

SYNOPSIS MAJOR PROJECT

Automated and Scalable Platform for Subscription Management with Alerts

(ES-452: Major Project - Dissertation)

*Submitted in partial fulfillment of the requirements
for the award of the degree of*

Bachelor of Technology Computer Science & Engineering

Supervisor:

Ms. Apurva Jain

Assistant Professor



Submitted by:

Akshat Jain

(00115607223)

Department of Computer Science & Engineering
Dr. Akhilesh Das Gupta Institute of Professional Studies
(Formerly ADGITM)

FC-26, SHASTRI PARK, NEW DELHI

Affiliated to



GURU GOBIND SINGH INDRAPRASTHA UNIVERSITY
Sector - 16C Dwarka, Delhi - 110075, India

2022-26

Table of Contents

Particulars	Page No
Automated and Scalable Platform for Subscription Management with Alerts	
Declaration	
ABSTRACT	
1. Introduction including Problem Statement	01
2. Objectives and Scope of the Project	02
3. Tools / Platform/Methodology	04
4. Hardware and Software Requirement specifications	07
5. Testing Technologies used	08
6. Significance of the Project	09
References	
Appendixes	

Declaration

We, the undersigned, hereby declare that the proposed synopsis for the Major Project titled:

“Automated and Scalable Platform for Subscription Management with Alerts” submitted in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology (B.Tech.) in Computer Science and Engineering**, is **based on our original work**.

We further declare that this work has **not been submitted, either in part or full**, to this or any other university/institute for the award of any degree or diploma.

All sources of information and content from other works have been **duly acknowledged** and referenced wherever applicable.

Student Details

Sl.	Name	Enrollment No.	Mobile No	Signature
1)	Akshat Jain	00115607223	9810684778	

Date: _____

Place: _____

ABSTRACT

In today's rapidly evolving digital environment, subscription-based services have become an integral part of everyday life. Users frequently subscribe to multiple platforms such as streaming services, cloud applications, utilities, and digital memberships. Managing these subscriptions manually often leads to missed renewal dates, unplanned expenses, and lack of visibility into recurring financial commitments. Existing expense tracking solutions focus primarily on one-time transactions and fail to provide intelligent mechanisms for managing recurring subscriptions.

This project, **Automated and Scalable Platform for Subscription Management with Alerts**, proposes a centralized system that enables users to track, manage, and analyze their subscriptions efficiently. The platform is designed using a microservices-based architecture to ensure scalability, reliability, and maintainability. It allows users to add subscriptions, monitor renewal cycles, calculate monthly recurring costs, and receive automated alerts before renewals.

The system leverages modern backend technologies, asynchronous communication, and containerized deployment to simulate real-world enterprise applications. A mobile-based user interface provides ease of access, analytics visualization, and notification handling. The expected outcome of the project is to enhance financial awareness, reduce unnecessary spending, and demonstrate the practical application of distributed system design and automation in solving real-world problems.

Introduction including Problem Statement

The rapid advancement of digital technologies and the widespread adoption of internet-based services have led to a significant increase in subscription-based business models. Today, individuals and organizations rely heavily on recurring services such as online streaming platforms, cloud storage solutions, software-as-a-service (SaaS) tools, digital learning platforms, utility services, and membership-based applications. These services operate on predefined billing cycles such as monthly, quarterly, or yearly subscriptions, offering convenience and continuous access to digital resources.

While subscription models simplify access to services, they also introduce several challenges for users. As the number of active subscriptions increases, users often find it difficult to keep track of renewal dates, billing amounts, and payment frequencies. Subscriptions are frequently renewed automatically without explicit user acknowledgment, leading to situations where users continue paying for services they no longer actively use. This lack of visibility can result in unplanned expenses, budget mismanagement, and reduced financial awareness.

In most cases, users depend on manual methods such as calendar reminders, email notifications, or bank statements to monitor their subscriptions. These approaches are fragmented and unreliable, as emails can be overlooked, reminders may not be updated, and bank statements do not clearly distinguish recurring payments from one-time transactions. Consequently, users may miss important renewal deadlines, experience service interruptions due to failed payments, or incur financial losses from forgotten or unused subscriptions.

