Product Documentation and Setup

Frontend (Angular)

Shreyas Phatak
+91 883001737

Table of Contents

JDOC

- Project Overview
  - Description

    An Angular-based frontend application to handle user interactions with the backend services (mock server) and document management.

  - Features

    The application includes key features as follows

    - Login
    - Sign Up

        - Logout
        - User Management
        - Document Management
        - Profile

    - Technologies

      The application is developed on Angular CLI version 14.0.7 along with additional libraries such as RxJS, Bootstrap, Chart.js, and angular material

- Installation Guide

    - Prerequisites

      To run the application, ensure the Node.js and Angular CLI are installed prior.

    - Repository

      The project can be downloaded from the link

- Project Structure

  The Angular project follows a modular structure that supports scalability and maintainability. Below is an overview of the folder structure:

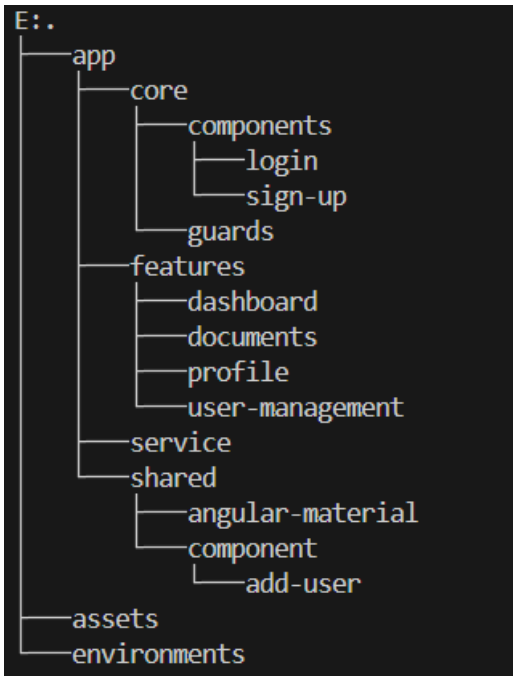


  Figure: Tree Structur

Setup and Configuration

This section explains the setup process for local and production environments, the configuration files required for Angular to run correctly, and any API/database configuration settings.

- Environment Setup

    - Clone the Repository: Clone the project

      from the Git repository: git clone
    - Install Dependencies: Navigate to the project directory and run the following command to install the necessary dependencies:

      npm install

    - Run the Development Server: To run the application locally, use the Angular CLI command:

      ng serve

      By default, the application will run on http://localhost:4200.

    - Local Environment Configuration: The local environment is configured using src/environments/environment.ts. This file typically contains configuration settings specific to the local setup, such as API endpoints for development.

- Code Architecture

  The architecture of this Angular application follows best practices for modularity, scalability, and maintainability. The project is organized using Angular's standard module, component, service, routing concepts and reusable components, and optimization.

    - Modular Structure

      The application is divided into multiple modules to encapsulate features and functionality. Each feature of the app is represented by a dedicated Angular module (also known as feature module), which organizes the related components, services, and other code into
      a cohesive unit. The goal is to create a maintainable and scalable project by organizing features into separate modules.

        - App Module (app.module.ts): This is the root module that bootstraps the application and imports other essential

modules (e.g., routing module, core module).

- Feature Modules: These represent different sections or features of the app (e.g., CoreModule). They help in lazy loading and reducing the initial bundle size.
- Shared Module: This module holds reusable components, pipes, directives, and services that can be used throughout the application.

- Components and Templates

  Components are the building blocks of an Angular application. Each feature module contains one or more components, which are responsible for rendering the user interface and handling user interactions.

- Services and Dependency Injection

  Angular's services handle logic and business rules that are not directly related to UI rendering. Services are typically used for data handling, interacting with APIs, or managing application state. Angular uses dependency injection (DI) to provide instances of services to components, which makes the app more testable and modular.

  - Core Services: These services manage critical application-wide functionality, like authentication, API communication, and user settings.
  - Feature Services: Each feature module may have its own set of services for handling domain-specific logic.

- Routing

  Angular uses the Router to navigate between different views or pages within the application. The routing configuration is typically handled by a dedicated module, often named AppRoutingModule, which defines the routes and links them to corresponding components.

