Name: Itisha Mishra
Roll No – CS23026 **ADVANCED APPLICATION DEVELOPMENT**

PRACTICAL NO. 1

Aim: Write a program to implement MongoDB data models.

Program:

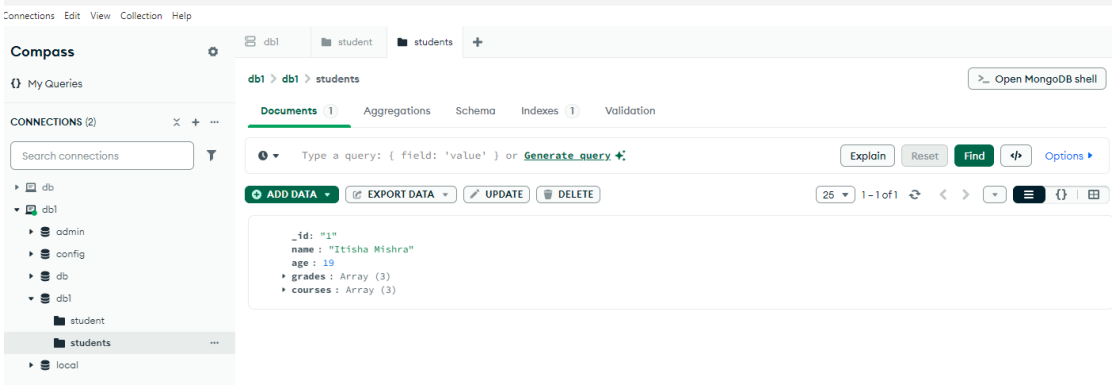
```
const { MongoClient } = require('mongodb');
const uri = "mongodb://localhost:27017/"; // Replace with your connection string
const client = new MongoClient(uri);
async function run() {
  try {
    await client.connect();
    console.log('Connected to MongoDB');
const db = client.db('db1');
const studentsCollection = db.collection('students');
const coursesCollection = db.collection('courses');
const newStudent = {
  _id: '1',
  name: 'Itisha Mishra',
  age: 19,
  grades: [90, 85, 92],
  courses: ['Math', 'Science', 'History']
};
const result = await studentsCollection.insertOne(newStudent);
console.log('Inserted Document:', result);
const studentSchema = {
  _id: { type: String, required: true }, // Using String for simplicity
  name: { type: String, required: true },
  age: { type: Number, min: 0 },
  grades: { type: Array, of: Number },
  courses: { type: Array, of: String }
};
const courseSchema = {
  _id: { type: String, required: true },
  name: { type: String, required: true },
  description: { type: String },
  instructor: { type: String }
};
  }
 finally {
    // Ensures that the client will close when you finish/error
    await client.close();
  }
}
run().catch(console.dir);
```

Output:

```
PS C:\Users\itisha mishra\desktop> notepad helloo.js
PS C:\Users\itisha mishra\desktop> node helloo.js
Connected to MongoDB
```

Name: Itisha Mishra
Roll No – CS23026 **ADVANCED APPLICATION DEVELOPMENT**

```
PS C:\Users\itisha mishra\desktop> node helloo.js
Connected to MongoDB
Inserted Document: { acknowledged: true, insertedId: '1' }
```



Name: Itisha Mishra
Roll No – CS23026 **ADVANCED APPLICATION DEVELOPMENT**

PRACTICAL NO. 2

Aim: Write a program to implement MongoDB data models and perform CRUD operation.

(Initialize npm):
npm init -y

```
PS C:\Users\itisha mishra> npm init -y
Wrote to C:\Users\itisha mishra\package.json:

{
  "name": "itisha-mishra",
  "version": "1.0.0",
  "main": "prac1.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}
```

(Install MongoDB Driver):
npm install mongodb

```
PS C:\Users\itisha mishra> npm install mongodb

added 12 packages, and audited 13 packages in 5s

found 0 vulnerabilities
```

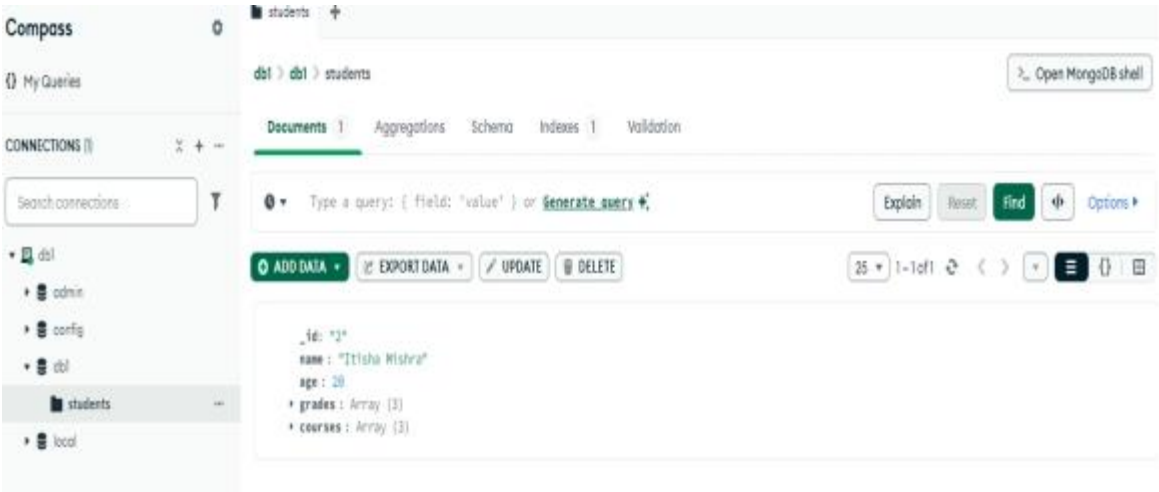
(Connect to MongoDB and Insert Data):

```
const { MongoClient } = require('mongodb');
const uri = "mongodb://localhost:27017/"; // Replace with your connection string
const client = new MongoClient(uri);
async function run() {
 try {
  await client.connect();
  console.log('Connected to MongoDB');
  const db = client.db('db1');
  const studentsCollection = db.collection('students');
  // Insert Data (Check for duplicates first)
  const newStudent = {
   _id: '2',
   name: 'Itisha Mishra',
   age: 20,
   grades: [90, 85, 92],
   courses: ['Math', 'Science', 'History']
  };
  // Check if a document with _id = "2" exists
  const existingStudent = await studentsCollection.findOne({ _id: '2' });
  if (!existingStudent) {
   const insertResult = await studentsCollection.insertOne(newStudent);
   console.log('Inserted Document:', insertResult);
  } else {
   console.log('Document with _id "2" already exists. Skipping insertion.');
  }
  // Read Data
  const query = { age: { $gte: 18 } }; // Fetch students aged 18 or above
  const cursor = studentsCollection.find(query);
  const results = await cursor.toArray();
```

```
    console.log('Found Documents:', results);
    // Update Data
    const filter = { name: 'Itisha Mishra' };
    const updateDoc = {
      $set: { age: 21 }
    };
    const updateResult = await studentsCollection.updateOne(filter, updateDoc);
    console.log('Updated Document:', updateResult);
    // Delete Data
    const deleteQuery = { age: { $lt: 18 } }; // Delete students below 18 years
    const deleteResult = await studentsCollection.deleteMany(deleteQuery);
    console.log('Deleted Documents:', deleteResult);
  } catch (error) {
    console.error('Error:', error);
  } finally {
    // Close the connection
    await client.close();
  }
}
run().catch(console.dir);
```

```
PS C:\Users\itisha mishra> node prac1.js
Connected to MongoDB
Inserted Document: { acknowledged: true, insertedId: '1' }
```

(Create Collection):



(Read Data)

```
c:\Users\itisha mishra>node prac1.js
Connected to MongoDB
Document with _id "2" already exists. Skipping insertion.
Found Documents: [
  {
    _id: '2',
    name: 'Itisha Mishra',
    age: 20,
    grades: [ 90, 85, 92 ],
    courses: [ 'Math', 'Science', 'History' ]
  }
]
```

(Update Data)

```
Updated Document: {
  acknowledged: true,
  modifiedCount: 1,
  upsertedId: null,
  upsertedCount: 0,
  matchedCount: 1
}
```

(Delete Data)

```
Deleted Documents: { acknowledged: true, deletedCount: 0 }
```

(Close Connection)

```
MongoDB connection closed.
```

Name: Itisha Mishra
Roll No – CS23026 **ADVANCED APPLICATION DEVELOPMENT**

PRACTICAL NO. 3

Aim: Write a program to perform validation of a form using AngularJS.

