MergerWare

MergerWare technical assessment assignment

Submitted by: Somya Rathi (20BCE2323)

GitHub link - https://github.com/somyarathi/LoanBasedApplication

About meteor.js framework

Meteor is an open-source, full-stack JavaScript framework for building modern web and mobile applications. It aims to simplify and streamline the process of developing real-time, reactive applications by providing a cohesive platform for both the client and server sides. Some of its features are

- Full-stack framework
- Real-time reactive
- Single language
- Data on the wire
- Integrated build system
- Package system
- Cross-platform development
- Isomorphic JavaScript
- Data security

Problem statement:

As part of the assignment, prepare an application which does the following functions

1. any number of Users can register using email

2. Users can choose roles (admin, borrower, lender)

3. The user (borrower) can request a loan and see existing past loans (basic functionality)

4. User (lender whenever he joins and can pick a role) and confirms to pay the loan and see

the existing past payments

5. The change of status is reflected on each user's dashboard.

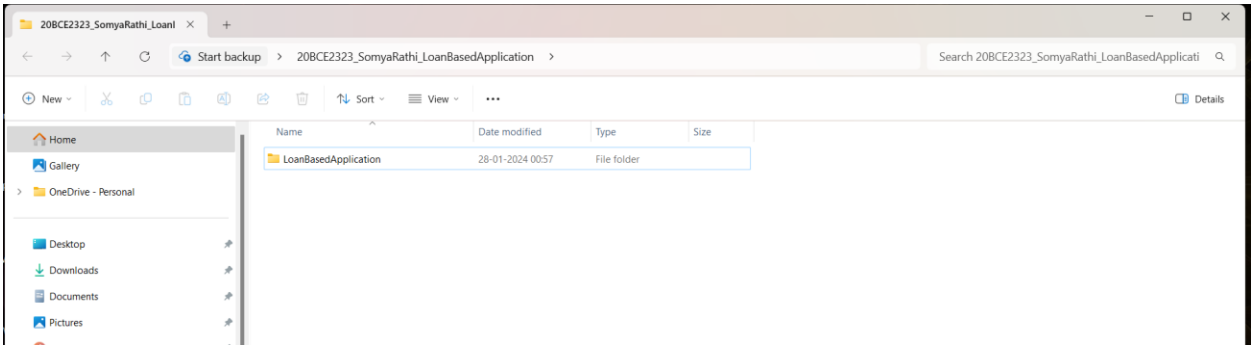
6. Admin users see the complete transactions

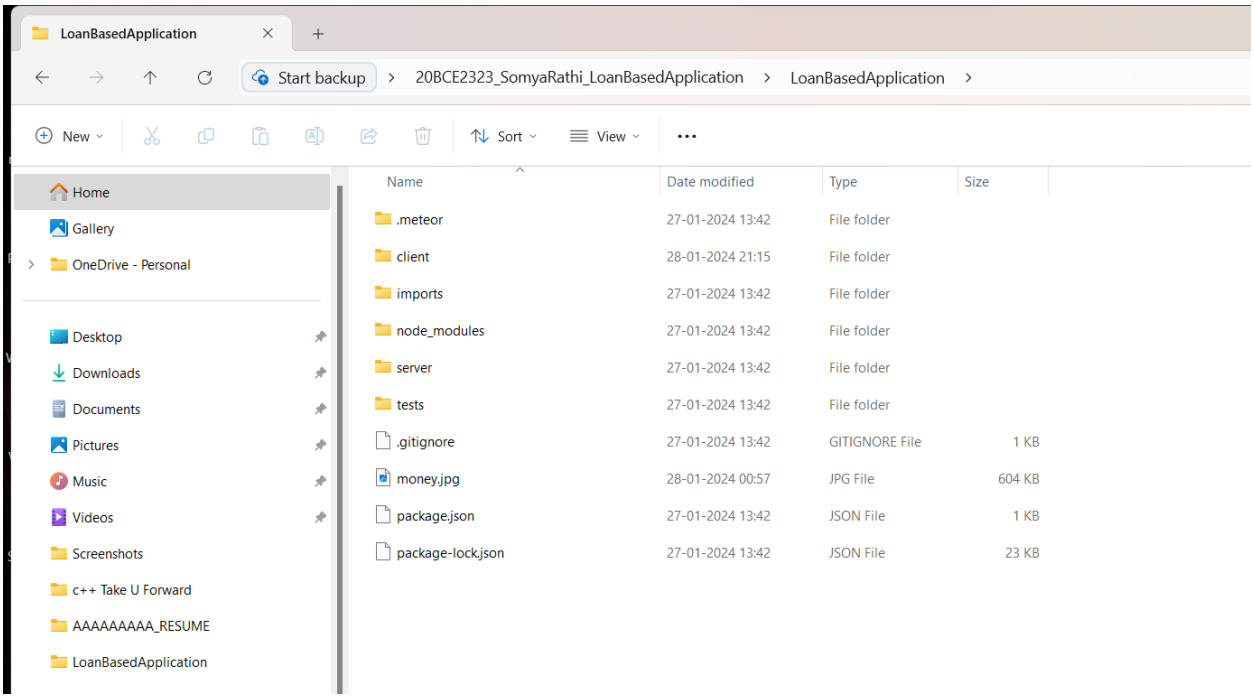
My Devised Solution:

step 1: process to create a meteor functional environment.

