

PROJECT SEMESTER REPORT

Payment Engine

by

Paras Bakshi

Roll No. 101917118

Under the Guidance of

Gaurav Jain - SDE1 at TATA 1MG

Dr. Javed Imran - Assistant Professor



Submitted to the

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

In Partial Fulfilment of the Requirements for the Degree of

Bachelor of Engineering in Computer Engineering

at

Thapar Institute of Engineering & Technology, Patiala

June 2023

Payment Engine

by Paras Bakshi

Place of work: TATA 1MG

Submitted to the Computer Science & Engineering Department, Thapar Institute of Engineering & Technology

June 2023

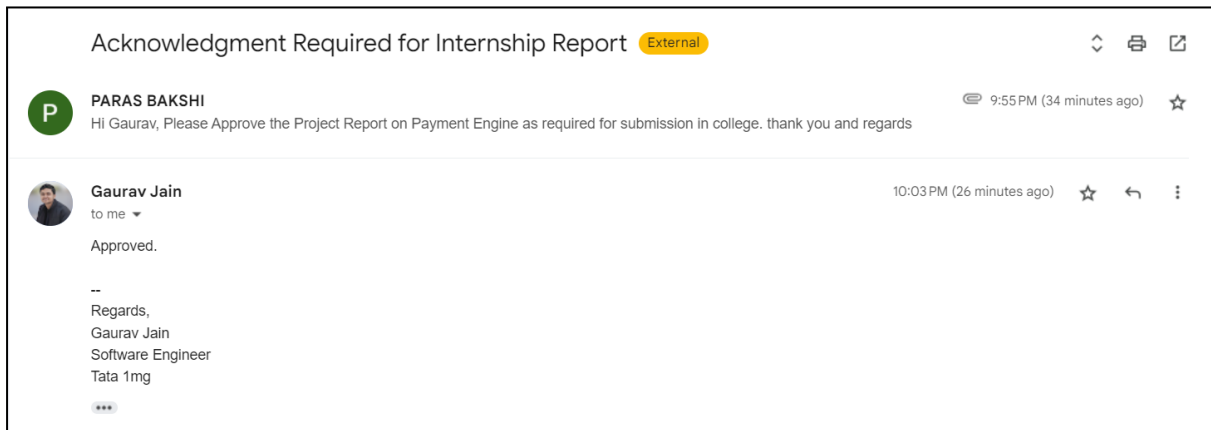
In Partial Fulfilment of the Requirements for the Degree of Bachelor of Engineering in
Computer Engineering

Abstract:

This report is based on the ongoing internship at TATA 1mg. Tata 1 Mg's Payment service is operating within the Vyked Backend framework and is connected to a PostgreSQL database. This framework was specifically developed by Tata 1 Mg to manage payment services and can integrate with databases while performing high-speed caching with Redis cache. The operations in this framework are based on Synchronous programming, which is ideal for programming reactive systems due to its blocking architecture. As a single-thread model, operations are performed one at a time in a specific order. However, the introduction of Asynchronous programming is a technique that enables programs to start potentially long-running tasks and still be responsive to other events while the task runs, reducing lag time between when a function is called and when its value is returned. This translates to a faster and more seamless flow in the real world.

Author Paras Bakshi

Certified by



Gaurav Jain
SDE1 TATA 1MG
(Industrial mentor)


A handwritten signature in black ink, appearing to read "G. Jain", with a long horizontal line extending to the right.

Certified by

Dr. Javed Imran
Assistant Professor
Computer Science & Engineering Department,
Thapar Institute of Engineering & Technology
(Faculty mentor)

CERTIFICATE (PROJECT SEMESTER TRAINING) FROM THE COMPANY OR THE ORGANIZATION

Welcome to TATA 1MG! Inbox x



Neeraj Kumar Singh <neeraj.singh@1mg.com>
to me ▾

Mon, 9 Jan, 17:57

Hi,

Congratulations on being a part of the TATA 1mg family.
We welcome you on-board and hope to have a long and successful journey together.

Please save the below information for your reference:

Employer Name	: TATA 1MG Healthcare Solutions Pvt. Ltd.
Employee Code	: INT1MGHC00144
Name	: Paras Bakshi
Department	: Technology
Designation	: Intern
Contact Number	: 8360385290
Location	: Gurgaon
Email ID	: int-paras.bakshi@1mg.com
Email ID Password	: QW4%&Dn7T&Fb8K%
HRBP Name	: Divya Jain
HRBP Email	: divya.jain1@1mg.com
IT Ticket Number	: 44716 (If you are yet to collect your Laptop, reach out to: Vicky Kumar, Presidency Tower-B, Floor No -05, Gurgaon, Phone- 7503485009)

Important - Fill the below form using your official email id to get your company ID Card: <https://forms.gle/gdZxzj1kA3ZRvgBH6>

Once you have activated your int-paras.bakshi@1mg.com email account using gmail, login to our internal Messaging/Chat application Flock by [clicking here](#).

To access your employee data, read Tata 1mg policies or apply for leaves, etc. - login to Employee Self Service portal Darwinbox <https://1mg.darwinbox.in/> which will get activated within 24 hours (post document verification).

Let us know in case of any discrepancy in above mentioned details.

Regards,
Neeraj Kumar Singh
Human Resource



Date: 15-12-2022

Dear Paras Bakshi,

This is to inform you that we, TATA 1MG Healthcare Solutions Private Limited, do hereby appoint you as an Intern in our organization.

Offer Details:

Date of Joining	09-01-2023
Designation (Title)	Intern
Stipend (Per Month)	INR 30,000
COE	Engineering
Department	Technology
Sub Department	Engineering
Internship Duration	6 Months

Documents:

You are requested to bring along the following documents on your date of joining and hand over (photocopy only) as mentioned below: -

Passport Size Photograph – 3 nos.