Program: (Html)

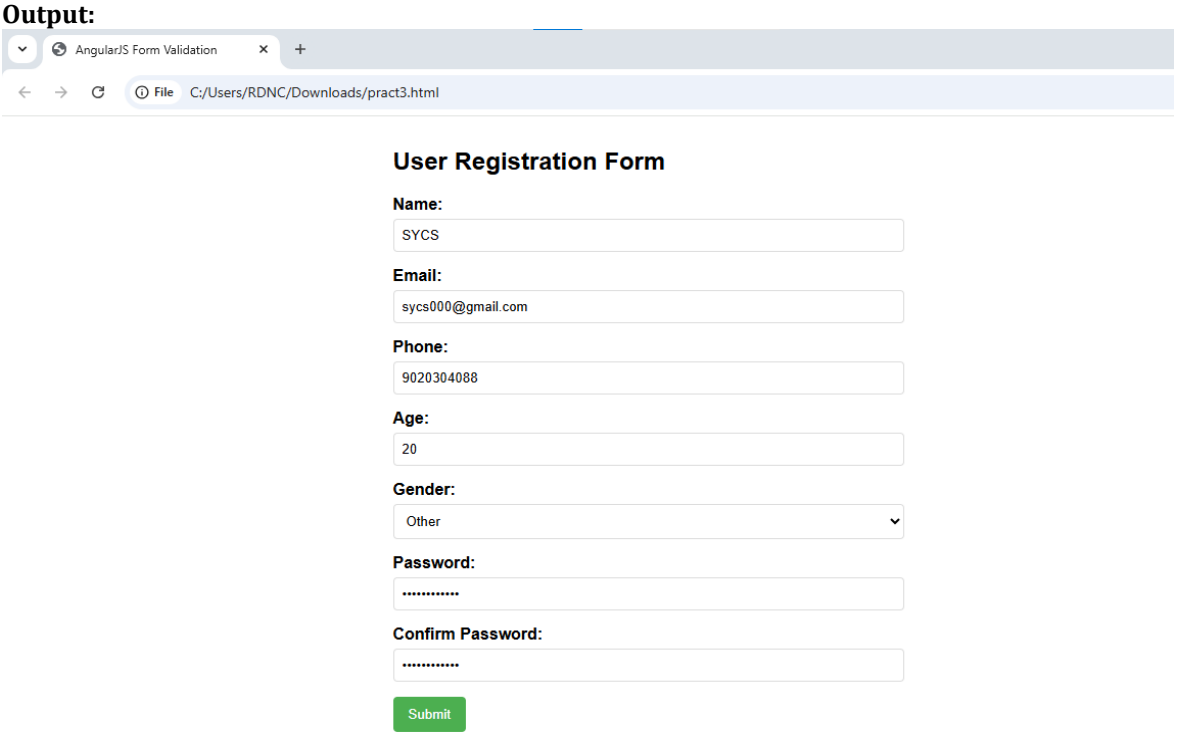
```
<!DOCTYPE html>
<html ng-app="validationApp">
<head>
  <title>AngularJS Form Validation</title>
  <style>
    .form-container {
      max-width: 500px;
      margin: 0 auto;
      padding: 20px;
      font-family: Arial, sans-serif;
    }
    .form-group {
      margin-bottom: 15px;
    }
    label {
      display: block;
      margin-bottom: 5px;
      font-weight: bold;
    }
    input, select {
      width: 100%;
      padding: 8px;
      border: 1px solid #ddd;
      border-radius: 4px;
      box-sizing: border-box;
    }
    button {
      padding: 10px 15px;
      background-color: #4CAF50;
      color: white;
      border: none;
      border-radius: 4px;
      cursor: pointer;
    }
    button:disabled {
      background-color: #cccccc;
      cursor: not-allowed;
    }
    .error {
      color: red;
      font-size: 12px;
      margin-top: 5px;
    }
    input.ng-invalid.ng-touched {
      border: 1px solid red;
    }
    .success-message {
      color: green;
      font-weight: bold;
      margin-top: 15px;
    }
  </style>
</head>
```

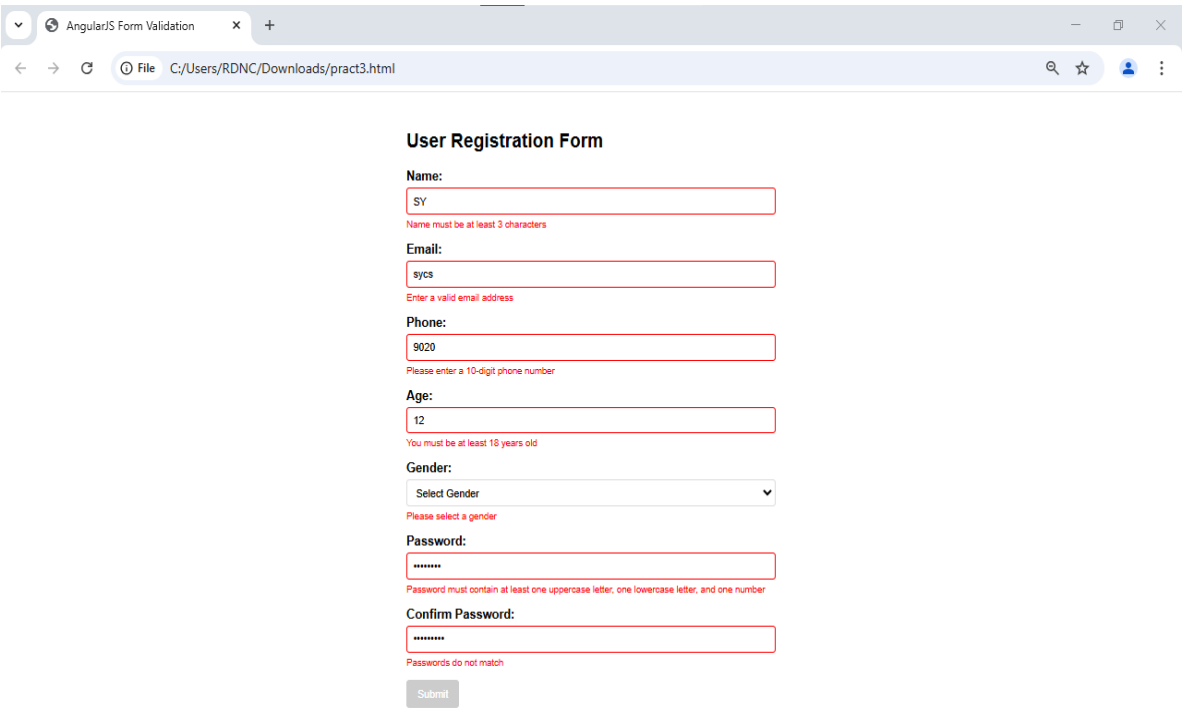
```
<body>
 <div class="form-container" ng-controller="validationController">
  <h2>User Registration Form</h2>

  <form name="userForm" ng-submit="submitForm(userForm.$valid)" novalidate>

    <!-- NAME -->
    <div class="form-group">
      <label>Name:</label>
      <input type="text" name="name" ng-model="user.name" required ng-minlength="3" ng-
maxlength="50">
      <div class="error" ng-show="userForm.name.$invalid && userForm.name.$touched">
        <span ng-show="userForm.name.$error.required">Name is required</span>
        <span ng-show="userForm.name.$error.minlength">Name must be at least 3
characters</span>
        <span ng-show="userForm.name.$error.maxlength">Name cannot exceed 50
characters</span>
      </div>
    </div>

    <!-- EMAIL -->
    <div class="form-group">
      <label>Email:</label>
      <input type="email" name="email" ng-model="user.email" required>
      <div class="error" ng-show="userForm.email.$invalid && userForm.email.$touched">
        <span ng-show="userForm.email.$error.required">Email is required</span>
        <span ng-show="userForm.email.$error.email">Enter a valid email address</span>
      </div>
    </div>

    <!-- PHONE -->
    <div class="form-group">
      <label>Phone:</label>
      <input type="text" name="phone" ng-model="user.phone" ng-pattern="/^[0-9]{10}$/"
required>
      <div class="error" ng-show="userForm.phone.$invalid && userForm.phone.$touched">
        <span ng-show="userForm.phone.$error.required">Phone number is required</span>
        <span ng-show="userForm.phone.$error.pattern">Please enter a 10-digit phone
number</span>
      </div>
    </div>

    <!-- AGE -->
    <div class="form-group">
      <label>Age:</label>
      <input type="number" name="age" ng-model="user.age" min="18" max="120" required>
      <div class="error" ng-show="userForm.age.$invalid && userForm.age.$touched">
        <span ng-show="userForm.age.$error.required">Age is required</span>
        <span ng-show="userForm.age.$error.min">You must be at least 18 years old</span>
        <span ng-show="userForm.age.$error.max">Age cannot exceed 120 years</span>
      </div>
    </div>

    <!-- GENDER -->
    <div class="form-group">
      <label>Gender:</label>
      <select name="gender" ng-model="user.gender" required>
        <option value="">Select Gender</option>
        <option value="male">Male</option>
```

```
        <option value="female">Female</option>
        <option value="other">Other</option>
      </select>
      <div class="error" ng-show="userForm.gender.$invalid && userForm.gender.$touched">
        <span ng-show="userForm.gender.$error.required">Please select a gender</span>
      </div>
    </div>

    <!-- PASSWORD -->
    <div class="form-group">
      <label>Password:</label>
      <input type="password" name="password" ng-model="user.password" required ng-
minlength="8" ng-pattern="/(?=.*[a-z])(?=.*[A-Z])(?=.*[0-9])/">
      <div class="error" ng-show="userForm.password.$invalid &&
userForm.password.$touched">
        <span ng-show="userForm.password.$error.required">Password is required</span>
        <span ng-show="userForm.password.$error.minlength">Password must be at least 8
characters</span>
        <span ng-show="userForm.password.$error.pattern">Password must contain at least one
uppercase letter, one lowercase letter, and one number</span>
      </div>
    </div>

    <!-- CONFIRM PASSWORD -->
    <div class="form-group">
      <label>Confirm Password:</label>
      <input type="password" name="confirmPassword" ng-model="user.confirmPassword"
required compare-to="user.password">
      <div class="error" ng-show="userForm.confirmPassword.$invalid &&
userForm.confirmPassword.$touched">
        <span ng-show="userForm.confirmPassword.$error.required">Please confirm your
password</span>
        <span ng-show="userForm.confirmPassword.$error.compareTo">Passwords do not
match</span>
      </div>
    </div>

    <!-- SUBMIT BUTTON -->
    <button type="submit" ng-disabled="userForm.$invalid">Submit</button>
    <!-- SUCCESS MESSAGE -->
    <div class="success-message" ng-show="formSubmitted">
      Form submitted successfully!
    </div>
  </form>
</div>
<!-- Load AngularJS -->
<script src="https://cdnjs.cloudflare.com/ajax/libs/angular.js/1.8.2/angular.min.js"></script>

<script>
  // Create Angular app
  var validationApp = angular.module('validationApp', []);
  // Create custom directive for password matching
  validationApp.directive('compareTo', function() {
    return {
      require: "ngModel",
      scope: {
        compareTo: "=compareTo"
      },
      link: function(scope, element, attributes, ngModel) {
```

```
        ngModel.$validators.compareTo = function(modelValue) {
          return modelValue === scope.compareTo;
        };
        scope.$watch("compareTo", function() {
          ngModel.$validate();
        });
      }
    };
  });

  // Controller
  validationApp.controller('validationController', ['$scope', function($scope) {
    // Initialize user object
    $scope.user = {
      name: '',
      email: '',
      phone: '',
      age: '',
      gender: '',
      password: '',
      confirmPassword: ''
    };
    $scope.formSubmitted = false;
    // Form submit handler
    $scope.submitForm = function(isValid) {
      if (isValid) {
        console.log('Form submitted successfully');
        console.log($scope.user);
        $scope.formSubmitted = true;
        // In a real application, you would send the data to the server here
        // $http.post('/api/users', $scope.user).then(function(response) {
        //   console.log('Server response:', response);
        // });
      } else {
        console.log('Form has errors');
      }
    };
  }]);
 </script>
</body>
</html>
```

