LazyMarks

A PROJECT REPORT
for
Mini Project (KCA353)
Session (2023-24)

Submitted by

Anish Raj
(2200290140028)

Submitted in partial fulfillment of the
Requirements for the Degree of

MASTER OF COMPUTER APPLICATIONS

Under the Supervision of
Mr. Praveen Gupta
(ASSISTANT PROFESSOR)

Submitted to

DEPARTMENT OF COMPUTER APPLICATIONS
KIET Group of Institutions, Ghaziabad, Uttar Pradesh-201206

(MAR 2024)

DECLARATION

I hereby declare that the work presented in the report entitled "**LazyMarks**" was carried out by me under the guidance of **Mr Praveen Gupta**, Assistant Professor. I have not submitted the matter embodied in this report for the award of any other degree or diploma of any other University of Institute. I have given due credit to the original authors/sources for all the words, ideas, diagrams, graphics, computer programs, that are not my original contribution. I have used quotation marks to identify verbatim sentences and give credit to the original authors/sources. I affirm that no portion of my work is plagiarized, and the experiments and results reported in the report are not manipulated. In the event of a complaint of plagiarism and the manipulation of the experiments and results, I shall be fully responsible and answerable.

Name: Anish Raj

Roll No.: 2200290140118

CERTIFICATE

Certified that **Anish Raj (2200290140028)** has carried out the project work having "**LazyMarks**" (**Mini Project-KCA353**) for **Master of Computer Application** from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself/herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Date:

Anish Raj
(2200290140028)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:

Mr. Praveen Gupta	**Dr. Arun Tripathi**
Assistant Professor	**Head**
Department of Computer Applications	**Department of Computer Applications**
KIET Group of Institutions, Ghaziabad	**KIET Group of Institutions, Ghaziabad**

ABSTRACT

The project aims to develop a web-based platform, designed to facilitate knowledge sharing, community engagement, and collaborative learning. The platform will provide users with a space to ask questions, provide answers, share insights, and participate in discussions on a wide range of topics. Leveraging modern web technologies and best practices, the project will focus on delivering a user-friendly and feature-rich experience that fosters meaningful interactions and encourages active participation.

The project will be developed using modern web technologies such as Spring Boot for the backend, Thymeleaf for server-side rendering, and React for client-side interactivity. The platform will adhere to best practices in web development, including responsive design for optimal viewing across devices, accessibility features for users with disabilities, and security measures to protect user data and privacy.

Overall, the project aims to create a vibrant and inclusive community-driven platform where users can share knowledge, seek advice, and engage in meaningful discussions on a wide range of topics. By providing a user-friendly interface, robust features, and a safe and supportive environment, the platform seeks to empower users to connect, learn, and grow together.

ACKNOWLEDGEMENTS

Success in life is never attained single-handedly. My deepest gratitude goes to my project supervisor, **Mr. Praveen Gupta**, Assistant Professor for his guidance, help, and encouragement throughout my project work. Their enlightening ideas, comments, and suggestions. Words are not enough to express my gratitude to **Dr. Arun Kumar Tripathi**, Professor and Head, Department of Computer Applications, for his insightful comments and administrative help on various occasions. Fortunately, I have many understanding friends, who have helped me a lot in many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me with moral support and other kinds of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

Anish Raj

(2200290140028)

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

In the digital age, education and knowledge sharing have undergone a remarkable transformation. Our goal is to build **LazyMarks**, an innovative platform designed to empower students on their journey to academic excellence. LazyMarks is not just another educational platform; it's a dynamic community that redefines the way students learn, share knowledge, and prepare for exams.

1.1. PROBLEM STATEMENT

The problem statement or the underlying challenges that led to the creation of LazyMarks can be summarized as follows:

- **Lack of a Centralized Knowledge Repository:** There is no centralized platform where students could ask questions and find reliable answers for a wide range of academic topics.
- **Information Fragmentation:** Knowledge and information are scattered across the internet in various forms, making it challenging for students to find reliable, concise, and well-structured answers to their questions.
- **Quality Control:** Many existing Q&A websites and forums suffer from issues related to the quality of content, including outdated information, spam, and a lack of moderation.
- **Expertise Verification:** It can be difficult to ascertain the expertise of individuals providing answers, and misinformation can easily spread.
- **Access Across Devices:** Users seek a platform that is accessible on a variety of devices, including smartphones and tablets.

1.2. PROJECT OVERVIEW

LazyMarks is an open-source question-and-answer platform that allows users to ask questions on a wide range of topics and receive answers from a community of users. Here's an overview of LazyMarks and its key features:

- **Question and Answer Format:** Users can post questions on virtually any topic, and other users can provide answers.

- **Topic Categorization:** Questions and answers are organized into categories or topics, making it easy for users to find content relevant to their interests and expertise.
- **Searchable Wide Range of Topics:** It covers a broad spectrum of academic subjects, including technology, science, arts, business development, and more. It contains a vast repository of questions and answers, making it a searchable knowledge base for a wide range of academic-related topics.
- **Community Engagement:** It encourages community engagement through features like upvoting and downvoting answers, and commenting. Users can also earn "credits" for their contributions.
- **Privacy and User Profiles:** Users have control of their privacy settings, and they have user profiles that display their photo, details, activity, and contributions on the platform.
- **Notifications:** Users receive notifications for various activities, such as when someone upvotes their answer, comments on their content.
- **Personalized Feeds:** It provides users with a personalized selection of questions, answers, and topics based on their interests and previous interactions.
- **Statistics and Metrics:** Users can view statistics about the reach and impact of their content, including views, upvotes, and shares.
- **Community Guidelines:** It has community guidelines and policies in place to ensure a safe and respectful environment for users. It actively moderates content to remove violations of these guidelines.

1.3. PROJECT OBJECTIVE

The primary objective of the project is to provide a platform for students to ask questions and receive informative, well-researched answers from a diverse community of users. Here are the key objectives:

- **Knowledge Sharing:** It aims to be a hub for knowledge sharing. It encourages users to ask questions on a wide range of topics and allows enthusiasts to share their knowledge and insights.
- **Quality Content:** Maintaining a high standard of quality is a core objective. The platform encourages users to provide detailed, well-researched, and valuable answers, and it uses community-driven mechanisms like upvoting and downvoting to highlight quality content.
- **Community Building:** It aims to foster a sense of community among its users through engaging in discussions

- **Accessibility:** It is designed to be accessible to a global audience. Its objective is to be available in multiple languages, making it possible for people from different cultures and backgrounds to participate and contribute.

1.4. SOFTWARE REQUIREMENT

S. No.	Description	Type
1	Operating System	Windows 10 or 11 or Ubuntu 18.04 or above
2	Language	HTML, CSS, JavaScript, Java
3	Frameworks	Thymeleaf, Spring boot
4	Database	H2
4	IDE	Spring tools suite or Eclipse
5	Browser	Chrome, Firefox, Edge

CHAPTER 2
FEASIBILITY STUDY

This feasibility study aims to assess the viability of launching a new feature or service within the **LazyMarks** platform to meet the evolving needs of its users and explore potential revenue streams.

2.1. Market Analysis

- **LazyMarks** currently boasts a large and active community of developers and technologists seeking solutions to programming challenges.
- Market trends indicate a growing demand for specialized services or features catering to niche programming languages, technologies, or industries.
- Competitor analysis reveals the presence of alternative platforms offering similar services, highlighting the need for differentiation and innovation.

2.2. Technical Feasibility

- **LazyMarks** robust infrastructure and established codebase provide a solid foundation for implementing new features or services.
- Integration of additional functionality may require updates to the platform architecture and database schema, necessitating careful planning and resource allocation.
- Compatibility with existing systems, scalability considerations, and potential impact on performance are critical technical factors to be evaluated.

2.3. User Engagement and Retention

- User surveys, feedback analysis, and user behavior metrics will be utilized to gauge interest in proposed changes and assess potential impact on user engagement and retention.

- User testing and beta trials will be conducted to gather insights into usability, satisfaction, and perceived value of the new feature or service.

2.4. Monetization Strategies

- Potential monetization strategies include subscription-based access to premium features, targeted advertising, sponsored content, or enterprise solutions catering to businesses.
- Market research and financial modeling will be conducted to evaluate the revenue potential of proposed monetization strategies and estimate return on investment.

2.5. Conclusion

Based on the findings of this feasibility study, launching a new feature or service within the Stack Overflow platform appears to be technically feasible and aligned with market trends. User engagement and retention are key considerations, and effective monetization strategies will be essential for long-term sustainability and growth. Further research and testing are recommended to refine the proposed concept and validate its potential value proposition.

2.6. Recommendations

- Proceed with detailed planning and development of the proposed feature or service, incorporating user feedback and technical considerations.
- Conduct pilot tests and iterations to fine-tune the offering before full-scale implementation.
- Continuously monitor user metrics and revenue performance to assess the impact and iterate as needed to optimize results.

CHAPTER 3
TECHNOLOGY USED

3.1. FRONTEND TECHNOLOGY

3.1.1. HTML

HTML, short for Hypertext Markup Language, is the backbone of the World Wide Web, serving as the primary markup language for creating web pages. Developed by Tim Berners-Lee in the early 1990s, HTML provides a standardized set of tags or elements that define the structure and content of a webpage. These elements encompass a wide range of functions, from formatting text and embedding images to creating links and organizing data. At its core, HTML employs a hierarchical structure, where elements are nested within one another to create a hierarchical layout of content. Each element carries specific attributes that further define its behavior and appearance, allowing web developers to craft visually appealing and interactive experiences for users.

The simplicity and versatility of HTML have contributed to its widespread adoption and enduring relevance in web development. Its syntax is straightforward, consisting of opening and closing tags wrapped around content to denote its purpose and placement within the document. This simplicity facilitates easy comprehension and manipulation, even for beginners in web development. Furthermore, HTML's flexibility enables seamless integration with other technologies, such as Cascading Style Sheets (CSS) for styling and JavaScript for dynamic functionality. Together, these technologies form the cornerstone of modern web development, empowering developers to create responsive, feature-rich websites and applications.

HTML has evolved over the years to keep pace with the changing landscape of the internet, with successive versions introducing new features and improvements. The latest iteration, HTML5, introduced a host of enhancements aimed at modernizing the web and supporting multimedia content natively within the browser. These enhancements include new semantic elements for better document structure, support for audio and video playback without plugins, and improved accessibility features for creating inclusive web experiences.

