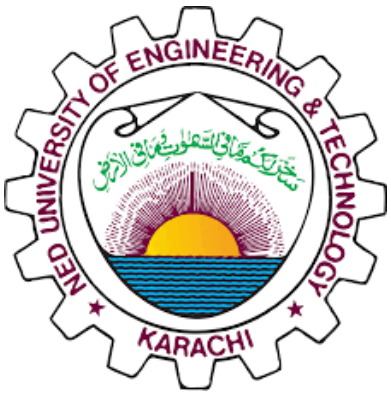
UNDERGRADUATE FINAL YEAR DESIGN PROJECT

Department of Software Engineering

NED University of Engineering and Technology



SyncStream

Group Number: SE-21017 **Batch: 2021-2025**

Group Member Names:

Muhammad Anas	SE-21020
Bakhtiar Ahmed	SE-21029
Ammar Ahmed Khan	SE-21033
Fawad Tariq	SE-21050

Approved by

. . . . . .

Engr. Sana Fatima

Project Advisor

Author's Declaration

We declare that we are the sole authors of this project. It is the actual copy of the project that was accepted by our advisor(s) including any necessary revisions. We also grant NED University of Engineering and Technology permission to reproduce and distribute electronic or paper copies of this project.

Muhammad Anas	Bakhtiar Ahmed	Ammar Ahmed Khan	Fawad Tariq
SE-21020	SE-21029	SE-21033	SE-21050
anas4407015@	ahmed4406659@	khan4405225@	tariq4402339@
cloud.neduet.edu.pk	cloud.neduet.edu.pk	cloud.neduet.edu.pk	cloud.neduet.edu.pk

Table Of Contents

List of Figures

List of Abbreviations

API	Application Programming Interface
CRM	Customer Relationship Management
EAI	Enterprise Application Integration
ERP	Enterprise Resource Planning
GDPR	General Data Protection Regulation
iPaaS	Integration Platform as a Service
P2P	Peer-to-Peer
SDGs	Sustainable Development Goal
SDLC	Software Development Life Cycle
SOAP	Simple Object Access Protocol
SOA	Service-Oriented Architecture
UAP	Unified Application Platform
WOIA	Web-Oriented Integration Architecture

United Nations Sustainable Development Goals

The Sustainable Development Goals (SDGs) are the blueprint to achieve a better and more sustainable future for all. They address the global challenges we face, including poverty, inequality, climate change, environmental degradation, peace, and justice. There are a total of 17 SDGs as mentioned below. Check the appropriate SDGs related to the project.

☐ No Poverty

☐ Zero Hunger

☐ Good Health and Well-being

☐ Quality Education

☐ Gender Equality

☐ Clean Water and Sanitation

☐ Affordable and Clean Energy

☑ Decent Work and Economic Growth

☑ Industry, Innovation, and Infrastructure

☐ Reduced Inequalities

☐ Sustainable Cities and Communities

☑ Responsible Consumption and Production

☐ Climate Action

☐ Life Below Water

☐ Life on Land

☐ Peace, Justice, and Strong Institutions

☑ Partnerships to Achieve the Goals

Similarity Index Report

Following students have compiled the final year report on the topic given below for partial fulfillment of the requirement for Bachelor's degree in Software Engineering.

Project Title SyncStream

S. No.	Student Name	Seat Number
1.	Muhammad Anas	SE-21020
2.	Bakhtiar Ahmed	SE-21029
3.	Ammar Ahmed Khan	SE-21033
4.	Fawad Tariq	SE-21050

This is to certify that Plagiarism test was conducted on the complete report, and overall similarity index was found to be less than 20%, with maximum 5% from a single source, as required.

Signature and Date

Engr. Sana Fatima

Project Advisor

Executive Summary

Most growing companies face a common challenge: their core systems like CRMs, ERPs, E-commerce platforms, and marketing tools are not built to communicate seamlessly with one another. As businesses scale and adopt more tools, they often encounter manual data entry, disconnected records, and fragmented workflows. These inefficiencies lead to wasted time, increased errors, and missed opportunities.

SyncStream solves this by providing a centralized platform that automates real-time data synchronization across multiple business applications. Whether it's syncing customer data between Shopify and HubSpot or building custom integrations for internal systems, SyncStream makes it simple. With no-code setup for ease and the flexibility for advanced customizations, the platform is designed for both technical and non-technical users. Its modular architecture ensures smooth integration without the burden of repetitive, manual processes.

SyncStream supports both real-time and scheduled data syncs, giving teams full control over how and when data flows. The platform allows dynamic registration and configuration of custom platforms, making it possible to integrate virtually any system with ease. Features like configurable API flows, field mappings, activity logs, and automated jobs make SyncStream especially powerful for growing startups and enterprise teams. Our ideal users include small to large-scale enterprises, e-commerce businesses, and any organization juggling multiple tools in need of scalable, secure, and low-maintenance integration.

Chapter 1

INTRODUCTION

1.1 Background Information

In the modern era of digitization, companies depend on several software systems to execute a number of operations, ranging from marketing and sales to inventory, finance, customer support, and others. These systems can be Shopify, WooCommerce, NetSuite, HubSpot, etc. While these systems can be great independently, in most cases, they are not designed to communicate with one another out of the box. Though these sites are in different domains, at times they deal with common data entities like customers, orders, products, transactions, etc. In the absence of an integration method, this common data is replicated across systems, creating inefficiency and data inconsistencies. This disconnection results in inefficiencies like manual data handoffs, delayed updates, out-of-sync records, and human mistakes. For instance, in an e-commerce setup, disconnected systems can result in stock inconsistencies and order delays. In a marketing context, poor sync between CRMs and email tools can result in irrelevant targeting or lost prospects. As companies grow and data piles up, it is no longer feasible to manually process these workflows. Decentralized systems result in bottlenecks in processes, increase expense, and harm customer experience. Without automation, teams have to dedicate time on duplicate data copying between platforms rather than on value-creating tasks.

In order to overcome this, An integration platform is required that facilitates real-time syncing of data between various platforms. Such a system enhances data accuracy, business responsiveness, and scalability and forms the basis for automated processes and smart decision-making. management, and improving operational efficiency.

1.2 Significance and Motivation

Our application integration system, **SyncStream**, automates data synchronization across various platforms, making business processing ways smooth and efficient. It connects two platforms so that data can be exchanged between them automatically upon any change. Entering information into records manually is never efficient, taking up considerable resources and time.

SyncStream reduces manual work thus limiting the such human errors involved during data transfer. It also enables organizations to optimize their resources by assigning people (who were earlier doing manual data entry) to something more tactical work. Ultimately, a more smooth and efficient operational process produces a better overall throughput for the business. Moreover, the ease of integration between multiple platforms leads to an-always up-to-date data which helps the teams to keep records accurate and enhance their decision-making abilities.

This integration system ultimately results in increased productivity, decreased data entry costs and real time synchronization. In the ever-evolving landscape of business operations, adapting swiftly and efficiently is crucial. An application integration system is the technology that enables organizations with the solution to survive by streamlining data management processes in such environment. This enables the organizations to focus on strategic initiatives as opposed to consuming time in manual tasks thus con¬tributing to innovation and growth. Our solution truly empowers the companies not only to streamline traditional data entry activities but also help these organizations to scale in the next phase of digital future.

1.3 Aims and Objectives

Our proposed solution will act as a connector or interface between multiple business applications for e-commerce platforms (e.g., Shopify), ERP and CRM platforms (e.g., Salesforce, HubSpot, Microsoft Dynamics, etc) to connect them fro data exchange. The end goal is to perform proper system synchronization, real-time data transfer, and efficient communication between platforms.

Manually inputting data into a system can be time-consuming, prone to human errors and it certainly cannot be counted as data accuracy — this is where the platform will play a role by automating the process.

- Sensitive data will be protected and secured. In addition, will implement secure authentication and authorization to prevent the system from unauthorized access.

- The application will be designed in such a way that it is scalable to serve as an integration platform for other services.

- The application will be user-friendly such that users with less domain knowledge can do the integrations.

1.4 Methodology

We will be using an **incremental** and **agile** methodology to ensure adaptability and flexibility throughout the Software Development Life Cycle (SDLC). This approach will allow for continuous improvement and refinement during the process, ensuring that the project can adjust and evolve based on emerging requirements and feedback. The methodology will be broken down into the following phases:

- **Requirement Analysis and Documentation:** The system's requirements and features will be thoroughly studied, analyzed, and documented to ensure a complete understanding of the project scope. This phase will involve gathering detailed specifications to set the foundation for the development process.

- **Study of Platform Integrations:** The working mechanisms of the platforms, modules, features, and APIs provided by the e-commerce platforms (such as Shopify) and ERP/CRM systems (like Salesforce, HubSpot, Microsoft Dynamics) will be studied in-depth. This step is essential to understand how the systems interact and to ensure seamless integration.

- **Design Phase:** After thorough research and analysis, the system architecture will be planned, designed, and analyzed. This includes creating both high-level and low-level views of the system. The design will be documented with detailed diagrams to visualize the system's components and interactions.

- **Development Phase:** With the proper requirements and designs in place, development will begin. This phase will focus on building a web application that serves as the integration platform. It will involve implementing the frontend interface, developing backend APIs, and integrating middleware to enable communication between the platforms being integrated by the user.

- **Testing Phase:** The system will undergo thorough testing to ensure it performs the intended functions without compromising security and reliability. This phase will include

functional testing, security assessments, and user acceptance testing to ensure the system meets quality standards.

- **Deployment and Continuous Improvement:** Following the testing phase, the system will be deployed, and continuous feedback will be gathered to refine and enhance the platform. This iterative approach ensures that the platform evolves in response to user needs and changing requirements.

This approach is designed to ensure that the developed platform meets all specified requirements with the highest standards of quality, adaptability, and user satisfaction. The flexibility inherent in the agile methodology will allow us to incorporate any necessary changes during the development process, ensuring the final product aligns with both business objectives and user needs.

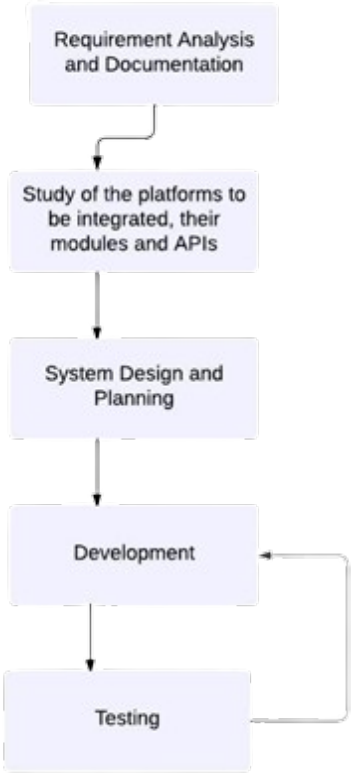


Figure 1.1: Methodology Flowchart

1.5 Gantt Chart

The below Gantt Chart provides a visual representation of our project's timeline, outlining the key phases, tasks, and milestones involved in the development of the system. It defines when each activity is scheduled to begin and end, the duration of each task, and how tasks are interdependent. This allows users and stakeholders to understand the overall project structure, monitor progress, and ensure that the project stays on schedule throughout its lifecycle.

