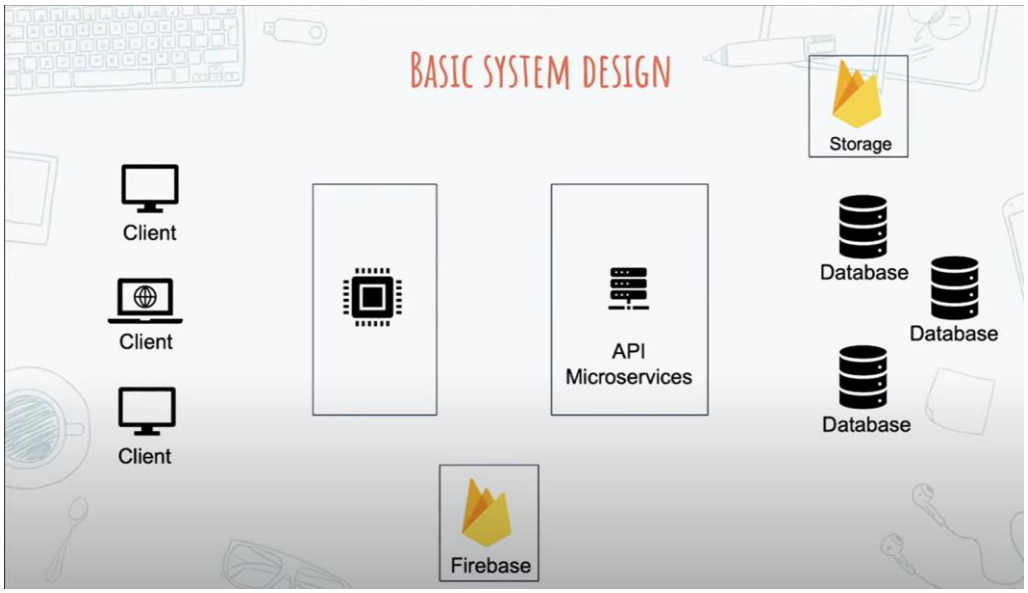
Social Media Application

Project Description :

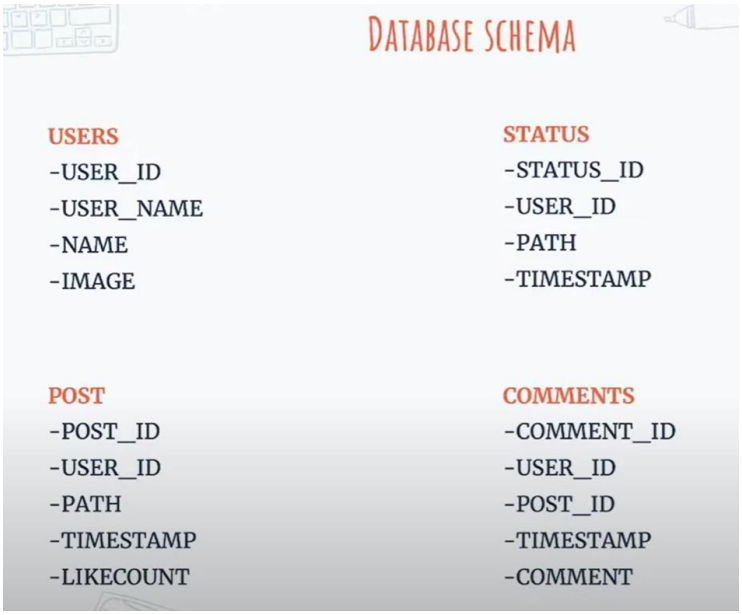
The goal of this project is to develop a social media app that provides users with a platform to connect, communicate, and share content with others. The app aims to create an engaging and user-friendly experience while incorporating innovative features to enhance social interactions. This project involves developing a user-friendly social media app that enables people to connect, communicate, and share content. Key features include user registration, customizable profiles, a dynamic news feed, content sharing (text, photos, videos, links), social interactions

Project Structure:

1. Front end:



2. Back end:



Pre-requisites :

Here are the prerequisites for a Java Spring Boot project with MySQL and MongoDB, along with relevant links for download and installation:

1-Java Development Kit (JDK):JDK is required to compile and run Java applications, providing the necessary tools and libraries. Download and install the latest JDK version from Oracle's website.