Output:


AngularJS Form Validation
File C:/Users/RDNC/Downloads/pract3.html
User Registration Form
Name:
SYCS
Email:
sycs000@gmail.com
Phone:
9020304088
Age:
20
Gender:
Other
Password:
Confirm Password:
Submit


AngularJS Form Validation
File C:/Users/RDNC/Downloads/pract3.html
User Registration Form
Name:
SY
Name must be at least 3 characters
Email:
sycs
Enter a valid email address
Phone:
9020
Please enter a 10-digit phone number
Age:
12
You must be at least 18 years old
Gender:
Select Gender
Please select a gender
Password:
Password must contain at least one uppercase letter, one lowercase letter, and one number
Confirm Password:
Passwords do not match
Submit

Name: Itisha Mishra
Roll No – CS23026 **ADVANCED APPLICATION DEVELOPMENT**

PRACTICAL NO. 4

Aim: Write a program to create and implement modules and controllers in Angular JS.
Program:
(html)

```
<!DOCTYPE html>
<html>
<head>
 <title>AngularJS 1.x Modules and Controllers</title>
 <style>
  body {
   font-family: Arial, sans-serif;
   margin: 0;
   padding: 20px;
   color: #333;
  }
  .container {
   max-width: 900px;
   margin: 0 auto;
  }
  .section {
   border: 1px solid #ddd;
   border-radius: 4px;
   padding: 15px;
   margin-bottom: 20px;
   background-color: #f9f9f9;
  }
  h1, h2, h3 {
   color: #2c3e50;
  }
  input, select, button {
   margin: 5px 0;
   padding: 8px;
   border-radius: 4px;
   border: 1px solid #ccc;
  }
  button {
   background-color: #3498db;
   color: white;
   border: none;
   cursor: pointer;
  }
  button:hover {
   background-color: #2980b9;
  }
  table {
   width: 100%;
   border-collapse: collapse;
   margin-top: 10px;
  }
  table, th, td {
   border: 1px solid #ddd;
  }
  th, td {
   padding: 10px;
   text-align: left;
  }
  th {
```

```
    background-color: #f2f2f2;
  }
  .active {
    background-color: #e0f7fa;
  }
  .text-danger {
    color: #e74c3c;
  }
  .nav-tabs {
    list-style-type: none;
    margin: 0;
    padding: 0;
    overflow: hidden;
    border-bottom: 1px solid #ccc;
  }
  .nav-tabs li {
    float: left;
  }
  .nav-tabs li a {
    display: block;
    padding: 10px 15px;
    text-decoration: none;
    color: #3498db;
    cursor: pointer;
  }
  .nav-tabs li a:hover {
    background-color: #eee;
  }
  .nav-tabs li a.active {
    background-color: #f9f9f9;
    border: 1px solid #ccc;
    border-bottom-color: transparent;
    color: #333;
  }
  .tab-content {
    padding: 15px;
    border: 1px solid #ccc;
    border-top: none;
  }
 </style>
</head>
<body ng-app="mainApp">

<div class="container">
 <h1>AngularJS 1.x Modules and Controllers Demo</h1>

 <!-- Tabbed interface to demonstrate communications between controllers -->
 <div ng-controller="TabController">
  <ul class="nav-tabs">
    <li ng-repeat="tab in tabs">
      <a ng-class="{ active: isActiveTab(tab.id) }" ng-
click="setActiveTab(tab.id)">{{ tab.title }}</a>
    </li>
  </ul>

  <div class="tab-content">
    <!-- First tab: UserController demonstration -->
    <div ng-show="activeTab === 1">
      <div class="section" ng-controller="UserController">
```

```
      <h2>User Management Module</h2>
      <p>This section demonstrates a simple user management controller.</p>

      <form ng-submit="addUser()">
        <div>
          <label>Name:</label>
          <input type="text" ng-model="newUser.name" required>
        </div>
        <div>
          <label>Email:</label>
          <input type="email" ng-model="newUser.email" required>
        </div>
        <div>
          <label>Role:</label>
          <select ng-model="newUser.role" required>
            <option value="">Select a role</option>
            <option value="Admin">Admin</option>
            <option value="User">User</option>
            <option value="Guest">Guest</option>
          </select>
        </div>
        <button type="submit">Add User</button>
      </form>

      <h3>User List</h3>
      <input type="text" ng-model="userSearch" placeholder="Filter users...">
      <table>
        <thead>
          <tr>
            <th>Name</th>
            <th>Email</th>
            <th>Role</th>
            <th>Actions</th>
          </tr>
        </thead>
        <tbody>
          <tr ng-repeat="user in users | filter:userSearch" ng-class="{ active: selectedUser ===
user }">
            <td>{{ user.name }}</td>
            <td>{{ user.email }}</td>
            <td>{{ user.role }}</td>
            <td>
              <button ng-click="selectUser(user)">Select</button>
              <button ng-click="removeUser($index)">Remove</button>
            </td>
          </tr>
        </tbody>
      </table>

      <div ng-show="selectedUser">
        <h3>Selected User</h3>
        <p>Name: {{ selectedUser.name }}</p>
        <p>Email: {{ selectedUser.email }}</p>
        <p>Role: {{ selectedUser.role }}</p>
      </div>
    </div>
  </div>

  <!-- Second tab: ProductController demonstration -->
```

```
    <div ng-show="activeTab === 2">
      <div class="section" ng-controller="ProductController">
        <h2>Product Management Module</h2>
        <p>This section demonstrates a product management controller with service
dependency.</p>

        <form ng-submit="addProduct()">
          <div>
            <label>Product Name:</label>
            <input type="text" ng-model="newProduct.name" required>
          </div>
          <div>
            <label>Price ($):</label>
            <input type="number" ng-model="newProduct.price" min="0" required>
          </div>
          <div>
            <label>Category:</label>
            <select ng-model="newProduct.category" required>
              <option value="">Select a category</option>
              <option value="Electronics">Electronics</option>
              <option value="Clothing">Clothing</option>
              <option value="Food">Food</option>
              <option value="Books">Books</option>
            </select>
          </div>
          <div>
            <label>In Stock:</label>
            <input type="checkbox" ng-model="newProduct.inStock">
          </div>
          <button type="submit">Add Product</button>
        </form>

        <h3>Product Inventory</h3>
        <input type="text" ng-model="productSearch" placeholder="Filter products...">
        <label>
          <input type="checkbox" ng-model="showOnlyInStock">
          Show only in-stock items
        </label>

        <table>
          <thead>
            <tr>
              <th>Name</th>
              <th>Price</th>
              <th>Category</th>
              <th>Status</th>
              <th>Actions</th>
            </tr>
          </thead>
          <tbody>
            <tr ng-repeat="product in filterProducts() | filter:productSearch" ng-class="{ active:
selectedProduct === product }">
              <td>{{ product.name }}</td>
              <td>${{ product.price.toFixed(2) }}</td>
              <td>{{ product.category }}</td>
              <td ng-class="{'text-danger': !product.inStock}">
                {{ product.inStock ? 'In Stock' : 'Out of Stock' }}
              </td>
              <td>
```

```
            <button ng-click="toggleStock(product)">
              {{ product.inStock ? 'Mark Out of Stock' : 'Mark In Stock' }}
            </button>
            <button ng-click="removeProduct($index)">Remove</button>
          </td>
        </tr>
      </tbody>
    </table>

    <h3>Product Statistics</h3>
    <p>Total Products: {{ products.length }}</p>
    <p>In-Stock Products: {{ getStockCount() }}</p>
    <p>Average Price: ${{ getAveragePrice() }}</p>
  </div>
</div>

<!-- Third tab: Communication between controllers demo -->
<div ng-show="activeTab === 3">
  <div class="section">
    <h2>Communication Between Controllers</h2>
    <p>This section demonstrates communication between controllers using services.</p>

    <div ng-controller="NotificationController">
      <h3>Notification Center</h3>
      <button ng-click="sendMessage('This is a test message from the Notification controller!')">
        Send Test Message
      </button>
      <button ng-click="clearMessages()">Clear All Messages</button>

      <h4>Current Messages:</h4>
      <div ng-show="messages.length === 0">
        <p><em>No messages available</em></p>
      </div>
      <ul>
        <li ng-repeat="msg in messages track by $index">
          {{ msg }} <button ng-click="removeMessage($index)">Dismiss</button>
        </li>
      </ul>
    </div>

    <div ng-controller="ReceiverController">
      <h3>Message Receiver</h3>
      <button ng-click="checkMessages()">Check for Messages</button>
      <button ng-click="sendReply()">Send Reply</button>

      <div ng-show="lastMessage">
        <h4>Last Message Received:</h4>
        <p>{{ lastMessage }}</p>
      </div>
    </div>
  </div>
</div>
</div>
</div>
</div>

<!-- Load AngularJS -->
<script src="https://cdnjs.cloudflare.com/ajax/libs/angular.js/1.8.2/angular.min.js"></script>
```

