Web-Based Evaluations

Final Report

Class CPSC 488 Section 01

Group Number: 5

Revised: 11/22/2023

Originally Created: 4/25/2022

Tanuj Rane [txr1029@sru.edu](mailto:txr1029@sru.edu), ranetanuj@gmail.com

Dalton Stenzel [drs1030@sru.edu](mailto:drs1030@sru.edu), daltonrstenzel@gmail.com

Logan Racer [lsr1006@sru.edu](mailto:lsr1006@sru.edu), loganracer27@gmail.com

Anthony Cinicola [ajc1033@sru.edu](mailto:ajc1033@sru.edu)

Duncan Lawrence [dal1017@sru.edu](mailto:dal1017@sru.edu) duncanlawrence2000@gmail.com

J Abbigail Rowe [jhr1002@sru.edu](mailto:jhr1002@sru.edu) jakerowe@protonmail.com

Zachary Freilino zlf1001@sru.edu, zfreilino@gmail.com

David Gillette dgg1001@sru.edu, davidgsgillette@gmail.com

Mike Elias [mwe1002@sru.edu](mailto:mwe1002@sru.edu)

Chad Green [cwg1003@sru.edu](mailto:cwg1003@sru.edu)

Michael Mirabito [mpm1012@sru.edu](mailto:mpm1012@sru.edu)

Revised By:

Chris Thornton [cpt1003@sru.edu](mailto:cpt1003@sru.edu)

Frank Nkurunziza [fxn1001@sru.edu](mailto:fxn1001@sru.edu)

Ari Hilscher [cgh1004@sru.edu](mailto:cgh1004@sru.edu)

Contents

[All Sources Used 3](#_gjdgxs)

[Previous Groups 3](#_lfujra1tp9r5)

[Current Group 4](#_puv51gz1w28)

[Completion 4](#_30j0zll)

[Tasks that Were Completed 4](#_6qru7v5dfznh)

[Tasks that Were Not Completed 8](#_6fdmwb313coj)

[Contribution 9](#_1fob9te)

[Glossary 10](#_3znysh7)

[Problem Explanation 10](#_2et92p0)

[System Requirements 11](#_1eo16nhr6381)

[Use Case Diagrams 12](#_g7g3oeoyrxkb)

[Use Case of the Evaluator Role 12](#_yxsi55u289kd)

[Use Case of the SuperUser/Admin Role 13](#_n193b7lnlpcg)

[Use Case of the SuperSuperUser Role 14](#_hs5e5q2r3i4r)

[Use Case of the Evaluator Admin 15](#_kqzs2vlbickh)

Class Diagrams 16

[Sequence Diagrams 16](#_twni0xj5wdbl)

[CompanySuperUser on the Manage Users Page 16](#_dweb01u72ezu)

[CompanySuperUser on the Manage Roles Page 16](#_x0leaf9f67jk)

[Evaluator Edit Account Information 17](#_s3zxq7m584hr)

[Evaluator Perform Evaluation 17](#_hwt02z6ag0j)

[Uploading Evaluation Forms 18](#_5dtgp6h1hhbk)

[Uploading Groups 19](#_cuihby1a4lyh)

[Uploading Companies 20](#_fois49ropbb8)

[Entity/Relationship Diagram 20](#_3wi1756cbjsv)

[Caveats/Minefields 20](#_tyjcwt)

[File Path Names 21](#_6qa733c2ehhu)

[Code Reusability 21](#_3dy6vkm)

[Testing 22](#_1t3h5sf)

[Logger 22](#_gh5ul98y9j8)

[Deployment/Maintenance 23](#_g0xbaqj53koc)

[Post-Mortem Analysis 23](#_4d34og8)

# All Sources Used

## Previous Groups

[1] Baeldung *Get All Data From a Table With Hibernate*. <https://www.baeldung.com/hibernate-select-all>

[2] W3school. Bootstrap https://www.w3schools.com/bootstrap/bootstrap\_ref\_all\_classes.asp

[3] Baeldung *Spring Security Roles and Privileges* <https://www.baeldung.com/role-and-privilege-for-spring-security-registration>

[4] <https://www.baeldung.com/spring-boot-testing>

[5] <https://spring.io/guides>

[6] <https://reflectoring.io/spring-boot-test/>

[7] <https://www.arhohuttunen.com/spring-boot-webmvctest/>

[8] <https://semaphoreci.com/community/tutorials/stubbing-and-mocking-with-mockito-2-and-junit>

[9] <https://stackoverflow.com/questions/41770156/spring-add-custom-user-details-to-spring-security-user>

[10] <https://stackoverflow.com/questions/14268451/spring-security-userdetailsservice-implementation-login-fails>

[11] <https://howtodoinjava.com/spring-security/inmemory-jdbc-userdetails-service/>

[12] <https://www.baeldung.com/spring-security-authentication-with-a-database>

[13] <https://www.baeldung.com/circular-dependencies-in-spring>

[14] <https://stackoverflow.com/questions/46297832/required-a-bean-of-type-org-springframework-security-core-userdetails-userdetai>

[15] <https://github.com/itzg/spring-security-spa/issues/1>

[16] <https://www.javainuse.com/webseries/spring-security-jwt/chap5>

[17] <https://stackoverflow.com/questions/36824973/cant-find-securitymockmvcconfigurers>

[18] <https://www.baeldung.com/spring-security-integration-tests>

[19] <https://stackabuse.com/get-http-post-body-in-spring/>

[20] <https://blog.devgenius.io/spring-boot-deep-dive-on-unit-testing-92bbdf549594>