Self-attested documents of all your educational and degree certificate (10th to Highest qualification)

Permanent & Temporary Address proof Copy of Pan Card ID proof (Voter ID Card / Passport / Driving License / Aadhar Card)

NOC letter From College Authority

Kindly return the duplicate copy of this offer letter, signed as a token of your acceptance.

For TATA 1MG Healthcare Solutions Private Limited,

Tanmay Saksena
Authorized Signatory

TATA 1MG HEALTHCARE

Tata 1mg Healthcare Solutions Pvt. Ltd. (formerly known as 1MG Healthcare Solutions Private Limited and Delhi Mediart Private Limited) Registered Office: Level 3, Vasant Square Mall, Pocket V, Sector B, Vasant Kunj, South Delhi, New Delhi-110070, India

TABLE OF CONTENT

Contents

1. Company Profile	6-9
2. Introduction	10-11
3. Background	12-13
4. Objectives	14-15
5. Methodology	16-25
6. Observations and Findings	26-27
7. Limitations	28-29
8. Conclusions and Future Work	30-32
9. Bibliography/References	33-34
10. Peers Review Form	35

1. Company Profile

1.1 Overview



Figure 1: Company Logo

Name: Tata 1MG

Headquarters: Gurgaon, Haryana, India

Founded: 2015

Founder: Prashant Tandon, Gaurav Agarwal & Vikas Chauhan

Parent Company: Tata Digital (Majority stake)

Tata 1mg.com brings to you an online platform, which can be accessed for all your health needs ranging from products, services and healthcare information. They are trying to make healthcare a hassle-free experience for you. Get your allopathic, ayurvedic, homoeopathic medicines, vitamins, nutrition supplements and other health-related products delivered in the comfort of your home. The company also provides consistent services including Lab Tests, Online Doctor Consultations and even specific plans for corporate wellness.

1.2 Industrial Role

Healthcare has become one of India's largest sectors, both in terms of revenue and employment. Healthcare comprises hospitals, medical devices, clinical trials, outsourcing, telemedicine, medical tourism, health insurance and medical equipment. The Indian healthcare sector is growing at a brisk pace due to its strengthening coverage, services and increasing expenditure by public as well as private players.

1.3 History

In 2012, BLPL (Bright Lifecare Private Limited) started a digital health platform called Healthkart Plus. In 2015, Healthkart Plus separated from BLPL to form 1MG Technologies Private Limited. Healthkart continued as a platform for fitness and health products. 1 mg was a generic drug search business.

In healthcare, the information about medicines and lab tests is either unavailable or incomprehensible to ordinary individuals. So the founders decided to create a platform that stood for transparent, authentic and accessible information for all; hence the platform maintains an online medicine database with information on side effects; generic substitutes and provides home delivery services for pharmacy, FMCG and lab tests.

In June 2021, Tata Digital Ltd acquired a majority stake in 1mg to form Tata 1mg.

1.4 Our Work & Culture

At Tata 1mg we strongly believe that a great culture is an important ingredient for a start-up's success. Our culture promotes radical candour, fast-paced iterations, collaboration and a flat hierarchy. Listed below are the core values that enshrine our culture.

- **Be the CEO of your own outcomes**
- **Done is better than perfect**
- **Team and not individual**
- **Accountability with empathy**

These important core values have led to the organisation being recognised consistently for its efforts and practices including several awards and accolades both in the start-up and healthcare industry.

1.5 Our Services & the Role of Business Intelligence

As a key player in the market Healthcare Tata 1mg has worked to provide numerous services to all of its customers. The following is an overview of each of them:



Figure 2: Our Services

1.5.1 E-Pharmacy

The most vital service provided by the company is the E-Pharmacy service which lets customers order both pharmaceutical and non-pharmaceutical healthcare products from its website. There are proper processes in place for the ordering of RX (prescription) and

OTC(Over the Counter) allopathic, ayurvedic, homoeopathic medicines, vitamins, nutrition supplements and other medical products

1.5.2 E-Consultations

This service offers customers the chance to connect with doctors virtually and get proper consultations from registered and certified experts. The service includes the provision of prescriptions for medicines and tests.

1.5.3 Diagnostics

1mg provides an easy to access and hassle-free diagnostic service to the customers. This ranges from at-home testing, efficient test reporting mechanisms and trustworthy expert analysis of the reports. The company provides all major lab tests to its customers including full body check-ups.

1.5.4 Authentic Information Database

The company maintains an extensive and detailed database of healthcare-related content focused on medicines and other needs written by qualified doctors and medical professionals. The content covers a wide range of topics from medicine salt information to recommendations from experts. It helps to simplify healthcare for the common man.

1.5.5 Patient Support Program

The Patient Support Programme, also known as the Patient Assistance Programme, is a premium programme that the company offers which includes schemes aimed to improve the accessibility and the availability of rare drugs for patients in need of them.

1.5.6. Corporate Wellness

This programme allows for the company to provide various organisations and companies with a holistic programme/product for employee wellbeing by creating an organizational

culture of health. This is done through proper platforms, ease of access to 1MG services and overall provision of healthcare resources.

2. 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.

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.

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.

2.1 Advantages

The Payment Engine project bring several advantages:

- 1. Improved Performance:** The shift from the synchronous Vyked framework to the asynchronous Sanic framework enables the Payment Engine to handle multiple requests concurrently. This asynchronous processing capability significantly improves performance

and reduces response times. With the ability to handle multiple requests simultaneously, the system can effectively utilize available resources and maximize throughput.

