**Online Customer Care and Service Centre**

|  |  |  |  |
| --- | --- | --- | --- |
| **Project Id :** | **A073** | *<SC.ID>* |  |
| **Release Id :** |  |  |  |

**Application Design**



|  |  |  |  |
| --- | --- | --- | --- |
|  | **Prepared By** | **Reviewed by** | **Approved By** |
| **Name** | Linjo Rejoy |  |  |
| **Role** | Programmer Analyst Trainee |  |  |
| **Signature** | Linjo |  |  |
| **Date** | 18 Dec 2020 |  |  |

**Contents**

[1.0 Introduction 4](#_Toc59177768)

[1.1 Purpose of the Document 4](#_Toc59177769)

[1.2 Scope 4](#_Toc59177770)

[1.2.1 In Scope 4](#_Toc59177771)

[1.2.2 Out Scope 4](#_Toc59177772)

[1.3 Intended Audience 4](#_Toc59177773)

[2.0 Conventions and Standards 5](#_Toc59177774)

[3.0 Assumptions, Dependencies and Risks 5](#_Toc59177775)

[4.0 System Architecture 6](#_Toc59177776)

[5.0 Use Case Realization 7](#_Toc59177777)

[5.1 Flow of Events 7](#_Toc59177778)

[5.1.1 Login Sequence 7](#_Toc59177779)

[5.1.2 Admin Sequence 7](#_Toc59177780)

[5.1.3 Analyst Sequence 8](#_Toc59177781)

[5.1.4 Email Analyst Sequence 9](#_Toc59177782)

[5.1.5 Admin Controller Feedback 9](#_Toc59177783)

[5.1.6 Admin Add Feedback 10](#_Toc59177784)

[5.1.7 Admin Email to Analyst 10](#_Toc59177785)

[5.2 System Object Model 11](#_Toc59177786)

[5.2.1 Model Classes 11](#_Toc59177787)

[5.2.2 Repositories 13](#_Toc59177788)

[5.2.3 Validator Classes 13](#_Toc59177789)

[5.2.4 Service Classes 14](#_Toc59177790)

[5.2.5 Controller Classes 14](#_Toc59177791)

[5.2.6 Security 15](#_Toc59177792)

[5.2.7 Exporter Classes 15](#_Toc59177793)

[5.2.8 Application Classes 15](#_Toc59177794)

[5.3 UI Controls 16](#_Toc59177795)

[5.3.1 User Registration 16](#_Toc59177796)

[5.3.2 Analyst Registration 17](#_Toc59177797)

[5.3.3 Login Admin 18](#_Toc59177798)

[5.3.4 Login User 18](#_Toc59177799)

[5.3.5 Login Analyst 18](#_Toc59177800)

[5.3.6 Forgot User ID-User 19](#_Toc59177801)

[5.3.7 Forgot Password-User 19](#_Toc59177802)

[5.3.8 Reset Password-User 20](#_Toc59177803)

[5.3.9 Forgot Analyst ID-Analyst 20](#_Toc59177804)

[5.3.10 Forgot Password-Analyst 21](#_Toc59177805)

[5.3.11 Reset Password-Analyst 21](#_Toc59177806)

[5.3.12 Create Complaint-User 22](#_Toc59177807)

[5.3.13 Complaint View-User 22](#_Toc59177808)

[5.3.14 Complaint View-Analyst 23](#_Toc59177809)

[5.3.15 Complaint View-Admin 24](#_Toc59177810)

[5.3.16 Add Feedback-Admin 24](#_Toc59177811)

[5.3.17 Email View-Analyst 25](#_Toc59177812)

[5.3.18 Email-Admin 25](#_Toc59177813)

[5.4 UI Design 26](#_Toc59177814)

[5.4.1 Admin Login 26](#_Toc59177815)

[5.4.2 Register User/Analyst 26](#_Toc59177816)

[5.4.3 Register Analyst 27](#_Toc59177817)

[5.4.4 Create Complaint User 27](#_Toc59177818)

[5.4.5 View Complaint User 28](#_Toc59177819)

[5.4.6 All Complaints 28](#_Toc59177820)

[5.4.7 User to Analyst Mail 29](#_Toc59177821)

[5.4.8 5.4.6 Analyst/Admin Home 29](#_Toc59177822)

[5.4.9 Admin Home 30](#_Toc59177823)

[5.4.10 View All Complaints Analyst/Admin 30](#_Toc59177824)

[5.4.11 View Complaint Analyst/Admin 31](#_Toc59177825)

[5.4.12 Create Feedback Admin 32](#_Toc59177826)

[5.4.13 Analyst Emails 32](#_Toc59177827)

[5.4.14 Analyst Email View 33](#_Toc59177828)

[5.5 Component Description 34](#_Toc59177829)

[5.5.1 Controller Classes 34](#_Toc59177830)

[5.5.2 Service Classes 48](#_Toc59177831)

[6.0 Database Design 61](#_Toc59177832)

[6.1 Table Structure 61](#_Toc59177833)

[6.1.1 User Table 61](#_Toc59177834)

[6.2 E- R Diagram (Relationship between Tables) 67](#_Toc59177835)

[6.3 Standards and Guidelines Followed 68](#_Toc59177836)

[6.3.1 JAVA 68](#_Toc59177837)

[7.0 Change Log 68](#_Toc59177838)

# Introduction

## Purpose of the Document

The purpose of the detail design document is to define the detailed design for all the components of Online Customer Care and Service Centre that are specified in the Business Requirement document. All the low-level components, interfaces and integrations are detailed out in this document.

## Scope

### In Scope

* User Register/Login/Create Complaint/View Complaint
* Analyst Register/Login/View Complaint
* Admin Login/View Complaint/Download
* Create/View RL
* Create/View Status Report

### Out Scope

* Migration of current employee list into the application

## Intended Audience

The intended audience for this document is

* Developers, testers, other associates in the project
* Client
* Delivery Assurance Group, Process Engineering group

# Conventions and Standards

*Online Customer Care and Services system will follow Oracle’s java coding standard*

# Assumptions, Dependencies and Risks

**Assumptions**

* The Customer Care Application will employ 3 types of users. a normal User, an Analyst, and an Admin.
* Once the User after registration login to the webapp, He/she can submit a complaint.
* Once submitted the admin can forward the complaint and assign it to any of the existing Analysts’.
* The Analyst can view the issue and do the necessary action and mark it as resolved.
* If the issue requires more attention, the analyst can forward it to the next level analyst.

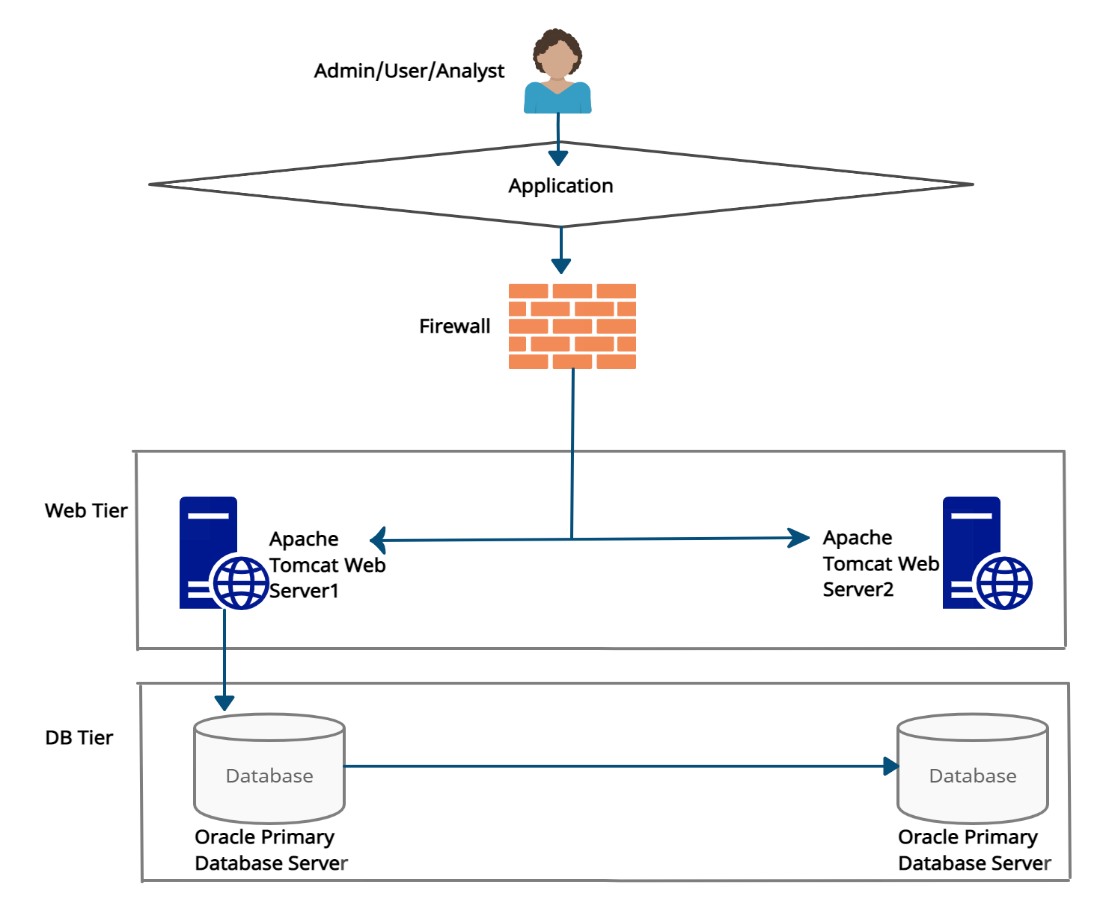
**Dependencies**

* MySQL 8.0 Relational Database
* JDK 1.8 or above
* Spring Boot starter web
* Spring boot starter security
* Tomcat Server
* Bootstrap
* JQuery
* Junit and Mockito Testing
* Spring Core

**Risks**

1. The current system design application can perform optimally with less than 1000 employees.

# System Architecture

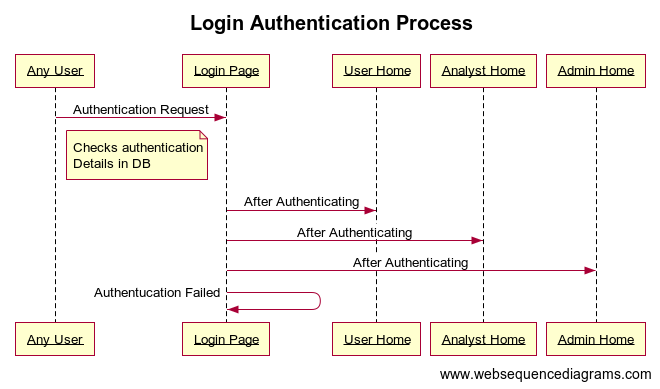


We have used Spring security to perform authorization and authentication.