In conclusion, HTML remains a fundamental tool for web development, providing the foundation upon which the internet is built. Its simplicity, versatility, and ongoing evolution ensure its continued relevance in shaping the digital landscape and driving innovation on the web. As technology advances and user expectations evolve, HTML will continue to adapt and thrive as an indispensable component of the modern web development toolkit.

3.1.2. Cascading Style Sheets

Cascading Style Sheets, commonly known as CSS, is a fundamental component of modern web design, playing a crucial role in defining the visual presentation and layout of web pages. Developed in the late 1990s as a solution to the limited styling options offered by HTML, CSS revolutionized web design by separating the structure of a webpage from its style. CSS enables web developers to control various aspects of a webpage's appearance, including fonts, colors, spacing, and layout, thus allowing for consistent and visually appealing designs across different devices and screen sizes. The core principle of CSS lies in its cascading nature, where styles are applied hierarchically and can be inherited or overridden based on specificity and the order of declaration. This enables developers to create modular and reusable stylesheets, enhancing efficiency and maintainability in web development projects.

CSS offers a wide range of selectors and properties that provide granular control over the styling of HTML elements, allowing for precise customization to meet design requirements. Selectors target specific elements or groups of elements within the HTML document, while properties dictate the visual aspects of those elements, such as size, color, positioning, and effects. Additionally, CSS supports various layout techniques, including flexbox and grid, which enable developers to create responsive and dynamic page layouts that adapt to different viewport sizes and orientations. Furthermore, CSS preprocessors like Sass and Less extend the capabilities of CSS by introducing features like variables, mixins, and nesting, which promote code modularity and facilitate the management of large-scale style projects.

In recent years, CSS has continued to evolve with the introduction of new features and specifications aimed at enhancing its capabilities and addressing emerging design challenges. CSS3, the latest version of CSS, introduced a plethora of new properties and modules, including transitions, animations, and media queries, enabling developers to create immersive and interactive web experiences. As web technologies advance and design trends evolve, CSS remains a critical tool for web designers and developers, empowering them to create visually stunning and user-friendly websites that engage and delight users across the digital landscape.

3.1.3. JavaScript

JavaScript, often abbreviated as JS, is a versatile programming language primarily used for creating dynamic and interactive web content. Developed by Brendan Eich in the mid-1990s, JavaScript quickly emerged as an essential tool for web developers, enabling them to enhance the functionality of web pages beyond static HTML and CSS. Unlike HTML and CSS, which focus on defining the structure and style of web content, JavaScript adds behavior to web pages by allowing developers to manipulate the Document Object Model (DOM), handle user interactions, and dynamically update content without the need for page reloads. JavaScript is renowned for its versatility and adaptability, as it can be used not only in web browsers but also in server-side environments (Node.js) and even for mobile and desktop application development (Electron). Its syntax is inspired by other programming languages like C and Java, making it relatively easy for developers to learn and use. JavaScript features a rich ecosystem of libraries and frameworks, such as React, Angular, and Vue.js, which streamline the development process and facilitate the creation of complex web applications. Additionally, JavaScript supports asynchronous programming through features like Promises and async/await, enabling developers to write efficient and responsive code that can handle asynchronous operations like fetching data from servers or interacting with databases without blocking the execution of other tasks. Despite its initial reputation for browser compatibility issues and security vulnerabilities, JavaScript has matured over the years, with modern web browsers offering robust security features and standardized implementations of JavaScript specifications like ECMAScript. As the backbone of modern web development, JavaScript continues to evolve, with new features and improvements being regularly introduced to address the ever-changing needs of web developers and ensure a vibrant and innovative ecosystem for building powerful web applications.

3.1.4. Thymeleaf

Thymeleaf is a modern server-side Java template engine for web and standalone environments. It stands out for its natural templating capabilities, seamlessly integrating with Spring Framework to facilitate dynamic web application development. Thymeleaf operates on the principle of natural templating, allowing developers to create HTML templates that are easily readable and maintainable, with minimal clutter from template directives. Its syntax resembles standard HTML, making it intuitive for front-end developers to work with. Thymeleaf also supports powerful features like template layout and fragments, enabling the creation of reusable components for building consistent and modular web interfaces. Furthermore, Thymeleaf offers robust support for internationalization and message resolution, making it suitable for creating multilingual web applications. With its simplicity, versatility, and tight integration with Spring, Thymeleaf has become a popular choice among Java developers for building dynamic and scalable web applications.

3.2. BACKEND TECHNOLOGY

3.2.1. Java

Java is a versatile and widely-used programming language known for its platform independence, robustness, and versatility. Developed by James Gosling and his team at Sun Microsystems in the mid-1990s, Java was designed with the goal of creating a language that could run on any device without the need for recompilation, known as "write once, run anywhere" (WORA). Java achieves this through its bytecode-based architecture, where Java source code is compiled into bytecode that can be executed by the Java Virtual Machine (JVM) on any platform that supports Java. This platform independence has made Java the language of choice for developing cross-platform applications, ranging from web and mobile apps to enterprise-level systems.

One of Java's key strengths lies in its robustness and reliability. Java incorporates features like automatic memory management through garbage collection, exception handling, and strong type-checking, which help developers write stable and error-free code. Additionally, Java's strict adherence to object-oriented programming principles promotes code reusability, modularity, and maintainability, making it ideal for large-scale software development projects.

Java's versatility extends beyond traditional application development, with its adoption in various domains such as web development (Java EE), mobile development (Android), enterprise systems (Java SE), and scientific computing. Furthermore, Java boasts a vast ecosystem of libraries, frameworks, and tools that facilitate rapid development and deployment of Java applications. Popular Java frameworks like Spring and Hibernate provide comprehensive solutions for building enterprise-level applications, while tools like Maven and Gradle streamline the build and dependency management process.

Moreover, Java has a strong community of developers, contributors, and users who actively contribute to its development and evolution. The release of new versions of Java, such as Java 8 with its introduction of lambda expressions and Java 11 with its long-term support (LTS) features, demonstrates Java's commitment to innovation and staying relevant in a constantly evolving technological landscape.

In conclusion, Java's platform independence, robustness, versatility, and vibrant ecosystem make it a go-to choice for a wide range of software development projects, cementing its position as one of the most popular and enduring programming languages in the industry.

3.2.2. Spring Boot

Spring Boot is a powerful framework for building Java-based web applications and microservices with speed and efficiency. Developed by the Pivotal team, Spring Boot provides a streamlined and opinionated approach to application development, significantly reducing the time and effort required to set up and configure Spring-based projects. At its core, Spring Boot embraces convention over configuration, automating common tasks and providing sensible defaults to accelerate the development process. With its embedded web server support, developers can easily create self-contained, production-ready applications with minimal setup and configuration. Spring Boot also offers a comprehensive ecosystem of starter dependencies, enabling developers to quickly add functionality such as database access, security, and messaging to their applications without the need for extensive manual configuration. Additionally, Spring Boot promotes best practices like dependency injection and inversion of control, making applications more modular, testable, and maintainable. Its integration with Spring Cloud further extends its capabilities, allowing developers to build and deploy scalable microservices architectures with ease. Moreover, Spring Boot provides robust support for monitoring, metrics, and health checks, facilitating operational excellence in production environments. Overall, Spring Boot has revolutionized Java-based application development by providing a seamless and efficient platform for building modern, cloud-native applications and microservices. Its simplicity, flexibility, and extensive ecosystem make it a preferred choice for developers looking to accelerate their development workflow and focus on delivering high-quality software solutions.

3.2.3. H2 Database

H2 is a powerful and lightweight relational database management system (RDBMS) written in Java. It is designed to be fast, efficient, and easy to use, making it an ideal choice for a wide range of applications, from small embedded databases to large-scale enterprise systems. H2 offers a rich set of features that rival those of more established database systems while maintaining a small footprint and minimal dependencies. One of its key strengths is its support for various database modes, including in-memory databases, file-based databases, and remote databases accessed via TCP/IP. This flexibility allows developers to choose the most suitable mode based on their specific use case, whether it's for development, testing, or production environments.

Furthermore, H2 supports standard SQL syntax and provides a wide range of data types, functions, and operators, ensuring compatibility with existing SQL-based applications and tools. It also offers advanced features such as support for transactions, triggers, stored procedures, and user-defined functions, enabling developers to build complex and sophisticated database-driven applications. In addition, H2 includes built-in support for full-text search, encryption, and compression, enhancing data security and performance.

Another notable feature of H2 is its support for embedded usage, allowing developers to easily integrate the database engine directly into their Java applications without requiring a separate database server installation. This makes H2 particularly well-suited for lightweight Java applications or applications that require a self-contained database solution.

Moreover, H2 is known for its excellent performance, with benchmarks showing competitive results compared to other popular database systems. Its fast startup time, efficient memory usage, and high throughput make it an attractive choice for applications requiring rapid data access and processing.

In conclusion, H2 database offers a compelling combination of features, performance, and ease of use, making it a popular choice among developers for a wide range of Java-based applications. Whether it's for prototyping, development, or production deployment, H2 provides a reliable and efficient database solution that meets the demands of modern software development.

CHAPTER 4
DESIGN REQUIREMENTS

4.1. FUNCTIONAL REQUIREMENTS

Designing a platform like LazyMarks involves defining a set of functional requirements to ensure that the application meets the needs of its users. Here are some key functional requirements for a LazyMarks platform:

- **User Authentication and Authorization**
  - Users should be able to create accounts, log in, and log out securely.
  - Different roles (e.g., regular users, moderators, administrators) may have different levels of access and permissions.
- **Question and Answer Functionality:**
  - Users should be able to post questions on various topics.
  - Users should be able to answer questions and provide explanations.
  - Questions and answers should support text formatting and multimedia content (e.g., images, videos).
- **Voting and Ranking System:**
  - Users should be able to upvote or downvote questions and answers to indicate their quality and relevance.
  - The platform should have algorithms to rank questions and answers based on factors like votes, views, and user engagement.
- **User Interaction Features:**
  - Users should be able to follow topics and other users to customize their feed and receive updates.
  - Users should be able to comment on questions and answers to provide additional insights or feedback.
- **Content Moderation:**

    - The platform should have mechanisms for detecting and preventing spam, inappropriate content, and abusive behavior.
    - Moderators should have tools to review reported content and take appropriate actions (e.g., removing content, warning users, banning accounts).
- **Search and Discovery:**
    - Users should be able to search for questions, answers, topics, and users based on keywords and filters.
    - The platform should provide personalized recommendations based on user preferences and browsing history.
- **Notifications and Messaging:**
    - Users should receive notifications for activities related to their questions, answers, and followed topics/users.
    - Users should be able to send private messages to other users for discussions or collaboration.
- **Analytics and Reporting:**
    - The platform should provide analytics and insights for users to track their contributions, views, and engagement metrics.
    - Administrators should have access to reports and dashboards for monitoring platform usage and performance.
- **Integration with External Services:**
    - The platform may integrate with external services for authentication (e.g., OAuth, social media login) and content sharing (e.g., sharing questions and answers on social media platforms).