2. Scalability: The asynchronous nature of the Sanic framework allows the Payment Engine to scale more efficiently. By eliminating the need to wait for each request to complete before moving on to the next one, the system can handle a larger number of concurrent requests without sacrificing performance. This scalability is crucial for accommodating increased user demand and ensuring a smooth experience even during peak usage periods.

3. Enhanced User Experience: The improved performance and scalability of the Payment Engine contribute to a better user experience. Users can expect faster response times, minimizing waiting periods and increasing overall satisfaction. The ability to handle multiple requests concurrently also means that the system can support a higher volume of user interactions without experiencing delays or slowdowns, providing a responsive service.

4. Simplified Database Interaction: The integration of Tortoise ORM in the Payment Engine simplifies the interaction with the database. Tortoise ORM serves as an object-relational mapper, allowing developers to interact with the database using an object-oriented approach. This simplification enhances code maintainability and readability while reducing the potential for errors. By leveraging Tortoise ORM, developers focus more on business logic and application functionality rather than low-level database operations.

5. Greater Efficiency: The overall architecture of the Payment Engine project is designed for efficiency. The combination of asynchronous processing, improved scalability, and simplified database interaction leads to optimized resource utilization and reduced overhead. The system can handle a higher volume of requests with fewer resources, resulting in cost savings and improved operational efficiency.

In summary, the operation changes implemented in the Payment Engine project bring advantages such as improved performance, scalability, enhanced user experience, simplified

database interaction, and greater efficiency. These benefits collectively contribute to a more robust and responsive payment service.

3. Background

The Payment Engine project aims to enhance the existing payment service by introducing a new architecture based on asynchronous programming and improved database interaction. This section provides the necessary context for the project and outlines the proposed layout for achieving the project goals.

3.1 Introduction

The initial Payment service, built on the synchronous Vyked framework, followed a linear, step-by-step approach to handle requests and responses. However, as the user base and transaction volume grew, it became apparent that the existing architecture had limitations in terms of performance and scalability.

3.2 Limitations of the Existing Architecture

The synchronous nature of the Vyked framework restricted the system's ability to handle multiple requests simultaneously. Each request had to be completed before moving on to the next one, resulting in longer response times and potential bottlenecks during peak usage periods. Additionally, the traditional database interaction methods posed challenges in terms of code maintainability and flexibility.

3.3 Proposed Layout for the Payment Engine

To address the limitations of the existing architecture, the Payment Engine project proposes a new layout that leverages the advantages of asynchronous programming and improved database interaction. The following components form the foundation of the proposed layout:

1. **Asynchronous Programming Framework:** The project adopts the Sanic framework, which offers asynchronous capabilities, allowing the system to handle multiple requests

concurrently. This shift enables improved performance, reduced response times, and increased scalability to accommodate growing user demands.

2. Integration with Tortoise ORM: The Payment Engine integrates with Tortoise ORM, an object-relational mapper that simplifies database interactions in an object-oriented manner. By utilizing Tortoise ORM, developers can focus on business logic and benefit from enhanced code maintainability and readability.

3.4 Motivation for Choosing the Project

The decision to undertake the Payment Engine project was motivated by several factors. First and foremost, the need for a more efficient and scalable architecture became evident as the existing payment service faced performance limitations. Addressing these limitations would enhance the user experience and ensure smooth operations even during peak load periods.

Furthermore, the integration of modern frameworks and tools, such as the Sanic framework and Tortoise ORM, offered opportunities for code optimization and improved development practices. By adopting these technologies, the project aimed to streamline the development process, reduce maintenance efforts, and increase overall efficiency.

In conclusion, the Payment Engine project seeks to overcome the limitations of the existing payment service by introducing a new architecture based on asynchronous programming and improved database interaction. The proposed layout, involving the adoption of the Sanic framework and integration with Tortoise ORM, aims to achieve enhanced performance, scalability, and overall efficiency. The decision to undertake this project was driven by the motivation to improve the user experience, optimize development practices, and address the evolving needs of a growing user base.

4. Objectives

1. Improved Performance: The objective of improving performance in the Payment Engine project entails reducing response times and minimizing delays. By transitioning to the asynchronous Sanic framework, the system can handle multiple requests concurrently, resulting in faster processing and reduced waiting times for users. This improvement in performance ensures a more efficient and responsive payment service.

2. Increased Scalability: The project aims to design and implement a scalable architecture for the Payment Engine. The adoption of an asynchronous programming framework allows the system to handle a higher volume of concurrent requests without compromising performance. This scalability ensures that the payment service can accommodate the growing user demand and scale seamlessly as the user base expands, providing a consistent experience even during peak usage periods.

3. Efficient Resource Utilization: The Payment Engine project focuses on optimizing resource utilization. By leveraging asynchronous programming, the system can effectively utilize available resources by handling multiple requests concurrently. This approach ensures efficient use of computing resources, reducing the need for additional hardware or infrastructure. Furthermore, the integration with Tortoise ORM simplifies database interactions, reducing unnecessary overhead and improving overall resource efficiency.

4. Seamless User Experience: The objective of providing a seamless user experience involves eliminating waiting periods and ensuring smooth operations throughout the payment process. With improved performance and scalability, users can expect faster response times and minimal delays when initiating and completing payment transactions. The Payment Engine project aims to create a frictionless experience for users, enhancing their satisfaction and trust in the payment service.

5. Simplified Development and Maintenance: The project aims to simplify the development and maintenance of the Payment Engine. The integration with Tortoise ORM allows developers to interact with the database in an object-oriented manner, simplifying database operations and reducing the potential for errors. This streamlined approach improves code maintainability and readability, making it easier to modify, update, and maintain the payment service over time. Additionally, simplified development practices reduce the time and effort required for future enhancements and feature additions.