- Reusable Components and Shared Module

  Common components (add user form) are placed in the Shared Module to promote reusability and maintainability. This module is imported by feature modules to avoid duplicating code and ensure that all reusable components are available throughout the app.

- Testing

This section outlines the Angular application's testing strategy, the tools and libraries employed, and guidelines for writing and running tests.

- Testing Frameworks

  Angular provides built-in support for unit and end-to-end testing using the following tools:

  - Unit Testing:

    - Jasmine: The testing framework used for writing unit tests in Angular. Jasmine provides a syntax for defining tests and assertions.
    - Karma: The test runner used to execute Jasmine tests in a real browser environment.
    - TestBed: Angular's testing utility that allows you to configure and initialize an Angular environment for unit testing.

- Writing Tests

  Tests in Angular are typically organized into the following categories:

  - Unit Tests: Unit tests verify individual components, services, and other logic in isolation. These tests are written using Jasmine and executed with Karma.
  - Component Tests: Components are tested to ensure they correctly render templates, handle events, and manage input/output properties.

- Running Tests

  - Run Unit Tests with Karma: The ng test command runs unit tests in the browser using Karma and Jasmine.

    ng test

    This command will open a browser window and execute all unit tests defined in the project. Once the tests are complete, the results will be displayed in the terminal.

  - Continuous Testing: During development, Karma watches for file changes and reruns the tests automatically. It allows developers to get instant feedback on code changes.

- Test Coverage

  Testing coverage ensures that all parts of the code are being tested adequately. The following tools can be used for measuring and ensuring code coverage:

- Code Coverage with Karma: To enable code coverage reporting, run,

  ng test --code-coverage

  This command will generate a code coverage report in the /coverage folder in your project. The report includes detailed information on which lines of code have been executed and which haven't.

  To install karma and jasmine and show the coverage, run,

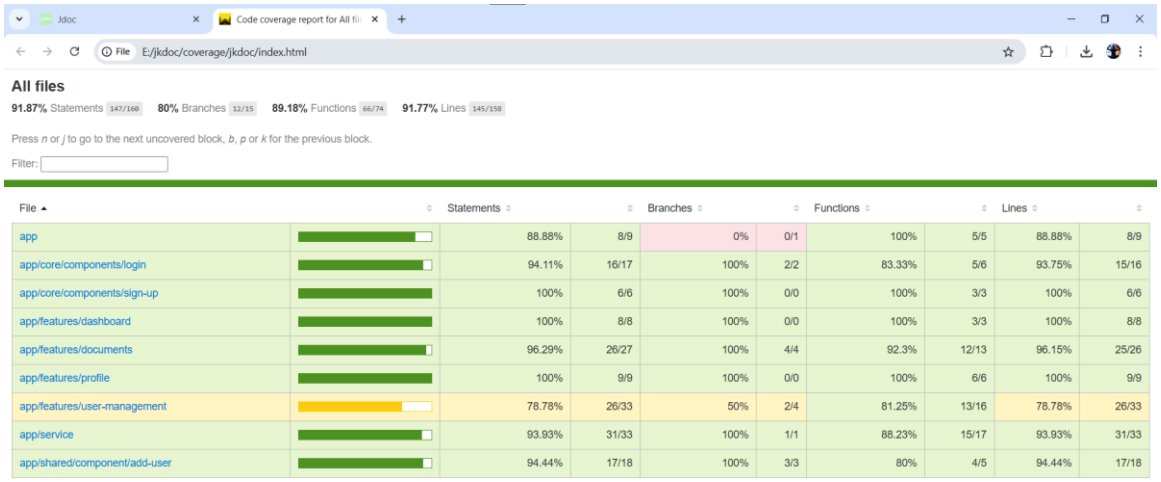
  npm install karma karma-chrome-launcher karma-jasmin

  npm install karma-coverage --save-dev

  Once the command is successful, browse the index.html to view the code coverage which is available at the following path,

  project-name/coverage/docu-mint/index.html

  Unit testing has been done and the current code coverage is shown in the picture below,