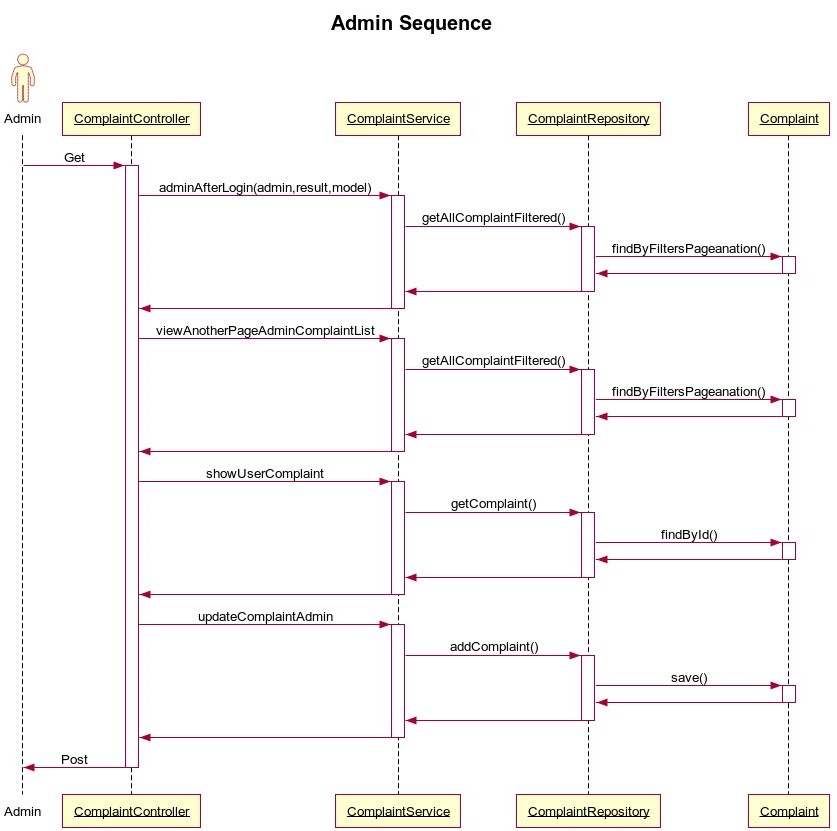
# Use Case Realization

## Flow of Events

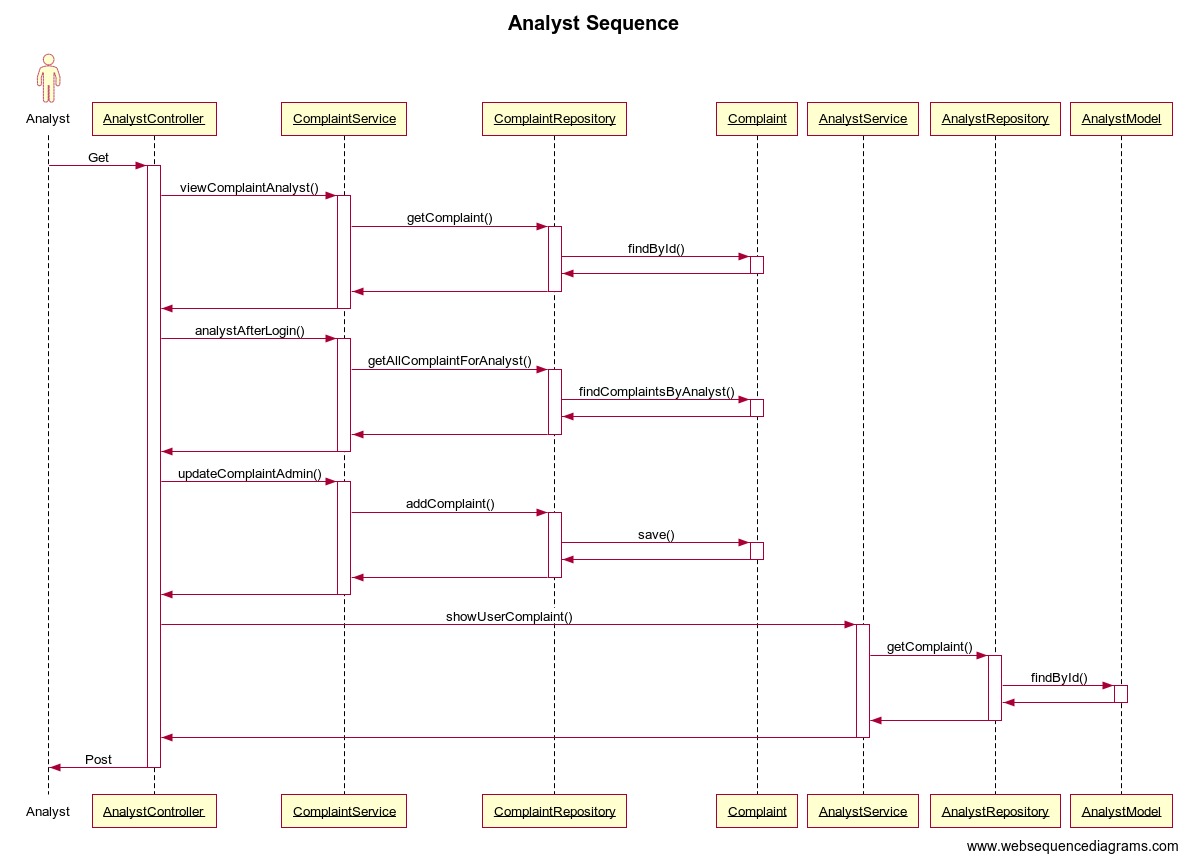
### Login Sequence

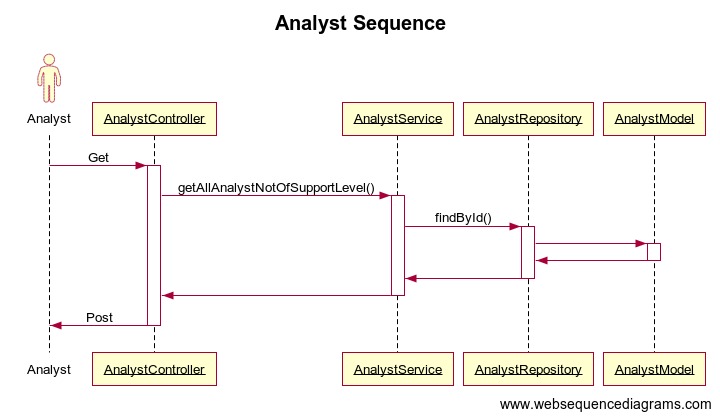


### Admin Sequence

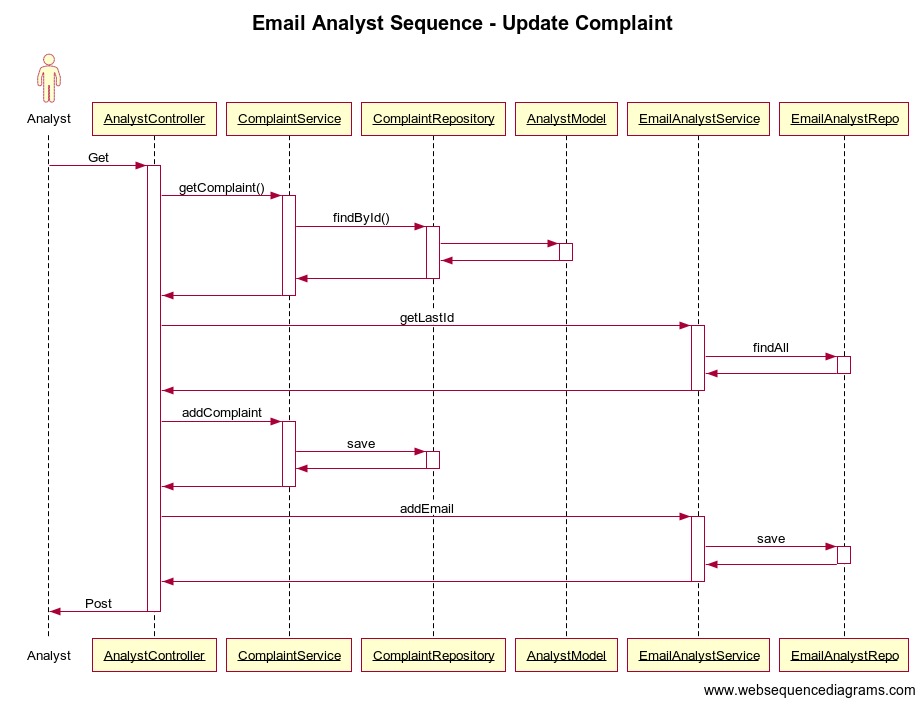


### Analyst Sequence

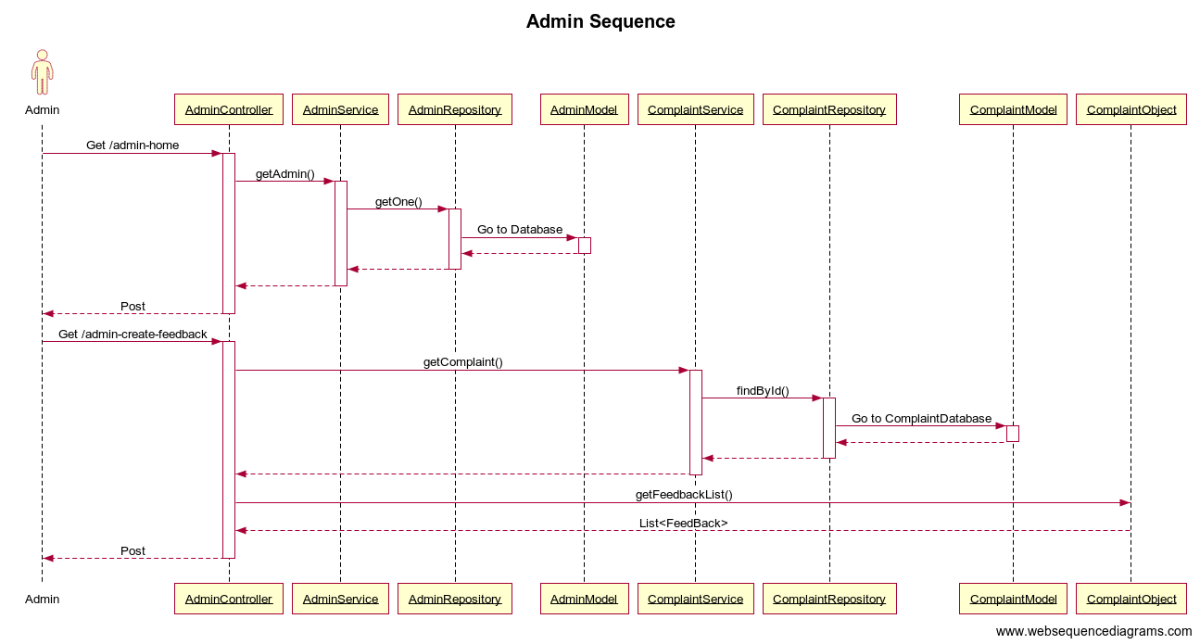




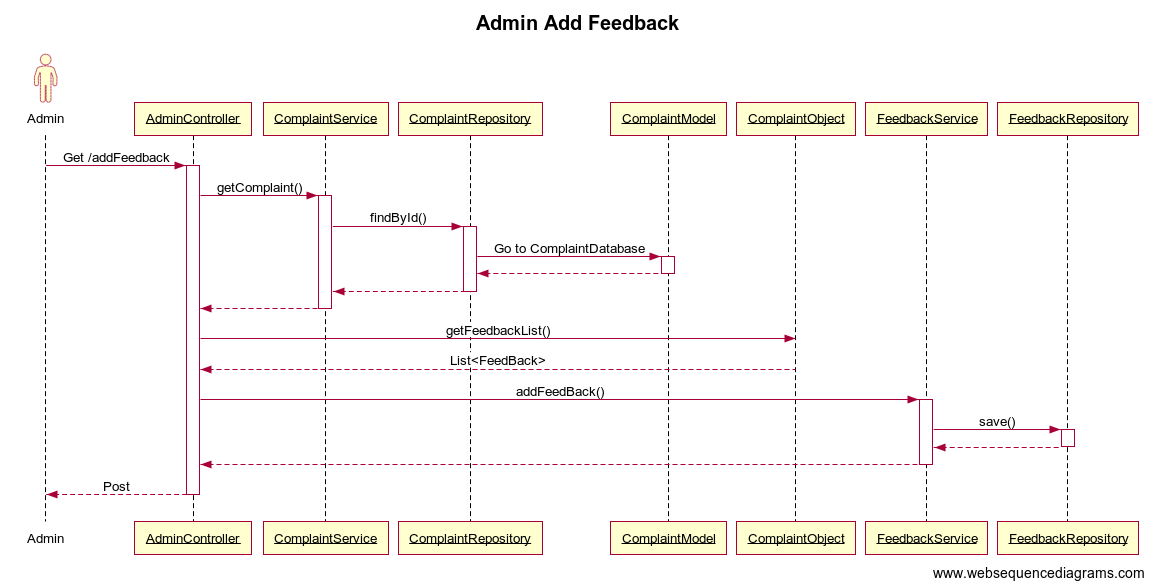
### Email Analyst Sequence



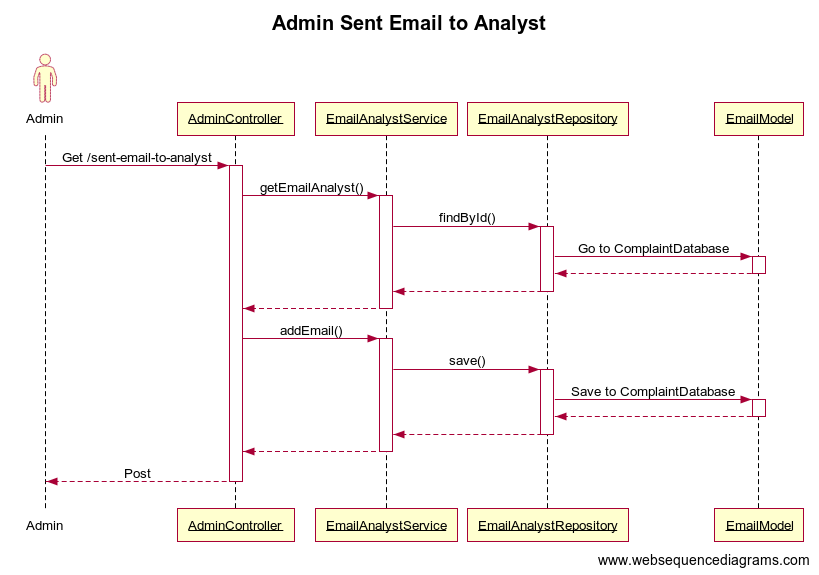
### Admin Controller Feedback



### Admin Add Feedback



### Admin Email to Analyst



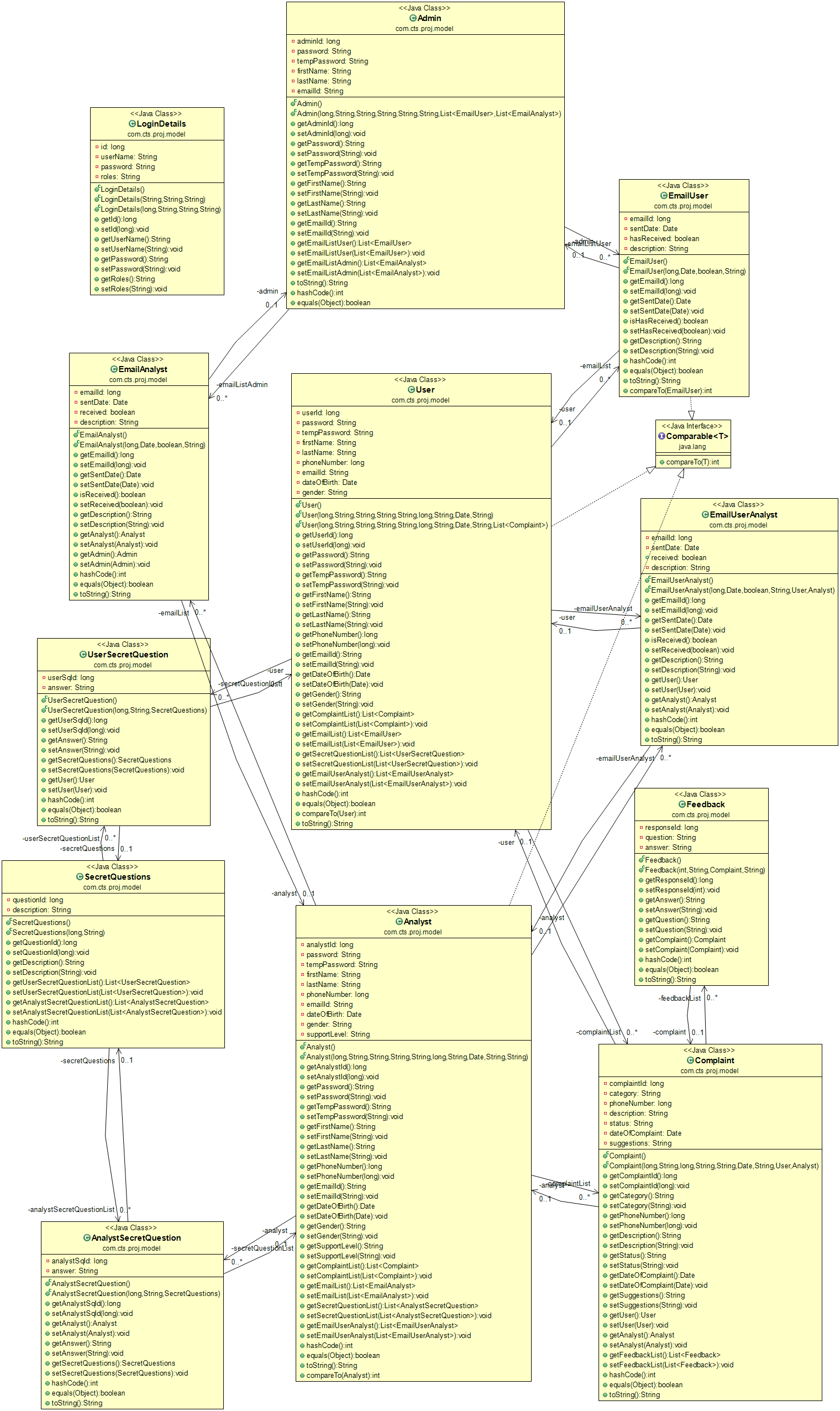
## System Object Model

### Model Classes

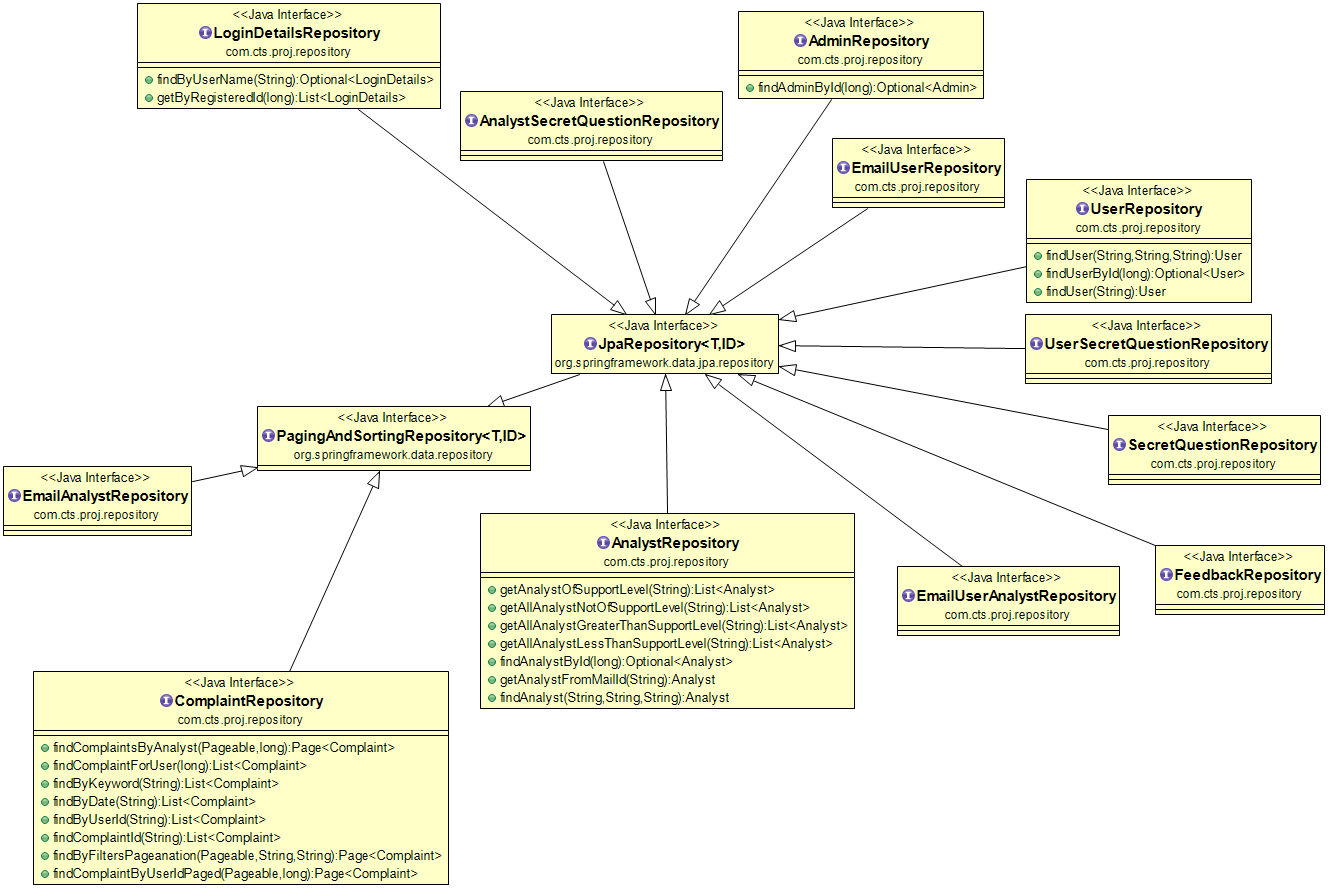
Model Classes required for storing the information as objects for easy writing and reading.

Overall 12 model classes are present.

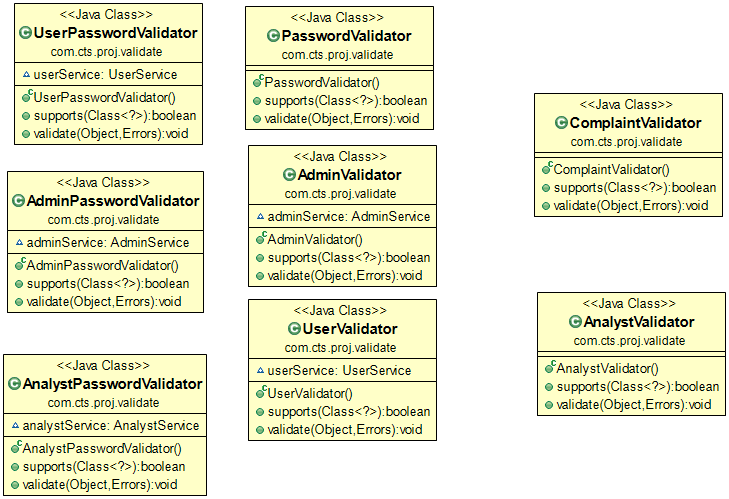
1. **User**: to store the information about the user
2. **Analyst**: to store info about the analyst
3. **Admin**: to store the information about the admin
4. **Complaint**: to store the info about the Complaint
5. **Feedback**: to store the feedback provided by user
6. **EmailAnalyst**: To store email object from 1 analyst or admin to another analyst
7. **EmailUser**: To store email object from admin to user
8. **EmailUserAnalyst**: To store email object from user to analyst
9. **UserSeretQuestions**: To Store secret question submitted by user
10. **AnalystSecretQuestion**: To Store secret question submitted by Analyst
11. **SecretQuestions**: To Store secret questions description
12. **LoginDetails**: This database holds all the login details info which is to be extracted by Spring Security for verification.