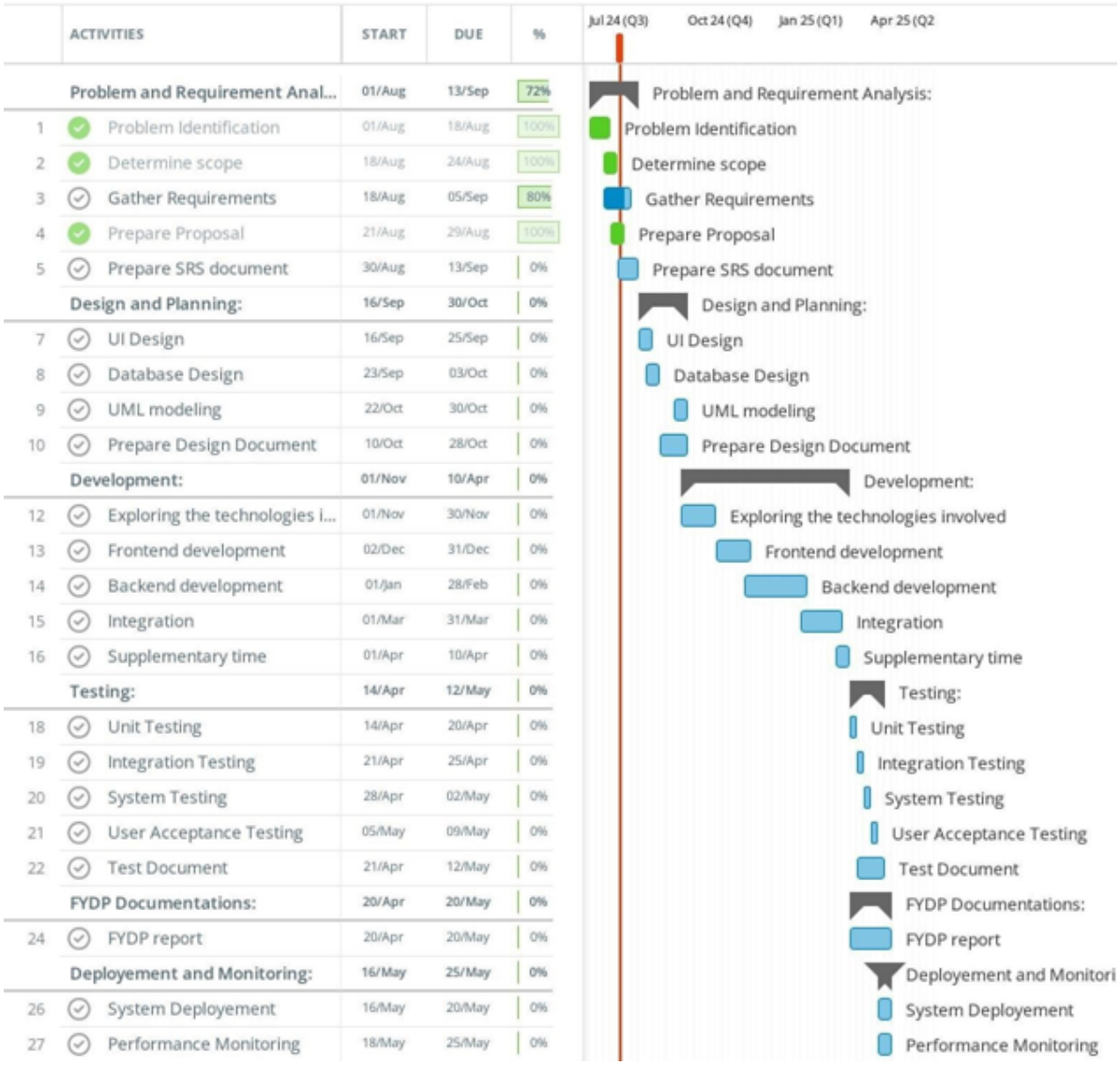


Figure 1.2: Gantt Chart

1.6 Report Outline

Our proposed solution, **SyncStream**, is an automated data integration solution that will automatically transfer data either in real-time as event does occur or at regular intervals for E-commerce platforms like Shopify and ERP/CRM systems like Salesforce, HubSpot and Microsoft Dynamics etc. This solution helps in submitting cases through automation process which reduces manual intervention, increase operational efficiency and elminates errors. Key features of SyncStream include:

- **Automated Data Synchronization:** SyncStream will ensure seamless data synchronization between multiple platforms so that businesses can enjoy real-time or scheduled updates of data without needing a manual effort. Doing so will give every system the latest information, thus reducing human error and improving workflow efficiency. Define and Customize Data Mappings with a Powerful Data Mapping Framework: The platform will provide a robust framework that allows a user to define and customize data mappings to meet their particular data needs. That versatility allows SyncStream to cater to a wide range of business needs.

- **Customizable Data Mapping Framework:** The platform will provide a robust framework that allows users to define and customize data mappings, ensuring that the specific data requirements of their organization are met. This flexibility ensures that SyncStream can accommodate a wide range of business needs.

- **User-Friendly Interface:** SyncStream will contain an easy to use interface that will allow you to set up, configure, and manage your integrations. Users can create and manage these integrations with little technical knowledge and no programming skills required.

- **Data Security and Compliance:** SyncStream will prioritize the security of sensitive data. The platform will implement secure authentication mechanisms and adhere to industry standards and regulations, such as GDPR, to protect user data and ensure compliance with relevant data protection laws.

- **Comprehensive Documentation and Support:** Users will be provided detailed documentation to help them configure and get integration software set up properly. And also

technical support that will help users to tackle their problems during the time of integration.

- **Scalability and Flexibility:** The system will be designed to accommodate future growth, allowing it to scale and expand to work with many more platforms and services. This allows SyncStream to grow with the needs of companies throughout time.
- **Error Handling and Notifications:** SyncStream will configure an enhanced error-handling mechanism that will alert users about the issues during synchronization in order to allow for quick diagnosis/resolution and reduced down-time. Given that integration of data takes time,

By streamlining the integration process, SyncStream will empower businesses to efficiently manage their data, improve decision-making, and enhance operational productivity. Our solution will not only improve existing workflows but also set the foundation for long-term digital transformation.

Chapter 2

LITERATURE REVIEW

2.1 Introduction

Enterprise system integration has evolved from simple point to point connections to highly developed platforms that facilitate easy application connectivity over the decades. For informed platform development, it is essential to understand this evolution and current state of integration technologies.

2.2 Enterprise Application Integration (EAI)

Enterprise Application Integration (EAI) was what began the creation of modern integration in the early 1990's as organizations found themselves faced with a problem of connecting its systems that were gradually getting out of alignment. The notions of EAI, as put forward by Linthicum [1] and Hohpe & Woolf [2] seminal works, defined EAI as unrestricted data and business process sharing between connected applications and data sources within an enterprise. The research focused on why EAI was created, and it stressed that EAI was primarily trying to overcome the difficulty of point to point integrations in large companies that had grown too complicated to manage. These early works emphasized the significance of a layered architecture for EAI solutions which comprises communication services, distribution layers, transformation services and process management components.

The case study of MACom from Irani et al [3] illustrated the real world challenges and potential of application integration. The project, conducted over six months, intended to put together a standardized and flexible infrastructure for the integration of both intra and inter organizational business processes. The complexity of modern enterprise IT environments was further exemplified by MACom's extensive system landscape — more than 2,000 custom systems and over 2,000 critical packaged systems (including SAP R/3). The approach followed by the organization integrated seven intra organizational processes and five inter organizational processes such as customer relationship management, supply chain management and collaborative product commerce.

Then the EAI approach brought several integration methods: screen scraping, interface redesign, and object integration. EAI also got attention especially from Hohpe & Woolf because they emphasized the role of XML as the standard data exchange format in EAI, which facilitates seamless communication of different systems under the challenges of performance and security. Although the research showed XML to be platform independent and good for structured data format, organizations had to consider network performance impact and adjust security measures for sensitive data.

2.3 Service-Oriented Architecture (SOA)

Traditional EAI was quickly becoming limited as organizations grew and diversified, until finally the limitations of systems led to the advent of Service Oriented Architecture (SOA). When Service Oriented Architecture (SOA) originated, it was a game changing enterprise system integration architecture that solved many of the limitations of traditional monolithic architectures and closely integrated approaches. Transition from conventional middleware technologies, i.e., peer-to-peer (P2P) and workflow based middleware to service oriented architecture (SOA) presented a paradigm shift in the way organizations tackled system integration opportunities and problems.

Significant work by Chen et al. [4] on SOA based platforms introduced a Unified Application Platform (UAP), which showed SOA to be suitable for integration of enterprise applications. It pointed out that their research demonstrates a SOA that allows systems to talk to each other via standardized services and provides extraordinary flexibility and reusability. A sophisticated layered architecture was built on the UAP platform through implementing a running platform, application frame, program frame, service frame, and toolbox layers using J2EE technology. SOA principles were shown to be practical in enterprise settings by providing services that were applied to each layer specifically; data access, workflow management, and system security.

Advancing SOA concepts, Jiang et al. [5] researched how web services could be better made available in the enterprise settings. With their innovative ideas, they proposed the web service redundancy and web service map idea to increase the reliability and availability. Their work showed how such reliability requirements of enterprise applications could be met by deploying redundant web services on multiple hosts and using sophisticated mapping mechanisms. This

mathematical analysis has also shown that redundant configurations can significantly improve service availability, compensating for one of the fundamental difficulties of enterprise system integration. Jiang's work on propogation of web service maps represented an important step in the implementation of SOA. This mechanism kept up to date the information about web services and when they had been available to the enterprise. By caching web service maps by clients, message exchanges were reduced, and service availability was ensured although the registry servers were unavailable, the SOA was optimized for real enterprise environments.

Together, these works established that SOA was the better way to perform enterprise integration – it was more flexible, extensivable and less coupled than the erstwhile methods. Altogether, the research showed that with SOA-based platforms, an increased level of workflow flexibility could be supported together with comparatively small integration workloads, which is highly beneficial in the present-day enterprise context in which system adaptability and performance play the key role. Through these implementing and improving iterations, evolution of SOA made important preparatory preconditions for the latter generations of the enterprise integration technologies.

2.4 RESTful Web Services

The evolution sequence proceeded through the shift from the use of simple object access protocol web services to RESTful architecture. In their comparative study Meng et al. [6], pointed out that for Web services, which are based on the SOAP and fitted for a large number of complex environments and structures, the service performance could be notably impaired by the redundant data in the SOAP message and time-consuming serialization procedure. For example, their work showed that RESTful service, and JSON in particular, is vastly superior to traditional service in large numbers of concurrent requests and has lower fluctuation when loaded.

The transition from the initial integration based on the service paradigm of web services defined with Simple Object Access Protocol (SOAP) to the architecture based on the REST model contributed to the development of integration technology. The research conducted by Meng et al. provide a consistent explanation about the fact that, based on the research carried out it was evident that REST became a fresh architectural style on distributed hypermedia systems and

achieved high popularity due to the existing W3C/IETF standard including HTTP, XML, URI, and MIME. Regarding REST research, they primarily focused on its basic concepts, which include resource-oriented architecture where resources are abstracted to concepts equivalent to entities and are exposed by standard universal resource identifiers (URIs).

In their empirical research, they brought quantitative data about REST for proving its advantages compared to more mainstream web services. They identified and compared performance of their REST-based services when exposed to high levels of concurrency; we discovered that JSON response packets in RESTful services are about one-third the size of that of SOAP message packets. Their testing proved that while traditional services could be more efficient with small loads, RESTful services outrank other types of services further and permanently retained a more steady effect at the growing load, particularly in large distributed systems. This performance advantage was attributed to several key factors: SOME of the advantages of REST design include a flexible data format choice, least complexity of the communication process, non-storing of client states and optimization of HTTP features. Of particular importance was the usage of caching and condition GET requests which greatly helped in reducing transfer of duplicate data and improved server performance.

Daniel et al.'s [7] work extended these lines of research by focusing on the evolution of service-oriented architecture to that of Web-oriented architecture (WOA). They have recognized REST's weakness in the enterprise domain and has suggested an approach known as the Web-Oriented Integration Architecture (WOIA). In the face of this, WOIA brought a middleware layer in the mix that retained REST's basic syntax while providing features considered indispensable in a modern day integration project. THIS architectural development made it possible for RESTful services to accommodate complex business logic and data slicing, thereby making the services feasible for broad-scale enterprise integration.

Collectively, these pieces of research confirmed that REST provided substantial benefits with respect to performance, scalability and relative simplicity, however, substantial architectural updates similar to WOIA were needed to expand its use within the enterprise context to accommodate business complexity. The research as a whole indicated that when enhanced and implemented, RESTful services can offer what Meng et al., provided at the same time as the set of enterprise capabilities described in Daniel et al.'s work – thus making the services even more suitable for use in contemporary integration processes.

The evolution from traditional SOAP-based services to REST, and subsequently to enhanced architectures like WOIA, represented a significant advancement in integration technology, offering organizations more efficient and flexible options for system integration while maintaining the robustness required for enterprise applications.

2.5 Integration Platform as a Service (iPaaS)

iPaaS came into being with the introduction of cloud computing, making integration as a distinct advancement in integration modalities. From Hyrynsalmi et al. 's [8] large-scale survey of Finnish software firms several crucial observations on cloud-based integration platforms were made. Their research identified three main types of integration implementations: on-premises, hybrid and cloud-based integration models. Though cloud-based platforms were considered as scalable and less expensive, there were some organizations including small-singer and mid-cap organizations who opted for on-premises due to better control and clear picture of the expenses.

