A Report on Major Project

**SupportSync - AI Driven Automated Ticket Resolution**

*SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF*

**BATCHELOR OF TECHNOLOGY IN**

**COMPUTER ENGINEERING OF**

**VISHWAKARMA INSTITUTE OF TECHNOLOGY**

## Savitribai Phule Pune University

*BY*

### Shitole Sahil Manoj (PRN. 12110167) Bokariya Sujal Samir (PRN. 12220214) Tambe Ajinkya Sharad (PRN. 12220014)

**Thakare Prathamesh Prabhakar (PRN. 12220016)**

*UNDER THE GUIDANCE OF*

Prof. Dr. Sandip R. Shinde



DEPARTMENT OF COMPUTER ENGINEERING

BANSILAL RAMNATH AGARWAL CHARITABLE TRUST’S

VISHWAKARMA INSTITUTE OF TECHNOLOGY, PUNE - 411037

(An Autonomous Institute affiliated to Savitribai Phule Pune University) 2024 - 2025

**BANSILAL RAMNATH AGARWAL CHARITABLE TRUST’S**

**VISHWAKARMA INSTITUTE OF TECHNOLOGY**

### (An Autonomous Institute affiliated to Savitribai Phule Pune University)

**PUNE – 411037**



**C E R T I F I C A T E**

This is to certify that the Major Project titled **SupportSync** submitted by group members **Shitole Sahil Manoj (12110167), Bokariya Sujal Samir (12220214), Tambe Ajinkya Sharad (12220014), Thakare Prathamesh Prabhakar (12220016)** in partial fulfillment for the award of Degree of Bachelor of Technology in Computer Engineering of Vishwakarma Institute of Technology, Savitribai Phule Pune University. This Major Project is a record of bonafide work carried out by them under my guidance during the academic year 2024-25.

## Guide Head

**[Prof. Dr. S. R. Shinde] [Prof. Dr. S. R. Shinde]**

Department of Computer Engineering

## Sign of External Examiner Date:

Bansilal Ramnath Agarwal Charitable Trust’s **Vishwakarma Institute of Technology, Pune-37 Department Of Computer Engineering**

Project Synopsis

Group No : G42 Group Members:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Roll**  **No** | **Name** | **Class** | **Contact No** | **Email-ID** |
| 27 | Shitole Sahil Manoj | BTech - D | 7620919774 | [sahil.shitole21@vit.edu](mailto:sahil.shitole21@vit.edu) |
| 73 | Bokariya Sujal Samir | BTech - D | 9309708418 | [sujal.bokariya22@vit.edu](mailto:sujal.bokariya22@vit.edu) |
| 75 | Tambe Ajinkya  Sharad | BTech - D | 7720837166 | [ajinkya.tambe22@vit.edu](mailto:ajinkya.tambe22@vit.edu) |
| 76 | Thakare Prathamesh  Prabhakar | BTech - D | 7843087679 | [prathamesh.thakare22@vit.edu](mailto:prathamesh.thakare22@vit.edu) |

Academic Year : 2024-25

Project Title : SupportSync: AI-Driven Automated Ticket Resolution Project Area : Software Engineering / AI Expert Systems / GenAI Sponsor Company :

Company Address :

Internal Guide : Prof. Dr. Sandip R. Shinde

Name of the External Guide:

Contact No:

### Signature of Internal Guide

**ACKNOWLEDGEMENT**

It is our genuine pleasure to express our sincere gratitude to our project guide Prof. (Dr.) Sandip R. Shinde for inspiring our interest in this area of research. We are grateful for his expertise, scrutiny and constant guidance, which has helped us to a great extent in all the stages of our project. The successful and timely completion of this project was accomplished by the combined efforts of all group members.

Shitole Sahil Manoj (PRN. 12110167) Bokariya Sujal Samir (PRN. 12220214) Tambe Ajinkya Sharad (PRN. 12220014)

Thakare Prathamesh Prabhakar (PRN. 12220016)

**INDEX**

|  |  |  |
| --- | --- | --- |
| **Sr.no.** | **Table of contents** | **Page no** |
| 1 | Software Project Synopsis | 6 |
| 2 | Feasibility Status Report | 8 |
| 3 | Use Case Analysis Document | 16 |
| 4 | Software Requirements Specification | 20 |
| 5 | Software Project Plan | 28 |
| 6 | Behavior: Architecture Diagram | 33 |
| 7 | Behavior: Class Diagram | 36 |
| 8 | Behavior: Sequence Diagram | 38 |
| 9 | Software Implementation Document | 40 |
| 10 | System Testing Document | 45 |
| 11 | Conclusion | 50 |
| 12 | References | 51 |

|  |  |  |
| --- | --- | --- |
| **Sr.no.** | **List of figures/Tables** | **Page no** |
| 1 | Literature Review | 11 |
| 2 | Alternative v/s SupportSync | 15 |
| 3 | Use Case 1 | 17 |
| 4 | Use Case 2 | 18 |
| 5 | Use Case 3 | 19 |
| 6 | Problem Statement | 22 |
| 7 | Key Portability Characteristics | 26 |
| 8 | Project Goals | 29 |
| 9 | Schedule and Milestones | 31 |
| 10 | Deliverables | 32 |
| 11 | Architecture Diagram | 34 |
| 12 | Class Diagram | 37 |
| 13 | Sequence Diagram | 39 |
| 14 | System Workflow | 43 |

**Vishwakarma Institute of Technology, Pune**

**SupportSync: AI-Driven Automated Ticket Resolution**

# Software Project Synopsis

### Approvals Signature Block

|  |  |  |
| --- | --- | --- |
| **Project Responsibility** | **Signature** | **Date** |
| *Project Guide (Internal)* |  |  |
| *Project Guide (External)* |  |  |
| *Documentation Leader* |  |  |

Department of Computer Engineering

**1. Context**

Imagine a large organization dealing with a constant flow of support tickets from both customers and internal teams. With each passing day, the volume of tickets grows, spanning across multiple platforms like JIRA, ServiceNow, and others. These systems serve their purpose tracking issues, assigning tasks, and facilitating resolutions, but what about the pile of tickets it has gathered then. Employees spend hours sorting through these tickets, searching for relevant information, and trying to understand the context of past issues. Sometimes, the answers they need are buried deep within the system or scattered across different platforms, leading to missed details or repeated troubleshooting. As a result, resolutions are delayed, and the entire support process becomes a game of catch-up.

**2. Problem**

The key issues with traditional ticketing systems are the inefficiency in processing tickets, the lack of intelligent automation, and the siloed nature of ticketing platforms. Employee often have to manually sift through tickets, seeking relevant information and solutions. This results in delayed resolutions, missed opportunities for proactive support, and a lack of context for quickly understanding ticket history. Additionally, the systems are not equipped to leverage past data and AI-based solutions to assist in troubleshooting and resolution, further exacerbating the problem. The absence of a unified platform to integrate multiple ticketing systems with advanced AI tools, like embeddings and natural language processing, further complicates support workflows.

**3. Solution**

To address these challenges, SupportSync offers a comprehensive solution by acting as an independent platform that integrates AI-driven automation with existing ticketing systems like JIRA and ClickUp. It serves as a unified hub that pulls support tickets from these platforms, enabling intelligent management and resolution of issues. Once the tickets are gathered, SupportSync stores them in a vector database, which facilitates the

analysis of both historical and new ticket data. When a new support ticket is raised, SupportSync compares it against past issues in real time, offering relevant, context-aware suggestions. The system then passes this analysis and similar tickets to an AI large language model (LLM), which refines the solution to provide more accurate, contextually relevant responses, enhancing the overall user experience.

If the initial solution provided by the system does not fully resolve the issue, SupportSync includes a chatbot that allows users to engage in further discussion. This chatbot not only refines the solution but also supports multimedia, enabling users to upload documents such as PDFs to provide additional context. By incorporating this extra information, the chatbot is able to deliver more accurate, detailed, and context-aware solutions, improving response accuracy and accelerating the resolution process.

**Vishwakarma Institute of Technology, Pune**

**SupportSync: AI-Driven Automated Ticket Resolution**

# Feasibility Study Report

### Approvals Signature Block

|  |  |  |
| --- | --- | --- |
| **Project Responsibility** | **Signature** | **Date** |
| *Project Guide (Internal)* |  |  |
| *Project Guide (External)* |  |  |
| *Documentation Leader* |  |  |

**1. INTRODUCTION**

The rapid advancement of technology has led to an increased demand for efficient IT support systems capable of resolving issues in real-time. Traditional methods of handling IT tickets, such as manual searching and problem resolution, are often time-consuming and redundant, leading to inefficiencies in the IT support process. With the growing complexity of technical issues and the need for faster resolutions, there is a critical requirement for more intelligent, automated systems that can improve both speed and accuracy in ticket management.

**1.1 Purpose**

The purpose of this feasibility study is to assess the viability of the Support Sync application in addressing the challenges faced by IT professionals when resolving support tickets. The study will evaluate the system's potential to streamline ticket resolution by leveraging historical data, machine learning, and automation. It aims to provide insights into the application's capability to reduce redundancies, improve efficiency, and enhance the problem-solving process within IT support teams. This document will also analyze technical feasibility, operational effectiveness, and the economic aspects of implementing the system.

**1.2 Methodology**

The methodology section outlines the approach taken to conduct this feasibility study This approach combines qualitative (based on observations and feedback) and quantitative (based on data and numbers) techniques to thoroughly evaluate the project’s potential and to understand both its strengths and limitations. Here’s a breakdown of each key step:

The development of Support Sync begins with data collection, where we aim to understand current IT support processes and potential user needs. This involves analyzing how organizations handle support tickets, examining the tools and techniques employed, and gathering feedback from potential users. Additionally, a review of relevant literature and examples of similar systems helps identify gaps in existing practices and highlights areas where Support Sync can add value.

Next, system evaluation focuses on assessing Support Sync’s technical features and capabilities. Each component, such as integrations with platforms like Jira, ClickUp, and Pinecone, is thoroughly analyzed to evaluate its functionality in ticket retrieval, data storage, and automated solution generation. This step ensures that the system aligns with project objectives and addresses gaps in existing solutions.

In parallel, a research review is conducted to gather insights from studies on IT support automation and machine learning models. This includes reviewing research papers, articles, and case studies related to ticket management and similar systems. By examining successes, failures, and best practices, this step helps validate the system design and ensures it aligns with proven approaches while identifying potential pitfalls.

To mitigate challenges, a risk assessment is undertaken to identify potential obstacles in implementing Support Sync. Key risks such as system compatibility, data security, and user adoption are analyzed. Understanding these risks early enables us to plan effective safeguards, ensuring the project remains robust and resilient.

Finally, a cost-benefit analysis evaluates the feasibility and value of implementing Support Sync. By estimating the resources needed for implementation and maintenance and comparing them with potential savings, such as time saved on ticket resolutions, improved efficiency, and reduced repetitive tasks, we assess whether the benefits justify the investment. If the analysis shows a favorable outcome, it reinforces the project’s feasibility and overall value.