[21] <https://spring.io/guides/gs/testing-web/>

## Current Group

[22] <https://www.w3schools.com/howto/howto_js_tabs.asp>

[23] <https://stackoverflow.com/questions/57093656/a-bean-of-type-org-springframework-mail-javamail-javamailsender-that-could-not>

[24] <https://stackoverflow.com/questions/521171/a-java-collection-of-value-pairs-tuples>

[25] <https://getbootstrap.com/docs/4.0/components/modal/>

[26] <https://www.w3schools.com/css/css_align.asp>

[27] <https://www.w3schools.com/css/css3_buttons.asp>

[28] <https://www.thymeleaf.org/doc/tutorials/2.1/thymeleafspring.html>

[29] <https://www.thymeleaf.org/doc/tutorials/2.1/thymeleafspring.html#creating-a-form>

[30] <https://spring.io/guides/gs/handling-form-submission/>

[31] <https://www.baeldung.com/spring-mvc>

[32] https://www.baeldung.com/thymeleaf-in-spring-mvc

# 

# Completion

Project Status: Incomplete

Our group corrected a multitude of underlying issues in the program, in addition to enhancing the program with new features and making it more complete as a whole. The following is a compiled list of every change, addition, or fix that we implemented, followed by a list of items that we were unable to complete.

## Tasks that Were Completed

* Access controls for each user role type
  + Super User
  + Admin
  + Evaluator Admin
  + User (reviewee and/or reviewer)
* Group Management options now work such as:
  + Addin users to a group
  + Removing a user from a group
  + Editing a user in a group
  + Moving a user from one group to another
  + Archiving and unarchiving groups
  + Manually creating a group
* Reviewers can request a self-evaluation from a reviewee in any of their groups
* Reviewees can see the groups they belong too
  + Reviewee can fill out a self-evaluation if they have been requested to by a reviewer
* Made excel sheet uploading more efficient
  + Strayed program away from hardcoded values used in the process
* Created an edit groups page
  + Added functionality to all buttons
  + Added access to the group’s evaluators
* Users can edit their account information
  + Controlled it so users can not switch their role or company
  + Made a new page for it
* Created several html pages
  + Maps to endpoints that we created to specific operations
* Created sequence diagrams
* Created use case diagrams
* Created activity diagrams
* Created class diagrams
* Host code with Tomcat.

## Tasks that Were Not Completed

* Create project into a war file
* Allow Super Users to create departments of a company
* Make deadlines for when self-evaluations are due
* Synchronous/Asynchronous self-evaluations
* Evaluators can choose if all the group evaluators or just them can see the self-evaluation
* Create manual add for companies’ page
* Display reviewees current level of report progress
* Company logo for all pages
* Groups and other data are unable to be exported as excel file
* Super Users creating branches of a company

# Contribution

Spring security login/websecurity: 65% from Daily Code Buffer, Java Brains, & Sharma Manish

Front-end to back-end communication: 40% from Thymeleaf & Eugen Paraschiv

Searching/retrieving data from database: 45% from Baeldung, Oliver Gierke, & Thanh Tran

CSS-Bootstrap for web page design: 90% from Mark Otto

Displaying charts on webpage: 80% from ZetCode

General chart generation: 40% from TutorialsPoint

Pdf report generation: 65% from ThinkTibits

Testing Concepts: 90% from [20, 8, 7, 6, 5, 4]

MockMvc: 100% from [21, 17, 7]

Integration Tests: 90% from [21, 20, 19, 18, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 4]

Copy-pasted code in integration tests: 100% from [6]

Copy-pasted code with renamed variables for HTML tabs: 100% from [22]

Copy-pasted code for JavaMailSender class: 100% from [23]

Bootstrap modals: 70% from [25]

Button Styling and Page Alignment: 100% from [26,27]

Form Creation: 40% from [29, 30]

# Glossary

ADMIN - Administrative user, responsible for managing company users and roles.

categoryDataset - Dataset to be used for bar chart and area chart generation.

Company - Collection of locations, departments, administrators, roles, and users representing a business utilizing this service.

Company File - Excel File created to upload companies to the system containing all needed information.

eval-evaluationLog

EVALUATOR - Evaluator user, responsible for completing evaluations

EVAL\_ADMIN - Evaluation Administrator user, responsible for managing evaluation forms and evaluation groups.

EVALUATOR\_EVAL - Evaluator and evaluee/reviewee

EvalGroup - Group of users created that will be evaluated by set evaluators.

Evaluation File - Excel file which contains instructions for the program to assemble an Evaluation Form.

Evaluation Template - Evaluation form containing blank responses from which evaluations are conducted.

id- Identification number associated with most objects in the system such as groups, users, reviewees, evaluators, etc.

pieDataset - Dataset to be used for pie chart and ring chart generation.

rev-reviewee or user assigned to a group.

role- A role is assigned to a user to give them their required permissions in the system.

sort- A particular arrangement of values based on an arrangement term such as by first name, or email.

sortOr -The order in which the arrangement of values is placed in such as ascending or descending.

SuperSuperUser - Administrative user responsible for managing companies and other admins.

