Web-Based Evaluations

Final Report

Class CPSC 488 Section 02

Group Number: 2

Revised: 4/23/2023

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

Revised By:

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)

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)

[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 myriad 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

* Fix the privileges to use collections of users and locations
* Fix Upload\_Roles
  + Instead of locations, Roles should be based on the Company Name, Location Group, Location, City, Province, Country
* Upload Company – World is not spelled correctly in the last column in the Excel document
* Missing Role for Adding a User (special case)
* Local company Admin bug
  + Supersuperuser could be edited and deleted by the ADMIN, a change was made to address this.
* Loading in Users for Thangiah Manufacturing LLC
  + Addressed issue regarding the upload of users into a department not specified.
* Is the evaluation form associated with a company?
  + Eval Forms were determined to be separate from companies as they are created for companies.
* What if the listed department is not in the company?
  + Eval Forms were checked to find and fix a bug relating to if a department mentioned in the eval form does not exist in the company
  + Only present departments can now be uploaded to evaluation forms
* Departments are automatically added when a user is added with a department that is not in the list.
  + Departments are now no longer automatically added.
* Program crashes if password is not 5 characters on login creation.
* Host code with Tomcat.
* No one showing for editing after evaluation forms and groups are loaded.
  + Fixed the edit group page to show the appropriate evaluators for each group.
  + Fix for evaluators not showing up in the Edit Groups screen.
* When editing newly added Admin, no company or location information is shown.
* When editing newly added admin, no company or location information is shown.
* Made it so SuperSuperUsers cannot use the Manage Roles screen anymore, and they are limited in the users they can upload to only Admins now
* Fixed the issue that was allowing duplicate roles to be uploaded in the manage roles screen.
* Fix page to add companies, roles, locations and departments similar to the add user page.
* Made it so that the locations and roles were displayed when looking at the companies as a supersuper user.
* Moved old documentation and program files into a new folder to make it easier to identify old vs. new
* Made the companies page a table
  + Making the companies page display information in a table format
* Added loading indicators when files are being uploaded
* When logged in as admin, screen cuts off the display of data unless full screen is selected.
  + Fixed screen sizing issue.
* When displaying data, such as users, we made the display compact with smaller edit/delete buttons.
* Removed "My Evaluation" screen for SuperSuperUser
* Uploaded Documentation to include steps for "Upload Roles"
* Created an option to manually create a new Group
* Uploaded the case diagrams with the user perspective.
* Added JavaDoc folder and contents.
* Configured edit companies button for SuperSuperUser and established the interface/subpages
* Configured delete companies button for SuperSuperUser
* Created a new table relating to the evaluation of users.
* Adjusted the Groups page to a new format.
* Displayed Roles as a table, similarly to other pages.
* Fixed the Eval Templates Uploading on non-Windows Operating Systems
* Created a new function to force the SuperUser to change their password on first login.
* Addressed Company SuperUser being able to delete their own login.
  + SuperUser can no longer delete their own account.
* Made HTML page names consistent.
* Made it so admins cannot be created until companies have been uploaded/created.
* Made it so users cannot be created until roles have been uploaded.
* Made an option to have database persistence.
* Created a function to export non-added users during user upload to an excel file upon failed upload.
* Made the HTML buttons not run into each other.
* Refactored project to remove ‘group 3’ folder.
* Added confirmation for deactivating items.
* Updated Manage Roles table to show more relevant information.
* Added “View” page that displays role information in greater detail.
* Prevented groups from being created before users are loaded
* Updated password reset to be more in-line with standard practices (enter old password, new password, repeat new password).
* Reinstated the chart generation and analysis for groups.
* Fixed the group view table buttons.
* Admin2 can see all other admins if you click on the company.
  + Addressed this by creating the correct filtering related to the last 4 buttons on the manage user page.
* Add confirmation for deactivating items.
* When loading in the company, the data for the company is shown in the company page as locations.
* Implemented log4j
* Configure delete companies button for Supersuperusers.
* Fix displaying Supersuperuser under users screen.
* Configure edit companies button for Supersuperuser.
* Ensure all imported fields are populated in the edit user menu.
* Manage users fix table formatting .
* Created manage roles page edit option.
* Implemented new edit groups screen.
* Formated edit/deactivate buttons.
* Edit companies for Supersuperuser – extended editing.
* Created deactivation function for roles page.
* Manage Users table not showing Admins for SuperSuperUser.
  + The manage users table for the SuperSuperUser now shows the admins correctly.
* Evaluation Form Edits
  + Created button for "Create Evaluation Form."
  + Added an option to deactivate eval form.
* Made it so you can view an individual employee in a group.
* Made it so you can add an employee to a group manually.
* Made it so departments can be added to a company.
* Changed child company listing to button for editing child company.