**2. General Information**

**2.1 Technical Feasibility**

* **Limitations of Current Systems:** Traditional ticketing systems like Jira and ServiceNow, while effective for basic ticket management, depend heavily on manual processes for categorization and resolution. This reliance on human intervention slows down operations, particularly when technicians cannot access similar historical tickets. Similarly, static knowledge bases provide a repository of solutions but lack real-time integration with active tickets, often making it difficult to find tailored solutions for unique or complex issues. Basic AI-powered solutions have introduced some level of automation, but their capabilities are largely limited to ticket categorization and prioritization without deep contextual analysis.
* **Proposed Technical Capabilities of Support Sync:** Support Sync is designed to overcome these shortcomings by leveraging advanced AI technologies like the Llama-3.2 model. This enables it to analyze the context of each ticket and generate dynamic, relevant solutions. Seamless integration with platforms like Jira, ClickUp, and Pinecone ensures efficient ticket retrieval, data storage, and solution automation. The system is also scalable, supporting organizations as their ticket volume and complexity grow.
* **Addressing Technical Challenges**: Critical technical challenges, including system compatibility, data security, and AI performance, will be proactively managed. Robust API designs will ensure compatibility with existing platforms, while encryption and secure access controls will safeguard sensitive data. Regular updates and retraining of the AI model will maintain system relevance and accuracy over time.

Table 1: Literature Review

|  |  |  |  |
| --- | --- | --- | --- |
| Sr.  No. | Name of the Paper | **Publication Date** | **Objective, Aim, and Theme** |
| 1 | How can a company enhance employee satisfaction and time effectiveness by enabling AI in their ticketing system? | 15th August 2022 | This thesis explores how AI in ticketing systems can improve employee satisfaction by automating processes. It focuses on reducing workloads and enhancing service delivery in support and customer service by leveraging AI for better IT service ticket management. [1] |
| 2 | Automatic problem extraction and  analysis from unstructured text in IT tickets | Jan-Feb 2017 | The paper uses data mining and NLP to automate problem extraction from IT tickets, reducing human labor in IT service management and improving efficiency and cost-effectiveness. [2] |
| 3 | From Text to Recommendations: How Vector Databases are Revolutionizing Personalized Content Delivery | April 2024 | This paper explores how vector databases improve recommendation systems by matching content to user sentiment and the future potential of vector databases in content personalization. [3] |
| 4 | Evaluating the effectiveness of static word embeddings on | November 2020 | The paper examines the use of static word embeddings like word2vec for classifying IT support tickets, finding that quality |

|  |  |  |  |
| --- | --- | --- | --- |
|  | the classification of IT support tickets |  | embeddings can capture semantic relationships and improve classification accuracy. [4] |
| 5 | LLM-based Approaches for  Automatic Ticket Assignment: A Real- world Italian Application | November 2023 | This paper demonstrates how LLMs like GPT-4 can automate IT ticket assignment using zero-shot, few-shot, and ensemble learning to accurately route tickets and reduce the need for human intervention, enhancing service efficiency. [5] |
| 6 | IT ticket  classification: The simpler, the better | 21st October 2020 | This study compares the effectiveness of TF-IDF and linguistic features for IT ticket classification, suggesting that simple algorithms with well-chosen features can provide accurate, explainable results that outperform complex models. [6] |
| 7 | Optimizing IT Operations Through AI - A  Comprehensive Framework | January 2023 | The paper examines how AI, particularly LLMs, can optimize IT operations by automating incident response and resource allocation, aiming to improve decision- making, service quality, cost-efficiency, and operational performance in IT service management. [7] |
| 8 | Proactive Customer Support: Re- Architecting a  Customer | 6th January 2024 | This paper explores how AI and ML can make CRM systems more proactive by anticipating customer needs and resolving  issues in real-time, with a focus on future |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Support/Relationship Management  Software System Leveraging Predictive Analysis/AI and ML |  | trends, ethical considerations, and the potential to improve customer satisfaction and relationships. [8] |

**2.2 Operational Feasibility**

1. **Current Operational Inefficiencies:** The manual nature of current ticket resolution workflows creates inefficiencies, requiring IT teams to spend excessive time searching for solutions or resolving issues from scratch. Static knowledge bases offer limited value as they are not integrated with live ticketing systems and do not adapt to the specific contexts of tickets. Basic AI solutions, while improving categorization and prioritization, do not adequately support more complex or unique ticket scenarios.
2. **Streamlining IT Operations with Support Sync:** Support Sync addresses these inefficiencies by automating ticket categorization and providing context-aware solutions based on historical data and advanced AI models. This reduces the workload on IT teams and accelerates ticket resolution times. The system also improves decision-making processes by presenting IT professionals with highly relevant, actionable insights.
3. **Facilitating User Adoption:** To ensure smooth integration into existing workflows, Support Sync will feature an intuitive interface tailored to user needs. Comprehensive training programs will equip IT professionals with the knowledge to use the system effectively. By aligning with current operational practices, Support Sync minimizes disruptions and ensures quick adoption across teams.

**2.3 Economic Feasibility**

1. **Cost of Implementation:** The initial costs of implementing Support Sync include investments in AI development, system integration, and testing to ensure reliability and effectiveness. Additional expenditures are expected for secure cloud infrastructure, ongoing maintenance, and training programs for IT staff.
2. **Potential Cost Savings:** Support Sync offers significant savings by automating repetitive tasks, reducing labor requirements, and improving overall productivity. The ability to generate accurate, context-specific solutions reduces errors and avoids costs associated with repeated troubleshooting. By minimizing downtime caused by unresolved tickets, the system further enhances organizational efficiency.
3. **Ensuring a Positive ROI:** A phased implementation approach and detailed planning will mitigate risks such as budget overruns. Regular audits and updates will ensure the system remains relevant and well-utilized. Over time, the system’s scalability and operational benefits, such as reduced overheads and improved efficiency, will yield a high return on investment, justifying the upfront expenses.

**2.4 System Objectives**

The objectives of the Support Sync system are as follows:

* 1. **Improve Ticket Resolution Speed**: By retrieving historical tickets and their solutions from integrated platforms, Support Sync aims to reduce the time spent on resolving tickets.
  2. **Enhance Ticket Accuracy**: Using machine learning to suggest the most relevant solutions based on past tickets ensures higher accuracy and relevance in resolution.
  3. **Reduce Redundancy**: By automating ticket searches and solution suggestions, the system minimizes the redundancy of solving similar problems repeatedly.
  4. **Support Multi-modal Inputs**: The chatbot and system will accept various forms of input, including PDFs and images, to enhance context and accuracy in finding solutions.
  5. **Seamless Integration with Ticketing Platforms**: Support Sync will smoothly integrate with platforms like Jira and ClickUp, allowing easy synchronization of ticket statuses and solution suggestions.

**2.5 Assumptions and Constraints**

The following assumptions and constraints were considered during the project:

### Assumptions:

* + - * Tickets are filed with all the necessary fields (e.g., description, comments, and other contextual data) for the LLM to generate relevant solutions.
      * IT professionals are familiar with existing ticketing platforms like Jira or ClickUp, which the system integrates with.
      * The application will have access to historical data of previously resolved tickets to generate solutions for similar open tickets.

### Constraints:

* + - * The system's performance depends on the availability of sufficient historical data to train the LLM model.
      * The chatbot’s effectiveness is limited by the quality and completeness of the tickets it retrieves.
      * System integration with ticketing platforms is constrained by the API capabilities and restrictions of those platforms.
      * The application requires a stable internet connection for real-time data retrieval and processing.

**3. Alternatives**

Several alternative systems or approaches could be considered instead of the Support Sync system:

1. **Manual Ticket Resolution**: In this approach, IT professionals rely solely on their expertise and experience to solve incoming support tickets without assistance from advanced automation or AI systems. Skilled professionals can sometimes offer highly tailored, case-specific solutions. It also doesn’t require additional system implementation costs. This process is **time-consuming** and can be **inconsistent**—some issues may take much longer to resolve, especially if they’re complex or if the technician handling the ticket isn’t familiar with similar past cases. Manual resolution can also lead to **errors** and redundancy, as technicians may unknowingly spend time solving issues that have been addressed before.
2. **Standalone Knowledge Bases**: Organizations can maintain an internal knowledge base—a centralized repository of previously resolved issues, troubleshooting guides, FAQs, and other helpful resources. IT professionals can refer to this knowledge base to help solve tickets. Knowledge bases are typically static and don’t provide real-time or context-specific solutions for current tickets. Since they lack advanced integration and contextual understanding, technicians might struggle to find exactly what they need for complex or nuanced issues. Additionally, maintaining an up-to-date knowledge base requires **continuous manual updates**.
3. **Basic AI Chatbots**: Many organizations deploy simple AI-driven chatbots to assist with basic ticket resolution. These chatbots use programmed responses or basic machine learning algorithms to address frequently asked questions or common technical issues. While helpful for simple inquiries, these chatbots lack **integration with ticketing platforms** and don’t have **advanced capabilities** like deep learning or contextual understanding. They are generally limited to responding to pre-defined questions and cannot dynamically suggest solutions based on historical data, as Support Sync would.
4. **Consulting External IT Experts**: Organizations may choose to outsource certain support functions to external IT consultants who specialize in problem resolution for complex issues. Relying on external consultants can be **costly** and may lead to **delays** if they’re not immediately available. Additionally, for routine or frequently recurring issues, this approach becomes inefficient, as it doesn’t provide a scalable, automated solution like Support Sync would.

