## Agriculture Loan Portal

**Overview:**

This application is built to help customers to apply Agriculture loans online.

## Users of the System:

1. Admin
2. Customer

## Functional Requirements:

**Customer**

* Build an application that customers can apply for Agriculture loans online.
* The customers should fill their profile.
* The customers should have an option to upload the document proofs such as Aadhaar card, pan card, pay slips, bank statements etc.
* The customers can apply for Agriculture loans through the portal.
* Once the loan application is filled. The customers can submit for further approval process.
* Customer can provide reviews.

## Admin

* The admin can see list the list of applied loans.
* The admin can view the list of documents submitted and either approve/disapprove the loan application.
* Once the loan is sanctioned, the repayment schedule should get generated automatically.
* The admin should have provision to generate reports.

While the above ones are the basic functional features expected, the below ones can be nice to have add-on features:

* Filters for like Low to High or showcasing buses based on the customer’s price range, specific Company etc.
* Email integration for intimating new personalized offers to customers.
* Multi-factor authentication for the sign-in process
* Payment Gateway

## Output/ Post Condition:

* Records Persisted in Success & Failure Collections
* Standalone application / Deployed in an app Container

## Non-Functional Requirements:

|  |  |
| --- | --- |
| **Security** | * App Platform – Username/Password-Based Credentials * Sensitive data has to be categorized and stored in a secure manner * Secure connection for transmission of any data |
| **Performance** | * Peak Load Performance (during Festival days, National holidays etc.) |

|  |  |
| --- | --- |
|  | * Admin application < 2 Sec * Non-Peak Load Performance * Appointment Application< 2 Sec * Admin Application < 2 Sec |
| **Availability** | * 99.99 % Availability |
| **Standard Features** | * Scalability * Maintainability * Usability * Availability * Failover |
| **Logging &**  **Auditing** | * The system should support logging(app/web/DB) & auditing at   all levels |
| **Monitoring** | * Should be able to monitor via as-is enterprise monitoring tools |
| **Cloud** | * The Solution should be made Cloud-ready and should have a   minimum impact when moving away to Cloud infrastructure |
| **Browser Compatible** | * All latest browsers |

**Technology Stack**

|  |  |
| --- | --- |
| Front End | Angular 10+ |
| Server Side | Spring Boot |
| Database | MySQL or Oracle or MSSQL |

## Platform Prerequisites (Do’s and Don’ts):

1. The react app should run in port 8081.
2. Spring boot app should run in port 8080.

## 

## Key points to remember:

1. The id (for frontend) and attributes(backend) mentioned in the SRS should not be modified at any cost. Failing to do may fail test cases.
2. Remember to check the screenshots provided with the SRS. Strictly adhere to id mapping and attribute mapping. Failing to do may fail test cases.
3. Strictly adhere to the proper project scaffolding (Folder structure), coding conventions, method definitions and return types.
4. Adhere strictly to the endpoints given below.

## This is a basic SRS document, so understand them well and please feel free to explore and come with new ideas.

**Application assumptions:**

1. The login page should be the first page rendered when the application loads.
2. Manual routing should be restricted by using Auth Guard by implementing the

Can Activate interface. For example, if the user enters as http://localhost:8080/signup or http://localhost:8080/home the page should not navigate to the corresponding page instead it should redirect to the login page.

1. Unless logged into the system, the user cannot navigate to any other pages.
2. Logging out must again redirect to the login page.
3. To navigate to the admin side, you can store a user type as admin in the database with a username and password as admin.
4. Use admin/admin as the username and password to navigate to the admin dashboard.

## Validations:

1. Basic email validation should be performed.
2. Basic mobile validation should be performed.

## Project Tasks: API Endpoints: Admin Side:

|  |  |  |  |
| --- | --- | --- | --- |
| Action | URL | Method | Response |
| Admin Login | /admin/login | POST-Sends email ID and password | Return True/False |
| Admin SignUp | /admin/signup | POST-Sends Admin Model data | Admin added |
| Admin Edit loan | /admin/editLoan/{id} | POST-Sends updated loan information | Edit loan information |
| Admin view loan | /admin/getAllLoans | GET - fetches all loans | View loans |
| Delete loan | /admin/deleteLoan/{id  } | DELETE – posts the  loan id | Loan deleted |
| Generate Repayment  Schedule | /admin/generateSched ule | GET – get the repayment schedule  for the candidates | Get repayment schedule |
| Edit Repayment Schedule | /admin/editRepayment Schedule/{loanId} | POST – Edit the repayment Schedule | Repayment schedule edited |
| Delete Repayment Schedule | /admin/deleteRepaym entSchedule/{loanId} | POST – Delete repayment schedule | Payment Schedule deleted |