## Tasks that Were Not Completed

* Make it so groups can be exported to an excel file
* Display all active groups with options
* Group archive adjustments
* Evaluation form controls are running into each other
* The departments should be updated when the company is changed when creating a user
* Create manual add for companies page
* Display reviewees current level of report progress
* Company logo for all pages
* Background image for starting screen
* Parent/child company system
* Add User: split division/branch into separate location and division/branch fields
* Enable error messages for adding departments
* Remove repeated company name listings.
* SupersuperUser: Once the companies are loaded, there should be an edit/view delete for each of the companies.
  + Clicking on Edit/View/Delete will take you to a page for working with the companies
  + The department should be displayed in the page that has all the details of the company in a manner that does not go into multiple pages.
  + If there are a large number of departments in a company, should not have to use 3 or more pages to display it.
* Weights for drop downs
* Privileges determination
* SuperSuperUser
  + When the edit of unit is clicked it should display a screen that allows departments to be added in addition to it being deactivated
  + What happens if duplicate companies/departments are loaded?
* Admin
  + Evaluations need to be separated by years (includes users, forms, and any type of evaluation done)
  + What happens when any duplicate data is loaded?
* Option to archive groups, users
* Restrict questions in workflow (eval forms)
* SuperSuperUser
  + Deactivate button is clipped
  + ‘Child’ is still used in deactivate table instead of ‘Unit’
  + After adding button, when Back is clicked, it does not display the department until refresh screen is clicked
  + Buttons need to be spaced out
  + Explain how privileges work
* Need a back button for the View Role screen
* No Evaluation Groups
* No move users between groups
* Explain how to log in as a USER

# 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

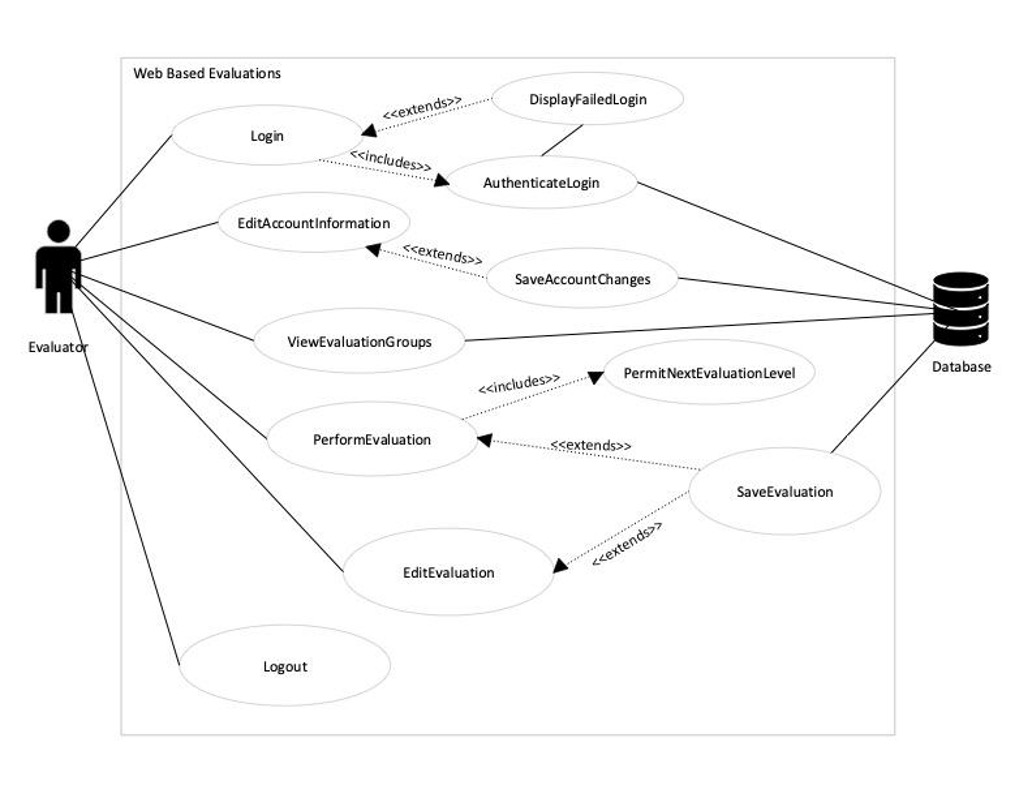
The issue at hand is the need for a system that provides the ability to view and perform evaluations on staff or employees at a company or organization. The requirements include the ability to customize the evaluation criteria and include qualitative and quantitative questions. Evaluations must be appropriately 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