**3.1 *Comparisons of Alternatives***

Table 2: Alternatives v/s SupportSync

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. No** | **Alternative** | **Description** | **Comparison with Support Sync** |
| 1 | Manual Ticket Resolution | IT professionals handle tickets individually, relying on their own experience and knowledge. | Support Sync automates solution retrieval, reducing the time and effort required for repetitive issues. It prevents redundancy by suggesting previously solved tickets, while manual resolution often leads to inefficiencies and inconsistency, especially for complex issues. |
| 2 | Standalone Knowledge Bases | An internal repository with documented solutions and guides that technicians can search manually. | Support Sync integrates with live ticketing data and dynamically suggests solutions based on ticket context. Unlike static knowledge bases, it provides real-time, tailored solutions and doesn’t require manual  updates, making it more responsive to |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | current issues. |
| 3 | Basic AI Chatbots | Chatbots provide automated responses to frequently asked questions or common issues. | Support Sync is powered by advanced models like Llama-3.2, allowing it to understand ticket context and generate relevant solutions. Basic chatbots lack this depth and typically don’t integrate with ticketing platforms or use advanced machine |
| 4 | Ticketing System Upgrades | Enhancing existing ticketing systems with advanced search or machine learning for classification and prioritization. | Support Sync goes beyond ticket classification, offering historical data embedding and automated solution generation based on similar tickets. Simple system upgrades typically don’t provide this level of automation and can’t generate tailored solutions based on ticket history and context. |
| 5 | Consulting External IT Experts | Outsourcing issue resolution to specialized IT consultants for complex or technical issues. | While external consultants can offer expertise, they come with high costs and delays. Support Sync provides an in-house, scalable solution that quickly retrieves relevant solutions, improving efficiency without the ongoing expenses associated with external consulting. |

**Vishwakarma Institute of Technology, Pune**

**SupportSync: AI-Driven Automated Ticket Resolution**

# Use Case Analysis Document

**Approvals Signature Block**

|  |  |  |
| --- | --- | --- |
| **Project Responsibility** | **Signature** | **Date** |
| *Project Guide (Internal)* |  |  |
| *Project Guide (External)* |  |  |
| *Documentation Leader* |  |  |

## 1. Use Case Template

Table 3: Use Case 1

|  |  |  |
| --- | --- | --- |
| USE CASE 1 | User Authentication | |
| Goal | To allow users to securely log in and access the SupportSync platform | |
| Purpose | Authentication via Auth0 JWT to ensure only authorized users can  access the system. | |
| Preconditions | The user has a valid account in the system. | |
| Success  Condition | User is successfully authenticated and granted access to the  dashboard. | |
| Failed Condition | User is not authenticated and is presented with an error message. | |
| Postconditions | The user is logged into the system and redirected to the dashboard. | |
| Primary Actors | User, Auth0 | |
| Secondary Actors | None | |
| Trigger | The user submits their login credentials. | |
| DESCRIPTION | Step | Basic Course of Action |
| 1 | User enters valid credentials (email/password). |
| 2 | System sends credentials to Auth0 for verification. |
| 3 | Auth0 responds with a JWT token upon successful  authentication. |
| 4 | System validates the JWT token and logs the user in. |
| 5 | User is redirected to the dashboard page. |
| DESCRIPTION | Step | Alternate Course of Action |
| 1 | User enters invalid credentials (incorrect email or password). |
| 2 | System sends credentials to Auth0. |
| 3 | Auth0 responds with an error (invalid credentials). |
| 4 | System displays an error message and prompts the user to retry. |
| DESCRIPTION | Step | Extension Points |
| 1 | User has a valid session token. |

|  |  |  |
| --- | --- | --- |
|  | 2 | If the session expires, the user is prompted to log in again. |
| DESCRIPTION | Step | Error Scenario |
| 1 | User enters valid credentials, but there is an issue with the Auth0  API. |
| 2 | The system displays an error message: "Authentication service  unavailable." |
| 3 | The user is prompted to try again later. |

Table 4: Use Case 2

|  |  |  |
| --- | --- | --- |
| USE CASE 2 | Ticket Integration with Ticketing Platforms | |
| Goal | To fetch support tickets from integrated ticketing platforms (JIRA,  ServiceNow, etc.) and display them in the SupportSync. | |
| Purpose | Allow SupportSync to retrieve ticket data from external platforms  to integrate with the AI-driven solution system. | |
| Preconditions | The system has valid API keys and integration setup for ticketing  platforms. | |
| Success Condition | The system successfully fetches and displays tickets from the  connected ticketing platforms. | |
| Failed Condition | The system fails to fetch ticket data and shows an error message. | |
| Postconditions | The system displays the fetched tickets for the user to view and  work with. | |
| Primary Actors | User, Ticketing Platforms (JIRA, ServiceNow, etc.) | |
| Secondary Actors | None | |
| Trigger | The user navigates to the tickets section in the dashboard. | |
| DESCRIPTION | Step | Basic Course of Action |
| 1 | User navigates to the "Tickets" section. |
| 2 | The system sends API requests to the ticketing platforms |

|  |  |  |
| --- | --- | --- |
|  |  | (JIRA, ServiceNow). |
| 3 | The ticketing platforms respond with ticket data. |
| 4 | The system processes and displays the fetched tickets to the  user. |
| DESCRIPTION | Step | Alternate Course of Action |
| 1 | User navigates to the "Tickets" section. |
| 2 | The system fails to fetch data from the platforms due to an  API issue |
| 3 | The system displays an error message indicating failure to  fetch tickets. |
| 4 | The user is prompted to retry fetching tickets. |
| DESCRIPTION | Step | Extension Points |
| 1 | The system successfully fetches tickets from multiple  platforms. |
| 2 | The user can switch between different platforms to view  tickets (if applicable). |
| DESCRIPTION | Step | Error Scenario |
| 1 | The API for one of the ticketing platforms is down. |
| 2 | The system fails to fetch tickets from that platform but  proceeds to fetch from others. |
| 3 | An error message is shown for the failed platform, and the  user can continue viewing tickets from the other platforms. |

Table 5: Use Case 3

|  |  |
| --- | --- |
| USE CASE 3 | AI Solution Generation |
| Goal | To generate solutions for tickets using AI and embeddings. |
| Purpose | Allow the system to suggest possible solutions to tickets based on  the stored embeddings. |

|  |  |  |
| --- | --- | --- |
| Preconditions | The system has embeddings stored for past tickets and access to  an AI model for solution generation. | |
| Success Condition | The system generates a relevant solution for the selected ticket. | |
| Failed Condition | The system fails to generate a solution and shows an error message. | |
| Postconditions | The user is provided with a solution generated by AI. | |
| Primary Actors | User, AI Model (e.g., Llama-3.2-90b-text-preview) | |
| Secondary Actors | Pinecone (Vector Database) | |
| Trigger | The user selects a ticket to resolve. | |
| DESCRIPTION | Step | Basic Course of Action |
| 1 | The system retrieves the embeddings of the selected ticket and compares them with stored embeddings in the vector  database. |
| 2 | The system sends the embeddings to the AI model for  solution generation. |
| 3 | The AI model processes the ticket and returns a suggested  solution |
| 4 | The system displays the solution to the user. |
| DESCRIPTION | Step | Alternate Course of Action |
| 1 | User selects a ticket. |
| 2 | The system fails to generate a solution due to an issue with  the AI model |
| 3 | The system shows a message: "Unable to generate solution at  this time." |
| 4 | The user can retry or proceed with manual ticket resolution |
| DESCRIPTION | Step | Extension Points |
| 1 | The system allows users to further refine the solution through  the integrated chatbot. |

|  |  |  |
| --- | --- | --- |
|  | 2 | The solution generation process can be enhanced as more  data is available. |
| DESCRIPTION | Step | Error Scenario |
| 1 | The system fails to compare embeddings due to missing or  corrupted data. |
| 2 | The system shows an error: "Data retrieval failure." |
| 3 | The user is prompted to retry or escalate the issue. |

**Vishwakarma Institute of Technology, Pune**

**SupportSync: AI-Driven Automated Ticket Resolution**

# Software Requirements Specification

### Approvals Signature Block

|  |  |  |
| --- | --- | --- |
| **Project Responsibility** | **Signature** | **Date** |
| *Project Guide (Internal)* |  |  |
| *Project Guide (External)* |  |  |
| *Documentation Leader* |  |  |

**1. INTRODUCTION**

The introduction to this Software Requirements Specification (SRS) document outlines the essential aspects of SupportSync, an AI-driven SaaS platform designed to streamline IT support ticket resolution. The SRS provides detailed requirements for developers to build, test, and deploy SupportSync in alignment with user expectations and system goals. This section offers an overview of the project's purpose, scope, definitions, and references, setting the foundation for the requirements and constraints described in later sections. SupportSync integrates with platforms like JIRA, Salesforce, and ServiceNow to automate ticket handling and resolution, enhancing productivity for IT professionals and support teams.

**1.1 Purpose**

This **Software Requirements Specification (SRS)** section provides a detailed overview of **SupportSync**, outlining the functional and non-functional requirements, as well as the design and architecture of the system. SupportSync is a ticket resolution platform designed to help IT professionals solve recurring issues they face while developing software by providing AI-powered solutions based on past resolved issues.

This document is intended to guide the development and deployment of the platform, ensuring that the end product meets the business needs of the stakeholders, especially IT professionals who use ticketing systems like JIRA, ClickUp, or other similar platforms. It serves as a reference for developers, system architects, project managers, and other stakeholders involved in the implementation and operation of the system.

**1.2 SCOPE**

The software product **SupportSync** is a web-based platform designed to help IT professionals resolve recurring issues in projects by providing AI-driven solutions based on previously solved tickets. It integrates with ticketing platforms such as **JIRA**, **ClickUp**, and potentially others. The solution aims to achieve more on point solutions

in problem-solving by storing closed (resolved) tickets and offering context-driven solutions through embeddings and AI models like **Llama-3.2-90b-text-preview**.

The platform allows developers to fetch open and in-progress tickets, search for similar closed tickets, and get AI-generated solutions. Additionally, it supports chat functionality to explore more detailed solutions, including handling multimedia files such as PDFs. The solution will initially be targeted at organizations using **JIRA**, but the platform’s design allows it to be extended to other ticketing platforms as well.

The primary benefits of **SupportSync** include:

* Reducing time spent on solving previously resolved issues.
* Automating ticket resolution by providing AI-generated solutions.
* Streamlining workflows for IT professionals by integrating with widely used ticketing platforms.
* Supporting multimedia input for more personalized solutions.

This section describes the requirements for **SupportSync**, ensuring that it meets the needs of both IT professionals and organizations in terms of functionality, security, reliability, and scalability.

**2. Overall Description**

The background affecting the SupportSync product is rooted in the increasing complexity and scale of IT systems, where teams frequently encounter recurring issues that have already been resolved in the past. Existing ticketing systems like JIRA, ServiceNow, and Salesforce are widely used to manage issues, but they lack a mechanism for easily accessing past solutions to similar problems. This inefficiency results in wasted time, duplicated efforts, and frustration among IT professionals. SupportSync aims to address this by leveraging AI, embeddings, and a vector database to quickly retrieve relevant solutions from resolved tickets, streamlining workflows and reducing redundancy in problem-solving. This background sets the foundation for the platform's functional and non-functional requirements.

**Problem Statement**

Table 6: Problem Statement

|  |  |
| --- | --- |
| Problem | The problem of recurring issues in IT projects being solved multiple times by different teams due to the lack of a centralized solution repository. |
| Affects | IT professionals, developers, and support teams using ticketing systems like JIRA, ServiceNow, and Salesforce. |
| Impact | The impact of this problem is wasted time, duplicated efforts, inefficiencies, and frustration among teams when solving the same issues repeatedly. |
| Successful Solution | A successful solution would streamline workflows, reduce redundancy, automate issue resolution, and allow quick access to previously resolved solutions, improving overall efficiency and reducing problem-solving  time. |

**2.1 Product Perspective**

SupportSync is part of a broader ecosystem of ticketing and support tools, integrating seamlessly with major platforms like JIRA, Salesforce, and ServiceNow to enhance ticket management and resolution. The platform leverages a modular architecture where each component, such as the ticketing interface, AI solution generation, and vector storage, functions cohesively to process, analyze, and respond to support tickets.