Note (pre-assumed that meteor and noje.js is installed in the system)

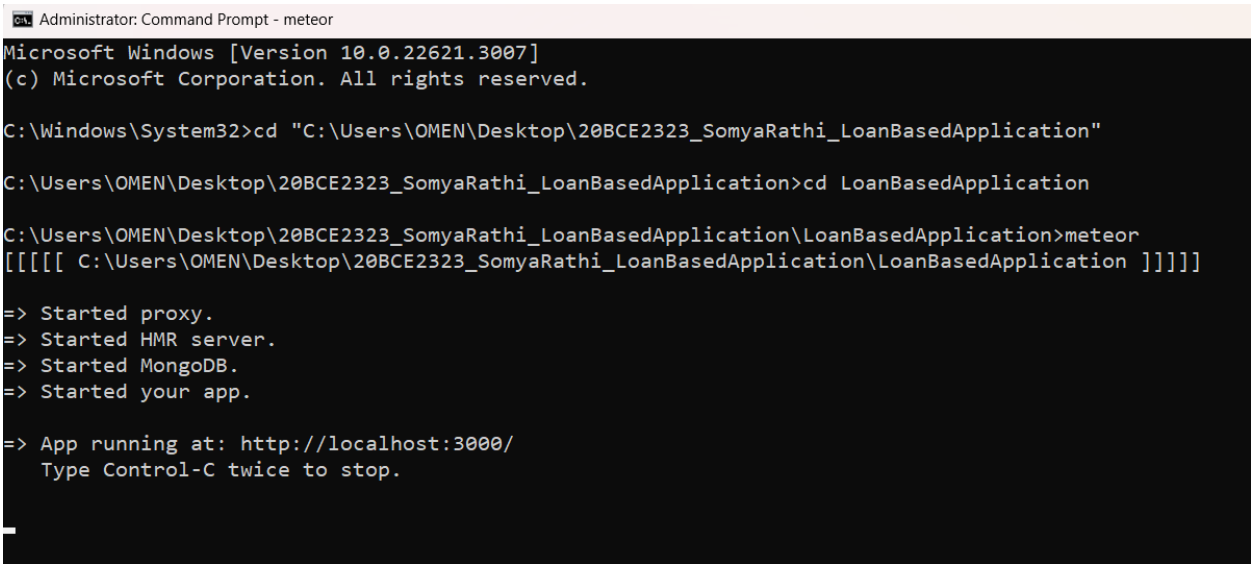
1. in command prompt create a directory that stores meteor required files
   meteor create <name-of-file>
2. change directory to that file
   cd <name=of-file>



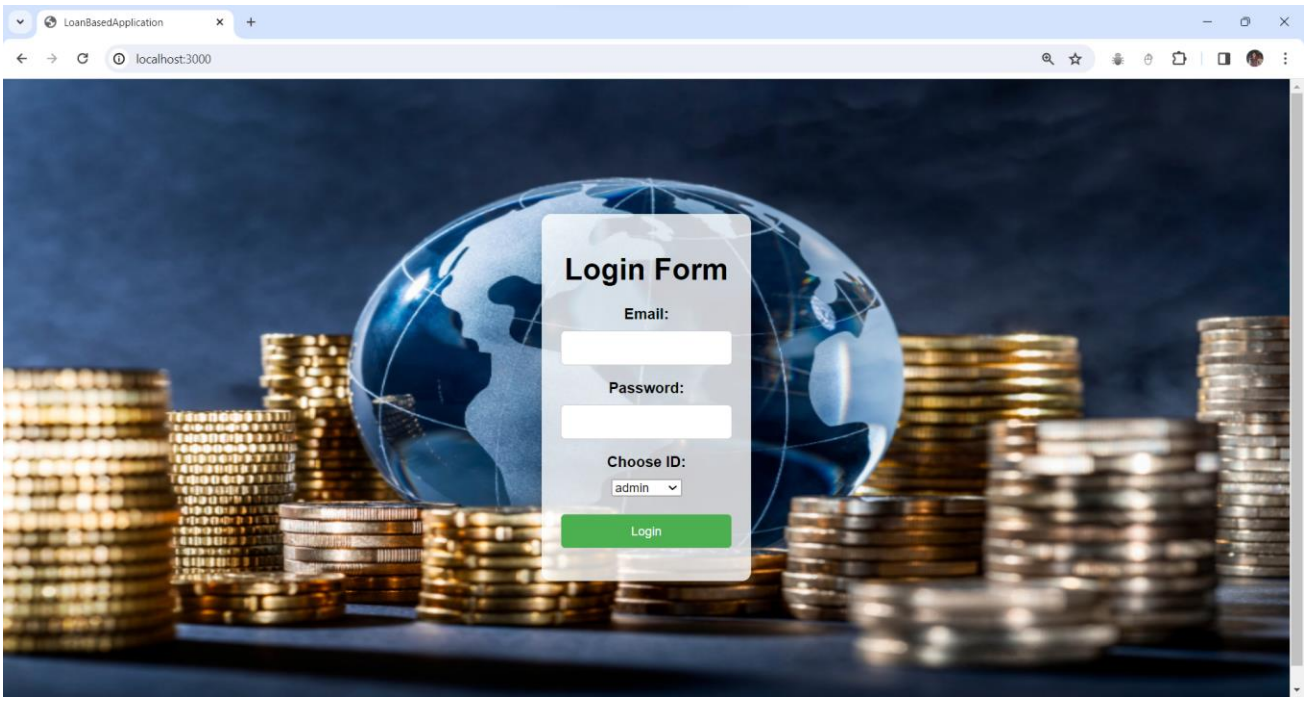


If successfully created the meteor environment for the project, it will have all these predefined file with the .meteor file at top as depicted in screenshot

3. now to run the scripted functions of frontend and backend go to command prompt and change the directory to the application folder and command meteor



This depicting that the application has been successfully launched without any error and is running on http://localhost:3000/



This ensures that the connection established is valid and the system is supporting the code.