Ebert et al.'s [9] subsequent elaboration of the concept of iPaaS also detailed its main feature and advantages, explaining how suites that run in the cloud process integration with no need for hardware or middleware. This enabled them to focus on the benefits of iPaaS in as per usability and elasticity through the use of features such as the ability to drag and drop and prebuilt adapters. But it also revealed important concerns as regards data protection, privacy, and ability to forecast performance within clouds.

Stephen and Mitchell [10] further explain that iPaaS provides elastic scalability, allowing businesses to scale their integrations across global operations, mergers, and cloud migrations without the need for additional infrastructure. iPaaS supports hybrid and multi-cloud environments, automating workflows and enabling real-time data synchronization. They emphasized iPaaS's overall benefits, including its cost-effectiveness through a subscription-based model and its ability to streamline workflows. Its user-friendly visual tools help businesses maintain operational efficiency while reducing the complexity and cost associated with legacy systems. Thus, iPaaS has become a key enabler for modern, scalable integration solutions that support the agility and efficiency required in today's fast-paced business landscape.

2.6 Automated Data Mapping

The latest developments of integration technology have been directed into the area of automation and intelligent mapping. Birgersson et al.'s [11] innovative work on automating data mapping between different specifications introduced three models: Like Distance, Shortest Distance, Maximum Flow and Data value all depict the value of the data. They showed that these rates could be substantially decreased if the two approaches were used in parallel while keeping the accuracy of data mapping predictions very high. It was found that the Distance model was most successful with the known data set, while the Flow model was best suited at handling new data mappings and the Data Value model as a filter against wrong mappings.

Over time, and with the current developments in integration platforms, studies reflect an emerging importance on smart, automated, efficient and user friendly platforms. The transition from the previous forms of EAI, through the development of SOA and up to current iPaaS and automated mapping tools can be seen as the development process of the field in the face of growing integration complexity and the need for advancements in that area. These advances in integration technology coupled with enhancements from security, scalability, and usability issues provide important information for our project's intent of constructing a lightweight integration system that can meet present day business requirements while at the same time being ready to handle the anticipated problems.

The literature reveals that successful integration platforms must balance multiple factors: ease of use, technical capability, security and extensibility. The key points derived from the evolution of integration technologies will inform the specific features of our development of the lightweight integration platform as we strive to deploy an integration system that avoids the various flaws inherent in the existing approaches.

2.7 Conclusion

The progression from traditional EAI through SOA to modern iPaaS and automated mapping solutions demonstrates the field's response to increasing integration complexity and the demand for more efficient solutions. These advances in integration technology, combined with improvements in security, scalability, and ease of use, provide crucial insights for our project's

goal of developing a lightweight integration platform that can address contemporary business needs while preparing for future challenges.

Chapter 3

SOFTWARE REQUIREMENT SPECIFICATIONS

3.1 Introduction

The System Requirements Specification (SRS) for SyncStream outlines the essential functionalities, performance benchmarks, and technical prerequisites needed to design and implement the platform effectively. SyncStream is envisioned as an integration solution that bridges the gap between e-commerce platforms like Shopify and ERP/CRM systems such as Salesforce, HubSpot, and Microsoft Dynamics. By automating data synchronization and enabling seamless communication between these systems, SyncStream addresses the inefficiencies of manual workflows, data inconsistencies, and operational delays.

This document serves as a comprehensive guide for developers, stakeholders, and project managers to understand the platform's requirements. It details the functional capabilities such as real-time data synchronization, customizable data mapping, and secure user authentication. Additionally, it specifies non-functional requirements like performance, usability, security, and availability, ensuring the platform operates reliably and meets user expectations.

Through this chapter, we aim to provide a clear blueprint for the development process, ensuring the resulting system aligns with business objectives and user needs while adhering to technical and compliance standards. This structured approach ensures SyncStream delivers a robust, scalable, and user-friendly solution for modern integration challenges.

3.2 Functional Requirements

3.2.1 Data Synchronization

- Enable real-time data synchronization between Platforms.

- Allow scheduling of data synchronization at specific intervals.

3.2.2 Data Mapping

- Provide a visual data mapping interface
- Allow users to map source and destination platform data for synchronization.

3.2.3 User Registration And Authentication

- Allow users to register using email/password.
- Implement secure authentication system for users to access the platform.
- Restrict access to features based on user roles.

3.2.4 Workflow Automation

- Enable the creation of automated workflows triggered by specific events, such as order creation, customer sign-up, or data updates.
- Support multi-step workflows.
- Allow users to pause, modify, or cancel workflows in progress.

3.2.5 User Logs

- Allows users to check every activity details through the user logs option
- Allows admin to see logs of all users.

3.3 Non-Functional Requirements

3.3.1 Performance

- Operate with high speed and efficiency to handle real-time and scheduled tasks seamlessly.
- Optimize response times and processing speeds to provide a smooth and reliable user experience.
- Synchronization delays must not exceed 15 seconds for real-time operations.

3.3.2 Usability

- Provide a simple, user-friendly interface accessible from any web browser.

3.3.3 Security

- Ensure data is encrypted during transfer using security protocols.

3.3.4 Availability

- Maintain at least 99% system uptime each month.

3.4 Technical Requirements

3.4.1 Frontend

- Use **React.js** to develop an intuitive and responsive user interface.

3.4.2 Backend

- Use **Python (Django)** for backend development to manage APIs.

3.4.3 Integration

- Leverage APIs from the platforms for seamless integration.

3.4.4 Database

- Use a relational database like **MySQL** to store configuration and logs.

3.5 Business Requirements

3.5.1 Cost Efficiency

- Reduce manual data entry tasks by at least 50%.

3.5.2 Target Audience

- Focus on small to large-sized enterprises needing automated integrations.

3.6 Legal and Compliance Requirements

3.6.1 Data Protection

- Ensure compliance with data regularity and protection rules for handling user data.

3.6.2 API Usage

- Follow the terms of service for APIs used in the integration process.

Chapter 4

SOFTWARE DESIGN AND ARCHITECTURE

4.1 Introduction

The Software Design and Architecture of the SyncStream offers a comprehensive blueprint of the overall structure of the system, including an overview of how the different components will work together to satisfy the functional and non-functional requirements defined in the previous chapter.

This chapter provides the basis for providing the system's requirements into a sound and scalable software solution. They are built as an integration platform to connect across your e-commerce and ERP/CRM systems. Architecture focuses on enabling data transfer between platforms while also delivering a user-friendly interface to end-users and administrators Its design will utilize industry-standard modern technologies, for scalability, maintainability and user-friendliness. In this chapter, we are going to understand the high-level architecture of SyncStream. They are made in accordance with industry best practices and with a view towards flexibility and adaptability to future business requirements.

SyncStream's User Interface has been built for the end-user with considerations for simplicity, an intuitive layout, and a smooth user experience. The UI design is modern with easy-to-follow layouts, ensuring user friendliness for all the users of the system.

In this chapter, we will explore the high-level architecture of SyncStream, including the key components such as the frontend, backend, database structure, and third-party integrations. We will also highlight design decisions that ensure the platform's reliability, security, and performance. These decisions are guided by industry best practices, with a strong emphasis on flexibility and adaptability to future business needs.

4.2 User Interface (UI) Design

The **User Interface** of SyncStream has been designed with the end-user in mind, focusing on simplicity, ease of navigation, and a seamless experience across all platforms. The UI follows

a modern and intuitive layout, ensuring that both new users and experienced administrators can easily navigate through the platform's various functionalities.

4.2.1 Signup Page

The **Signup** offers new users to register to the platform. It gives the user simple instructions and fields to fill in all important information (email, password, etc.). Signing up is reasonably easy, in agreement with the instructions on the screen. The signup process is pretty simple and has simple prompts leading users through each step. It makes the login page follows the successful sign up so users can access their new account. Signup authentication is also integrated and safely verifies all user data.

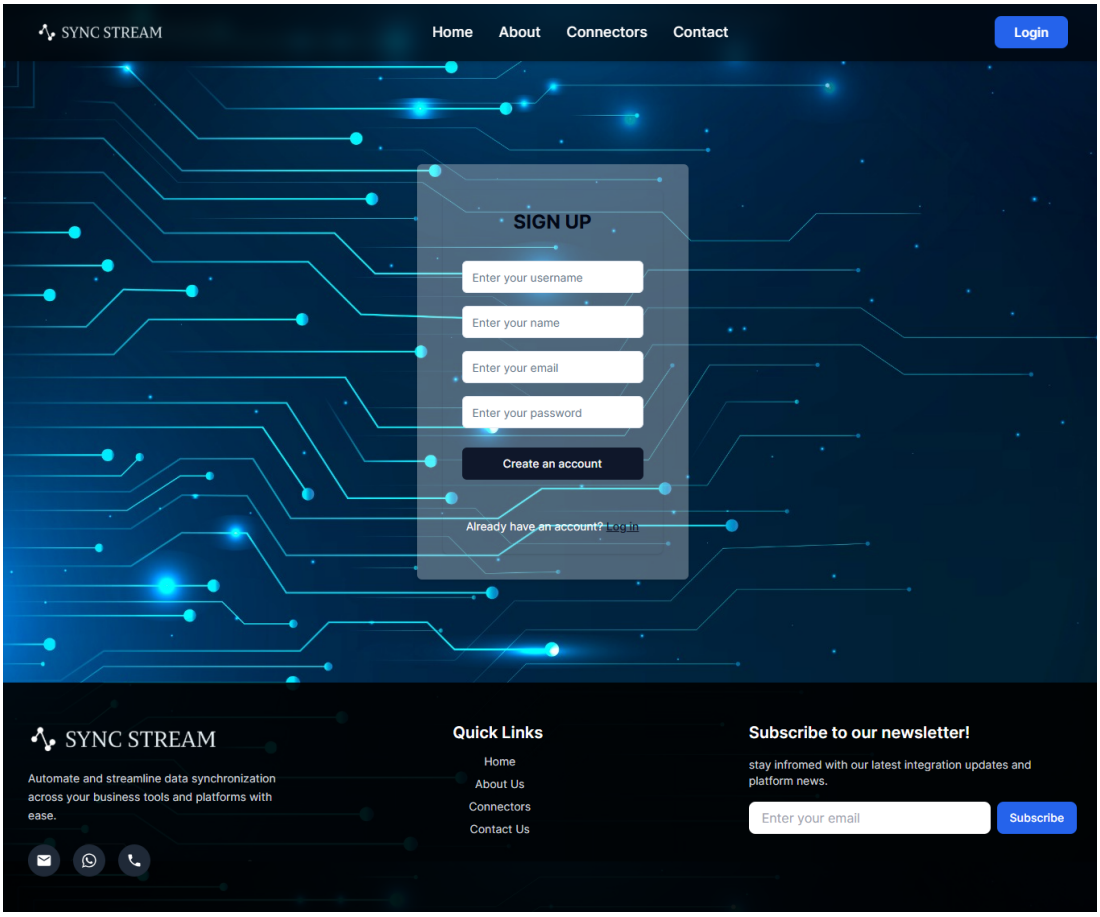


Figure 4.1: SignUp Page UI

4.2.2 Login Page

The **Login** page provides a secure and user-friendly entry point for users to access the platform. The design is all focused on clarity and simplicity. Users can log in using their existing account credentials, and the authentication process is streamlined guiding users through each step. This ensures a smooth and hassle-free login experience for returning users.