# 

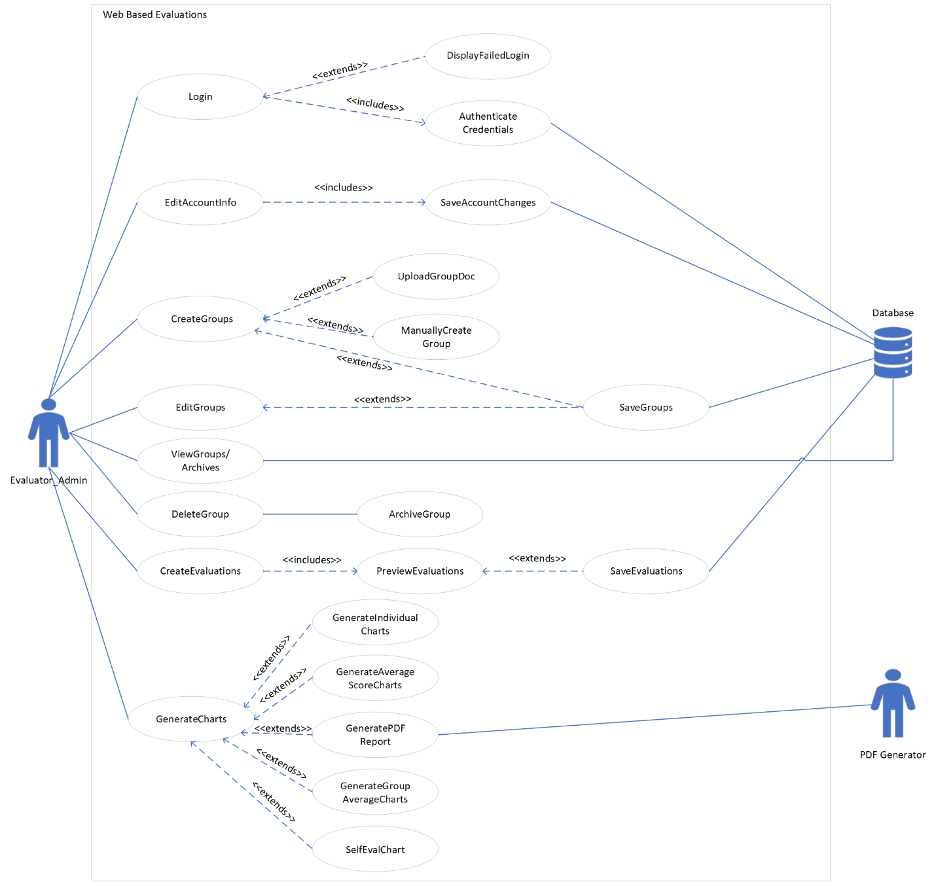
## 

## 

## Use Case of the SuperSuperUser Role

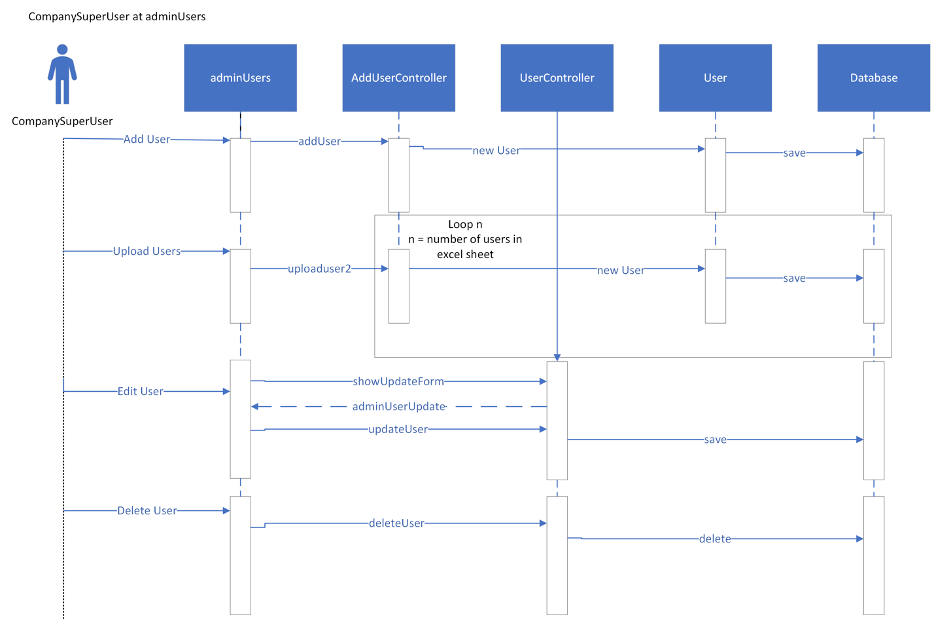


## Use Case of the Evaluator