VNR VJIET

Name of the Laboratory:

Name of the Experiment:

Experiment No: Dates

High Level Design Document

On

Yulu App

Submitted By

Group Details:

P.Lasya- 22071A0547

P.UmaMaheswar-22071A0548

VNR VJIET

Name of the Laboratory:

Name of the Experiment:

Experiment No: Dates

Table of Contents

VNR VJIET

Name of the Laboratory:

Name of the Experiment:

Experiment No: Dates

1. Project Overview

1.1Purpose

The purpose of this document is to specify the high-level design for the Yulu App. This document will act as an outline for implementation and discuss the design considerations.

1.2 Audience

This high-level design is intended to be used by members of the development team that will implement the functionality of the Yulu app. This document will also be used to communicate the high-level design and design considerations to the staff members.

1.3 Design Process

The Yulu app's design process centers on harnessing modern technologies to ensure a seamless user experience and robust backend infrastructure. Developed using MongoDB, Express.js, React.js, and Node.js (MERN stack), with React Native for mobile development, the app prioritizes efficient communication between the client and server. MongoDB stores user and ride data flexibly, while Express.js facilitates RESTful API creation. React.js delivers modular, responsive interfaces, and Node.js optimizes server-side performance. React Native ensures platform consistency, and RESTful APIs enable efficient data transfer. Overall, the Yulu app aims for scalability, flexibility, and a superior user experience in urban mobility.

2.Requirements

The requirements for the Yulu app maintain consistency with the initial phases of analysis and planning. In essence, the primary objectives are to deliver a user-friendly micro-mobility solution that seamlessly integrates into users' urban lifestyles. This entails providing a convenient means for users to access bicycles and electric scooters through the app, enabling them to locate nearby vehicles, unlock them, and initiate rides effortlessly. Security is paramount, necessitating robust authentication mechanisms to safeguard user identities and authorize access securely. Additionally, the app facilitates smooth booking and reservation processes, allowing users to reserve vehicles in advance and manage their rides with flexibility. Integration with a secure payment gateway is essential to facilitate seamless transactions, ensuring a hassle-free experience for users. Real-time tracking of vehicles' locations and statuses is crucial for users to accurately locate nearby vehicles and receive live updates on availability. A rating and feedback system will enable users to provide valuable insights, contributing to service improvement and enhancing overall user satisfaction. Data security and privacy are of utmost importance, requiring the implementation of encryption protocols and secure authentication mechanisms to protect user information. These requirements collectively aim to deliver a comprehensive micro-mobility solution that meets the needs of users while upholding high standards of security and reliability.

2.1 Proposed solution

The proposed architecture for Yulu adopts a modern microservices-based approach, capitalizing on cloud services to ensure scalability, reliability, and performance. It encompasses frontend clients tailored for both web and mobile interfaces, backed by a robust API server for handling business logic and data management, and relies on cloud-based storage solutions for content management.

The front-end components of the app encompass a website catering to desktop users and a dedicated mobile application optimized for phones and tablets. These interfaces prioritize user-friendliness, featuring intuitive search and navigation functionalities to facilitate swift access to desired content. Leveraging contemporary technologies such as React.js and React Native, the website and app deliver swift responsiveness across diverse devices.

At the core of the architecture lies a server responsible for processing requests from the front-end and furnishing users with requisite information. This server, crafted using Node.js and Express.js, interfaces with a database housing all essential app data, spanning user profiles, educational content, and analytics data. For enhanced reliability and scalability, the database can be hosted on leading cloud platforms like Amazon Web Services (AWS) or Google Cloud Platform (GCP).

By embracing this proposed solution, Yulu ensures a seamless user experience characterized by swift responsiveness, intuitive navigation, and reliable access to essential functionalities across web and mobile platforms.