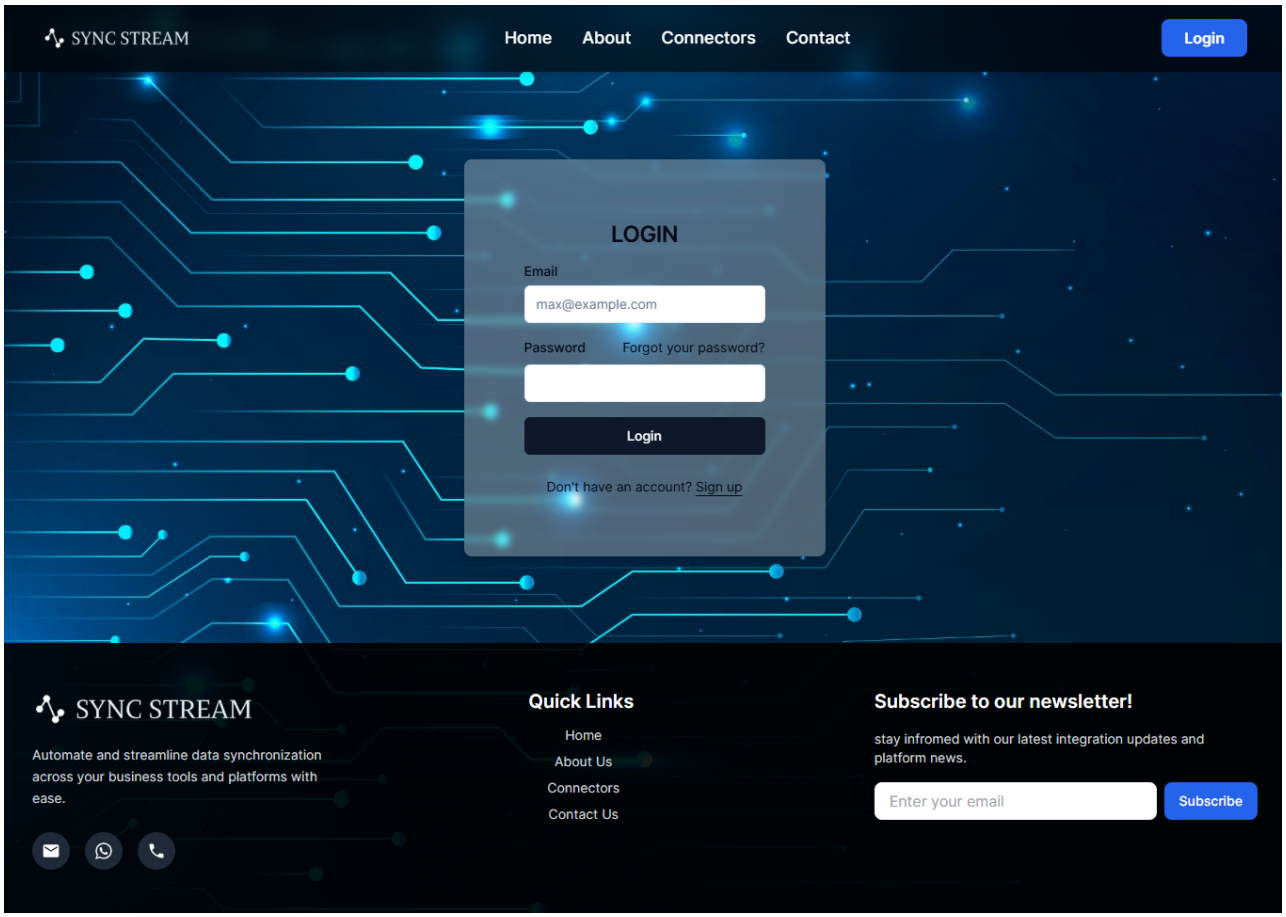


Figure 4.2: Login Page UI

4.2.3 Home Page

The **Home Page** is the first screen that is displayed when the user arrives at the website. Home-page has a navbar at top which simplifies navigation, clear and easy-to-understand. When user is not logged in, access to few pages is avaialble. Once the user registers/logins, the access to more pages is visible through the navbar. With a user-personalized design, all the options are accessible, right on the home page, and easy to find.

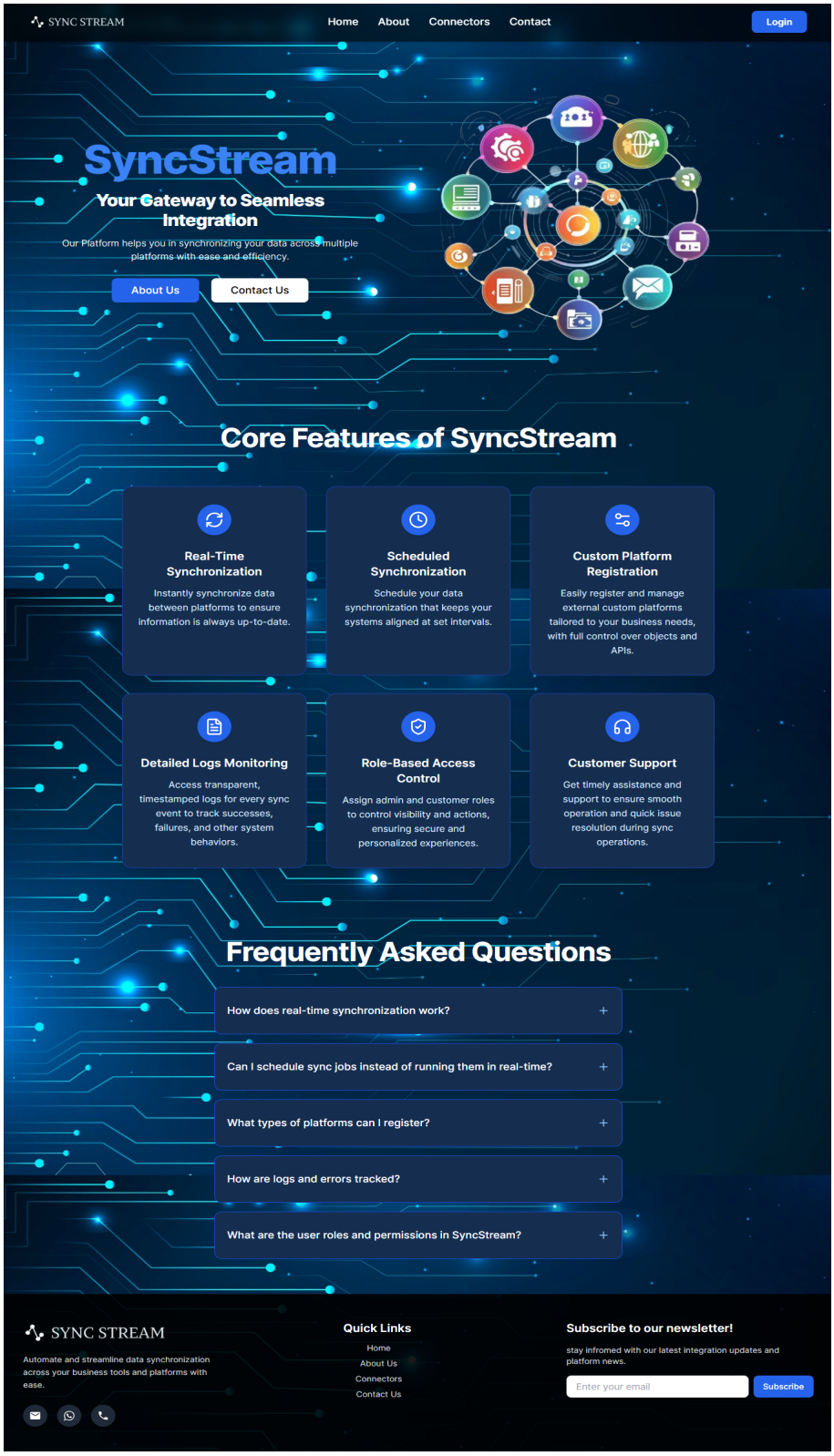


Figure 4.3: Home Page UI Without Login

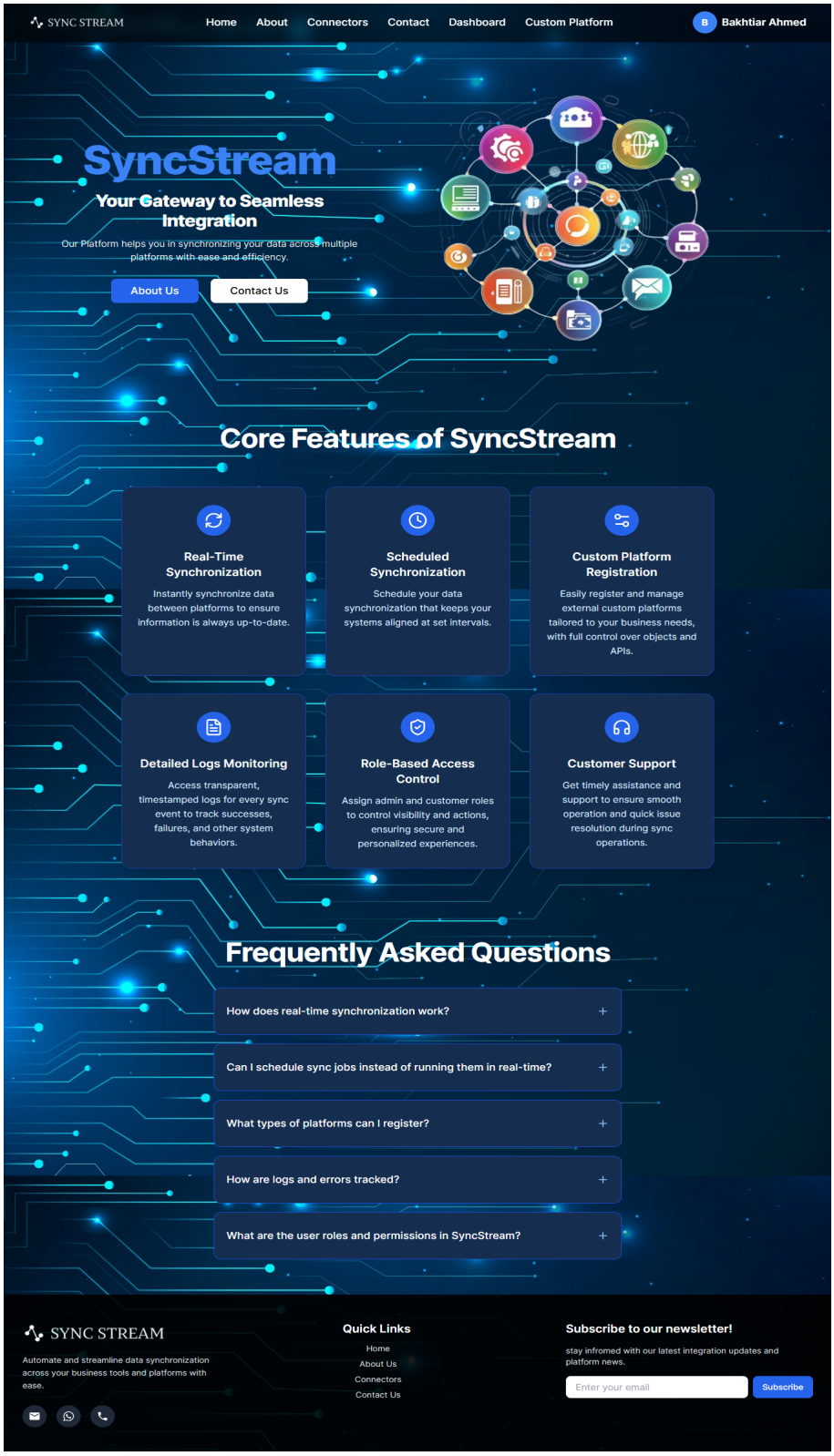


Figure 4.4: Home Page UI With Login

4.2.4 About Us Page

SyncStream's **About Us** page emphasizes the functionality and intention of the platform—it presents SyncStream as a robust integration tool designed to automate real-time data synchronization between different business applications. It states the intention of the platform to make it easier to manage data and enhance working efficiency through smooth interconnectivity between systems such as Shopify, HubSpot, Salesforce, and more. The page also describes SyncStream's vision for a future in which businesses run smoothly with minimal human intervention and lists the values of customer focus, innovation, integrity, and efficiency as those of the platform. Overall, it presents SyncStream as a trustworthy, scalable solution for today's businesses that want smooth integrations.