File	Statements		Branches		Functions		Lines	
app	88.88%	8/9	0%	0/1	100%	5/5	88.88%	8/9
app/core/components/login	94.11%	16/17	100%	2/2	83.33%	5/6	93.75%	15/16
app/core/components/sign-up	100%	6/6	100%	0/0	100%	3/3	100%	6/6
app/features/dashboard	100%	8/8	100%	0/0	100%	3/3	100%	8/8
app/features/documents	96.29%	26/27	100%	4/4	92.3%	12/13	96.15%	25/26
app/features/profile	100%	9/9	100%	0/0	100%	6/6	100%	9/9
app/features/user-management	78.78%	26/33	50%	2/4	81.25%	13/16	78.78%	26/33
app/service	93.93%	31/33	100%	1/1	88.23%	15/17	93.93%	31/33
app/shared/component/add-user	94.44%	17/18	100%	3/3	80%	4/5	94.44%	17/18

Figure: Code coverage

- API Documentation

  As we are not directly dealing with the backend or REST APIs, we use JSON

Server for mocking the API methods.

To install the JSON Server use the following command,
npm run json-server

This should enable the server to run at http://localhost:3000

- Screenshots

  This section provides some of the screenshots of the application.

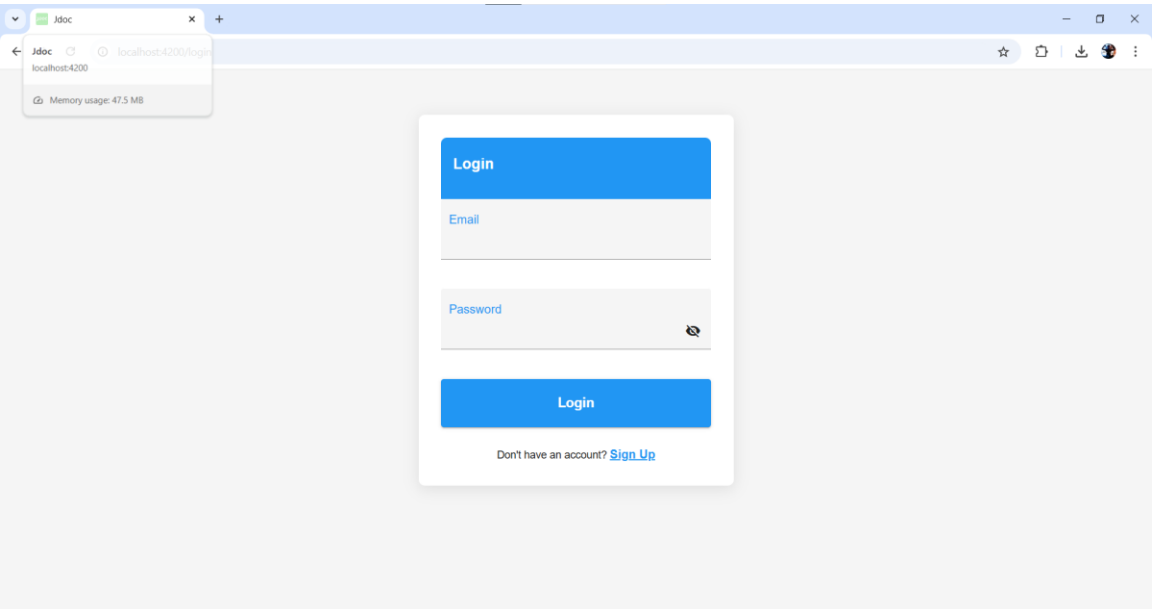


Figure: Login Screen