- Download JDK: https://www.oracle.com/java/technologies/javase-jdk11-downloads.html

2-Integrated Development Environment (IDE): An IDE offers a comprehensive development environment for writing, debugging, and managing code. IntelliJ IDEA, Eclipse, or Visual Studio Code are popular choices for Java development.

- IntelliJ IDEA: https://www.jetbrains.com/idea/download/
- Eclipse: https://www.eclipse.org/downloads/
- Visual Studio Code: https://code.visualstudio.com/download

3-Spring Boot: Spring Boot simplifies Java application development by providing predefined configurations, automatic dependency management, and a streamlined development experience. Use Spring Initializr or Spring Tools for your IDE to create a Spring Boot project.

- Spring Initializr (Online): https://start.spring.io/
- Spring Tools 4 for Eclipse: https://spring.io/tools

- Spring Tools for Visual Studio Code: Install via Extensions in Visual Studio Code

4-MySQL Database: MySQL is a popular relational database management system. Install MySQL Community Server and optionally MySQL Workbench, a graphical tool for managing MySQL databases.

- MySQL Community Server: https://dev.mysql.com/downloads/installer/
- MySQL Workbench: https://dev.mysql.com/downloads/workbench/

5-MySQL Connector/J: MySQL Connector/J is the official JDBC driver for connecting Java applications to MySQL databases. Include this dependency in your project to enable connectivity and interaction with MySQL.

- Maven:
  - Add the following dependency to your project's pom.xml:

    xml

    ```
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>8.0.27</version>
    </dependency>
    ```

  - Maven Repository: https://mvnrepository.com/artifact/mysql/mysql-connector-java

- **Gradle:**
  - Add the following dependency to your project's build.gradle:

    ```
    implementation 'mysql:mysql-connector-java:8.0.27'
    ```

    - Gradle Repository: https://search.maven.org/artifact/mysql/mysql-connector-java/8.0.27/jar

6-Apache Cassandra: Apache Cassandra is a highly scalable and distributed NoSQL database system designed for handling massive amounts of data across multiple commodity servers, providing high availability and fault tolerance. It was initially developed by Facebook and later open-sourced as an Apache Software Foundation project.

- Apache Cassandra Community Edition:
https://cassandra.apache.org/_/download.html

7-Apache Cassandra Java Driver: Apache Cassandra is a highly scalable and distributed NoSQL database known for its ability to handle large amounts of data across multiple commodity servers while maintaining high availability and fault tolerance. When integrating Cassandra into a Spring Boot project, you'll need to include some dependencies to interact with the database.

To include Cassandra dependency in a Spring Boot project, you need to add the following Maven/Gradle dependency in your project's build file:

For Maven:

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-cassandra</artifactId>
</dependency>
```

This dependency will bring in all the necessary classes and configurations to interact with Cassandra from your Spring Boot application.

Once you have included the dependency, you need to configure the connection to your Cassandra cluster in the application.properties (or application.yml) file.

- Gradle:

```
implementation 'org.springframework.boot:spring-boot-starter-data-cassandra'
```

Role Based Access:-

1-Admin Role:

- **Responsibilities:** Overall, the admin's role in a social media app encompasses user management, content moderation, community management, security, technical maintenance, user support, policy enforcement, and collaboration with legal authorities to ensure a safe, engaging, and compliant social media environment.
- **Permissions:**

1. User Management: Admins are responsible for managing user accounts, including user registration, authentication, and account verification processes. They handle user data, such as usernames, passwords, and profile information, and assist users in case of account-related issues or concerns.

2. Content Moderation: Admins play a crucial role in maintaining the quality and appropriateness of content shared on the social media app. They review user-generated content, including posts, comments, and media files, to ensure they adhere to community guidelines, terms of service, and legal requirements. Admins have the authority to remove or flag inappropriate or violating content, spam, or abusive behavior.
3. Community Management: Admins foster a positive and inclusive community environment within the app. They encourage respectful interactions among users and address disputes or conflicts that may arise. Admins may engage in community engagement initiatives, such as highlighting notable user content, organizing contests, or addressing user feedback and suggestions.
4. Security and Privacy: Admins oversee the implementation of security measures to protect user data and privacy. They monitor the app for potential security vulnerabilities, implement security protocols, and respond to security incidents promptly. Admins also handle user privacy-related concerns and ensure compliance with relevant data protection regulations.
5. Technical Maintenance: Admins manage the technical aspects of the social media app, including server maintenance, software updates, and bug fixes. They work closely with the development team to ensure the app's performance, reliability, and scalability. Admins also monitor server loads, optimize performance, and troubleshoot technical issues to provide a seamless user experience.
6. User Support: Admins provide user support by addressing queries, concerns, and complaints submitted by users through various channels, such as in-app support tickets, emails, or community forums. They assist users with account-related issues, content removal appeals, and general inquiries to ensure a positive user experience.
7. Policy Enforcement: Admins enforce the app's terms of service, community guidelines, and content policies. They handle user reports of policy violations, investigate reported content or accounts, and take appropriate actions, such as warnings, content removal, or account suspension/banning when necessary.
8. Collaboration with Legal Authorities: In case of legal requests or investigations, admins may collaborate with legal authorities, providing user data or information as required by law, while ensuring user privacy and complying with legal procedures.

2-User Role:

- **Responsibilities:** The user's role in a social media app involves creating and customizing their account, connecting with others, sharing content, engaging with others' content, exploring and discovering new accounts and trends, private messaging, managing settings and notifications, developing social influence, reporting

issues, and providing feedback to contribute to the overall user experience on the platform.