USER - Reviewee user otherwise known as the user being evaluated

# Problem Explanation

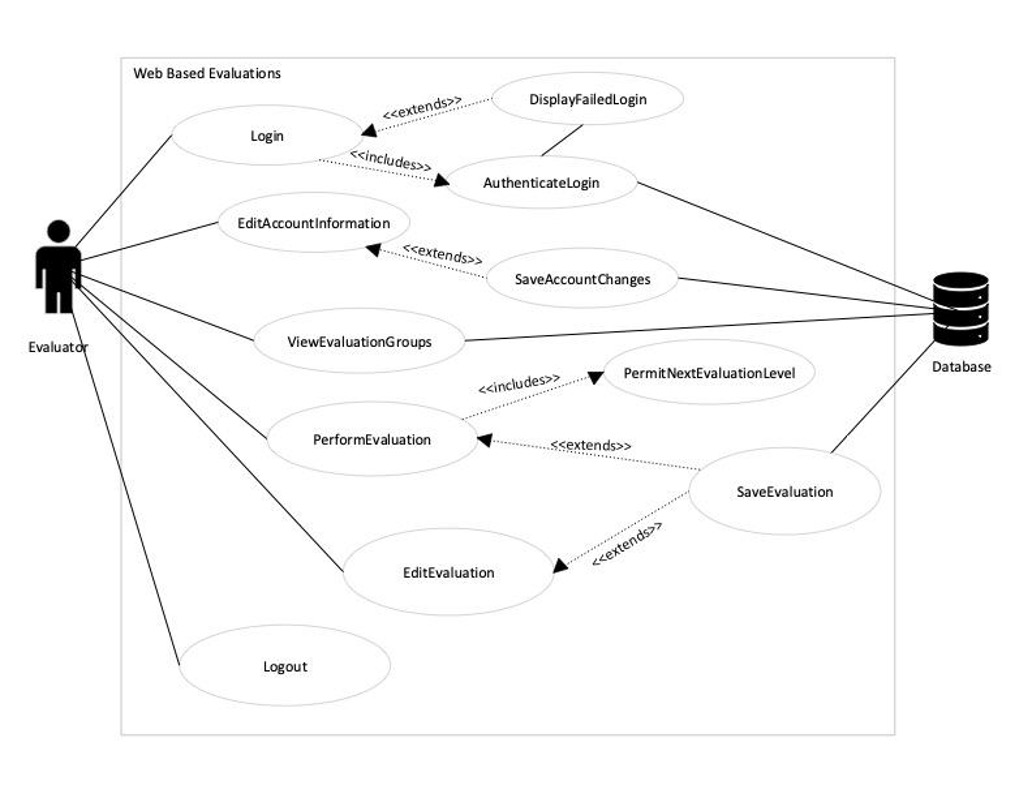
The issue at hand is the need for a system that provides the ability to view and perform evaluations on staff or employees at a company or organization. The requirements include the ability to customize the evaluation criteria and include qualitative and quantitative questions. Evaluations must be appropriately assigned to the correct people in particular companies/schools/etc and must include the option to have several evaluators able to perform evaluations on the same person or group. Administrators with the particular power over evaluations must provide the evaluation forms, groups, and order of evaluations. The ability to view completed evaluations must be present depending on determining factors such as rank. User accounts must be present and will be ranked to have some sort of order and power distribution. Information from the program, such as user data and finished evaluation reports, must be held from within a database. It must also be able to generate analysis based on the completed evaluations, including things such as chart generation, PDF reports, and score analysis spreadsheets.

# System Requirements

* Java 17
* Maven 3.5+
* Gradle 7.5+
* RAM: 128 MB
* Disk space: 124 MB for JRE; 2 MB for Java Update
* Minimum Pentium 2 266 MHz processor
* Browsers: Internet Explorer 9 and above, Microsoft Edge, Firefox, Chrome
* OS X 10.8.3+ or Windows Vista SP2+ or Ubuntu Linux 12.04 LTS+ (or equivalently modern linux distributions)