**2.1.1 System Interfaces**

**SupportSync** will integrate with existing ticketing systems such as **JIRA**, **ClickUp**, and potentially others. Through API tokens and secure connections, the platform will fetch relevant data from these systems to track the status of tickets (open, in progress, closed).

This integration eliminates the need for manual data entry and ensures that the platform has access to the most up-to-date ticket information.

The system interfaces are essential because they allow the platform to operate seamlessly alongside existing ticketing systems without requiring users to adopt a completely new system or alter their current workflows. It ensures that users can continue working within their familiar environments while benefiting from the AI-powered ticket resolution provided by the system.

**2.1.2 User Interfaces**

The platform will provide a web-based user interface (UI) that is accessible from any modern browser on any device (laptop, desktop, tablet). This UI will allow developers to:

* View open and in-progress tickets.
* Access closed tickets stored in the vector database.
* Receive AI-generated solutions for open tickets.
* Interact with the chatbot for more detailed problem resolution.

The web interface ensures **SupportSync** is platform-agnostic, meaning it works across different operating systems and devices, providing flexibility for users who might be working remotely or from various locations. This design choice enhances accessibility and ensures ease of use for all stakeholders.

**2.1.3 Hardware Interfaces**

Since **SupportSync** is a web-based application, there are no specific hardware interface requirements. The system will operate on a cloud-based infrastructure, meaning users can access the platform from any device with a modern browser and an internet connection. The platform's hardware requirements are flexible and scalable, ensuring it can run on various devices without being dependent on any specific hardware configuration.

**2.1.4 Software Interfaces**

The platform will interface with **JIRA**’s API and other ticketing systems, enabling seamless data retrieval. The system will pull ticket data, including open, in-progress, and closed tickets, and store relevant information in a **Pinecone** vector database. The vector database allows for efficient comparison and retrieval of similar tickets, helping the system provide relevant solutions based on past resolved issues. The use of external software systems like **JIRA** and **ClickUp** ensures that **SupportSync** can be integrated into existing workflows without requiring significant changes to the organization's infrastructure.

**2.1.5 Communications Interfaces**

**SupportSync** will use HTTPS for secure communication between the user’s device and the platform’s backend. This ensures that all data exchanged, including sensitive ticket information, is encrypted and secure. Security is a critical aspect of the platform, as the information involved could include sensitive customer data or intellectual property.

**2.1.6 Memory Constraints**

Given that SupportSync is cloud-based, memory constraints are not a major concern for the platform. Currently, we are utilizing **Firebase** for storing user data, which provides a scalable solution for handling large volumes of ticket data and user interactions. On the free tier of Firebase, we can leverage Firebase's cloud infrastructure to efficiently store and retrieve data, while maintaining the flexibility to scale as needed. As the platform grows, Firebase offers paid tiers that will allow for expanded capacity, ensuring that the system can continue to manage increasing amounts of data as more tickets are processed and more users interact with the platform. This approach ensures that the system remains performant and accessible even during the early stages, while also accommodating future growth without compromising service quality.

**3. Specific Requirements**

This section will define the specific software requirements for SupportSync to ensure that the system meets user needs and expectations. Each requirement will be documented in a way that is clear, complete, and verifiable, covering all aspects necessary for design and testing. Requirements will address all user inputs, system outputs, and functions triggered by these interactions, detailing how each component should operate within the system.

To ensure high-quality requirements, each will be:

* **Correct** and free of errors or assumptions,
* **Unambiguous** with clear terms,
* **Complete** without missing any details,
* **Consistent** with previous documentation and each other,
* **Ranked** for priority and flexibility,
* **Verifiable** to allow for testing,
* **Modifiable** to accommodate future changes, and
* **Traceable** to allow tracking from design to testing.

Each requirement will be uniquely numbered and cross-referenced with related documents for clear organization. By adhering to these principles, SupportSync’s requirements will support effective design, testing, and maintenance, ensuring that the platform’s functionality is robust and fully aligned with user needs.

**3.1 External Interfaces**

**SupportSync** will integrate with ticketing systems like **JIRA** through API tokens, which will allow the system to access and process ticket data. External integrations are crucial to ensure the platform’s functionality and to avoid requiring users to manually input ticket data. By integrating with **Pinecone** for storing ticket embeddings, **SupportSync** ensures that previously resolved tickets can be stored and retrieved efficiently. This integration helps provide developers with quick solutions to recurring issues based on past experiences.

**3.2 Functions**

The core functions of **SupportSync** include:

* + Fetching data from the ticketing system (e.g., **JIRA**) via API.
  + Storing and processing ticket data (especially closed tickets) by converting them into embeddings.
  + Searching for similar tickets from the vector database.
  + Generating AI-based solutions using the **Llama-3.2-90b-text-preview** model.
  + Providing a chatbot interface for further exploration and troubleshooting.
  + Handling multimedia input, such as PDFs, to help developers with more personalized solutions.

These functions are designed to streamline the issue resolution process by leveraging AI and embedding technology, making it easier for developers to find solutions to common issues and move on to new tasks more efficiently.

**3.3 Performance Requirements**

The system is designed to handle large datasets, including potentially thousands of tickets stored in the vector database. The response time for generating solutions should be under 5 seconds, ensuring that developers can quickly find and apply solutions to their issues. This performance requirement ensures that **SupportSync** can be used in high-pressure environments where time is of the essence.

**3.4 Logical Database Requirements**

The logical database for **SupportSync** is based on **Pinecone**, a vector database used to store ticket embeddings. The database must support fast retrieval and comparison of embeddings to provide developers with quick and accurate results when searching for similar closed tickets. The database will store all relevant information from closed tickets, including the ticket title, description, and resolution, to facilitate the matching

process. This ensures that solutions are generated based on both the content of the tickets and the context in which they were solved.

**3.5 Design Constraints**

The design of SupportSync will adhere to specific constraints influenced by standards, hardware, and system requirements to ensure compatibility, performance, and security. Key design constraints include:

1. **Industry Standards**: The system must comply with standard security and data protection regulations, like GDPR, to secure user data and maintain privacy.
2. **Scalability and Cloud-Based Hosting**: SupportSync is designed to operate on Vercel (frontend) and Railway (backend) cloud infrastructure, requiring a scalable architecture capable of handling variable loads and traffic.
3. **Compatibility**: The platform must support integration with various ticketing platforms (e.g., JIRA, ServiceNow) and be compatible with the cloud services and technologies in use, including Firebase (user data storage).
4. **Performance and Latency**: System response times should be optimized to handle real-time ticket processing and AI-based query generation without significant delay.

By accounting for these design constraints, SupportSync’s architecture will maintain high performance, security, and interoperability, supporting the robust SaaS solution needed for ticket management and AI-driven support.

**3.5.1 Standards Compliance.**

**SupportSync** will comply with industry standards for API security, authentication, and data privacy. This includes using **OAuth** for secure API access and ensuring that all data exchanged is encrypted. Compliance with standards such as **GDPR** ensures that **SupportSync** can be used globally while protecting user data.

**3.6 Software System Attributes**

SupportSync’s software system attributes are designed to ensure a secure, reliable, and accessible user experience across different environments. With a commitment to 99.9% uptime, the platform utilizes Vercel for frontend and Railway for backend deployment, providing automatic scalability and high availability. Security is central to the system, with data encryption, strict access controls, and secure storage safeguarding sensitive information. Additionally, portability is emphasized through development in React and FastAPI, which ensures compatibility across operating systems and cloud environments. SupportSync’s design also incorporates maintainability and testability, allowing for straightforward updates and robust testing to continuously meet high performance and reliability standards. These attributes collectively enable a resilient and versatile platform that meets the demanding requirements of IT support.

**3.6.1 Reliability**

Reliability is key for SupportSync, aiming for **99.9% uptime**. The platform will be deployed on **Vercel** (frontend) and **Railway** (backend) to ensure high availability and scalability. Both platforms provide automatic failover and scaling to meet demand, reducing downtime and ensuring uninterrupted service. Regular uptime monitoring and health checks will maintain continuous availability for users.

**3.6.2 Availability**

SupportSync will be hosted on **cloud infrastructure** provided by Vercel and Railway, ensuring **continuous availability**. Vercel’s global CDN ensures fast access for users, while Railway offers automatic scaling to handle varying loads. Multi-region support from both platforms ensures the platform remains operational globally, even during region-specific outages.

**3.6.3 Security**

SupportSync will implement **end-to-end encryption** for data transmission between the frontend and backend. **OAuth** and **JWT** will be used for secure user authentication. The platform will store ticket data securely with encryption at rest. Both Vercel and Railway follow strict security practices, such as secure deployment pipelines, vulnerability detection, and IAM for access control, ensuring the safety and privacy of user data.

**3.6.4 Portability**

The portability of SupportSync is essential to ensure that the software is adaptable across various environments and platforms. Given that the solution is built using portable, widely supported technologies like JavaScript (React for frontend) and Python (FastAPI for backend), portability is achievable across different operating systems and cloud environments.

Table 7: Key Portability Characteristics

|  |  |
| --- | --- |
| Characteristic | Rank |
| Correctness | 1 |
| Efficiency | 2 |
| Portability | 3 |
| Flexibility | 4 |
| Maintainability | 5 |
| Reliability | 6 |
| Testability | 7 |
| Usability | 8 |
| Availability | 9 |
| Integrity/Security | 10 |

### Breakdown of Portability-Related Factors:

* **Percentage of Host-Dependent Code:** The host-dependent code is minimized, estimated at less than 5%, mainly limited to minor configurations for deployment environments (e.g., Vercel for frontend and Railway for backend).
* **Use of Proven Portable Languages:** SupportSync is built with React (JavaScript) for the frontend and FastAPI (Python) for the backend, both of which are highly portable and supported across platforms.
* **Operating System and Compiler Requirements:** Since SupportSync is deployed on cloud services (Vercel and Railway), it does not rely on specific OS- dependent components, enhancing portability.

These factors will be measured by deploying SupportSync across different cloud platforms and verifying system functionality to ensure smooth portability across environments.

**Vishwakarma Institute of Technology, Pune**

**SupportSync: AI-Driven Automated Ticket Resolution**

# Software Project Plan

### Approvals Signature Block

|  |  |  |
| --- | --- | --- |
| **Project Responsibility** | **Signature** | **Date** |
| *Project Guide (Internal)* |  |  |
| *Project Guide (External)* |  |  |
| *Documentation Leader* |  |  |

**1. OVERVIEW**

This **Software Project Plan** (SPP) outlines the key elements of the **SupportSync** development process, including the project goals, scope, schedule, milestones, and deliverables. It is intended to serve as a blueprint for the development, deployment, and ongoing maintenance of the system. This document is meant for the development team, project managers, and stakeholders, providing them with clear expectations and a detailed overview of the project's trajectory.