- **Permissions:**

1. Account Creation: Users begin by creating an account on the social media app. They provide necessary information such as name, username, and profile picture, which helps identify and personalize their presence on the platform.
2. Profile Customization: Users have the ability to customize their profiles by adding a bio, additional personal details, and choosing privacy settings. They can also upload profile pictures and cover photos to represent themselves visually.
3. Connecting with Others: Users can connect with friends, family, colleagues, and people with shared interests on the social media app. They can search for specific users, import contacts, or discover new connections through mutual friends or suggested accounts.
4. Content Sharing: Users can share various types of content, including text posts, photos, videos, links, and more. They can express their thoughts, share experiences, and showcase their creativity through the content they post on the platform.
5. Engagement and Interaction: Users can interact with others' content through likes, comments, and shares. They can express their appreciation, provide feedback, or engage in discussions with fellow users. This interaction helps foster connections and build relationships within the social media community.
6. Discovery and Exploration: Users can explore content shared by other users, discover new accounts, and engage with trending topics or hashtags. They can follow accounts of interest to stay updated on their content and discover new perspectives, ideas, and trends.
7. Private Messaging: Users can engage in one-on-one conversations with other users through private messaging features. This allows for more personal and direct communication within the social media app.

Project Flow:

- Frontend Development
- Backend Development
- Integration
- Containerization of the Application
- Deployment to Kubernetes Cluster

Milestone 1: Frontend Development:

Frontend development involves building the user interface (UI) and implementing the visual elements of the social media application. It focuses on creating an intuitive and engaging user experience that allows users to interact with the application seamlessly. The following activities are part of the frontend development process:

Activity1: User Registration and Login:

Users are presented with a registration form to create a new account. They provide necessary details such as name, email, and password.

After registration, users can log in using their credentials on subsequent visits.

Activity2: User Profile:

Once logged in, users can access and customize their profile. They can upload a profile picture, provide a bio, and update personal information.

Users may have the option to adjust privacy settings to control who can view their profile and content.

Activity3: News Feed:

Users are presented with a dynamic news feed as the main screen of the app.

The news feed displays posts from users and accounts that the user follows, as well as recommended content based on their interests.

The feed may be organized chronologically or algorithmically to prioritize relevant and engaging content.

Activity4: Content Creation and Sharing:

Users can create and share different types of content, such as text posts, photos, videos, or links.

The app provides a content creation interface, allowing users to compose their posts, add captions, attach media files, and include relevant tags or location information.

Once created, users can choose to publish their posts, making them visible to their followers and the wider community.