# Use Case Diagrams

## Use Case of the Evaluator Role



## Use Case of the SuperUser/Admin Role

# 

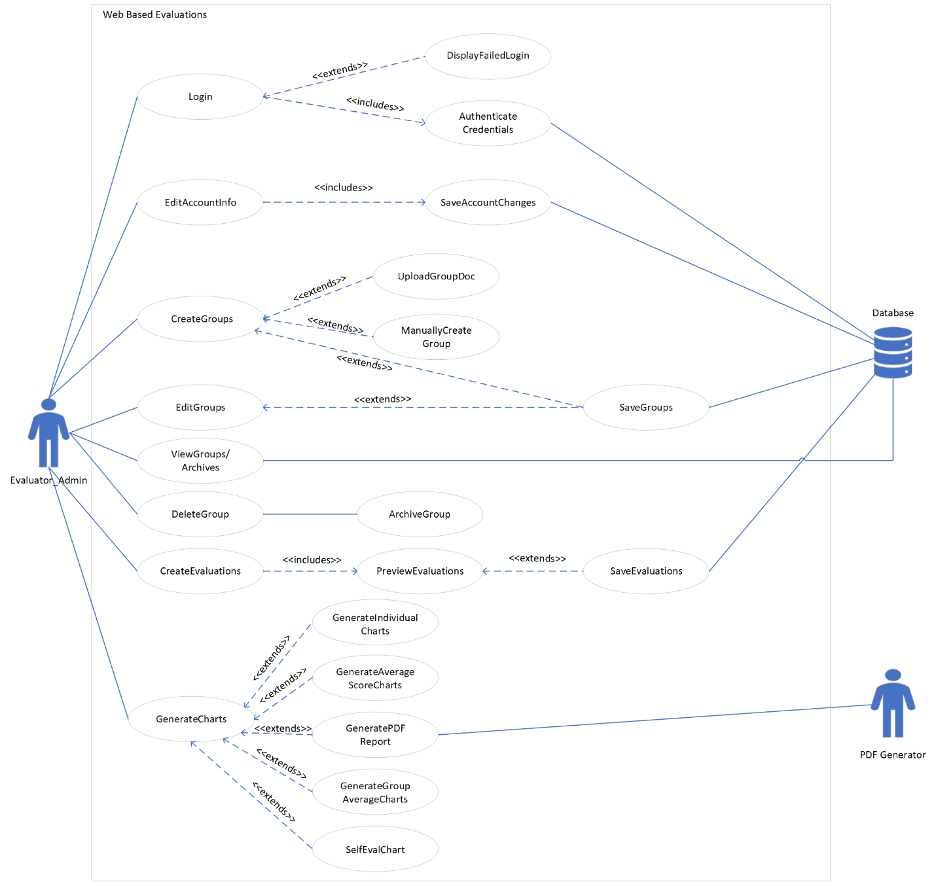
## 

## 

## Use Case of the SuperSuperUser Role

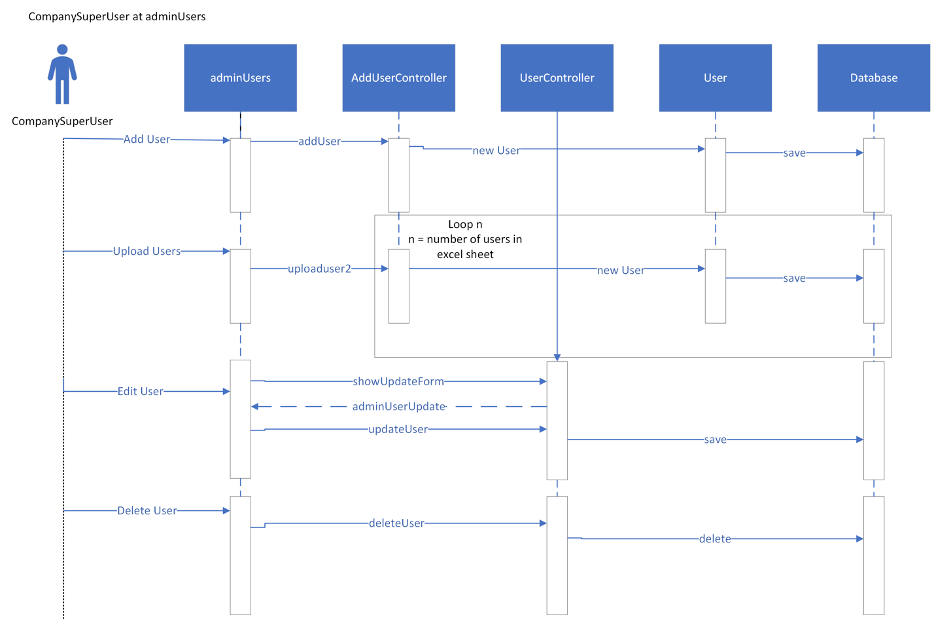


## Use Case of the Evaluator Admin

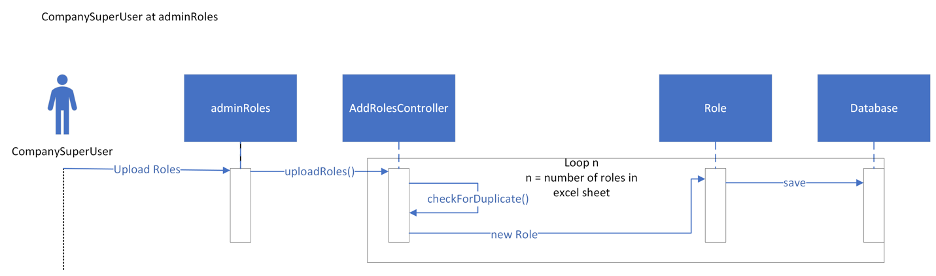


# Sequence Diagrams

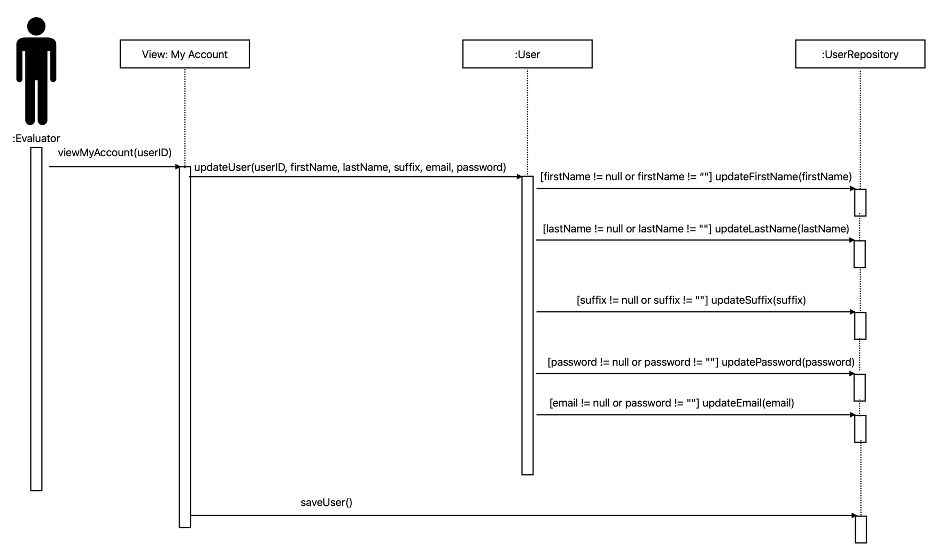
## CompanySuperUser on the Manage Users Page