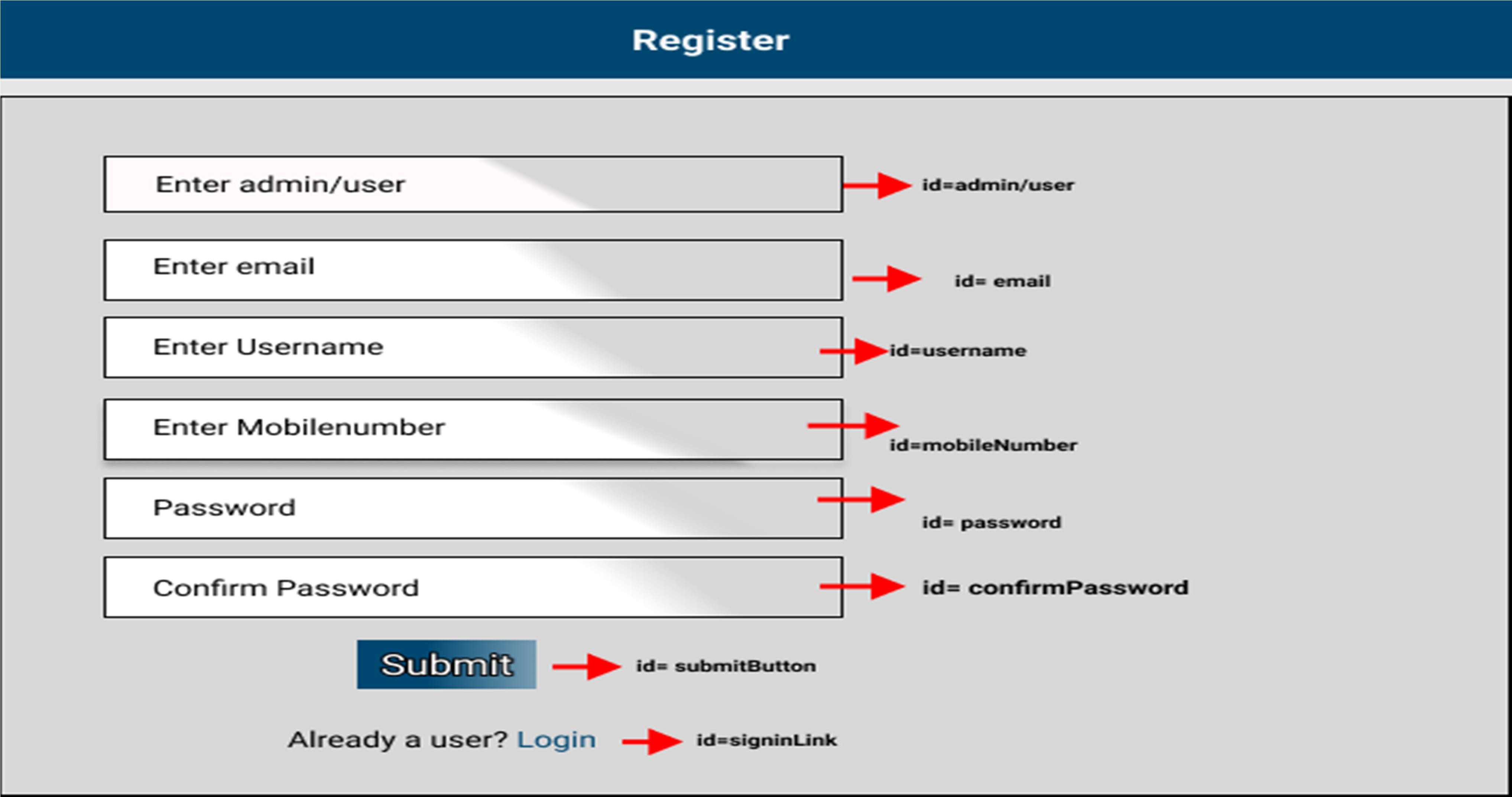
**User Side:**

|  |  |  |  |
| --- | --- | --- | --- |
| Action | URL | Method | Response |
| User Login | /user/login | POST-Sends email ID and password | Return True/False |
| Admin SignUp | /user/signup | POST-Sends User Model data | User added |
| Add Profile | /user/addProfile | POST – Sends user data | User profile created |
| View Profile | /user/getProfile | GET – recieves user data | View Profile |
| Edit Profile | /user/editProfile/{userId} | POST – posts updated user data | Updated profile |
| Delete Profile | /user/deleteProfile/{userId} | POST – posts delete user data | Delete profile |
| Add documents | /user/addDocuments | POST – Sends document data | Document uploaded |
| View Documents | /user/getDocuments | GET – view document data | View documents |
| Edit Documents | /user/editDocuments/{docu mentId} | POST – posts updated document | Edit documents |
| Delete Documents | /user/deleteDocuments/{doc umentId} | POST – sends document id | Delete documents |
| Apply new loan | /user/addLoan | POST – post Loan data | Loan added |
| View applied loan | /user/viewLoan | GET – Get applied loan information | View Applied loan |
| Edit loan | /user/editLoan/{loanId} | PUT – Update Loan information | Edit Applied loan |
| Delete loan | /user/deleteLoan/{loanId} | DELETE – delete  the loan application | Delete loan application |

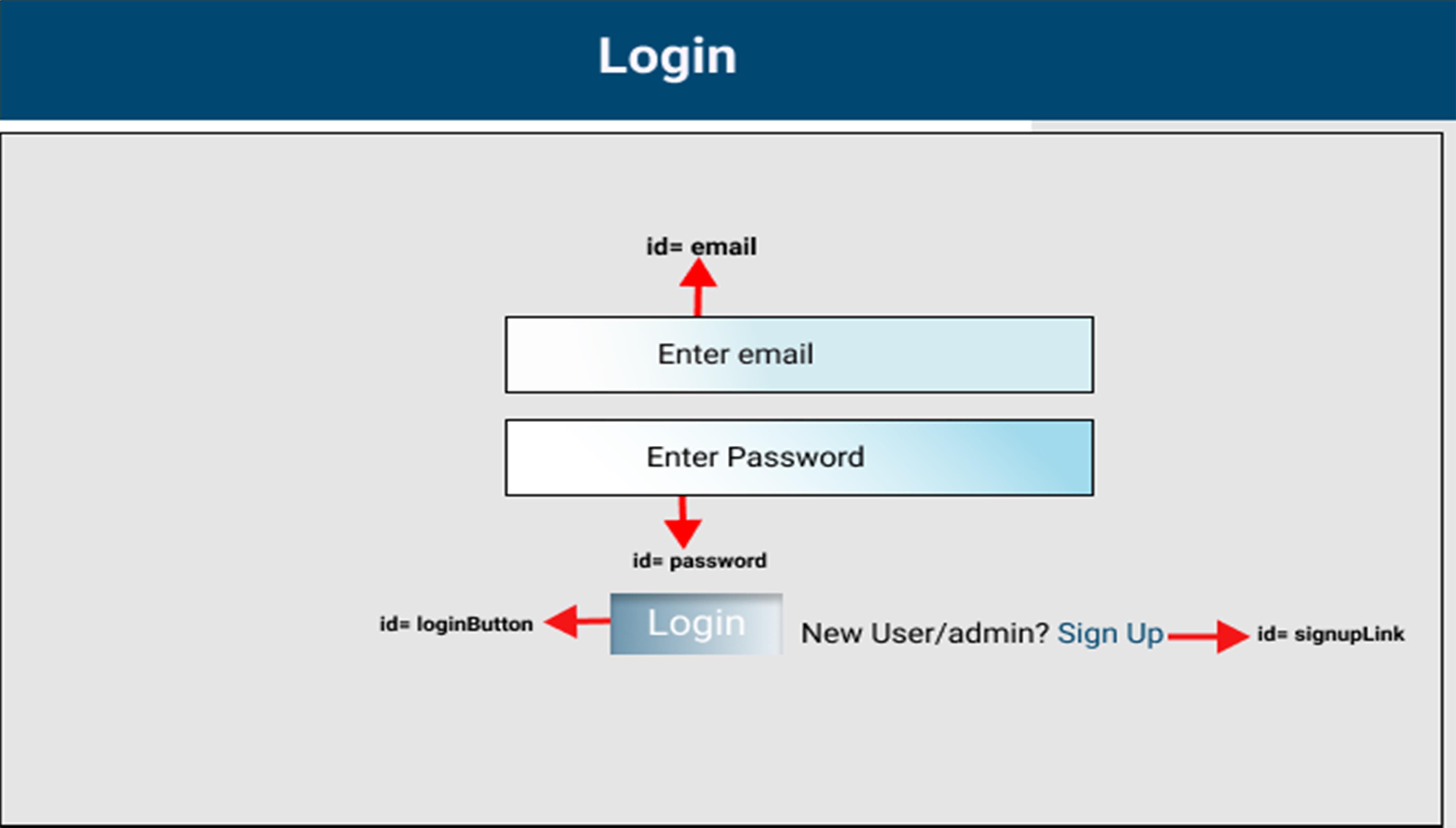
# Frontend:

## Customer:

1. Auth: Design an auth component where the customer can authenticate login and signup credentials
2. Signup: Design a signup page component inside the auth where the new customer has options to sign up by providing their basic details.
   1. Ids: Refer to the screenshot below for the id details.
   2. Your frontend should use the ids provided.
   3. Routing URL: http://localhost:8080/user/signup
   4. Output screenshot:

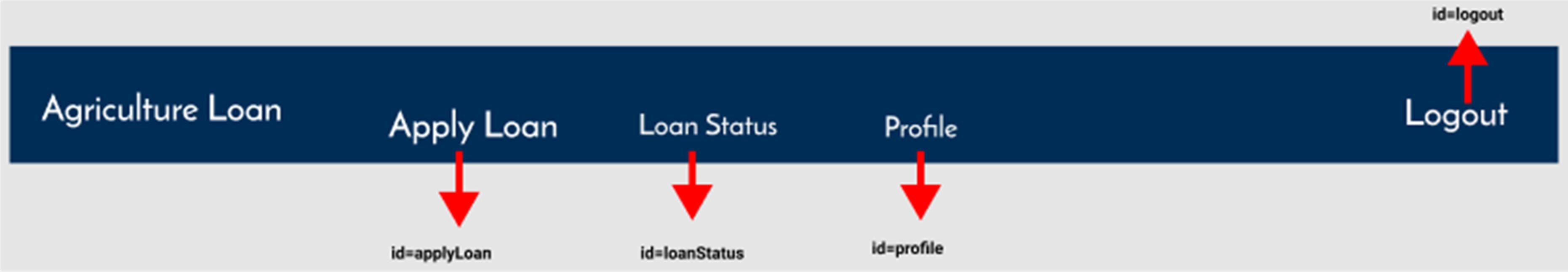


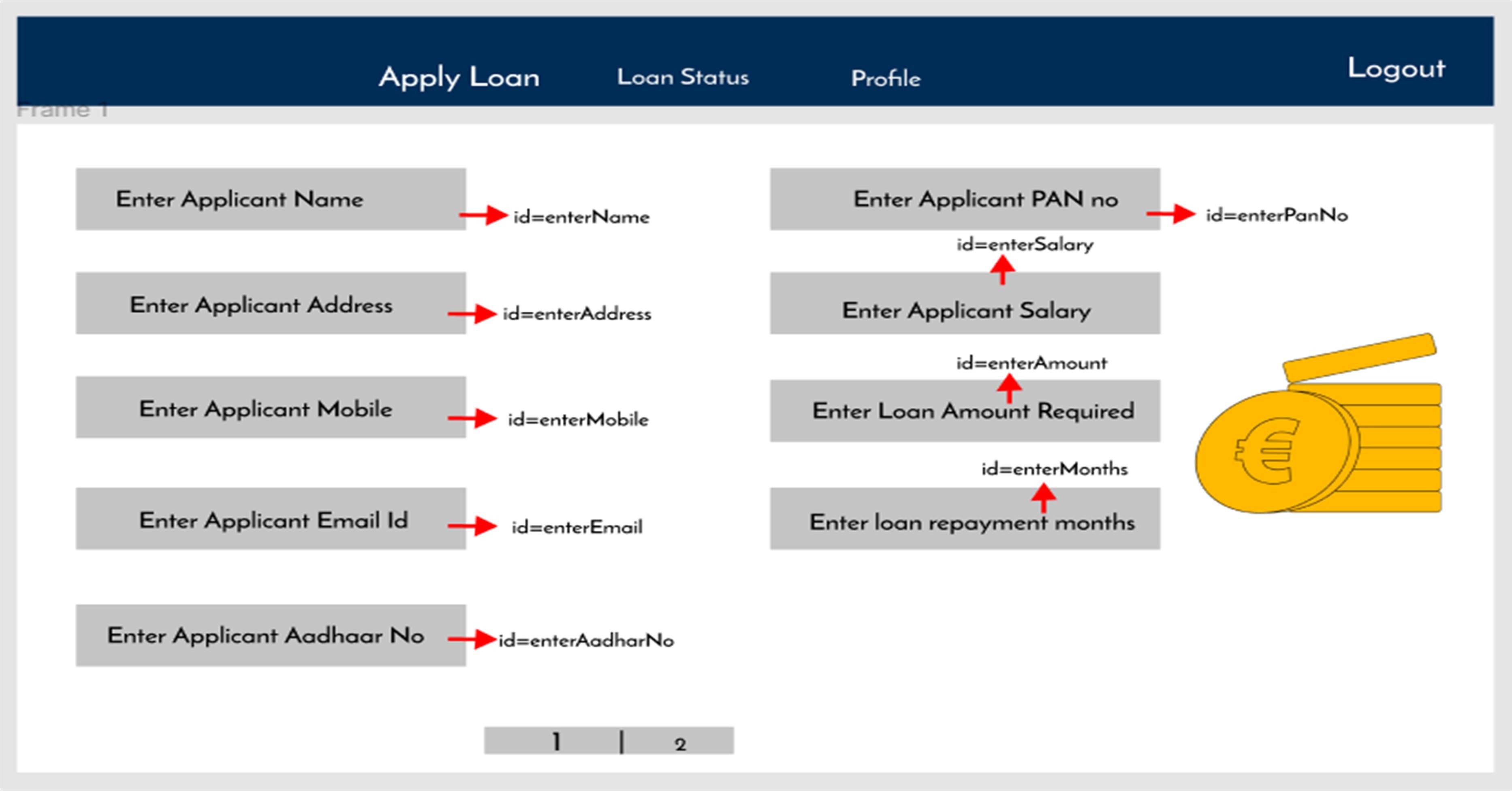
1. Login: Design a login page component inside the auth where the existing customer can log in using the registered email id and password.
   1. Ids: Refer to the screenshot below for the id details.
   2. Routing URL: http://localhost:8080/user/login
   3. Output Screenshot:

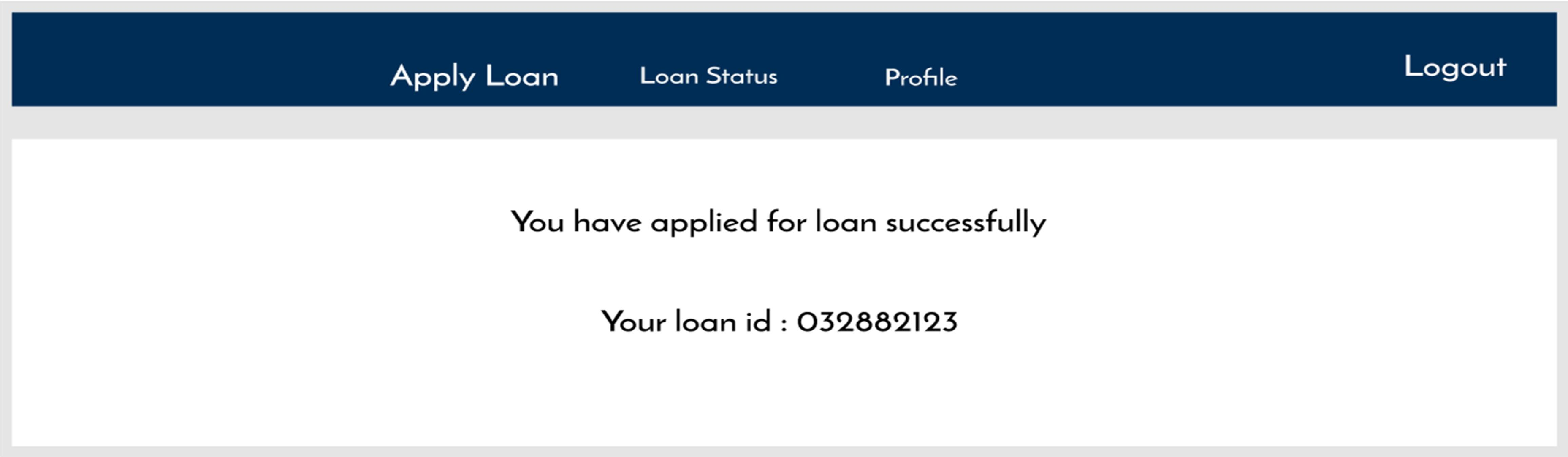
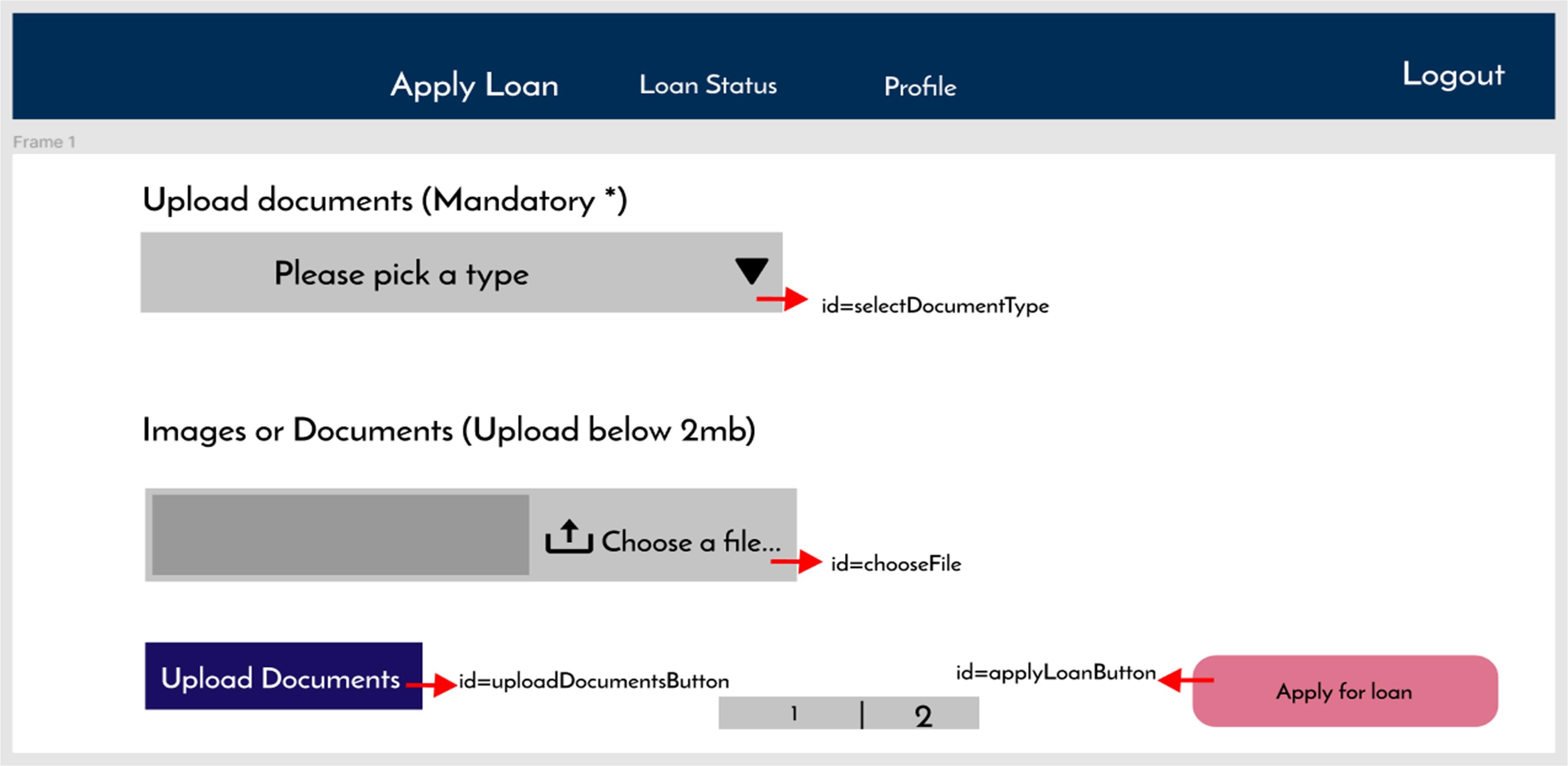