Activity5: Social Interactions:

Users can engage with posts through various actions like liking, commenting, and sharing.

They can view and leave comments on posts, and the app may support threaded discussions for better conversation organization.

Users can like or react to posts to show appreciation or express emotions.

Activity6: Explore and Discover:

The app provides a dedicated section for users to explore new content and discover accounts based on their interests.

Users can browse trending topics, explore popular hashtags, or discover recommended accounts.

The explore section may use algorithms to suggest personalized content and accounts based on user preferences and interactions.

Activity7: Messaging and Notifications:

Users can access a messaging feature to have private one-on-one conversations with other users.

The app sends notifications to users for activities such as new followers, likes, comments, or messages.

Notifications keep users updated about interactions and ensure they don't miss important updates from their network.

Activity7: Settings and Privacy:

Users have access to settings where they can manage their account preferences, notification settings, and privacy controls.

They can customize their news feed preferences, manage followers, and adjust privacy settings for their profile and content visibility.

Responsive Design:

The front end of the social media app is designed to be responsive, adapting to different screen sizes and device types.

The app ensures a seamless user experience across platforms, including desktop, mobile devices, and tablets.

Activity8: Iterative Development:

The front-end development follows an iterative approach, with continuous improvements and updates based on user feedback and emerging trends.

New features and enhancements are introduced in subsequent releases to enhance the user experience and address user needs.

Milestone 2: Backend Development:

Backend development involves building the server-side components and logic of the social media application. It focuses on handling the business logic, processing requests from the frontend, and interacting with the database. The following activities are part of the backend development process:

Activity1: User Authentication and Authorization:

The back end handles user authentication, including registration, login, and password management.

Upon successful authentication, the back end generates and issues authentication tokens (such as JWT) to the client for subsequent API requests.

Authorization mechanisms are implemented to control access to different resources and ensure users can only perform actions allowed by their role or permissions.

Activity2: User Profile and Account Management:

The back end provides APIs for managing user profiles, allowing users to update their profile information, upload profile pictures, and adjust privacy settings.

User account-related actions, such as email verification, password reset, and account deletion, are handled securely and with proper validation.

Activity3: Messaging and Notifications:

APIs handle messaging functionality, allowing users to send and receive private messages.

Notifications for activities such as new followers, likes, comments, and messages are generated by the back end and delivered to users via appropriate channels (e.g., push notifications, emails).

Search and Discovery:

The back end implements search functionality, allowing users to search for specific content, users, or hashtags within the app.

