Workbench can be downloaded freely online. In order to get MySQL, follow the steps:

1. Download MySQL Workbench v8.0.34 from <https://dev.mysql.com/downloads/workbench/>
2. Run the MySQL Installer
   1. An account with Administrator or Power User privileges is required
3. In the setup page, select the COMPLETE installation
   1. Unless you choose otherwise, MySQL Workbench is installed in the User directory
4. Installer may request that the computer be rebooted

5 COPYING EXCEL FILES

At this point, the Dance Registration Excel files should be copied from the medium (or downloaded from the GitHub) and saved onto the Hard Disk of the machine. In order to load the Excel files into the system, they will need to be extracted from the program and put into an accessible location. The Excel files correspond as listed:

* Admin uploads:
  + AdminManagerLocation.xlsx
* Manager uploads:
  + Activities.xlsx
  + Charges.xlsx
  + Instructor.xlsx
  + ParentChild.xlsx

The populated data in the Excel files can be altered, if the column orders stay the same. If an alternate Excel file needs to be uploaded, there needs to be a tag signifying which location the file belongs:

* ADMIN for AdminManagerLocation.xlsx
* ACTIVITIES for Activities.xlsx
* CHARGES for Charges.xlsx
* INSTRUCTOR for Instructor.xlsx
* PARENTCHILD for ParentChild.xlsx

6 RUNNING DANCE REGISTRATION SYSTEM

**6.1 Using Eclipse IDE 2023-09 R**

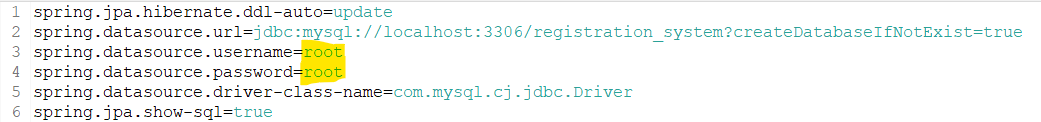
To install and set up this program:

1. Select the appropriate directory where the Dance Registration System resides
   1. Verify that the pom.xml file is contained in the workspace

**A screenshot of a computer

Description automatically generated**

1. Locate the ‘applications.properties’ file from “src/main/resources” and modify and username and password to reflect the database credentials



1. Locate the ‘DatabaseBackupService.java’ file from “src/main/java/edu.sru.thanigah.group2/fall2023registrationsystem.service” and modify the username and password to reflect the database credentials

A close-up of a computer code

Description automatically generated

1. After the user database credentials are properly set, the program can be launched

To launch the program:

1. Right click the “Fall2023registrationsystemApplication.java” file
2. Hover over “Run As...”
3. Select Spring Boot Application

A screenshot of a computer

Description automatically generated

7 DANCE REGISTRATION CLASSES

**7.1 Configuration Classes**

7.1.1 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.

7.1.2 ContextConfig.java

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

7.1.3 WebConfig.java

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

7.1.4 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.

**7.2 Controller Classes**

7.2.1 AccountController.java

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

7.2.2 AdminController.java

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

7.2.3 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.

7.2.4 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.

7.2.5 ErrorController.java

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

7.2.6 ExportController.java

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

7.2.7 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.

7.2.8 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.

7.2.9 ParentController.java

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

7.2.10 TransactionController.java

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

7.2.11 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.

**7.3 Domain Classes**

7.3.1 Activities.java

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

7.3.2 Admin.java

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

7.3.3 Charges.java

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

7.3.4 ChildId.java

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

7.3.5 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.

7.3.6 Instructor.java

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

7.3.7 Location.java

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

7.3.8 Manager.java

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

7.3.9 Parent.java

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

7.3.10 Transactions.java

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

**7.4 Model Classes**

7.4.1 CustomUserDetails.java

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

7.4.2 User.java

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

**7.5 Repository Classes**

7.5.1 ActivitiesRepository.java

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

7.5.2 AdminRepository.java

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

7.5.3 ChargesRepository.java

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

7.5.4 ChildRepository.java

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

7.5.5 InstructorRepository.java

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

7.5.6 LocationRepository.java

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

7.5.7 ManagerRepository.java

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

7.5.8 MapCustomUserRepository.java

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

7.5.9 ParentRepository.java

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

7.5.10 TransactionRepository.java

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

7.5.11 UserRepository.java

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

**7.6 Security Classes**

7.6.1 PasswordConstraintValidator.java

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

7.6.2 ValidPassword.java

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

8 INTERACTIONS BETWEEN CLASSES

**8.1 Admin to Manager Interactions**

* Admins handle the registration of new managers in the system
* Admins can reset and delete manager accounts

**8.2 Manager to Parent Interactions**

* Managers handle the registration of new parent accounts
* Managers provide information about activities, schedules, and locations to parents
* Managers handle inputting payments on parent accounts
* Managers can track the parent’s account balance

**8.3 Manager to Instructor Interactions**

* Managers are responsible for assigning instructors to activities
* Managers oversee when different activities occur

**8.4 Manager to Child Interactions**

* Managers handle the registration of new children
* Managers handle assigning children to new classes
* Managers can remove children from activities

**8.5 Parent to Child Interactions**

* Parents can view their child’s data like age, birthdate, etc.
* Parents can see what activities their children are enrolled in
* Parents can see the account balance, which is associated with how many activities their child has attended

**8.6 Instructor to Child Interactions**

* Instructors can see what classes they are teaching
* Instructors can view the class list to see what children are in what classes

9 UML DIAGRAMS ON FUNCTIONALITY

The following UML diagrams will help better

explain the functionalities of the Dance Registration System, as well as the interactions between various agents.

For class diagrams, see the PlantUML > Class folder.

**9.1 Use Case Conditions**

9.1.1 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

9.1.2 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

9.1.3 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** | |  |
|  | | |
| **Transactions** |  | |
|  | |
| **User Registration** |  | |
|  |  | |