Admin

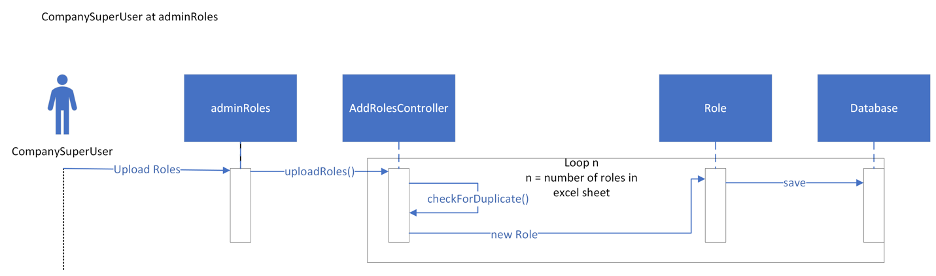


# Sequence Diagrams

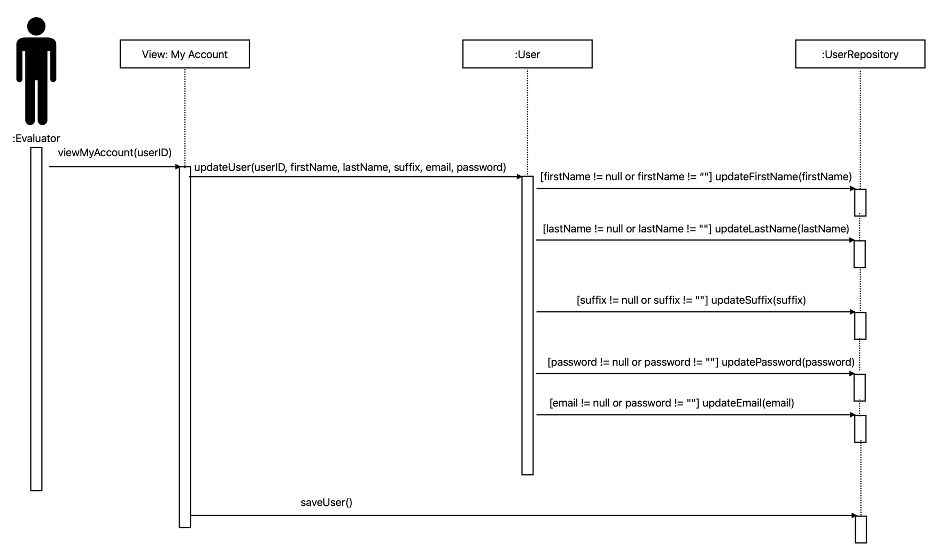
## CompanySuperUser on the Manage Users Page