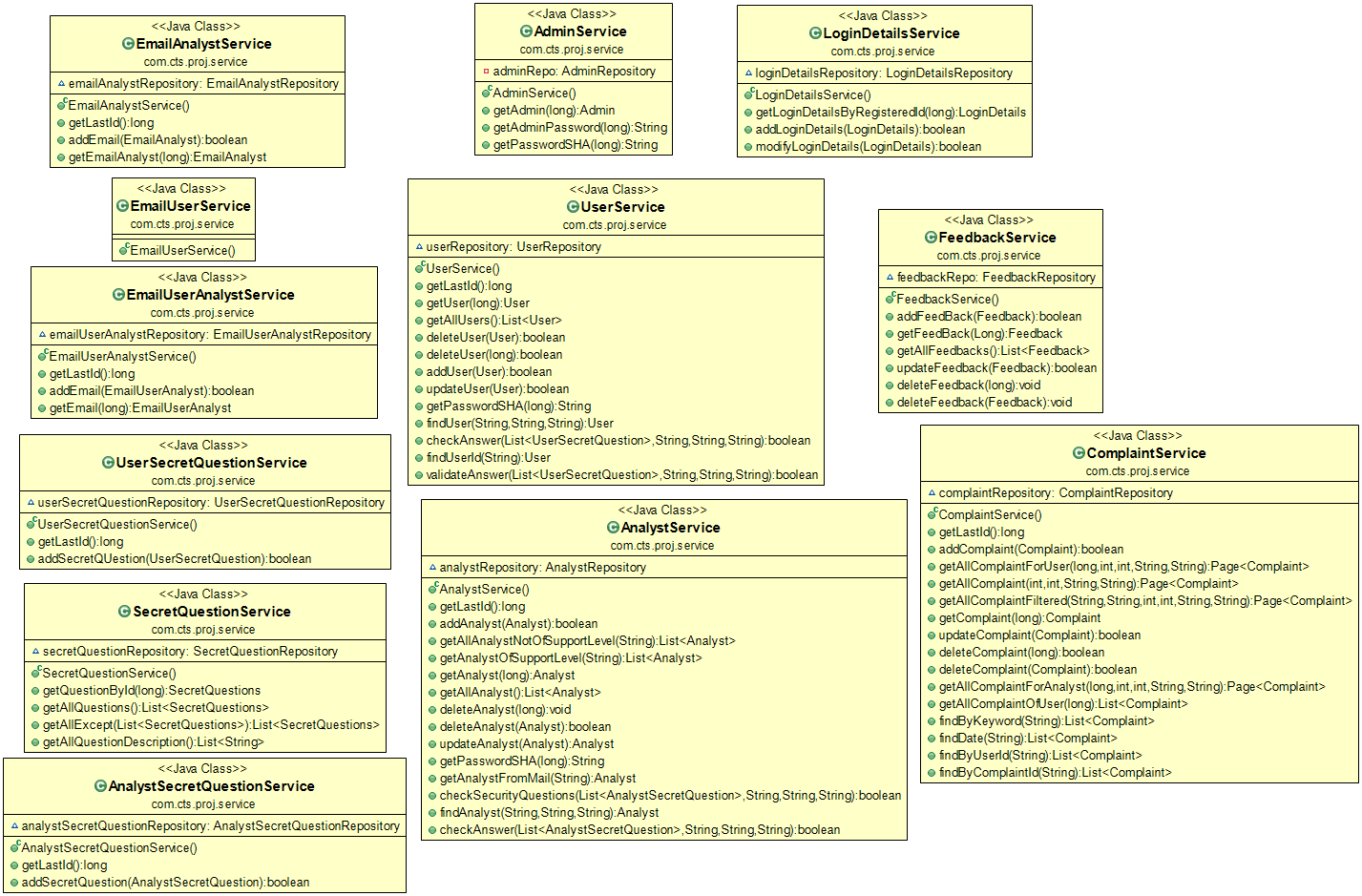
### Repositories



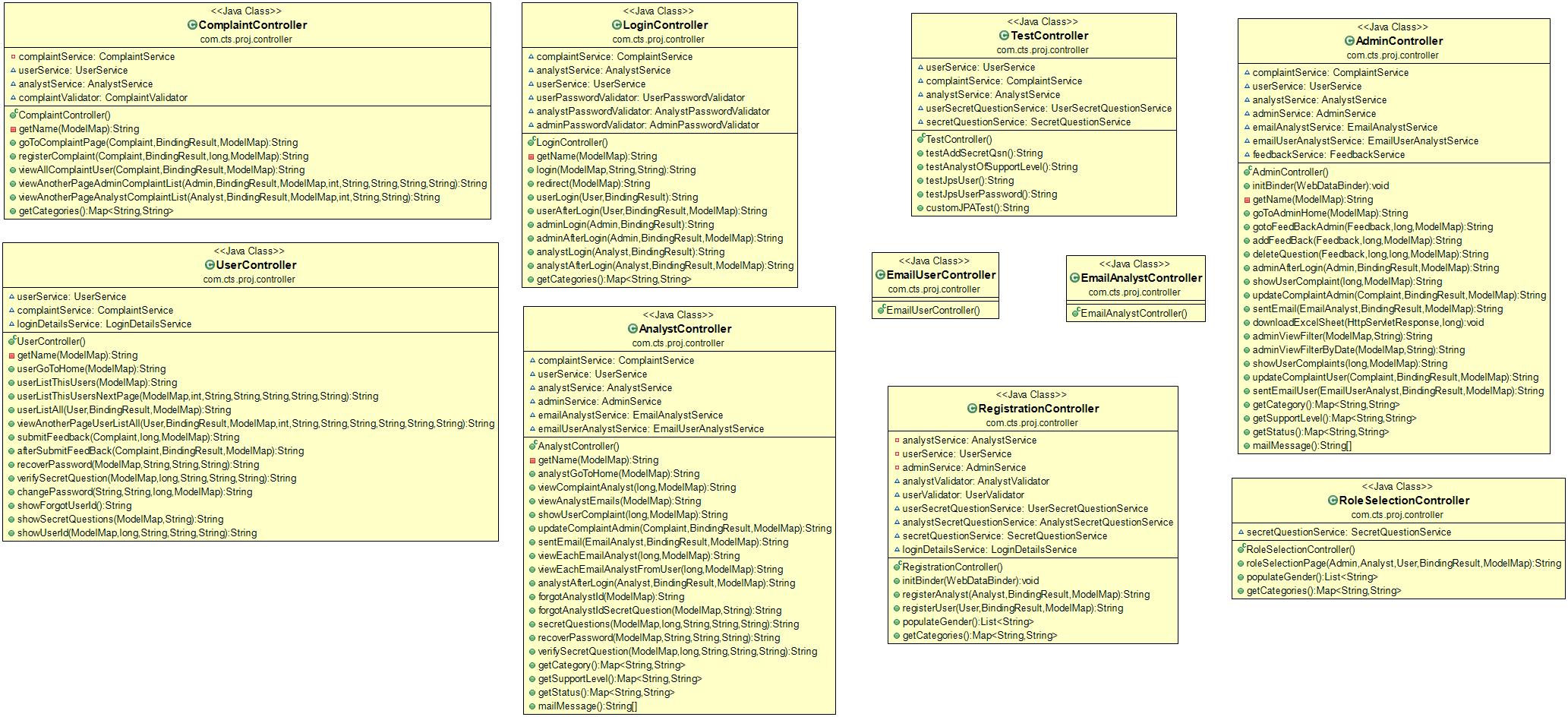
### Validator Classes



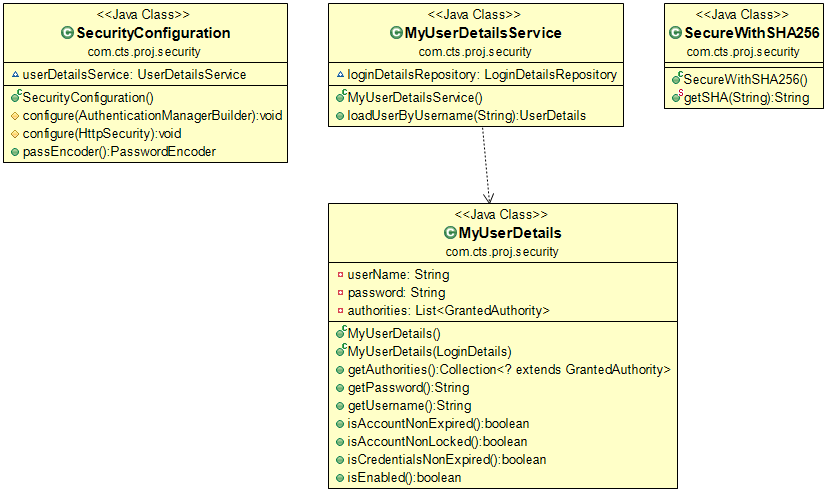
### Service Classes



### Controller Classes



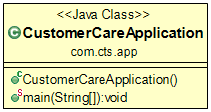
### Security



### Exporter Classes



### Application Classes



## UI Controls

### User Registration

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Data Element** | **Control type** | **Default**  **Valu es** | **Editable**  **Field** | **Restriction s** | **Mandatory/**  **Non**  **Mandatory** | **Data Type** | **Data**  **Size** |
| First Name | Textbox | No | Yes | Only Alphabets | Mandatory | Varchar | 45 |
| Last Name | Textbox | No | Yes | Only Alphabets | Mandatory | Varchar | 45 |
| Phone Number | Textbox | No | Yes | Should be 10 digits long | Mandatory | Varchar | 15 |
| Email | Textbox | No | Yes | Should be a Proper email ID Format | Mandatory | Varchar | 45 |
| DOB | Textbox | No | Yes | Valid Date | Mandatory | Date | - |
| Gender | Drop Down |  | Yes | - | Mandatory | Varchar | 10 |
| Password | Textbox | No | Yes |  | Mandatory | Varchar | 256 |
| Confirm Password | Textbox | No |  |  | Mandatory | Varchar | 256 |
| Security Question 1 | Textbox | No | Yes | - | Mandatory | Varchar | 45 |
| Security Question 2 | Textbox | No | Yes | - | Mandatory | Varchar | 45 |
| Security Question 3 | Textbox | No | Yes | - | Mandatory | Varchar | 45 |

### Analyst Registration

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Data Element** | **Control type** | **Default**  **Valu es** | **Editable**  **Field** | **Restriction s** | **Mandatory/**  **Non**  **Mandatory** | **Data Type** | **Data**  **Size** |
| First Name | Textbox | No | Yes | Only Alphabets | Mandatory | Varchar | 45 |
| Last Name | Textbox | No | Yes | Only Alphabets | Mandatory | Varchar | 45 |
| Phone Number | Textbox | No |  | Should be 10 digits long | Mandatory | Varchar | 15 |
| Email | Textbox | No |  | Should be a Proper email ID Format | Mandatory | Varchar | 45 |
| DOB | Textbox | No | Yes | Valid Date | Mandatory | Date | - |
| Gender | Drop Down |  | Yes | - | Mandatory | Varchar | 10 |
| Support Level | Drop Down | No | Yes | - | Mandatory | Varchar | 2 |
| Password | Textbox | No | Yes |  | Mandatory | Varchar | 256 |
| Confirm Password | Textbox | No | Yes |  | Mandatory | Varchar | 256 |
| Security Question 1 | Textbox | No | Yes | - | Mandatory | Varchar | 45 |
| Security Question 2 | Textbox | No | Yes | - | Mandatory | Varchar | 45 |
| Security Question 3 | Textbox | No | Yes | - | Mandatory | Varchar | 45 |

### Login Admin

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Data Element** | **Control type** | **Default**  **Valu es** | **Editable**  **Field** | **Restriction s** | **Mandatory/**  **Non**  **Mandatory** | **Data Type** | **Data**  **Size** |
| Admin ID | Textbox | No | Yes |  | Mandatory | INT |  |
| Password | Textbox | No | Yes |  | Mandatory | Varchar | 256 |

### Login User

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Data Element** | **Control type** | **Default**  **Valu es** | **Editable**  **Field** | **Restriction s** | **Mandatory/**  **Non**  **Mandatory** | **Data Type** | **Data**  **Size** |
| User ID | Textbox | No | Yes |  | Mandatory | INT |  |
| Password | Textbox | No | Yes |  | Mandatory | Varchar | 256 |

### Login Analyst

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Data Element** | **Control type** | **Default**  **Valu es** | **Editable**  **Field** | **Restriction s** | **Mandatory/**  **Non**  **Mandatory** | **Data Type** | **Data**  **Size** |
| Analyst ID | Textbox | No | Yes |  | Mandatory | INT |  |
| Password | Textbox | No | Yes |  | Mandatory | Varchar | 256 |

### Forgot User ID-User

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Data Element** | **Control type** | **Default**  **Valu es** | **Editable**  **Field** | **Restriction s** | **Mandatory/**  **Non**  **Mandatory** | **Data Type** | **Data**  **Size** |
| Email | Textbox | No | Yes | Should be a Proper email ID Format | Mandatory | Varchar | 45 |
| Security Question 1 | Textbox | No | Yes | - | Mandatory | Varchar | 45 |
| Security Question 2 | Textbox | No | Yes | - | Mandatory | Varchar | 45 |
| Security Question 3 | Textbox | No | Yes | - | Mandatory | Varchar | 45 |

### Forgot Password-User

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Data Element** | **Control type** | **Default**  **Valu es** | **Editab le**  **Field** | **Restriction s** | **Mandatory/**  **Non**  **Mandatory** | **Data Type** | **Data**  **Size** |
| User ID | Textbox | No | Yes |  | Mandatory | INT |  |
| Phone Number | Textbox | No | Yes | Should be 10 digits long | Mandatory | Varchar | 15 |
| Email | Textbox | No | Yes | Should be a Proper email ID Format | Mandatory | Varchar | 45 |
| Security Question 1 | Textbox | No | Yes | - | Mandatory | Varchar | 45 |
| Security Question 2 | Textbox | No | Yes | - | Mandatory | Varchar | 45 |
| Security Question 3 | Textbox | No | Yes | - | Mandatory | Varchar | 45 |

### Reset Password-User

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Data Element** | **Control type** | **Default**  **Valu es** | **Editable**  **Field** | **Restriction s** | **Mandatory/**  **Non**  **Mandatory** | **Data Type** | **Data**  **Size** |
| New Password | Textbox | No | Yes |  | Mandatory | Varchar | 256 |
| Confirm Password | Textbox | No | Yes |  | Mandatory | Varchar | 256 |

### Forgot Analyst ID-Analyst

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Data Element** | **Control type** | **Default**  **Valu es** | **Editable**  **Field** | **Restriction s** | **Mandatory/**  **Non**  **Mandatory** | **Data Type** | **Data**  **Size** |
| Email | Textbox | No | Yes | Should be a Proper email ID Format | Mandatory | Varchar | 45 |
| Security Question 1 | Textbox | No | Yes | - | Mandatory | Varchar | 45 |
| Security Question 2 | Textbox | No | Yes | - | Mandatory | Varchar | 45 |
| Security Question 3 | Textbox | No | Yes | - | Mandatory | Varchar | 45 |

### Forgot Password-Analyst

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Data Element** | **Control type** | **Default**  **Valu es** | **Editable**  **Field** | **Restriction s** | **Mandatory/**  **Non**  **Mandatory** | **Data Type** | **Data**  **Size** |
| Analyst ID | Textbox | No | Yes |  | Mandatory | INT |  |
| Phone Number | Textbox | No | Yes | Should be 10 digits long | Mandatory | Varchar | 15 |
| Email | Textbox | No | Yes | Should be a Proper email ID Format | Mandatory | Varchar | 45 |
| Security Question 1 | Textbox | No | Yes | - | Mandatory | Varchar | 45 |
| Security Question 2 | Textbox | No | Yes | - | Mandatory | Varchar | 45 |
| Security Question 3 | Textbox | No | Yes | - | Mandatory | Varchar | 45 |

### Reset Password-Analyst

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Data Element** | **Control type** | **Default**  **Valu es** | **Editable**  **Field** | **Restriction s** | **Mandatory/**  **Non**  **Mandatory** | **Data Type** | **Data**  **Size** |
| New Password | Textbox | No | Yes |  | Mandatory | Varchar | 256 |
| Confirm Password | Textbox | No | Yes |  | Mandatory | Varchar | 256 |

### Create Complaint-User

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Data Element** | **Control type** | **Default**  **Valu es** | **Editable**  **Field** | **Restriction s** | **Mandatory/**  **Non**  **Mandatory** | **Data Type** | **Data**  **Size** |
| Complaint ID | Textbox | Yes | No | Only Digits | Mandatory | INT |  |
| User ID | Textbox | Yes | No | Only Digits | Mandatory | INT |  |
| Phone Number | Textbox | Yes | Yes | Should be 10 digits long | Mandatory | Varchar | 15 |
| Category | Drop Down | No | Yes | - | Mandatory | Varchar | 45 |
| Complaint | Textbox | No | Yes | Minimum 10 words | Mandatory | Varchar | 1024 |

### Complaint View-User

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Data Element** | **Control type** | **Default**  **Valu es** | **Editable**  **Field** | **Restriction s** | **Mandatory/**  **Non**  **Mandatory** | **Data Type** | **Data**  **Size** |
| Complaint ID | Textbox | Yes | No | Only Digits | Mandatory | INT |  |
| User ID | Textbox | Yes | No | Only Digits | Mandatory | INT |  |
| Phone Number | Textbox | Yes | No | Should be 10 digits long | Mandatory | Varchar | 15 |
| Category | Drop Down | Yes | No | - | Mandatory | Varchar | 45 |
| Complaint | Textbox | Yes | No | Minimum 10 words | Mandatory | Varchar | 1024 |

### Complaint View-Analyst

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Data Element** | **Control type** | **Default**  **Valu es** | **Editable**  **Field** | **Restriction s** | **Mandatory/**  **Non**  **Mandatory** | **Data Type** | **Data**  **Size** |
| Complaint ID | Textbox | Yes | No | Only Digits | Mandatory | INT |  |
| User ID | Textbox | Yes | No | Only Digits | Mandatory | INT |  |
| Phone Number | Textbox | Yes | No | Should be 10 digits long | Mandatory | Varchar | 15 |
| Email | Textbox | No | Yes | Should be a Proper email ID Format | Mandatory | Varchar | 45 |
| Category | Drop Down | Yes | No | - | Mandatory | Varchar | 45 |
| Status | Drop Down | Yes | Yes | - | Mandatory | Varchar | 45 |
| Assigned Analyst | Drop Down | Yes | Yes | - | Mandatory | Varchar | 45 |
| Complaint | Text Area | Yes | No | Minimum 10 words | Mandatory | Varchar | 1024 |