Existing expense management and budgeting applications primarily focus on tracking individual transactions and categorizing expenses. Although they provide an overview of spending patterns, they lack dedicated mechanisms to intelligently identify, manage, and analyze recurring subscription payments. These systems do not offer automated reminders, subscription lifecycle management, or actionable insights related to recurring financial commitments. Additionally, many such solutions are not designed with scalability and modularity in mind, limiting their ability to support growing user bases and evolving requirements.

The problem addressed by this project is the absence of an intelligent, automated, and scalable platform that specifically targets subscription and recurring bill management. There is a clear need for a centralized system that can consolidate subscription data, track renewal cycles, generate timely alerts, and provide meaningful insights into recurring expenses. Furthermore, such a system must be designed using modern architectural principles to ensure scalability, reliability, and ease of maintenance as the number of users and subscriptions increases. This project aims to address these challenges by proposing a structured and efficient solution for subscription management with automated alerts.

Objectives and Scope of the Project

2.1 Objectives of the Project

The primary objective of this project is to design and develop an automated and scalable platform for subscription management with alerts, capable of addressing the challenges associated with managing multiple recurring subscriptions in a digital environment. The system aims to provide users with better control, visibility, and awareness of their subscription-based expenses while demonstrating the practical application of modern software engineering principles.

One of the key objectives of the project is to design and implement a centralized subscription management platform that allows users to store and manage all their subscription-related information in a single system. Instead of relying on scattered sources such as emails, bank statements, or manual reminders, the platform consolidates subscription details such as service name, billing amount, renewal date, and billing frequency. This centralization improves usability and reduces the likelihood of missed renewals or forgotten subscriptions.

Another major objective is to automate reminders and alerts for upcoming subscription renewals. Timely notifications play a crucial role in preventing unwanted charges and service disruptions. The system is designed to generate alerts before renewal dates, enabling users to take informed decisions such as continuing, pausing, or cancelling subscriptions. Automation of alerts minimizes manual effort and ensures consistent notification delivery.

The project also aims to provide meaningful insights into monthly and recurring subscription expenses. By analyzing subscription data, the platform calculates total monthly spending, recurring costs, and category-wise expense distribution. These insights help users understand their spending patterns and identify areas where cost optimization is possible. Such analytical features enhance financial awareness and promote responsible spending behavior.

A significant technical objective of the project is to design the system using a scalable and modular microservices architecture. Each functional component of the system is structured as an independent service with a clearly defined responsibility. This approach improves maintainability, fault isolation, and scalability, making the system suitable for real-world deployment scenarios. The modular design also allows individual services to be enhanced or replaced without affecting the overall system.

Another important objective is to demonstrate the real-world application of event-driven communication and containerization technologies. The project utilizes asynchronous messaging mechanisms to enable loose coupling between services, ensuring reliable communication and improved performance. Containerization is employed to simplify deployment, ensure consistency across environments, and support horizontal scaling. Through this objective, the project showcases industry-relevant practices used in modern distributed systems.

Overall, the objectives of the project extend beyond functional implementation and focus on applying theoretical concepts of software architecture, distributed systems, and automation to solve a practical, real-world problem effectively.

2.2 Scope of the Project

The scope of this project defines the boundaries and limitations of the system while outlining the features and functionalities included in the current implementation. The primary scope of the project includes subscription tracking, alert generation, and analytics visualization. Users are able to add, update, view, and manage their subscription details within the platform. The system maintains structured records of subscription data and tracks recurring billing cycles efficiently.

Alert generation is a core part of the project scope. The system is designed to generate automated notifications for upcoming renewals, helping users avoid missed payments or unintended renewals. Alerts are generated based on predefined rules and timelines, ensuring consistency and reliability. While the current implementation focuses on basic alert mechanisms, the design allows for future expansion to multiple notification channels.