6. Adherence to Industry Standards and Security: The Payment Engine project emphasizes adherence to industry standards and robust security measures. This includes implementing secure payment protocols, encrypting sensitive data, and following best practices for handling financial transactions. By ensuring the security and integrity of user data, the project instills trust in the payment service and protects both the organization and its customers from potential security breaches or fraud.

By achieving these objectives, the Payment Engine project aims to create a high-performing, scalable, and secure payment service that provides a seamless user experience while reducing development and maintenance complexities.

5. Methodology

5.1 Training

The comprehensive bootcamp conducted prior to the project launch was a vital component in ensuring my readiness and proficiency in relevant areas. The primary objective of the bootcamp was to equip me with the necessary training and skills required for the successful implementation of the Payment Engine project. The sessions were thoughtfully designed to provide a solid grasp of key concepts and technologies essential to the project's scope.

The bootcamp encompassed a wide range of topics that were directly related to the project's requirements. I actively participated in classroom sessions that covered important subjects such as Git, Linux, Python, databases, DevOps, AWS, frontend development, and backend development. These sessions were structured to encourage interactive discussions, hands-on exercises, and practical assignments, allowing me to deepen my understanding and proficiency in each domain.

Throughout the training, the curriculum was meticulously crafted to align with the project's objectives, ensuring that I gained the necessary knowledge and skills to contribute effectively to the implementation of the Payment service at Tata 1MG. The sessions were conducted by experienced faculty members, industry experts, and guest speakers who shared their invaluable insights and real-world experiences. This exposure to diverse perspectives not only enriched my learning but also provided me with practical insights into industry best practices.

Moreover, the bootcamp fostered a collaborative learning environment, facilitating interactions with other students who were also working on related projects. This collaborative setting promoted knowledge sharing, teamwork, and peer learning. Engaging with fellow students allowed me to exchange ideas, tackle challenges collectively, and further enhance my problem-solving abilities.

Overall, the bootcamp played a crucial role in equipping me with the necessary skills, knowledge, and collaborative mindset required to contribute effectively to the successful implementation of the Payment Engine project at Tata 1MG.

5.2 Selection of Tools and Technologies

5.2.1 Sanic

The Payment Engine leverages Sanic, a Python web framework specifically designed for asynchronous programming. Sanic's non-blocking I/O capabilities enable the handling of multiple requests concurrently, making it ideal for high-performance and real-time applications that require low latency and high throughput. By utilizing `asyncio`, the core of Sanic, the Payment Engine achieves fast and efficient network services.

5.2.2 PostgreSQL Tortoise ORM

The Payment Engine project incorporates PostgreSQL Tortoise ORM, providing numerous advantages tailored to its requirements. The ease of use and flexibility of PostgreSQL Tortoise ORM enhance the development experience. With its scalability and robustness, PostgreSQL Tortoise ORM ensures reliable data storage and integrity within the Payment Engine.

5.2.3 Pytest

Pytest, a popular testing framework for Python, is utilized extensively in the Payment Engine project. It simplifies the writing and execution of tests, allowing developers to create comprehensive test suites efficiently. Pytest's built-in features for API testing enhance the effectiveness and reliability of testing the Payment Engine's APIs.

5.2.4. Third-party Payment Gateway APIs (Juspay and Paytm)

The Payment Engine integrates third-party payment gateway APIs, namely Juspay, Paytm.

These APIs enable secure and seamless payment processing, supporting various payment methods such as wallets and cards.

5.2.5 Postman

Postman, a widely adopted tool for API testing and validation, plays a crucial role in the development of the Payment Engine. It allows developers to send HTTP requests to the Payment Engine's APIs and receive real-time responses, facilitating thorough API testing and validation.

5.2.6 Excel Skills in Confluence and Spotlight for API Contracts:

Excel skills within Confluence, a collaboration and documentation tool, are employed for efficient documentation and team collaboration throughout the Payment Engine project. Spotlight for API Contracts assists in managing and ensuring compliance with the contracts governing the use of APIs within the Payment Engine.

5.2.7 Devtron

Devtron serves as a valuable tool for managing and monitoring the Payment Engine application. Its features, including real-time monitoring, log aggregation, and automated health checks, contribute to maintaining the optimal performance and reliability of the Payment Engine.

By utilizing these tools and technologies, the Payment Engine project benefits from an efficient and scalable architecture, secure payment processing, comprehensive testing capabilities, effective documentation, and robust application management and monitoring.



Figure 3 Tools and Technologies

5.3 Problem Solving and Research

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.

As 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.

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.