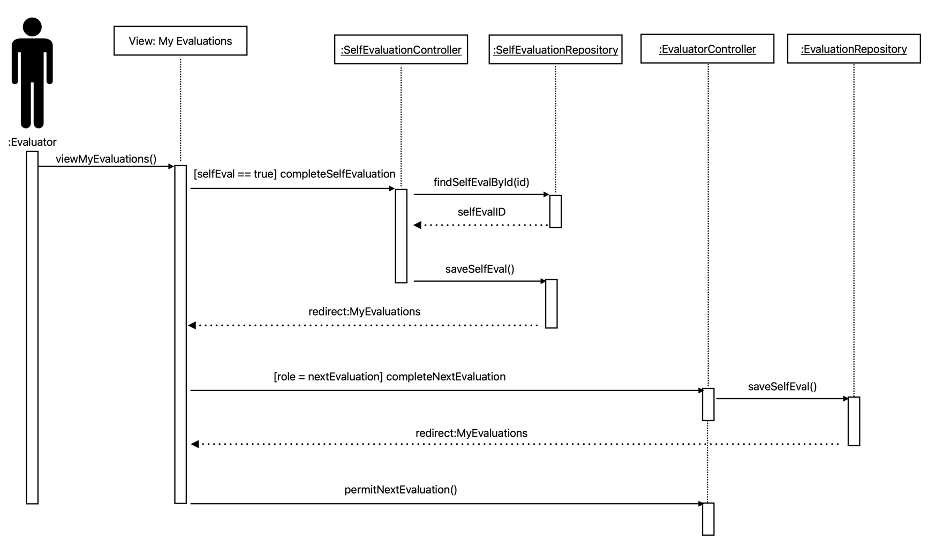
## CompanySuperUser on the Manage Roles Page