### Complaint View-Admin

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Data Element** | **Control type** | **Default**  **Valu es** | **Editable**  **Field** | **Restriction s** | **Mandatory/**  **Non**  **Mandatory** | **Data Type** | **Data**  **Size** |
| Complaint ID | Textbox | Yes | No | Only Digits | Mandatory | INT |  |
| User ID | Textbox | Yes | No | Only Digits | Mandatory | INT |  |
| Phone Number | Textbox | Yes | No | Should be 10 digits long | Mandatory | Varchar | 15 |
| Email | Textbox | No | Yes | Should be a Proper email ID Format | Mandatory | Varchar | 45 |
| Category | Drop Down | Yes | No | - | Mandatory | Varchar | 45 |
| Status | Drop Down | Yes | Yes | - | Mandatory | Varchar | 45 |
| Assigned Analyst | Drop Down | Yes | Yes | - | Mandatory | Varchar | 45 |
| Complaint | Textbox | Yes | No | Minimum 10 words | Mandatory | Varchar | 1024 |

### Add Feedback-Admin

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Data Element** | **Control type** | **Defa ult**  **Valu es** | **Editable**  **Field** | **Restrictions** | **Mandatory/**  **Non**  **Mandatory** | **Data Type** | **Data**  **Size** |
| Question  Description | Textbox | No | Yes | - | Mandatory | Varchar | 1024 |

### Email View-Analyst

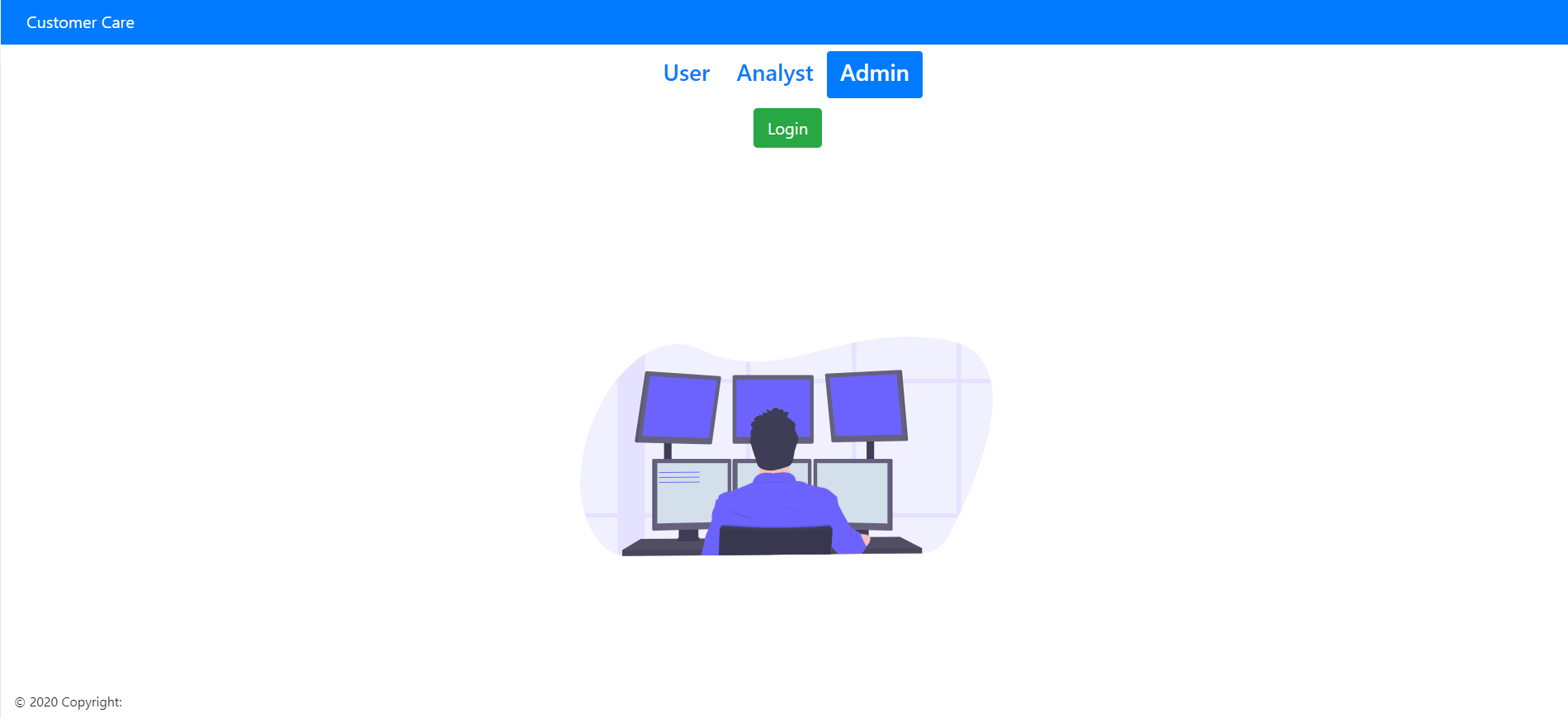
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Data Element** | **Control type** | **Default**  **Valu es** | **Editable**  **Field** | **Restriction s** | **Mandatory/**  **Non**  **Mandatory** | **Data Type** | **Data**  **Size** |
| Sent By | Textbox | Yes | No | - | Mandatory | Varchar | 45 |
| Recipient | Textbox | Yes | No | - | Mandatory | Varchar | 45 |
| Date | Textbox | Yes | No | - | Mandatory | Date | - |
| Description | Text Area | Yes | No | - | Mandatory | Varchar | 1024 |

### Email-Admin

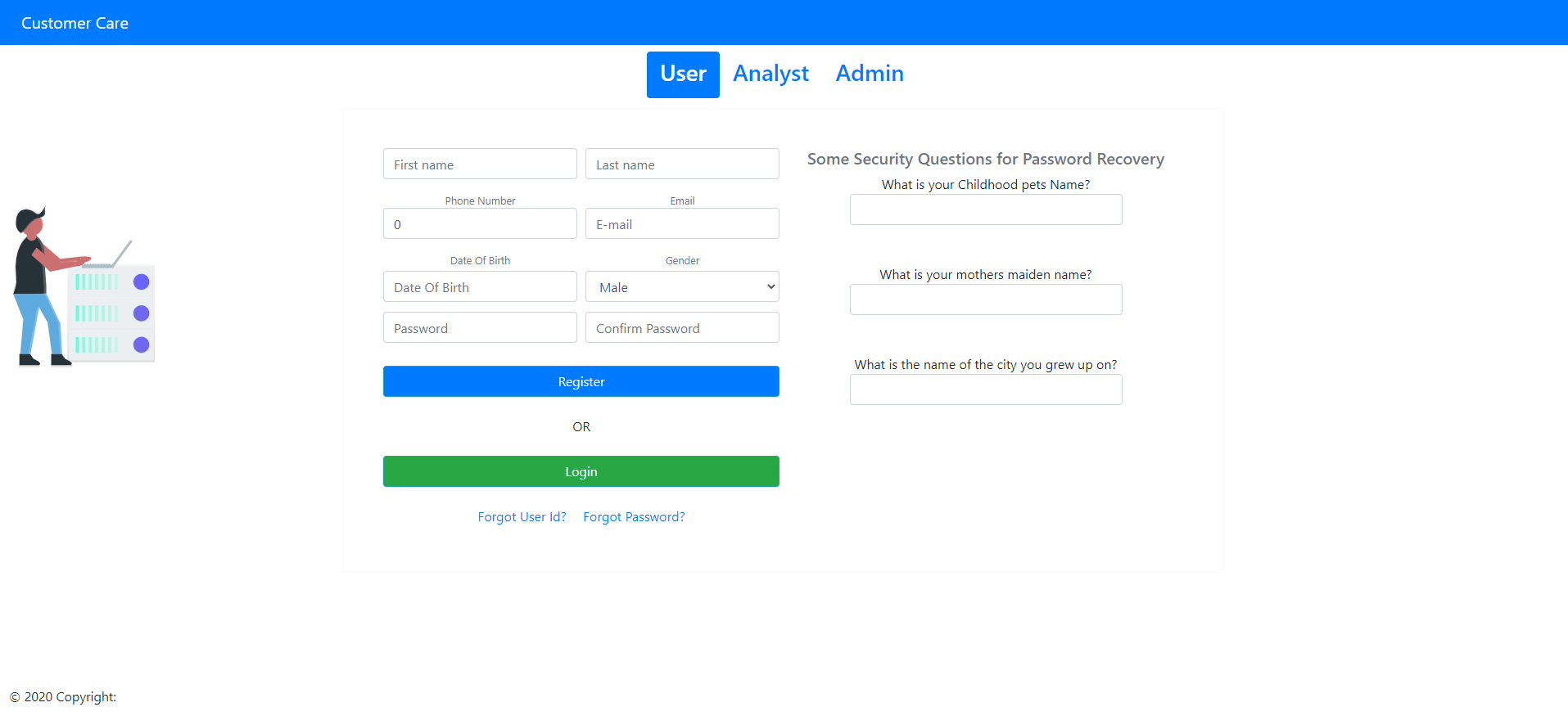
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Data Element** | **Control type** | **Default**  **Valu es** | **Editable**  **Field** | **Restriction s** | **Mandatory/**  **Non**  **Mandatory** | **Data Type** | **Data**  **Size** |
| Email ID | Textbox | Yes | No | - | Mandatory | INT |  |
| Email To | Textbox | Yes | Yes | - | Mandatory | Varchar | 45 |
| Email To ID | Textbox | Yes | Yes | - | Mandatory | INT |  |
| Date | Textbox | Yes | Yes | - | Mandatory | Date |  |
| Complaint | Textbox | Yes | Yes | - | Mandatory | Varchar | 1024 |
| Mail Description | Text Area | No | Yes | - | Mandatory | Varchar | 1024 |

