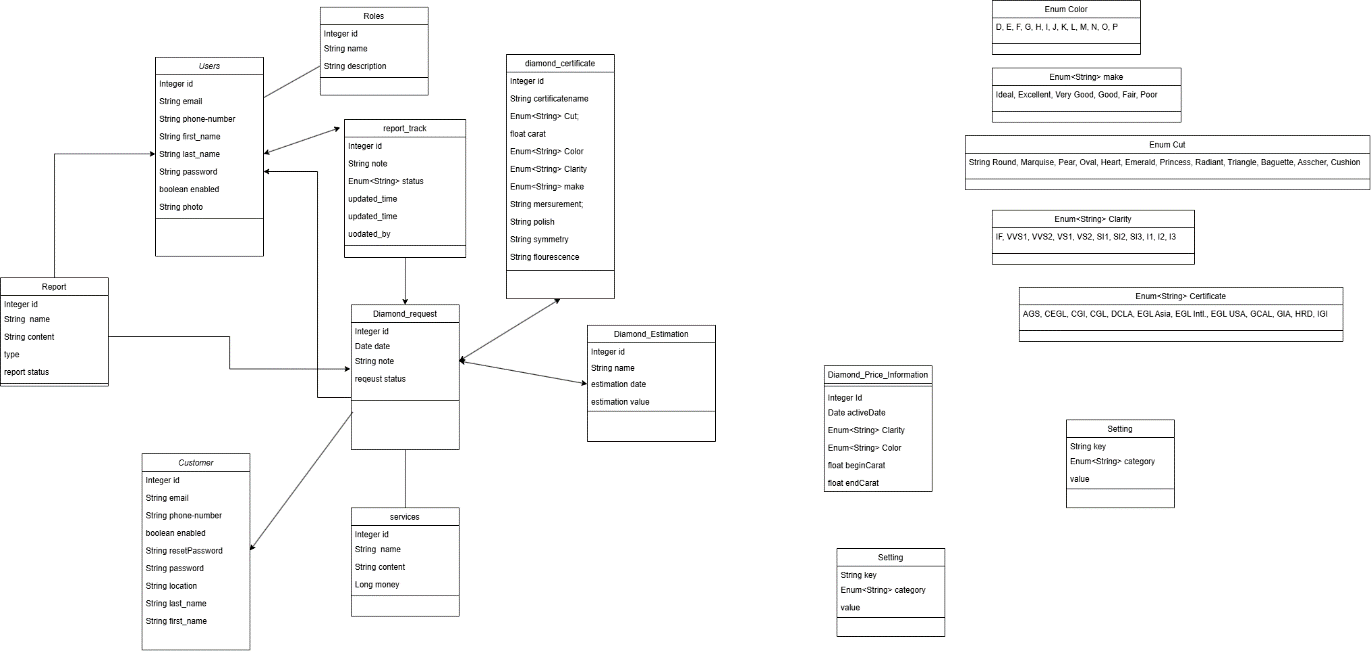
**ERD/DB Diagram:**



**Function requirements:**

1. User Login/Logout
2. Dashboard and Reporting
3. Manage Users
4. Manage Customers
5. Manage Services
6. Manage Request Services
7. Manage Certificates
8. View Home Page
9. Customer Login/Logout
10. Diamond Valuation Estimation
11. Valuation Request Management
12. Handling Unclaimed