**2. GOALS AND SCOPE**

**2.1 Project Goals**

Table 8: Project Goals

|  |  |  |
| --- | --- | --- |
| **Project Goal** | **Priority** | **Comment/Description/Reference** |
| **Functional Goals:** | 2 |  |
| Goal 1 |  | Integrate with major ticketing systems like JIRA, Salesforce, and ServiceNow for ticket retrieval and updates. |
| Goal 2 |  | Develop AI-based support for real-time ticket  resolution. |
| **Business Goals:** | 1 |  |
| Time to market |  | Ensure timely delivery within the 17-week timeline  for quicker implementation. |
| Cost |  | Optimize resources using a cloud deployment on  Vercel and Railway, minimizing infrastructure costs. |
| **Technological**  **Goals:** | 3 |  |
| Technical goal 1 |  | Use of cloud technologies, scalable architecture with |

|  |  |  |
| --- | --- | --- |
| **Project Goal** | **Priority** | **Comment/Description/Reference** |
|  |  | Firebase for user data, and integration with Pinecone for vector storage to ensure performance and reliability. |
| **Quality Goals:** | 2 |  |
| Quality goal 1 |  | Achieve 99.9% uptime to meet support team  availability needs. |
| Quality goal 2 |  | Data security and encryption during storage and transmission. |
| **Constraints:** | 2 |  |
| Constraint 1 |  | Compliance with data protection standards. |
| Constraint 2 |  | Environmental constraints may include server  locations due to latency considerations. |

**2.2 Project Scope**

**Boundaries:** The project will deliver a comprehensive support ticket resolution platform that integrates with selected ticketing systems, provides real-time AI responses, and features a chatbot for extended support. SupportSync will not cover ticketing platforms beyond JIRA, Salesforce, and ServiceNow in this phase.

**Long-Term Deliverables:** The initial deliverables may evolve, supporting further integrations and more advanced AI capabilities.

**Project Phases:** The project is divided into distinct phases from requirements analysis to post-deployment support to ensure structured development and implementation.

**Resources:** Approximately 5-7 people, including software developers, project managers, and AI specialists, will contribute to the project’s development.

**2.2.1 Included**

The **SupportSync** project will include the following features and capabilities:

1. **Integration with Ticketing Platforms**: The platform will support integration with popular ticketing systems like **JIRA** and **ClickUp**, allowing it to automatically retrieve and process ticket data from these systems.
2. **Embedding and Vector Database**: **SupportSync** will store closed (resolved) tickets in a vector database, which will be used to compare and retrieve similar tickets. This feature will allow the system to suggest solutions based on past tickets.
3. **AI-Powered Solution Generation**: The platform will use AI models to generate solutions for open tickets, taking into account previous ticket resolutions stored in the database. The AI model will analyze the ticket context and suggest the most appropriate solution based on similar past tickets.
4. **Search and Filter Capabilities**: Users will be able to search for open and closed tickets, using filters to narrow down results based on keywords, ticket type, status, or priority. This will allow users to quickly locate relevant tickets and solutions.
5. **Chatbot Interface for Further Assistance**: A chatbot will be integrated into the system to provide further assistance in resolving tickets. Users will be able to interact with the chatbot for additional information or clarification on suggested solutions.
6. **Support for Multimedia Files**: The system will be capable of processing and interpreting multimedia files, including PDFs and images, associated with tickets. This will allow developers to handle tickets with more complex problem descriptions.
7. **User Authentication and Security**: The system will include secure user authentication and authorization mechanisms to ensure that only authorized users

can access ticket data and solutions. This will include role-based access control to manage permissions.

**2.2.2 Excluded**

The following features will not be included in the **SupportSync** project scope:

1. **Support for Custom Ticketing Systems**: Initially, **SupportSync** will support only a limited number of third-party ticketing platforms, specifically **JIRA** and **ClickUp**. Other ticketing systems may be supported in the future, but they are not part of the initial release.
2. **Ticket Creation and Editing**: The platform will not allow users to create or edit tickets directly. It will only retrieve and process tickets that have already been created in the integrated ticketing systems.
3. **Advanced Reporting Features**: While **SupportSync** will provide basic ticket search and solution suggestion features, advanced reporting, analytics, and dashboards are excluded from the current project scope. These features may be considered for future updates.
4. **Full Automation of Ticket Resolution**: While the system will suggest solutions for tickets, full automation of the resolution process (e.g., automatically closing tickets based on AI-generated solutions) is not included in the initial release. This feature may be considered for future versions of the platform.
5. **Multilingual Support**: Initially, the platform will support only the English language for both the user interface and the AI-generated solutions. Multilingual support may be added in future versions.

**3. Schedule and Milestones**

Table 9: Schedule and Milestones

|  |  |  |  |
| --- | --- | --- | --- |
| **Milestones** | **Description** | **Milestone Criteria** | **Planned Date** |
| M0 | Start Project | Budget Release, Stakeholders identified,  Initial goals defined. | 2024-08-14 |
|  | Project goals  and scope defined | PRS or SRS reviewed, Integration points defined. | 2024-08-21 |
| M1 | Start Planning | Scope and system concept described;  Requirements Analysis completed. | 2024-08-28 |
|  | Requirements  Gathering | Conduct meetings with stakeholders,  Finalize SRS, Define integration points. | 2024-09-02 |
| M2 | Start System Design | System architecture design, Database schema, and API interfaces defined. | 2024-09-06 |
|  | Complete Design | Design infrastructure, Define  communication protocols, and core components. | 2024-09-08 |
| M3 | Start Execution | Development begins: Initial implementation of ticket system integration. | 2024-09-12 |
|  | Initial Development | Ticket integration, Vector DB integration, Initial UI development. | 2024-09-15 |
| M4 | Confirm Execution | AI-driven solution generation and chatbot integration. | 2024-09-19 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Milestones** | **Description** | **Milestone Criteria** | **Planned Date** |
|  | Alpha version | Basic functionality coded, UI complete, Embedding and solution generation  functional. | 2024-09-22 |
| M5 | Start Testing and QA | Perform unit and integration testing, Conduct User Acceptance Testing (UAT). | 2024-10-05 |
|  | Quality  Assurance | Bugs fixed, Issues addressed. | 2024-10-15 |
| M6 | Start Deployment | Deploy to production on Vercel and Railway, Documentation complete. | 2024-10-22 |
| M7 | Release Product | Product system tested, Documentation  reviewed | 2024-10-28 |
| M8 | Close Project | Final reports and project evaluation  completed | 2024-11-03 |

**4. Deliverables**

Table 10: Deliverables

|  |  |  |  |
| --- | --- | --- | --- |
| **Identifier** | **Deliverable** | **Planned Date** | **Receiver** |
| D1 | Software Requirements Specification | 2024-09-02 | Project Team,  Stakeholders |
| D2 | System Architecture Design | 2024-09-08 | Development  Team, Stakeholders |
| D3 | Database Schema and API Designs | 2024-09-08 | Development  Team, External API Partners |

|  |  |  |  |
| --- | --- | --- | --- |
| **Identifier** | **Deliverable** | **Planned Date** | **Receiver** |
| D4 | Integrated Ticketing System (JIRA,  ServiceNow integration) | 2024-09-15 | Project Team, End-  users |
| D5 | Vector Database Integration  (Pinecone) | 2024-09-15 | Project Team,  AI/ML team |
| D6 | User Interface (UI) for the Web  Platform | 2024-09-15 | End-users,  Stakeholders |
| D7 | AI-Driven Solution Generation and Chatbot Integration | 2024-09-22 | End-users, Stakeholders |
| D8 | Unit and Integration Testing Reports | 2024-10-15 | Project Team,  Stakeholders |
| D9 | Final Product Deployment to  Production | 2024-10-22 | End-users,  Stakeholders |
| D10 | End-user Training Material | 2024-10-28 | End-users,  Administrators |
| D11 | Final Project Documentation (User  Guides, System Manuals) | 2024-10-28 | End-users,  Stakeholders |
| D12 | Post-Deployment Support and  Monitoring Reports | Ongoing | End-users, Project  Team |

**Vishwakarma Institute of Technology, Pune**

**SupportSync: AI-Driven Automated Ticket Resolution**

# Behavior: Architecture Diagram

**Approvals Signature Block**

|  |  |  |
| --- | --- | --- |
| **Project Responsibility** | **Signature** | **Date** |
| *Project Guide (Internal)* |  |  |
| *Project Guide (External)* |  |  |
| *Documentation Leader* |  |  |

## Architecture Diagram

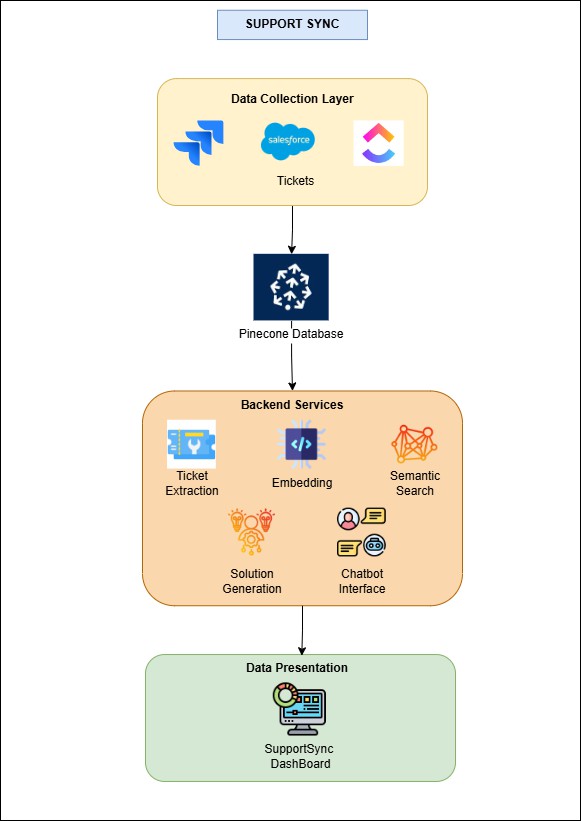


Figure 1: Architecture Diagram

The SupportSync architecture is designed to simplify issue resolution for IT teams through integrated systems and AI-powered support. At its core, the User Interface Layer enables developers to log in, access, and manage tickets directly. The Authentication Layer secures access with Auth0, restricting platform use to authorized users only. An Integration Layer connects SupportSync with ticketing systems like JIRA and ServiceNow to pull and store DONE tickets. These resolved tickets are embedded as vectors within the Data Storage Layer using Pinecone, allowing for quick searches to find relevant past solutions. The AI Solution Engine leverages OpenAI embeddings and an LLM model to generate solution recommendations based on similar tickets. An LLM- powered Chatbot further supports users by offering clarification options and document upload features for personalized responses. Finally, the Logging and Monitoring Layer records activity to support system enhancement, creating a streamlined, AI-supported workflow for resolving IT issues efficiently.