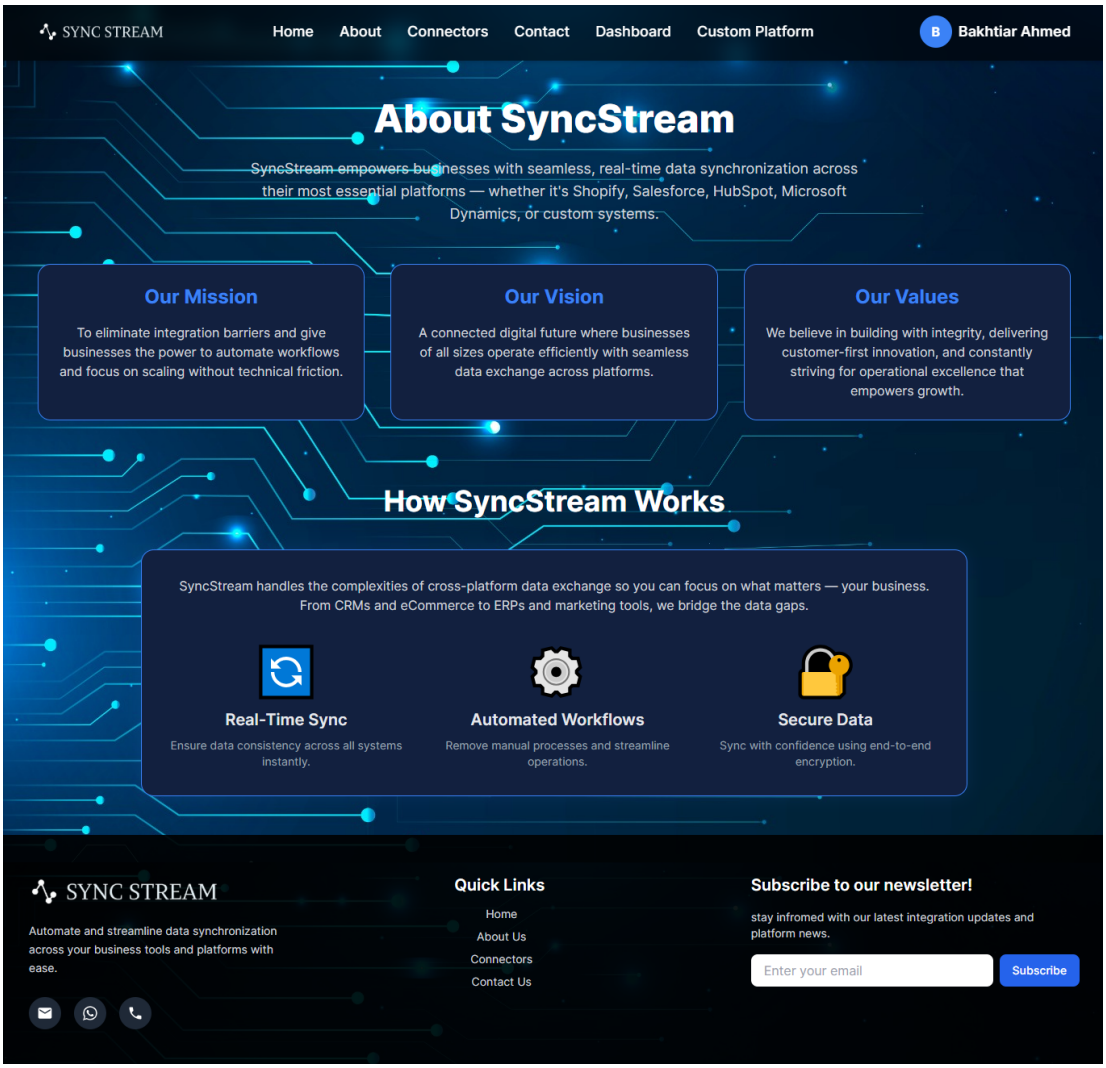


Figure 4.5: About Us Page UI

4.2.5 Contact Us Page

SyncStream's **Contact Us** page is an active point of interaction between customers and the SyncStream support team. It embodies the platform's receptiveness to questions, comments, and support inquiries, thereby making it simple for customers to call for support, partnerships, or questions about the product. The section generally contains the necessary contact information such as email, phone number, and WhatsApp links, alongside a subscription form to keep users informed. It reiterates SyncStream's dedication to timely support, honest communication, and good user and potential collaborator relations.

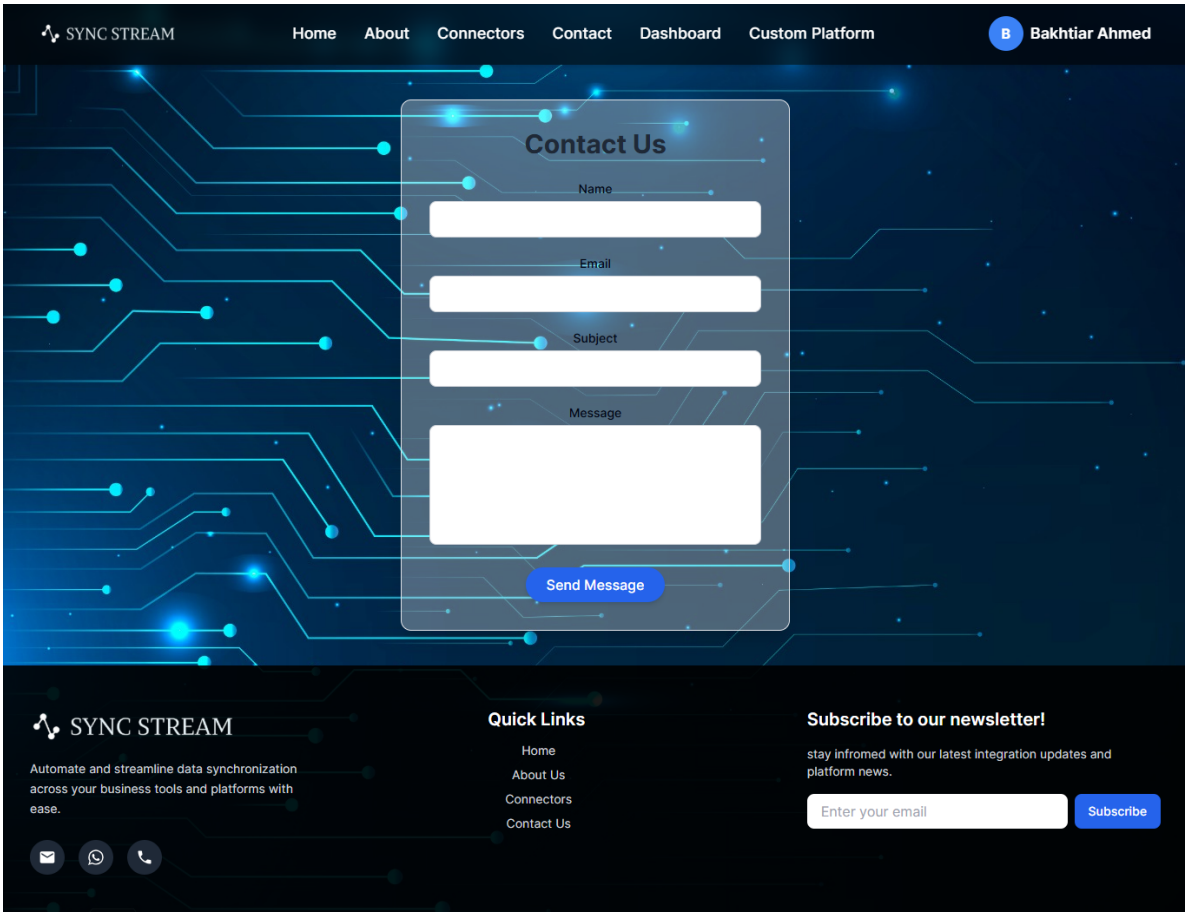


Figure 4.6: Contact Us Page UI

4.2.6 Custom Platform Registration Page

The **Custom Platform Registration** page within SyncStream enables users to register third-party platforms not pre-integrated. It offers a form to provide the platform name, base URL,

and authentication credentials. Once registered, users can specify objects (e.g., customers or orders) and their respective APIs, thus facilitating flexible integration with nearly any platform that can communicate through RESTful.

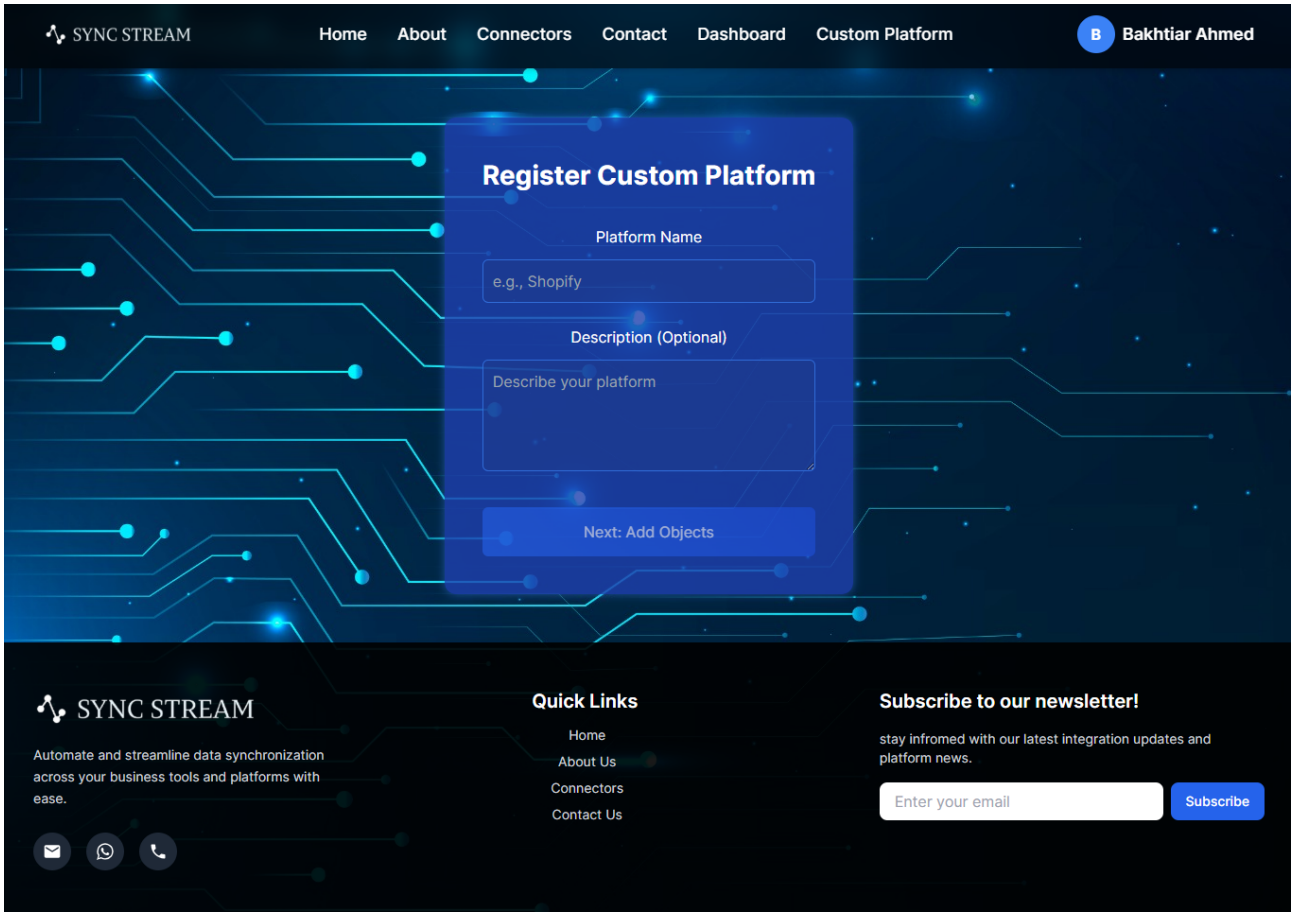


Figure 4.7: Custom Platform Registration Page UI

4.2.7 Dashboard

The **Dashboard** is the central hub of the SyncStream platform, giving users a clear and organized view of their integrations and system performance. Once a connector is purchased, users can access the Dashboard to monitor real-time data synchronization and stay updated on the progress of their workflows. The layout is intuitive and user-friendly, making it easy to find essential features like purchased connectors and active integrations.

For administrators, the Dashboard provides full, unrestricted access at all times. This allows them to oversee user activities, track system performance, and ensure everything is running smoothly. The goal of the Dashboard is to make managing integrations simple and efficient,

giving both users and admins the tools they need to take quick action and maintain control over their processes.