## UI Design

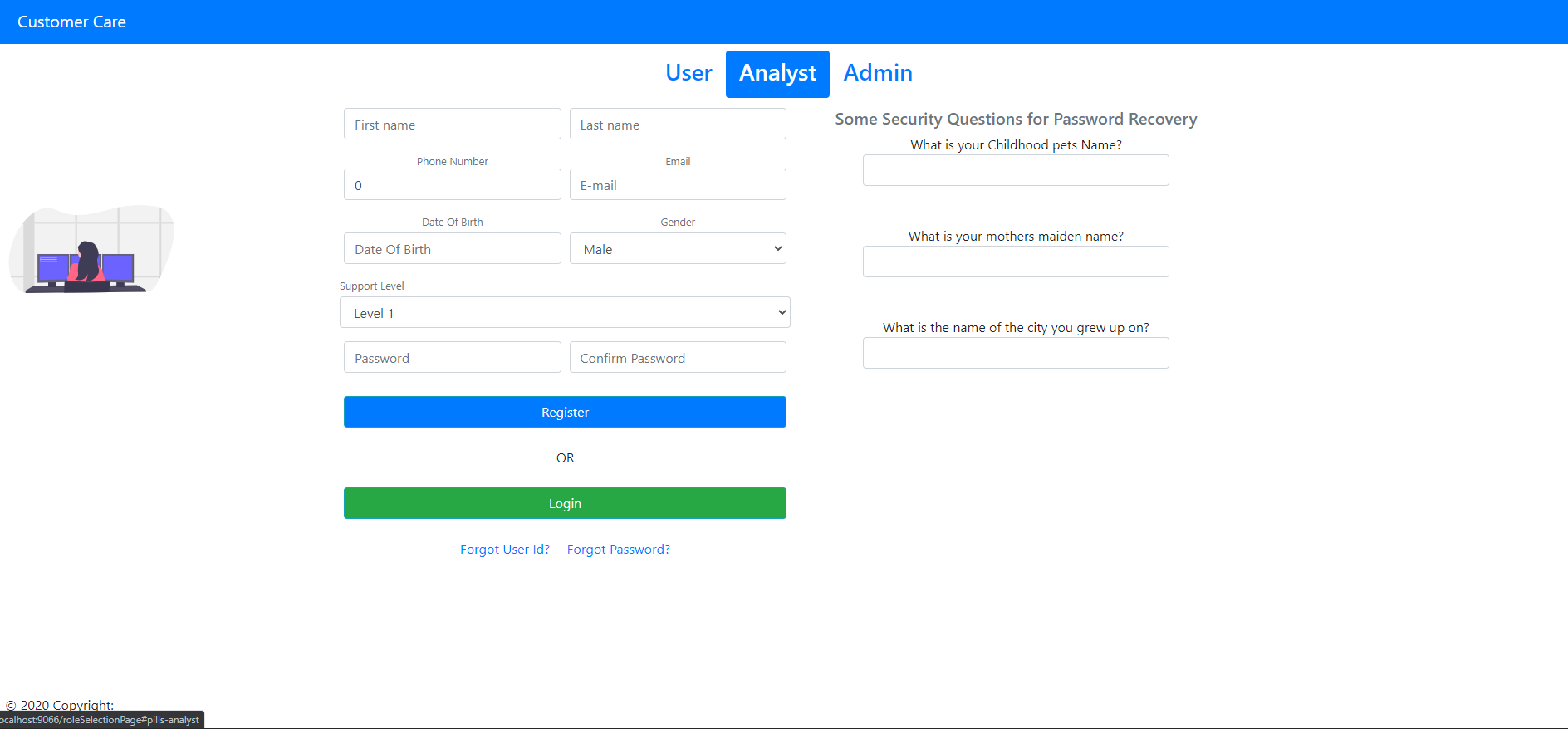
### Admin Login



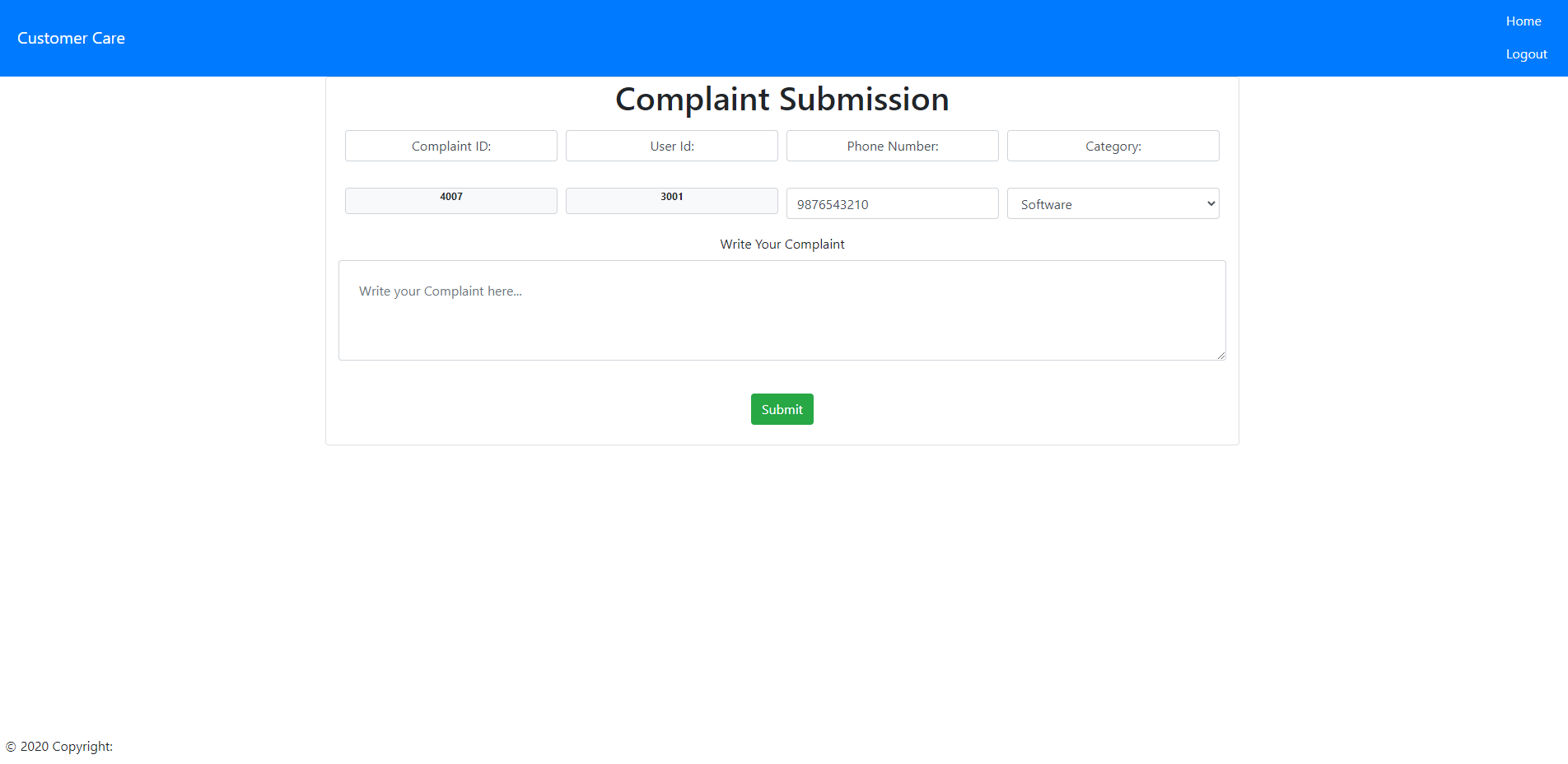
### Register User/Analyst



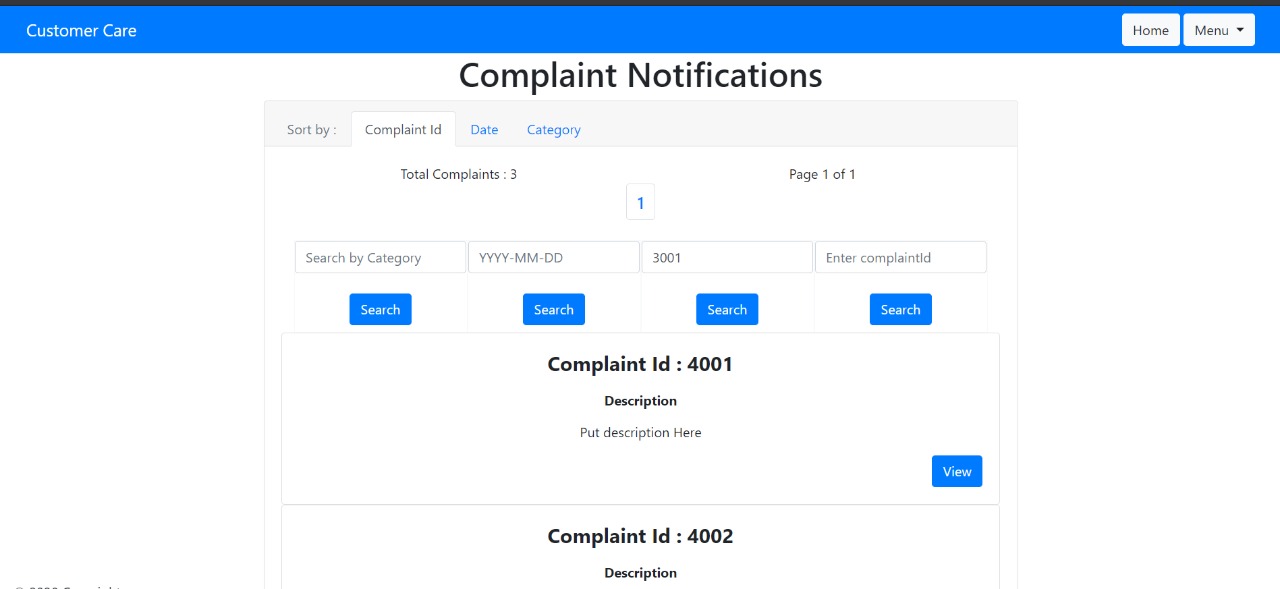
### Register Analyst

****

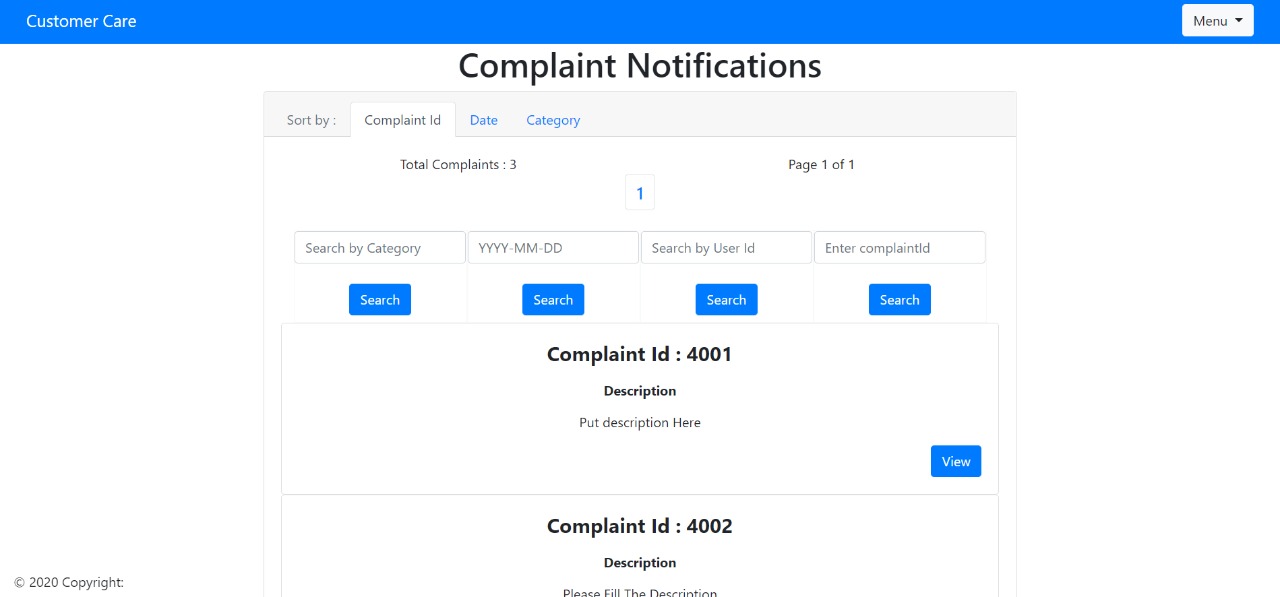
### Create Complaint User

****

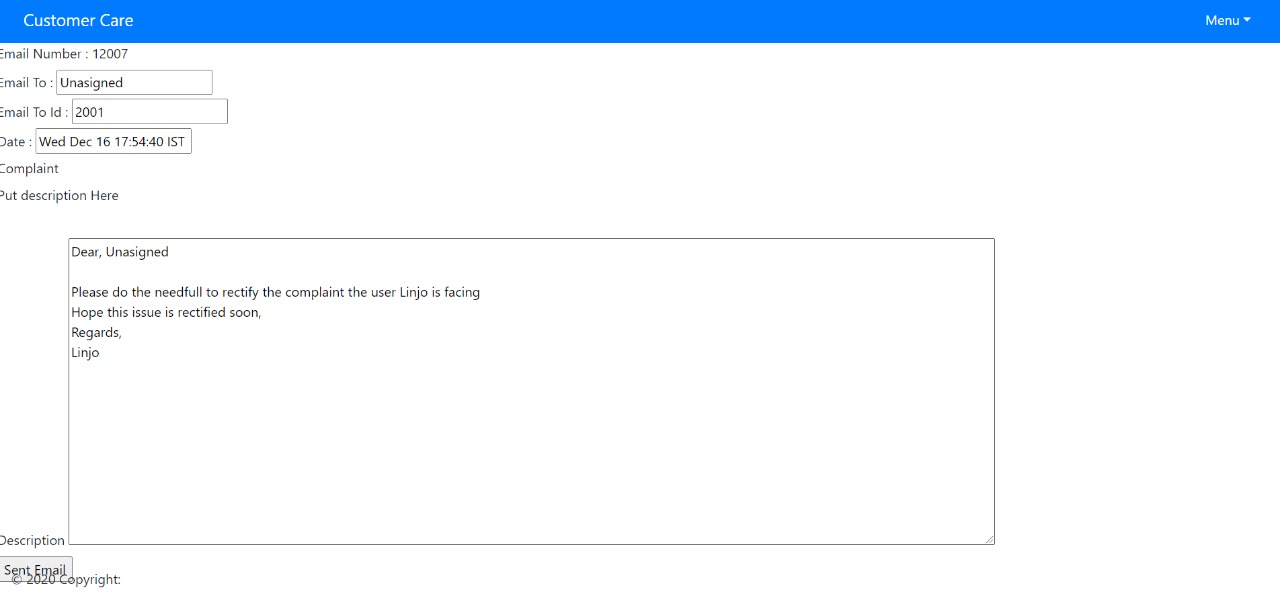
### View Complaint User



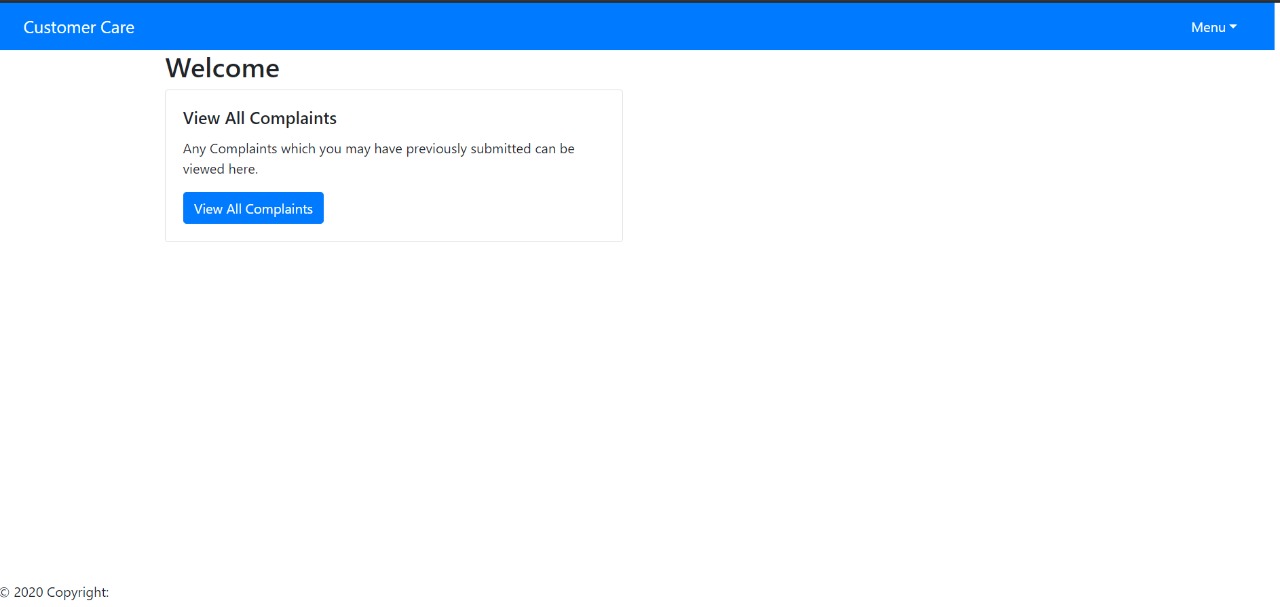
### All Complaints



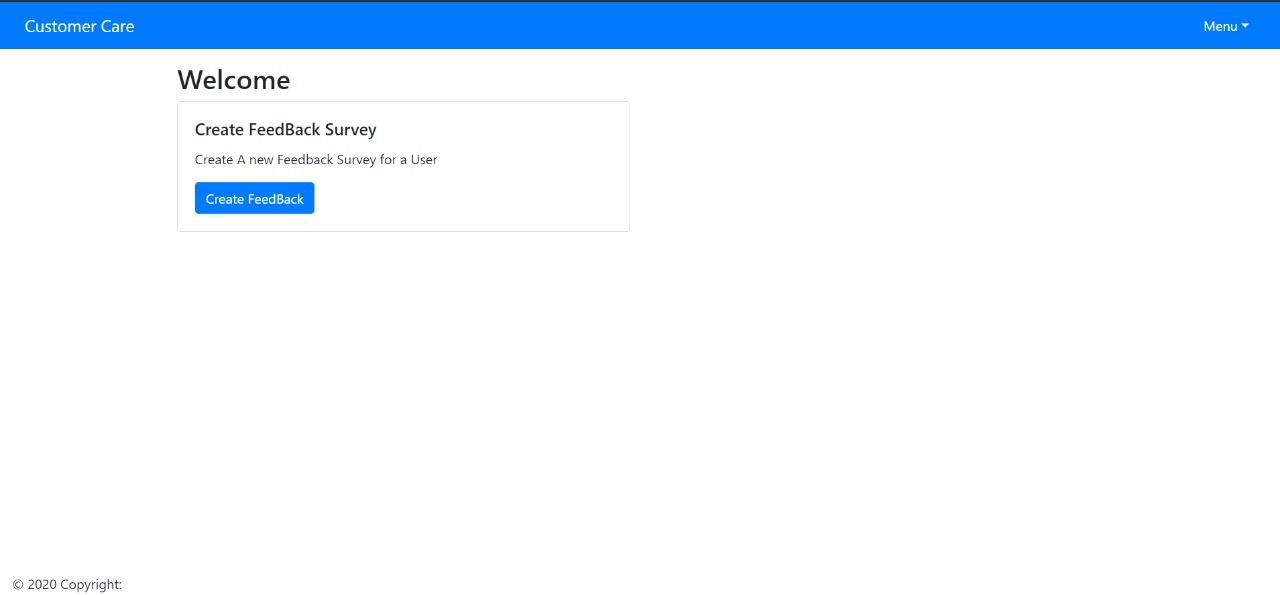
### User to Analyst Mail



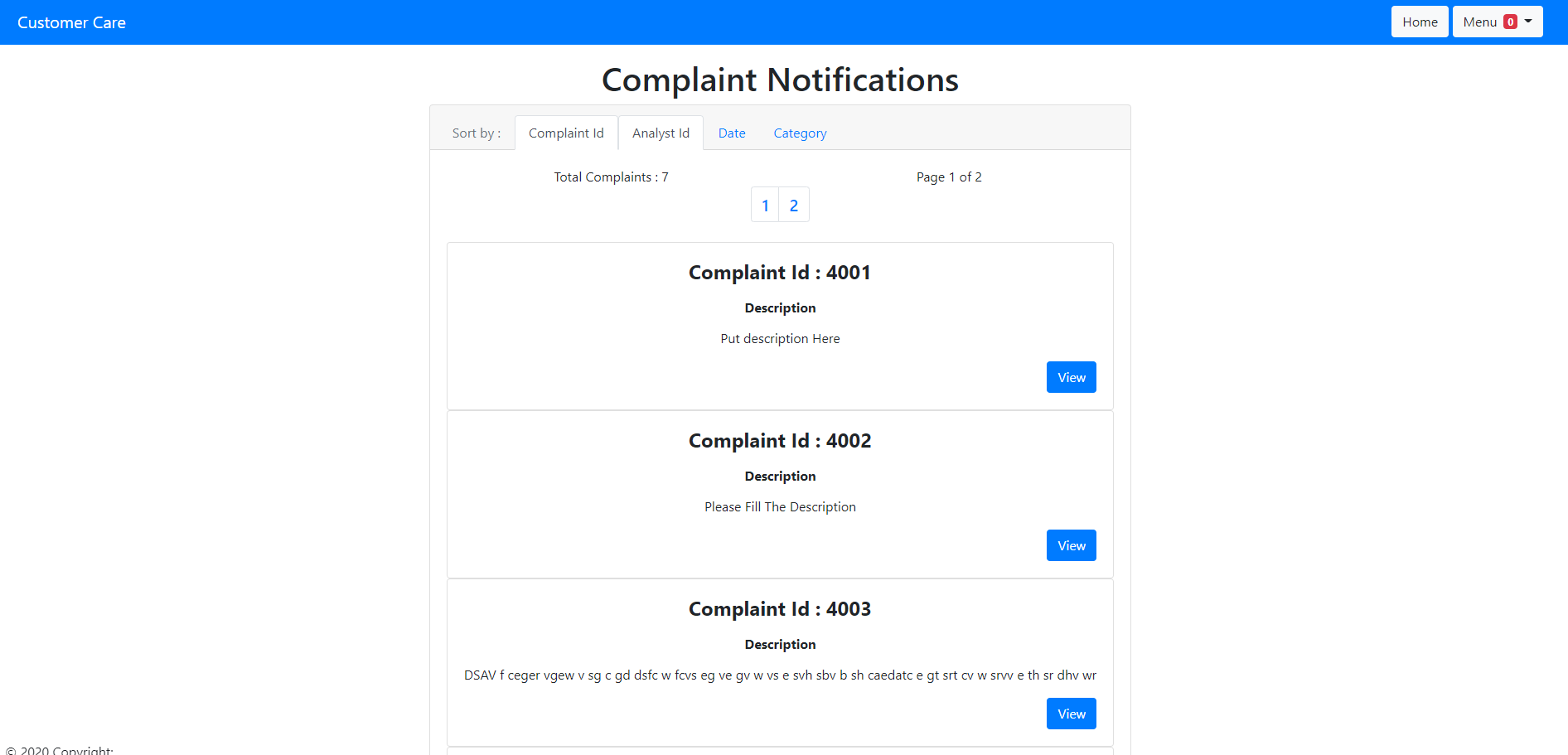
### 5.4.6 Analyst/Admin Home



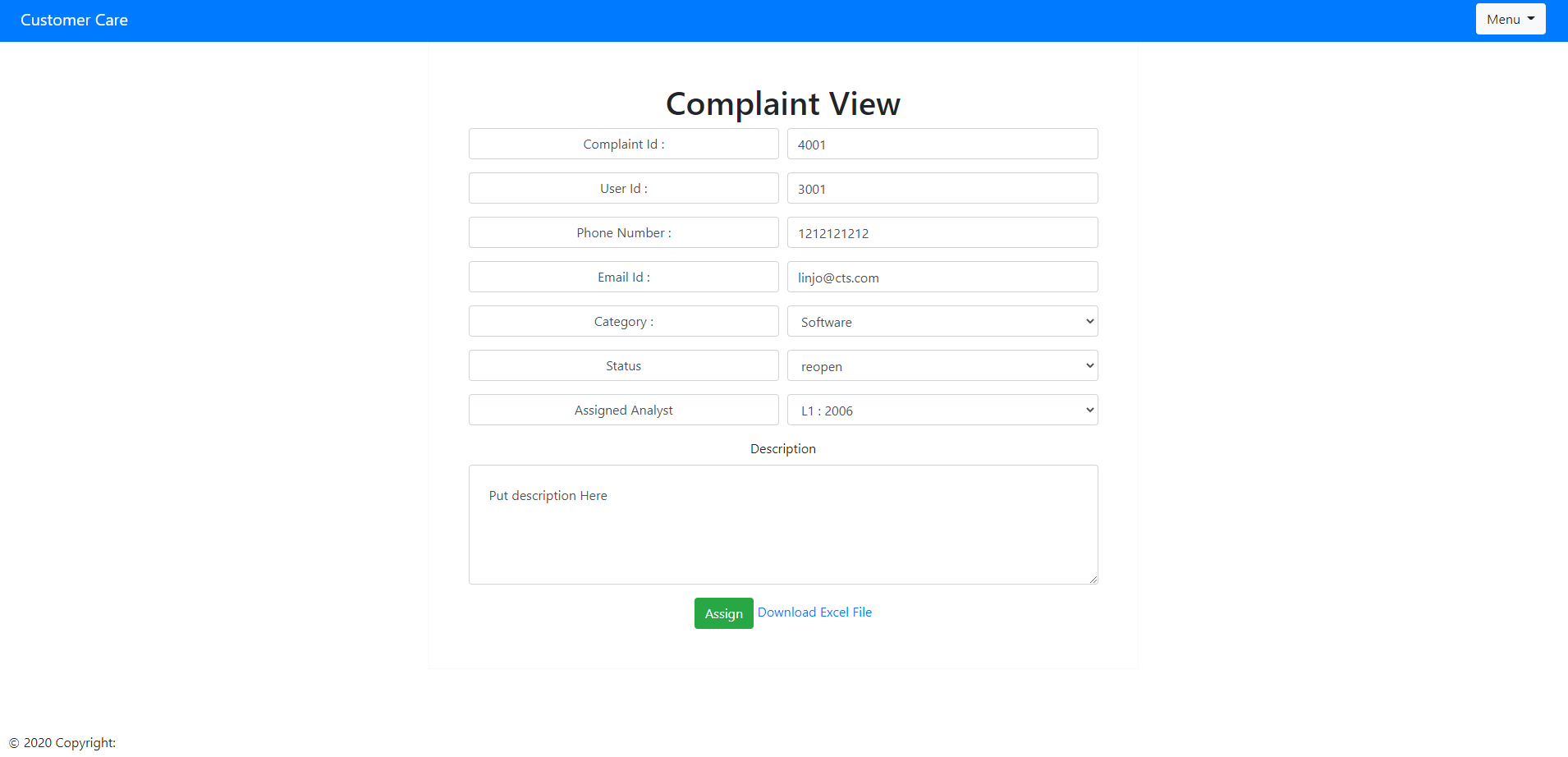
### Admin Home

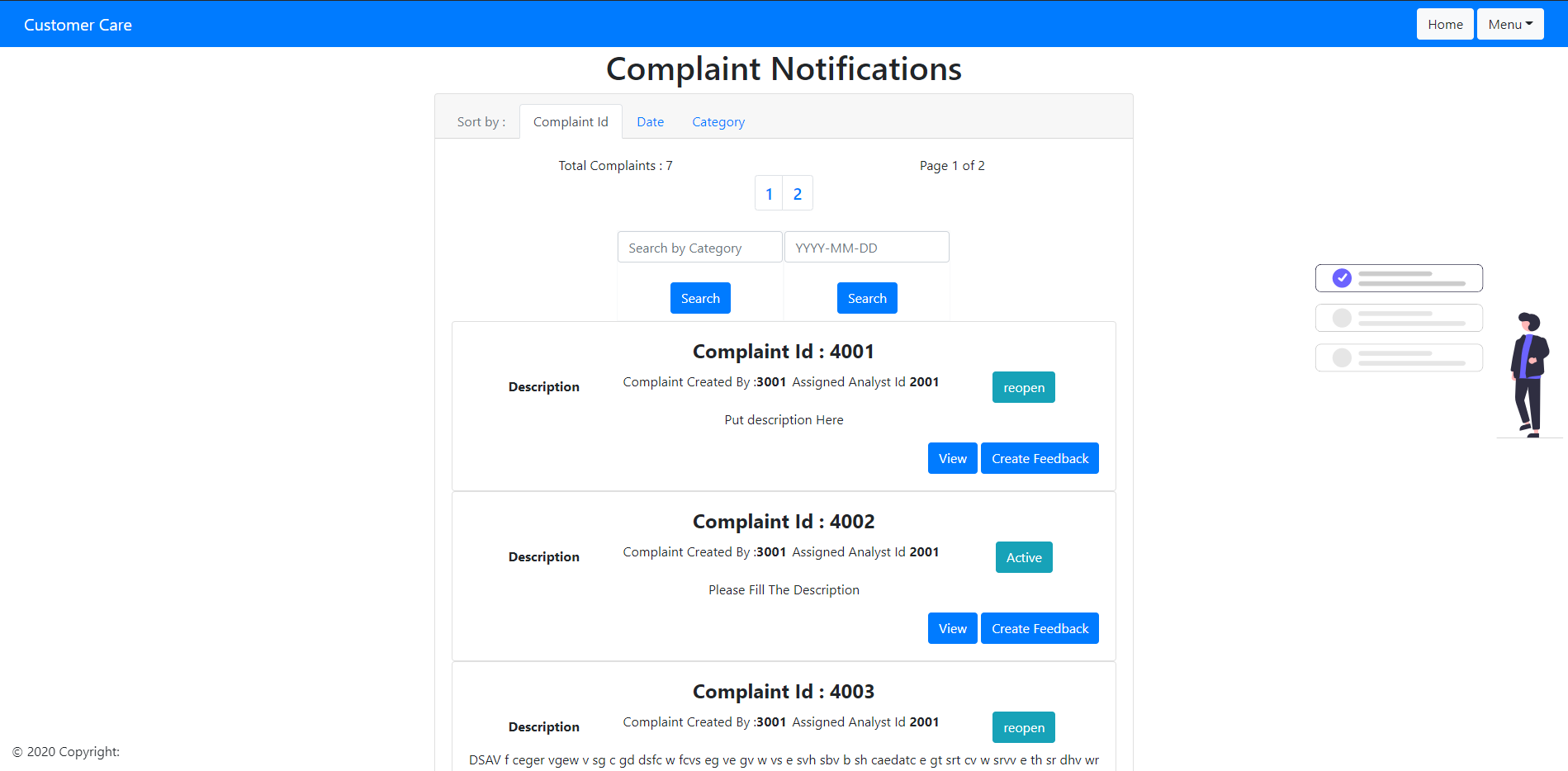


### View All Complaints Analyst/Admin

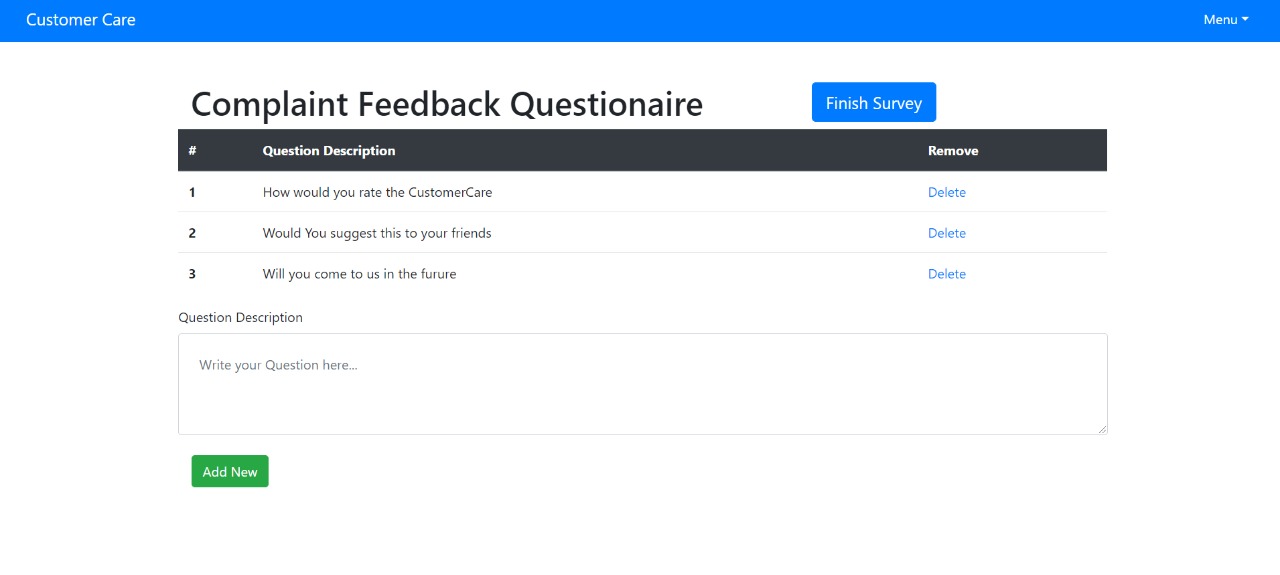


### View Complaint Analyst/Admin

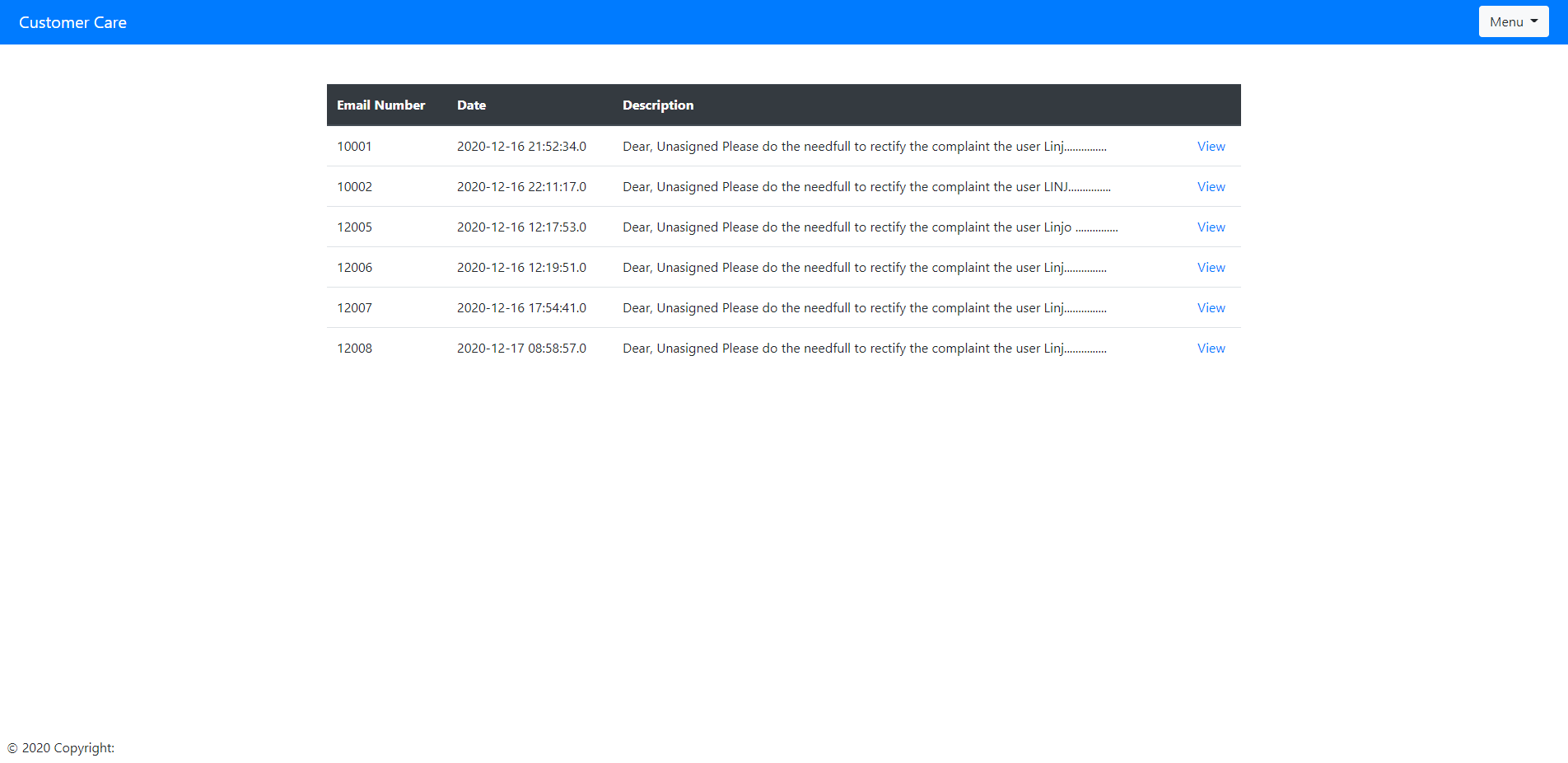




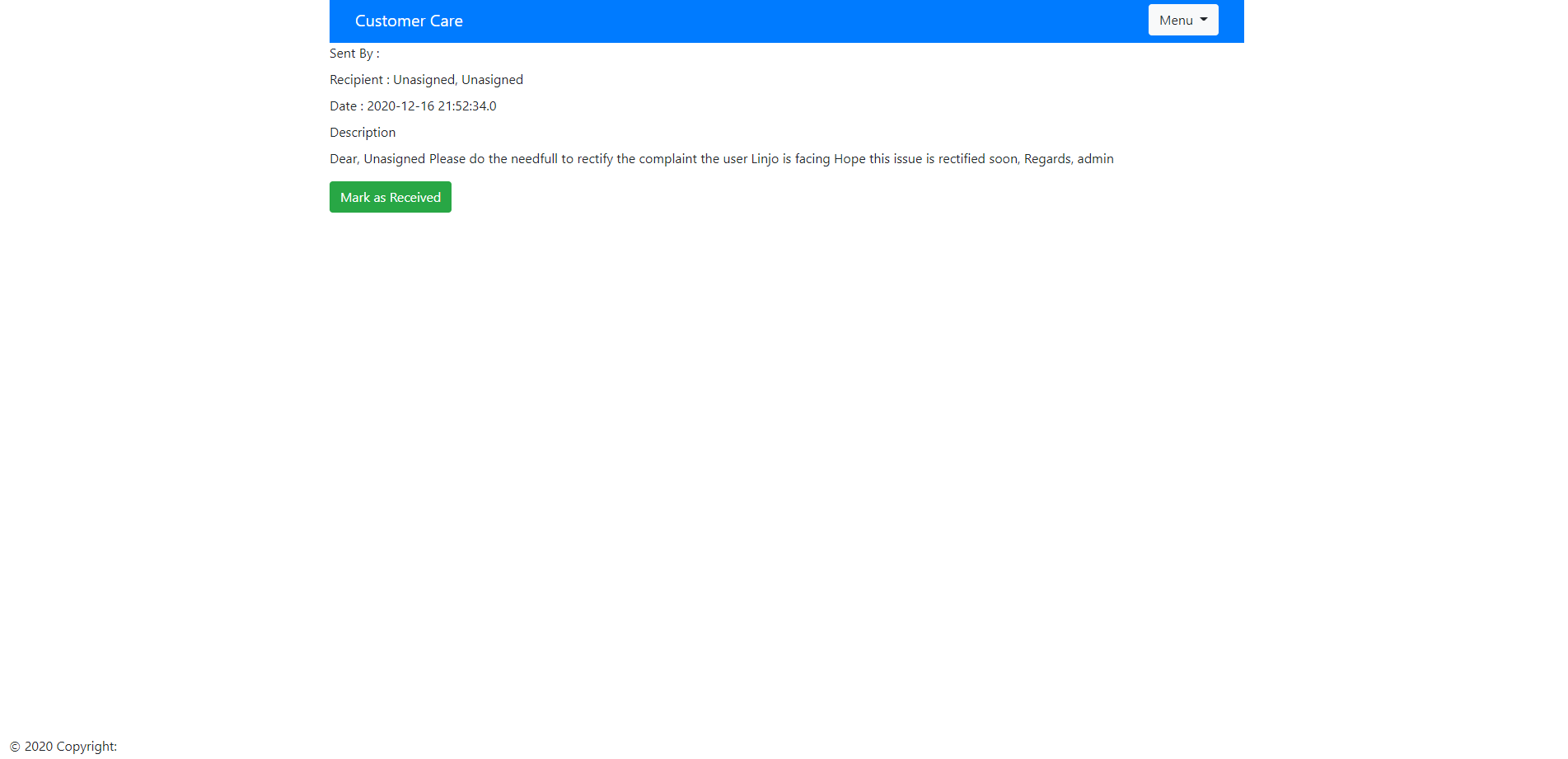
### Create Feedback Admin



### Analyst Emails



### Analyst Email View



## Component Description

### Controller Classes

|  |  |  |
| --- | --- | --- |
| Class | Method | Program Logic |
| AdminController | **initBinder**  **Parameters**  WebDataBinder binder    **Return Type**  void | * Convert date into “ yyyy-MM-dd” format. |
| **getName**  **Parameters**  ModelMap model  **Return Type**  String | * Return the name of user logged in. |
| **goToAdminHome**  **Parameters**  ModelMap model  **Return Type**    String | * Returns the admin-home jsp page. |
| **gotoFeedBackAdmin**  **Parameters**  @ModelAttribute Feedback feedback, @RequestParam long complaintId, ModelMap model  **Return Type**    String | * Returns the feedback-creation-admin jsp page. |
| **addFeedBack**  **Parameters**  @ModelAttribute Feedback feedback, @RequestParam long complaintId, ModelMap model  **Return Type**    String | * Accept Feedback object as parameter and add to database. Then redirect to feedback-creation-admin jsp page. |
| **deleteQuestion**  **Parameters**  @ModelAttribute Feedback feedback,  @RequestParam long responseId, @RequestParam long complaintId, ModelMap model  **Return Type**    String | * Accept response Id as parameter and delete the corresponding feedback. Then redirect to feedback-creation-admin jsp page. |
| **adminAfterLogin**  **Parameters**  @Validated @ModelAttribute Admin admin, BindingResult result, ModelMap model.  **Return Type**    String | * Return complaint-notification-admin jsp page |
| **showUserComplaint**  **Parameters**    @RequestParam long complaintId, ModelMap model  **Return Type**    String | * Return complaint-admin-view jsp page. |
| **updateComplaintAdmin**  **Parameters**    @Validated @ModelAttribute Complaint complaint, BindingResult result, ModelMap model)  **Return Type**    String | * Accept Complaint object as parameter and assign support analyst. * Then add mail content and return admin-to-analyst-mail jsp page. |
| **sentEmail**  **Parameters**    @ModelAttribute EmailAnalyst emailAnalyst, BindingResult results, ModelMap model  **Return Type**    String | * Sent mail to analyst and return admin-home jsp page. |
| **downloadExcelSheet**  **Parameters**    HttpServletResponse response, @RequestParam long complaintId  **Return Type**    void | * Accept complaint Id and provide option to download excel file. |
| **adminViewFilter**  **Parameters**    ModelMap model, String keyword  **Return Type**    String | * Filter based on keyword and return complaint-notification-admin jsp page. |
| **adminViewFilterByDate**  **Parameters**  ModelMap model, String date  **Return Type**    String | * Filter based on date and return complaint-notification-admin jsp page. |
| **showUserComplaints**  **Parameters**    @RequestParam long complaintId, ModelMap model  **Return Type**    String | * Return complaint-view-user jsp page. |
| **updateComplaintUser**  **Parameters**    @Validated @ModelAttribute Complaint complaint, BindingResult result, ModelMap model  **Return Type**    String | * Set mail content and return user-to-analyst-mail jsp page. |
| **sentEmailUser**  **Parameters**  @ModelAttribute EmailUserAnalyst email, BindingResult results, ModelMap model  **Return Type**    String | * Sent mail to analyst and return user-home jsp page. |
| **getCategory()**  **Parameters**  None  **Return Type**  Map<String, String> | * Return category list. |
| **getSupportLevel**  **Parameters**  **None**  **Return Type**  Map<String, String> | * Return supportLevel list. |
| **getStatus()**  **Parameters**  **Return Type**  Map<String, String> | * Return sstatus list. |
| **mailMessage()**  **Parameters**  **None**  **Return Type**  String[] | * Return mail body, |
| AnalystController | **getName**  **Parameters**    ModelMap model  **Return Type**  String | * Return the name of user logged in. |
| **analystGoToHome**  **Parameters**    ModelMap model  **Return Type**  String | * Return analyst-home jsp page. |
| **viewComplaintAnalyst**  **Parameters**    @RequestParam long complaintId, ModelMap model  **Return Type**  String | * Return complaint-analyst-view jsp page. |
| **viewAnalystEmails**  **Parameters**    ModelMap model  **Return Type**  String | * Return emails-analyst jsp page. |
| **showUserComplaint**  **Parameters**    @RequestParam long complaintId, ModelMap model  **Return Type**  String | * Accept complaint Id as parameter and add complaint details and return complaint-analyst-view jsp page. |
| **updateComplaintAdmin**  **Parameters**    @Validated @ModelAttribute Complaint complaint,  BindingResult result, ModelMap model  **Return Type**  String | * Set mail content and return analyst -to-analyst-mail jsp page. |
| **sentEmail**  **Parameters**    @ModelAttribute EmailAnalyst emailAnalyst, BindingResult results, ModelMap model  **Return Type**  String | * Sent mail to analyst and return analyst -home jsp page. |
| **viewEachEmailAnalyst**  **Parameters**    @RequestParam long emailId, ModelMap model  **Return Type**  String | * Return each-email-analyst jsp page. |
| **viewEachEmailAnalystFromUser**  **Parameters**    @RequestParam long emailId, ModelMap model  **Return Type**  String | * Return each-email-analyst-from-user jsp page. |
| **analystAfterLogin**  **Parameters**    @Validated @ModelAttribute Analyst analyst, BindingResult result, ModelMap model  **Return Type**  String | * Return complaint-notification-analyst jsp page. |
| **forgotAnalystId**  **Parameters**    ModelMap model  **Return Type**  String | * Return forgot-analyst-Id jsp page. |
| **forgotAnalystIdSecretQuestion**  **Parameters**    ModelMap model, String mail  **Return Type**  String | * Accept Email Id as parameter and find corresponding analyst. * If a valid analyst is obtained, then return forgot-analyst-id-sq-question jsp page else return forgot-analyst-id jsp page. |
| **secretQuestions**  **Parameters**    ModelMap model, @PathVariable long analystId, String ans1, String ans2, String ans3  **Return Type**  String | * Accept answers of security questions as parameter and find corresponding analyst. * If a valid analyst is obtained, then return display-analyst-id jsp page else return forgot-analyst-id jsp page. |
| **recoverPassword**  **Parameters**    ModelMap model, String analystId, String mob, String email  **Return Type**  String | * Accept user Id, email Id and mobile number as parameter and find corresponding analyst. * If a valid analyst is obtained, then password-recovery-analyst jsp page else return forgot-password-analyst jsp page. |
| **verifySecretQuestion**  **Parameters**    ModelMap model, @PathVariable long analystId, String ans1, String ans2,String ans3  **Return Type**  String | * Accept answers of security questions as parameter and verify analyst. * If a valid analyst is obtained, then return set-new-pwd-analyst jsp page else return password-recovery-analyst jsp page. |
| **changePassword**  **Parameters**    String newPwd, String confirmPwd, long analystId, ModelMap model  **Return Type**  String | * Accept password as parameter. * If password is valid, then set as new password and return analyst-home jsp page.Else return set-new-pwd-analyst jsp page. |
| **getCategory**  **Parameters**  None  **Return Type**  Map<String, String> | * Return category list. |
| **getSupportLevel**  **Parameters**  None  **Return Type**  Map<String, String> | * Return support level list. |
| **getStatus**  **Parameters**  None  **Return Type**  Map<String, String> | * Return sstatus list. |
| **mailMessage**  **Parameters**  None  **Return Type**  String[] | * Return mail body. |
| UserController | **getName**  **Parameters**    ModelMap model  **Return Type**  String | * Returns the name of user logged in. |
| **userGoToHome**  **Parameters**    ModelMap model  **Return Type**  String | * Return user-home jsp page. |
| **userListThisUsers**  **Parameters**    ModelMap model  **Return Type**  String | * Accept the parameters and add complaint details of user to page and return complaint-notification-user-personnel jsp page. |
| **userListThisUsersNextPage**  **Parameters**    ModelMap model, @PathVariable int pageNumber, @Param String sortBy,@Param String sortDir, String keyword, String date, String complaintId  **Return Type**  String | * Accept the parameters and add complaint details of user to page and return complaint-notification-user-personnel jsp page. |