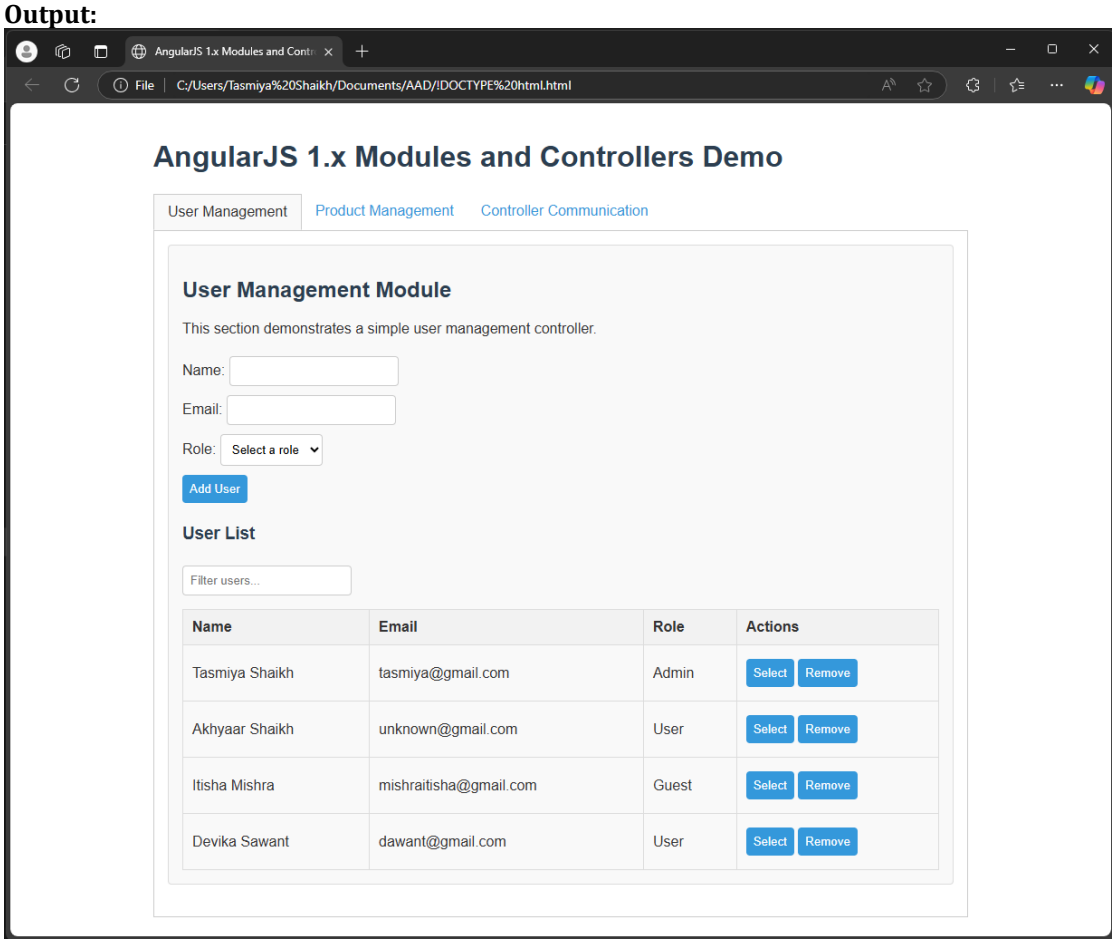
```
<!-- Application Script -->
<script>
 // Create main application module
 var mainApp = angular.module('mainApp', []);

 // Create a service for sharing data between controllers
 mainApp.service('SharedDataService', function() {
  var sharedData = {
   messages: []
  };

  return {
   getMessages: function() {
    return sharedData.messages;
   },
   addMessage: function(message) {
    sharedData.messages.push(message);
   },
   clearMessages: function() {
    sharedData.messages = [];
   },
   removeMessage: function(index) {
    sharedData.messages.splice(index, 1);
   }
  };
 });

 // Create a utility service
 mainApp.service('UtilityService', function() {
  return {
   generateId: function() {
    return Math.floor(Math.random() * 10000);
   },
   formatDate: function(date) {
    return date.toLocaleDateString();
   },
   calculateAverage: function(items, property) {
    if (items.length === 0) return 0;
    var sum = items.reduce(function(total, item) {
     return total + (item[property] || 0);
    }, 0);
    return sum / items.length;
   }
  };
 });

 // Tab Controller
 mainApp.controller('TabController', function($scope) {
  $scope.tabs = [
   { id: 1, title: 'User Management' },
   { id: 2, title: 'Product Management' },
   { id: 3, title: 'Controller Communication' }
  ];

  $scope.activeTab = 1;

  $scope.setActiveTab = function(tabId) {
   $scope.activeTab = tabId;
  };
```

```
  $scope.isActiveTab = function(tabId) {
    return $scope.activeTab === tabId;
  };
});

// User Controller
mainApp.controller('UserController', function($scope) {
  // Initialize controller data
  $scope.users = [
    { name: 'John Doe', email: 'john@example.com', role: 'Admin' },
    { name: 'Jane Smith', email: 'jane@example.com', role: 'User' },
    { name: 'Mike Johnson', email: 'mike@example.com', role: 'Guest' }
  ];

  $scope.newUser = {
    name: '',
    email: '',
    role: ''
  };

  $scope.selectedUser = null;

  // Controller methods
  $scope.addUser = function() {
    $scope.users.push(angular.copy($scope.newUser));
    $scope.newUser = { name: '', email: '', role: '' };
  };

  $scope.removeUser = function(index) {
    if ($scope.selectedUser === $scope.users[index]) {
      $scope.selectedUser = null;
    }
    $scope.users.splice(index, 1);
  };

  $scope.selectUser = function(user) {
    $scope.selectedUser = user;
  };
});

// Product Controller
mainApp.controller('ProductController', function($scope, UtilityService) {
  // Initialize controller data with dependency on UtilityService
  $scope.products = [
    { name: 'Laptop', price: 999.99, category: 'Electronics', inStock: true },
    { name: 'T-Shirt', price: 24.99, category: 'Clothing', inStock: true },
    { name: 'Book', price: 14.99, category: 'Books', inStock: false }
  ];
  $scope.newProduct = {
    name: '',
    price: 0,
    category: '',
    inStock: true
  };
  $scope.showOnlyInStock = false;

  // Controller methods
  $scope.addProduct = function() {
```

