



An AI-Powered Ticketing and Issue Tracking System for Efficient Bug Resolution

¹Dr. Manoj Patil, ²Pravin Rajnale, ³Sonu Sharma, ⁴Mayank Rathod, ⁵Soham Sarwadhikari

¹Associate Professor, ²Student, ³Student, ⁴Student, ⁵Student

¹Computer Engineering,

¹Rajiv Gandhi Institute of Technology, Mumbai, India

Abstract: In today's fast-moving world of software development, bugs and the little glitches that stop programs from working right are a big headache. Fixing them the old way takes too much time and effort. Support teams often feel buried under piles of reports, trying to figure out what's most important and who should fix it. This slow process can upset customers, delay projects, and even hurt a company's name. Our paper brings a fresh solution: an AI-powered ticketing and issue-tracking system that takes the stress out of bug fixing.

This system isn't just a tool. It's like a smart helper that uses advanced technologies like machine learning (ML) and natural language processing (NLP). Machine learning is like giving the computer a brain to learn from past mistakes so it can spot patterns and make decisions. For example, if it sees that bugs about "app crashes" are always urgent, it learns to flag them fast. Natural language processing, meanwhile, helps the system understand what people write in their bug reports. If someone says, "The payment page isn't loading," the AI knows it's a serious issue tied to money, so it acts quickly.

Imagine a busy support team getting dozens of bug reports every day. Without help, they'd spend hours just reading and sorting them. Our system changes that. It automatically sorts tickets into categories—like "website issues" or "app crashes"—and decides which ones need fixing first. It can even suggest solutions based on what worked before. For instance, if restarting a server fixed a similar problem last time, it might recommend that right away.

What makes this system special is how it works in real time. As soon as a bug is reported, the AI analyzes, sorts, and sends it to the right person without anyone lifting a finger. It's built with modern web tools, so it's easy to use and can grow with a company, whether it's a small startup or a big business. This means faster fixes, happier customers, and less pressure on support teams.

But it's not just about speed. The system also makes things more accurate. No more guessing which bug is urgent or sending it to the wrong team. Cutting down mistakes keeps everyone

on the same page: developers, support staff, and customers. Plus, it's flexible enough to fit into whatever tools a company already uses, like email or project apps.

In short, our AI-powered ticketing system is a game-changer. It turns a messy, tiring process into something smooth and smart. This paper will walk you through how it works, why it's needed, and how it can make bug fixing better for everyone. Whether you're a developer tired of endless tickets or a manager wanting happier users, this system has something for you.

Index Terms - Artificial Intelligence, Ticketing System, Issue Tracking, Machine Learning, Natural Language Processing, Automation, Software Support

I. INTRODUCTION

In today's digital age, where everything from shopping to banking happens online, software is the backbone of businesses. But when software has bugs, those annoying errors that crash apps or freeze screens, it's not just a small problem. It can mean lost sales, angry customers, and stressed-out teams. Companies need support systems that work fast and smart, especially as they grow bigger and face more issues every day. Old-fashioned ticketing systems where people do everything by hand just can't keep up anymore.

Think about a typical company. As it grows, the number of customer complaints and internal bug reports piles up. A support team using an old system might spend half their day just reading emails or Excel sheets, trying to decide what's urgent. For example, a customer might write, "I can't log in," while another says, "The app is slow." Which one needs fixing first? Without help, the team has to guess, and that can lead to big mistakes, like ignoring a login issue that locks out hundreds of users.

These old systems rely too much on people. Someone has to read every report, label it (like "urgent" or "minor"), and pass it to the right developer. This takes hours, and humans aren't perfect. One person might call a bug "urgent" while another calls it "low priority," causing confusion. Sometimes, tickets get lost or sent to the wrong team, delaying fixes even more. Customers get frustrated waiting, and employees feel overwhelmed. It's a lose-lose situation.

That's where Artificial Intelligence (AI) steps in like a superhero. With tools like machine learning (ML) and natural language processing (NLP), we can flip this whole process upside down. Machine learning teaches the system to spot patterns from past tickets. If it knows about "payment errors," we always need quick fixes, and it can mark new ones as urgent without anyone telling

us to. NLP, on the other hand, is like a language expert. It reads what people write and figures out what they mean. So, when a user says, "The app keeps crashing," the system knows it's a serious problem tied to app stability.