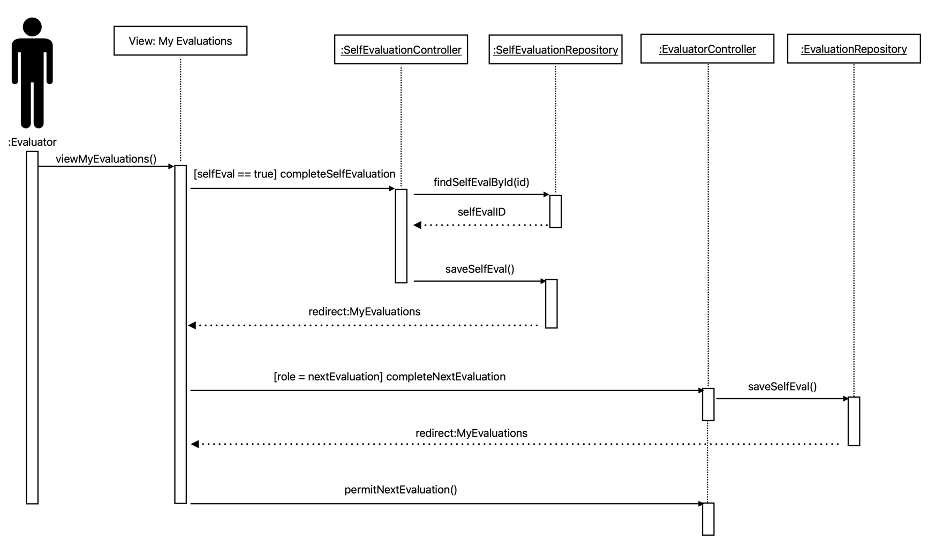
## CompanySuperUser on the Manage Roles Page