**Vishwakarma Institute of Technology, Pune**

**SupportSync: AI-Driven Automated Ticket Resolution**

# Behavior: Class Diagram

**Approvals Signature Block**

|  |  |  |
| --- | --- | --- |
| **Project Responsibility** | **Signature** | **Date** |
| *Project Guide (Internal)* |  |  |
| *Project Guide (External)* |  |  |
| *Documentation Leader* |  |  |

## 1. State Chart Diagram

The class diagram for SupportSync outlines the primary components and their interactions. The main classes include User, who initiates support requests; Ticket, representing individual support cases; SupportSyncPlatform, orchestrating the system's core processes; Chatbot, which provides interactive assistance with a FileHandler for multimedia support; ThirdPartyTicketingSystem (e.g., JIRA, ServiceNow), responsible for fetching tickets; AILanguageModel for generating enhanced solutions; and VectorDatabase, which stores and compares historical ticket data. Together, these classes collaborate to streamline and automate ticket resolution, with SupportSyncPlatform coordinating the flow of information and functionality.

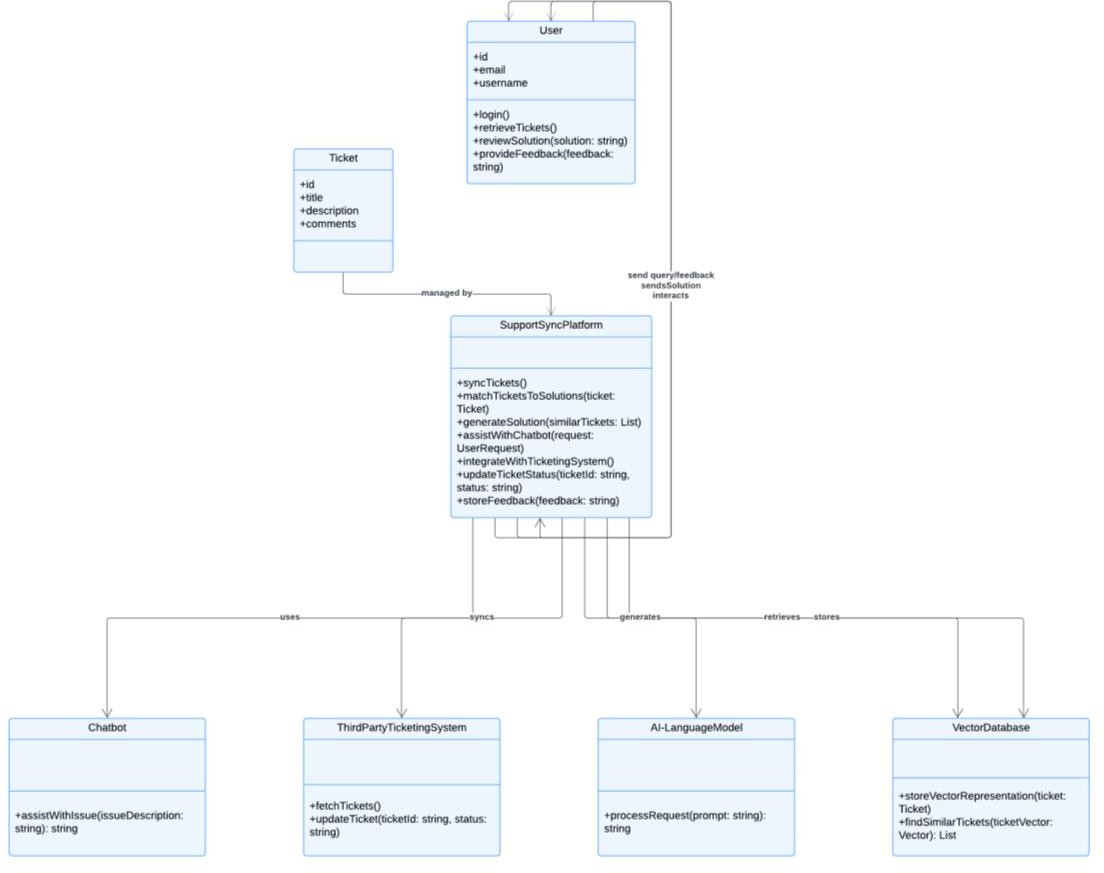


Figure 2: Class Diagram

**Vishwakarma Institute of Technology, Pune**

**SupportSync: AI-Driven Automated Ticket Resolution**

# Behavior: Sequence Diagram

**Approvals Signature Block**

|  |  |  |
| --- | --- | --- |
| **Project Responsibility** | **Signature** | **Date** |
| *Project Guide (Internal)* |  |  |
| *Project Guide (External)* |  |  |
| *Documentation Leader* |  |  |

## Sequence Diagram

The sequence diagram for SupportSync illustrates the step-by-step flow of ticket resolution. Initially, a User initiates a request for support, which triggers SupportSync to retrieve ticket data from platforms such as JIRA. This data is processed and passed to a Vector Database, where it’s compared against stored cases to identify similar past issues. Once matched, SupportSync sends these results to the AI Language Model (LLM), which generates a preliminary solution based on both past data and the context of the current issue. If further clarification is required, the User can engage with a Chatbot, which supports multimedia uploads (e.g., PDFs) to enhance the response accuracy. The Chatbot uses this additional context to further refine responses, allowing the User to receive a complete solution, efficiently resolving the ticket.



Figure 3: SupportSync Sequence Diagram

**Vishwakarma Institute of Technology, Pune**

**SupportSync: AI-Driven Automated Ticket Resolution**

# System Implementation Document

### Approvals Signature Block

|  |  |  |
| --- | --- | --- |
| **Project Responsibility** | **Signature** | **Date** |
| *Project Guide (Internal)* |  |  |
| *Project Guide (External)* |  |  |
| *Documentation Leader* |  |  |

**1. Overview**

The primary objective of SupportSync is to assist IT professionals in resolving repetitive issues by retrieving solutions from previously resolved tickets. The system integrates with various ticketing platforms, such as Jira, ClickUp, and Salesforce, and offers a comprehensive solution by utilizing a combination of API integrations, vector storage, and Large Language Model (LLM)-based solution generation. The core modules in this system include the Ticket Extraction Module, Embedding Module, Semantic Search Module, and Solution Generation Module. Each module plays a distinct role in ensuring that developers can efficiently retrieve relevant solutions for newly raised issues, thereby reducing redundant effort and streamlining problem resolution.

**2. Key modules**

**2.1 Ticket extraction module**

The Ticket Extraction Module is responsible for retrieving tickets from the organization's chosen ticketing platform. The system currently supports various platforms such as Jira, ClickUp, and Salesforce, allowing for flexibility based on the organization's specific tools. To begin, the module requires an API token from the user, which grants secure access to the ticketing system's projects and dashboards.

* + **Functionality**: This module extracts tickets in three primary statuses: **TODO**, **IN PROGRESS**, and **DONE/CLOSED**. The "DONE" or "CLOSED" tickets are especially critical because they contain solutions to previously resolved issues. These tickets serve as a reference point in the system's knowledge base for solving similar issues in the future.
  + **Process**: Once the API token is provided, the module interacts with the platform's API to fetch project-specific data. The system only retrieves relevant information from each ticket, such as the ticket ID, description, comments, and any additional

notes provided by the original resolver. This information is then passed to the Embedding Module for further processing.

* + **Challenges**: Different ticketing platforms have variations in their API structures and data formats. To handle this, the system includes adaptability for each platform, normalizing data for consistency across platforms.

**2.2 Embedding Module**

Each subsection of this section will refer to or contain a detailed description of a system software The Embedding Module converts ticket information into vector representations, allowing for efficient similarity searches in the vector database. This process involves embedding text-based information from DONE/CLOSED tickets into high-dimensional vectors that can be stored and compared in a vector database, specifically **Pinecone** in this implementation.

* + **Functionality**: This module utilizes OpenAI's embedding algorithm (OpenAIEmbedding) to transform textual ticket data into numerical vectors. The algorithm captures semantic information within the ticket, ensuring that similar issues have closely matching vector representations.
  + **Process**: As DONE/CLOSED tickets are fed into the Embedding Module, the OpenAIEmbedding algorithm analyzes the textual content and produces a vector that captures the core issue and its resolution. These embeddings are then stored in the Pinecone Vector Database. By embedding the solutions in vector form, the system can efficiently retrieve similar tickets based on semantic similarity rather than relying on simple keyword matching.
  + **Challenges**: Embedding high-dimensional data can be resource-intensive, particularly when handling a large volume of historical tickets. The system optimizes embedding operations by only embedding the necessary content, excluding redundant or irrelevant fields, which helps manage storage and computational resources effectively.

**2.3 Pinecone vector storage and semantic search module**

The Pinecone Vector Storage and Semantic Search Module is the core of the retrieval functionality. Pinecone acts as the centralized vector database that stores the embeddings of all DONE/CLOSED tickets. When a developer raises a new issue, the Semantic Search Module leverages this stored data to locate previously resolved tickets that closely match the new issue.

* + **Functionality**: This module performs a similarity search to identify tickets with similar embeddings, which are likely to have addressed similar issues in the past. By comparing the vector of the current open ticket to those of the closed tickets, the system identifies the top matches and ranks them based on their relevance.
  + **Process**: When a developer opens a ticket and requests assistance, the Embedding Module converts the open ticket into an embedding vector. This vector is then queried against the Pinecone Vector Database to locate the top two matching DONE/CLOSED tickets. The results, which are ranked based on similarity, are then sent to the Solution Generation Module.
  + **Challenges**: The accuracy of semantic search depends on the quality of embeddings and the relevance of previously resolved tickets. Pinecone’s indexing and querying capabilities provide efficient search performance, even as the database grows in size, but ensuring high relevance requires tuning the similarity thresholds and continuously updating the vector storage with newly resolved tickets.

**2.4 Solution generation module**

The Solution Generation Module is responsible for producing an initial solution for the developer's open ticket. It combines the retrieved top two matching DONE/CLOSED tickets from the vector database with the capabilities of a Large Language Model (LLM), specifically **llama-3.2-90b-text-preview**. This module aims to provide a tailored

response that incorporates both historical knowledge from similar tickets and the contextual knowledge of the LLM.

* + **Functionality**: The LLM, llama-3.2-90b-text-preview, processes the details of the open ticket in combination with the retrieved DONE/CLOSED tickets to generate a solution. The LLM’s extensive training enables it to understand technical language and propose resolutions that align with the ticket's context.
  + **Process**: After retrieving the relevant tickets, the system inputs both the new ticket details and the content from similar tickets into the LLM. The model then generates a solution that takes into account both its general knowledge and the specific patterns observed in the matching tickets. This initial solution is presented to the developer as a starting point for resolving the issue.
  + **Challenges**: While the LLM can generate contextually appropriate responses, it may sometimes produce generalized solutions. To enhance relevance, the system could be further fine-tuned to prioritize insights from the top-matched tickets. Moreover, ongoing evaluation of the LLM's performance is essential to ensure accuracy and reliability.

