Mid Way Report

on

Payment Engine

Faculty Mentor:

Javed Imran

Industrial Mentor:

Gaurav Jain

Submitted by: Paras Bakshi

Roll No: *101917118*

Computer Science and Engineering Department
Thapar Institute of Engineering & Technology, Patiala

1. Introduction:

Tata 1 Mg's Payment service operates within the Vyked Backend framework, which is connected to a PostgreSQL database. This framework was specifically designed by Tata 1 Mg to manage payment services and has the capability to integrate with databases, while also performing high-speed caching with Redis cache. The operations within this framework are based on Synchronous programming, which is optimal for programming reactive systems due to its blocking architecture. As a single-thread model, operations are performed in a specific order, one at a time. However, introducing Asynchronous programming as a technique allows programs to start potentially long-running tasks and remain responsive to other events while the task runs, reducing the lag time between when a function is called and when its value is returned. This results in a faster and smoother workflow in the real world.

2. Problem Statement:

Vyked Backend framework, which is based on synchronous programming. As a result, this can cause a bottleneck in the application's performance and scalability, particularly in situations where multiple requests are made concurrently. Furthermore, the synchronous nature of the Vyked Backend framework can also result in poor responsiveness and high latency, as the program must wait for each function to finish executing before responding to user requests. This can be particularly problematic in applications that require real-time communication or low latency, such as chat or video conferencing platforms. To address these issues, many modern frameworks have shifted towards asynchronous programming models, such as event-driven architectures, that allow for non-blocking I/O and concurrent processing of requests. By adopting these models, applications can achieve higher throughput, lower latency, and better responsiveness, making them more suitable for modern, high-performance use cases.

3. Tools and Technologies used:

- Use of Sanic which is a Python web framework that is built specifically for asynchronous programming. Unlike traditional synchronous frameworks, Sanic leverages non-blocking I/O to handle multiple requests concurrently, making it well-suited for high-performance and real-time applications that require low latency and high throughput. By using asyncio, the core of Sanic allows for the creation of fast and efficient network services.

- Used PostgreSQL Toise ORM in project can which provide several benefits, depending on the specific requirements of the project. Some of the potential advantages of using PostgreSQL Toise ORM because ease of use, flexible, scalable and PostgreSQL is known for its robustness, reliability, and data integrity features.

- Pytest which is a popular testing framework for Python that allows developers to write and run tests easily and efficiently. When it comes to testing APIs, pytest provides several built-in features that make testing APIs easier and more effective.

- Interacted with the Juspay and Paytm are third-party payment gateway APIs that are integrated in our project to enable payment processing using various payment methods like wallets and cards.

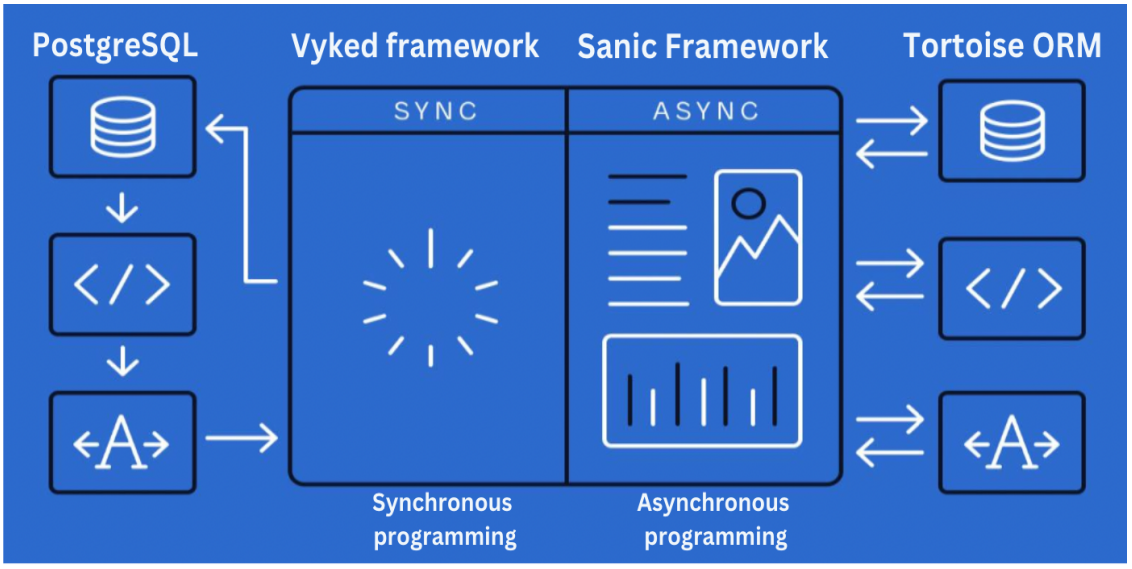
- Used Postman which is a popular tool used for testing and validating APIs. It allows developers to send HTTP requests to APIs and receive responses in real-time.

- Excel skills in Confluence which is a popular collaboration and documentation tool used by many organizations and Spotlight for API contracts refers to using a tool or platform to help manage and ensure compliance with the contracts that govern the use of APIs.

- Used Devtron which is a tool for managing and monitoring our Payment Engine application. It provides features like real-time monitoring, log aggregation, and automated health checks of applications in a pre-stage environment.

4. Methodology

The initial Payment service was based on the synchronous Vyked framework, which means that requests and responses were handled in a linear, step-by-step manner. However, the new project Payment Engine is based on Sanic, which is an asynchronous programming framework. This means that the service can handle multiple requests at the same time, without having to wait for each request to be completed before moving on to the next one.



Synchronous Vyked vs Asynchronous Sanice

In addition, In Payment Engine integrates with Tortoise ORM, which is an object-relational mapper that makes it easier to interact with a database in an object-oriented way. Overall, Payment Engine project has a more efficient and scalable architecture, allowing it to handle more requests and respond more quickly to user interactions.