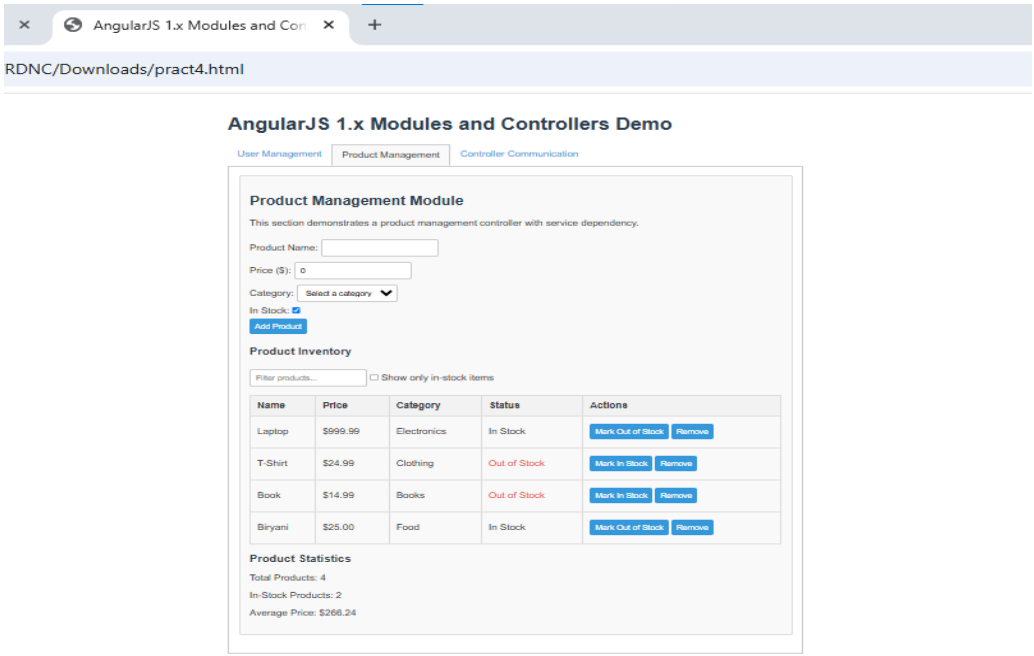
```
    $scope.products.push(angular.copy($scope.newProduct));
    $scope.newProduct = { name: '', price: 0, category: '', inStock: true };
  };

  $scope.removeProduct = function(index) {
    $scope.products.splice(index, 1);
  };

  $scope.toggleStock = function(product) {
    product.inStock = !product.inStock;
  };

  $scope.filterProducts = function() {
    if ($scope.showOnlyInStock) {
      return $scope.products.filter(function(product) {
        return product.inStock;
      });
    }
    return $scope.products;
  };

  $scope.getStockCount = function() {
    return $scope.products.filter(function(product) {
      return product.inStock;
    }).length;
  };

  $scope.getAveragePrice = function() {
    var avg = UtilityService.calculateAverage($scope.products, 'price');
    return avg.toFixed(2);
  };
});

// Notification Controller (demonstrates service usage)
mainApp.controller('NotificationController', function($scope, SharedDataService) {
  $scope.messages = SharedDataService.getMessages();
  $scope.sendMessage = function(message) {
    SharedDataService.addMessage(message);
  };
  $scope.clearMessages = function() {
    SharedDataService.clearMessages();
  };
  $scope.removeMessage = function(index) {
    SharedDataService.removeMessage(index);
  };
});

// Receiver Controller (demonstrates controller communication)
mainApp.controller('ReceiverController', function($scope, SharedDataService, UtilityService) {
  $scope.lastMessage = '';

  $scope.checkMessages = function() {
    var messages = SharedDataService.getMessages();
    $scope.lastMessage = messages.length > 0 ?
                    messages[messages.length - 1] :
                    'No messages available';
  };

  $scope.sendReply = function() {
```

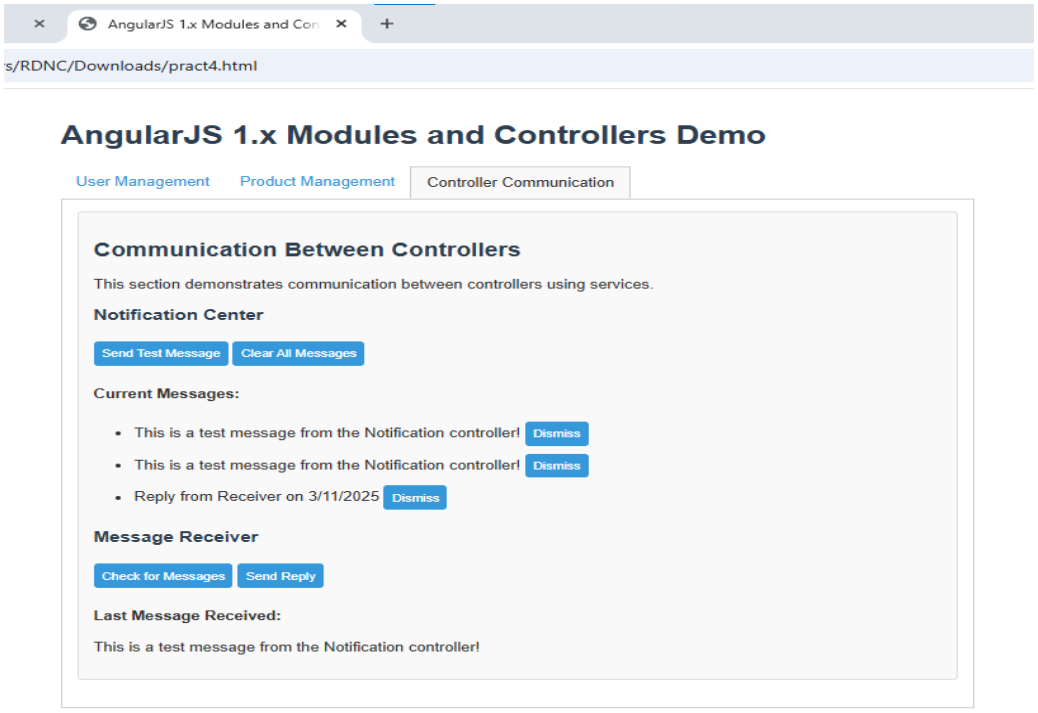
```
    var date = new Date();
    var formattedDate = UtilityService.formatDate(date);
    SharedDataService.addMessage('Reply from Receiver on ' + formattedDate);
  };
 });
</script>
</body>
</html>
```

Output:



Name: Itisha Mishra
Roll No – CS23026 **ADVANCED APPLICATION DEVELOPMENT**





Name: Itisha Mishra
Roll No – CS23026 **ADVANCED APPLICATION DEVELOPMENT**

PRACTICAL NO. 5

Aim: Write a program to implement Error Handling in Angular JS.

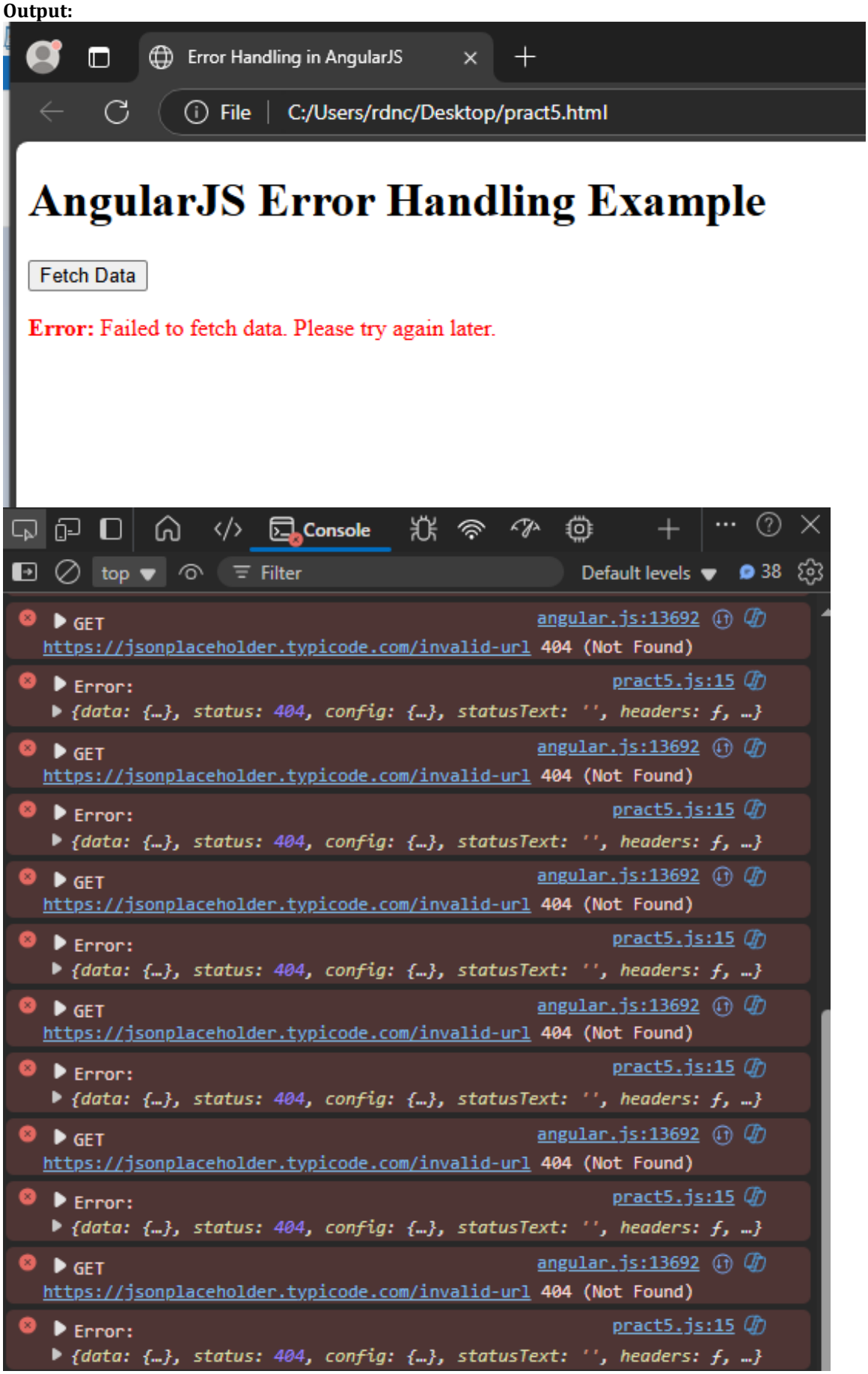
Program:
(Js)

```
angular.module('errorHandlingApp', [])
.controller('MainController', function($http)

{
  var vm = this;
  vm.data = '';
  vm.errorMessage = '';

  vm.fetchData = function() {
    $http.get('https://jsonplaceholder.typicode.com/invalid-url')
      .then(function(response) {
        vm.data = response.data;
      })
      .catch(function(error) {
        vm.errorMessage = 'Failed to fetch data. Please try again later.';
        console.error('Error:', error);
      })
  };
};
```