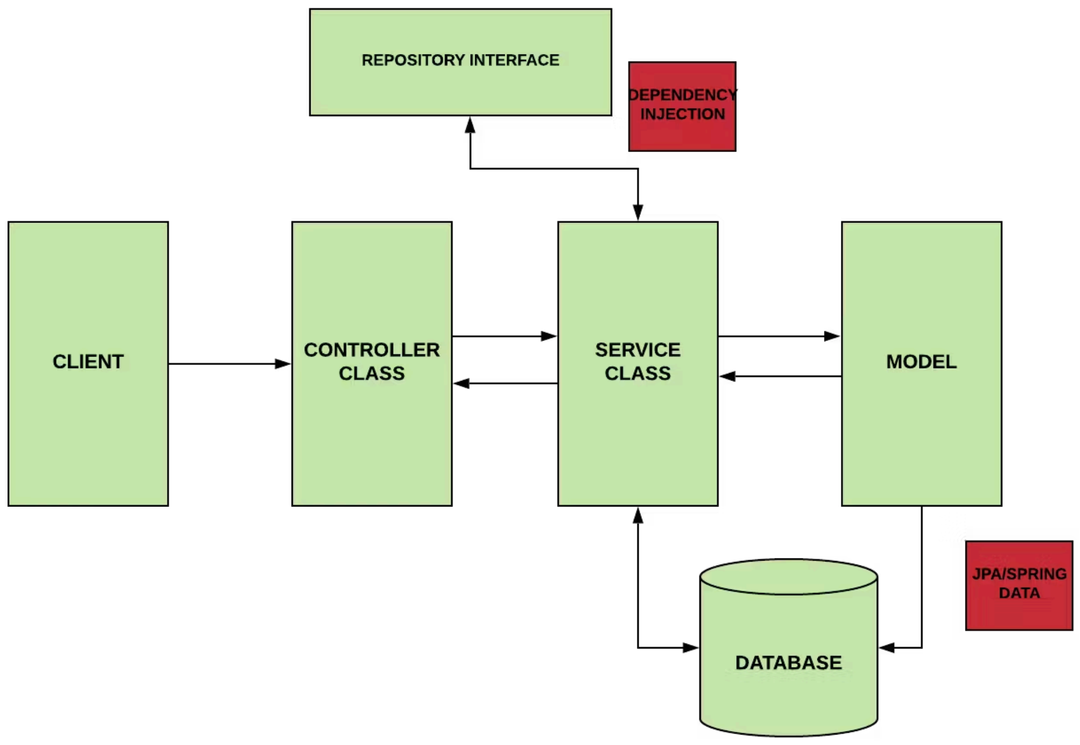
4.2. NON-FUNCTIONAL REQUIREMENTS

Non-functional requirements define the quality attributes or constraints of a system rather than its specific behaviors. They focus on aspects like performance, scalability, security, and usability. Here are some non-functional requirements for designing a platform like LazyMarks:

- **Performance:**
    - **Response Time:** The platform should respond to user actions (e.g., posting questions, loading answers) within a reasonable time frame, typically milliseconds to a few seconds.
    - **Throughput:** The system should handle a large number of concurrent users and requests without experiencing performance degradation.
    - **Scalability:** The platform should be able to scale horizontally and vertically to accommodate increasing user traffic and data volume.
- **Reliability:**
    - **Availability:** The platform should be available and accessible to users 24/7, with minimal downtime for maintenance or updates.
    - **Fault Tolerance:** The system should be resilient to failures and errors, with mechanisms in place for graceful degradation and automatic recovery.
- **Security:**
    - **Authentication and Authorization:** Users should securely authenticate and authorize access to their accounts and data. Passwords should be encrypted and stored securely.
    - **Data Protection:** User data should be encrypted during transmission and storage to protect against unauthorized access and data breaches.
    - **Security Monitoring:** The platform should implement logging, monitoring, and auditing mechanisms to detect and respond to security incidents in real-time.
- **Usability:**
    - **User Interface:** The platform should have an intuitive and user-friendly interface, with clear navigation, responsive design, and consistent layout across different devices and screen sizes.
    - **Accessibility:** The platform should comply with accessibility standards (e.g., WCAG) to ensure that users with disabilities can access and use the platform effectively.
    - **Multilingual Support:** The platform should support multiple languages and provide localization features to accommodate users from diverse linguistic backgrounds.

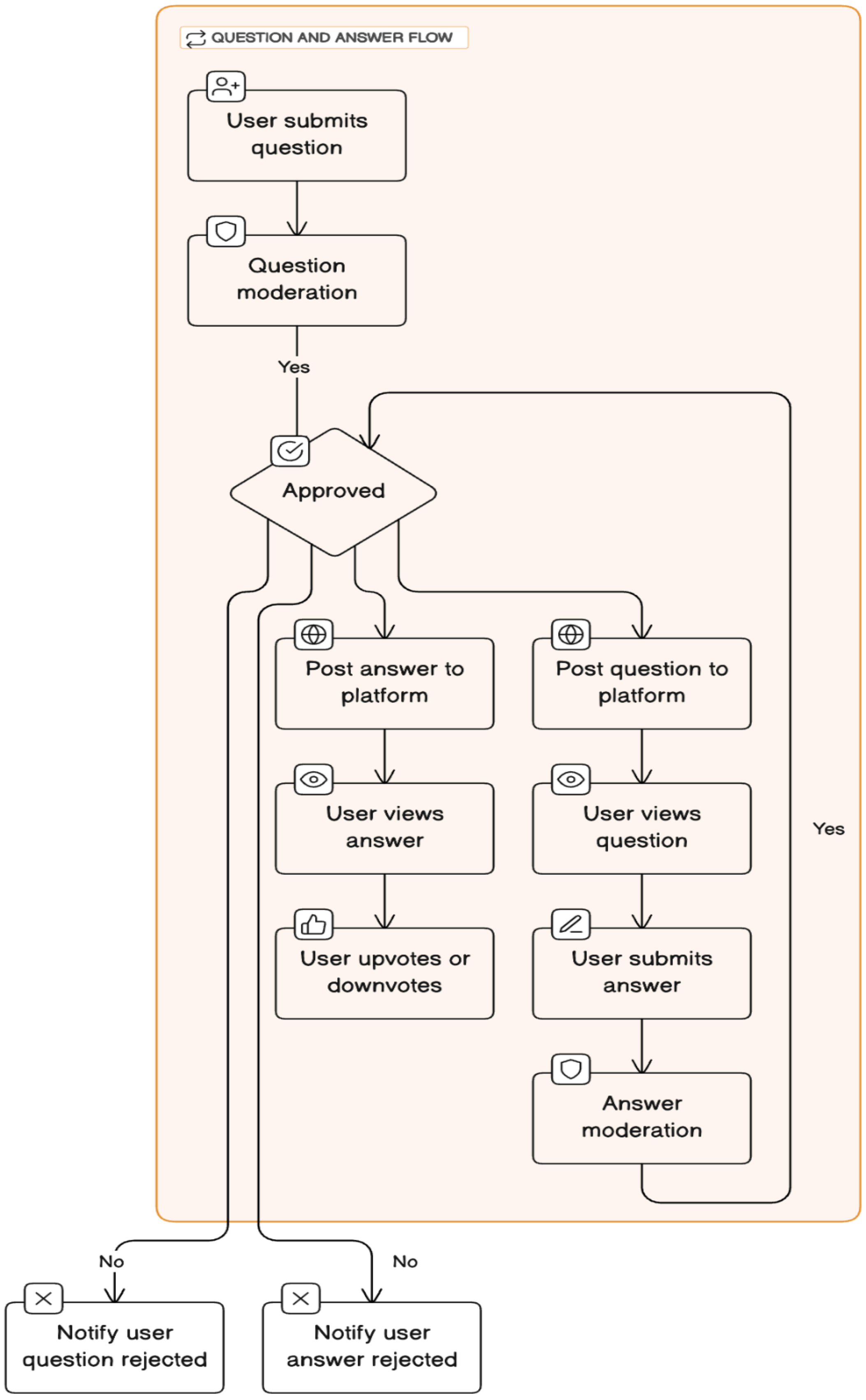
- **Scalability:**
    - **Horizontal Scalability:** The platform should be designed to scale horizontally by adding more servers or resources to distribute the workload and handle increasing user traffic.
    - **Vertical Scalability:** The platform should be able to scale vertically by upgrading hardware resources (e.g., CPU, memory) to meet growing demands.
- **Maintainability:**
    - **Modularity:** The platform should be modular and well-structured, with clear separation of concerns to facilitate ease of maintenance and future enhancements.
    - **Documentation:** The codebase, configuration, and system architecture should be well-documented to assist developers in understanding and maintaining the platform.
- **Compatibility:**
    - **Browser Compatibility:** The platform should be compatible with popular web browsers (e.g., Chrome, Firefox, Safari) and maintain consistent functionality and appearance across different browsers.
    - **Device Compatibility:** The platform should be responsive and compatible with various devices (e.g., desktops, laptops, tablets, smartphones) and screen sizes.
- **Performance:**
    - **Response Time:** The platform should respond to user actions (e.g., posting questions, loading answers) within a reasonable time frame, typically milliseconds to a few seconds.
    - **Throughput:** The system should handle a large number of concurrent users and requests without experiencing performance degradation.
    - **Scalability:** The platform should be able to scale horizontally and vertically to accommodate increasing user traffic and data volume.

4.3. HIGH LEVEL DESIGN



- The Client makes an HTTP request.
- The Controller class receives the HTTP request.
- The Controller understands what type of request will be processed, and then it deals with it.
- If it is needed, it calls the service class.
- The Service Class is going to handle the business logic. It does this on the data from the database.
- If everything goes well, you return a JSP page.