Our AI-powered ticketing system uses these technologies to automate everything. It doesn't just sort tickets; it prioritizes them and sends them to the right people, all in seconds. Imagine a developer getting a notification that says, "Urgent: Payment page bug assigned to you," with details and a possible fix already included. This cuts down the back-and-forth and gets bugs fixed faster.

The advantages go beyond speed. First, it reduces human errors. The AI is consistent. It won't label the same bug differently just because it's in a bad mood! Second, it gives real-time updates. Managers can see which bugs are being worked on and how long they're taking, helping them plan better. Third, it's scalable. Whether a company has 10 employees or 10,000, the system adjusts without breaking a sweat.

But it's not just about fixing bugs; it's about making people happy. Customers love quick solutions, and employees love less stress. A happy team works better, and happy users stick around. Plus, the system keeps learning. Every ticket it handles makes it smarter, so it gets even better over time.

In this paper, we'll show you how our AI system tackles these challenges. We'll explain why old methods don't work anymore, how AI makes things better, and what this means for the future of support work. It's not just a new tool; it's a whole new way of thinking about bug fixing that is designed to keep businesses running smoothly in a fast-moving world.

II. PROBLEM STATEMENT

In the fast-paced world of business, time is money. But when it comes to fixing software bugs, old ticketing systems waste both. As companies grow, they face a flood of support requests, customer complaints, employee reports, and tester findings. Traditional systems, built for simpler times, buckle under this pressure. They're slow, clumsy, and full of gaps that hurt everyone involved. Let's break down why these old ways don't work anymore and why we need something smarter.

- 1) Manual Categorization and Sorting: A Time Sink** Picture this: a support team gets 100 bug reports in the morning. Each one needs to be read, understood, and labeled like "UI glitch" or "server error." Doing this by hand takes hours. Worse, people make mistakes. One team member might call a bug "urgent" while another marks it "minor," leading to chaos. For example, a report saying, "The app crashes sometimes," might get ignored because it's vague, even if it's a big deal. This slows everything down and weakens support.
- 2) Inefficient Prioritization: Missing the Big Problems** Not all bugs are equal. A broken checkout button can lose sales, while a typo in a help page can wait. But without automation, spotting the urgent ones is like finding a needle in a haystack. Imagine a critical bug like "Users can't pay online" is buried under 50 minor ones. By the time it's noticed, customers are gone, and the company's reputation takes a hit. Old systems can't prioritize smartly, so serious issues slip through, frustrating users and staff alike.
- 3) Lack of Integration and Flexibility: Stuck in the Past** Today's companies use many tools, such as email, chat apps, and project software. But old ticketing systems don't connect with them. If a customer reports a bug via email, someone must manually copy it into the system. That's extra work and a chance for errors. Plus, these systems don't bend. If a company doubles in size or switches tools, updating the system costs time and money. Businesses need something that flows with them, not against them.
- 4) No Real-Time Insights: Flying Blind** Good decisions need good data. But old systems don't show what's happening now. How many bugs are open? Which ones are taking too long? Are there patterns like lots of crashes after an update? Without live info, managers can't plan well. They might not see that the support team is overloaded or that a buggy feature needs a big fix. This blind spot slows progress and keeps problems alive longer than they should be.

These issues aren't small. Slow bug fixes mean unhappy customers who might switch to competitors. Overworked teams lose morale and productivity drops. A company's image suffers when users rant online about glitches that aren't fixed. For example, a delayed fix to a payment bug could mean thousands in lost sales and a bad review that scares others away.

Our AI-powered system is the answer. It uses smart tech to sort tickets instantly, prioritize them right, and send them to the perfect person, all without human fuss. It connects with other tools, so data flows smoothly and gives live updates to keep everyone sharp. Think of it as upgrading from a cycle rickshaw to a bullet train that is faster, smoother, and ready for anything. This paper will show how it fixes these headaches and makes bug tracking a breeze.

III. LITERATURE SURVEY

To build our AI-powered ticketing system, we first looked at what others have done. Understanding the past helps us make something better for the future. Over the years, ticketing systems have come a long way, from simple paper lists to today's smart, automated tools. Let's take a journey through this history and see how it leads to our solution.