1. Outcome for login page (client HTML script + Server-side JavaScript)

```
<head>
  <title>LoanBasedApplication</title>
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>

<body>
  <div class="container">
    <h1>Login Form</h1>
    <form id="loginForm" action="#" method="post"
onsubmit="redirectToSelectedPage(event)">
      <label for="email">Email:</label>
      <input type="email" id="email" name="email" required>

      <label for="password">Password:</label>
      <input type="password" id="password" name="password" required>

      <label for="id">Choose ID:</label>
      <select id="id" name="id" required>
```

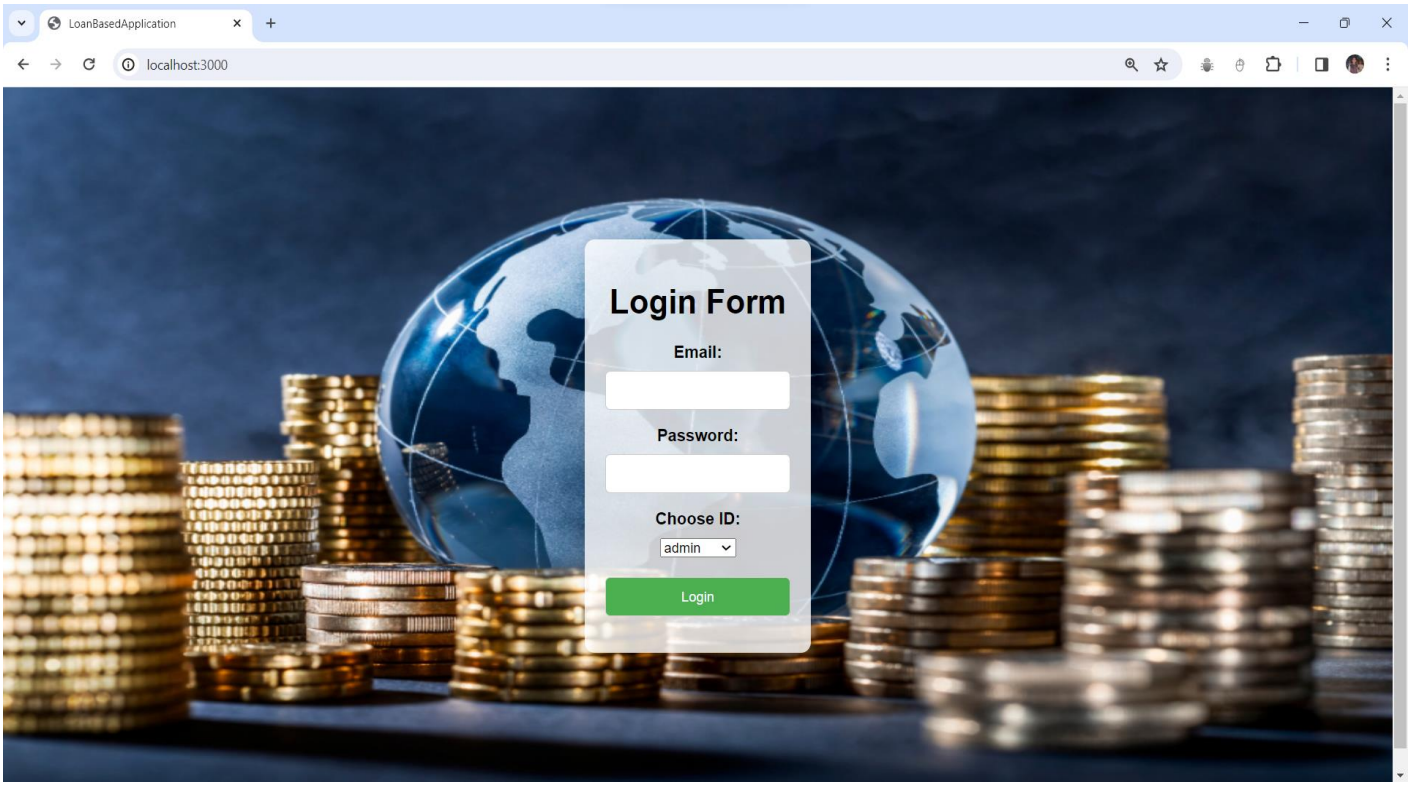
```
            <option value="admin">admin</option>
            <option value="borrower">borrower</option>
            <option value="lender">lender</option>
        </select>

        <br><br>
        <input type="submit" value="Login">
    </form>
</div>

<script>
    function redirectToSelectedPage(event) {
        event.preventDefault(); // Prevent the form from submitting
        var selectedId = document.getElementById('id').value;
        if (selectedId === 'admin') {
            window.location.href = 'admin.html';
        } else if (selectedId === 'borrower') {
            window.location.href = 'borrower.html';
        } else if (selectedId === 'lender') {
            window.location.href = 'lender.html';
        }
    }
</script>
</body>

```

Styling of this page (CSS code)

```
body {
  padding: 10px;
  font-family: sans-serif;
  background-color: white

}
body {
  display: flex;
  align-items: center;
  justify-content: center;
  height: 100vh;
  margin: 0;
  background-image: url('https://discover.rbcroyalbank.com/wp-content/uploads/Untitled-design-2023-07-31T120240.836.jpg'); /* Replace with the actual URL of your image */
  background-repeat: no-repeat;
```

```
  background-attachment: fixed;
  background-size: 100% 100%;
}

.container {
  text-align: center;
  background-color: rgba(255, 255, 255, 0.8); /* Semi-transparent white
background */
  padding: 20px;
  border-radius: 10px;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1); /* Add a subtle shadow */
}

h1 {
  margin-bottom: 20px; /* Adjust the margin as needed */
}

label {
  display: block;
  margin-bottom: 8px;
  font-weight: bold;
}

input {
  width: 100%;
  padding: 10px;
  margin-bottom: 15px;
  box-sizing: border-box;
  border: 1px solid #ccc;
  border-radius: 5px;
}

input[type="submit"] {
  background-color: #4CAF50;
  color: white;
  cursor: pointer;
}

input[type="submit"]:hover {
  background-color: #45a049;
}
```

Output:



2. Implementation of borrowing functionality (client HTML script + Server-side JavaScript + CSS script for styling)

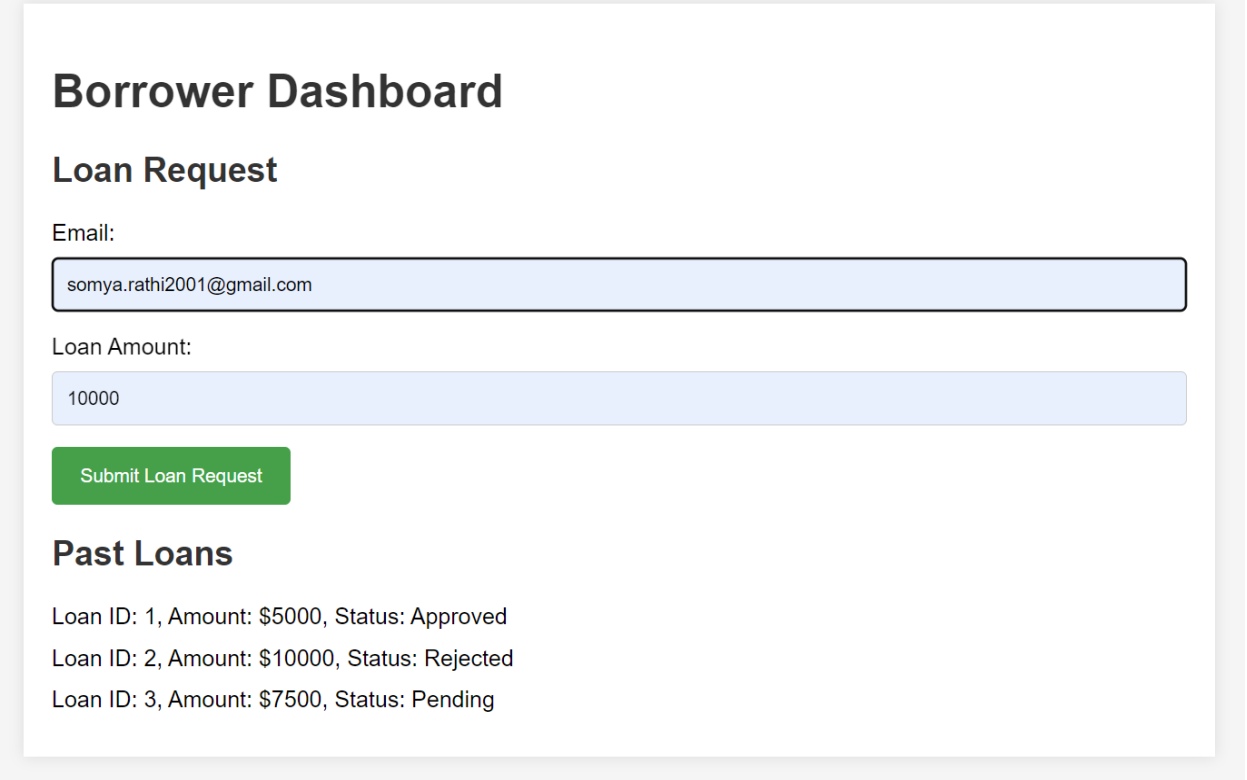
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Borrower Dashboard</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
            background-color: #f5f5f5;
        }

        .container {
            max-width: 800px;
            margin: 20px auto;
            background-color: #fff;
            padding: 20px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
```

```
}

h1, h2 {
    color: #333;
}

form {
    margin-top: 20px;
}

label {
    display: block;
    margin-bottom: 8px;
}

input[type="text"],
input[type="email"],
input[type="number"],
input[type="password"] {
    width: 100%;
    padding: 10px;
    margin-bottom: 15px;
    border: 1px solid #ccc;
    border-radius: 4px;
    box-sizing: border-box;
}

input[type="submit"] {
    background-color: #4caf50;
    color: #fff;
    padding: 12px 20px;
    border: none;
    border-radius: 4px;
    cursor: pointer;
}

input[type="submit"]:hover {
    background-color: #45a049;
}

ul {
    list-style: none;
    padding: 0;
    margin: 0;
}
```

```
        li {
            margin-bottom: 10px;
        }
    </style>
</head>
<body>

<div class="container">
    <h1>Borrower Dashboard</h1>

    <!-- Loan Request Form -->
    <h2>Loan Request</h2>
    <form id="loanRequestForm" action="#" method="post"
onsubmit="submitLoanRequest(event)">
        <label for="email">Email:</label>
        <input type="email" id="email" name="email" required>

        <label for="amount">Loan Amount:</label>
        <input type="number" id="amount" name="amount" required>

        <input type="submit" value="Submit Loan Request">
    </form>

    <!-- Past Loans Section -->
    <h2>Past Loans</h2>
    <ul id="pastLoansList"></ul>
</div>

<script>
    //for past loans
    const pastLoansData = [
        { id: 1, amount: 5000, status: 'Approved' },
        { id: 2, amount: 10000, status: 'Rejected' },
        { id: 3, amount: 7500, status: 'Pending' }
    ];

    // Display past loans on page load
    window.onload = displayPastLoans;

    function displayPastLoans() {
        const pastLoansList = document.getElementById('pastLoansList');
        pastLoansList.innerHTML = ''; // Clear previous content

        pastLoansData.forEach(loan => {
```

```
            const listItem = document.createElement('li');
            listItem.textContent = `Loan ID: ${loan.id}, Amount: $${loan.amount},
Status: ${loan.status}`;
            pastLoansList.appendChild(listItem);
        });
    }

    function submitLoanRequest(event) {
        event.preventDefault(); // Prevent the form from submitting

        // Get form values
        const email = document.getElementById('email').value;
        const amount = document.getElementById('amount').value;