|  | **userListAll**  **Parameters**    @Validated @ModelAttribute User user, BindingResult result, ModelMap model  **Return Type**  String | * Add complaint details of all users to page and return complaint-notification-user jsp page. |
| **viewAnotherPageUserListAll**  **Parameters**    @Validated @ModelAttribute User user, BindingResult result,ModelMap model, @PathVariable int pageNumber, @Param String sortBy, @Param String sortDir, String keyword, String date, String userId, String complaintId  **Return Type**  String | * Filter the coplaints based on category, date, user Id and complaint Id and return complaint-notification-user jsp page. |
| **submitFeedback**  **Parameters**    @ModelAttribute Complaint complaint, @RequestParam long complaintId, ModelMap model  **Return Type**  String | * Accept complaint Id as parameter and get complaint and feedback details and return user-feedback-submission jsp page. |
| **afterSubmitFeedBack**  **Parameters**    @ModelAttribute Complaint complaint, BindingResult result, ModelMap model  **Return Type**  String | * Set answer for feedback questions and add it to the complaint. * Then return to user-home jsp page. |
| **recoverPassword**  **Parameters**    ModelMap model,String userId,String mob,String email  **Return Type**  String | * Accept user Id, email Id and mobile number as parameter and find corresponding user. * If a valid user is obtained, then password-recovery- user jsp page else return forgot-password- user jsp page. |
| **verifySecretQuestion**  **Parameters**    ModelMap model,@PathVariable long userId,String ans1,String ans2,String ans3  **Return Type**  String | * Accept answers of secret questions as parameter and verify user. * If a valid user is obtained, then return set-new-pwd- user jsp page else return password-recovery- user jsp page. |
| **changePassword**  **Parameters**    String newPwd, String confirmPwd, long userId, ModelMap model  **Return Type**  String | * Accept password as parameter. * If password is valid, then set as new password and return user-home jsp page.Else return set-new-pwd-user jsp page. |
| **showForgotUserId**  **Parameters**  **Return Type**  String | * Return forgot-user-id jsp page. |
| **showSecretQuestions**  **Parameters**    ModelMap model,String email  **Return Type**  String | * Accept email Id as parameter and find corresponding user. * If a valid user is obtained, then forgot-user-id-sq-question jsp page else return forgot-user-id jsp page. |
| **showUserId**  **Parameters**    ModelMap model,@PathVariable long userId,String ans1,String ans2,String ans3  **Return Type**  String | * Accept answers of secret questions as parameter and verify user. * If a valid user is obtained, then return show-user-id jsp page else return forgot-user-id-sq-question jsp page. |
| ComplaintController | **getName**  **Parameters**    ModelMap model  **Return Type**  String | * Return the name of user logged in. |
| **goToComplaintPage**  **Parameters**    @ModelAttribute Complaint complaint, BindingResult result, ModelMap model  **Return Type**  String | * Add the complaint template and return complaint-creation jsp page. |
| **registerComplaint**  **Parameters**    @Validated @ModelAttribute Complaint complaint, BindingResult result, @RequestParam long complaintId, ModelMap model  **Return Type**  String | * Accept complaint as parameter. If a valid is received then set the complaint details and return complaint-submission-user jsp page. * Else return complaint-creation jsp page. |
| **viewAllComplaintUser**  **Parameters**    @ModelAttribute Complaint complaint, BindingResult result, ModelMap model  **Return Type**  String | * Add all complaints and return complaint-notification-user jsp page. |
| **viewAnotherPageAdminComplaintList**  **Parameters**    @Validated @ModelAttribute Admin admin,  BindingResult result, ModelMap model, @PathVariable int pageNumber, @Param String sortBy, @Param String sortDir, String keyword, String date  **Return Type**  String | * Sort complaints based on id, date, category and return complaint-notification-admin jsp page. |
| **viewAnotherPageAnalystComplaintList**  **Parameters**    @Validated @ModelAttribute Analyst analyst, BindingResult result, ModelMap model, @PathVariable int pageNumber, @Param String sortBy, @Param String sortDir  **Return Type**  String | * Sort complaints based on id, date,category and return complaint-notification-analyst jsp page. |
| **getCategories()**  **Parameters**  None  **Return Type**  Map<String, String> | * Return complaint category list. |
| LoginController | **getName**  **Parameters**    ModelMap model  **Return Type**  String | * Return the name of user logged in. |
| **userLogin**  **Parameters**    ModelMap model, String error, String logout  **Return Type**  String | * Return login jsp page. |
| **Redirect**  **Parameters**  ModelMap model  **Return Type**  String | * If id between 1000 and 2000 is received then redirect to admin-home jsp page. * If id between 2000 and 3000 is received then redirect to analyst-home jsp page. * If id between 3000 and 4000 is received then redirect to user-home jsp page. * Else return admin-home jsp page. |
| **getCategory**  **Parameters**  None  **Return Type**  Map<String, String> | * Return category list. |
| RegistrationController | **initBinder**  **Parameters**  WebDataBinder binder  **Return Type**  void | * Convert date into “ yyyy-MM-dd” format. |
| **registerAnalyst**  **Parameters**  @Validated @ModelAttribute Analyst analyst, BindingResult result, ModelMap model  **Return Type**  String | * If a valid analyst is received as parameter, then set the password and secret questions and save analyst to database and return analyst-reg-status jsp page. * Else return role-selection jsp page. |
| **registerUser**  **Parameters**    @Validated @ModelAttribute User user, BindingResult result, ModelMap model  **Return Type**  String | * If a valid user is received as parameter, then set the password and secret questions and save user to database and return user-reg-status jsp page. * Else return role-selection jsp page. |
| **populateGender**  **Parameters**  **Return Type**  List<String> | * Return list. |
| **getCategories**  **Parameters**  **Return Type**  Map<String, String> | * Return category list. |
| RoleSelectionController | **roleSelectionPage**  **Parameters**    @ModelAttribute Admin admin, @ModelAttribute Analyst analyst,  @ModelAttribute User user, BindingResult result, ModelMap model  **Return Type**  String | * Return role-selection jsp for Admin, Analyst and user. |
| **populateGender**  **Parameters**  **Return Type**  List<String> | * Return list. |
| **getCategories**  **Parameters**  **Return Type**  Map<String, String> | * Return category list. |

### Service Classes

|  |  |  |  |
| --- | --- | --- | --- |
| Class | Method | Parameter Logic | |
| AnalystService | getLastId()  **Parameters**  None  **Return Type**  long lstId | * Call the getLastId() of AnalystService * Check whether analystId greater than 2000 if it is greater then return analystId | |
| getAnalyst()  **Parameters**  long analystId  **Return Type**  User | * Call the getAnalystr() of AnalystService by passing analystId * it will return corresponding user | |
| addAnalyst()  **Parameters**  Analyst analyst  **Return Type**  boolean | * Call the addAnalyst() of AnalystService by passing analyst * It will return true if analyst is added | |
| getAllAnalystNotOfSupportLevel()  **Parameters**  String supportLevel  **Return Type**  List<Analyst> | * Call the getAllAnalystNotOfSupportLevel()of AnalystService by passing analyst by passing supportLevel * It will return list of analyst. | |
|  | | |
|  | getAllAnalyst()  **Parameters**  None  **Return Type**  List<Analyst> | * Call the getAllAnalyst() of AnalystService. It will return the list of Analyst | |
| deleteAnalyst()  **Parameters**  long AnalystId  **Return Type**  void | * Call the deleteAnalyst() of AnalystService by passing analystId parameter to delete the corresponding analyst | |
| deleteAnalyst()  **Parameters**  Analyst  **Return Type**  void | * Call the deleteAnalyst() of AnalystService by passing analyst parameter to delete the corresponding analyst | |
| updateAnalyst()  **Parameters**  Analyst  **Return Type**  Analyst | * Call the updateAnalyst() of AnalystService by passing Analyst to update the Analyst. | |
| getPasswordSHA()  **Parameters**  long analystId  **Return Type**  String | * Call the getPasswordSHA() of AnalystService by passing AnalystId to get the encrypted password. | |
| checkAnswer()  **Parameters**  List<analystSecretQuestion>  String ans1  String ans2  String ans3  **Return Type**  Boolean | * Call the CheckAnswer() of AnalystService by passing list of secret questions and answers to check whether answers typed by the analyst is correct or not to change password. | |
| findAnalyst()  **Parameters**  long analystId  String mob  String email  **Return Type**  Analyst | * Call the findAnalyst() of AnalystService by passing AnalystId,mob,email to get the corresponding analyst | |
| getLastId()  **Parameters**  None  **Return Type**  long lstId | * Call the getLastId() of AnalystSecretQuestionService * Check whether analystSecretQuestion id greater than 8000 if it is greater then return Id | |