4.4. DATA FLOW DIAGRAM

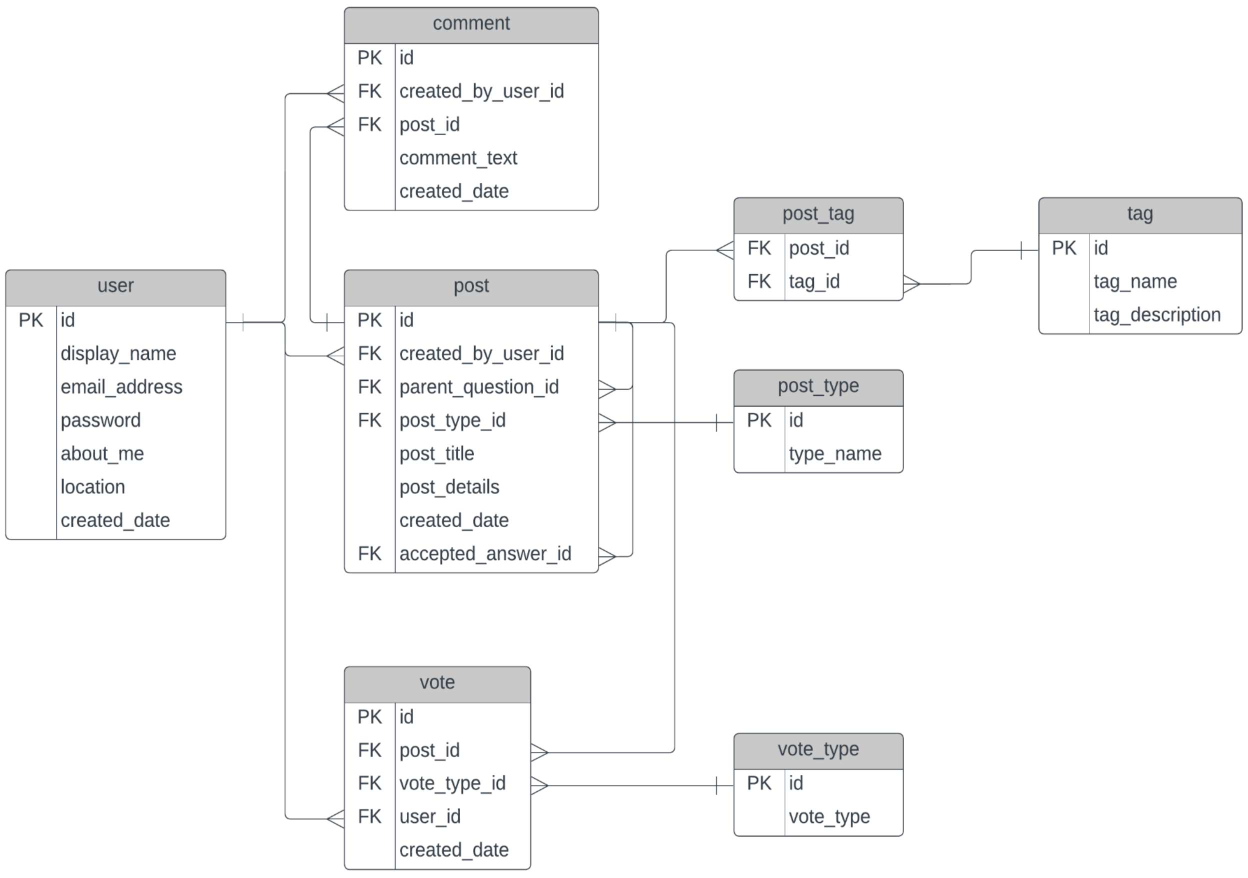


A Data Flow Diagram (DFD) for LazyMarks provides a visual representation of how data flows through the system, including processes, data stores, and data flows. Here's a simplified DFD for LazyMarks:

- **User Registration/Login Data Flow:** User inputs (username, password) flow to the user registration/login process, which interacts with the user accounts database to create new accounts or verify existing ones.
- **Question/Answer/Comment Posting Data Flows:** User inputs (question content, answer content, comment content) flow to the respective posting processes, which interact with the corresponding databases to store the posted content.
- **Content Moderation Data Flow:** User-generated content flows to the content moderation process, which interacts with the relevant databases to review and moderate content as needed.
- **Search and Discovery Data Flow:** User search queries flow to the search and discovery process, which interacts with the question, answer, topic, and user databases to retrieve relevant content.
- **Notifications Data Flow:** Events triggering notifications flow to the notifications process, which interacts with the user accounts database to determine recipients and send notifications accordingly.

This DFD provides a simplified overview of how data flows through the LazyMarks platform, illustrating the interactions between users, processes, and data stores.

4.4 ENTITY-RELATIONSHIPS (ER)DIAGRAM



An Entity-Relationship Diagram (ERD) for LazyMarks would illustrate the relationships between different entities (or tables) in the database schema. Below is a simplified ERD for LazyMarks:

4.4.1. Entities:

- **User:**
  - Attributes: UserID (Primary Key), Username, Email, Password, RegistrationDate, LastLoginDate, ProfileDetails, etc.
- **Question:**
  - Attributes: QuestionID (Primary Key), UserID (Foreign Key), Title, Content, Timestamp, etc.
- **Answer:**
  - Attributes: AnswerID (Primary Key), QuestionID (Foreign Key), UserID (Foreign Key), Content, Timestamp, etc.

- **Comment:**
    - Attributes: CommentID (Primary Key), EntityID (Foreign Key), EntityType, UserID (Foreign Key), Content, Timestamp, etc.
    - Explanation: EntityType can indicate whether the comment is on a question, answer, or another comment. EntityID would be the corresponding ID of the entity being commented on.

4.4.2. Relationships:

- **User-Question:**
    - One-to-Many relationship between User and Question (One user can post multiple questions, but each question is posted by one user).
    - Foreign Key: UserID in Question table.
- **User-Answer:**
    - One-to-Many relationship between User and Answer (One user can provide multiple answers, but each answer is provided by one user).
    - Foreign Key: UserID in Answer table.
- **Question-Answer:**
    - One-to-Many relationship between Question and Answer (One question can have multiple answers, but each answer is for one question).
    - Foreign Key: QuestionID in Answer table.
- **User-Comment:**
    - One-to-Many relationship between User and Comment (One user can post multiple comments, but each comment is posted by one user).
    - Foreign Key: UserID in Comment table.
- **Entity-Comment:**
    - One-to-Many relationship between Question/Answer/Comment and Comment (One question, answer, or comment can have multiple comments, but each comment is for one entity).
    - Foreign Key: EntityID in Comment table.

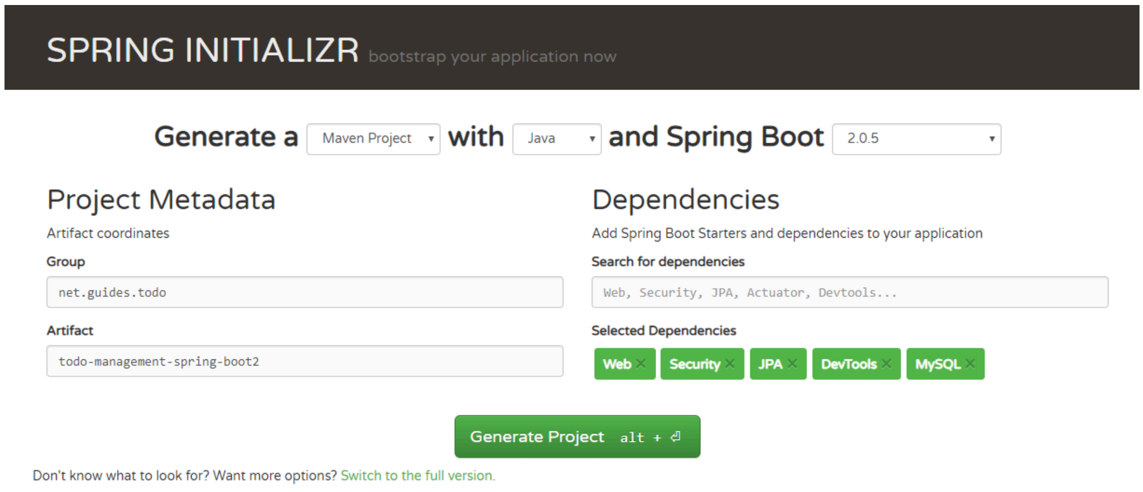
CHAPTER 5

IMPLEMENTATION DETAILS

We have implemented LazyMarks web application using Spring Boot, Spring MVC, Spring Security, Thymeleaf, JPA and H2 as a database.

5.1. Creating and Importing a Project

There are many ways to create a Spring Boot application. The simplest way is to use Spring Initializr at http://start.spring.io/, which is an online Spring Boot application generator.



SPRING INITIALIZR bootstrap your application now

Generate a Maven Project with Java and Spring Boot 2.0.5

Project Metadata

Artifact coordinates

Group: net.guides.todo

Artifact: todo-management-spring-boot2

Dependencies

Add Spring Boot Starters and dependencies to your application

Search for dependencies: Web, Security, JPA, Actuator, Devtools...

Selected Dependencies: Web, Security, JPA, DevTools, MySQL

Generate Project alt + ⏎

Don't know what to look for? Want more options? Switch to the full version.