        //processing loan request
        const newLoan = { id: pastLoansData.length + 1, amount: amount, status:
'Pending' };
        pastLoansData.push(newLoan);

        // Display updated past loans
        displayPastLoans();
    }
</script>

</body>
</html>
```

Output:



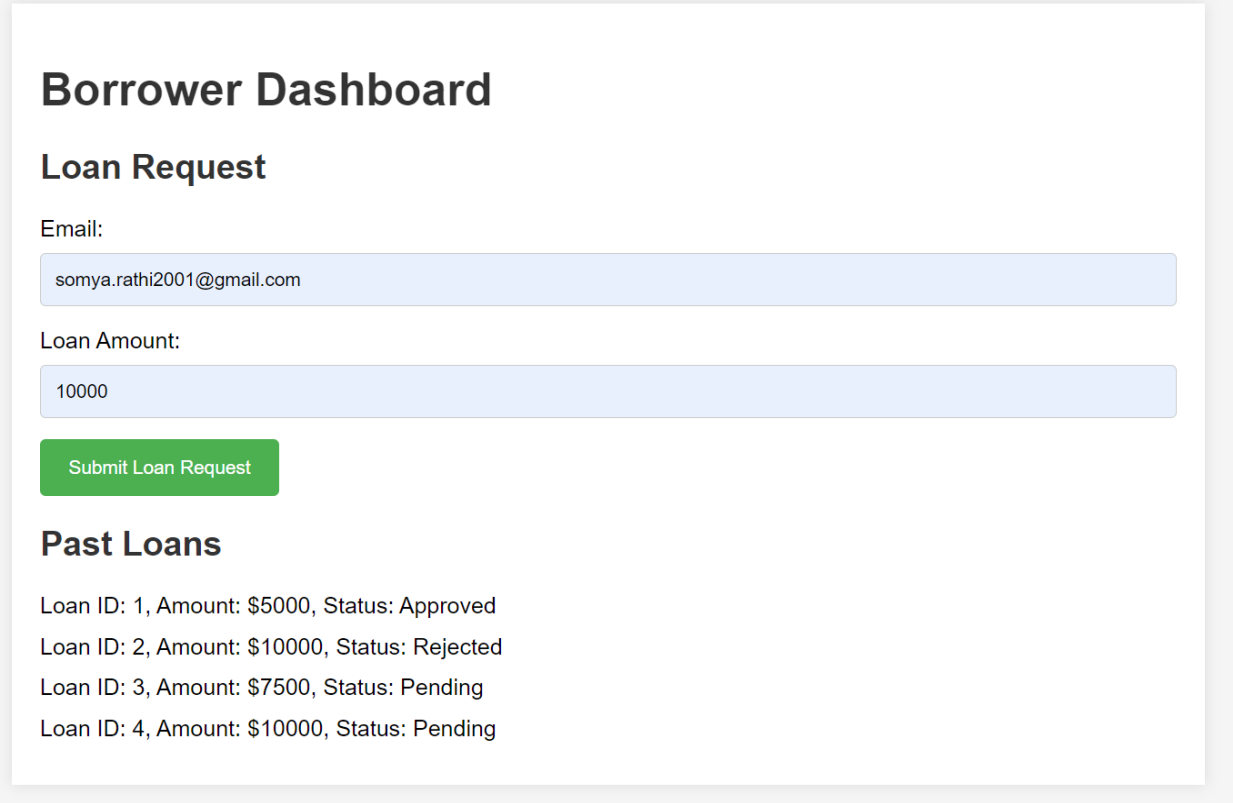
In the below picture, the loan request is processed by the borrower by writing his/her email address and the loan amount and it is reflected in past loans and along with their status.

Ex

Somya.rathi2001@gmail.com

10000

Gives a loan ID 4 with the amount stated and its status in the past loan section.



3.implementation of lenders functionality(client HTML script + Server-side JavaScript + CSS script for styling)

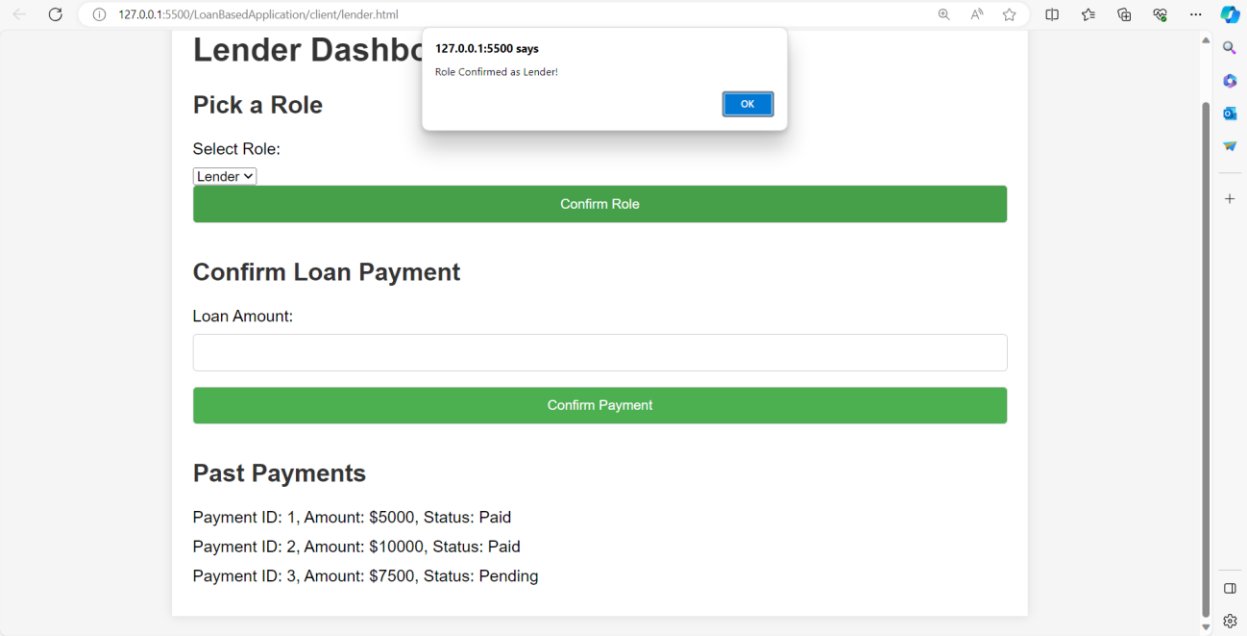
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Lender Dashboard</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
            background-color: #f5f5f5;
        }

        .container {
            max-width: 800px;
            margin: 20px auto;
```

```
        background-color: #fff;
        padding: 20px;
        box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    }

    h1, h2 {
        color: #333;
    }

    form {
        margin-top: 20px;
    }

    label {
        display: block;
        margin-bottom: 8px;
    }

    input[type="text"],
    input[type="number"],
    input[type="submit"] {
        width: 100%;
        padding: 10px;
        margin-bottom: 15px;
        border: 1px solid #ccc;
        border-radius: 4px;
        box-sizing: border-box;
    }

    input[type="submit"] {
        background-color: #4caf50;
        color: #fff;
        cursor: pointer;
    }

    input[type="submit"]:hover {
        background-color: #45a049;
    }

    ul {
        list-style: none;
        padding: 0;
        margin: 0;
    }
```