Analytics and visualization form another important aspect of the project scope. The system processes stored subscription data to generate insights such as total monthly recurring expenses and subscription trends. These analytics are presented in a user-friendly manner through dashboards and summary views, enhancing the overall usability of the platform.

The system is intentionally designed to be extensible, allowing future enhancements without major architectural changes. Potential extensions include AI-based extraction of subscription details from natural language input, optical character recognition (OCR) for bill scanning, support for shared or family subscriptions, and integration with external payment or notification services. The modular nature of the architecture ensures that such enhancements can be incorporated with minimal disruption.

Although the project focuses on core subscription management functionality, it also considers scalability and large-scale deployment as part of its scope. The architecture supports increasing numbers of users and subscriptions by distributing workloads across independent services. This makes the system suitable for both academic demonstration and real-world application.

Certain features are deliberately kept outside the scope of the current implementation to maintain focus and feasibility. These include direct bank integrations, automatic payment execution, and real-time financial transaction processing. Such features may be considered for future versions of the system.

In summary, the scope of this project encompasses the design and implementation of a reliable, scalable, and automated subscription management platform while providing sufficient flexibility for future growth and enhancement. The defined scope ensures a balance between functional completeness, technical depth, and practical feasibility.

Tools / Platform / Methodology

The development of the proposed system follows a structured, modular, and layered approach, making use of modern software engineering methodologies and tools. The selection of architecture, platforms, and technologies has been carefully made to ensure scalability, maintainability, reliability, and ease of future enhancement. The methodology adopted in this project reflects current industry practices for building distributed and enterprise-grade applications.

3.1 Methodology

Microservices Architecture

The system is designed using a microservices architecture, where the overall application is decomposed into smaller, independent services, each responsible for a specific business function. Examples include user management, subscription management, notification handling, and analytics. Each microservice operates independently, has its own data storage, and communicates with other services through well-defined interfaces. This approach improves system scalability, fault isolation, and maintainability. In case of failure in one service, the rest of the system continues to function without disruption.

Event-Driven Communication

To enable loose coupling between services, the project adopts an event-driven communication model. Instead of synchronous service-to-service calls, events are published and consumed asynchronously. This design improves system responsiveness and scalability, especially in scenarios involving alerts and notifications. Event-driven communication also allows services to evolve independently without requiring changes in other components, making the system more flexible and resilient.

RESTful API Design

The system exposes its functionalities through RESTful APIs, which follow standard HTTP methods such as GET, POST, PUT, and DELETE. RESTful design principles ensure consistency, simplicity, and interoperability between services and client applications. APIs are stateless, making them easy to scale horizontally and suitable for distributed environments. Clear separation between client and server logic further enhances maintainability.

Modular and Loosely Coupled Services

The project emphasizes modularity and loose coupling, where each component is designed with a single responsibility and minimal dependency on other components. This approach allows developers to modify or extend individual services without impacting the entire system. It also simplifies testing, debugging, and deployment. The modular design plays a crucial role in supporting future enhancements and system evolution.

3.2 Tools and Platforms

Backend Framework: Spring Boot

Spring Boot is used as the primary backend framework due to its robustness, ease of configuration, and strong support for building microservices. It simplifies application development by providing auto-configuration, embedded servers, and seamless integration with databases and messaging systems. Spring Boot also supports secure, scalable, and production-ready applications.

Frontend Framework: React Native

React Native is used to develop the mobile application interface. It allows the creation of cross-platform applications using a single codebase, ensuring consistency across different devices. React Native provides a responsive and interactive user interface, making it suitable for displaying subscription data, alerts, and analytics.

Database: MySQL

MySQL is chosen as the relational database management system for storing structured data such as user profiles, subscription details, and alert information. It provides reliability, data integrity, and efficient query performance. The use of a relational database ensures structured data management and supports transactional operations.