(html)

```
<!DOCTYPE html>
<html lang="en" ng-app="errorHandlingApp">
<head>
 <meta charset="UTF-8">
 <title>Error Handling in AngularJS</title>
 <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
 <script src="pract5.js"></script>
</head>
<body ng-controller="MainController as ctrl">
 <h1>AngularJS Error Handling Example</h1>
 <button ng-click="ctrl.fetchData()">Fetch Data</button>
 <p>{{ ctrl.data }}</p>
 <div ng-if="ctrl.errorMessage" style="color: red;">
  <strong>Error:</strong> {{ ctrl.errorMessage }}
 </div>
</body>
</html>
```

Output:


Error Handling in AngularJS
File
C:/Users/rdnc/Desktop/pract5.html
AngularJS Error Handling Example
Fetch Data
Error: Failed to fetch data. Please try again later.
Console
top
Filter
Default levels
38
GET https://jsonplaceholder.typicode.com/invalid-url 404 (Not Found)
angular.js:13692
Error:
pract5.js:15
{data: {…}, status: 404, config: {…}, statusText: '', headers: ƒ, …}

Name: Itisha Mishra
Roll No – CS23026 **ADVANCED APPLICATION DEVELOPMENT**

PRACTICAL NO. 6

Aim: Create an application for Customer / Students records using AngularJS.

Program:
(Install)
Node.js
AngularJs - npm install -g angular

(Create Project Folder)
mkdir student_records_app
cd student_records_app

(Structure)

```
student_records_app/
│
├── index.html
├── app/
│   ├── app.module.js
│   ├── app.controller.js
│   └── app.service.js
├── css/
│   └── styles.css
└── data/
    └── students.json
```

(HTML)

```
<!DOCTYPE html>
<html lang="en" ng-app="studentApp">
<head>
  <meta charset="UTF-8">
  <title>Student Records Management</title>
  <link rel="stylesheet" href="css/styles.css">

  <!-- AngularJS Library -->
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>

  <!-- AngularJS App Files -->
  <script src="app/app.module.js"></script>
  <script src="app/app.controller.js"></script>
</head>
<body ng-controller="StudentController as ctrl">

  <h1>Student Records Management</h1>

  <!-- Add Student Form -->
  <input type="text" ng-model="ctrl.newStudent.name" placeholder="Name">
  <input type="email" ng-model="ctrl.newStudent.email" placeholder="Email">
  <input type="text" ng-model="ctrl.newStudent.phone" placeholder="Phone">
  <button ng-click="ctrl.addStudent()">Add Student</button>

  <!-- Students Table -->
  <table border="1">
    <tr>
      <th>Name</th>
      <th>Email</th>
      <th>Phone</th>
      <th>Actions</th>
    </tr>
    <tr ng-repeat="student in ctrl.students track by $index">
      <td>{{ student.name }}</td>
      <td>{{ student.email }}</td>
```

```
        <td>{{ student.phone }}</td>
        <td>
          <button ng-click="ctrl.editStudent($index)">Edit</button>
          <button ng-click="ctrl.deleteStudent($index)">Delete</button>
        </td>
      </tr>
    </table>
</body>
</html>
```

(CSS)

```
/* General Page Styling */
body {
        font-family: Arial, sans-serif;
        background-color: #f4f4f4;
        text-align: center;
        margin: 20px;
        padding: 0;
      }
      /* Header */
      h1 {
        color: #333;
        margin-bottom: 20px;
      }
      /* Form Styling */
      input {
        padding: 8px;
        margin: 5px;
        border: 1px solid #ccc;
        border-radius: 5px;
        width: 200px;
        font-size: 14px;
      }
      button {
        padding: 8px 12px;
        margin: 5px;
        border: none;
        border-radius: 5px;
        background-color: #28a745;
        color: white;
        font-size: 14px;
        cursor: pointer;
      }
      button:hover {
        background-color: #218838;
      }

      /* Table Styling */
      table {
        width: 60%;
        margin: 20px auto;
        border-collapse: collapse;
        background: white;
        box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);
      }
      th, td {
        padding: 12px;
        border: 1px solid #ddd;
        text-align: left;
```

```
    }
    th {
      background-color: #007bff;
      color: white;
    }
    td {
      background-color: #ffffff;
    }

    /* Action Buttons */
    td button {
      margin: 3px;
      padding: 5px 8px;
      font-size: 12px;
    }

    /* Edit Button */
    td button:nth-child(1) {
      background-color: #ffc107;
      color: black;
    }

    td button:nth-child(1):hover {
      background-color: #e0a800;
    }

    /* Delete Button */
    td button:nth-child(2) {
      background-color: #dc3545;
    }

    td button:nth-child(2):hover {
      background-color: #c82333;
    }
```

(app.module.js)

```
angular.module('studentApp', []);
```

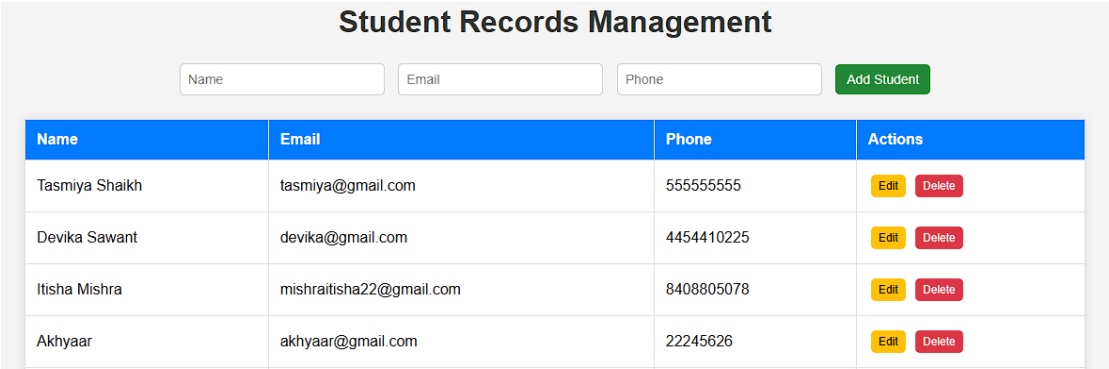
(app.controller.js)

```
angular.module('studentApp')
 .controller('StudentController', function() {
   var vm = this;

   // Sample Student Data
   vm.students = [
     { name: 'Avinash', email: 'avinash@example.com', phone: '123-456-7890' },
     { name: 'John Doe', email: 'john@example.com', phone: '987-654-3210' }
   ];

   // Add New Student
   vm.addStudent = function() {
     if (vm.newStudent) {
       vm.students.push(vm.newStudent);
       vm.newStudent = {}; // Reset input fields
     }
   };

   // Edit Student
   vm.editStudent = function(index) {
```

```
    vm.newStudent = angular.copy(vm.students[index]);
    vm.students.splice(index, 1); // Remove current entry to avoid duplication
  };

  // Delete Student
  vm.deleteStudent = function(index) {
    vm.students.splice(index, 1);
  };
});
```

(Open index.html)

Output:



Name: Itisha Mishra
Roll No – CS23026 **ADVANCED APPLICATION DEVELOPMENT**

PRACTICAL NO. 7

Aim: Write a program to create a simple web application using Express, Node JS and Angular.

JS.
Program:
(Back):

```
const express = require('express');
const bodyParser = require('body-parser');
const cors = require('cors');

const app = express();
const port = 3000;

app.use(cors());
app.use(bodyParser.json());

let students = [
  { id: 1, name: 'Avinash', age: 22, course: 'Computer Science' },
  { id: 2, name: 'John Doe', age: 24, course: 'Mathematics' }
];

// Fetch all students
app.get('/api/students', (req, res) => {
  res.json(students);
});

// Add a new student
app.post('/api/students', (req, res) => {
  const newStudent = req.body;
  newStudent.id = students.length + 1;
  students.push(newStudent);
  res.json(newStudent);
});

// Delete student
app.delete('/api/students/:id', (req, res) => {
  const studentId = parseInt(req.params.id);
  students = students.filter(student => student.id !== studentId);
  res.sendStatus(204);
});

app.listen(port, () => {
  console.log(`Server running at http://localhost:${port}`);
});
```

(front):

```
<!DOCTYPE html>
<html lang="en" ng-app="studentApp">

<head>
  <meta charset="UTF-8">
  <title>Student Management System</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
  <script src="app.js"></script>
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha3/dist/css/bootstrap.min.css">
</head>
```

```
<body ng-controller="StudentController as ctrl">

  <div class="container">
    <h1 class="my-4">Student Management System</h1>

    <input type="text" ng-model="ctrl.newStudent.name" class="form-control mb-2"
placeholder="Name">
    <input type="number" ng-model="ctrl.newStudent.age" class="form-control mb-2"
placeholder="Age">
    <input type="text" ng-model="ctrl.newStudent.course" class="form-control mb-2"
placeholder="Course">
    <button class="btn btn-primary mb-4" ng-click="ctrl.addStudent()">Add Student</button>

    <table class="table table-bordered">
      <thead>
        <tr>
          <th>ID</th>
          <th>Name</th>
          <th>Age</th>
          <th>Course</th>
          <th>Action</th>
        </tr>
      </thead>
      <tbody>
        <tr ng-repeat="student in ctrl.students">
          <td>{{ student.id }}</td>
          <td>{{ student.name }}</td>
          <td>{{ student.age }}</td>
          <td>{{ student.course }}</td>
          <td>
            <button class="btn btn-danger" ng-click="ctrl.deleteStudent(student.id)">Delete</button>
          </td>
        </tr>
      </tbody>
    </table>
  </div>
</body>
</html>
```