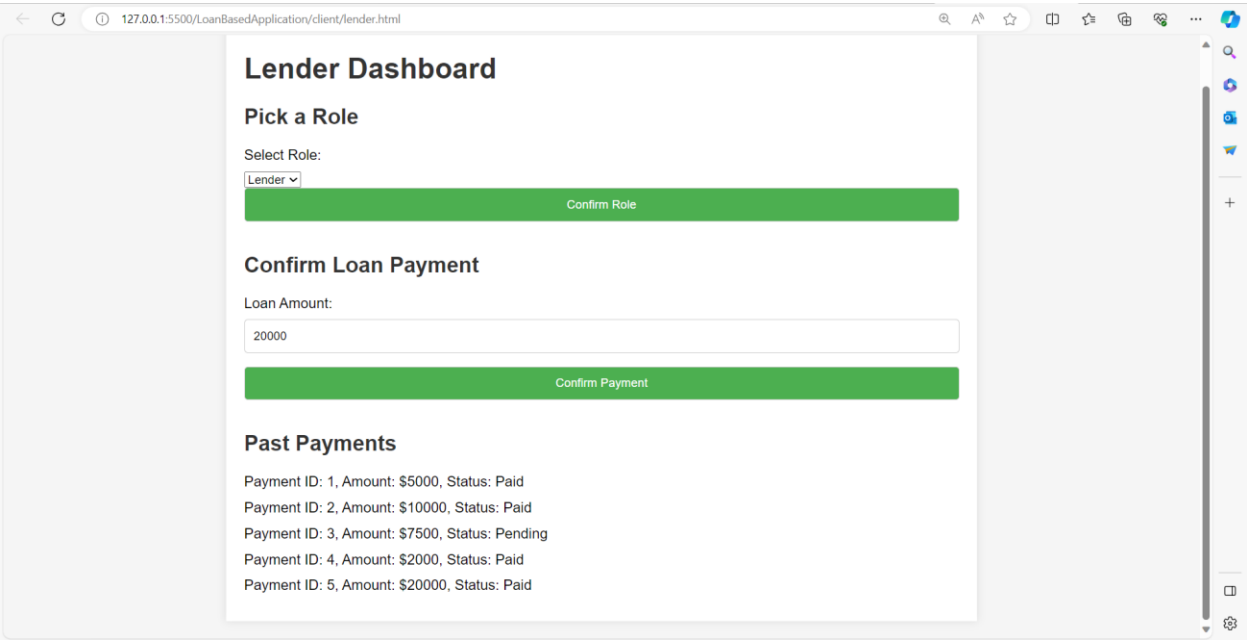
```
        li {
            margin-bottom: 10px;
        }
    </style>
</head>
<body>

<div class="container">
    <h1>Lender Dashboard</h1>

    <!-- Role Selection Form -->
    <h2>Pick a Role</h2>
    <form id="roleSelectionForm" action="#" method="post"
onsubmit="confirmRole(event)">
        <label for="role">Select Role:</label>
        <select id="role" name="role" required>
            <option value="lender">Lender</option>
        </select>

        <input type="submit" value="Confirm Role">
    </form>

    <!-- Loan Payment Confirmation Form -->
    <h2>Confirm Loan Payment</h2>
    <form id="loanPaymentForm" action="#" method="post"
onsubmit="confirmLoanPayment(event)">
        <label for="loanAmount">Loan Amount:</label>
        <input type="number" id="loanAmount" name="loanAmount" required>

        <input type="submit" value="Confirm Payment">
    </form>

    <!-- Past Payments Section -->
    <h2>Past Payments</h2>
    <ul id="pastPaymentsList"></ul>
</div>

<script>
    //data for past payments
    const pastPaymentsData = [
        { id: 1, amount: 5000, status: 'Paid' },
        { id: 2, amount: 10000, status: 'Paid' },
        { id: 3, amount: 7500, status: 'Pending' }
    ];
```

```
    // Display past payments on page load
    window.onload = displayPastPayments;

    function displayPastPayments() {
        const pastPaymentsList = document.getElementById('pastPaymentsList');
        pastPaymentsList.innerHTML = ''; // Clear previous content

        pastPaymentsData.forEach(payment => {
            const listItem = document.createElement('li');
            listItem.textContent = `Payment ID: ${payment.id}, Amount:
$${payment.amount}, Status: ${payment.status}`;
            pastPaymentsList.appendChild(listItem);
        });
    }

    function confirmRole(event) {
        event.preventDefault(); // Prevent the form from submitting
        alert("Role Confirmed as Lender!");
    }

    function confirmLoanPayment(event) {
        event.preventDefault(); // Prevent the form from submitting

        // Get form values
        const loanAmount = document.getElementById('loanAmount').value;

        //for processing loan payment confirmation
        const newPayment = { id: pastPaymentsData.length + 1, amount: loanAmount,
status: 'Paid' };
        pastPaymentsData.push(newPayment);

        // Display updated past payments
        displayPastPayments();
    }
</script>

</body>
</html>
```