**Usecase list:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | User Login/Logout | Login | Admin/Manager/Staff/Valuation Staff | User logs into the system using their credentials. |
| 2 | Logout | User logs out of the system. |
| 3 | Dashboard and Reporting | View Dashboard and Statistics | Admin/Manager | Views the dashboard for statistical analysis and reporting of valuation requests and other key metrics. |
| 4 | Manage User | View User | Admin/Manager/Staff/Valuation Staff | Views details of user accounts. |
| 5 | Create User | Admin | Admin creates a new user account. |
| 6 | Edit User | Admin edits details of an existing user account. |
| 7 | Delete User | Admin deletes an existing user account. |
| 8 | Manage Customer | View Customers | Admin/Manager/Staff/Valuation Staff | Views details of customer profiles. |
| 9 | Create Customer | Admin/Staff/Manager | Creates a new customer profile. |
| 10 | Edit Customer | Edits details of an existing customer profile. |
| 11 | Delete Customer | Deletes an existing customer profile |
| 12 | Manage Services | View Services | Admin/Manager/Staff/Valuation Staff | Views details of existing services. |
| 13 | Create Service | Admin | Admin creates a new valuation service. |
| 14 | Edit Service | Admin edits details of an existing service. |
| 15 | Delete Service | Admin deletes an existing service. |
| 16 | Manage Request Services | View Valuation Request | Admin/Manager/Staff/Valuation Staff | User views details of valuation requests. |
| 17 | Submit Valuation Request | Staff/Customer | User submits a valuation request for a diamond. |
| 18 | Process Valuation Request | Staff/Valuation Staff | User processes the valuation request and assigns it for valuation. |
| 19 | Update Valuation Request Status | User updates the status of the valuation request as it progresses through the valuation process. |
| 20 | Approve Sealing Report | Manager | Manager approves the sealing report for unclaimed valuation results. |
| 21 | Approve Commitment Form | Manager approves the commitment form for lost receipts. |
| 22 | Manage Diamond Certificate | View Diamond Certificate | Admin/Manager/Staff/Valuation Staff | User views details of diamond certificates. |
| 23 | Create Diamond Certificate | Valuation Staff | Valuator creates a diamond certificate based on the valuation results. |
| 24 | Edit Diamond Certificate | Valuator edits details of an existing diamond certificate. |
| 25 | Print Diamond Certificate | Prints the diamond certificate for the customer |
| 26 | Delete Diamond Certificate | Deletes an existing diamond certificate |
| 27 | View Home Page | View Home Page | Customer | Customer views the home page, which introduces the company, services offered, diamond knowledge resources, search and valuation functionalities, and blog. |
| 28 | Login/Logout | Login | Customer logs into the system using their credentials. |
| 29 | Logout | Customer logs out of the system. |
| 30 | Manage Account | Edit Profile | Edits details of account profile. |
| 31 | Diamond Valuation Estimation | Estimate Diamond Value by Criteria | Customer inputs criteria such as diamond origin, shape & cut, measurements, carat weight, color, clarity, cut, proportions, polish, symmetry, and fluorescence to estimate the diamond's value. |
| 32 | Estimate Diamond Value by Certification Number | Customer inputs a certification number issued by the company to estimate the diamond's value. |
| 33 | Valuation Request Management | Submit Valuation Request | Customer submits a valuation request form. |
| 34 | View Valuation Request Status | Customer views the status of their submitted valuation request. |
| 35 | Handling Unclaimed Valuation Results | Request Commitment Form for Lost Receipt | Customer requests a commitment form to receive their diamond sample if they have lost the valuation receipt. |

**UI(Fontend):**

**Technical Requirements:**

Login User

LU1 – Successful Login

LU2 – Fail Login

Create User

# CU3 - Input Validation for Email Field

# CU4 - Input Validation for Required Fields

# CU5 - Handle Image Upload and Preview

# CU6 - Submit Form with Valid Data

# CU7 - Submit Form with Invalid Email

# CU8 - Cancel Form Submission

# CU9 - Load EditUser Component

# Edit User

# EU10 - Fetch and Display User Data

# EU11 - Input Validation for Email Field

# EU12 - Input Validation for Required Fields

# EU13 - Handle Image Upload and Preview

# EU14 - Submit Form with Valid Data

# EU15 - Submit Form with Invalid Email

# EU16 - Submit Form with Duplicate Email

# EU17 - Cancel Form Submission

# Delete User

# DU18 - Open Delete Confirmation Dialog

# DU19 - Cancel Delete Action

# DU20 - Confirm Delete Action

Create Customer

CC22 - Successful Form Submission

CC23 - Form Submission with Invalid Email

CC24 - Form Submission with Missing Required Fields

CC25 - Form Submission with Invalid Phone Number

CC26 - Cancel Form Submission

Edit Customer

EC27 - Render Form with Customer Data

EC28 - Successful Form Submission

EC29 - Cancel Form Submissions

EC30 - Password Field Behavior

Delete Customer

DC31 - Open Delete Confirmation Dialog

DC32 - Cancel Deletion

DC33 - Confirm Deletion

TEST CASE ADD DIAMOND REQUEST

**1. Form Validation**

**Description:** Validate form behavior when submitting with invalid or missing data.

**Test Steps:**

* Attempt to submit the form with invalid or empty fields:
  + Customer not selected.
  + Status not selected.
  + No service selected.

**Expected Result:**

* Validation errors should appear next to respective fields:
  + Error message "Customer is required" should appear if customer is not selected.
  + Error message "Status is required" should appear if status is not selected.
  + Error message "At least one service must be selected" should appear if no service is selected.
* Form should not submit until all required fields are filled correctly.