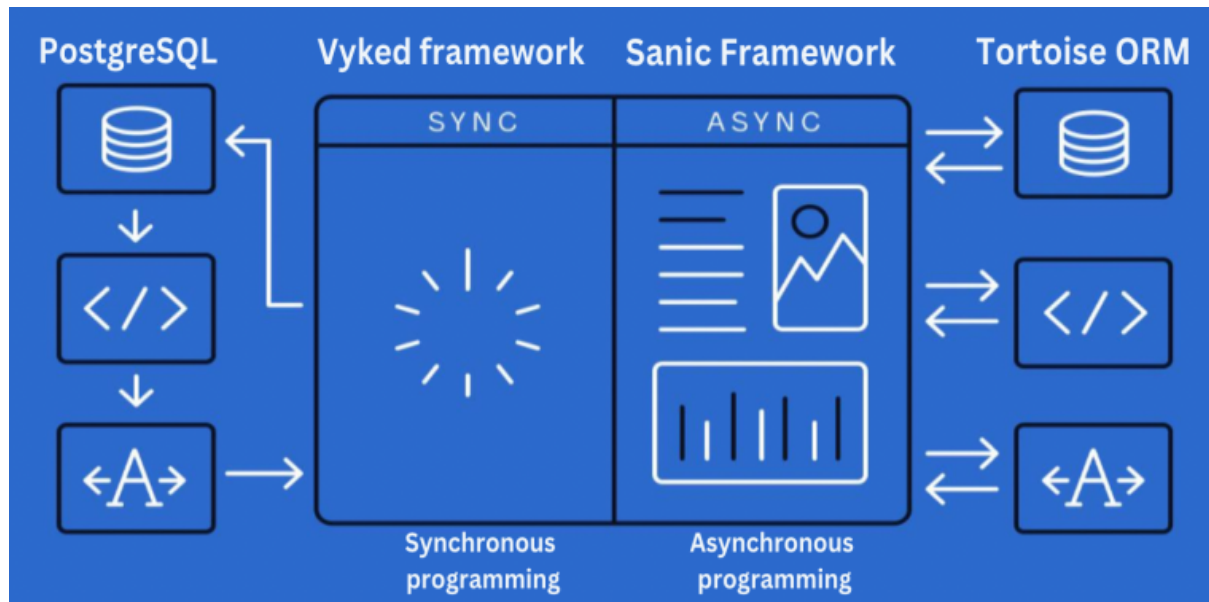


Figure 4 Synchronous Vyked framework vs asynchronous Sanic

5.4 Design and Development

The design and development of the Payment Engine involved a systematic approach to create a robust and efficient payment processing system. Here is an overview of the design and development process:

5.4.1 Requirements Gathering

The project team collaborated with stakeholders to gather and analyze the requirements for the Payment Engine. The requirements included functionality, performance, security, integration with third-party payment gateways, and scalability.

5.4.2 System Architecture Design

Based on the requirements, the system architecture of the Payment Engine was designed. The architecture incorporated components such as the user interface, API layer, business logic, data storage, and integration with external payment gateways. The design focused on achieving high performance, scalability, and security.

5.4.3 Database Design

The database schema for the Payment Engine was designed to store and manage transaction data, user information, and payment details. The design ensured data integrity, optimized query performance, and supported future scalability.

5.4.4 Development Iterations

The development process followed an agile methodology, breaking the project into iterative sprints. Each sprint focused on delivering specific features and functionalities of the Payment Engine. Developers wrote clean and modular code, following coding best practices and coding standards.

5.4.5 Integration with Third-Party Payment Gateways

The Payment Engine was integrated with third-party payment gateway APIs, such as Juspay and Paytm. API integrations allowed for seamless and secure payment processing using various payment methods.

5.4.6 Testing and Quality Assurance

Comprehensive testing was conducted throughout the development process. Unit tests, integration tests, and end-to-end tests were performed to ensure the functionality, reliability, and security of the Payment Engine. Quality assurance processes were followed to identify and fix any bugs or issues.

5.4.7 Security Measures

Security measures were implemented to protect sensitive user data and ensure secure payment transactions. Measures included data encryption, secure authentication and authorization, input validation, and protection against common security vulnerabilities.

5.4.8 Documentation

Detailed documentation was created to aid in the understanding, maintenance, and future enhancements of the Payment Engine. Documentation included system architecture diagrams, API documentation, database schema documentation, and user guides.

5.4.9 Deployment and Monitoring

The Payment Engine was deployed to a production environment following best practices. Continuous monitoring tools, such as Devtron, were used to monitor the performance, health, and availability of the application. Any issues or performance bottlenecks were identified and addressed promptly. The design and development of the Payment Engine followed a structured approach, ensuring the system met the requirements, was scalable, secure, and capable of processing payments seamlessly.

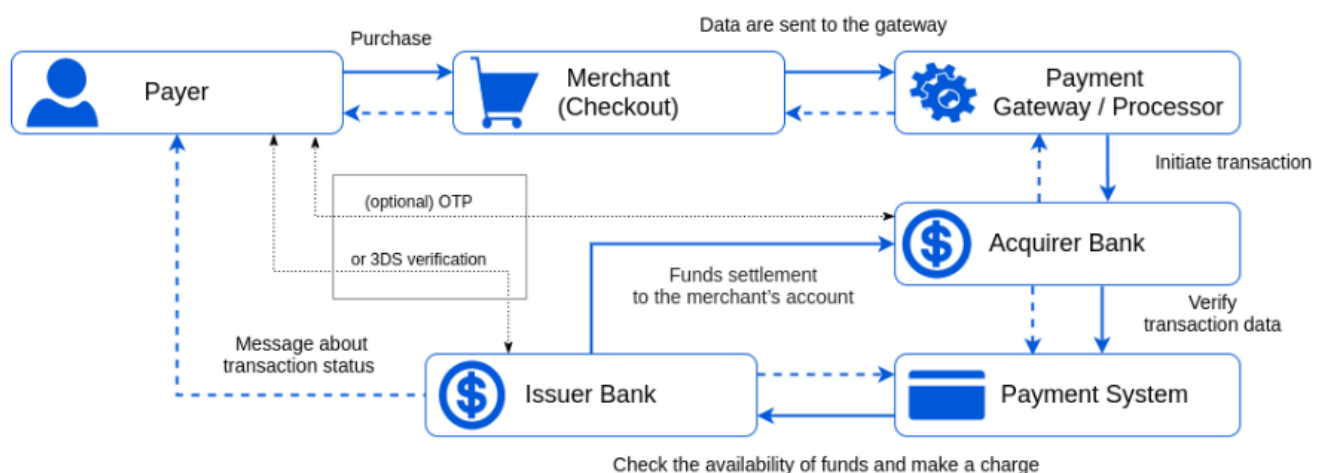


Figure 5 Overview of Payment Engine Design

5.5 Documentation and Knowledge Sharing

In the Payment Engine project, documentation and knowledge sharing played a crucial role in ensuring effective collaboration, seamless onboarding of new team members, and long-term maintainability of the system. To achieve these goals, a comprehensive approach was taken, leveraging various tools and practices.