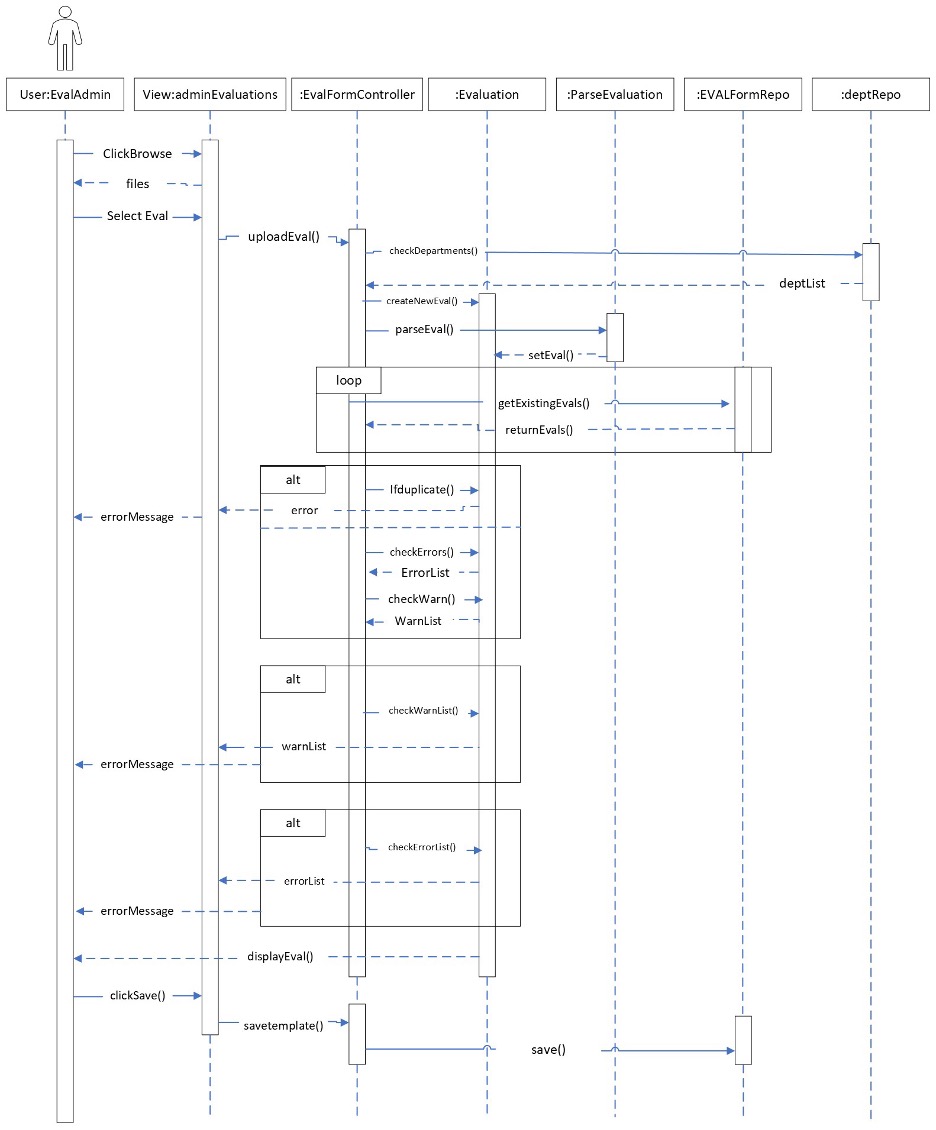
## Evaluator Edit Account Information



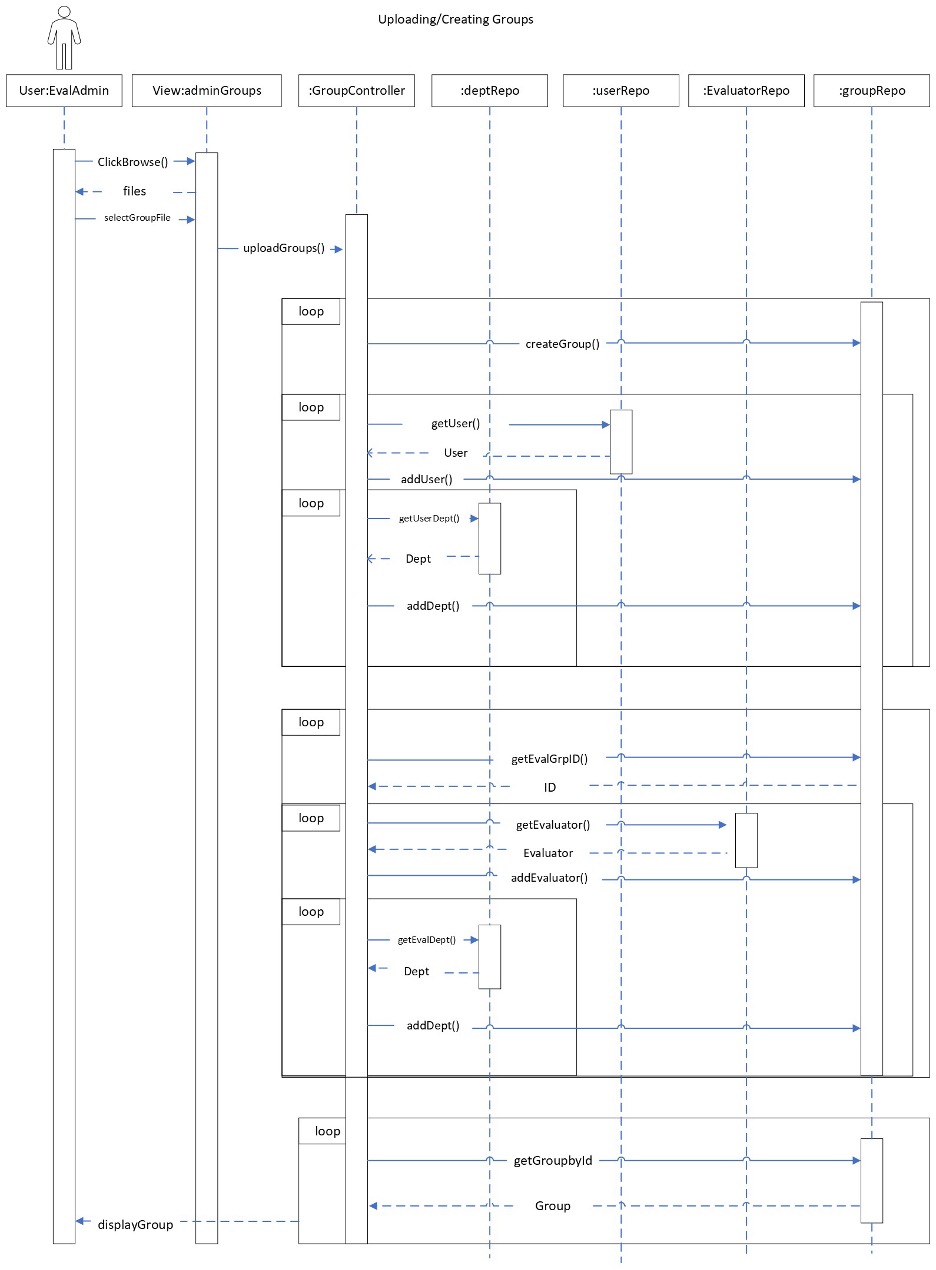
## Evaluator Perform Evaluation



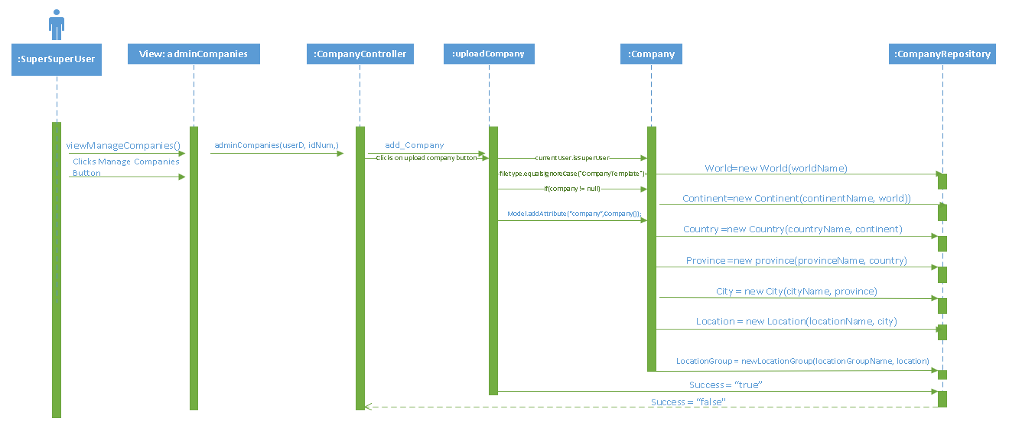
## Uploading Evaluation Forms



## Uploading Groups



## Uploading Companies



## Changing Group Evaluators

A diagram of a change evaluator

Description automatically generated

## Removing User from Group

A diagram of a person's process

Description automatically generated

## Archive Controller

*A diagram of a person

Description automatically generated*

## Updating Companies

A diagram of a flowchart

Description automatically generated

## Uploading Roles

A diagram of a flowchart

Description automatically generated

## Adding Companies

A diagram of a flowchart

Description automatically generated

## Class Diagrams

WebBasedEvaluationsApplication.java

A screenshot of a computer

Description automatically generated

City.java

A screenshot of a computer

Description automatically generated

Company.java

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

Continent.java

A screenshot of a computer

Description automatically generated

Country.java

A screenshot of a computer

Description automatically generated

Department.java

A screenshot of a computer

Description automatically generated

Location.java

A screenshot of a computer

Description automatically generated

LocationGroup.java

A screenshot of a computer

Description automatically generated

Province.java

A screenshot of a computer

Description automatically generated

World.java

A screenshot of a computer program

Description automatically generated

MailSenderConfiguration.java

A screenshot of a computer

Description automatically generated

PDFConfiguration.java

A close-up of a sign

Description automatically generated

SecurityConfiguration.java

A screenshot of a computer

Description automatically generated

AddDepartmentController.java

A screenshot of a computer

Description automatically generated

AddRolesController.java

A screenshot of a computer

Description automatically generated

AddUserController.java

A screenshot of a computer

Description automatically generated

ArchiveController.java

A white rectangular object with black text

Description automatically generatedCompanyController.java

A screenshot of a computer program

Description automatically generated

DataVizualizationController.java

A screenshot of a computer

Description automatically generated

EvalFormController.java

A screenshot of a computer

Description automatically generated

EvaluatorController.java

A white box with black text

Description automatically generated

GroupController.java

A screenshot of a computer

Description automatically generated

HomePage.java

A screenshot of a computer

Description automatically generated

ResetPasswordController.java

A screenshot of a computer error

Description automatically generated

RevieweeController.java

A white rectangular object with black text

Description automatically generated

RolesController.java

A screenshot of a computer

Description automatically generated

SelfEvaluationController.java

A screenshot of a computer

Description automatically generated

UserController.java

A screenshot of a computer

Description automatically generated

Archive.java

A screenshot of a computer program

Description automatically generated

CreateDataset.java

A screenshot of a computer program

Description automatically generated

EvalRole.java

A screenshot of a computer program

Description automatically generated

EvalTemplates.java

A screenshot of a computer program

Description automatically generated

EvaluationLog.java

A screenshot of a computer program

Description automatically generated

Evaluator.java

A screenshot of a computer

Description automatically generated

EvaluatorId.java

A screenshot of a computer

Description automatically generated

Group.java

A screenshot of a computer

Description automatically generated

MyUserDetails.java

A screenshot of a computer

Description automatically generated

PasswordResetToken.java

A screenshot of a computer

Description automatically generated

Privilege.java

A screenshot of a computer

Description automatically generated

ResetPassword.java

A screenshot of a computer

Description automatically generated

Reviewee.java

A screenshot of a computer program

Description automatically generated