## Customer side:

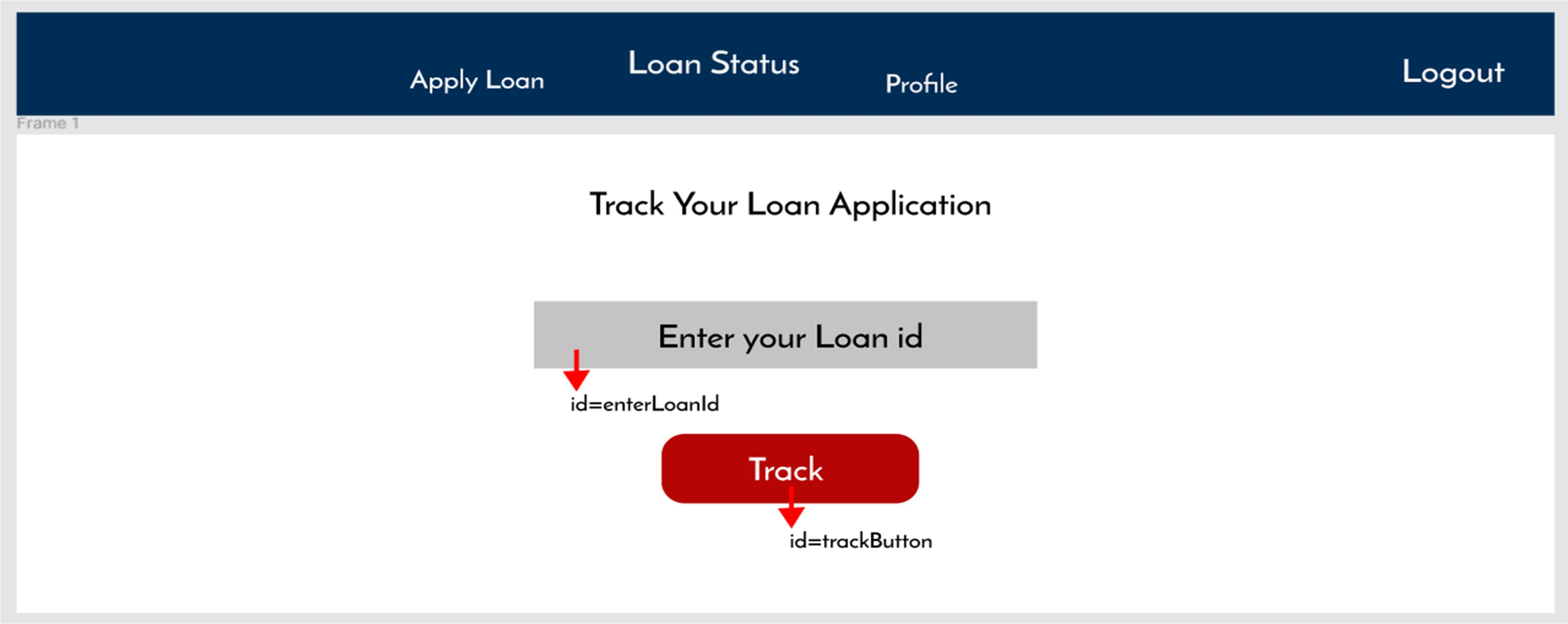
1. Apply loan: Design an customer apply loan component
   1. Ids: Refer to the screenshot below for the id details.
   2. Routing URL: http://localhost:8080/user/addLoan
   3. Output screenshot:

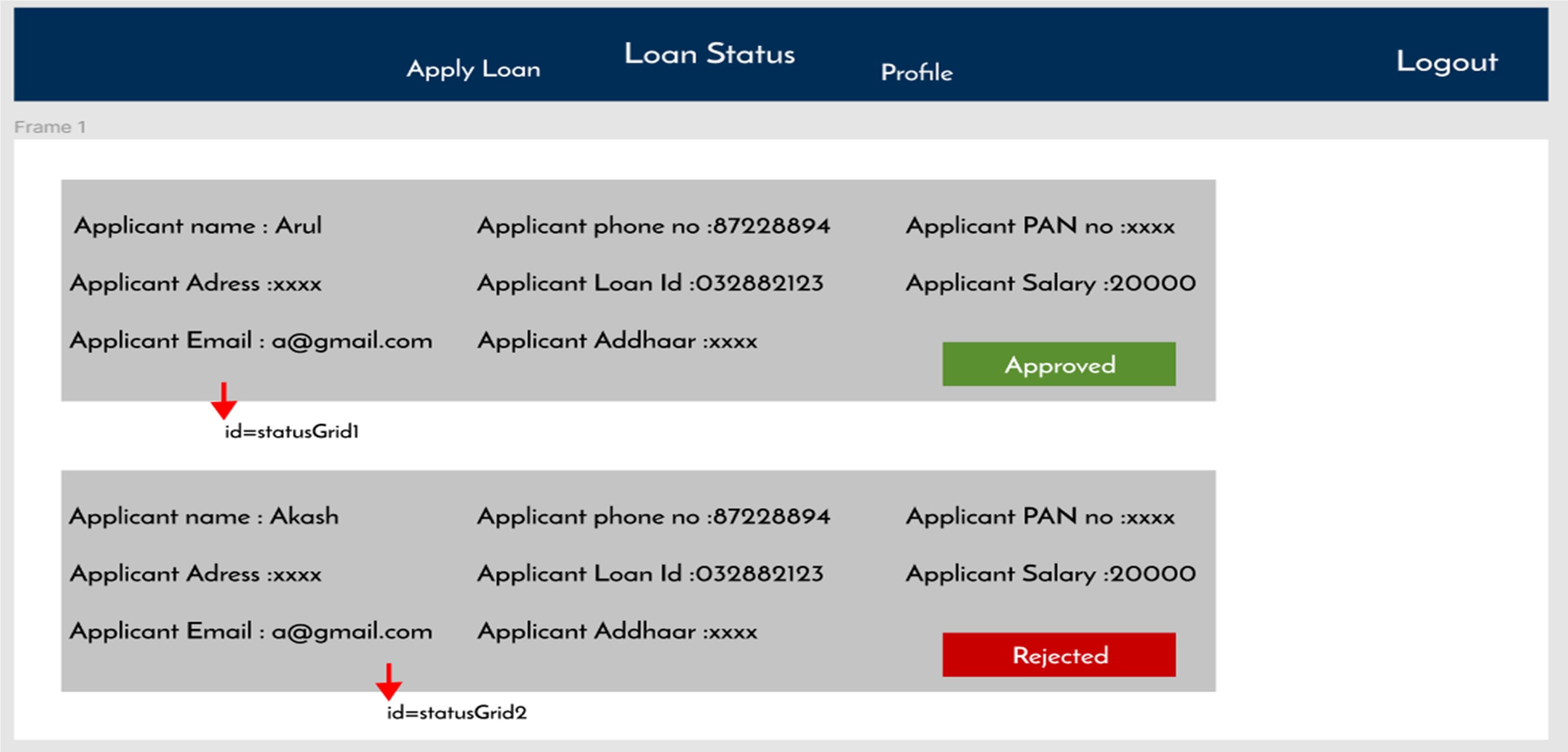




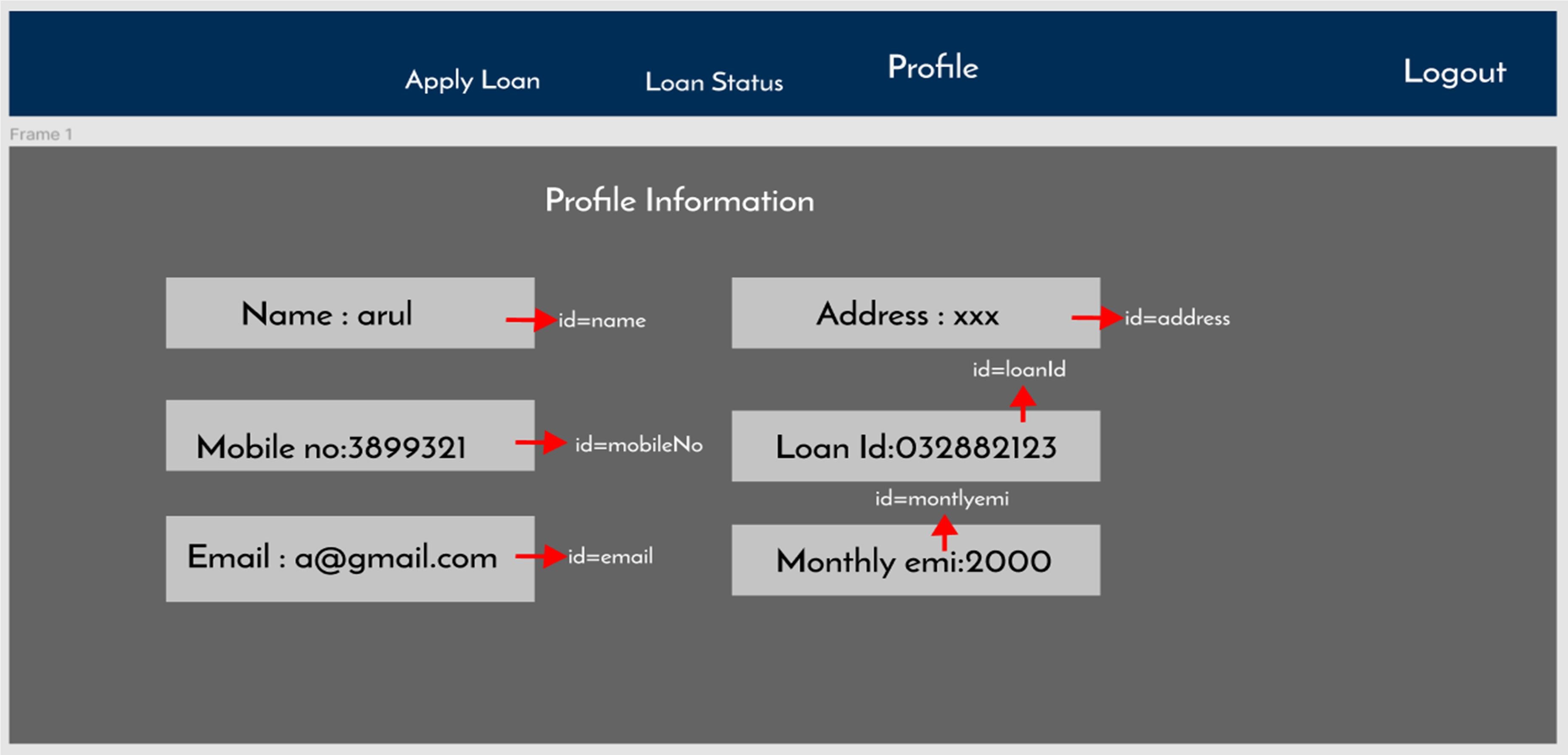


1. Loan status: Design an customer loan status page.The Page contains loan status apply new loan
   1. Ids: Refer to the screenshot below for the id details.
   2. Routing URL: http://localhost:8080/user/viewLoan
   3. Output screenshot:



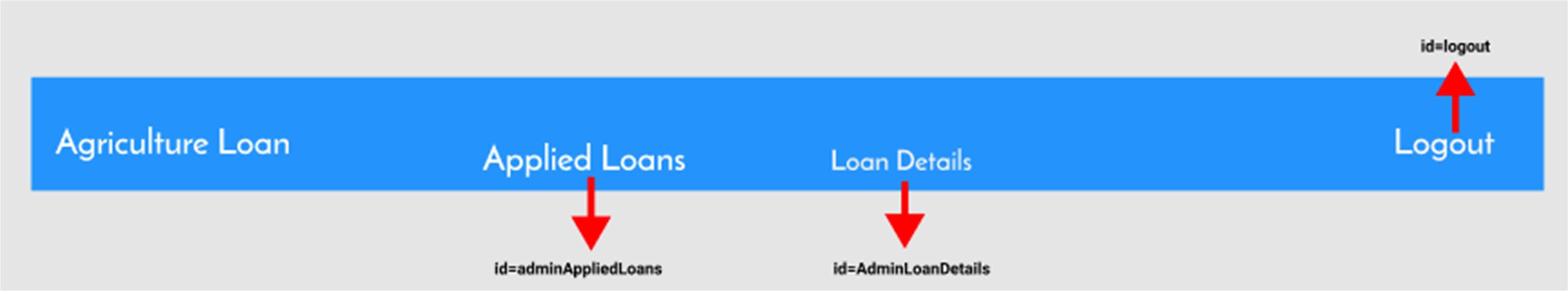


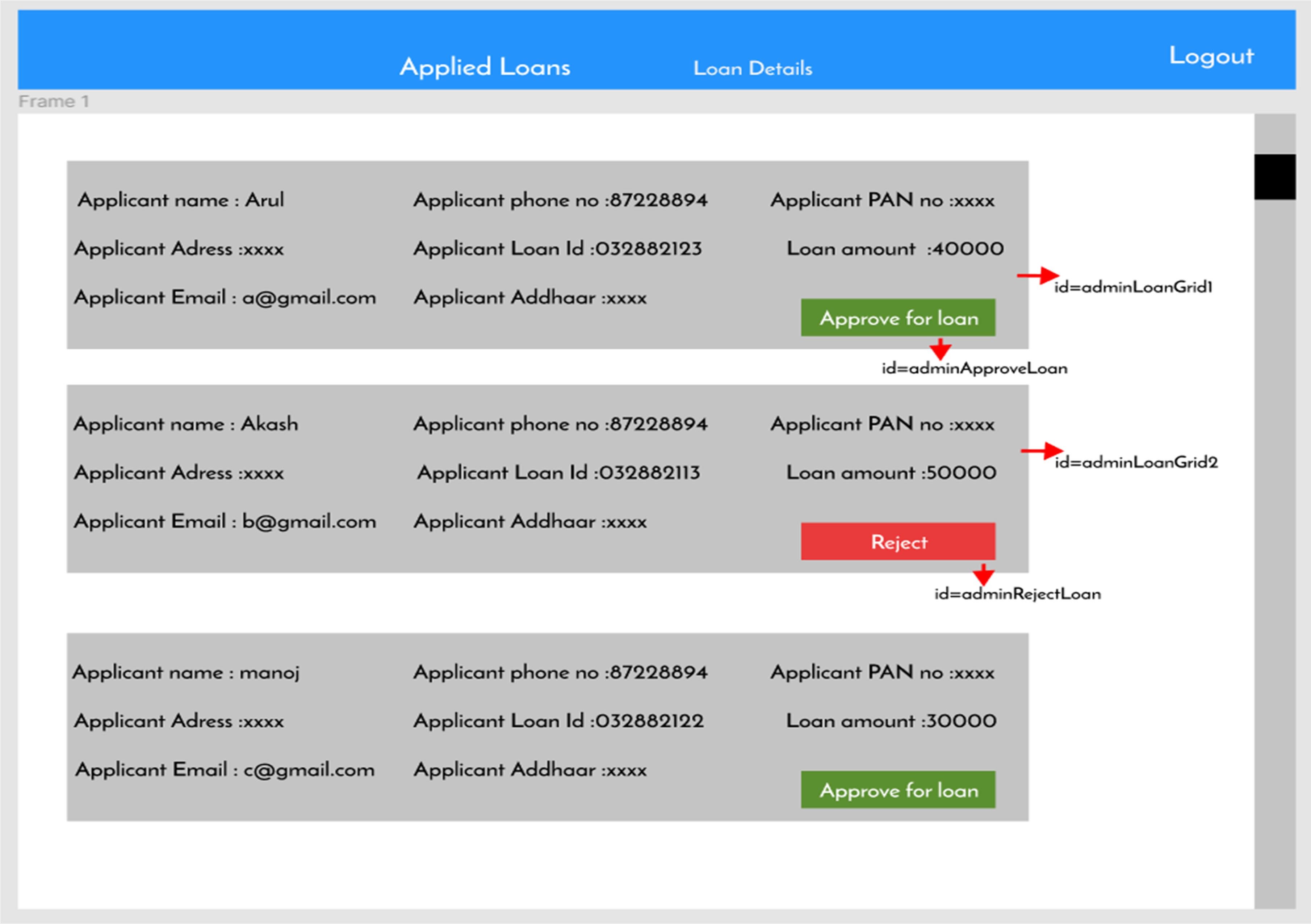
1. profile: Design an customer profile component
   1. Ids: Refer to the screenshot below for the id details.
   2. Routing URL: http://localhost:8080/user/getProfile
   3. Output screenshot:



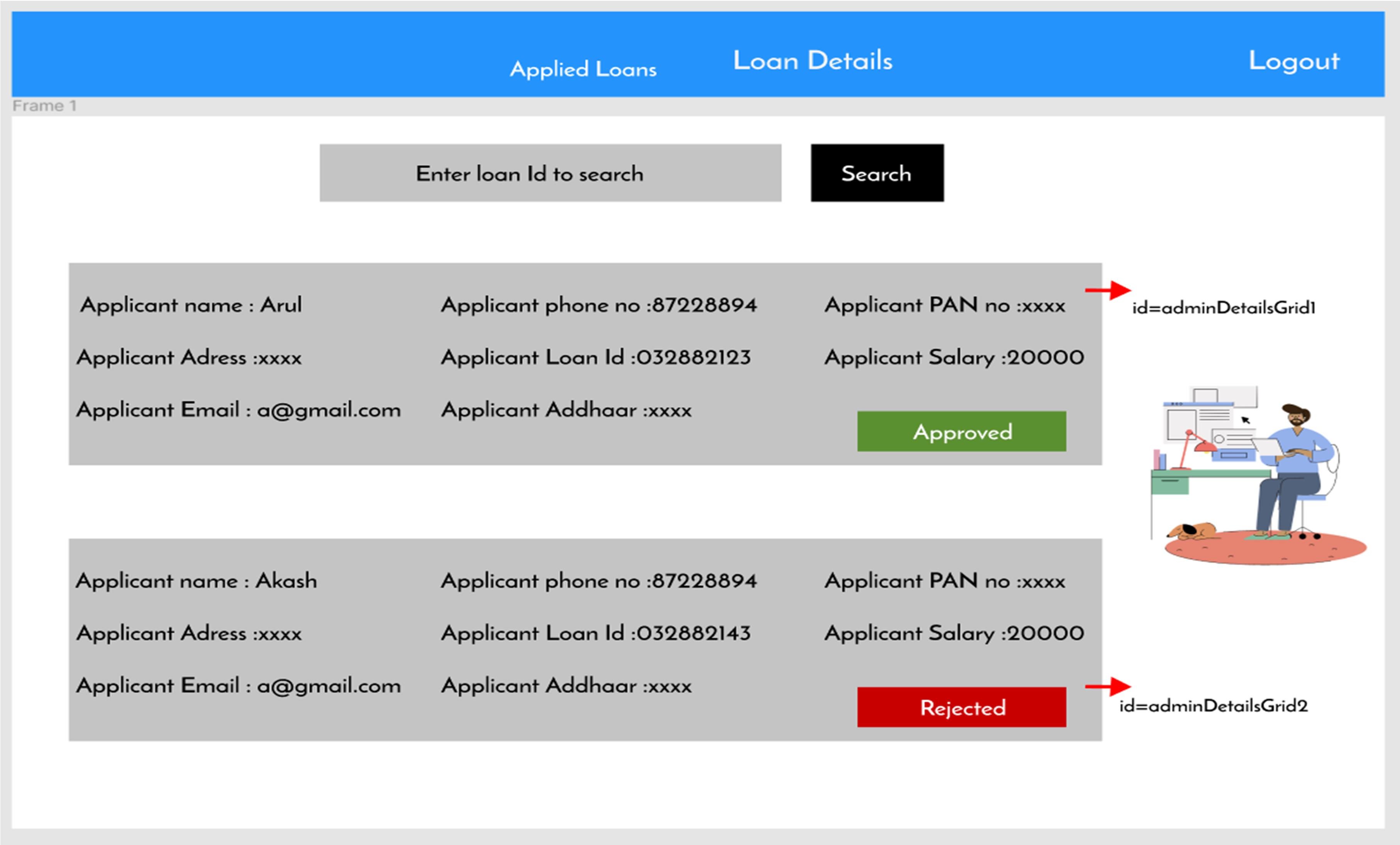
## Admin side:

1. Applied loans: Design an admin applied loan component
   1. Ids: Refer to the screenshot below for the id details.
   2. Routing URL: http://localhost:8080/admin/getAllLoans
   3. Output screenshot:





1. Approved loans: Design an admin approved loan component
   1. Ids: Refer to the screenshot below for the id details.
   2. Routing URL: http://localhost:8080/admin/getAllLoans
   3. Output screenshot:



v

# Backend:

## Class and Method description:

**Model Layer:**

1. **UserModel**: This class stores the user type (admin or the customer) and all user information.
   1. Attributes:
      1. email: String
      2. password: String
      3. username: String
      4. mobileNumber: String
      5. userRole: String
2. **LoginModel**: This class contains the email and password of the user.
   1. Attributes:
      1. email: String
      2. password: String
3. **AdminModel**: This class stores the details of the admin
   1. Attributes:
      1. email:String
      2. password:String
      3. mobileNumber:String
      4. userRole:String
4. **LoanApplicantModel:** This class stores the job details.
   1. Attributes:
      1. loanId: Int
      2. loantype: String
      3. applicantName: String
      4. applicantAddress: String
      5. applicantMobile: String
      6. applicantEmail: String
      7. applicantAadhaar: String
      8. applicantPan: String
      9. applicantSalary: String
      10. loanAmountRequired: String
      11. loanRepaymentMonths: String
5. **DocumentModel:** This class store the uploaded documents as image or pdf.
   1. Attributes:
      1. documentId: Int
      2. documenttype: String
      3. documentupload:Byte[]

## Controller Layer:

1. **AuthController**: This class control the user /admin signup and signin
   1. Methods:
      1. isUserPresent(LoginModel data): This method helps to check whether the user present or not and check the email and password are correctand return the boolean value.
      2. isAdminPresent(LoginModel data): This method helps to check whether the admin present or not and check the email and passwordare correct and return the boolean value.
      3. saveUser(UserModel user): This method helps to save the user datain the database.
      4. saveAdmin(UserModel user): This method helps to save the admindata in the database.
2. **UserController**: This class helps to add/delete/view/update User Details
   1. Methods:
      1. addUser(UserModel data): This method adds user to the application.
      2. getUser(String UserID): This method is used to get the list of users.
      3. editUser(String UserID). This method helps to edit the users.
      4. deleteUser(String UserID): This method helps to delete the users
3. **loanController:** This class helps to add/delete/view/update loan.
   1. Methods:
      1. addLoan(LoanModel Data): This method applies for new loan.
      2. editLoan(int LoanId): This method edits the applied loan.
      3. getLoan(LoanModel Data): This method is used to get the list of loans applied
      4. deleteLoan(int loanId): This method deletes the loan application.
4. **AdminController:** This class helps to add/delete/view/update profile.
   1. Methods:
      1. approveLoan(LoanModel data) - The admin can verify the loan application
      2. verifyDocuments(DocumentModel data) - The admin can verify the documents before approving the loan
      3. editLoan(int loanId) - Edit the loans
      4. deleteLoan(int loanId) - delete the loans
      5. generateSchedule(int loanId) - Generate repayment schedule based on the number of months
      6. editSchedule(int loanId) - Edit the repayment schedule vii.deleteSchedule(int loanId) - delete the repayment schedule.

**How to run the Project**

**Back End**

**API endpoint:**

8080

**Platform Guidelines:**

To run the command use **Terminal**in the platform.

**Spring Boot:**

Navigate to the springapp directory => **cd springapp**

To start/run the application '**mvn spring-boot:run**'

**To Connect Database Open Terminal**

**Cmd:mysql -u root –protocol=tcp -p**

**Password: examly  
  
Front End**

**Step 1:**

Open the terminal

Use “nvm use 14” command to change node version to 14

**Step 1:**

Use "cd reactapp" command to go inside the reactapp folder

Install Node Modules **- "**npm install**"**

**Step 2:**

Write the code inside src folder

Create the necessary components

**Step 3:**

Click the run test case button to run the test cases

**Note :**

* Click PORT 8081 to view the result / output
* If any error persists while running the app,delete the node modules and reinstall them

.