Algorithms may be employed to provide recommendations and discover new content or accounts based on user preferences and interactions.

Milestone 3: Integration:

Integration is the process of combining and connecting the frontend and backend components of the doctor booking application to create a unified and fully functional system. It involves establishing communication channels, exchanging data, and ensuring seamless interaction between the frontend UI and backend APIs. The following activities are part of the integration process:

Database Management:

The back end manages the database to store and retrieve user data, posts, comments, and other relevant information.

Database schemas are designed to efficiently organize and query data, considering relationships between entities (e.g., users, posts, comments) and performance optimization techniques.

Activity2: Content Management:

APIs are implemented to handle content creation and management, allowing users to create and share posts, upload media files, and attach additional information such as captions, tags, and location data.

The back end processes and stores the content, ensuring proper validation, sanitization, and file storage management.

Milestone 4: Containerization of the Application

Activity1: Containerization Framework Selection:

Choose a containerization framework like Docker or Kubernetes to containerize the social media app. Docker is commonly used for creating and managing containers, while Kubernetes provides orchestration capabilities for containerized applications.

Activity2: Container Definition:

Create a Dockerfile or Kubernetes manifest file that defines the specifications and dependencies of the social media app. This includes the base image, required libraries, environment variables, and application code.

Activity3: Container Build:

Use the Dockerfile or Kubernetes manifest file to build the container image. The image encapsulates the entire application, including the code, dependencies, and runtime environment. This can be done using commands like docker build or kubectl apply.

Activity4: Container Registry:

Store the built container image in a container registry like Docker Hub or a private registry. This allows for easy retrieval and distribution of the container image across different environments.

Activity5: Container Deployment:

Deploy the container image to the desired environment, such as a development, staging, or production environment. This can be done using tools like Docker Swarm or Kubernetes clusters.

Continuous Integration/Continuous Deployment (CI/CD) Integration: Integrate the containerization process into your CI/CD pipeline. This ensures that changes to the social media app's codebase trigger automated container builds, tests, and deployments.

Milestone 5: Deployment to Kubernetes Cluster

Kubernetes provides a platform for automating the deployment, scaling, and management of containerized applications. Deploying the application to a Kubernetes cluster enables efficient orchestration and scalability.

Activity1: Container Orchestration:

If using Kubernetes, leverage its orchestration capabilities to manage the containerized social media app. This includes features like scaling, load balancing, health monitoring, and rolling updates.

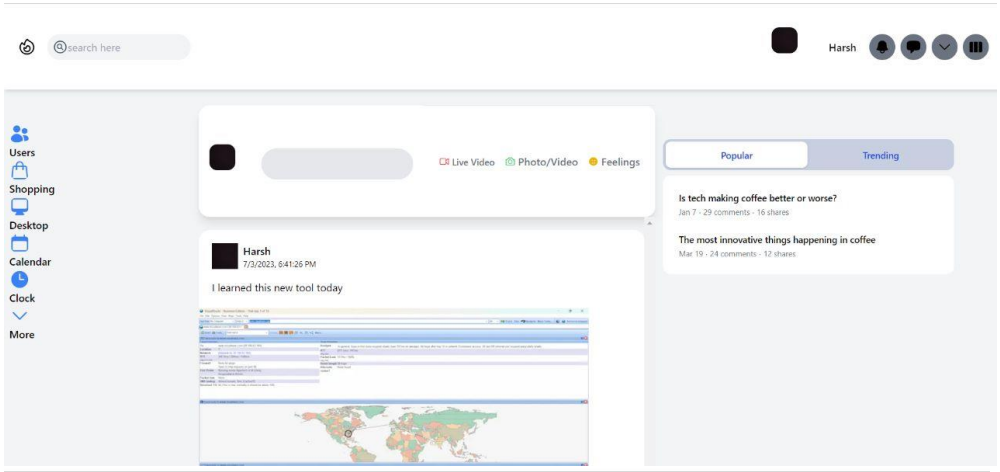
Activity2: Scalability and High Availability:

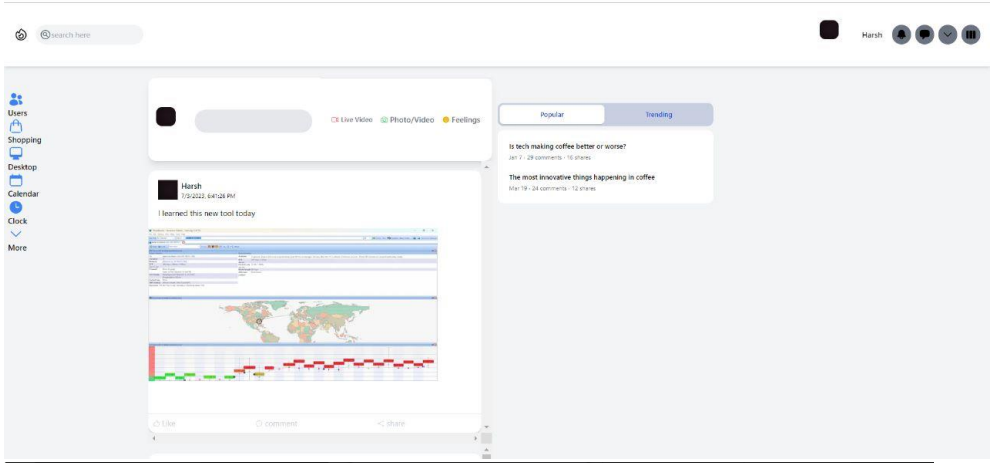
Leverage container orchestration platforms like Kubernetes to scale the containerized app horizontally by adding or removing instances based on resource demands. This helps ensure high availability and scalability of the social media app.