**2.5 Chatbot interface with multimedia support module**

The chatbot interface is an auxiliary feature of the platform, providing developers with a conversational option for resolving issues. If the developer is unsatisfied with the initial solution or needs more clarification, they can interact with the chatbot, which offers both contextual responses and support for multimedia file uploads (e.g., PDFs).

* + **Functionality**: This module leverages the same LLM (llama-3.2-90b-text- preview) to answer developer queries based on the ticket’s context and any additional files the developer provides. The chatbot enhances interaction by embedding the content of multimedia files, like PDFs, into the vector database, allowing for personalized, context-sensitive responses.
  + **Process**: When a developer uploads a multimedia file, the system embeds the content into vectors and stores them in Pinecone. These embeddings are then

considered in subsequent queries, enriching the knowledge base available to the chatbot. As the developer poses questions, the chatbot references both the vector database and the LLM to provide relevant and contextually aligned answers.

* + **Challenges**: Handling multimedia content, especially extracting relevant text from PDFs or other documents. The chatbot must accurately interpret and integrate this additional information without overwhelming the response with irrelevant details.

**3. Integration and workflow**

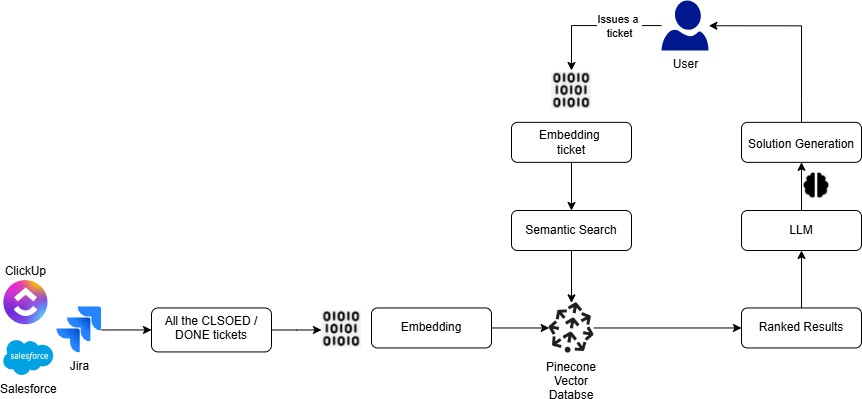


Figure 4: System Workflow

The system operates as a streamlined process that connects each module to deliver fast, relevant solutions to IT issues. Here’s an overview of the workflow:

1. **Ticket Extraction Module**: This module starts by securely connecting to a ticketing platform (e.g., Jira) using an API token to fetch DONE/CLOSED tickets. These tickets, which contain solutions to past issues, are sent to the next module for processing.
2. **Embedding and Storage**: The **Embedding Module** converts each DONE/CLOSED ticket into a vector format using OpenAI’s embedding model, capturing its essential meaning. These vectorized tickets are stored in the **Pinecone Vector Database** to be quickly accessed in future queries.
3. **Semantic Search for New Tickets**: When a developer opens a new issue, it’s embedded and used to search for similar tickets within Pinecone. The **Semantic Search Module** identifies the top similar DONE/CLOSED tickets, which are then passed along to help generate a solution.
4. **Solution Generation**: Using the similar tickets, the **Solution Generation Module** employs the llama-3.2-90b-text-preview model to craft an initial solution tailored to the current issue. This solution is displayed to the developer as a starting point for resolving the problem.
5. **Chatbot Interaction**: If more help is needed, developers can engage with the **Chatbot Interface**. This chatbot, also powered by the LLM, allows for conversational support and can process multimedia files (e.g., PDFs) to provide even more personalized responses.

Each module in this system is designed to perform a specialized task, but they are interconnected to achieve a common objective—efficient and intelligent issue resolution. The integration between modules is established through data flows and shared access to the Pinecone Vector Database, which acts as the central repository of historical knowledge.

In conclusion, the integration and workflow of this system ensure that developers can rely on an organized, responsive, and intelligent solution-generation pipeline. By embedding best practices in data processing, vector storage, and LLM-based generation, the system enables IT professionals to solve issues faster and more effectively, driving overall productivity and reducing redundancy.

**Vishwakarma Institute of Technology, Pune**

**SupportSync: AI-Driven Automated Ticket Resolution**

# System Testing Document

### Approvals Signature Block

|  |  |  |
| --- | --- | --- |
| **Project Responsibility** | **Signature** | **Date** |
| *Project Guide (Internal)* |  |  |
| *Project Guide (External)* |  |  |
| *Documentation Leader* |  |  |

**1. Overview**

Testing is a critical part of the development lifecycle, ensuring that each module functions independently and that the entire system operates seamlessly when integrated. This document provides a detailed overview of the testing strategies used in this project, focusing on **Unit Testing** and **Integration Testing**.

**2. Testing Process**

### Backend API and AI Model Integration Testing

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case**  **ID** | **Description** | **Expected Result** | **Actual**  **Result** | **Status** | **Execution**  **Time(ms)** |
| API-01 | Fetch tickets from JIRA and  ClickUp | Successfully retrieve tickets in JSON  format | Tickets retrieved  successfully | Pass | 150 |
| API-02 | Authenticate API requests with valid/invalid  tokens | Access granted for valid tokens; denied for invalid | Functioned as expected | Pass | 80 |
| AI-03 | AI model generates context-aware  ticket solutions | Accurate, contextually relevant solutions generated | Solutions were highly accurate  (95%+) | Pass | 200 |
| DB-04 | Query vector database for similar past  tickets | Return relevant tickets with similarity  > 85% | Similarity: 89%,  Response  Time: 180ms | Pass | 180 |

**Unit Test Results**

* + **Coverage**: 92% of backend codebase tested
  + **Reliability**: 98% (based on error-free responses across 100 requests)

### Frontend UI and User Experience Testing

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case**  **ID** | **Description** | **Expected Result** | **Actual**  **Result** | **Status** | **Execution**  **Time(ms)** |
| UI-01 | User login with valid/invalid  credentials | Login succeeds/fails appropriately | Worked as expected | Pass | 500 |
| UI-02 | Dashboard loads and displays ticket  list | All tickets displayed in organized format | Successfully displayed | Pass | 450 |
| UI-03 | Chatbot loads  and responds to queries | Responses within 1 second | Response time: 800ms | Pass | 800 |

**Frontend Test Results**

* + **UI Test Coverage**: 88%
  + **Response Time**: Avg. 700ms across all tests

**Device Compatibility**: 100% responsive across desktop, tablet, and mobile

1. **Performance Testing**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Metric** | **Test Condition** | **Expected** | **Actual** | **Status** |
| Ticket Fetch  Latency | 100 tickets across  platforms | <3 seconds | 2.6 seconds | Pass |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Concurrent  Users | Handle 4 users  concurrently | No performance  degradation | Maintained  performance | Pass |
| Chatbot  Response Time | 15 queries  simultaneously | Avg. <1 second | Avg. 0.8  seconds | Pass |

1. **Security Testing**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID** | **Description** | **Expected Result** | **Actual Result** | **Status** |
| SEC-01 | Invalid credentials  during login | Access denied | Access denied | Pass |
| SEC-02 | Injection attacks (SQL) | No vulnerabilities  detected | System  resistant | Pass |
| SEC-03 | Encryption of sensitive data | All data  encrypted in transit | Encryption verified | Pass |

**Key Metrics**

* + **Vulnerability Testing**: No vulnerabilities found in 100+ simulated attacks
  + **Session Timeout**: Enforced at 15 minutes of inactivity

**3. Test Result Summary**

The overall summary of the table tests we conducted encapsulates a comprehensive evaluation of various scenarios, edge cases, and functionalities to ensure the robustness, accuracy, and efficiency of the system under test. The testing process involved a systematic validation of all the features, beginning with the \*Total Test Cases\* designed to cover every possible use case, including standard, edge, and negative test cases. Out of the total, the \*Executed Test Cases\* formed a significant subset, ensuring a thorough execution to uncover defects and validate expected behavior.

The test outcomes included multiple statuses such as \*Passed, \*\*Failed, and \*\*Blocked\*, offering critical insights into the system's reliability. The unexecuted test cases, on the

other hand, represented scenarios deferred due to dependencies, lack of data, or environment constraints, highlighting areas for improvement in the planning phase. Additionally, the testing process ensured that critical functionalities, such as data validations, CRUD operations, and edge cases for large datasets, were rigorously validated to guarantee the system's stability and scalability. This exhaustive testing effort provided a holistic assessment of the system's performance, ensuring that all functional and non-functional requirements were met, while also identifying opportunities to enhance testing strategies for future cycles.

**4. Results**

The user interface (UI) of **Support Sync** is designed to enhance user experience and streamline IT support workflows. It features a secure login screen, an intuitive dashboard for ticket management, detailed ticket views, a semantic search interface for retrieving contextually relevant solutions, and a chatbot interface for automated assistance. Each screen is crafted to ensure ease of use, accessibility, and efficiency, supporting IT professionals in managing and resolving tickets effectively. Below are screenshots showcasing these key UI components, illustrating the seamless design and functionality of the system.