2.2 Capacity Planning

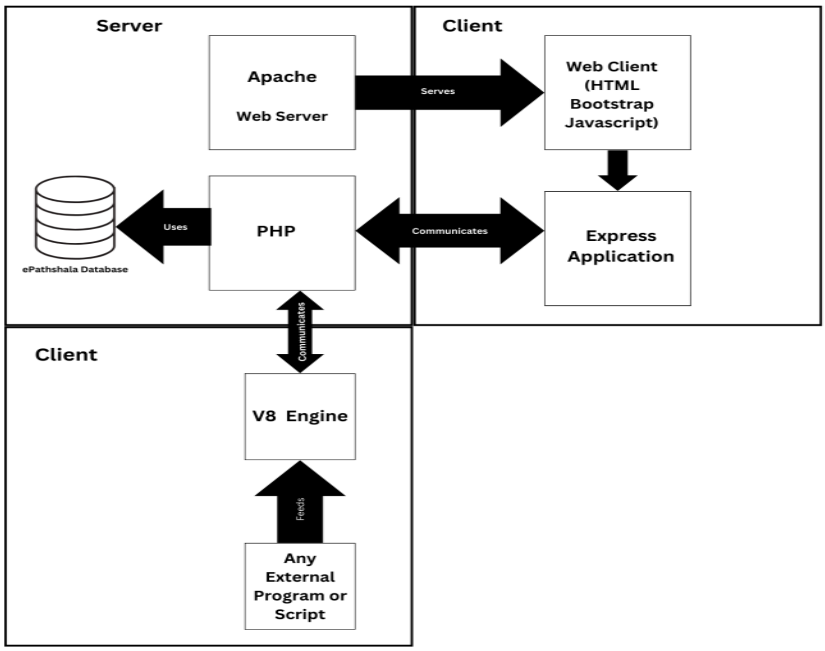
In the Yulu app, data persistency is achieved through a lightweight and efficient approach similar to Java Serialization, tailored to suit the specific requirements of micro-mobility transactions. Serialized objects, termed "YuluElements," encapsulate essential data for ride transactions, ensuring minimal data overhead. Each YuluElement, on average, is designed to occupy no more than 1.5 KB of storage space, efficiently containing pertinent information for each ride.

To estimate the storage requirements for accommodating a substantial volume of ride transactions, consider a scenario where the system stores 100,000 YuluElements. This represents a considerable number of ride transactions, and based on the assumed size of 1.5 KB per YuluElement, the total storage space required would approximate around 150 MB. This calculation offers a clear understanding of the storage needs for handling significant volumes of ride data efficiently.

Given the compact nature of serialized YuluElements and their modest storage footprint, capacity planning for the Yulu app's storage infrastructure is straightforward. Allocating less than 1 GB of storage to the system would be more than adequate to accommodate foreseeable growth and ensure optimal performance. This approach enables efficient utilization of storage resources while maintaining scalability and responsiveness as the user base expands and the volume of ride transactions increases.

3.Architecture

3.1 Design



Server:

Apache Web Server:

In the Yulu architecture, Apache serves as the primary web server responsible for hosting and delivering the frontend web application to users' browsers. It delivers the user interface components and facilitates user interactions through web browsers, allowing users to access and interact with the Yulu app seamlessly.

Yulu Database:

The Yulu database stores structured data related to the application, including user profiles, ride information, vehicle availability, and transaction logs. It provides data persistence and retrieval capabilities, enabling the application to store, organize, and query data efficiently to support various functionalities and user interactions.

Web Client:

The web client in the Yulu app refers to the frontend components that users interact with through web browsers. It includes user interfaces, interactive features, and presentation logic implemented using web technologies such as HTML, CSS, and JavaScript. The web client communicates with backend services via HTTP requests to fetch data, book rides, view ride history, and perform other actions.

Express Application:

Express is a minimalist web framework for Node.js that simplifies the development of web applications and APIs. In the Yulu architecture, the Express application serves as the backend server responsible for handling HTTP requests from the web client, processing business logic, and interacting with the database. It defines routes, middleware, and controllers to handle different types of requests and generate appropriate responses.

Node.js Runtime:

Node.js is a JavaScript runtime environment that allows developers to run JavaScript code on the server-side. In the Yulu architecture, Node.js is utilized by the Express application to execute JavaScript code, enabling efficient server-side processing of requests and responses. It leverages the V8 engine to execute JavaScript code, providing high performance and scalability for handling concurrent connections and real-time interactions.

External Components:

The Yulu architecture may incorporate external components or scripts to provide specific functionality or perform auxiliary tasks. These components could include data migration tools, batch processing scripts, or third-party integrations for analytics, authentication, or payment processing. By integrating these components, the Yulu app extends its functionality and integrates with other parts of the architecture to enhance its capabilities and provide a seamless user experience.

3.2 Access:

The Yulu app relies on a robust server-side environment powered by Node.js to handle user requests, process data, and execute essential server-side functionalities. Node.js, known for its efficiency and scalability, adeptly manages interactions with databases, executes business logic, and dynamically generates content tailored to user needs. This server-side environment ensures seamless operation of the platform, facilitating efficient communication between the client-side interface and the underlying server infrastructure.

On the client side, JavaScript plays a pivotal role, enabling dynamic and interactive user experiences within the Yulu application. Leveraging JavaScript's capabilities, Yulu enhances user interactivity by dynamically manipulating HTML and CSS elements, responding to user actions, and facilitating asynchronous communication with the server via AJAX requests. Additionally, the integration of modern JavaScript frameworks further enhances client-side scripting, providing streamlined solutions for DOM manipulation, event handling, and AJAX interactions, thereby enriching the overall user experience.