Output:



In the below picture, two confirmed loans are passed to the borrower by lender and the status of payment is paid by lender to borrower amount have to be filled and confirm payment have to be clicked



4. implementation of admin functionality (client HTML script + Server-side JavaScript + CSS script for styling)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Admin Dashboard</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
            background-color: #f5f5f5;
        }

        .container {
            max-width: 800px;
            margin: 20px auto;
            background-color: #fff;
            padding: 20px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
        }

        h1, h2 {
            color: #333;
        }

        form {
            margin-top: 20px;
        }

        label {
            display: block;
            margin-bottom: 8px;
        }

        input[type="text"],
        input[type="number"],
        input[type="submit"] {
            width: 100%;
            padding: 10px;
            margin-bottom: 15px;
```

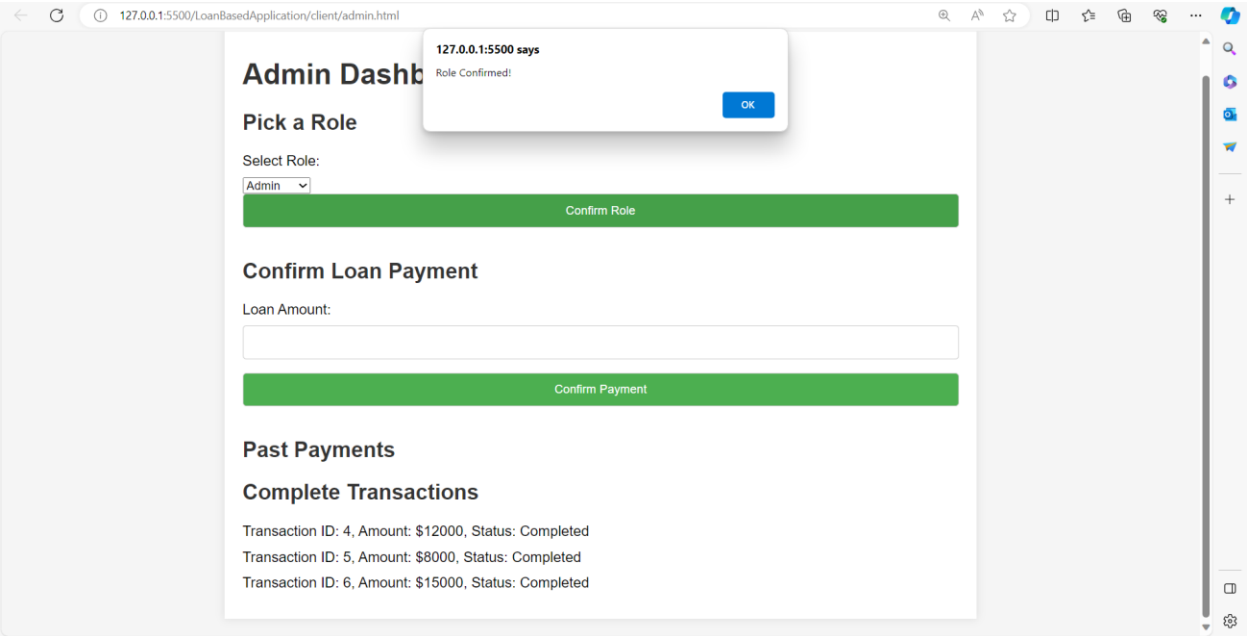
```
            border: 1px solid #ccc;
            border-radius: 4px;
            box-sizing: border-box;
        }

        input[type="submit"] {
            background-color: #4caf50;
            color: #fff;
            cursor: pointer;
        }

        input[type="submit"]:hover {
            background-color: #45a049;
        }

        ul {
            list-style: none;
            padding: 0;
            margin: 0;
        }

        li {
            margin-bottom: 10px;
        }
    </style>
</head>
<body>

<div class="container">
    <h1>Admin Dashboard</h1>

    <!-- Role Selection Form -->
    <h2>Pick a Role</h2>
    <form id="roleSelectionForm" action="#" method="post"
  onsubmit="confirmRole(event)">
        <label for="role">Select Role:</label>
        <select id="role" name="role" required>
            <option value="admin">Admin</option>
            <option value="lender">Lender</option>
            <option value="borrower">Borrower</option>
        </select>

        <input type="submit" value="Confirm Role">
    </form>

```

```
    <!-- Loan Payment Confirmation Form -->
    <h2>Confirm Loan Payment</h2>
    <form id="loanPaymentForm" action="#" method="post"
onsubmit="confirmLoanPayment(event)">
        <label for="loanAmount">Loan Amount:</label>
        <input type="number" id="loanAmount" name="loanAmount"
required>

        <input type="submit" value="Confirm Payment">
    </form>

    <!-- Past Payments Section -->
      <h2>Past Payments</h2>
      <ul id="pastPaymentsList"></ul>

      <!-- Complete Transactions Section (for Admin) -->
      <h2>Complete Transactions</h2>
      <ul id="completeTransactionsList"></ul>
  </div>

  <script>
      //data for past payments
      const pastPaymentsData = [
          { id: 1, amount: 5000, status: 'Paid' },
          { id: 2, amount: 10000, status: 'Paid' },
          { id: 3, amount: 7500, status: 'Pending' }
      ];