Messaging System: Apache Kafka

Apache Kafka is used as the messaging platform to support asynchronous, event-driven communication between services. Kafka enables reliable message delivery, scalability, and fault tolerance. It is particularly suitable for handling notification triggers, alert generation, and inter-service communication without tight coupling.

API Gateway: Kong

Kong is used as an API Gateway to manage incoming client requests. It acts as a single entry point for all services, handling request routing, authentication, and rate limiting. The API Gateway improves security, simplifies client interactions, and enhances overall system manageability.

Containerization: Docker

Docker is used to containerize individual services, ensuring consistency across development, testing, and deployment environments. Containerization simplifies deployment, enables horizontal scaling, and improves resource utilization. It also allows services to be deployed independently without environmental conflicts.

Version Control: Git

Git is used for version control to manage source code changes efficiently. It supports collaborative development, change tracking, and rollback functionality. Version control ensures code integrity and simplifies project maintenance.

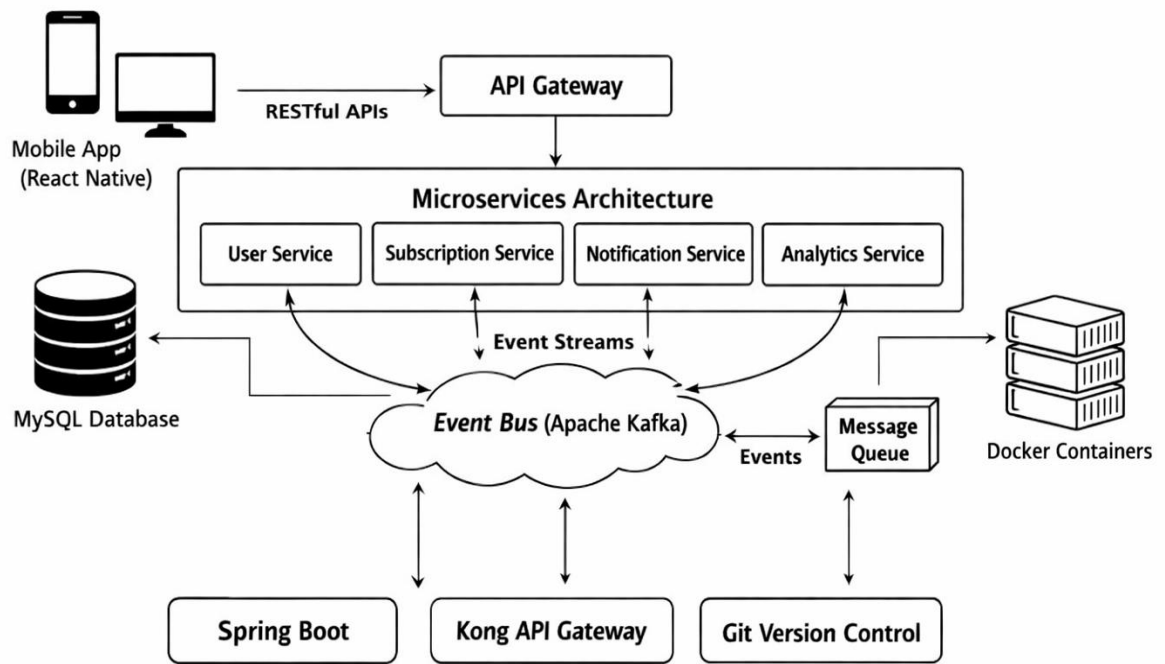


Fig 3.1 System Architecture

Hardware and Software Requirement Specifications

Hardware Requirements

Processor: Intel i5 or equivalent

RAM: Minimum 8 GB

Storage: 256 GB or higher

Internet Connectivity

Software Requirements

Operating System: Windows / Linux / macOS

Java Development Kit (JDK 17 or above)

Node.js and npm

Docker Engine

MySQL Server

IDE: IntelliJ IDEA / VS Code

Testing Technologies Used

Testing is a critical phase in the software development lifecycle and plays an essential role in ensuring the reliability, correctness, and stability of the system. For this project, multiple testing techniques are applied at different levels to validate individual components as well as the overall system behavior. A combination of unit testing, integration testing, API testing, and manual user interface testing is used to ensure that the platform functions as expected under various conditions.