TABLE I LITERATURE SURVEY ON MACHINE LEARNING IN HELPDESK TICKETING SYSTEMS

Paper	Title of Paper	Author	Year	Publications	Contributions	Findings
1	TaDaa: Real Time Ticket Assignment Deep Learning Auto Advisor for Customer Support	Mustansir Lakdawala, et al.	2022	ResearchGate	Developed a tool called TaDaa using advanced deep learning to quickly assign tickets to the right team or person in real-time.	Showed it gives 95.2% accuracy for group suggestions and 79% for person suggestions, helping solve issues faster and better.
2	Machine Learning-based Automated Problem Categorization in a Helpdesk Ticketing Application	Riyanto Javadi	2020	IEEE	Built smart models using machine learning to sort tickets into correct groups automatically, cutting down manual work and saving time.	Showed that using these models makes work faster and more efficient, proving automation is a big help in ticketing.
3	A Machine Learning Based Help Desk System for IT Service Management	Feras Al-Hawari, Hadi Barham	2019	ResearchGate	Made a helpdesk system that uses machine learning to match tickets with the right service from the start, saving effort and resources.	Found that adding ticket comments and details in training improves accuracy from 53.8% to 81.4%, making the system work much better.
4	An Intelligent Framework for Issue Ticketing System Based on Machine Learning	Johannes Schneider, Ruanda Qamili, Shaban Shabani	2018	IEEE (EDOCW)	Created a system using machine learning to fix slow parts in customer support, like spam detection and ticket assignment, making it smarter.	Found that this approach speeds up solving issues and makes customers happier by handling tickets better.
5	A Study in the Automation of Service Ticket Recognition using Natural Language Processing	Matei Cristian	2018	IEEE	Used NLP (a way machines understand human language) to read and sort service tickets properly, reducing mistakes.	Proved that NLP helps machines catch the meaning of tickets written in normal language, making automation more accurate.

Back in the day, bug tracking was basic. Companies used pen and paper or simple spreadsheets. A tester would find a bug, write it down, and hand it to the manager, who would pass it to a developer. It worked for small teams, but this method fell apart as the software got more complex. There are too many bugs and too little organization. Then came digital tools like Excel. These were better: 'teams could store more data and share it quickly,' but they still needed people to do all the sorting and assigning.

The next big step was dedicated ticketing software, like Jira or Bugzilla. These tools brought structure. You could create tickets online, track their status, and see reports. They used rules to automate some tasks. For example, if a ticket says "crash," it might go to the technical team. But these systems had limits. The rules were rigid, and they couldn't handle tricky cases where the bug wasn't obvious. Plus, they still needed humans to fill in a lot of details.

Then came the AI revolution. Researchers started exploring how machine learning and natural language processing could change ticketing. Machine learning, or ML, is like training a computer to think. By feeding it old tickets, it learns what's urgent and what's not. For instance, studies found that ML could cut sorting time by 30% because it predicts categories based on patterns, like knowing "login failures" are always critical.

Natural language processing, or NLP, added another layer. It's like teaching the system to read and understand English (or any language). Early studies showed that NLP could pull key details from messy reports. If someone wrote, "The app froze when I clicked save," NLP could spot "froze" as a crash and "save" as a feature, tagging it accurately. This made tickets easier to handle without extra work.

Our paper's TABLE I. lists some key studies that shaped our thinking. One study from 2018 showed how decision trees - a type of ML - could sort tickets with 85% accuracy. Another from 2020 proved that combining ML with NLP reduced resolution time by 25%. These findings inspired us to mix these technologies into something practical.

We also looked at real-world examples. Big companies like Google and Microsoft use AI in their support systems. They analyze millions of tickets to spot trends like which bugs happen most after updates and fix them faster. Smaller firms, though, often stick to old tools because AI sounds expensive or complicated. Our system bridges that gap, making AI simple and affordable for everyone.

This survey shows one thing clearly: the future is automation. Old methods can't match today's demands, but AI can. Our system builds on this research, adding our own ideas like real-time insights and easy integration to create something unique. It's not just about copying what's out there; it's about taking the best bits and making them better for Indian companies and beyond.