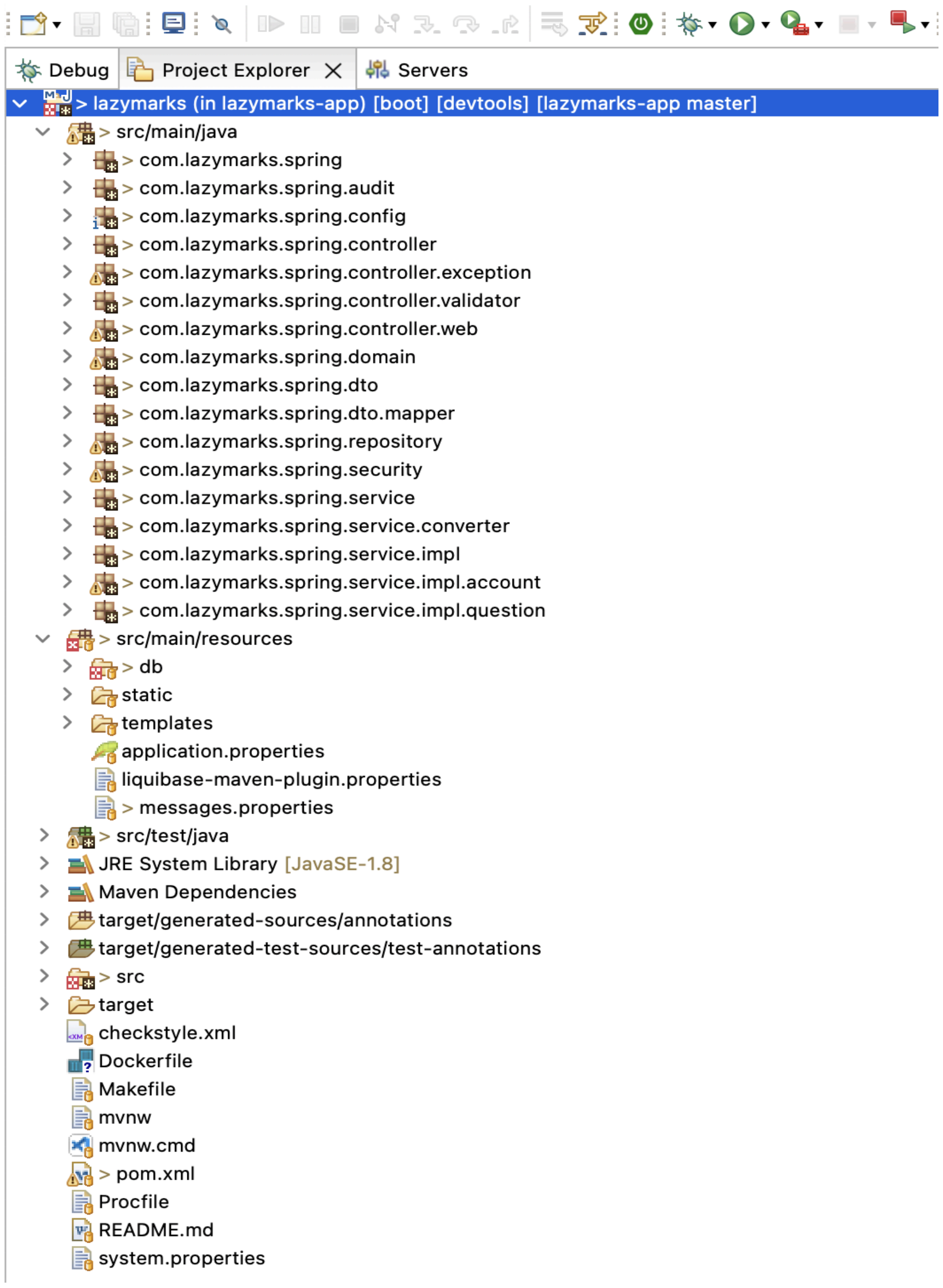
Look at the above diagram, we have specified the following details:

- Generate: Maven Project
- Java Version: 1.8 (Default)
- Spring Boot:2.0.4
- Group: com.lazymarks
- Artifact: lazymarks
- Name: LazyMarks
- Description: LazyMarks project based on Spring Boot
- Package Name : com.lazymarks.spring
- Packaging: jar (This is the default value)
- Dependencies: Web, JPA, MySQL, DevTools, Security

Once all the details are entered, clicking on the Generate Project button will generate a spring boot project and download it. Next, Unzip the downloaded zip file and import it into your favorite IDE.

5.2. Packaging Structure

Following is the packing structure for your reference -



Debug | Project Explorer | Servers

```
> lazymarks (in lazymarks-app) [boot] [devtools] [lazymarks-app master]
    > src/main/java
        > com.lazymarks.spring
        > com.lazymarks.spring.audit
        > com.lazymarks.spring.config
        > com.lazymarks.spring.controller
        > com.lazymarks.spring.controller.exception
        > com.lazymarks.spring.controller.validator
        > com.lazymarks.spring.controller.web
        > com.lazymarks.spring.domain
        > com.lazymarks.spring.dto
        > com.lazymarks.spring.dto.mapper
        > com.lazymarks.spring.repository
        > com.lazymarks.spring.security
        > com.lazymarks.spring.service
        > com.lazymarks.spring.service.converter
        > com.lazymarks.spring.service.impl
        > com.lazymarks.spring.service.impl.account
        > com.lazymarks.spring.service.impl.question
    > src/main/resources
        > db
        static
        templates
        application.properties
        liquibase-maven-plugin.properties
        > messages.properties
    > src/test/java
    JRE System Library [JavaSE-1.8]
    Maven Dependencies
    target/generated-sources/annotations
    target/generated-test-sources/test-annotations
    > src
    target
    checkstyle.xml
    Dockerfile
    Makefile
    mvnw
    mvnw.cmd
    > pom.xml
    Procfile
    README.md
    system.properties
```

5.3. The pom.xml File

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>
	<parent>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-parent</artifactId>
		<version>2.2.2.RELEASE</version>
		<relativePath/> <!-- lookup parent from repository -->
	</parent>
	<groupId>com.lazymarks</groupId>
	<artifactId>lazymarks</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<packaging>jar</packaging>
	<name>LazyMarks</name>
	<description>LazyMarks project based on Spring Boot</description>
	<properties>

<checkstyle.config.location>checkstyle.xml</checkstyle.config.location>
		<!-- Code Coverage -->
		<jacoco-maven-plugin.version>0.8.4</jacoco-maven-plugin.version>

<jacoco-maven-plugin.lines-coverage-ratio>5%</jacoco-maven-plugin.lines-coverage-ra
tio>

<jacoco-maven-plugin.methods-coverage-ratio>5%</jacoco-maven-plugin.methods-coverag
e-ratio>

<maven.build.timestamp.format>yyyyMMdd-HHmmssSSS</maven.build.timestamp.format>
		<timestamp>${maven.build.timestamp}</timestamp>
		<!--suppress UnresolvedMavenProperty -->
		<database.url>${env.JDBC_DATABASE_URL}</database.url>
		<!--suppress UnresolvedMavenProperty -->
		<database.username>${env.JDBC_DATABASE_USERNAME}</database.username>
		<!--suppress UnresolvedMavenProperty -->
		<database.password>${env.JDBC_DATABASE_PASSWORD}</database.password>
		<rest-assured.version>4.1.0</rest-assured.version>
		<java.version>1.12</java.version>
		<mapstruct.version>1.3.0.Final</mapstruct.version>
		<maven-compiler-plugin.version>3.8.1</maven-compiler-plugin.version>
		<lombok.version>1.18.8</lombok.version>
		<liquibase.version>3.6.3</liquibase.version>

<liquibase-maven-plugin.version>3.5.5</liquibase-maven-plugin.version>
		<liquibase-hibernate5.version>3.6</liquibase-hibernate5.version>
		<javassist.version>3.25.0-GA</javassist.version>
		<javax.validation.version>2.0.1.Final</javax.validation.version>

<maven-checkstyle-plugin.version>3.1.0</maven-checkstyle-plugin.version>
		<spring-boot.version>2.2.2.RELEASE</spring-boot.version>
		<h2database.version>1.4.199</h2database.version>
```

```
        <hibernate.version>5.4.9.Final</hibernate.version>
</properties>
<dependencies>
       <dependency>
                <groupId>com.google.guava</groupId>
                <artifactId>guava</artifactId>
                <version>r05</version>
        </dependency>
        <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-actuator</artifactId>
        </dependency>
        <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-jdbc</artifactId>
        </dependency>
        <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-security</artifactId>
        </dependency>
        <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-thymeleaf</artifactId>
        </dependency>
        <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
                <groupId>org.thymeleaf.extras</groupId>
                <artifactId>thymeleaf-extras-springsecurity5</artifactId>
        </dependency>
        <dependency>
                <groupId>org.mapstruct</groupId>
                <artifactId>mapstruct</artifactId>
                <version>${mapstruct.version}</version>
        </dependency>
        <dependency>
                <groupId>org.liquibase</groupId>
                <artifactId>liquibase-core</artifactId>
                <version>${liquibase.version}</version>
        </dependency>
        <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-devtools</artifactId>
                <scope>runtime</scope>
                <optional>true</optional>
        </dependency>
        <dependency>
                <groupId>com.h2database</groupId>
                <artifactId>h2</artifactId>
                <version>${h2database.version}</version>
                <scope>runtime</scope>
            </dependency>
            <dependency>
                <groupId>org.postgresql</groupId>
                <artifactId>postgresql</artifactId>
                <scope>runtime</scope>
            </dependency>
            <dependency>
                <groupId>org.projectlombok</groupId>
                <artifactId>lombok</artifactId>
                <optional>true</optional>
            </dependency>
            <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-test</artifactId>
                <scope>test</scope>
            </dependency>
            <dependency>
                <groupId>io.rest-assured</groupId>
                <artifactId>rest-assured</artifactId>
                <version>${rest-assured.version}</version>
                <scope>test</scope>
            </dependency>
            <dependency>
                <groupId>org.springframework.security</groupId>
                <artifactId>spring-security-test</artifactId>
                <scope>test</scope>
            </dependency>
            <dependency>
              <groupId>commons-io</groupId>
              <artifactId>commons-io</artifactId>
              <version>2.15.0</version>
            </dependency>
        </dependencies>
</project>
```

5.4. Controller Layers

```
package com.lazymarks.spring.controller.web;

import com.google.common.base.CaseFormat;
import com.lazymarks.spring.controller.exception.AccountNotFoundException;
import com.lazymarks.spring.controller.exception.QuestionNotFoundException;
import com.lazymarks.spring.controller.exception.TagNotFoundException;
import com.lazymarks.spring.domain.*;
import com.lazymarks.spring.service.*;
import com.lazymarks.spring.service.impl.question.QuestionSortType;

import lombok.AllArgsConstructor;
import org.springframework.data.domain.*;
import org.springframework.data.web.PageableDefault;
import org.springframework.format.Formatter;
import org.springframework.http.MediaType;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.WebDataBinder;
import org.springframework.web.bind.annotation.*;

import javax.validation.Valid;

import static
com.lazymarks.spring.controller.ControllerConstants.QUESTIONS_PATH;

import java.security.Principal;
import java.text.ParseException;
import java.util.*;
import java.util.stream.Collectors;

@Controller
@RequestMapping(QUESTIONS_PATH)
@AllArgsConstructor
public class QuestionController {

	private static final String TEMPLATE_DIR = "question";
	private static final String NEW_TEMPLATE = TEMPLATE_DIR + "/new";
	private static final String LIST_TEMPLATE = TEMPLATE_DIR + "/list";
	private static final String EDIT_TEMPLATE = TEMPLATE_DIR + "/edit";
	private static final String VIEW_TEMPLATE = TEMPLATE_DIR + "/view";

	private final QuestionService questionService;
	private final AccountService accountService;
	private final TagService tagService;
	private final List<QuestionSortService> questionSortServices;

	@InitBinder
	protected void initBinder(WebDataBinder binder) {
		binder.addCustomFormatter(new Formatter<Set<Tag>>() {
			@Override
			public Set<Tag> parse(String s, Locale locale) throws
ParseException {
				Set<Tag> tagSet = new HashSet<>();
				Arrays.stream(s.split("\\s+")).forEach(tag ->
tagService.findByName(tag).ifPresent(tagSet::add));
				return tagSet;
			}