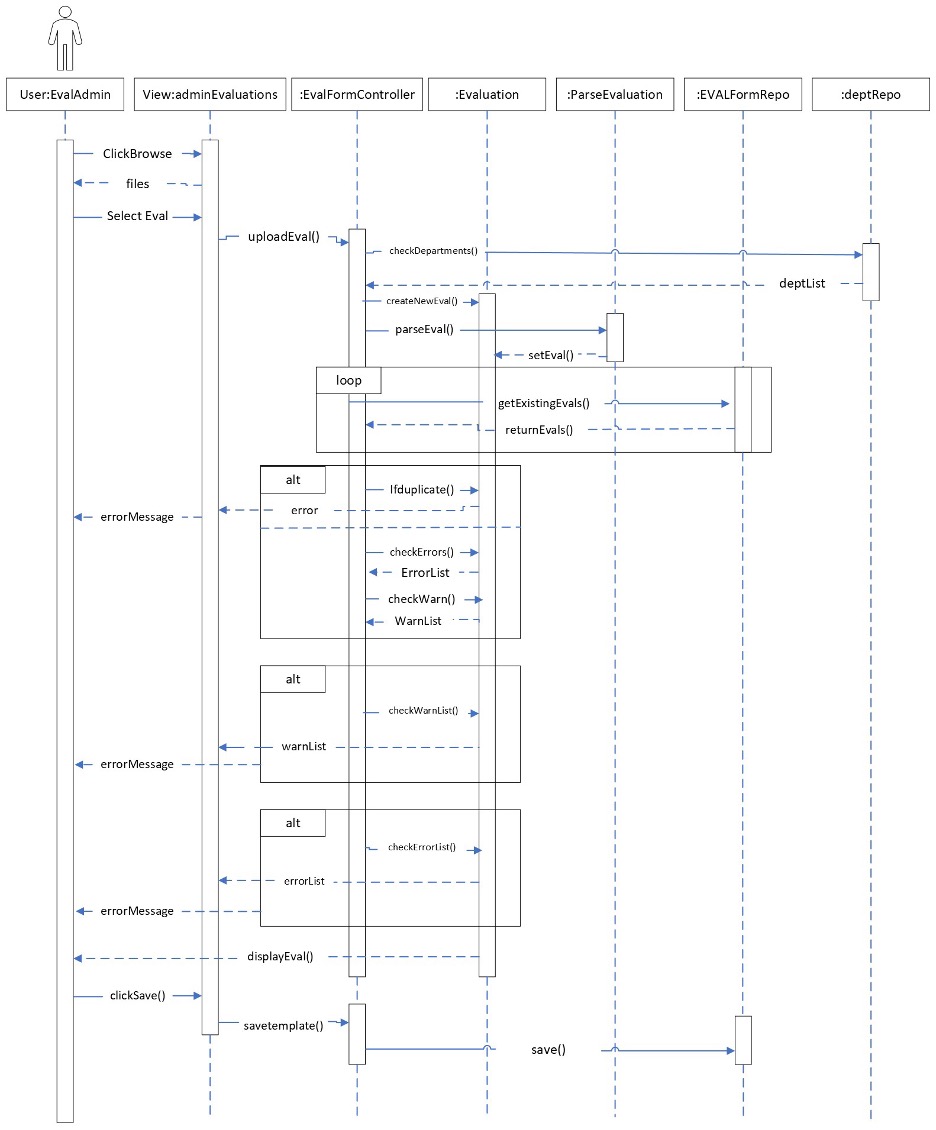
## Evaluator Edit Account Information



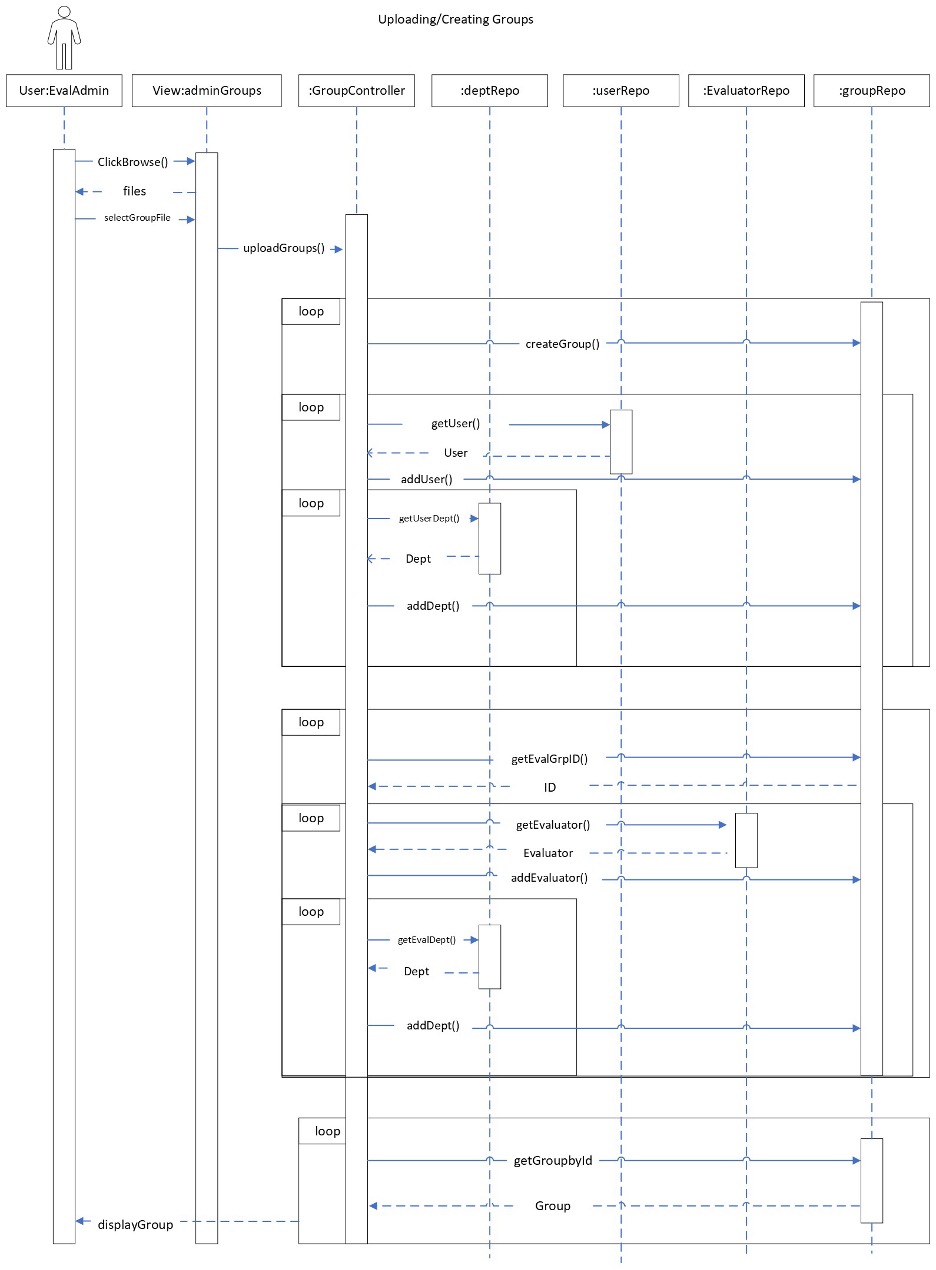
## Evaluator Perform Evaluation



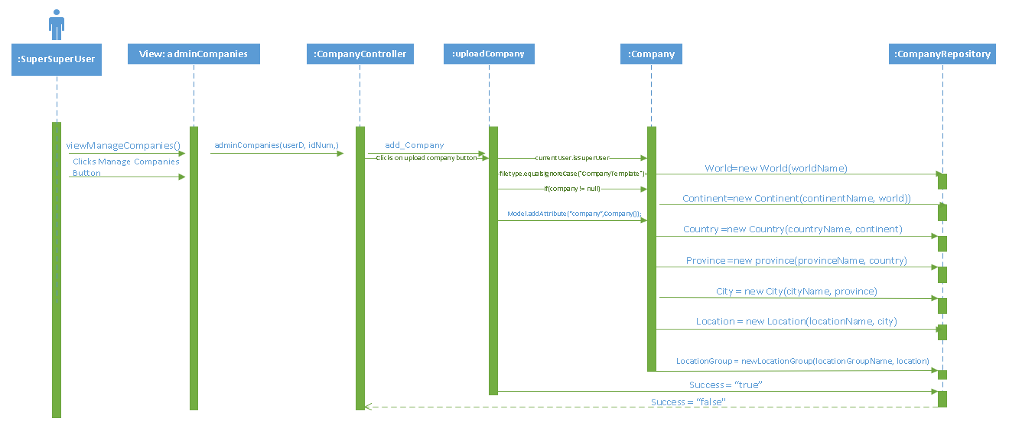
## Uploading Evaluation Forms



## Uploading Groups



## Uploading Companies



# 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. The Groups page is also partly incomplete and has missing functionality, such as archiving groups and moving/editing/removing group employees.

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
    - 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.

# 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.

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

Every project made by anyone is likely to be subject to either convoluted solutions or patch-up jobs in order to function and this project is no exception. Below are a few examples of which less-elegant solutions were provided.