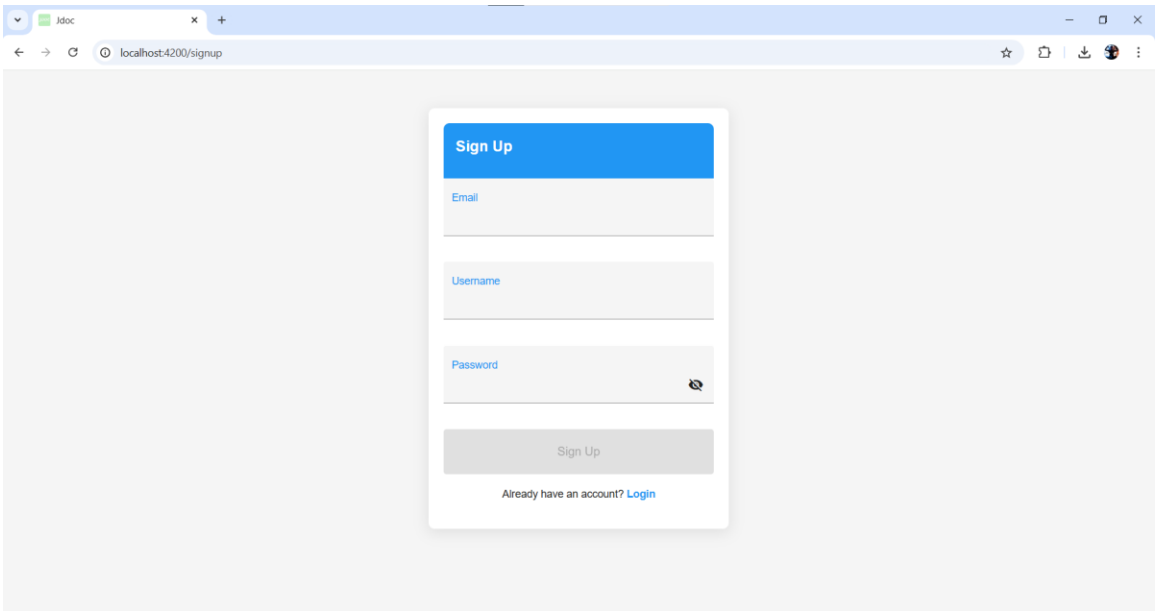


Figure: Sign Up Screen

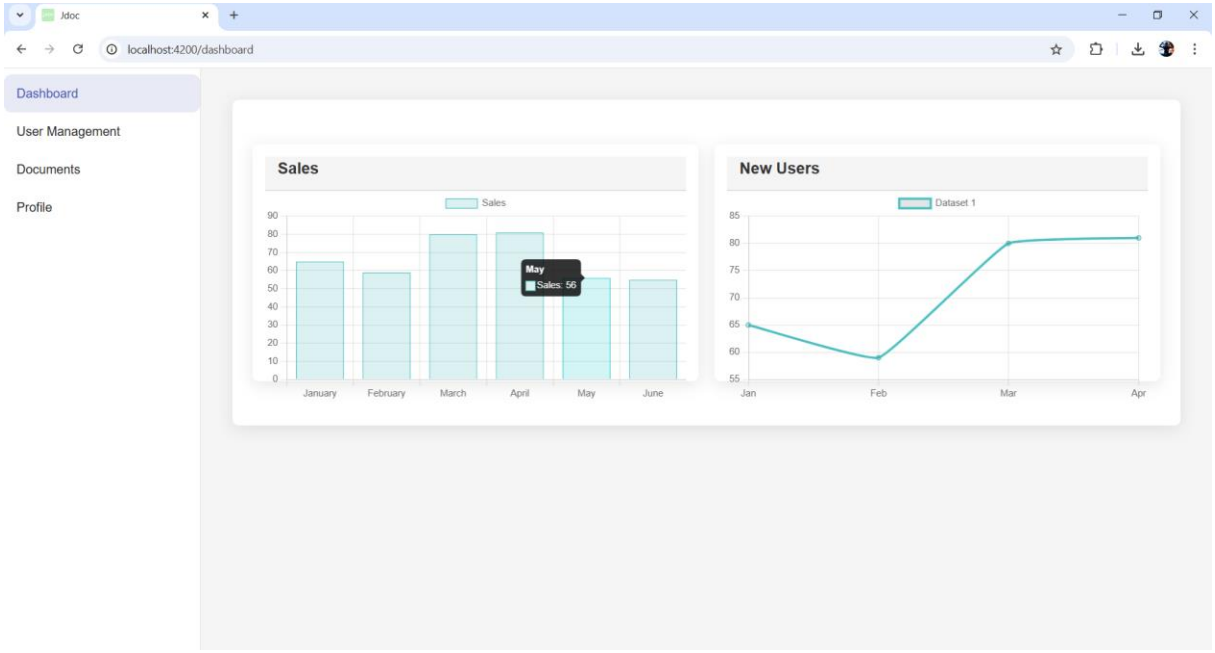


Figure: Dashboard Screen

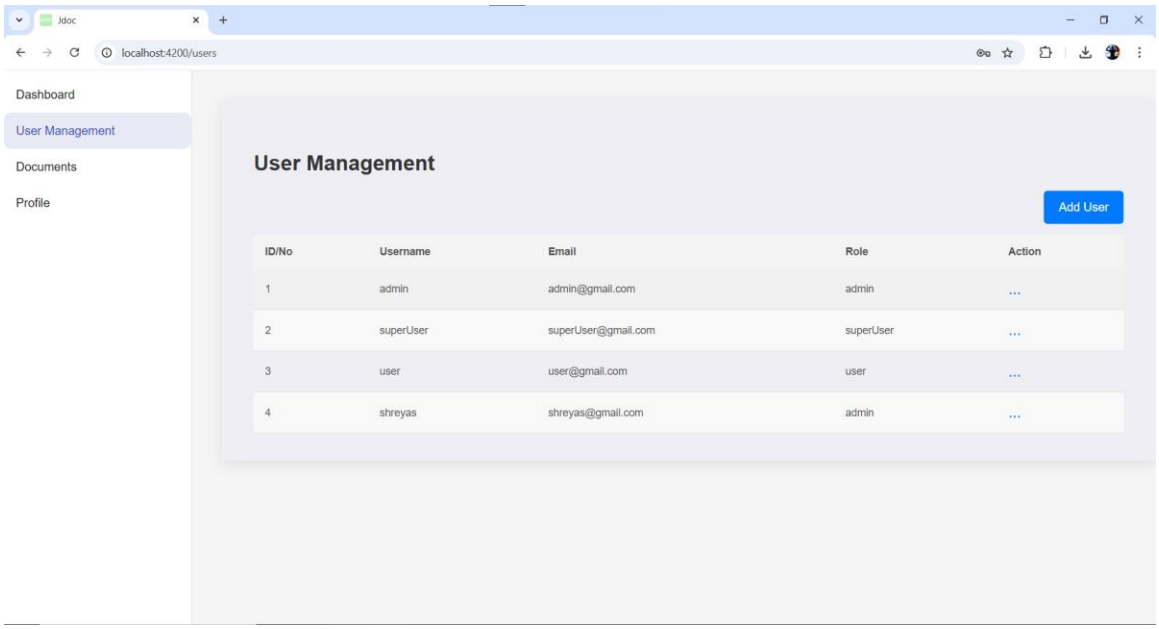


Figure: User List Screen

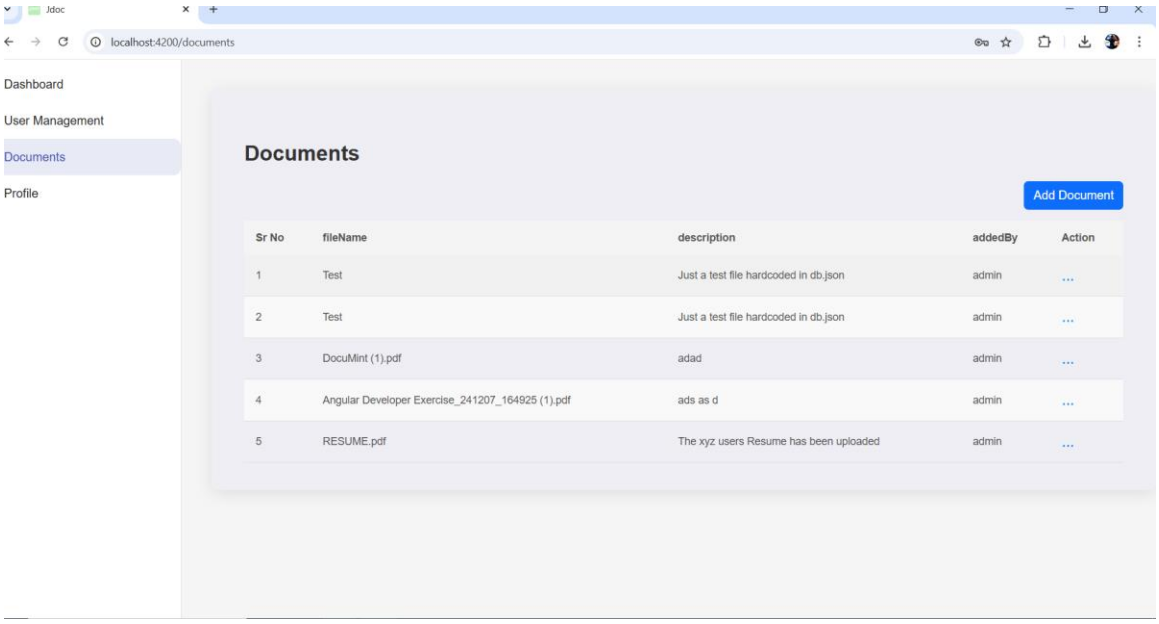


Figure: Document List Screen

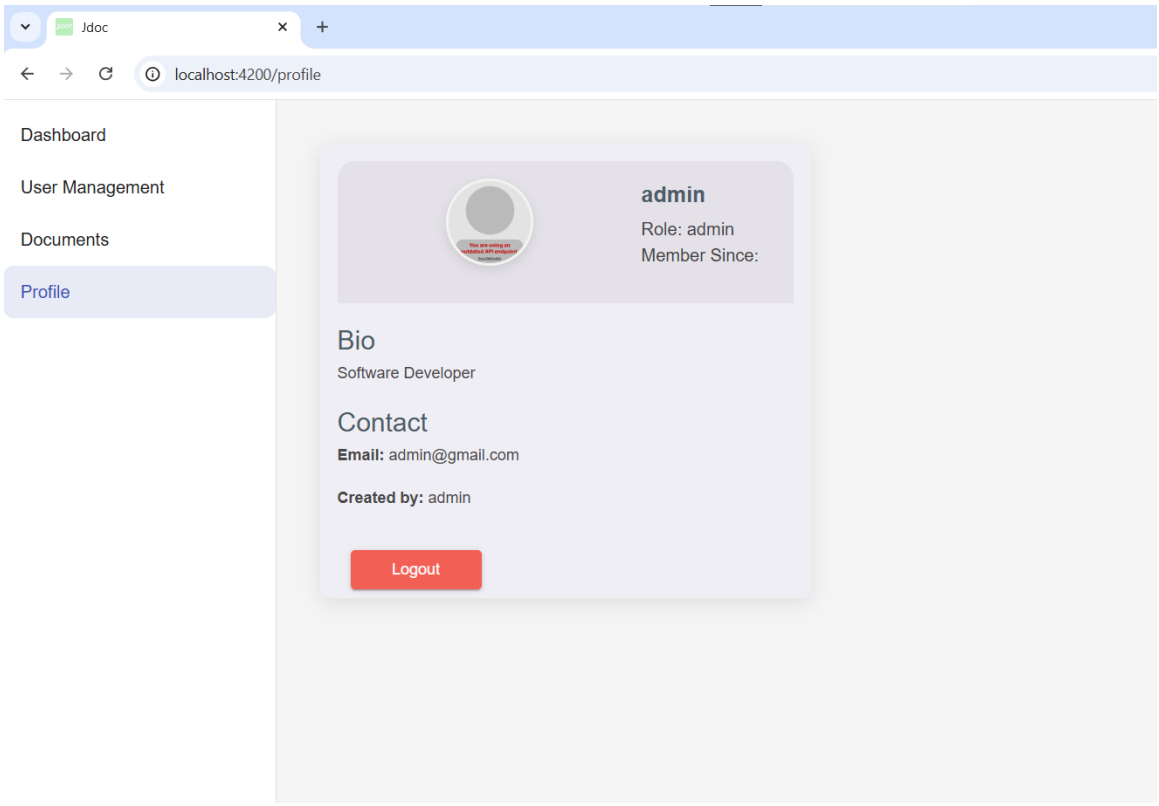


Figure: User Profile Screen