Unit Testing is performed to verify the correctness of individual components and business logic within the system. Each service is tested independently to ensure that methods and functions produce the expected outputs for given inputs. Unit tests help in identifying defects at an early stage of development and make the codebase more maintainable. By isolating components during testing, issues can be detected and resolved before they propagate to higher levels of the system.

Integration Testing is used to validate the interaction between different modules and services. Since the system follows a modular and distributed architecture, it is important to ensure that data flows correctly between components. Integration testing verifies that services communicate properly and that the overall workflow functions as intended. This form of testing helps identify issues related to data consistency, service dependencies, and communication mechanisms.

API Testing is conducted to validate the correctness, reliability, and performance of RESTful APIs exposed by the system. Tools such as Postman are used to test API endpoints by sending requests and verifying responses. API testing ensures that endpoints handle valid and invalid inputs correctly, return appropriate status codes, and enforce expected data formats. This approach also helps confirm that the system adheres to RESTful design principles.

Manual User Interface Testing is performed on the mobile application to ensure a smooth and user-friendly experience. This includes validating screen navigation, form inputs, data display, and error handling. Manual testing helps identify usability issues that automated tests may not capture, such as layout inconsistencies, responsiveness, and overall user interaction flow.

Together, these testing practices help identify defects early, improve system reliability, and ensure stable and predictable system behavior. A structured testing approach contributes significantly to the overall quality and robustness of the project.

Significance of the Project

The significance of this project lies in its practical application of modern software engineering concepts to solve a real-world problem related to subscription and recurring bill management. With the increasing adoption of subscription-based services, users face growing challenges in tracking renewals, managing recurring expenses, and maintaining financial awareness. This project addresses these challenges by providing an automated and structured solution that simplifies subscription management and reduces unnecessary financial inefficiencies.

From a technical perspective, the project demonstrates the effective use of microservices architecture, which is widely adopted in modern enterprise systems. By decomposing the application into independent and modular services, the system achieves improved scalability, fault isolation, and maintainability. The use of containerization technologies further enhances deployment consistency and portability across different environments. Additionally, the implementation of event-driven design showcases how asynchronous communication can improve system responsiveness and decouple service dependencies.

Academically, this project contributes significantly to the understanding of distributed systems and backend application design. It provides hands-on experience with concepts such as service communication, modular architecture, and system scalability, which are essential in contemporary software development. The project bridges the gap between theoretical knowledge and practical implementation, making it a valuable learning experience for students.

From a practical standpoint, the system is designed with extensibility in mind, allowing it to be evolved into a production-ready application with minimal architectural changes. Features such as automated alerts, analytics, and centralized subscription tracking can be further enhanced and integrated with real-world services. The project also highlights the importance of automation and intelligent notification mechanisms in personal finance management, encouraging better spending habits and informed decision-making.

Overall, the project holds academic, technical, and practical significance by demonstrating how modern technologies and architectural principles can be effectively applied to address everyday problems in a scalable and efficient manner.

Reference

1. Airbnb Engineering Blog – Articles on search ranking, trust, and verification: <https://medium.com/airbnb-engineering>
2. Magic Bricks - <https://www.magicbricks.com/>
3. 99 Acers - <https://www.99acres.com/>
4. Trulia Engineering – Recommendations in rental/property platforms: <https://www.trulia.com/research/>
5. ISO/IEC 25010:2011 – *System and Software Quality Models* (for system verification & usability).
6. Scalable and Secure Real-Time Chat Application Development Using MERN Stack and Socket.io for Enhanced Performance - <https://www.macawpublications.com/Journals/index.php/FCR/article/view/51>
7. Socket.io Documentation – Real-time chat implementation: <https://socket.io/docs>
8. PostgreSQL Documentation – For handling property listings & relational data: <https://www.postgresql.org/docs>