PROJECT SYNOPSIS
on

LazyMarks

by

Anish Raj
Enrollment No: 2200290140028
Session: 2023-2024 (2nd Year)

Under the supervision of

Dr Amit Kumar, Professor
KIET Group of Institutions, Delhi-NCR, Ghaziabad

DEPARTMENT OF COMPUTER APPLICATIONS
KIET GROUP OF INSTITUTIONS, DELHI-NCR, GHAZIABAD - 201206

TABLE OF CONTENTS

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.

2. 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Control:** Many existing Q&A websites and forums suffer from issues related to the quality of content, including outdated information, spam, and a lack of moderation.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.

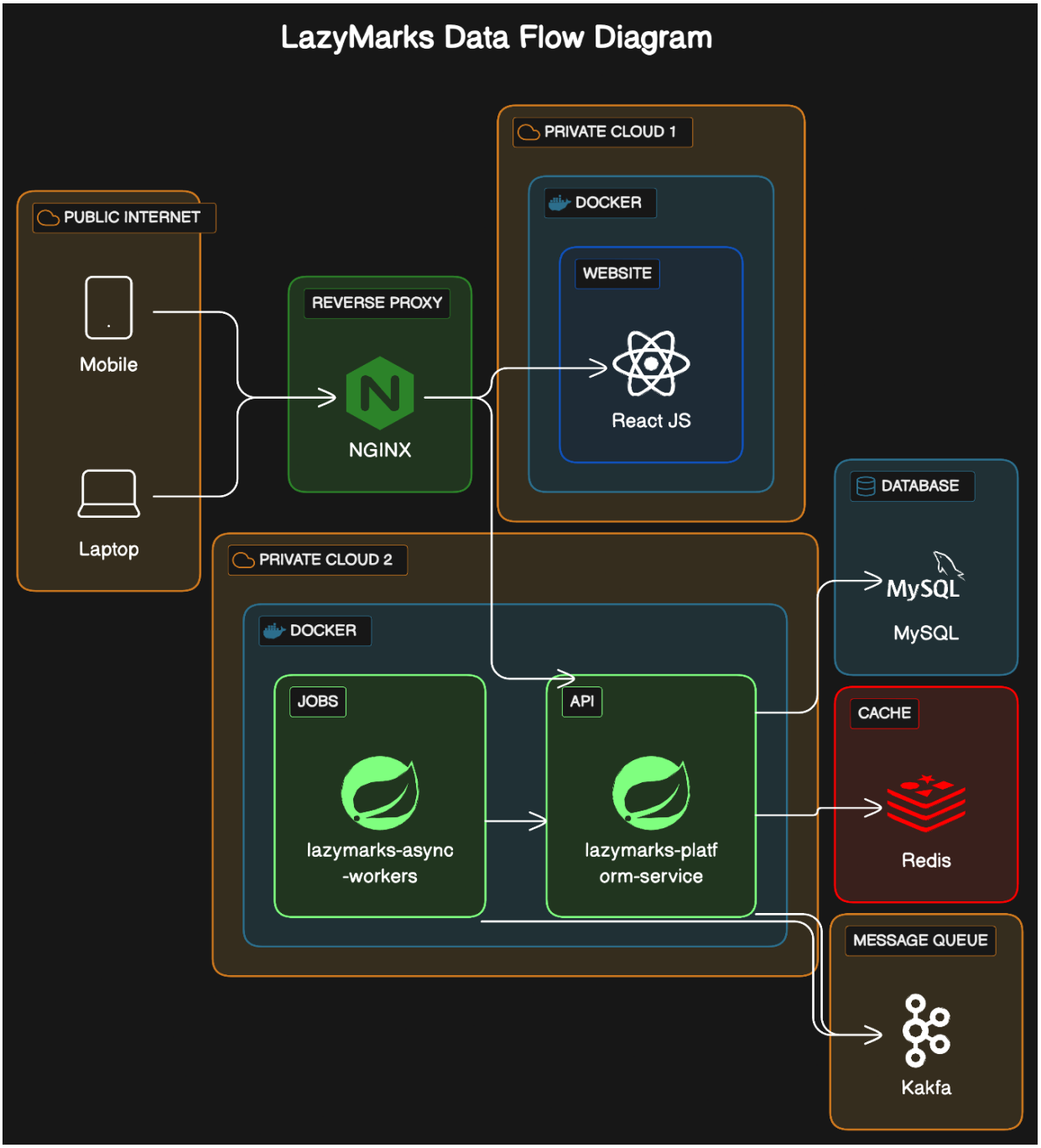
3. Project Overview

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.

3. Project Architecture

Project architecture is designed to handle a massive volume of traffic, user interactions, and content while maintaining high availability and performance. Here is a synopsis of project's architecture design:



Monolith Architecture:

All functionalities of the application exist in a single codebase, hence it is monolithic in nature. Application is developed in various layers like presentation, service, and persistence and then it is deployed as a single war file on the server.

- Presentation layer — responsible for handling HTTP requests
- Service layer — the application's business logic
- Persistence layer — data access objects responsible for accessing the database

Reverse Proxy:

Reverse proxies are commonly used to improve security, performance, and load balancing for web applications. Here are some key aspects of reverse proxies:

- Load Balancing
- Backend Routing
- Caching
- API Gateway
- Rate Limiting

Caching:

Caching is an integral part of the project's architecture. It reduces database and server load by storing frequently accessed data in memory. Redis is used for caching data like user profiles and frequently requested content.

Data Storage:

Project relies on relational databases for storing structured data like user accounts, questions, answers, and user interactions. MySQL is used for this purpose.

Content Delivery:

To ensure fast and reliable content delivery, the project leverages content delivery networks (CDNs). CDNs store and distribute static assets like images, stylesheets, and JavaScript files closer to the end-users, reducing latency and improving page load times.

Message Queues:

Message queues like Kafka are used to handle asynchronous processing of tasks. These queues manage activities such as sending notifications, background job processing, and email delivery.

Containerization:

Docker is a form of lightweight virtualization which helps in virtualization by providing a more efficient and flexible way to package, distribute, and run applications.

- **Isolation:** Each docker container runs in its own isolated environment, separate from the host system and other containers.
- **Resource Efficiency:** Docker containers share the host operating system's kernel. This reduces resource overhead, making Docker containers more lightweight and efficient.
- **Portability:** Docker containers package applications and their dependencies into a single, portable unit which can run on different environments with little to no modification.
- **Scalability:** Docker containers are highly scalable. You can quickly create, replicate, and deploy containers as needed to handle varying workloads.
- **Version Control:** Docker images can be version-controlled, allowing you to track changes to your application and easily roll back to previous versions if issues arise.
- **Rapid Deployment:** Docker facilitates rapid application deployment. Containers can be started or stopped in seconds, enabling quick updates and scaling of applications without downtime.

4. Software Requirements:

- **Languages:** Java 17, Javascript ES6, HTML, CSS
- **IDE:** Eclipse, VS Code
- **Reverse Proxy:** NGINX
- **Containerization:** Docker
- **Frameworks:** Spring boot, ReactJS
- **Database:** MySQL
- **Cache:** Redis
- **Message Queue:** Kafka

5. 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.