

StackOverflow App

DOCUMENTATION

Software Design

STUDENT: DOMȘA-STÎNĂ ANDA

GROUP: 30433

# TABLE OF CONTENTS

I. Back-End:

1.  [Introduction………………………………..................................................................................3](file:///C:\Users\User\Downloads\PT2021-2022_Documentation_Template.doc#_Toc95297885)

[2. Software Architecture…………………………..……...………………...……………………..4](file:///C:\Users\User\Downloads\PT2021-2022_Documentation_Template.doc#_Toc95297886)

[3. Requirements 11](file:///C:\Users\User\Downloads\PT2021-2022_Documentation_Template.doc#_Toc95297888)

[4. Design 14](file:///C:\Users\User\Downloads\PT2021-2022_Documentation_Template.doc#_Toc95297890)

II. Front-End:

1. Diagrams………………………………….……………………………………………………………………………………………15

2. Architecture………………………………………………………………………………………………………………………….15

3. Routing …………………………………………………………………………………………………………………………………15

III. Connection

[IV. Conclusions 15](file:///C:\Users\User\Downloads\PT2021-2022_Documentation_Template.doc#_Toc95297891)

[V. Bibliography 15](file:///C:\Users\User\Downloads\PT2021-2022_Documentation_Template.doc#_Toc95297891)

## **Back-End:**

1. *Introduction*

## **Front-End:**

The frontend of the project is developed using Angular, a powerful JavaScript framework for building dynamic and responsive web applications.

It consists of various components, each responsible for a specific part of the user interface and functionality. These components are designed and organized to deliver a seamless user experience. Some key components of the frontend are:

* Welcome Page: This component serves as the landing page for the Stack Overflow app. It provides a welcoming message, introduces the app's purpose and features, and encourages users to register or log in.
* Register Page: This component enables new users to create an account on the Stack Overflow app. It includes input fields for the email, desired username, and password. It also includes validation logic to ensure the entered information meets the necessary criteria. Once the user submits the registration form, the app can authenticate the user and redirect them to the main page or send a verification email to complete the registration process.
* Login Page: This page allows users to log in or register for a new account. It provides input fields for entering credentials, authentication logic to verify user information, and error handling for invalid login attempts.
* Main Questions Page: The main page serves as the central hub of the app, displaying featured properties, search filters, and navigation options. It provides a visually appealing and intuitive layout to showcase available accommodation and attract user attention.
* User Info Page: This page allows users to view and manage their personal information, such as name, email, and profile picture. It provides options to update their details, manage bookings, and view past activity.
* Question Details: This component shows the detailed information of a specific question. It includes the question title, description, author details, answers, and comments. Users can read and interact with the question, leave answers or comments, and upvote/downvote helpful content.
* Ask a Question Form: This component provides a form where users can ask a new question. It includes input fields for the question title, description, and tags. Users can submit their questions, and upon submission, the question is added to the Stack Overflow database for others to answer.

1. *Diagrams:*
   * Component Diagram

A component diagram is a visual representation in UML (Unified Modeling Language) that illustrates the components, interfaces, and dependencies of a software system. It provides a high-level view of the system's architecture and the relationships between its components.

In the context of a web application, a component diagram depicts the various components that make up the application, such as web servers, application servers, databases, and other software components. Each component is represented as a box or a rectangle, and the connections between components are depicted as lines or arrows that represent dependencies.

The purpose of a component diagram is to show how the different components interact and depend on each other to achieve the desired functionality of the system. It helps to visualize the structure of the software system, identify the key components, and understand their relationships and dependencies. This diagram is particularly useful for understanding the overall architecture of the system and facilitating communication among stakeholders involved in the development and maintenance of the software application.

A screenshot of a computer

Description automatically generated

* + Deployment Diagram

A deployment diagram is a UML diagram that represents the physical deployment of software components in a system, including hardware and network infrastructure. It provides a visual representation of the system's deployment architecture, showcasing the distribution and arrangement of components across different nodes.

In the context of a web application, a deployment diagram illustrates the nodes involved in supporting the application. These nodes typically include the web server, application server, database server, and any other servers or resources required for the application to function properly. Each node is depicted as a box or a rectangle, and the connections between nodes represent the communication paths or network links.

The purpose of a deployment diagram is to visualize how the software components are deployed on hardware infrastructure, such as servers or cloud-based resources, and how they interact with each other. It helps to understand the physical distribution of the system, identify potential points of failure or bottlenecks, and ensure that the necessary resources are appropriately allocated for the application to run effectively.

A screenshot of a computer

Description automatically generated

*Architecture:*

The front-end architecture of our Stack Overflow application follows a component-based approach. Each component is responsible for a specific part of the user interface and functionality, ensuring modularity and reusability.

## ***Component Hierarchy:***

The following components form the core of the Stack Overflow front-end:

* + Welcome Page
  + Register Page
  + Login Page
  + Main Question Page
  + Question Page
  + Ask Question Page
  + User Profile Page

The front-end architecture of the Stack Overflow application is designed to deliver a responsive and intuitive user interface. The component-based approach ensures modularity, reusability, and maintainability of the codebase.

A screenshot of a computer

Description automatically generated

1. *Routing:*

The components in my front-end architecture interact with each other through various mechanisms:

* + Routing: Angular's routing module is used to define the routes and navigation between different pages. Each component is associated with a specific route, allowing users to navigate seamlessly within the application.
  + Data Binding: Angular's data binding mechanism enables components to exchange data and reflect changes in real-time. Components can bind to properties or methods of other components to display dynamic content or trigger actions.
  + Service Communication: Services are used to handle data retrieval, API communication, and business logic. Components can interact with services to fetch or update data, enabling seamless integration with the back-end and providing a consistent user experience.

**Routing** plays a crucial role in the front-end architecture of my Stack Overflow application. It enables navigation between different pages and provides a seamless user experience. Let's delve into the routing implementation and its significance:

## ***Routing Configuration:***

In my Angular application, routing is configured using the Angular Router module. The routing configuration is defined in a separate module called AppRoutingModule, which is imported into the main AppModule.

The AppRoutingModule consists of route definitions that map specific URLs to corresponding components. For example, the following routes can be defined:

* welcome: Maps to the WelcomePageComponent, which serves as the landing page.
* register: Maps to the RegisterPageComponent, where users can create a new account.

These routes are defined using the RouterModule provided by Angular. Each route includes a path that represents the URL segment and a component that is associated with the path. Additional properties such as canActivate can be used to implement route guards for authentication and authorization.

## ***Router Outlet:***

To display the appropriate component based on the current URL, I use the <router-outlet> directive in the main template file (app.component.html). The <router-outlet> acts as a placeholder where Angular dynamically renders the component associated with the current route.

## ***Navigation:***

Navigation between pages is achieved using the Router service provided by the Angular Router module. The Router service allows me to programmatically navigate to a specific route using methods such as navigate(). For example, after a successful login, I can navigate the user to the main questions page by calling router.navigate(['/questions']).

Additionally, Angular Router provides features like route parameters, query parameters, and route guards, which allow for more advanced routing scenarios. These features enable me to pass data between components, handle dynamic URLs, and enforce access control rules.

Overall, routing in my Stack Overflow application ensures that users can navigate between different pages, access the desired content, and interact with the application seamlessly. It enhances the user experience and provides a structured and organized flow within the application.