In the AdminMethodsService class, there are two very similar methods that are depended on heavily from particular methods from within the UserController and AddUserController. The two methods provide a series of checks on particular attributes of a user in order to save either a new user or to changes made to an existing user. The summation of the lines of code used by both methods makes up approximately two-thirds of the class’s total code count. While combining both methods into a singular could prove to be a little difficult, removing the redundancy by creating a third method seems easier at first glance.

Another particularly bizarre decision made in the AdminMethodsService class comes from the sortCheck method present. The method is quite redundant as it checks what type of sorting is being asked of it and then forwards the information to an instance of UserService. The issue is apparent upon staring at the sorting method inside the UserService; the sorting method also checks what type of sorting is being requested. The change would be relatively simple but what is done, is done.

The groupUpload method definitely has room for improvement . As we worked on the project we ended up adding more fields in the group upload file and because of that we ended up just adding changes on top of the upload method resulting in the upload method having high time complexity. if we had a better idea of what the group file was supposed to look like in the beginning the group upload method would have been implemented differently.

Looking back on the chart generation and PDF reports there are a few things we would do differently. First, we would have used the setChartTheme method in the ChartFactory class more often. The charts generated are using the default theme, using the setChartTheme method the theme of the charts can be changed to be more visually appealing. The second thing we would do differently pertains to the PDF report generation. We decided to use iText which is a free java PDF library which allows the user to create, convert, and manipulate PDF documents. If I were to go back and redo the project I would opt to use a different library. I would most likely choose to use Jasper Reports. This is a java reporting tool which generates reports in PDF documents. Jasper Reports allows for more versatility and functionality when creating reports.

While the core of the Evaluation object and its subclasses were well designed, there is certainly room for improvement. For example, all section types are represented by the same Section class and all question types are represented by the same Question class. This means that all methods and variables are present in all classes of the same type, even if they are not necessary. For example, the methods pertaining to score computation are only necessary in sections which contain compute questions or dropdown questions. In a perfect implementation, these methods would only be available to the sections and questions that can make use of them. Using principles of inheritance classes, abstract classes, and design patterns, the Evaluation object could be designed more elegantly and logically in terms of proper Object-Oriented programming.

Much of the code reusability could have been utilized to a greater degree. For example, the Roles controller has many methods in it that need to acquire similar information. One instance of this is when trying to match a role based on the ID that was sent to the controller. In several of the methods that need to do this, the entire roles repository is searched based on the company of the user and matches the ID based on what it finds through there. This could have been scaled down to one line of code, as there is a much simpler way to do this. The initial approach was taken because the roles repository stores optional values, and so trying to find a value in the repository where it could potentially be null creates Java errors. There is a simple “.get()” property that can be applied to optional values if you are certain that a value exists. This could have been utilized instead of searching through the entire company roles repository every single time.

Similarly, there are many array lists created in the program several times that reference the same data. At many points in the program, lists are created randomly when information is needed from the database. In some cases, a similar list already exists in the scope and just needs to be filtered out a little. Code reusability exists in the program, however it was not always utilized properly.

There is also not a ton of consistency among the various UI components. Some screens with identical information show different keywords, such as “Add” instead of “Upload.” Other screens with activation/deactivation tabs are not used in other screens where it would make sense, such as Users and Companies. Some colors of buttons that perform the same actions vary, when it would be best practice to keep those uniform.

Better coding practices are also worth mentioning. It would be dishonest to say that our group consistently upheld proper Java coding practices and standards. While they certainly exist in many areas of the program, in other areas they do not. Some of this is due to our own fault as well as the lack of proper procedures from other groups as well. Looking back on the project, it would have been best for us to establish what the guidelines are, and implement it into each of the various classes, methods, variable names, and more. Comments were sporadic throughout the code, and sometimes none existed where it would have benefited others to understand what was trying to be accomplished. The timeframe of our project unfortunately did not give us the privilege of making these ideas consistent throughout the entire program.