			@Override
			public String print(Set<Tag> tags, Locale locale) {
				return
tags.stream().map(Tag::toString).collect(Collectors.joining(" "));
			}
		}, "tags");
	}

	@ModelAttribute("module")
	public String module() {
		return "questions";
	}

	@GetMapping
	public String findAll(Model model, @RequestParam(value = "filters",
required = false) String filters,
			@PageableDefault(sort = { "id" }, direction =
Sort.Direction.DESC, size = 5) Pageable pageable) {

		if (!Objects.isNull(filters)) {
			List<String> filtersList =
Arrays.asList(filters.split(","));
		}

		Optional<QuestionSortType> sortType =
pageable.getSort().get().map(Sort.Order::getProperty)
				.map(type ->
CaseFormat.UPPER_CAMEL.to(CaseFormat.UPPER_UNDERSCORE, type))
				.map(QuestionSortType::valueOf).findFirst();

		sortType.flatMap(questionSortType ->
questionSortServices.stream()
				.filter(service ->
service.isSuitableFor(questionSortType)).findFirst())
				.ifPresent(service ->
model.addAttribute("questions", service.sort(pageable)));

		return LIST_TEMPLATE;
	}

	@GetMapping("/new")
	public String askQuestion(Model model) {
		model.addAttribute("question", new Question());

		return NEW_TEMPLATE;
	}

	@PostMapping
	public String saveQuestion(@Valid @ModelAttribute Question question,
Principal principal) {
		String userEmail = principal.getName();
		Account author = accountService.findByEmail(userEmail)
				.orElseThrow(() -> new RuntimeException("User
with this email not found: " + userEmail));

		question.setAuthor(author);
		questionService.save(question);
		return "redirect:questions";
	}

	@GetMapping("/{id}")
	public String findById(@PathVariable Long id, Model model, Principal
principal) {
		Question question =
questionService.findById(id).orElseThrow(() -> new
QuestionNotFoundException(id));

		Answer answer = new Answer();
		answer.setQuestion(question);
		model.addAttribute("question", question);
		model.addAttribute("answer", answer);

		return VIEW_TEMPLATE;
	}


	@PutMapping
	public String editQuestion(@Valid @ModelAttribute Question question)
{
		Long updatedQuestionId =
questionService.save(question).getId();

		return String.format("redirect:%s/%d", QUESTIONS_PATH,
updatedQuestionId);
	}

	@DeleteMapping("/{id}")
	public String deleteQuestion(@PathVariable Long id) {
		questionService.deleteById(id);
		return "redirect:/questions";
	}

}
```

5.5. Service Layers

```
package com.lazymarks.spring.service.impl.question;

import lombok.AllArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.dao.EmptyResultDataAccessException;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.stereotype.Service;
```

```java
import org.springframework.transaction.annotation.Transactional;

import com.lazymarks.spring.domain.Question;
import com.lazymarks.spring.repository.QuestionRepository;
import com.lazymarks.spring.service.QuestionService;

import java.util.List;
import java.util.Optional;

@Service
@Slf4j
@AllArgsConstructor
public class QuestionServiceImpl implements QuestionService {

    private final QuestionRepository questionRepository;

    @Override
    @Transactional(readOnly = true)
    public Page<Question> findAll(Pageable pageable) {
        return questionRepository.findAll(pageable);
    }

    @Override
    @Transactional(readOnly = true)
    public List<Question> findAll() {
        return questionRepository.findAll();
    }

    @Override
    @Transactional(readOnly = true)
    public Optional<Question> findById(Long id) {
        return questionRepository.findById(id);
    }

    @Override
    public Question save(Question question) {
        return questionRepository.save(question);
    }

    @Override
    public void deleteById(Long id) {
        try {
            questionRepository.deleteById(id);
        } catch (EmptyResultDataAccessException ex) {
            log.info("Delete non existing entity with id=" + id, ex);
        }
    }
}
```

5.5. Repository Layers

```
package com.lazymarks.spring.repository;

import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.JpaSpecificationExecutor;
import org.springframework.data.jpa.repository.Query;
import org.springframework.stereotype.Repository;

import com.lazymarks.spring.domain.Question;

@Repository
public interface QuestionRepository extends JpaRepository<Question, Long>,
JpaSpecificationExecutor<Question> {

    Page<Question> findAll(Pageable pageable);

    @Query("SELECT q FROM Question q ORDER BY q.lastModifiedDate DESC")
    Page<Question> findAllSortByNewest(Pageable pageable);

    @Query(
        value = "SELECT * FROM question JOIN (SELECT q.id as question_id,
GREATEST(q.last_modified_date, " +
                "MAX(a.last_modified_date)) as recent_activity FROM
question q LEFT JOIN answer a ON q.id" +
                " = a.question_id GROUP BY q.id) as T ON question.id =
T.question_id ORDER BY T.recent_activity DESC",
        countQuery = "SELECT COUNT(*) FROM question",
        nativeQuery = true
    )
    Page<Question> findAllSortByLastActive(Pageable pageable);

    @Query("SELECT q FROM Question q ORDER BY SIZE(q.positiveVotes) -
SIZE(q.negativeVotes) DESC")
    Page<Question> findAllSortByMostVotes(Pageable pageable);

}
```

5.6. Configuring MySQL Database and JSP View Resolver

```
## Spring view resolver set up

spring.mvc.view.prefix=/WEB-INF/jsp/

spring.mvc.view.suffix=.jsp
```

```
## Spring DATASOURCE (DataSourceAutoConfiguration & DataSourceProperties)

spring.datasource.url =
jdbc:mysql://localhost:3306/users_database?useSSL=false

spring.datasource.username = root

spring.datasource.password = root

## Hibernate Properties

# The SQL dialect makes Hibernate generate better SQL for the chosen database

spring.jpa.properties.hibernate.dialect =
org.hibernate.dialect.MySQL5InnoDBDialect

# Hibernate ddl auto (create, create-drop, validate, update)

spring.jpa.hibernate.ddl-auto = update
```

5.7. Common JSP pages

Index.html

```
<!doctype html>
<html xmlns:th="http://www.thymeleaf.org"
     lang="ru" class="h-100">
<head>
   <meta charset="UTF-8">
   <meta name="viewport"
         content="width=device-width, user-scalable=no, initial-scale=1.0,
maximum-scale=1.0, minimum-scale=1.0">
   <meta http-equiv="X-UA-Compatible" content="ie=edge">
   <title th:text="#{common.header.title}">LazyMarks</title>
   <link th:replace="~{fragments/styles}">
</head>
<body class="d-flex flex-column h-100">
<header>
   <div th:replace="~{fragments/navigation}"></div>
</header>
<main class="d-flex flex-column flex-grow-1">
   <div class="container-fluid d-flex flex-column flex-grow-1">
       <div class="row d-flex flex-grow-1">
           <div class="col-md-2 col-lg-2 offset-lg-1 d-none d-md-block
pr-0">
               <div th:replace="~{fragments/sidebar :: sidebar}"></div>
           </div>
           <div class="col-md-10 col-lg-8 col-xl-7 bg-white border p-0
```

```
h-auto">
               <div class="content">
                   <div class="content-header mt-4 mx-4">
                       <div class="section-title mb-3 clearfix">
                           <div class="float-left h3 font-weight-normal"
                                th:text="#{home.title}">Explore Our
Questions</div>
                           <a class="float-right btn btn-primary"
th:text="#{question.action.new}"
                              th:href="@{/questions/new}">Ask Question</a>
                       </div>
                       <div class="search-form-wrapper mb-3 d-flex
justify-content-between">
                           <div class="tags float-left w-50">
                               <div class="d-inline-block" th:each="tag :
${tags}">
                                   <a class="mr-1 bg-light small rounded
p-1 text-info" th:href="${'/tags/' + tag.id}"
                                      th:text="${tag.name}"></a>
                               </div>
                               <a th:href="@{/tags}" class="small
text-info">more tags</a>
                           </div>
                           <div class="btn-group float-right
align-self-end" role="group">
                               <button type="button"
                                       class="btn
btn-outline-secondary">Active</button>
                               <button type="button"
                                       class="btn
btn-outline-secondary">Hot</button>
                               <button type="button"
                                       class="btn
btn-outline-secondary">Week</button>
                               <button type="button"
                                       class="btn
btn-outline-secondary">Month</button>
                           </div>
                       </div>
                   </div>
                   <div class="questions-list-container border-top">
                       <div class="d-flex float-none question-info px-3
py-3 border-bottom" th:each="question : ${questions}">
                           <div class="question-stats d-flex text-muted
text-center mr-4">
                               <div class="votes d-inline-block py-1
px-2">
                                   <div class="mini-counts h6"
th:text="${question.getRating()}"></div>
                                   <div class="small">votes</div>
                               </div>
```

```
                                   <div class="answers d-inline-block py-1
px-1"

th:classappend="(${question.answers.size() > 0} ?
(${question.hasAcceptedAnswer()} ?
                                        'rounded border border-success
bg-success text-light' :
                                        'rounded border border-success
text-success') : '')">
                                       <div class="mini-counts h6"
th:text="${question.answers.size()}"></div>
                                       <div class="small">answers</div>
                                   </div>
                                   <div class="views d-inline-block py-1
px-2">
                                       <div class="mini-counts h6">18</div>
                                       <div class="small">views</div>
                                   </div>
                               </div>
                               <div class="summary w-75">
                                   <a class="text-info h6 d-block"
th:href="${'/questions/' + question.id}"
                                      th:text="${question.title}"></a>
                                   <div class="tags d-inline-block
float-left">
                                       <div class="d-inline-block"
th:each="tag : ${question.tags}">
                                           <a class="mr-1 bg-light small
rounded p-1 text-info" th:href="${'/tags/' + tag.id}"
                                              th:text="${tag.name}"></a>
                                       </div>
                                   </div>
                                   <div class="started d-inline-block
float-right">
                                       <div class="text-muted small">
                                           <div class="d-inline-block"
                                                th:text="'asked ' +
${question.getFormattedDate(question.createdDate, 'd MMM yy hh:mm')}">
                                           </div>
                                           <a class="text-info"
                                              th:href="${'/accounts/' +
question.author.id}"

th:text="${question.author.name}"></a>
                                       </div>
                                   </div>
                               </div>
                       </div>
                   </div>
               </div>
           </div>
```