| EmailAnalystService | getLastId()  **Parameters**  None  **Return Type**  long | * Call the getLastId () of   AnalystSecretQuestionService | |
| addMail()  **Parameters**  EmailAnalyst email  **Return Type**  boolean | * Call the addEmail() of EmailAnalystService by passing email * It will return true if mail is added. | |
| getEmailAnalyst()  **Parameters**  long emailId  **Return Type**  EmailAnalyst | * Call the getEmailAnalyst() of EmailAnalystService by passing emailId * It will return EmailAnalyst | |
| EmailUserAnalystService | getLastId()  **Parameters**  None  **Return Type**  long lstId | * Call the getLastId() of EmailUserAnalystService * Check whether email is not empty and email id greater than 12000 if it is greater then return Id | |
| addEMail()  **Parameters**  EmailUserAnalyst email  **Return Type**  boolean | * Call the addEmail() of EmailUserAnalystService by passing email * It will return true if mail is added. | |
| geEtMail()  **Parameters**  long emailId  **Return Type**  EmailUserAnalyst | * Call the getEmail() of EmailUserAnalystService by passing emailId * It will return EmailUserAnalyst | |
| FeedbackService | addFeedback()  **Parameters**  Feedback feedback  **Return Type**  boolean | * Call the addFeedback() of FeedbackService by passing feedback * It will return true if feedback is added. | |
| getFeedback()  **Parameters**  long responseId  **Return Type**  Feedback | * Call the getFeedback() of FeedbackService by passing response id * It will return feedback object | |
| getAllFeedback()  **Parameters**  None  **Return Type**  List<Feedback> | * Call the getAllFeedback() of FeedbackService .It will return list of feedback objects. | |
| updateFeedback()  **Parameters**  Feedback feedback  **Return Type**  Boolean | * Call the updateFeedback() of FeedbackService by passing feedback * It will return true if it is updated | |
| deleteFeedback()  **Parameters**  long responseId  **Return Type**  Void | * Call the deleteFeedback() of FeedbackService by passing response id | |
| deleteFeedback()  **Parameters**  Feedback feedback  **Return Type**  Void | * Call the deleteFeedback() of FeedbackService by passing feedback . | |
| LoginDetailsService | getLoginDetailsByRegisteredId()  **Parameters**  long registeredId  **Return Type**  LoginDetails | * Call the getLoginDetailsByRegisteredId() of LoginDetailsService by passing registeredId. * It will return loginDetails | |
| addLoginDetails()  **Parameters**  LoginDetails loginDetails  **Return Type**  boolean | * Call the addLoginDetails() of LoginDetailsService by passing loginDetails. * It will return true if it is added. | |
| modifyLoginDetails()  **Parameters**  LoginDetails loginDetails  **Return Type**  boolean | * Call the modifyLoginDetails() of LoginDetailsService by passing loginDetails. * It will return true if it is updated. | |
| getQuestionById()  Parameters  long questionId  Return Type  SecretQuestions | * Call the getQuestionById() of SecretQuestionService by passing questionId. * It will return SecretQuestions object. | |
| getAllQuestions()  Parameters  None  Return Type  List<SecretQuestion> | * Call the getQuestion() of SecretQuestionService it will return list of secret questions. | |
| getAllExcept()  Parameters  List<SecretQuestion> alreadyAdded  Return Type  List<SecretQuestion> | * Call the getAllExcept() of SecretQuestionService by using list of secret questions it will return list of secret questions. | |
| getAllQuestionDescription()  Parameters  None  Return Type  List<String> | * Call the getAllQuestionDescription() of SecretQuestionService it will return list. | |
| getLastId()  Parameters  None  Return Type  long lstId | * Call the getLastId() of UserSecretQuestionService * Check whether user secret question id is greater than 7000 if it is greater then return question id. | |
| addSecretQuestion()  Parameters  UserSecretQuestion question  Return Type  boolean | * Call the addSecretQuestion() of UserSecretQuestionService by passing question.it will return true if it is added. | |
| getLastId()  Parameters  None  Return Type  long | * Call the getLastId() of ComplaintService .it will return id. | |
| addComplaint()  Parameters  Complaint complaint  Return Type  Boolean | * Call the addComplaint() of ComplaintService by passing complaint.it will return true if complaint is added. | |
| getComplaint()  Parameters  long complaintId  Return Type  Compalint | * Call the getComplaint() of ComplaintService by passing complaintId.it will return Complaint object. | |
| updateComplaint()  Parameters  Complant complaint  Return Type  boolean | * Call the updateComplaint() of ComplaintService by passing complaint.it will return true if it is updated. | |
| deleteComplaint()  Parameters  long complaintId  Return Type  Boolean | * Call the deleteComplaint() of ComplaintService by passing complaintId .it will return true if deleted. | |
| getAllComplaintOfUser()  Parameters  long userId  Return Type  List<Complaint> | * Call the getAllComplaintOfUser() of ComplaintService by passing userId .it will return list of complaint. | |
| findByKeyword()  Parameters  String keyword  Return Type  List<Complaint> | * Call the findByKeyword() of ComplaintService by passing keyword .it will return list of complaint. | |
| getAllComplaintForUser()  Parameters  long userId  int pageNumber  int count  String sortBy  String sortDir  Return Type  Page<complaint> | * Call the getAllComplaintUser() of ComplaintUser by passing userId,pageNumber,count,sortBy,sortDir. | |
| getAllComplaint()  Parameters  int pageNumber  int count  String sortBy  String sortDir  Return Type  Page<complaint> | * Call the getAllComplaint() of ComplaintUser by passing count,pageNumber,sortBy,sortDir. It will return page. | |
| getAllComplaintFiltered()  Parameters  String keyword  String date  int pageNumber  int count  String sortBy  String sortDir  Return Type  Page<complaint> | * Call the getAllComplaintFiltered() of ComplaintUser by passing keyword,date,count,pageNumber,sortBy,sortDir. It will return page | |
| getAllComplaintForAnalyst()  Parameters  long analystId  int pageNumber  int count  String sortBy  String sortDir  Return Type  Page<complaint> | * Call the getAllComplaintForAnalyst() of ComplaintUser by passing analystId,count,pageNumber,sortBy,sortDir. It will return page. | |
| findDate()  Parameters  String date  Return Type  List<Complaint> | * Call the findDate() of ComplaintUser by passing date. It will return list of complaint in the date. | |
| findByUserId()  Parameters  String userId  Return Type  List<Complaint> | * Call the findByUserId() of ComplaintUser by passing userId. It will return list of complaint of userId. | |
| findByComplaintId()  Parameters  String complaintId  Return Type  List<Complaint> | * Call the findByComplaintId() of ComplaintUser by passing complaintId. It will return list of complaint . | |
| getAdmin()  Paramenters  long adminId  Return Type  Admin | * Call the getAdmin() of AdminService by passing adminId * To get the Admin having corresponding adminId | |
| getAdminPassword()  Parameters  long adminId  Return Type  String | * Call the getAdminPassword() of AdminService by passing adminId. | |
| getAdminPassword()  Parameters  long adminId  Return Type  String | * Call the getPasswordSHA() of AdminService by passing analystId. | |
| getLastId()  Parameters  None  Return Type  long lstId | * Call the getLastId() of UserService * Check whether userId greater than 3000 if it is greater then return userId | |
| getUser()  Parameters  long userId  Return Type  User | * Call the getUser() of UserService by passing userId it will return corresponding user | |
| getAllUsers()  Parameters  None  Return Type  List<user> | * Call the getAllUsers() of UserService * It will return the list of users | |
| deleteUser()  Parameters  User user  Return Type  boolean | * Call the deleteUser() of UserService by passing user parameter to delete the corresponding user | |
| getPasswordSHA()  Parameters  long userId  Return Type  String | * Call the getPasswordSHA() of UserService by passing userId to get the encrypted password. | |
| findUser()  Parameters  long userId  String mob  String email  Return Type  User | * Call the findUser() of UserService by passing userId,mob,email to get the corresponding user | |
| checkAnswer()  Parameters  List<userSecretQuestion>  String ans1  String ans2  String ans3  Return Type  Boolean | * Call the CheckAnswer() of UserService by passing list of secret questions and answers to check whether answers typed by the user is correct or not to change password. | |
| findUserId()  Parameters  String email  Return Type  User | * Call the findUserId() by passing email to get corresponding user. | |
| validateAnswer()  Parameters  List<userSecretQuestion>  String ans1  String ans2  String ans3  Return Type  boolean | * Call the validAnswer() of UserService by passing list of secret questions and answers to check whether answers typed by the user correct or not to get user id. | |
| updateUser()  Parameters  User user  Return Type  boolean | * Call the updateUser() of UserService by passing user to update the user. | |