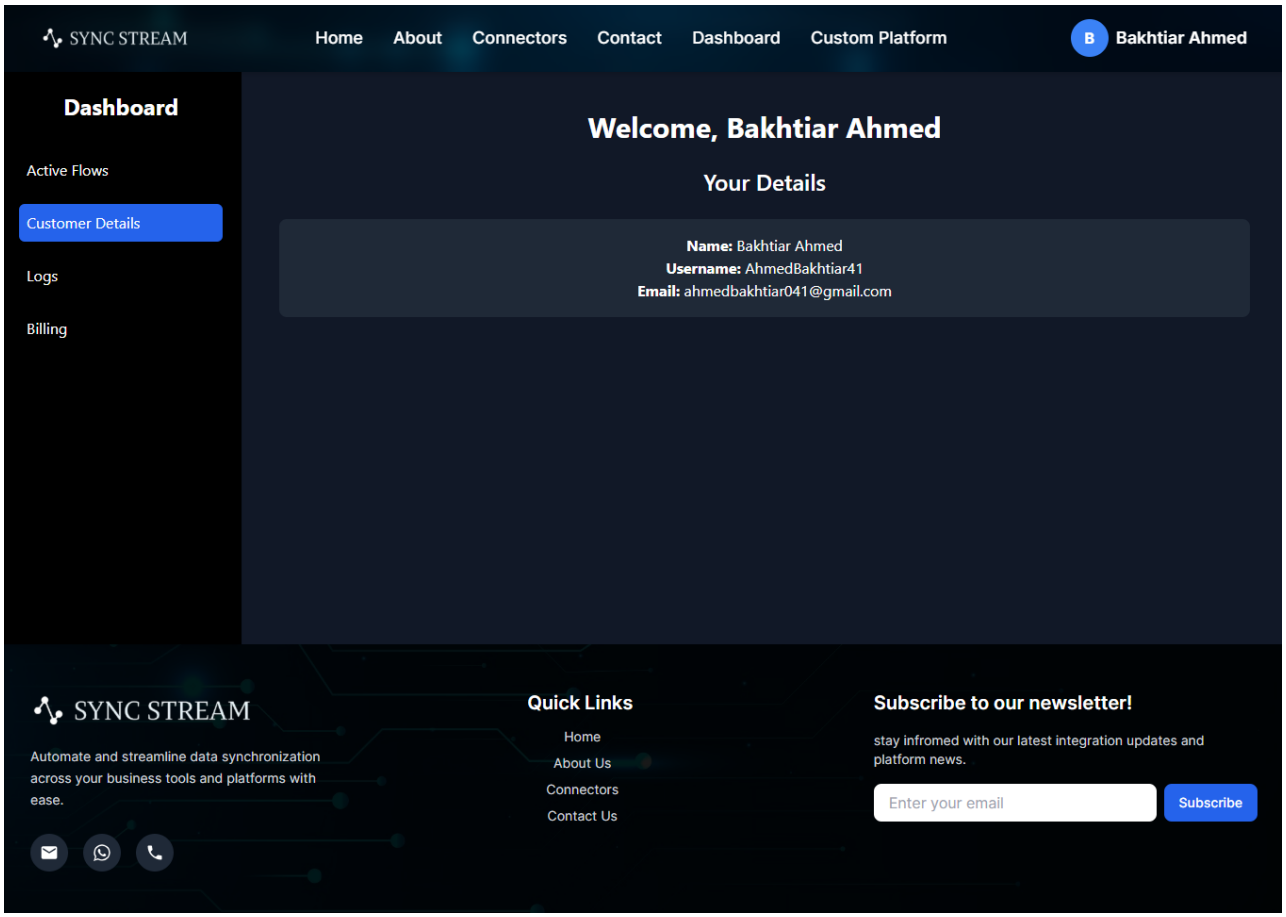


Figure 4.8: Dashboard UI

4.2.8 Subscription Page

The **Subscription page** summarizes plans being available from SyncStream, helping the users make and manage their choice based on business needs. Price, features, and upgrade features are presented concisely and clearly.

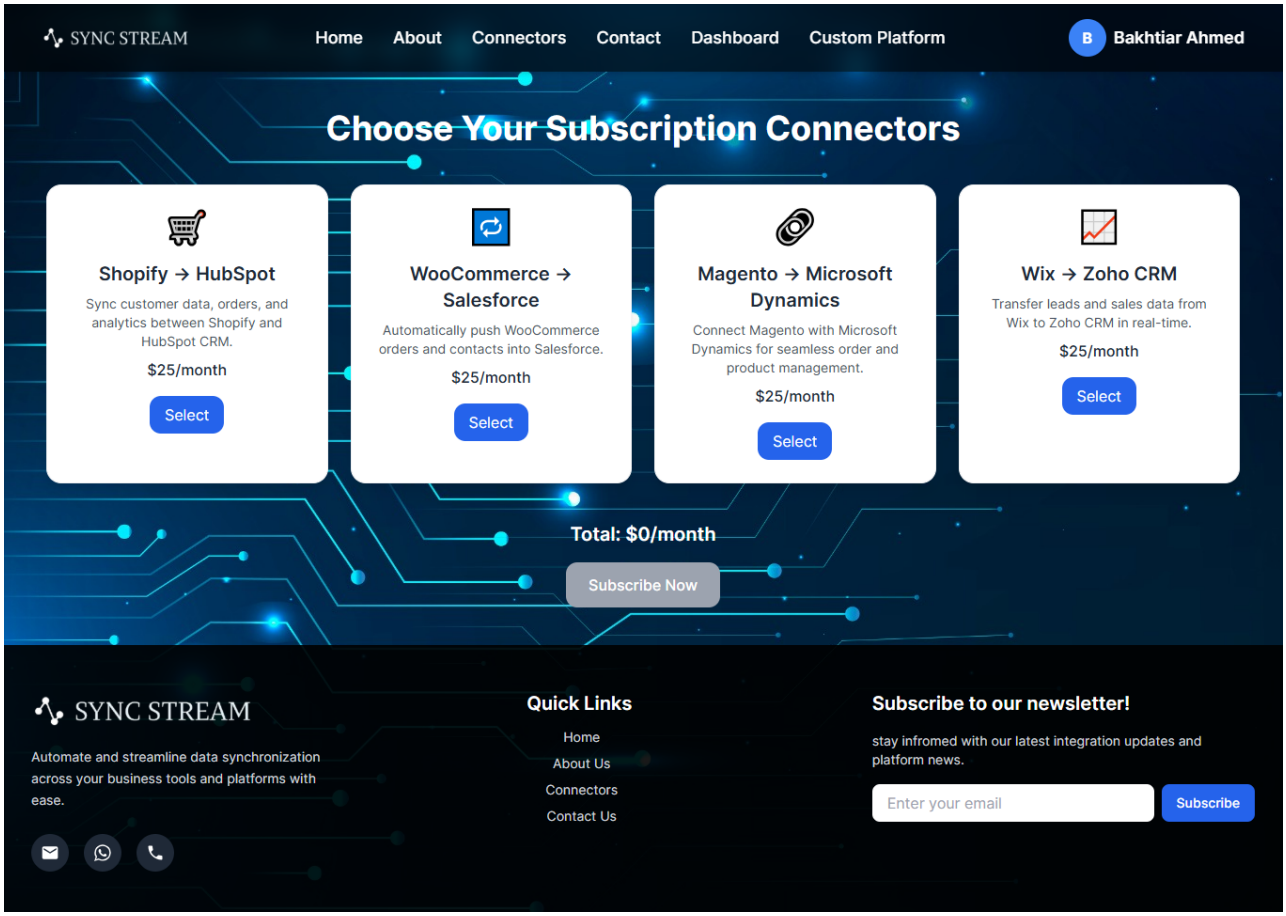


Figure 4.9: Subscription Page UI

4.3 Use Case Diagram

The **Use Case Diagram** provides a visual representation of the key functionalities of the Sync-Stream platform and illustrates the interactions between the actors (users and administrators) and the system. In the diagram, we identify two primary actors: the **User** and the **Admin**. The User interacts with the system to perform tasks such as logging in, purchasing connectors, and using the connectors for data synchronization. Admins, in contrast, manage the system, including user management, connector management, and system configuration. The diagram also showcases the relationships between various use cases, such as the mandatory **Secure Authentication** for both users and admins, and the optional processes like **Customize Connector** and **Error Notifications**. The diagram highlights the core functionalities of the platform, such as **User Signup**, **Login**, **Use Connector**, **Manage Users**, and **Monitor System**, and defines how these actions are linked together.

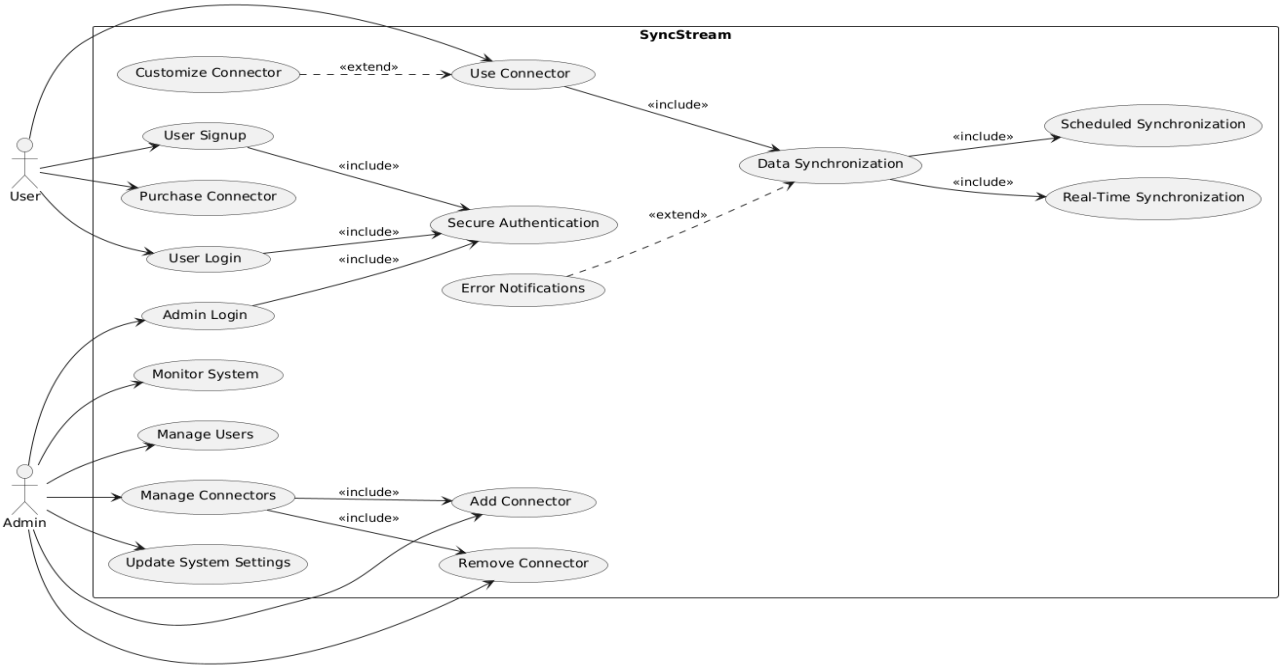


Figure 4.10: Use Case Diagram

4.4 Architecture Diagram

The **Architecture Diagram** provides a high-level view of the SyncStream system's structure and how its core components interact. It illustrates the seamless integration of the **Frontend**, **Backend**, **Database**, and external integrations to deliver a high-performance and scalable platform. The system architecture ensures that the platform can handle large volumes of data and real-time processing efficiently, while also providing flexibility for future growth. The **Frontend** is developed using **React.js**, a modern JavaScript library that allows for a responsive and dynamic user interface. React's component-based structure ensures an intuitive user experience, with real-time updates and easy navigation across the platform. The **Backend** is powered by **Python (Django)**, which manages the business logic, processes API requests, and integrates with external platforms. Django's powerful features, such as built-in authentication and ORM, ensure the system remains secure and efficient. The **Database**, built on **MySQL**, stores crucial data like user credentials, synchronization logs, and configuration settings for connectors. MySQL's scalability and reliability allow the platform to handle complex queries and large datasets effectively. SyncStream's integration with external systems like **Shopify**, **Salesforce**, and **Microsoft Dynamics** is facilitated through robust APIs, enabling seamless data synchro-

nization across different platforms. This modular and maintainable architecture is designed to ensure that SyncStream meets both current and future business needs while delivering a reliable, high-performance solution for system integrations.