(For execution):

Step 1: npm init -y

```
C:\Users\RDNC>npm init -y
Wrote to C:\Users\RDNC\package.json:

{
  "name": "rdnc",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit
1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "bootstrap": "^5.3.3",
    "express": "^4.21.2"
  },
  "devDependencies": {},
  "keywords": [],
  "description": ""
}
```

Name: Itisha Mishra
Roll No – CS23026 **ADVANCED APPLICATION DEVELOPMENT**

Step 2: npm install express body-parser cors

```
C:\Users\RDNC>npm install express body-parser cors

added 2 packages, and audited 74 packages in 2s

16 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

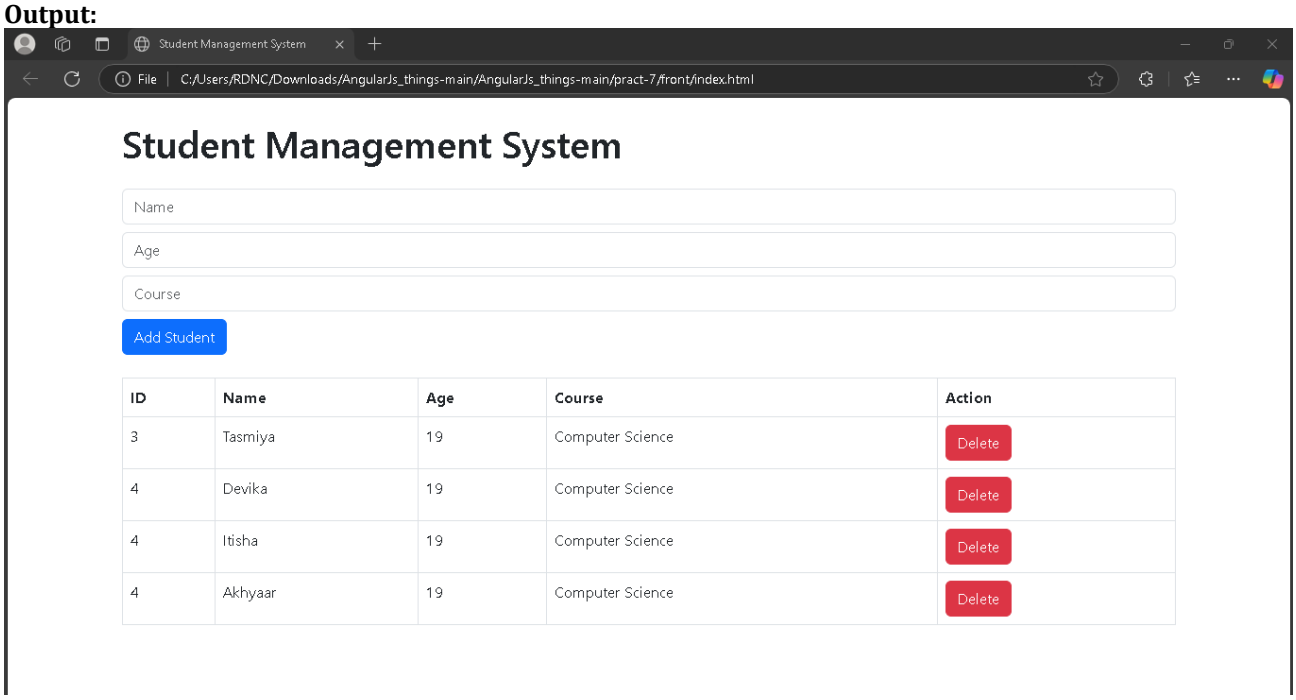
Step 3: cd back

```
C:\Users\RDNC\Downloads\AngularJs_things-main\AngularJs
_things-main\pract-7>cd back
```

Step 4: node server.js

```
C:\Users\RDNC\Downloads\AngularJs_things-main\AngularJs
_things-main\pract-7\back>node server.js
Server running at http://localhost:3000
```

Output:



Name: Itisha Mishra
Roll No – CS23026 **ADVANCED APPLICATION DEVELOPMENT**

PRACTICAL NO. 8

Aim: Write a program to create an app using Flutter for User Authentication.

(Project Structure)

```
auth_app/
|── android/                  # Android-specific files
|── ios/                      # iOS-specific files
|── lib/                      # Main Flutter code
|   |── main.dart             # Main entry point of the app
|   |── screens/              # UI Screens
|   |   |── login_screen.dart       # Login screen UI
|   |   |── register_screen.dart    # Registration screen UI
|   |   |── home_screen.dart        # Home screen UI (after login)
|   |── database/             # Database-related files
|   |   |── database_helper.dart    # SQLite database helper
|   |── services/             # Business logic and authentication
|   |   |── auth_service.dart       # Authentication logic (optional for
SharedPreferences)
|── pubspec.yaml              # Dependencies and project settings
|── README.md                 # Documentation
```

(main.dart)

```
import 'package:flutter/material.dart';
import 'screens/login_screen.dart';

void main() {
 runApp(MaterialApp(
  home: LoginScreen(),
 ));
}
```

(database_helper.dart)

```
import 'package:sqflite/sqflite.dart';
import 'package:path/path.dart';

class DatabaseHelper {
 static Database? _db;

 Future<Database> get db async {
  if (_db != null) {
   return _db!;
  }
  _db = await initDb();
  return _db!;
 }

 initDb() async {
  String path = join(await getDatabasesPath(), "users.db");
  var theDb = await openDatabase(path, version: 1, onCreate: _onCreate);
  return theDb;
 }

 void _onCreate(Database db, int version) async {
  await db.execute(
    "CREATE TABLE Users(id INTEGER PRIMARY KEY, username TEXT, password TEXT)");
 }

 Future<int> saveUser(String username, String password) async {
  var dbClient = await db;
  return await dbClient.insert("Users", {
   "username": username,
   "password": password,
  });
 }
```

```
  Future<bool> checkUser(String username, String password) async {
    var dbClient = await db;
    var res = await dbClient.rawQuery(
        "SELECT * FROM Users WHERE username=? AND password=?",
        [username, password]);
    return res.isNotEmpty;
  }
}
```

(register_screen.dart)

```
import 'package:flutter/material.dart';
import '../database/database_helper.dart';

class RegisterScreen extends StatelessWidget {
  final usernameController = TextEditingController();
  final passwordController = TextEditingController();
  final dbHelper = DatabaseHelper();

  RegisterScreen({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text("Register")),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          children: [
            TextField(
              controller: usernameController,
              decoration: const InputDecoration(labelText: "Username"),
            ),
            TextField(
              controller: passwordController,
              obscureText: true,
              decoration: const InputDecoration(labelText: "Password"),
            ),
            const SizedBox(height: 20),
            ElevatedButton(
              onPressed: () async {
                await dbHelper.saveUser(
                    usernameController.text, passwordController.text);
                ScaffoldMessenger.of(context).showSnackBar(const SnackBar(
                    content: Text("User Registered Successfully")));
                Navigator.pop(context);
              },
              child: const Text("Register"),
            ),
          ],
        ),
      ),
    );
  }
}
```

(login_screen.dart)

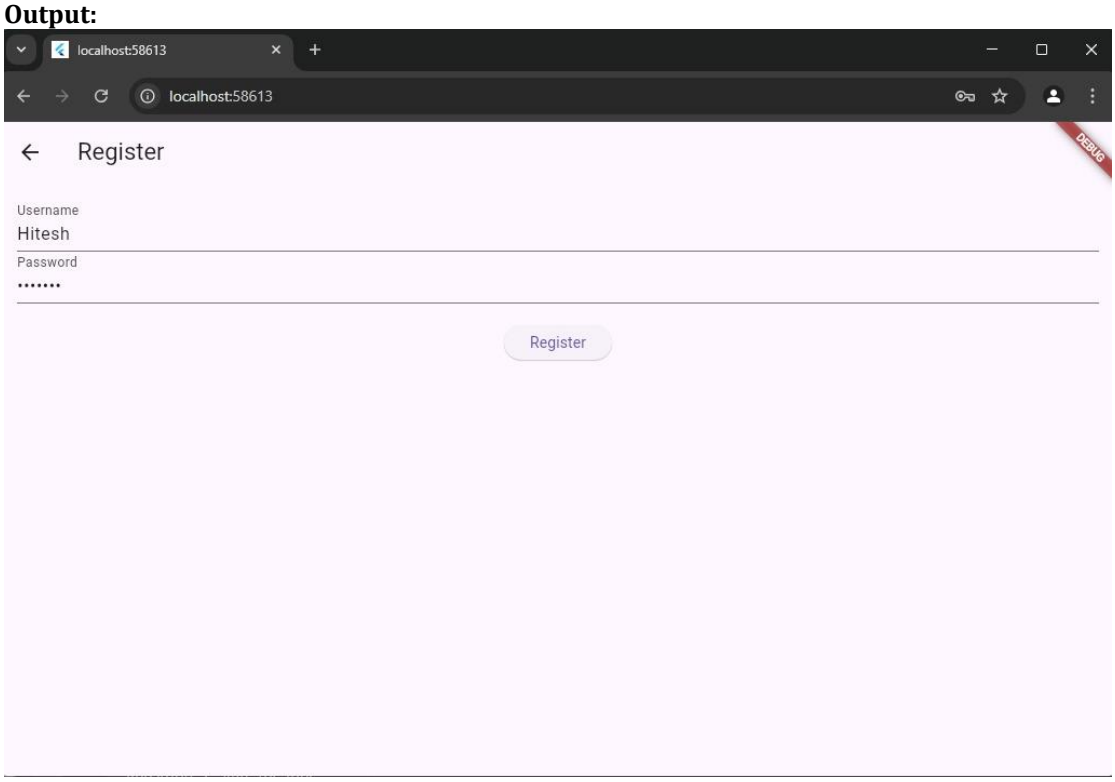
```
import 'package:flutter/material.dart';
import 'package:untitled/screens/home_screen.dart';
import 'package:untitled/screens/register_screen.dart';
import 'package:untitled/database/database_helper.dart';

class LoginScreen extends StatelessWidget {
  final usernameController = TextEditingController();
  final passwordController = TextEditingController();
  final dbHelper = DatabaseHelper();