One of the key aspects of documentation was the generation of detailed documentation for all routes within the Payment Engine. Using [spotlight.io](#), an intuitive documentation platform, all routes, including their endpoints, parameters, request/response structures, and authentication requirements, were meticulously documented. This documentation served as a centralized reference guide that could be easily accessed by developers, testers, and other stakeholders. It provided a clear understanding of the system's functionality and served as a valuable resource for troubleshooting, debugging, and making informed decisions during the development and maintenance phases.

Furthermore, the documentation extended beyond technical aspects and encompassed broader project-related information. This included architectural diagrams, data flow diagrams, and integration guidelines with third-party payment gateways. By documenting these details, the Payment Engine project ensured that the knowledge was effectively captured and shared, reducing dependency on specific individuals and promoting a collaborative environment.

In addition to formal documentation, knowledge sharing was fostered through regular team meetings, code reviews, and collaborative tools such as Confluence. Team members actively shared their experiences, insights, and lessons learned during these sessions, allowing for the exchange of ideas and best practices. This not only enhanced the collective knowledge of the team but also encouraged continuous improvement and innovation within the project.

Moreover, the project team established a culture of documentation and knowledge sharing by encouraging team members to contribute to the project's wiki pages and share their learnings. This helped create a repository of valuable information, tips, and tricks that could be accessed

by the entire team, even after the project's completion. By documenting challenges, solutions, and implementation details, the team ensured that future developers and maintainers would have access to critical information, accelerating the learning curve and reducing the time required to understand and work on the system.

Overall, the emphasis on documentation and knowledge sharing in the Payment Engine project promoted transparency, collaboration, and long-term sustainability. By leveraging tools like spotlight.io and fostering a culture of sharing, the project team successfully captured and disseminated knowledge, enabling smooth project execution and empowering future teams to build upon the existing foundation.

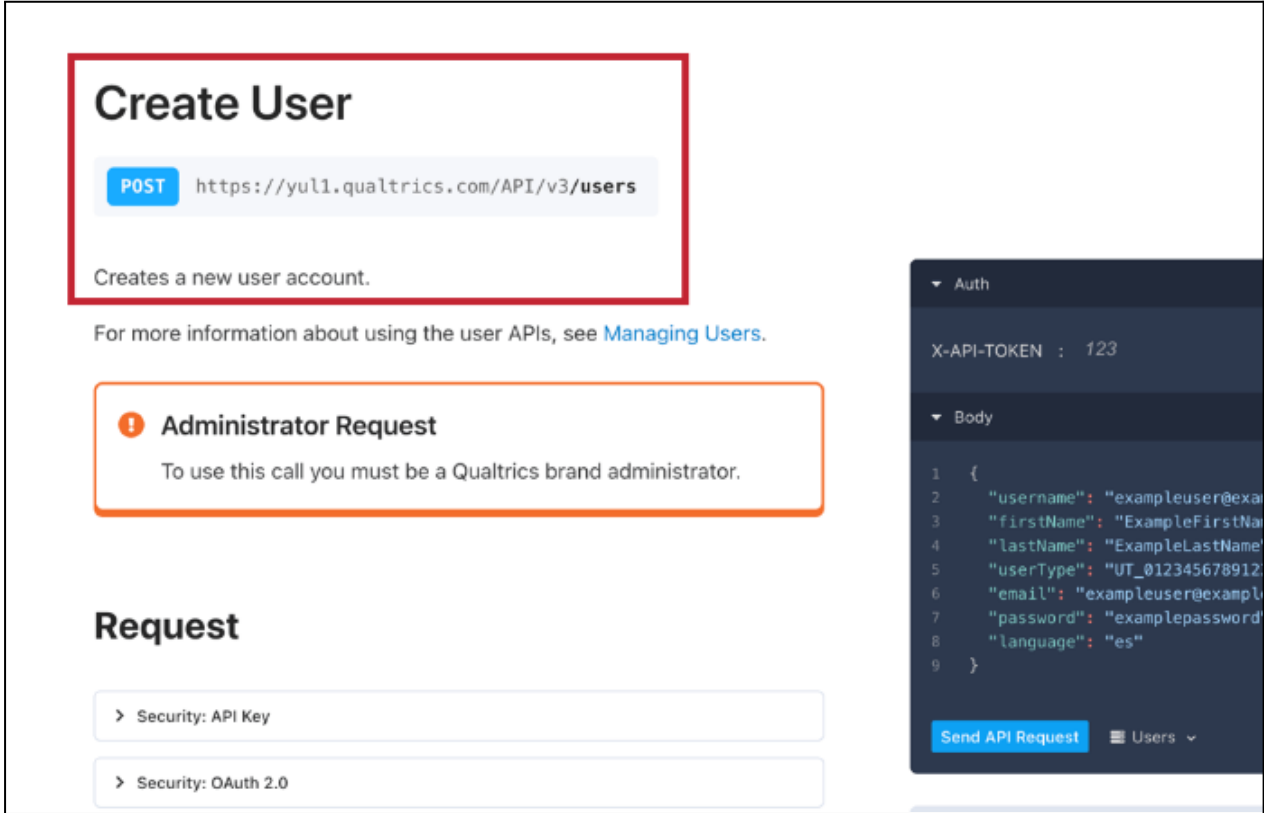


Figure 6 API Contracts for Create User in Juspay

GET	/A_PurchaseContract	Try out	Code Snippet
POST	/A_PurchaseContract	Try out	Code Snippet
GET	/A_PurchaseContract('{PurchaseContract}')	Try out	Code Snippet
GET	/A_PurchaseContract('{PurchaseContract}')/to_PurchaseContractItem	Try out	Code Snippet
GET	/A_PurchaseContract('{PurchaseContract}')/to_PurCtrPartners	Try out	Code Snippet
GET	/A_PurchaseContractItem(PurchaseContract='{PurchaseContract}',PurchaseContractItem='{PurchaseContractItem}')/to_PurchaseCo...	Try out	Code Snippet

Figure 7 Documentation of API Contracts

6. Observations and Findings

The implementation of the new Payment Engine, leveraging the asynchronous Sanic framework and modular code structure, led to several significant observations and findings that highlighted its improved efficiency and functionality.