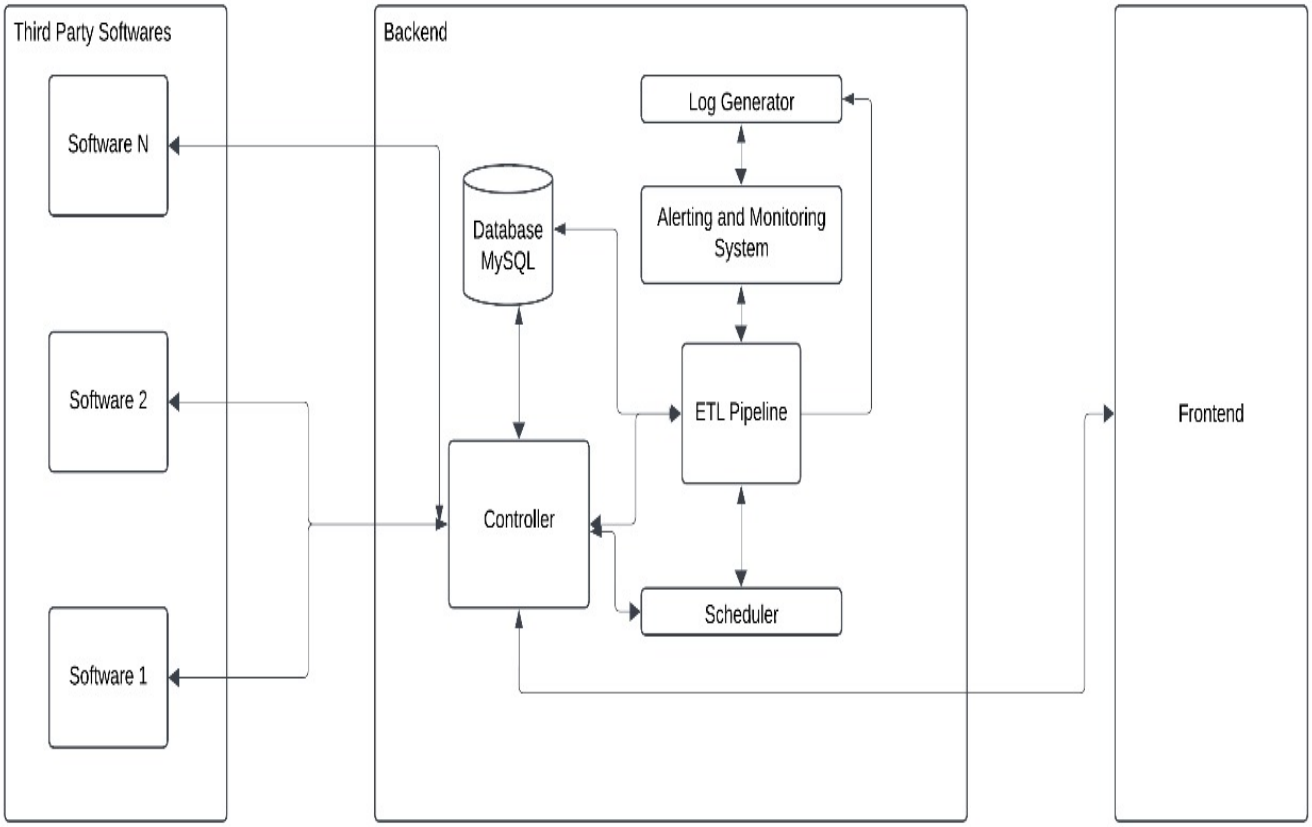


Figure 4.11: Architecture Diagram

4.5 Entity-Relationship Diagram

An **Entity-Relationship (ER) Diagram** visually represents the data structure of **SyncStream**, a system for automating data synchronization across platforms, featuring entities like **Field**, **Mapping**, **Api**, **Flow**, **Connection**, **Webhook**, **SyncLog**, **User**, and **Subscription**. The diagram shows **Field** and **Object** tied to **Platform**, with **Mapping** and **Mapping_details** defining data transformations, while **Api** and **ApiOperation** drive workflows via **Flow**, linked to **Connection** for platform integration. **Webhook** enables real-time triggers, **SyncLog** tracks activities, and **User** with **Subscription** manages user-specific flows, creating a **modular, scalable**

structure for SyncStream's operations, though it could benefit from additional **normalization** and relationship enhancements like **user-connection links**.

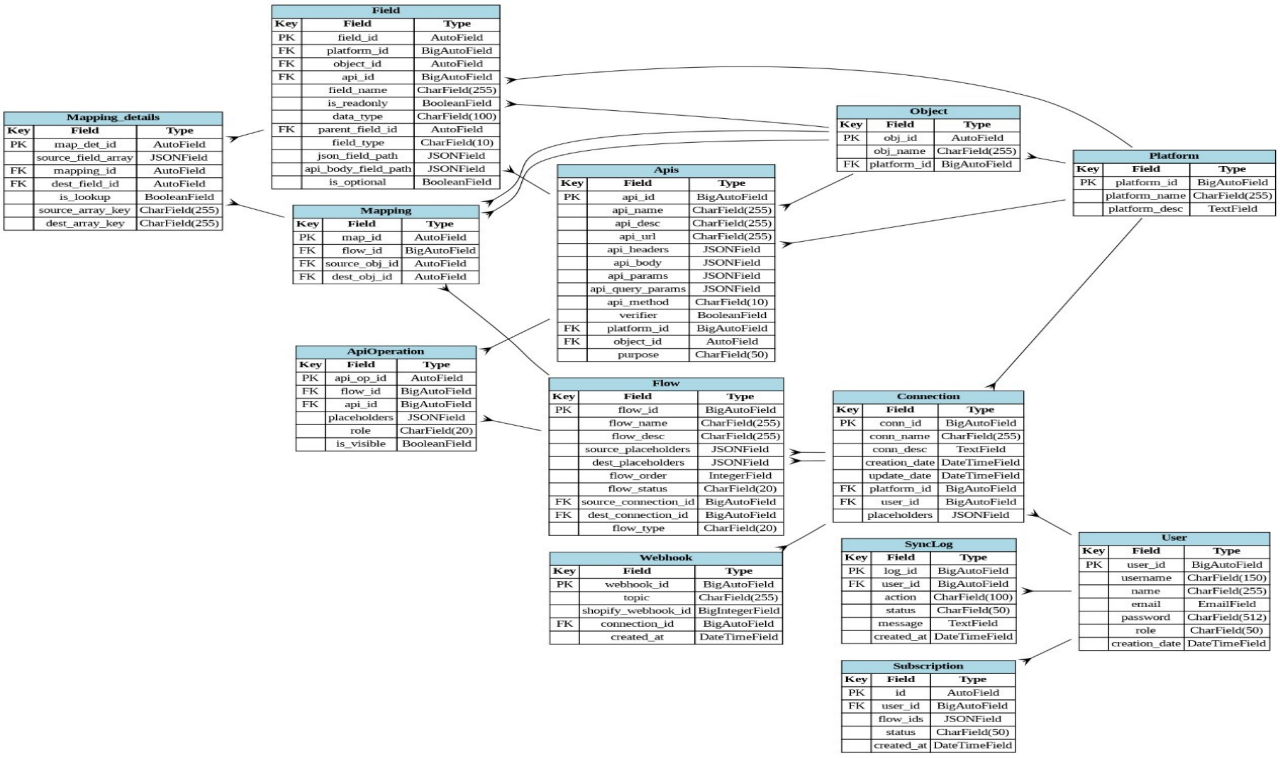


Figure 4.12: Entity-Relationship Diagram

Chapter 5

Development Methodologies

5.1 Introduction

SyncStream is developed as a modular platform with the capability of scaling up to meet the industry's requirements for sync capabilities across multiple systems. Currently, the focus is on constructing a solid system architecture working with custom configurations, intuitive workflows, and advanced integration management. The system offers different levels of complexity regarding users, platforms, and data objects aimed towards providing simplified user experiences. Focus is placed on how to achieve clean separation of concerns, role-based access controls, data views, and responsiveness, which guarantees the platform will remain agile, easy to maintain, and robust as it evolves with the demands placed on the system.

5.2 Frontend

SyncStream's frontend offers a clear and user-friendly interface that allows users to interact with the system at a high level. It is the main entry point where users can administer their platforms, set up synchronization, check logs, and define custom objects or APIs. The layout structure facilitates responsiveness, ease of navigation, and clarity which is important for smooth user experience across various roles. Unlike the rest of the interface which changes based on user role, some elements of the interface are available or hidden based on user permissions. Hence, administrators and customers are provided different scopes of functionalities.

5.2.1 React JS

React JS is a JavaScript library for building user interfaces, especially for single-page applications (SPAs) where dynamic updates are required.To efficiently manage React's complex state and handle the rendering of its UI components, SyncStream employs React at the frontend. With React, developers construct components that allow for easy modularization of user interfaces. This makes the application structure more scalable and helps maintain it better in the long run. Synchronization is especially critical for SyncStream as customers utilize dynamic

data, interact with many platforms, or navigate through the synchronization logs, hence immediate rendering is vital. React Virtural DOM guarantees that unnecessary performance lags are avoided and UIs are seamless.

5.2.2 TypeScript

TypeScript is a superset of JavaScript with static typing that improves the development workflow by detecting possible problems at compile-time, prior to execution. It offers enhanced documentation, improved code quality, and better developer tools like autocomplete and type checking, which makes writing code much easier. In SyncStream, TypeScript is used for complex data structures to ensure type safety, increase reliability during development, and mitigate the chances of runtime errors. Its usefulness is elevated in large applications like SyncStream that have numerous user APIs, interactions, and other configurations because it improves predictability and decreases unexpected issues during coding.

5.2.3 CSS

CSS is used to apply responsive styles and layout properties. Through the use of Flexbox, Grid Layout, media queries, and transitions, the web application supports smooth resizing across multiple screen sizes and devices. Advanced features like animations and shadows added visual polish to user interactions. A consistent design language was maintained using CSS variables and custom properties, enhancing modularity and reducing code duplication.

5.2.4 Tailwind CSS

Tailwind CSS is a utility-first CSS framework that aids in fast UI development using preexisting classes directly in the HTML, instead of having to use special CSS on every element. Quick styling and customization are made possible by this without relying on large CSS files. Tailwind CSS in SyncStream offers a flexible means of creating clean, clean, and responsive interfaces with the frontend automatically adjusting to various screen sizes. Its utility-based paradigm is especially its weight in gold when writing specialized layouts and applying design consistency across the platform, and thus is well-suited to complicated UIs such as the platform builder, sync setup, and log viewer.

5.2.5 Shadcn

Shadcn is a component library based on Radix UI and Tailwind CSS to provide extremely customizable, completely accessible UI components for React apps.We employed shadcn/ui in SyncStream to assist us in creating a fresh, new interface without having to begin each component from scratch. It's a component library which also plays beautifully with React and Tailwind CSS, and the thing that makes it so unique is how incredibly flexible it is—you don't receive merely pre-made buttons or forms, you receive snippets of code which are completely open to you and easy to modify. That is, we could easily add things like dialogs, dropdowns, tabs, and inputs, but still fine-tune them to our app's look. Because the components are also designed for accessibility too, it actually saved a lot of time from us ensuring that everything works for everybody. In general, it had us work faster and the UI consistent throughout the entire platform.

5.2.6 Icon Libraries

To improve UX and user comprehension, multiple icon libraries were used. Font Awesome, Tabler Icons, Feather Icons, and Material Icons were incorporated to visually represent actions and statuses across modules like dashboard, HRM, CRM, and finance. Icons were chosen for clarity and uniformity, and their scalable nature ensured responsiveness across devices.

5.2.7 Axios

Axios is an HTTP client for making backend requests from the front end using promises. It makes it easy to send asynchronous requests like downloading sync logs or platform information and process responses in a manner that will perfectly complement React component lifecycle. Axios is utilized in SyncStream to interact with backend APIs developed with Django REST Framework to initiate requests for platform creation, synchronization requests, and data retrieval like logs or configuration settings. Axios also offers functionality such as request cancellation, automatic JSON parsing, and error handling, which makes it easier to handle API calls effectively.