      // Display past payments on page load
      window.onload = displayPastPayments;

      function displayPastPayments() {
          const pastPaymentsList =
document.getElementById('pastPaymentsList');
          pastPaymentsList.innerHTML = ''; // Clear previous content

          pastPaymentsData.forEach(payment => {
              const listItem = document.createElement('li');
              listItem.textContent = `Payment ID: ${payment.id},
Amount: $${payment.amount}, Status: ${payment.status}`;
              pastPaymentsList.appendChild(listItem);
          });
      }

      //data for complete transactions for Admin
```

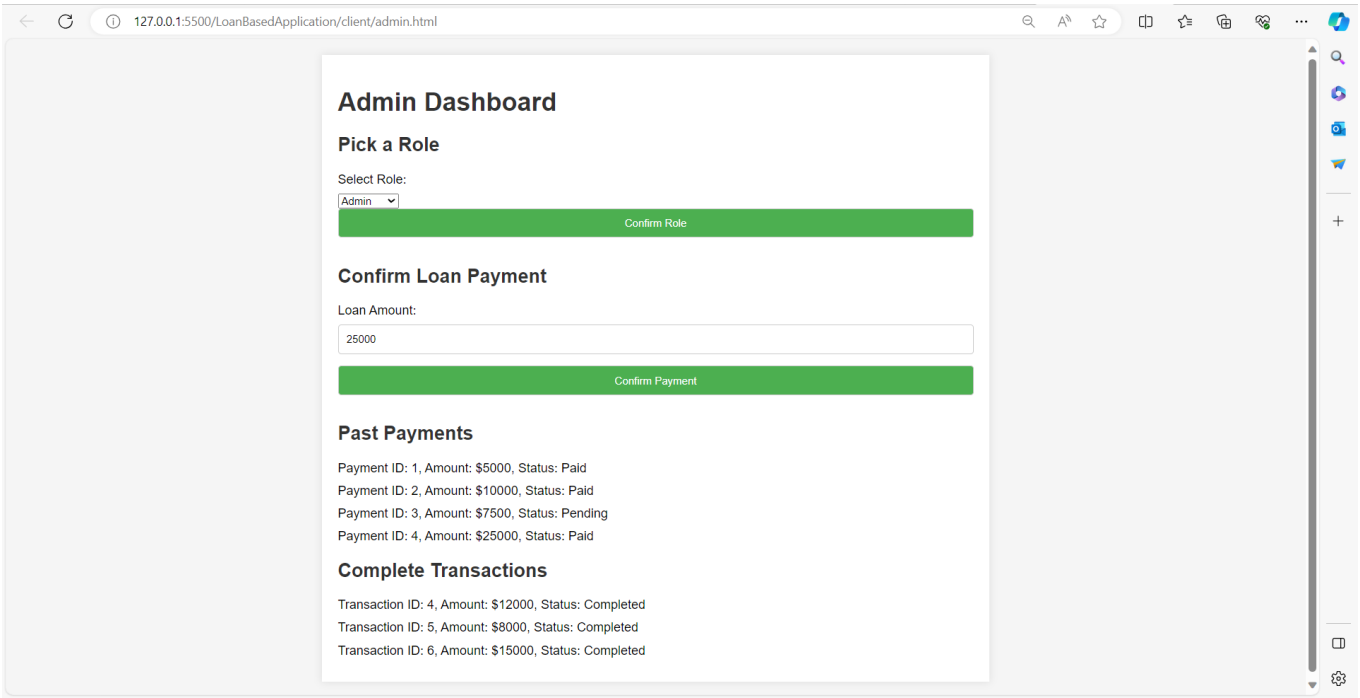
```
131.       const completeTransactionsData = [
132.           { id: 4, amount: 12000, status: 'Completed' },
133.           { id: 5, amount: 8000, status: 'Completed' },
134.           { id: 6, amount: 15000, status: 'Completed' }
135.       ];
136.
137.       // Display complete transactions on page load for Admin
138.       window.onload = displayCompleteTransactions;
139.
140.       function displayCompleteTransactions() {
141.           const completeTransactionsList =
   document.getElementById('completeTransactionsList');
142.           completeTransactionsList.innerHTML = ''; // Clear previous
   content
143.
144.           completeTransactionsData.forEach(transaction => {
145.               const listItem = document.createElement('li');
146.               listItem.textContent = `Transaction ID:
   ${transaction.id}, Amount: $${transaction.amount}, Status:
   ${transaction.status}`;
147.               completeTransactionsList.appendChild(listItem);
148.           });
149.       }
150.
151.       function confirmRole(event) {
152.           event.preventDefault(); // Prevent the form from submitting
153.           alert("Role Confirmed!");
154.       }
155.
156.       function confirmLoanPayment(event) {
157.           event.preventDefault(); // Prevent the form from submitting
158.
159.           // Get form values
160.           const loanAmount =
   document.getElementById('loanAmount').value;
161.
162.           //for processing loan payment confirmation
163.           const newPayment = { id: pastPaymentsData.length + 1,
   amount: loanAmount, status: 'Paid' };
164.           pastPaymentsData.push(newPayment);
165.
166.           // Display updated past payments
167.           displayPastPayments();
168.       }
169.   </script>
```

```
170.   </body>
171.   </html>
172.
```

Output:



In the below screenshot depicts the admin track functionality that is admin can verify both past payments and complete transactions for the borrower and lender as soon as we confirm the admin role and enter the loan amount its adds up to past payments and tabulates the history of the borrower and lender



Conclusion:

The following solution is capable of solving all the functionality asked in the problem statement using the framework of meteor.js and have created a web application for the same with different functionalities of admin, borrower, and lender separately.

GitHub link - https://github.com/somyarathi/LoanBasedApplication