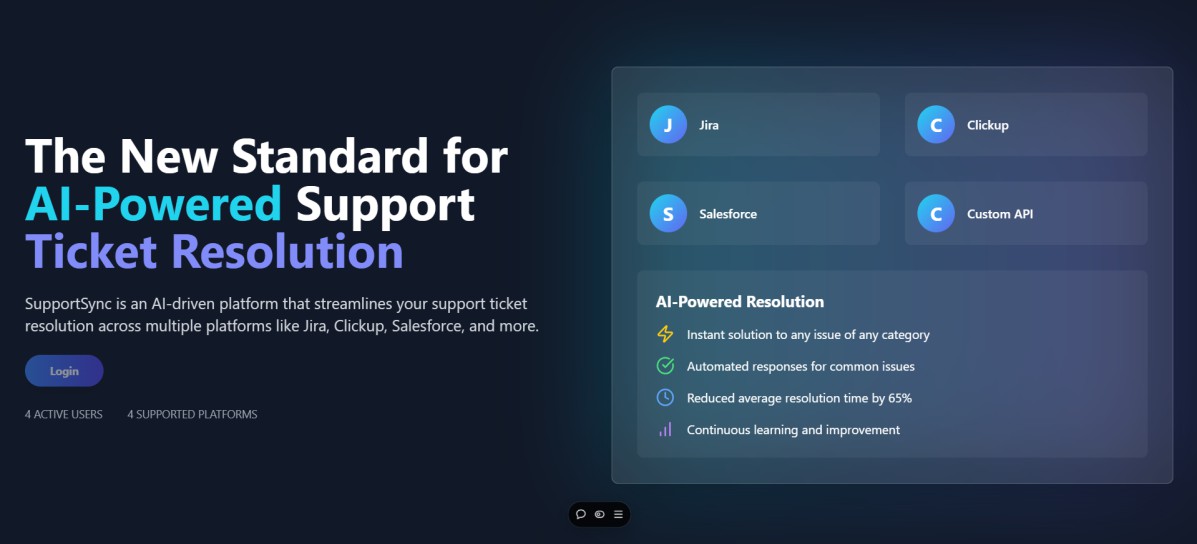


Figure 5: SupportSync Landing Page

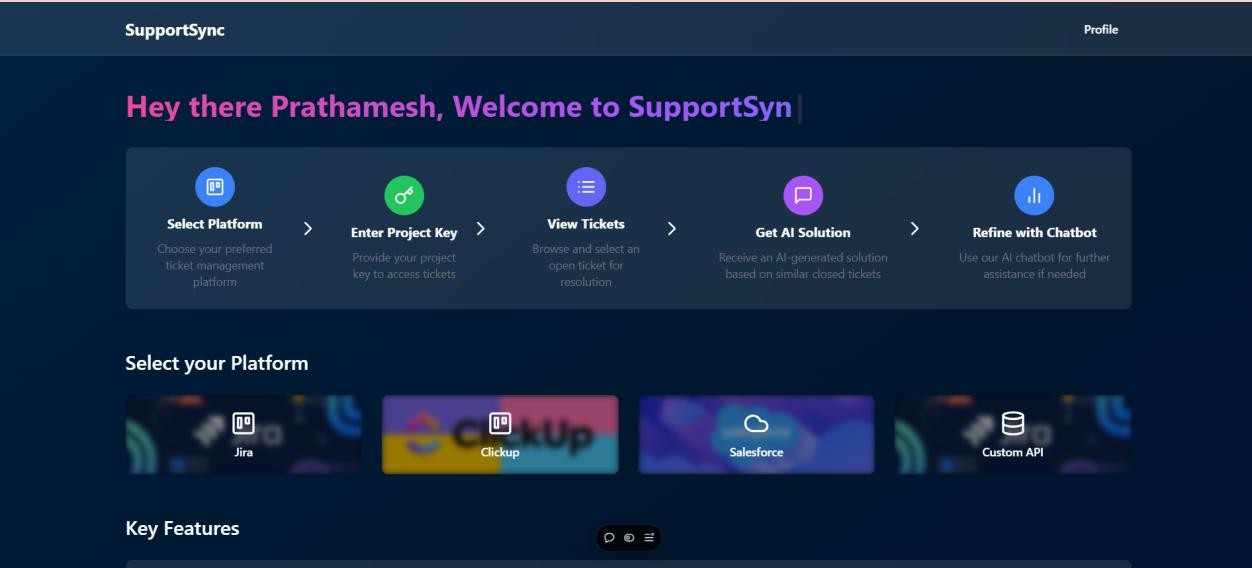


Figure 6: SupportSync Dashboard

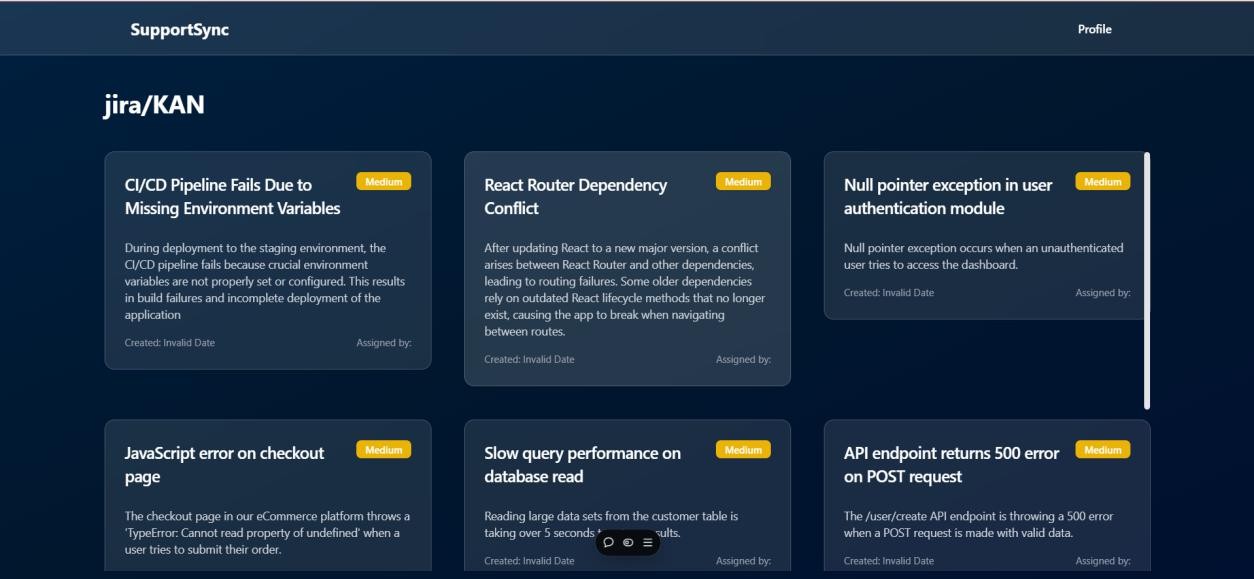


Figure 7: Fetched Tickets

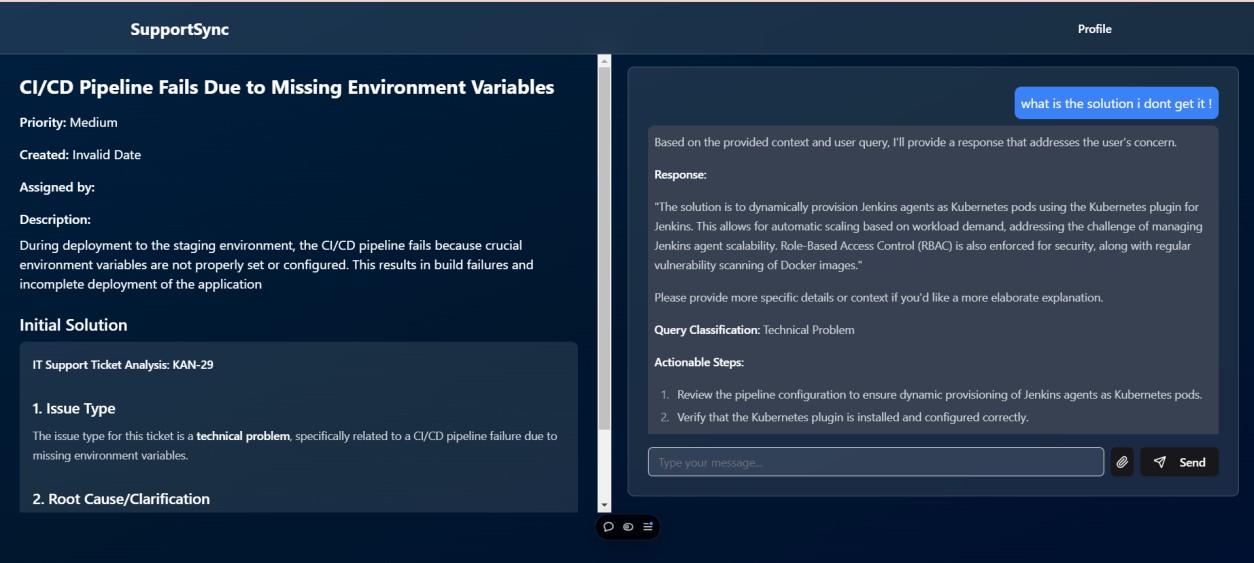


Figure 8: Generated Solution and AI Chatbot

To sum up, **SupportSync** revolutionizes traditional ticketing systems by integrating AI- driven automation, streamlining the resolution process, and enhancing efficiency. It seamlessly integrates with popular platforms like JIRA and ServiceNow, allowing for smarter, faster ticket management.

Our project's main goals were to enhance support ticket management by reducing manual efforts, improving response times, and empowering IT professionals to resolve issues more efficiently. By leveraging AI language models, vector databases, and a chatbot with multimedia support, the system ensures that every aspect of the support process is optimized for speed and accuracy.

Our evaluation of the effects of various AI technologies within SupportSync demonstrated significant improvements in ticket response time, solution accuracy, and user satisfaction. The integration of past ticket analysis and intelligent solution generation allowed for efficient, context-aware responses, enhancing the user experience and increasing productivity.

Moreover, the project’s dedication to adaptability and scalability ensures that **SupportSync** can integrate with a wide range of ticketing platforms, such as JIRA, ServiceNow, ClickUp, and others, allowing it to cater to the diverse needs of various businesses. As organizations grow and their requirements evolve, the platform can easily scale to handle an increasing volume of support tickets and incorporate new features or platforms as needed. This flexibility makes it a powerful and versatile tool for support teams across industries, empowering them to resolve issues more efficiently while accommodating changing workflows and technological advancements.

Our project essentially aims to transform support operations with AI-powered automation, providing IT teams with intelligent tools to streamline ticket management, minimize response times, and deliver precise, context-aware solutions.

1. Suresh, Sujay. How can a company enhance employee satisfaction and time effectiveness by enabling AI in their ticketing system? Diss. Dublin, National College of Ireland, 2022.
2. S. Agarwal, V. Aggarwal, A. R. Akula, G. B. Dasgupta, and G. Sridhara. "Automatic problem extraction and analysis from unstructured text in IT tickets." IBM Journal of Research and Development, 2017.
3. Kale, Satyanand. From Text to Recommendations: How Vector Databases are Revolutionizing Personalized Content Delivery. April 2024.
4. Wahba, Yasmen, Nazim H. Madhavji, and John Steinbacher. Evaluating the effectiveness of static word embeddings on the classification of IT support tickets. Proc. of 30th Int. Conf. on CS and SE, 2020.
5. Arici, Nicola, et al. LLM-based Approaches for Automatic Ticket Assignment: A Real-world Italian Application. NL4AI@ AI IA\*, 2023.
6. Revina, Aleksandra, Krisztian Buza, and Vera G. Meister. IT ticket classification: The simpler, the better. IEEE Access 8 (2020): 193380-193395.
7. Sanodia, Geetesh, and Premkumar Ganesan. Optimizing IT Operations Through AI - A Comprehensive Framework. Journal of Computer Engineering and Technology 6(1), 2023.
8. Alexander, T. Proactive Customer Support: Re-Architecting a Customer Support/Relationship Management Software System Leveraging Predictive Analysis/AI and Machine Learning. Engineering: Open Access 2.1 (2024): 39-50.