One key observation was the enhanced performance and responsiveness achieved through the asynchronous nature of Sanic. By handling multiple requests concurrently, without the need to wait for each request to complete before processing the next one, the Payment Engine demonstrated improved efficiency and reduced latency. This allowed for a higher throughput of payment transactions, resulting in a smoother and more seamless user experience.

The modular code structure implemented in the Payment Engine also proved to be instrumental in improving efficiency. The codebase was divided into logical modules, each responsible for specific functionalities or components of the payment process. This modular approach made the code more organized, maintainable, and easier to understand. It also facilitated the identification and isolation of issues or bugs, enabling quicker debugging and resolution.

Another significant finding was the successful implementation and functioning of the full flow of payment within the Payment Engine. Through meticulous development and testing, all the essential steps of the payment process, including user authentication, payment method selection, transaction validation, and confirmation, were seamlessly integrated. This end-to-end flow demonstrated the robustness and reliability of the Payment Engine, ensuring that users could complete their payments securely and without any disruptions.

Furthermore, the implementation of proper error handling mechanisms and logging within the Payment Engine provided valuable insights into the system's behavior and potential areas for improvement. Error logs and exception handling allowed for quick identification and resolution of issues, reducing downtime and improving overall system stability.

Overall, the observations and findings highlighted the significant improvements brought about by the new Payment Engine. The adoption of the asynchronous Sanic framework, along with the modular code structure and successful implementation of the full payment flow, resulted in enhanced efficiency, improved performance, and a seamless user experience. The project's success in achieving these objectives showcased the effectiveness of the chosen technologies and development practices in creating a robust and efficient payment processing system.

7. Limitations

7.1 Learning Curve

One of the primary constraints faced during the implementation of the Project Payment engine was the intern's limited prior knowledge of concepts and tools used in TATA 1MG. With a project duration of four months, including a two-month bootcamp to learn the necessary skills, the learning curve was significant. This constraint impacted the speed and effectiveness of the implementation process, and initial challenges and delays were encountered due to the intern's limited experience in this domain.

7.2 Complexity of the Project

The Payment Engine encompasses a considerable level of complexity due to its multi-faceted nature and the integration of various components and technologies. The system involves intricate payment processing logic, encompassing authentication, transaction validation, and communication with third-party payment gateways. Additionally, the integration of asynchronous programming using Sanic and the use of PostgreSQL with Tortoise ORM adds another layer of complexity. Ensuring data integrity, handling concurrency, and implementing robust error handling mechanisms further contribute to the system's intricacy. The complexity of the Payment Engine demands meticulous design, thorough testing, and continuous monitoring to ensure seamless operation and reliable performance in processing a wide range of payment transactions.

7.3 Integration Challenges

The integration of the Payment Engine posed several challenges due to its interactions with various external systems and APIs. One significant challenge was the seamless integration with third-party payment gateways like Juspay and Paytm, requiring thorough understanding of their integration protocols, authentication mechanisms, and handling of different payment methods. Ensuring compatibility and smooth communication between the Payment Engine and these external systems required extensive testing, debugging, and coordination with the respective providers. Additionally, integrating with other internal systems and databases within the organization, while maintaining data consistency and security, presented its own

set of challenges. Overcoming these integration challenges necessitated meticulous planning, clear communication, and rigorous testing to achieve a cohesive and reliable payment processing system.

7.4 Scalability Constraints

While the Payment Engine has been designed to handle a significant number of concurrent requests, there may be scalability constraints under extreme load conditions. If the system experiences an unexpectedly high volume of concurrent requests or a surge in user traffic, it may face performance degradation or even fail to handle the load efficiently. Continuous monitoring and performance testing should be conducted to ensure optimal scalability.

7.5 Third-Party API Dependencies

The integration of third-party payment gateway APIs, such as Juspay and Paytm, introduces a level of dependency on their services. Any disruptions or issues with these APIs, such as downtime or changes in their integration requirements, may impact the Payment Engine's functionality. Regular communication and monitoring with the third-party providers are necessary to address any potential challenges promptly.

8. Conclusions and Future Work

In conclusion, the implementation of the Payment Engine has proven to be a significant milestone in streamlining and enhancing the payment processing capabilities. The adoption of the asynchronous Sanic framework, along with the modular code structure, has resulted in improved efficiency, scalability, and responsiveness. The integration with third-party payment gateways, such as Juspay and Paytm, has enabled a seamless and secure payment experience for users, supporting various payment methods. The comprehensive documentation and knowledge sharing practices have promoted collaboration and facilitated future maintenance and development. Despite the complexities and integration challenges, the Payment Engine has demonstrated its robustness and reliability in handling payment transactions. Moving forward, continuous monitoring, optimization, and adherence to security standards will be crucial to ensuring the sustained success of the Payment Engine and meeting the evolving needs of the users and the organization. The Payment Engine stands as a testament to the effective implementation of modern technologies and best practices in the realm of payment processing, empowering the organization to provide a seamless and efficient payment solution to its users.

I have been able to learn a lot both professionally and personally as stated below: 8.1 Personal Learning.