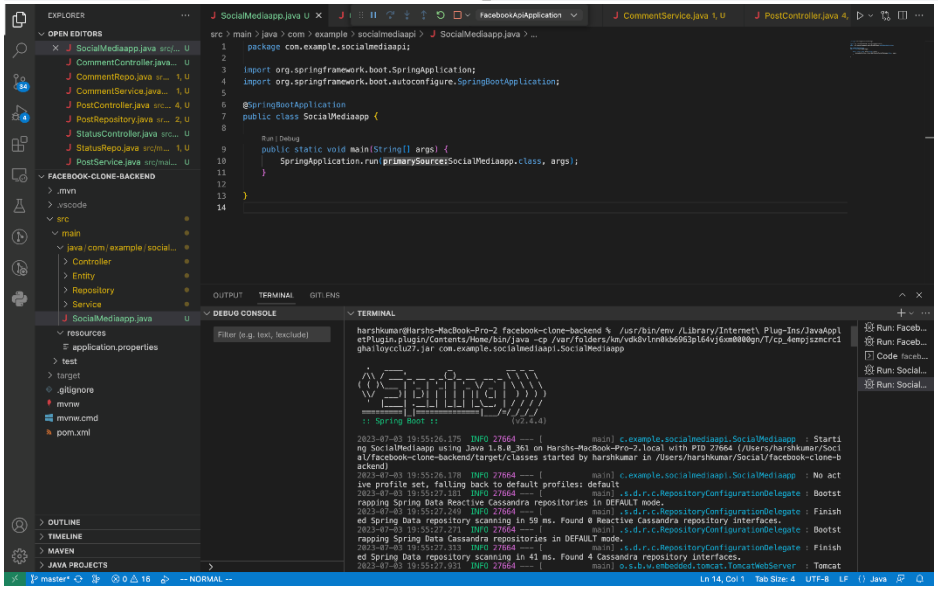
Activity3: Security Considerations:

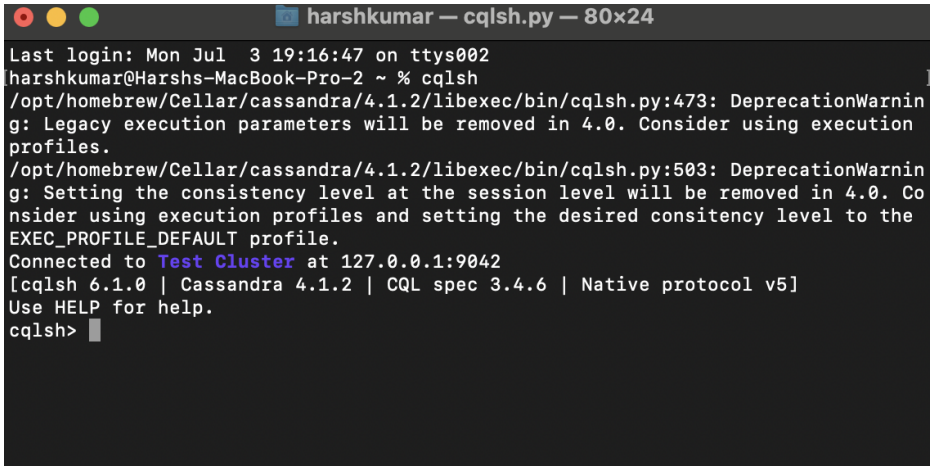
Apply security best practices when containerizing the social media app. This includes ensuring container images are regularly updated with security patches, using secure container registries, enforcing access controls, and following container hardening guidelines.

Project Screenshots:


search here
Harsh
Users
Shopping
Desktop
Calendar
Clock
More
Live Video
Photo/Video
Feelings
Popular
Trending
Is tech making coffee better or worse?
Jan 7 · 29 comments · 16 shares
The most innovative things happening in coffee
Mar 19 · 24 comments · 12 shares
Harsh
7/3/2023, 6:41:26 PM
I learned this new tool today


search here
Harsh
Users
Shopping
Desktop
Calendar
Clock
More
Live Video
Photo/Video
Feelings
Popular
Trending
Is tech making coffee better or worse?
The most innovative things happening in coffee
Harsh
I learned this new tool today


EXPLORER
OPEN EDITORS
SocialMediaapp.java
CommentController.java
CommentRepo.java
CommentService.java
PostController.java
PostRepository.java
StatusController.java
StatusRepo.java
PostService.java
FACEBOOK-CLONE-BACKEND
.mvn
.vscode
src
main
java/com/example/social...
Controller
Entity
Repository
Service
SocialMediaapp.java
resources
application.properties
test
target
.gitignore
mvnw
mvnw.cmd
pom.xml
OUTLINE
TIMELINE
MAVEN
JAVA PROJECTS
SocialMediaapp.java
FacebookApiApplication
CommentService.java 1, U
PostController.java 4,
src > main > java > com > example > socialmediaapi > SocialMediaapp.java > ...
package com.example.socialmediaapi;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class SocialMediaapp {
Run | Debug
public static void main(String[] args) {
SpringApplication.run(primarySource:SocialMediaapp.class, args);
}
}
OUTPUT
TERMINAL
GITLENS
DEBUG CONSOLE
Filter (e.g. text, !exclude)
TERMINAL
harshkumar@Harshs-MacBook-Pro-2 facebook-clone-backend % /usr/bin/env /Library/Internet\ Plug-Ins/JavaAppletPlugin.plugin/Contents/Home/bin/java -cp /var/folders/km/vdk8vlrn0kb6063pl64vj6xm0000gn/T/cp_4empjszmcrc1ghailoycclu27.jar com.example.socialmediaapi.SocialMediaapp
:: Spring Boot ::
(v2.4.4)
2023-07-03 19:55:26.175 INFO 27664 --- [main] c.example.socialmediaapi.SocialMediaapp : Starting SocialMediaapp using Java 1.8.0_361 on Harshs-MacBook-Pro-2.local with PID 27664 (/Users/harshkumar/Social/facebook-clone-backend/target/classes started by harshkumar in /Users/harshkumar/Social/facebook-clone-backend)
2023-07-03 19:55:26.178 INFO 27664 --- [main] c.example.socialmediaapi.SocialMediaapp : No active profile set, falling back to default profiles: default
2023-07-03 19:55:27.181 INFO 27664 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data Reactive Cassandra repositories in DEFAULT mode.
2023-07-03 19:55:27.249 INFO 27664 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 59 ms. Found 0 Reactive Cassandra repository interfaces.
2023-07-03 19:55:27.271 INFO 27664 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data Cassandra repositories in DEFAULT mode.
2023-07-03 19:55:27.313 INFO 27664 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 41 ms. Found 4 Cassandra repository interfaces.
2023-07-03 19:55:27.931 INFO 27664 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat
Run: Faceb...
Run: Faceb...
Code faceb...
Run: Social...
Run: Social...
master*
0 △ 16
-- NORMAL --
Ln 14, Col 1
Tab Size: 4
UTF-8
LF
Java



Project done by,

L.S.Rithin – 20BCE2665 – VIT VELLORE

Harsh Kumar - 20BCY10015 – VIT BHOPAL

Rishi Prabhakar - 20BCE2704 – VIT VELLORE

Sailesh Kumar K L - 20BCE2197 – VIT VELLORE