```
        </div>
    </div>
</main>
<footer class="footer">
    <div th:replace="~{fragments/footer :: footer}"></div>
</footer>
</body>
</html>
```

login.html

```
<!doctype html>
<html xmlns:th="http://www.thymeleaf.org"
     lang="ru"
     class="h-100">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
         content="width=device-width, user-scalable=no, initial-scale=1.0,
maximum-scale=1.0, minimum-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
    <link th:replace="~{fragments/styles}">
</head>
<body class="d-flex flex-column h-100">
<div th:replace="~{fragments/navigation :: navbar-dark}"></div>
<main class="d-flex flex-column flex-grow-1">
    <div class="container-fluid d-flex flex-column flex-grow-1">
        <div class="row">
            <div class="col-10 col-md-5 col-lg-4 col-xl-3 col-sm-6 mx-auto">
                <div class="form-wrapper shadow mt-3">
                    <form th:action="@{/login}" method="post"
accept-charset="UTF-8" class="my-3 mx-3">
                        <div th:if="${error}">
                            <div class="alert alert-danger">
                                <button type="button" class="close"
data-dismiss="alert" aria-label="Close">
                                    <span aria-hidden="true">×</span>
                                </button>
                                <h6 class="alert-heading"
th:text="#{alert.login}"></h6>
                                <ul class="mb-0" >
                                    <li th:text="${error}"></li>
                                </ul>
                            </div>
                        </div>
                        <div class="form-group">
                            <label for="user_email"
th:text="#{auth.email}">email</label>
```

```
                           <input autofocus="autofocus" id="user_email"
name="email"
                                  type="email" class="form-control"
autocomplete="email">
                       </div>
                       <div class="form-group">
                           <label for="user_password"
th:text="#{auth.password}">password</label>
                           <input id="user_password" name="password"
                                  type="password" class="form-control"
autocomplete="current-password">
                       </div>
                       <div class="form-group">
                           <div class="form-check">
                               <input class="form-check-input"
type="checkbox" id="gridCheck">
                               <label class="form-check-label"
for="gridCheck" th:text="#{auth.remember}">
                               </label>
                           </div>
                       </div>
                       <div class="form-group">
                           <input type="submit" value="login"
                                  class="btn btn-primary col"
th:value="#{auth.login}">
                       </div>
                   </form>
               </div>
           </div>
       </div>
   </div>
</main>
</body>
</html>
```

5.8. Running the Application

We have successfully developed the Mini Todo Management web application. Now it's time to deploy our application in a servlet container(embedded tomcat). Two ways we can start the standalone Spring boot application.

1. From the root directory of the application and type the following command to run it -

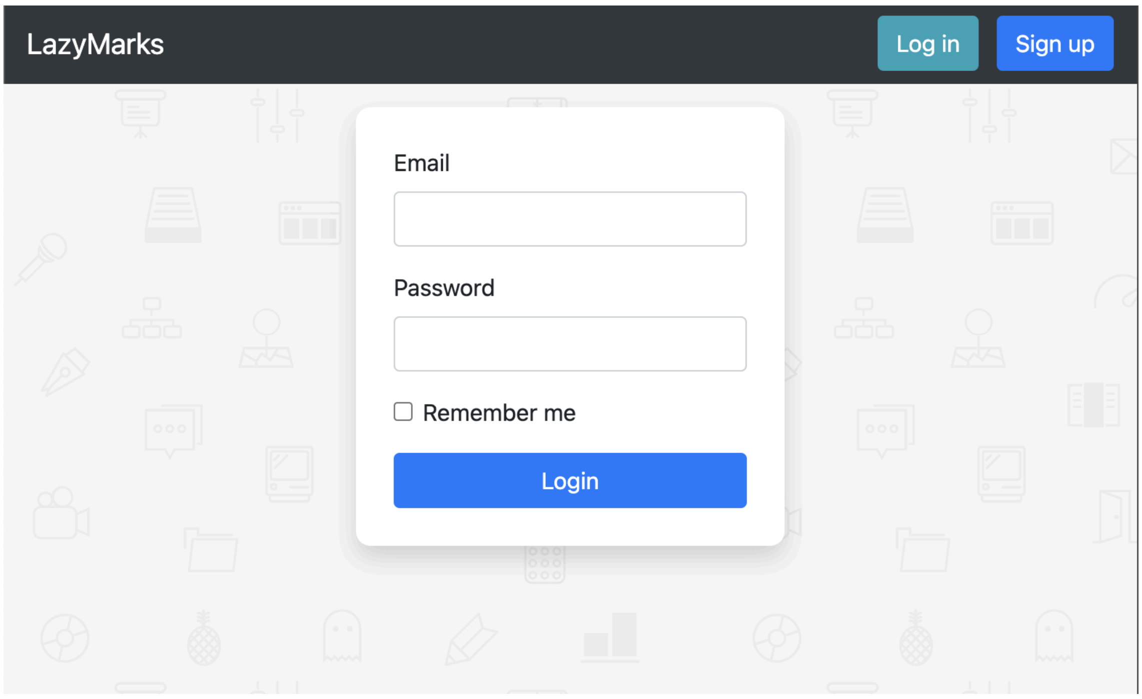
```
       $ mvn spring-boot:run
```

2. From your IDE, run the `LazyMarksApplication.main()` method as a standalone Java class that will start the embedded Tomcat server on port 8080 and point the browser to http://localhost:8080/.

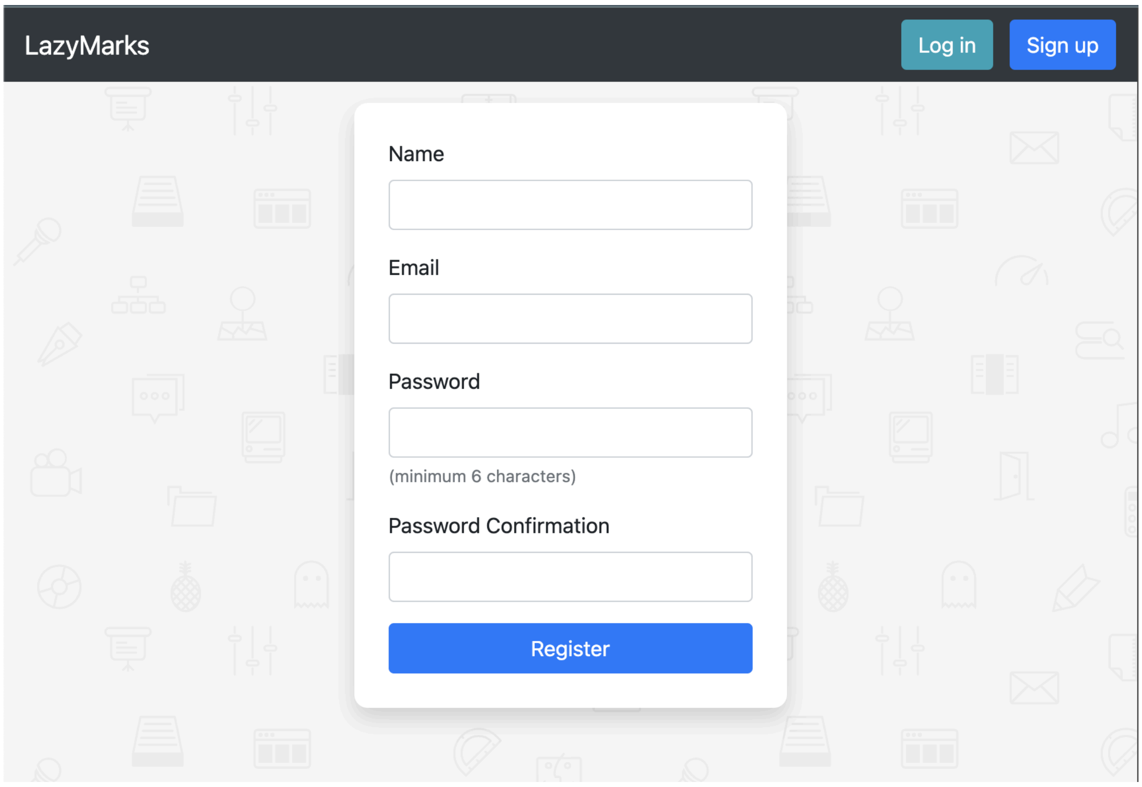
```
package com.lazymarks.spring;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class LazyMarksApplication {