8.1 Personal Learning

The development and implementation of the Payment Engine have been an invaluable learning experience for me. Throughout the project, I had the opportunity to gain in-depth knowledge and hands-on experience with various technologies and practices. Working with the asynchronous Sanic framework taught me the importance of leveraging non-blocking I/O for improved performance and responsiveness. I gained proficiency in modular code development, understanding how it enhances code organization, maintainability, and scalability. Integrating with third-party payment gateways like Juspay and Paytm exposed me to the intricacies of API integration and the significance of seamless communication with external systems. Additionally, the documentation and knowledge sharing aspects emphasized the value of comprehensive documentation, effective collaboration, and continuous learning. This project allowed me to strengthen my problem-solving skills,

adaptability, and ability to work in a collaborative team environment. Overall, the Payment Engine project has significantly contributed to my professional growth, providing me with valuable insights and practical experience in the field of payment processing and software development.

8.2 Professional Learning

The development and implementation of the Payment Engine have been instrumental in my professional growth, providing me with valuable lessons and insights. Through this project, I had the opportunity to expand my technical expertise and skills in areas such as asynchronous programming, modular code development, and API integration. Working with the Sanic framework taught me the importance of performance optimization and efficient handling of concurrent requests. The integration with third-party payment gateways enhanced my understanding of external system integration and the need for seamless communication. Additionally, the emphasis on comprehensive documentation and knowledge sharing emphasized the value of effective collaboration, clear communication, and maintaining up-to-date documentation. This project also honed my problem-solving abilities, project management skills, and the ability to work in a dynamic and agile development environment. Overall, the Payment Engine project has played a pivotal role in my professional development, equipping me with valuable experiences and skills that will contribute to my future success in software development and related domains.

8.3 Future Work

While the current implementation of the Payment Engine has been successful in integrating with third-party payment gateways like Juspay and Paytm, there is still future work pending to expand its capabilities further. One of the key areas of development is the integration of additional payment aggregators such as Razorpay and PayU. This would allow users to have a wider range of options when it comes to choosing their preferred payment method, enhancing the flexibility and convenience of the Payment Engine.

Furthermore, as the payment industry continues to evolve, there may be the introduction of new payment networks or technologies. It is essential for the Payment Engine to stay updated and adaptable to these emerging trends. This would involve researching and integrating new payment networks or technologies that offer enhanced security, efficiency, and support for evolving payment methods.

The integration of new payment aggregators and the incorporation of emerging payment networks will require thorough research, understanding of integration protocols, and close collaboration with the respective service providers. It will also involve rigorous testing and validation to ensure seamless interoperability and security.

By expanding the Payment Engine's capabilities to include more payment aggregators and adapting to new payment networks, the system can remain at the forefront of the payment processing industry, providing users with a diverse range of payment options and staying aligned with the evolving needs of the market.

9. Bibliography/References

- [1] Amazon s3. <https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html>. Accessed on February 23, 2023.
- [2] Continuous integration. <https://semaphoreci.com/cicd>. Accessed on February 23, 2023.
- [3] Openapi specification. <https://swagger.io/specification/>.
- [4] Sanic framework. <https://sanic.dev>. Accessed on February 23, 2023.
- [5] Python. <https://www.python.org/doc/>, n.d. Accessed on February 23, 2023.
- [6] Ambler, S. W. Agile development: Iterative and incremental models. In Wiley encyclopedia of computer science and engineering. Wiley, 2009, pp. 74–88.
- [7] CNCF. Kubernetes. <https://kubernetes.io/>. Accessed on Mar 13, 2023.
- [8] Fowler, M. Python Concurrency with AsyncIO. Manning, 2022.
- [9] Google. asynctest. <https://asynctest.readthedocs.io/en/latest/>. Accessed on Mar 13, 2023.
- [10] Google. Google single sign on. <https://cloud.google.com/architecture/identity/single-sign-on>. Accessed on Mar 13, 2023.
- [11] Hopkins, A. Python Web Development with Sanic. Packt Publishing, 2022.
- [12] Jaisingh, M. Software Development Life Cycle (SDLC), 3rd ed. Tata McGraw-Hill Education, 2011.

Annexure A. Evaluation Form for Peer Review

Name of the student: (to be reviewed)	Paras Bakshi	Roll no. of the student:	101917118
<p><i>This form has to be submitted by the student whose roll no. will be mentioned in the box above. Handover this to the panel at the time of final presentation.</i></p>			
Title of the project:	Payment Engine		
Name of the company:	TATA 1MG		
Project report (Tick the appropriate)	Excellent •	Good	Average
Project poster (Tick the appropriate)	Excellent •	Good	Average
Project video (Tick the appropriate)	Excellent •	Good	Average
Rate the work done	0 – 10 points	(Provide rating here) →	10
Give marks to the student on the basis of the overall performance	0 -5 marks	(Provide marks here) →	5
<p>Abstract of the project (max. 100 words): Tata 1 Mg's 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. Overall, Payment Engine project has a more efficient and scalable architecture, allowing it to handle more requests and respond more quickly to user interactions.</p>			
<p>Mention three strengths of the work done:</p> <ol style="list-style-type: none"> 1. Introducing Asynchronous programming as a technique allows programs to start potentially long-running tasks and remain responsive to other events while the task runs. 2. Reducing lag time between when function is called and with full test coverage. 3. Payment Engine integrates with Tortoise ORM, that makes it easier to interact with a database in an object-oriented way. 			
<p>Provide some useful recommendations (It may be some improvements, some suggestions to further raise the quality of the project):</p> <ol style="list-style-type: none"> 1. Try to achieve maximum test coverage as possible. 2. Conducted quality assurance prior to the release for production. 			
Name of the evaluator student:	Shikha Rani	Roll no. of the evaluator student:	101903629
Signature of the Evaluator student:	