Role.java

A screenshot of a computer

Description automatically generated

SelfEvaluation.java

A screenshot of a computer program

Description automatically generated

User.java

A white screen with black text

Description automatically generated A white background with black and red text

Description automatically generated with medium confidence

# Entity/Relationship Diagram

The ER diagram for this project is very large, so it wouldn’t be able to fit on a standard document without making the content unreadable. It is attached as a separate zoomable PDF in the Documents folder of the project.

# Caveats/Minefields

There are a few caveats/minefields that are present on the Manage Users page, such as issues with sorting when logged into a SuperSuperUser and various possible Whitelabel errors as detailed in the Evaluation Manual. In addition, various issues exist with the UI, such as the Groups button disappearing under certain conditions. Creating forms still needs to be implemented on the Evaluation Forms page. Evaluations still need to have the option to be viewed by all group evaluators or just the one that requested it.

In addition to this and other issues at large there are also a variety of small cleanups and fixes that should be addressed in future versions, such as the functionality of back buttons which don’t always redirect to the right page, and making the various tables more appropriately resize for different window sizes.

See the Evaluation Manual for a more detailed overview of the Caveats/Minefields present in the program.

# 

# File Path Names

* WebBasedEvaluations
  + company
    - City.java
    - Company.java
    - Continent.java
    - Country.java
    - Department.java
    - Location.java
    - LocationGroup.java
    - Province.java
    - World.java
  + configuration
    - H2SecurityConfiguration.java
    - MailSenderConfiguration.java
    - PDFConfiguration.java
    - SecurityConfigTest.java
    - SecurityConfiguration.java
  + controller
    - AddDepartmentController.java
    - AddRolesController.java
    - AddUserController.java
    - ArchiveController.java
    - CompanyController.java
    - DataVisualizationController.java
    - EvalFormController.java
    - EvaluatorController.java
    - GroupController.java
    - HomePage.java
    - RegistrationController.java
    - ResetPasswordController.java
    - RevieweeController.java
    - RolesController.java
    - SelfEvaluationController.java
    - UserController.java
  + domain
    - Archive.java
    - CreateDataset.java
    - EvalRole.java
    - EvalTemplates.java
    - EvaluationLog.java
    - Evaluator.java
    - EvaluatorId.java
    - Group.java
    - MyUserDetails.java
    - PasswordResetToken.java
    - Privilege.java
    - ResetPassword.java
    - Reviewee.java
    - revieweelist.java
    - Role.java
    - SelfEvaluation.java
    - User.java
    - VerificationToken.java
  + evalform
    - ComputeRange.java
    - Evaluation.java
    - GenerateEvalReport.java
    - GenerateEvalReportPoi.java
    - ParseEvaluation.java
    - PdfGenarator.java
    - Question.java
    - Section.java
  + excel
    - ExcelRead\_group.java
    - ExcelRead.java
  + model
    - UserModel.java
  + repository
    - ArchiveRepository.java
    - CityRepository.java
    - CompanyRepository.java
    - ContinentRepository.java
    - CountryRepository.java
    - DepartmentRepository.java
    - EvalRoleRepository.java
    - EvaluationLogRepository.java
    - EvaluationRepository.java
    - EvaluatorRepository.java
    - GroupRepository.java
    - LocationGroupRepository.java
    - LocationRepository.java
    - PasswordTokenRepository.java
    - PrivilegeRepository.java
    - ProvinceRepository.java
    - RevieweeRepository.java
    - RoleRepository.java
    - SelfEvaluationRepository.java
    - UserRepository.java
    - VerificationTokenRepository.java
    - WorldRepository.java
  + service
    - AdminMethodsService.java
    - EmailSenderService.java
    - EvaluatorService.java
    - GroupService.java
    - MyUserDetailsService.java
    - UserService.java
    - UserServiceImpl.java
    - VerificationService.java
  + WebBasedEvaluationsApplication.java

# Code Reusability

The software was built with the idea of OOP programming in mind, so the use of code built for reusability is certain. Not all aspects of every class and method provide the most elegant solution for the subject of recycling but there are plenty of instances.