Moreover, the use of CSS frameworks like Bootstrap and Animate ensures consistent styling and engaging animations, contributing to a compelling and user-friendly interface in the Yulu app. These frameworks provide a foundation for responsive design, allowing the app to adapt seamlessly to different screen sizes and devices, while also enhancing visual appeal and usability. Together, these frontend technologies create an immersive and intuitive user experience, making the Yulu app a preferred choice for micro-mobility services.

Hardware and Platform Requirements:

3.3 Hardware and Platform Requirements

Hardware Requirements:

Standard server hardware with sufficient processing power and memory to handle the expected workload.

Operating System:

Linux (Ubuntu 20.04 LTS recommended): The preferred operating system for hosting the Yulu application due to its stability, security, and compatibility with the required software packages.

VNR VJIET

Name of the Experiment:_____

Name of the Laboratory:

_____

Experiment No:_____ Dates_____

Packages Installed:

Node.js: The JavaScript runtime environment for executing server-side code.

npm (Node Package Manager): The package manager for installing and managing Node.js packages.

Express.js: A minimalist web framework for building server-side applications and APIs.

MongoDB: A NoSQL database for storing and managing data related to user profiles, ride transactions, and vehicle availability.

React.js: A JavaScript library for building user interfaces, used for developing the frontend components of the Yulu app.

Redux: A predictable state container for managing application state in complex React.js applications.

Webpack: A module bundler for JavaScript applications, used for bundling and optimizing frontend assets.

Babel: A JavaScript compiler that allows developers to use modern JavaScript syntax and features, ensuring compatibility across different environments.

Axios: A promise-based HTTP client for making AJAX requests from the frontend to the backend.

json web token: A library for generating and verifying JSON Web Tokens (JWT) for user authentication and authorization.

bcryptjs: A library for hashing passwords securely, used for storing user passwords in a hashed format.

Helmet: A security middleware for Express.js applications, used for setting HTTP headers to improve security.

Mongoose: An object data modeling (ODM) library for MongoDB, used for defining schemas and interacting with the database.

Cors: A middleware for Express.js applications, used for enabling cross-origin resource sharing (CORS).

Body-parser: A middleware for Express.js applications, used for parsing request bodies.

dotenv: A zero-dependency module for loading environment variables from a .env file into process.env.

PM2: A process manager for Node.js applications, used for managing application processes and ensuring high availability.

Nginx: A web server and reverse proxy server, used for serving static files and proxying requests to the Node.js backend.

These software packages and dependencies are essential for developing, deploying, and running the Yulu application, ensuring its functionality, security, and performance.

3.4 System Connectivity

The Yulu app's client-side interface communicates with the server-side using HTTP protocols over the internet. Hosted on designated servers, the Yulu server exposes RESTful APIs that the client-side interface interacts with for various micro-mobility operations. Both client and server systems require internet access, with necessary ports available for HTTP communication, ensuring seamless connectivity between components. Additionally, potential integration of WebSockets enables real-time communication, while geolocation services enhance location-based functionalities. Overall, robust system connectivity ensures a responsive user experience for Yulu app users.

4.Standards

4.1 Performance Standards

The app aims for fast loading times for content and resources, prioritizing efficiency, particularly in areas with limited internet connectivity. It ensures optimal performance across various devices by efficiently utilizing device resources such as CPU, memory, and battery. Additionally, the implementation of caching mechanisms helps reduce data usage and improves app responsiveness, enhancing the overall user experience. By adhering to these performance standards, Yulu ensures seamless operation and satisfaction for its users, regardless of their location or device capabilities.

4.2 Disaster Recovery

Yulu implements disaster recovery measures by storing all resources across multiple servers. This redundancy ensures that resources can be recovered swiftly in the event of a system failure. By distributing resources across multiple servers, Yulu enhances resilience and minimizes downtime, ensuring uninterrupted service availability and user satisfaction.

4.3 Reliability Requirements

The Yulu app is designed to achieve a high level of reliability to ensure uninterrupted service for users. The system aims for a system uptime of at least 99.9% over a 30-day period, excluding scheduled maintenance windows. To maintain data integrity, mechanisms are implemented to prevent data loss or corruption in case of system failures or crashes.

Additionally, regular backups of user data and content are provided to ensure recoverability in the event of data loss or catastrophic failures. These reliability measures ensure that users can rely on Yulu for their micro-mobility needs with confidence in the system's stability and data integrity.

5.Support

Yulu offers comprehensive support through various documentation, including the source code, design document, operations manual, and deployment plan. The design document provides insights into the system's architecture and functionalities, while the operations manual guides users on proper system usage. Additionally, the deployment plan outlines the process for setting up and configuring the Yulu application. With these resources, stakeholders can effectively understand, operate, and maintain the system, ensuring a seamless experience for all users.