**2. Form Submission**

**Description:** Test the form submission process.

**Test Steps:**

* Fill all required fields correctly (customer, note, status, and select at least one service).
* Click on the "Save" button.

**Expected Result:**

* Form should submit without validation errors.
* The saveRequest function should be called with correct form values.
* Upon successful submission:
  + A success message should briefly appear ("Add new Request successfully").
  + The user should be navigated to the "/requests" page.

**3. User Interaction**

**Description:** Test user interactions such as selecting a customer, selecting services, and changing status.

**Test Steps:**

* Select a customer from the autocomplete dropdown.
* Select one or more services using checkboxes.
* Change the status using the dropdown.

**Expected Result:**

* Customer ID should be correctly set when a customer is selected.
* Services should be correctly added/removed from service\_ids array based on checkbox selection.
* Status should update as per user selection in the dropdown.

**5. Cancel Button Functionality**

**Description:** Test the functionality of the "Cancel" button.

**Test Steps:**

* Click on the "Cancel" button.

**Expected Result:**

* The user should be navigated to the "/requests" page.

EDIT DIAMOND REQUETS

### **Test Case 1: Form Submission with Valid Data**

Description: Test the submission of the form with valid input data.

Test Steps:

1. Populate all required fields (customer\_id, note, status, and select at least one service).
2. Submit the form by clicking on the "Save Changes" button.

Expected Result:

* Form submission succeeds.
* saveRequest function is called with the correct values.
* Successful navigation to "/requests".
* Success message (Alert severity="success") is displayed.

### **Test Case 2: Form Submission with Invalid Data**

Description: Test the form submission with missing required fields.

Test Steps:

1. Leave one or more required fields (customer\_id, status, or services) empty.
2. Submit the form by clicking on the "Save Changes" button.

Expected Result:

* Form submission fails.
* Validation errors are displayed for each missing required field.
* Error message (Alert severity="error") is displayed indicating the issue.

### **Test Case 3: Editing an Existing Request**

Description: Test the pre-population of the form when editing an existing request.

Test Steps:

1. Mock an existing requestid to simulate editing.
2. Ensure that the form fields (customer\_id, note, status, service\_ids) are correctly pre-populated with the existing request data.

Expected Result:

* Form fields are populated with data fetched from getRequestById.
* The correct customer is selected in the autocomplete based on requestEdit.customer\_phone.
* Services checkboxes reflect the existing service\_ids selected.

### **Test Case 4: Cancel Button Functionality**

Description: Test the "Cancel" button functionality.

Test Steps:

1. Click on the "Cancel" button.
2. Verify that navigation occurs to "/requests" route.

Expected Result:

* Clicking the "Cancel" button navigates the user back to the "/requests" route without making any changes.

DELETE DIAMOND REQUEST

### **Test Case 1: Open Delete Confirmation Dialog**

Description: Test that verifies the delete confirmation dialog opens correctly when clicking the delete button.

Test Steps:

1. Navigate to the Requests component.
2. Locate a request entry in the table.
3. Click on the delete icon/button corresponding to the request.

Expected Result: The delete confirmation dialog should open with the message "Confirm Delete". The request details (such as customer name or ID) should be displayed in the dialog content.

### **Test Case 2: Cancel Deletion**

Description: Test that ensures canceling the deletion does not remove the request.

Test Steps:

1. Navigate to the Requests component.
2. Open the delete confirmation dialog for a request.
3. Click on the "Cancel" button in the confirmation dialog box.

Expected Result: The delete confirmation dialog should close, and the request should remain visible in the table without any changes. No success or error message should be displayed.

### **Test Case 3: Confirm Deletion**

Description: Test that verifies successful deletion of a request when confirming deletion in the dialog.

Test Steps:

1. Navigate to the Requests component.
2. Open the delete confirmation dialog for a request.
3. Click on the "Delete" button in the confirmation dialog box.

Expected Result: The request should be deleted from the UI. A success message should appear briefly, indicating "Delete request with id {request\_id} successfully!". The deleted request should no longer be visible in the table.

### **Additional Considerations:**

* UI Validation: Verify that the delete confirmation dialog displays the correct request details (such as ID, customer name) before confirming deletion.
* Edge Cases: Test with various scenarios, such as deleting the first or last request, deleting multiple requests consecutively, etc.
* Error Handling: Ensure the UI handles errors gracefully, such as network issues or server errors during deletion