The services package includes classes that contain methods that other classes need in order to properly out what is needed. The UserService is the source for pulling custom user objects from the custom UserRepository class. The User Service class has instances created and used by classes such as the UserController and AddUserController class for obtaining what uses to be displayed and in what format. The UserService class can easily be reused in a situation where there is a different User and UserRepository class as its sole duty is to provide a list of objects from a database.

The chart generation and pdf report generation were built with reusability in mind. The chart generation methods use a method to find the logs that match the id received through the path variable. These logs are then sent to the createDataset class where the section names and information are pulled and placed into a usable dataset. These values are pulled using methods in the evaluation class meaning that no matter the amount of scoring and non-scoring sections the correct values will be pulled. This means that any evaluation template can have their scores converted into a usable dataset. The chart generation page and buttons are displayed based on the group, reviewee, and evaluationLog tables in the database. This allows the list of reviewees on the chart generation page to be dynamic and update when users are added or deleted.

The Evaluation class and related classes lend themselves to reusability. All actions relating to creating and formatting evaluations are encapsulated by the Evaluation class and its subclasses (Section, Question, and ComputeRange). This includes necessary functions such as updateCompute(), processTooltips(), saveResponses(), getSectionByName(), and getQuestionById(). Setters, getters, and List processing methods are also included. Any other project of similar scope could implement these classes into their project with little modification needed. Despite this, there are several utility classes for our specific use case which were delegated to separated classes with implementation specific methods. This includes the ParseEvaluation class which loads the Evaluation object with data from a XML file, and the GenerateEvalReport class which takes evaluations from the database and produces an analysis file.

Each method and class that was created was written in a way where the could be used in multiple ways but still have the same structure and functionality.

# Testing

Much of the testing performed in the project was manual whitebox and blackbox testing. The whitebox testing varied depending on the group member, but a common approach was to print out information to the console based on input that was sent to the code. This allowed us to get an understanding of how the code responded to the input. Blackbox testing, on the other hand, was more represented visually. For example, if we uploaded a file, did the user interface correctly respond? If we clicked on a button, did it present the output as expected? Since a majority of our testing looked like this, not all possibilities for testing what could go wrong were able to be covered. This was certainly not the most effective way of testing, as we often had to perform a series of logical steps in order to test something out.

The program did have a handful of JUnit tests that existed prior to our group’s involvement, and at the time of this writing all of the tests still pass. They cover a variety of test cases, such as ensuring data types are what they should be. We added more unit tests throughout the project and they all passed.

Unfortunately, due to time constraints and higher-prioritized issues, testing was not further expanded upon beyond what was mentioned above.

# Logger

The logger we implemented is log4j version 2. Almost every interaction with the database, such as uploading users or uploading companies, has an appropriate set of log statements, which can be viewed either in the console within eclipse or by downloading the log file with the button available to admins.

# Deployment/Maintenance

At the time of this writing, the only way to run the program is by importing it into an Eclipse workspace. As far as deployment is concerned, this means that the project must be downloaded from the GitHub repository and imported into Eclipse. A connection to a MySQL server is also necessary, since all of the data is being stored in a local database instance. This makes deployment lack simplicity in a lot of ways, as practically all programs designed for consumers come packaged with all of the necessary tools to run as long as the computer meets the requirements. The Web-Based Evaluations program is heavily reliant on software that a majority of every-day users would likely not have on their machines.

With these ideas in mind, maintenance would be extremely cumbersome moving forward. Because the program is not in a truly deployable state, there is also no elegant way to update it. Any time changes are pushed to the repository, one would have to redownload and import the entire project to reflect adjustments made. Additionally, all program data is saved locally. This means that data is only consistent for each individual computer running the program. From a deployment mindset, this doesn’t make the program very useful for large organizations.

# Post-Mortem Analysis

Throughout working on this project there were many issues and that arose over time. Looking back at the project now with the knowledge we have gained from this class there would be various changes that we would have made differently. First, we would have immediately refactored the database to make the relationships between each class cleaner and simpler to help in the process of making methods. The logic that we used to perform these operations was definitely not the most efficient but if we refactored the database then we would be able to make direct connections between classes that would allow for quicker and more efficient operations to be able to be performed.

Since we were handed down this project it was not exactly conceivable to restructure the whole project, so it followed a software design pattern. The use of a design pattern would help with easing the number of changes that would be done when adding new classes or other parts of code. In the long run, the use of a design pattern would greatly impact the project in a good way with the possibility of adding classes to the application. In this case there would be very little code needed to be added to implement the new classes.

One of the largest areas of concern for the project is the security of the project that we were unable to get too due to lack of time. We did not have to time to penetration test the application to find all of its vulnerabilities. If we were able to find these vulnerabilities, then we could be able to secure these vulnerabilities so that the application is more secure. With time permitting we could have been able to threat model the project so we could help secure it against STRIDE attacks making the project more complete.