  LoginScreen({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text("Login")),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          children: [
            TextField(
              controller: usernameController,
              decoration: const InputDecoration(labelText: "Username"),
            ),
            TextField(
              controller: passwordController,
              obscureText: true,
              decoration: const InputDecoration(labelText: "Password"),
            ),
            const SizedBox(height: 20),
            ElevatedButton(
              onPressed: () async {
                bool isValid = await dbHelper.checkUser(
                    usernameController.text, passwordController.text);
                if (isValid) {
                  Navigator.push(
                      context,
                      MaterialPageRoute(
                          builder: (context) => const HomeScreen()));
                } else {
                  ScaffoldMessenger.of(context).showSnackBar(
                      const SnackBar(content: Text("Invalid Credentials")));
                }
              },
              child: const Text("Login"),
            ),
            TextButton(
              onPressed: () {
                Navigator.push(
                    context,
                    MaterialPageRoute(
                        builder: (context) => RegisterScreen()));
              },
              child: const Text("Register"),
            ),
          ],
        ),
      ),
```

```
    );
  }
}
```

(home_screen.dart)

```
import 'package:flutter/material.dart';

class HomeScreen extends StatelessWidget {
  const HomeScreen({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text("Home")),
      body: const Center(
        child: Text("Welcome!"),
      ),
    );
  }
}
```

Output:



Name: Itisha Mishra
Roll No – CS23026 **ADVANCED APPLICATION DEVELOPMENT**

PRACTICAL NO. 9

Aim: Write a program to create an app using Flutter to demonstrate navigation in an App.

(Project Structure)

```
navigation_app/
|── lib/
|   |── main.dart                # Main entry point
|   |── screens/                 # Screens folder
|   |   |── home_screen.dart         # Home Screen
|   |   |── about_screen.dart        # About Screen
|   |   |── contact_screen.dart      # Contact Screen
|   |   |── settings_screen.dart     # Settings Screen
|── pubspec.yaml                # Dependencies and configurations
```

(Create a Flutter Project)

```
flutter create navigation_app
cd navigation_app
```

(main.dart)-Entry point

```
import 'package:flutter/material.dart';
import 'screens/home_screen.dart';
void main() {
 runApp(MyApp());
}
class MyApp extends StatelessWidget {
 const MyApp({super.key});
 @override
 Widget build(BuildContext context) {
  return MaterialApp(
   debugShowCheckedModeBanner: false,
   title: 'Flutter Navigation',
   theme: ThemeData(primarySwatch: Colors.blue),
   home: HomeScreen(), // Start at the HomeScreen
  );
 }
}
```

(home_screen.dart)-Home page

```
import 'package:flutter/material.dart';
import 'about_screen.dart';
import 'contact_screen.dart';
import 'settings_screen.dart';
class HomeScreen extends StatelessWidget {
 const HomeScreen({super.key});
 @override
 Widget build(BuildContext context) {
  return Scaffold(
   appBar: AppBar(title: Text("Home Screen")),
   body: Center(
    child: Column(
     mainAxisAlignment: MainAxisAlignment.center,
     children: [
      ElevatedButton(
       onPressed: () {
        Navigator.push(
         context,
         MaterialPageRoute(builder: (context) => AboutScreen()),
        );
       },
       child: Text("Go to About Screen"),
```

```
          ),
          ElevatedButton(
            onPressed: () {
              Navigator.push(
                context,
                MaterialPageRoute(builder: (context) => ContactScreen()),
              );
            },
            child: Text("Go to Contact Screen"),
          ),
          ElevatedButton(
            onPressed: () {
              Navigator.push(
                context,
                MaterialPageRoute(builder: (context) => SettingsScreen()),
              );
            },
            child: Text("Go to Settings"),
          ),
        ],
      ),
    ),
  );
 }
}
```

(about_screen.dart)-About page

```
import 'package:flutter/material.dart';
class AboutScreen extends StatelessWidget {
 const AboutScreen({super.key});

 @override
 Widget build(BuildContext context) {
  return Scaffold(
   appBar: AppBar(title: Text("About Screen")),
   body: Center(
    child: Column(
     mainAxisAlignment: MainAxisAlignment.center,
     children: [
      Text("This is the About Screen", style: TextStyle(fontSize: 20)),
      SizedBox(height: 20),
      ElevatedButton(
       onPressed: () {
        Navigator.pop(context); // Go back to Home
       },
       child: Text("Back to Home"),
      ),
     ],
    ),
   ),
  );
 }
}
```

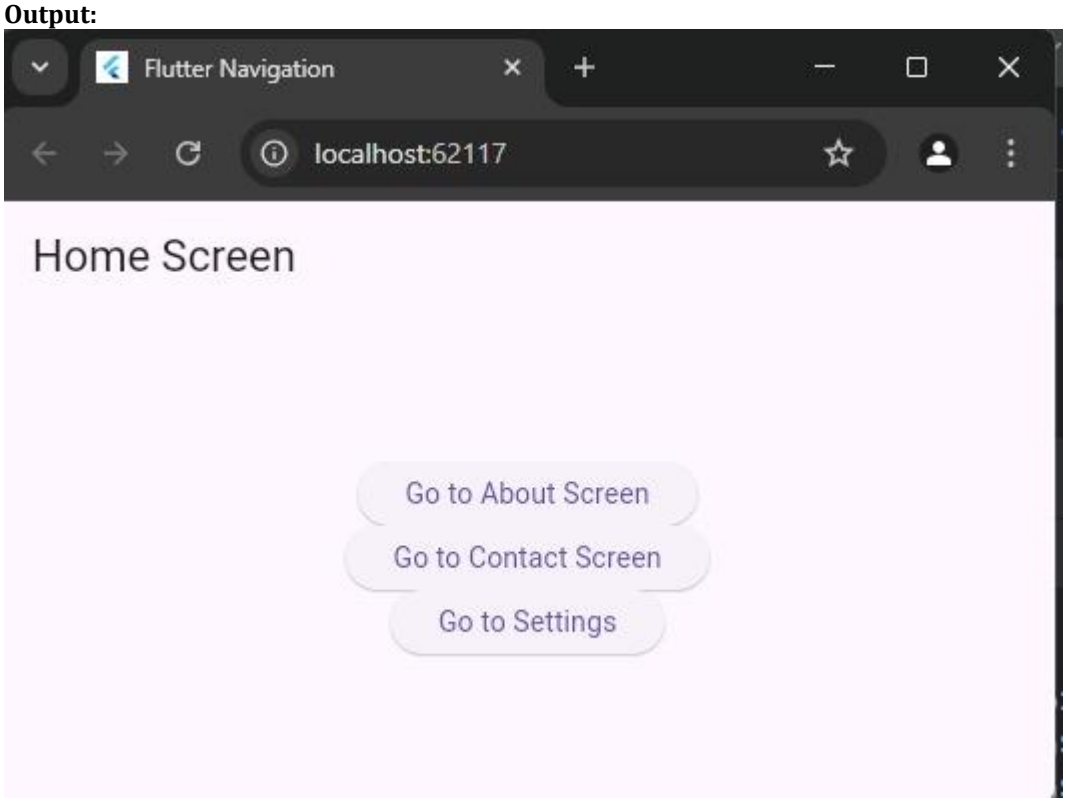
(contact_screen.dart)-Contact Page

```
import 'package:flutter/material.dart';
class ContactScreen extends StatelessWidget {
  const ContactScreen({super.key});
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text("Contact Screen")),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Text("This is the Contact Screen", style: TextStyle(fontSize: 20)),
            SizedBox(height: 20),
            ElevatedButton(
              onPressed: () {
                Navigator.pop(context); // Go back
              },
              child: Text("Back to Home"),
            ),
          ],
        ),
      ),
    );
  }
}
```

(settings_screen.dart)-Settings page

```
import 'package:flutter/material.dart';
class SettingsScreen extends StatelessWidget {
  const SettingsScreen({super.key});
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text("Settings Screen")),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Text("This is the Settings Screen", style: TextStyle(fontSize: 20)),
            SizedBox(height: 20),
            ElevatedButton(
              onPressed: () {
                Navigator.pop(context); // Go back
              },
              child: Text("Back to Home"),
            ),
          ],
        ),
      ),
    );
  }
}
```

Output:


Flutter Navigation
localhost:62117
Home Screen
Go to About Screen
Go to Contact Screen
Go to Settings