5.3 Backend

SyncStream's back-end was built to offer a secure, scalable, and stable platform upon which all the capabilities of the platform would reside. It is capable of supporting data management, user authentication, role-based authorization, and dynamic system synchronization orchestration management. Great importance was put into creating a feature base for modularity that will be able to support the configuration and management of platforms, API configuration, and sync processes. The backend is also traceable and logs all operations, so it's not only visible but also easy to debug. Particular attention was given to structure the data models and APIs in such a manner that they facilitate flexibility as well as maintainability as the system evolves.

5.3.1 Django

Django is a high-level, abstract Python web framework that supports rapid development and clean, pragmatic design. Django makes up the backend of SyncStream, providing core functionalities like database interaction, user authentication, URL mapping, and admin functions. Its ORM facility integrates to make querying and relational data model management for platforms, APIs, logs, and users easier, making the system uniform and easy to maintain. Django's strong architecture and inherent security features make it the perfect tool to build a system such as SyncStream, which holds sensitive user information, role-based permissions, and an associated set of entities.

5.3.2 Django REST Framework (DRF)

Django REST Framework is an extension of Django to simplify building powerful and flexible RESTful APIs. DRF in SyncStream provides backend operations as accessible to the frontend to allow it to communicate with platforms, objects, APIs, sync flows, logs, and accounts. DRF allows serialization, viewsets, permissions, and authentication tools through which it is possible to have complete control over accessing and modifying data. It has a crucial role in ensuring a clean and secure bridge between frontend activity and backend logic and enabling easy synchronization workflows and well-structured communication throughout the application.

5.3.3 MySQL

MySQL is the relational database SyncStream's backend utilizes to keep all structured data such as users, platforms, API definitions, sync flows, and logs. It is an incredibly powerful, heavily utilized database system with a good reputation for scalability and performance. MySQL in SyncStream facilitates smooth and consistent storage of intricate relationships between various entities engaged in synchronization processes. Django supports MySQL through its ORM so that developers can manage the database by writing Python code without sacrificing data integrity, using constraints, and facilitating transactions required by precise sync processes.

5.3.4 MySQL Workbench

MySQL Workbench was used as the main graphical user interface to interact with and manage the project database. It offered a graphical view of SyncStream's relational schema, which was useful in validating and comprehending interdependent tables like users, platforms, APIs, and sync logs. The tool also gave the team the ability to execute raw SQL queries, review real-time data, and debug sync issues more effectively. Whether debugging failed operations or ensuring that data mappings were properly applied, MySQL Workbench helped the back-end team to have clean, correct, and well-structured data throughout the project.

5.3.5 Postman

Postman was a key utility that was utilized in the backend API development of SyncStream. It offered a clean, yet effective interface to test, debug, and confirm REST API endpoints in the full development process. It was utilized by developers to mimic multiple types of requests—GET, POST, PUT, and DELETE—and headers, tokens, and payloads in an attempt to ensure that the APIs were returning as expected under different conditions. Postman also enabled role-based access testing by quick switching of customer and admin tokens to get consistent enforcement of permissions. On the whole, it greatly streamlined back-end development by enabling smooth iterations and some manual testing before being integrated with front-end.

5.4 Version Control and Collaborative Workflow

5.4.1 Git

Git was utilized as a version control basis in SyncStream development. It enabled tracking of differences in the codebase, the development of features in parallel, and the ability to have an uncontaminated project development history. All developers worked on different branches, which allowed them to experiment freely and restricted potential damage to the core application. Git's branch and merge functionality was needed to keep the workflow in sequence, add new features, debug bugs, and provide code reviews. Regular committing and clean commit messages ensured that the process was clean and traceable.

5.4.2 Github

GitHub was used as a remote platform where project repositories were kept and team collaboration happened. It allowed a centralized system by which developers push code, create pull requests, and perform code reviews before merging changes into the main branch. GitHub issue tracking, project boards, and milestones were used to track tasks, assign tasks, and stay up-to-date with developments. The site also made it easy to roll back, enable collaborative editing of documentation, and make development transparent. In general, GitHub played the central role between local development and shared merging, making contributions easy, reviewable, and goal-oriented.

Chapter 6

Software Testing

6.1 Introduction

Software testing is essential for ensuring that **SyncStream** operates reliably, satisfies integration requirements, and delivers a seamless data synchronization experience. Through structured testing, we validate system functionality, performance, and security, while identifying and fixing bugs before deployment. The main objectives of testing are to:

- Detect and resolve bugs in sync, API, and user modules.
- Verify system reliability across multi-platform data flows.
- Confirm platform configurations and custom APIs work as expected.
- Ensure data security, accurate logging, and stable system response.

6.2 Test Plan and Procedures

1. Test Planning

Testing was focused on core SyncStream features:

- User authentication (admin/customer roles)
- Custom Platform registration and configuration
- Real-time and scheduled data synchronization
- Logging and user-specific log visibility

2. Test Case Design

Each test case was designed with:

- Scenario description

- Expected system behavior
- Pass/fail criteria

Both positive and negative test scenarios were developed.

3. Test Execution

- **Functional Testing:** UI flows, authentication, API creation
- **Integration Testing:** Platform-object-API flow validation
- **Security Testing:** Role-based access and endpoint protection

4. Defect Tracking & Resolution

Bugs were logged in Trello, prioritized by severity, and resolved by the development team. Regression tests were executed after fixes.

5. Final Evaluation

Final testing included regression testing, usability walkthroughs, and synchronization validation.

6.3 Features Tested

Customer Module

- Sign up / Login
- Register platforms and create objects
- Create and test custom APIs
- View sync logs

Admin Module

- Login
- View all users and platforms
- View full system logs

6.4 Testing Strategy

6.4.1 Structured Testing Approach

We applied black-box and white-box testing to evaluate:

- API behaviors and response validation
- Mapping logic and sync execution
- Role-based UI interactions

6.4.2 Exploratory Testing

Edge cases like unauthorized access or incomplete API definitions were tested to identify unexpected behaviors.

6.5 Static Testing

Code Reviews

Code quality and consistency were ensured through GitHub pull requests and peer reviews.

Design Reviews

Tailwind CSS-based UI was reviewed for responsiveness, clarity, and accessibility.

Walkthroughs

Key flows—platform creation, API sync, and log tracking—were demonstrated in walkthroughs with the team.

6.6 Dynamic Testing

Unit Testing

Covered core components like `User`, `Platform`, `API`, and `SyncLog` models.

Integration Testing

Validated flows from user actions to log output, using tools like Postman for endpoint verification.

System Testing

Simulated real-world scenarios like syncing customer data between platforms and log traceability checks.

6.7 Resources and Environment

Testing was performed by internal team members only (Alpha Testing).

Environments:

- Development: Unit and initial integration testing
- Staging: System testing and final validation

6.8 Key Metrics

- **Defect Density:** Bugs per module
- **Test Coverage:** Percentage of executed vs planned test cases
- **Resolution Time:** Time taken to identify and fix bugs
- **System Uptime:** Measured system stability and availability

6.9 Tools and Setup

Testing Tools:

- **API Testing:** Postman
- **Bug Tracking:** Trello
- **Performance Testing:** Simulated load tests via custom scripts

Chapter 7

Conclusions

7.1 Summary

The primary aim of this project has been to develop an efficient integration platform capable of addressing contemporary business needs while preparing for future technological challenges. The project has set objectives to explore the evolution of integration technologies, identify their limitations, and design a system that ensures flexibility, scalability, and ease of use.

Throughout the literature review, the evolution of enterprise system integration has been studied, starting from Enterprise Application Integration (EAI), through Service-Oriented Architecture (SOA), to RESTful Web Services, and finally to modern Integration Platform as a Service (iPaaS) solutions and automated data mapping techniques. Each phase of evolution has highlighted critical challenges, including system complexity, scalability issues, and performance bottlenecks.

It has been observed that EAI addressed early integration challenges but faced issues related to system complexity and maintenance. SOA introduced flexibility and reusability, offering standardized services but still required careful management of service dependencies. RESTful web services have brought significant performance and scalability benefits, especially with the introduction of lightweight data formats like JSON. The development of Web-Oriented Integration Architecture (WOIA) has further enhanced REST's suitability for enterprise environments. Finally, the emergence of iPaaS platforms and intelligent data mapping methods has provided scalable, cloud-based, user-friendly solutions that significantly reduce the technical burden of system integration.

The project has successfully drawn upon these historical advancements to propose a modern integration approach that balances ease of use, technical capability, security, and extensibility. The findings have reaffirmed that an effective integration platform must not only streamline current workflows but also lay the groundwork for future digital transformation.

7.2 Recommendations for Future Work

Security and privacy measures could also be enhanced by incorporating more robust encryption methods and real-time monitoring for detecting data breaches during integration processes. Furthermore, future work could focus on building a more adaptive integration platform that automatically adjusts its resource consumption based on workload demands, improving cost-efficiency in cloud environments.

Integration of Artificial Intelligence (AI) can be a significant milestone in enhancing the performance of the system and making it easier to operate, which can bring revolution in the field of Platform-Integration Systems.

Working on these things can definitely enhance the efficiency of SyncStream and can provide its users a better experience overall.

References

[1] D. S. Linthicum, *Enterprise Application Integration*. Addison-Wesley Professional, 2000.

[2] G. Hohpe and B. Woolf, *Enterprise Integration Patterns*. Addison-Wesley Professional, 2004.

[3] Z. Irani, M. Themistocleous, and P. E. D. Love, "The impact of enterprise application integration on information system lifecycles," *Information & Management*, vol. 41, no. 2, pp. 177-187, 2003. doi: 10.1016/S0378-7206(03)00046-6. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378720603000466.

[4] X. Chen and H. Xu, "One Service-oriented Architecture Based Enterprise Application Integration Platform," *The 9th International Conference on Advanced Communication Technology*, Gangwon, Korea (South), 2007.

[5] M. Jiang and A. Willey, "Service-Oriented Architecture for Deploying and Integrating Enterprise Applications," *5th Working IEEE/IFIP Conference on Software Architecture (WICSA'05)*, Pittsburgh, PA, USA, 2005.

[6] J. Meng, S. Mei, and Z. Yan, "RESTful Web Services," *2009 International Conference on Computational Intelligence and Software Engineering*, 2009.

[7] S. Daniel, T. Przemyslaw, and H. Lars, "Integrating Information Systems Using Web Oriented Integration Architecture and RESTful Web Services," *2010 6th World Congress on Services*, Miami, FL, USA, 2010.

[8] K. Hyrynsalmi et al., "Adoption of Cloud Integration Services: The Case of Finnish Software Companies," *Proceedings of the 2017 International Conference on Cloud Computing*, 2017.

[9] E. Ebert, K. Weber, and S. Koruna, "Integration Platform as a Service," *Business & Information Systems Engineering*, vol. 59, no. 6, pp. 375–379, 2017.

[10] M. Stephen and S. Mitchell, "The role of iPaaS in future enterprise integrations: Simplifying complex workflows with scalable solutions," *International Journal of Trend in Scientific Research and Development (IJTSRD)*, vol. 5, no. 6, pp. 1999-2014, Oct. 2021. [Online]. Available: www.ijtsrd.com/papers/ijtsrd47504.pdf.

[11] M. Birgersson, G. Hansson, and U. Franke, “Data Integration Using Machine Learning,” *2016 IEEE 20th International Enterprise Distributed Object Computing Workshop (EDOCW)*, Vienna, Austria, 2016.

[12] Hyrynsalmi, S., Seppänen, M., Suominen, A. (2017). The Current State of Cloud-Based Integration Platforms.ResearchGate Publication

[13] Monus, A. (2022). ”SOAP vs REST vs JSON - a 2023 comparison.” Raygun Blog. Raygun ”SOAP vs. REST - Difference Between API Technologies.” AWS. Amazon Web Services, Inc. ”SOAP vs. REST Comparison: Differences in Performance, APIs More.” Stackify.

[14] Gartner Research. (2023). Magic Quadrant for Enterprise Integration Platform as a Service (iPaaS). (Major report used in the industry for iPaaS rankings and analysis)

[15] Hohpe, G., Woolf, B. (2003). Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Addison-Wesley.

[16] Ebert, C., Gallardo, G., Hernantes, J., Serrano, N. (2016). DevOps for Digital Leaders: Reignite Business with a Modern DevOps-Enabled Software Factory. IEEE Software.

[17] Shvaiko, P., Euzenat, J. (2013). ”Ontology Matching: State of the Art and Future Challenges.” IEEE Transactions on Knowledge and Data Engineering, 25(1), 158–176