# Database Design

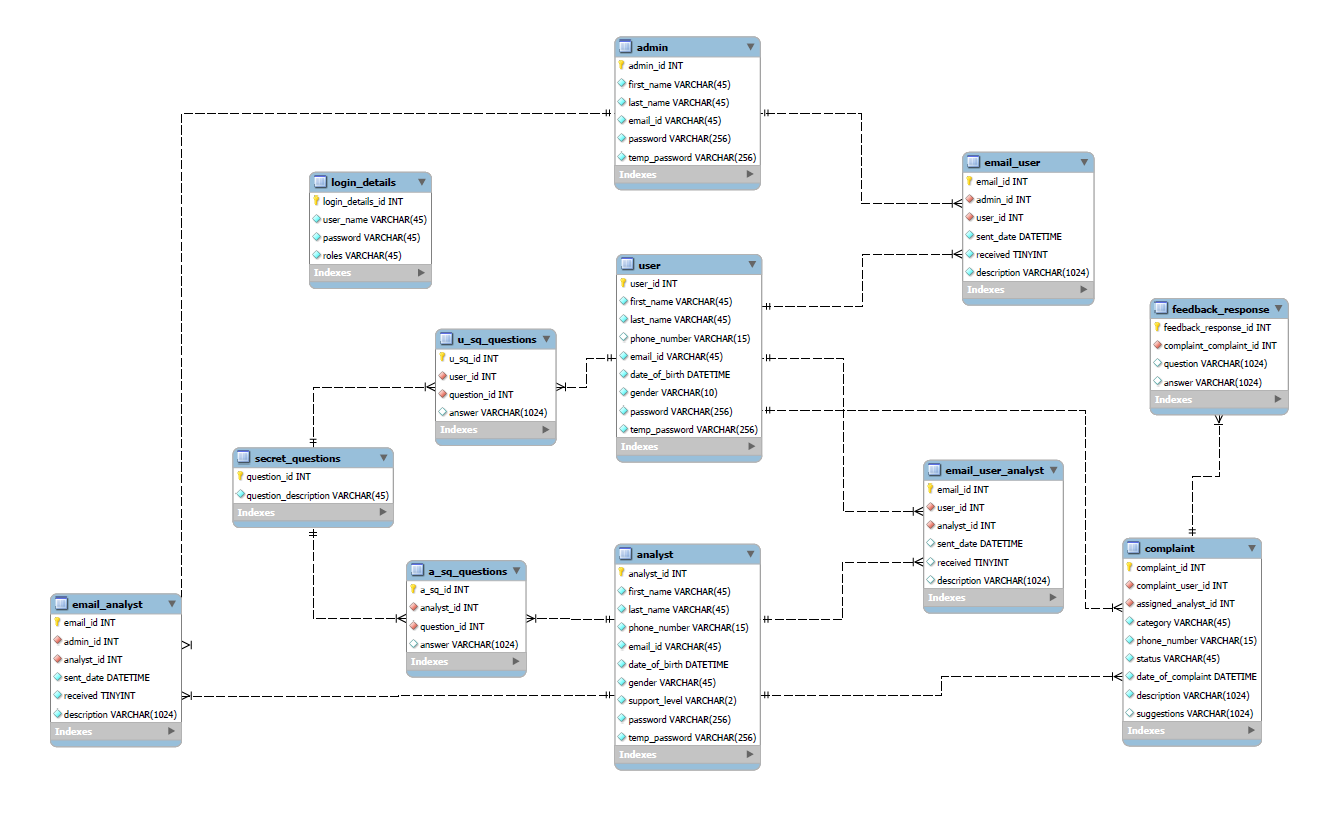
## Table Structure



### User Table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Table Name | Column Name | Column Description | Data Type | Data Size | Primary Key | Other Constraint |
| USER | USER\_ID | Column is used for storing User Id | INT |  | YES | NOT NULL  AUTO\_INCREMENT |
| FIRST\_NAME | Column for storing First Name | VARCHAR | 45 |  | NOT NULL |
| LAST\_NAME | Column for storing Last Name of User | VARCHAR | 45 |  | NOT NULL |
| PHONE\_NUMBER | Column for storing Phone Number of User | VARCHAR | 15 |  |  |
| EMAIL\_ID | Column for storing Email ID of User | VARCHAR | 45 |  | NOT NULL |
| DATE\_OF\_BIRTH | Column for storing Date of birth of User | DATETIME |  |  | NOT NULL |
| GENDER | Column for storing gender of User | VARCHAR | 10 |  | NOT NULL |
| PASSWORD | Column for storing password of User | VARCHAR | 256 |  | NOT NULL |
| TEMP\_PASSWORD | Column for storing temporary password | VARCHAR | 256 |  | NOT NULL |
| ADMIN | ADMIN\_ID | Column is used for storing Admin Id | INT |  | YES | NOT NULL  AUTO\_INCREMENT |
| FIRST\_NAME | Column for storing first name of admin | VARCHAR | 45 |  | NOT NULL |
| LAST\_NAME | Column for storing last name of admin | VARCHAR | 45 |  | NOT NULL |
| EMAIL\_ID | Column for storing Email ID of Admin | VARCHAR | 45 |  | NOT NULL |
| PASSWORD | Column for storing password of Admin | VARCHAR | 256 |  | NOT NULL |
| TEMP\_PASSWORD | Column for storing temporary password | VARCHAR | 256 |  | NOT NULL |
| ANALYST | ANALYST\_ID | Column is used for storing Analyst id. | INT |  | YES | NOT NULL AUTO\_INCREMENT |
| FIRST\_NAME | Column for storing the first name of Analyst. | VARCHAR | 45 |  | NOT NULL |
| LAST\_NAME | Column is used for storing last name of analyst | VARCHAR | 45 |  | NOT NULL |
| PHONE\_NUMBER | Column is used for Phone Number of Analyst | VARCHAR | 15 |  | NOT NULL |
| EMAIL | Column is used for storing Email ID | VARCHAR | 45 |  | NOT NULL |
| DATE\_OF\_BIRTH | Column is used for storing date of birth of Analyst | DATETIME |  |  | NOT NULL |
| GENDER | Column is used for storing gender of Analyst | VARCHAR | 45 |  | NOT NULL |
| SUPPORT\_LEVEL | Column is used for storing support level of Analyst | VARCHAR | 45 |  | NOT NULL |
| PASSWORD | Column is used for storing password of Analyst | VARCHAR | 256 |  | NOT NULL |
| TEMP\_PASSWORD | Column is used for storing temporary password of Analyst | VARCHAR | 256 |  | NOT NULL |
| COMPLAINT | COMPLAINT\_ID | Column is used for storing Complaint id. | INT |  | YES | NOT NULL AUTO\_INCREMENT |
| USER\_ID | Column is used for storing User id | INT |  |  | FOREIGN KEY  (USER.USER\_ID),  NOT NULL |
| ASSIGNED\_ANALYST\_ID | Column is used for storing Analyst id | INT |  |  | FOREIGN KEY  (ANALYST.ANALYST\_ID),  NOT NULL |
| CATEGORY | Column is used for storing Category of Complaint | VARCHAR | 45 |  | NOT NULL |
| PHONE\_NUMBER | Column is used for storing Phone Number | VARCHAR | 15 |  | NOT NULL |
| STATUS | Column is used for storing Status of Complaint | VARCHAR | 45 |  | NOT NULL |
| DATE\_OF\_COMPLAINT | Column is used for storing date of Complaint | DATETIME |  |  | NOT NULL |
| DESCRIPTION | Column is used for storing description of Complaint | VARCHAR | 1024 |  | NOT NULL |
| SUGGESTIONS | Column is used for storing Suggestions | VARCHAR | 1024 |  |  |
| SECRET\_QUESTIONS | QUESTION\_ID | Column is used for storing Question Id | INT |  | YES | AUTO\_INCREMENT  NOT NULL |
| QUESTION\_DESCRIPTION | Column is used for storing Question description | VARCHAR | 45 |  | NOT NULL |
| U\_SQ\_QUESTIONS | U\_SQ\_ID | Column is used for storing User secret Question id | INT |  | YES | NOT NULL |
| USER\_ID | Column is used for storing User id | INT |  |  | NOT NULL  FOREIGN KEY  (USER.USER\_ID) |
| QUESTION\_ID | Column is used for storing Question id | INT |  |  | NOT NULL  FOREIGN KEY  (SECRET\_QUESTION.QUESTION\_ID) |
| ANSWER | Column is used for storing Answer | VARCHAR | 1024 |  |  |
| EMAIL\_ANALYST | EMAIL\_ID | Column is used for storing Email id | INT |  | YES | NOT NULL  AUTO\_INCREMENT |
| ADMIN\_ID | Column is used for storing Admin id | INT |  |  | NOT NULL  FOREIGN KEY  (ADMIN.ADMIN\_ID) |
| ANALYST\_ID | Column is used for storing Analyst id | INT |  |  | NOT NULL  FOREIGN KEY  (ANALYST.ANALYST\_ID) |
| SENT\_DATE | Column is used for storing Sent date | DATETIME |  |  | NOT NULL |
| RECIEVED | Column is used for storing Received | TINYINT |  |  | NOT NULL |
| DESCRIPTION | Column is used for storing Description of complaint | VARCHAR | 1024 |  | NOT NULL |
| EMAIL\_USER | EMAIL\_ID | Column is used for storing Email id | INT |  | YES | NOT NULL  AUTO\_INCREMENT |
| ADMIN\_ID | Column is used for storing Admin id | INT |  |  | NOT NULL  FOREIGN KEY  (ADMIN.ADMIN\_ID) |
| USER\_ID | Column is used for storing User id | INT |  |  | NOT NULL  FOREIGN KEY  (USER.USER\_ID) |
| SENT\_DATE | Column is used for storing Sent date | DATETIME |  |  | NOT NULL |
| RECIEVED | Column is used for storing Received | TINYINT |  |  | NOT NULL |
| DESCRIPTION | Column is used for storing Description of complaint | VARCHAR | 1024 |  | NOT NULL |
| FEEDBACK\_RESPONSE | FEEDBACK\_RESPONSE\_ID | Column is used for storing Feedback response id | INT |  | YES | NOT NULL  AUTO\_INCREMENT |
| COMPLAINT\_COMPLAINT\_ID | Column is used for storing Complaint id | INT |  |  | NOT NULL  FOREIGN KEY  (COMPLAINT.COMPLAINT\_COMPLAINT\_ID) |
| QUESTION | Column is used for storing Questions | VARCHAR | 1024 |  |  |
| ANSWER | Column is used for storing Answers | VARCHAR | 1024 |  |  |
| LOGIN\_DETAILS | LOGIN\_DETAILS\_ID | Column is used for storing Login details id | INT |  | YES | NOT NULL  AUTO\_INCREMENT |
| USER\_NAME | Column is used for storing User name | VARCHAR | 45 |  | NOT NULL |
| PASSWORD | Column is used for storing  Password | VARCHAR | 45 |  | NOT NULL |
| ROLES | Column is used for storing  Roles | VARCHAR | 45 |  | NOT NULL |
| A\_SQ\_QUESTIONS | A\_SQ\_ID | Column is used for storing Analyst secret question id | INT |  | YES | NOT NULL  AUTO\_INCREMENT |
| ANALYST\_ID | Column is used for storing  Analyst id | INT |  |  | NOT NULL  FOREIGN KEY  (ANALYST.ANALYST\_ID) |
| QUESTION\_ID | Column is used for storing  Question id | INT |  |  | NOT NULL  FOREIGN KEY  (SECRET\_QUESTION.QUESTION\_ID) |
| ANSWER | Column is used for storing  Analyst secret question Answer | VARCHAR | 1024 |  |  |
| EMAIL\_USER\_ANALYST | EMAIL\_ID | Column is used for storing email id. | INT |  | YES | NOT NULL  AUTO\_INCREMENT |
| USER\_ID | Column is used for storing user id. | INT |  |  | NOT NULL  FOREIGN KEY  (USER.USER\_ID) |
| ANALYST\_ID | Column is used for storing analyst id. | INT |  |  | NOT NULL  FOREIGN KEY  (ANALYST.ANALYST\_ID) |
| SENT\_DATE | Column is used for storing sent date. | DATETIME |  |  |  |
| RECIEVED | Column is used for storing received. | TINYINT |  |  | NOT NULL |
| DESCRIPTION | Column is used for storing description. | VARCHAR | 1024 |  |  |

## E- R Diagram (Relationship between Tables)



## Standards and Guidelines Followed

### JAVA

1. Ensure that the class names, method names and variable names are followed exactly as specified in the class diagram
2. Naming standards to be followed:
   1. Variable
      1. loweCamelCase
      2. Short variable
   2. Class
      1. Is a noun
   3. Methods
      1. Is a Verb

# Change Log

*Please note that this table needs to be maintained even if a Configuration Management tool is used.*

|  |  |  |
| --- | --- | --- |
| **Version Number** |  | **Changes made** |
| V1.0 | *<First version>* |  |