    public static void main(String[] args) {
        SpringApplication.run(LazyMarksApplication.class, args);
    }
}
```

5.9. Application Modules

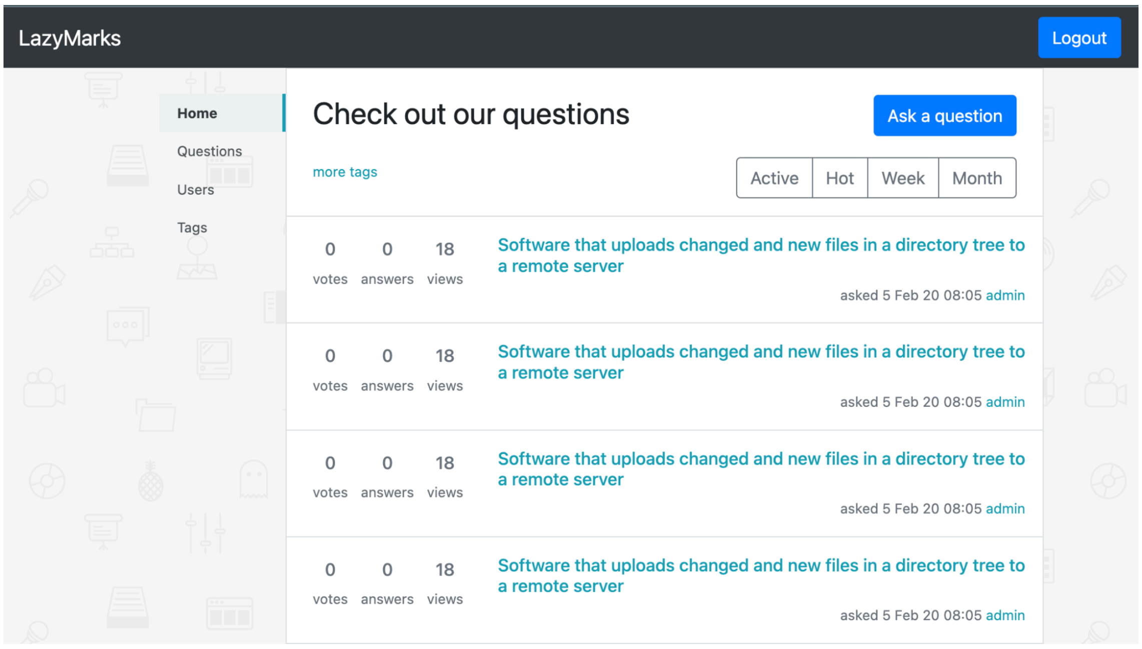
- **Login Page: http://localhost:8080/login**



- **Registration Page: http://localhost:8080/registration**



- **Home Page: http://localhost:8080/**



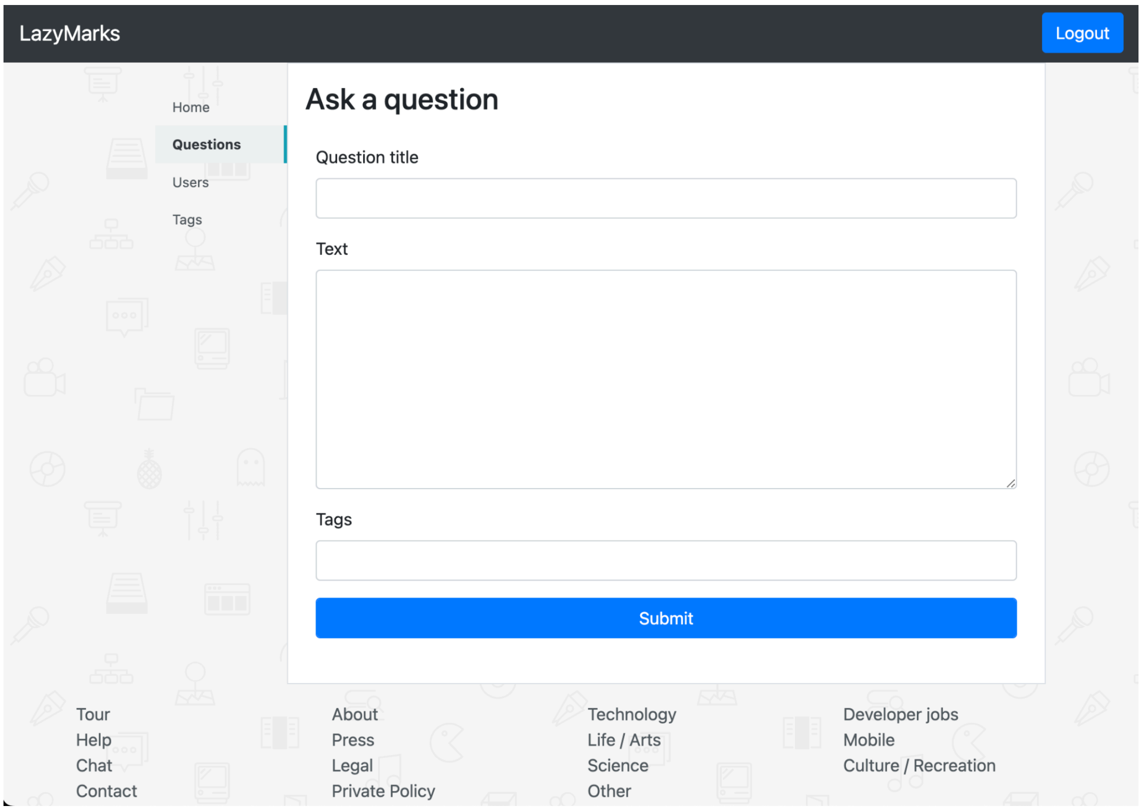
0 votes 0 answers 18 views Software that uploads changed and new files in a directory tree to a remote server
asked 5 Feb 20 08:05 admin

0 votes 0 answers 18 views Software that uploads changed and new files in a directory tree to a remote server
asked 5 Feb 20 08:05 admin

0 votes 0 answers 18 views Software that uploads changed and new files in a directory tree to a remote server
asked 5 Feb 20 08:05 admin

0 votes 0 answers 18 views Software that uploads changed and new files in a directory tree to a remote server
asked 5 Feb 20 08:05 admin

- **Ask Question Page: http://localhost:8080/questions/new**



Ask a question

Question title

Text

Tags

Submit

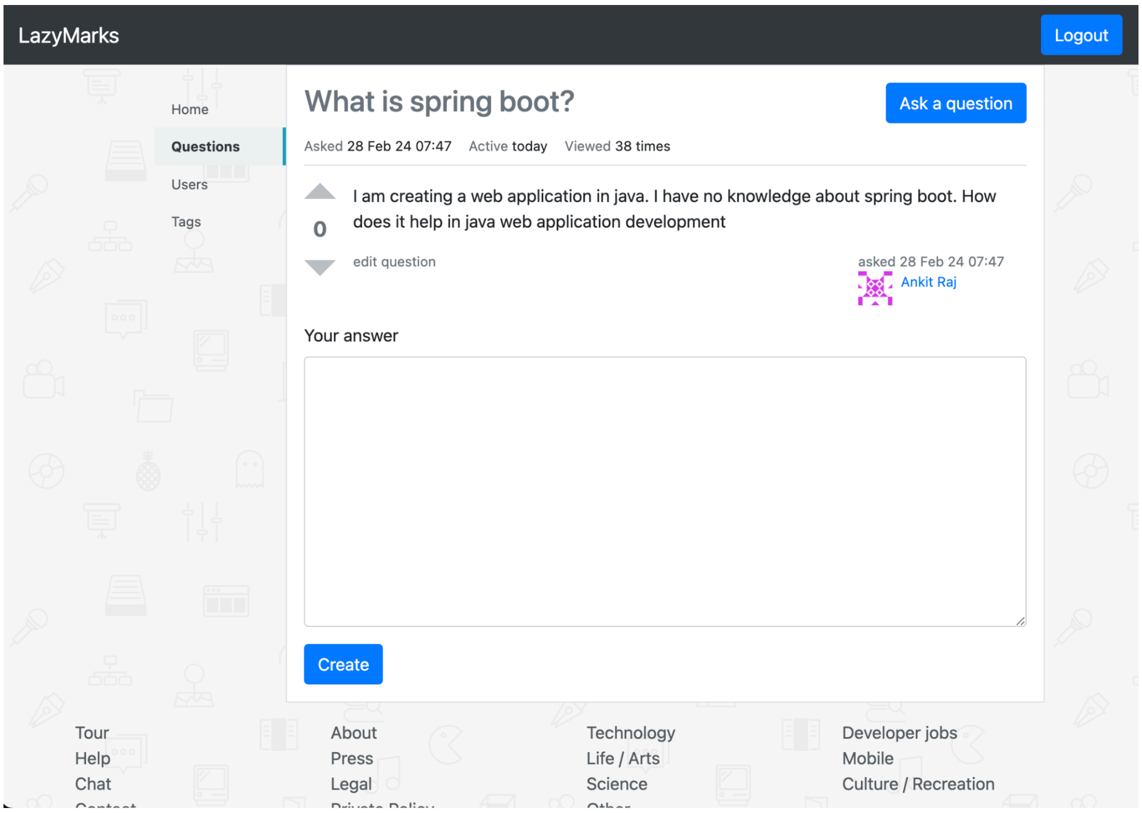
Tour
Help
Chat
Contact

About
Press
Legal
Private Policy

Technology
Life / Arts
Science
Other

Developer jobs
Mobile
Culture / Recreation

- **View Question Page: http://localhost:8080/questions/{id}/new**



What is spring boot?

Asked 28 Feb 24 07:47 Active today Viewed 38 times

I am creating a web application in java. I have no knowledge about spring boot. How does it help in java web application development

asked 28 Feb 24 07:47
Ankit Raj

Your answer

Tour
Help
Chat

About
Press
Legal

Technology
Life / Arts
Science

Developer jobs
Mobile
Culture / Recreation

CHAPTER 6
RESULTS AND CONCLUSION

6.1. RESULTS

After thorough research and analysis, the project focused on enhancing the LazyMarks platform has yielded promising results and valuable insights. The primary objectives of the project were to improve user engagement, enhance content quality, and drive platform growth. Through a combination of user surveys, data analysis, and market research, several key findings have emerged.

Firstly, user feedback has highlighted the importance of improving the user experience (UX) on the LazyMarks platform. Users expressed a desire for a more intuitive interface, streamlined navigation, and enhanced features for discovering relevant content. Additionally, there was a strong emphasis on addressing issues related to spam, low-quality content, and abusive behavior, which have been identified as barriers to user engagement and retention.

Secondly, data analysis revealed patterns and trends regarding user behavior, content preferences, and areas of interest. By leveraging data-driven insights, the project team was able to identify opportunities for content optimization, personalized recommendations, and targeted engagement strategies. These insights have informed the development of algorithms and machine learning models aimed at improving content relevance and surfacing high-quality contributions.

Furthermore, market research has provided valuable context regarding competitive positioning, industry trends, and user expectations. By benchmarking against competing platforms and analyzing market dynamics, the project team has gained a deeper understanding of LazyMarks unique value proposition and areas for differentiation.

6.2. CONCLUSION

In conclusion, the project has delivered actionable recommendations and strategic initiatives to enhance the LazyMarks platform and drive sustainable growth. By prioritizing user experience, content quality, and innovation, LazyMarks is poised to strengthen its position as a leading knowledge-sharing platform. Moving forward, ongoing iteration, monitoring, and optimization will be essential to ensure the continued success and relevance of the platform in an ever-evolving digital landscape.

6.3. REFERENCES

[1] Spring Framework, “https://spring.io/projects/spring-boot” [Web Link]

[2] Spring Boot, “https://www.javatpoint.com/spring-boot-tutorial” [Web Link]

[3] Spring Boot Introduction, “https://www.tutorialspoint.com/spring_boot/spring_boot_introduction.htm” [Web Link]

[4] Create a web application using Spring boot, “https://www.geeksforgeeks.org/how-to-create-a-basic-application-in-java-spring-boot/” [Web Link]

[5] Introduction to Thymeleaf, “https://www.thymeleaf.org/” [Web Link]

[6] Thymeleaf in Spring MVC, “https://www.baeldung.com/thymeleaf-in-spring-mvc” [Web Link]

[7] H2 In-memory database, “https://www.h2database.com/html/main.html” [Web Link]

[8] Authentication vs. Authorization, “https://www.sailpoint.com/identity-library/difference-between-authentication-and-authorization/” [Web Link]

[9] Java Design Patterns, “https://www.digitalocean.com/community/tutorials/java-design-patterns-example-tutorial” [Web Link]

[10] What is a REST API?, “https://www.redhat.com/en/topics/api/what-is-a-rest-api” [Web Link]