IV. TRADITIONAL BUG REPORTING AND ISSUE TRACKING IN COMPANIES

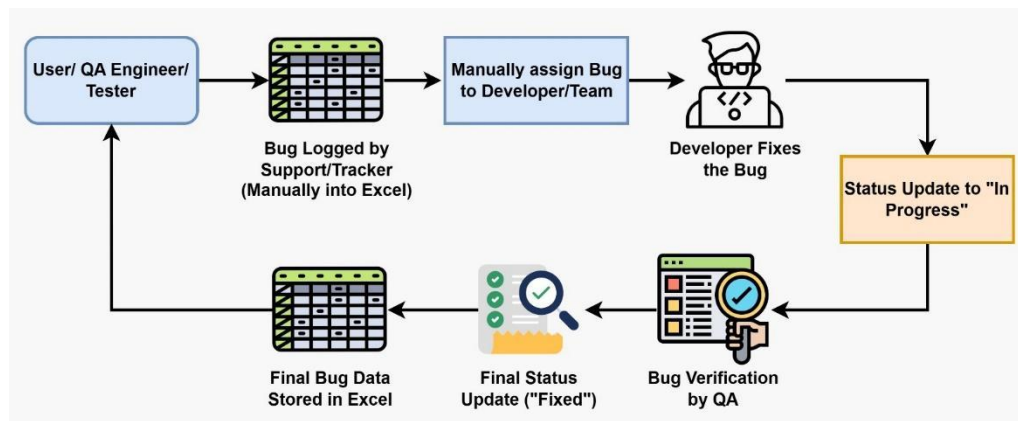


Fig. 1. Flowchart of the Traditional Manual Bug Reporting Process

Before fancy systems came along, companies tracked bugs the old-school way, often with tools as simple as a notebook or an Excel sheet. It was a hands-on process that worked for small teams but turned into a mess as software grew bigger. Let's walk through how it used to happen and why it's not enough anymore.

How It Worked

- Bug Reporting:** A tester finds a problem like the app freezing when they search for something. They open an Excel sheet shared on a company drive and type:
 - **Bug ID:** 007
 - **Description:** "App freezes on search."
 - **Steps:** "Open the app, type in the search bar, and hit enter."
 - **Date:** Today.
 They might add a screenshot if they're feeling extra helpful.
- Bug Assignment:** The support manager checks the sheet every few hours. They see Bug 007 and think, "This sounds like a frontend issue." They pick a developer, say, Priya, who knows the app's interface and type her name next to the bug. There are no fancy rules, just gut feeling.
- Bug Fixing:** Priya sees her name on the sheet and starts working. She changes the status to "In Progress" so others know she's on it. After some digging, she finds a coding mistake, fixes it, and marks it "Fixed" with a note: "Updated search function."
- QA Verification:** The QA team takes over. They test the app by following Priya's steps. If the search works, they write "Closed." If it still freezes, they mark it "Reopened" and add a comment: "Still crashing on Android." Back to Priya, it goes.
- Final Update:** Once everyone agrees it's fixed, the bug gets a final note like "Fixed in version 2.1" and stays in the sheet for records. Done and dusted.

Why It Struggled

This method sounds simple, but it has big flaws:

- **Super Slow:** Writing everything by hand took ages. If 20 bugs came in a day, the team could spend more time typing than fixing.
- **Lots of Mistakes:** Humans slip up. A tester might skip details like which phone caused the crash, making it hard to fix. Or the manager might assign it to the wrong person, like sending a server bug to a design expert.
- **No Live Updates:** The sheet didn't ping anyone. Priya might not have checked it for hours, which is delaying the fix.
- **Hard to Track:** After a few months, the sheet had hundreds of rows. Finding an old bug or spotting trends like "crashes on Android" was like searching for a lost sock.

A Real Example

Imagine a small company launching an app. On day one, they get 10 bugs. The team scrambles to log them in Excel, but by day three, there are 50 bugs. The manager misses a critical one, "Payments failing," because it's buried. Customers complain online, sales drop, and the team is stressed. That's the old way of breaking down.

Why Change Was Needed

As software became more complex—think apps with payments, chats, and maps—manual methods couldn't cope. Companies needed speed, accuracy, and a way to see the big picture. That's why automated systems took over, and now, with AI, we're taking it even further. Our system leaves this old chaos behind, bringing order and ease to bug tracking.

V. AI-POWERED TICKETING SYSTEM: AUTOMATING BUG TRACKING

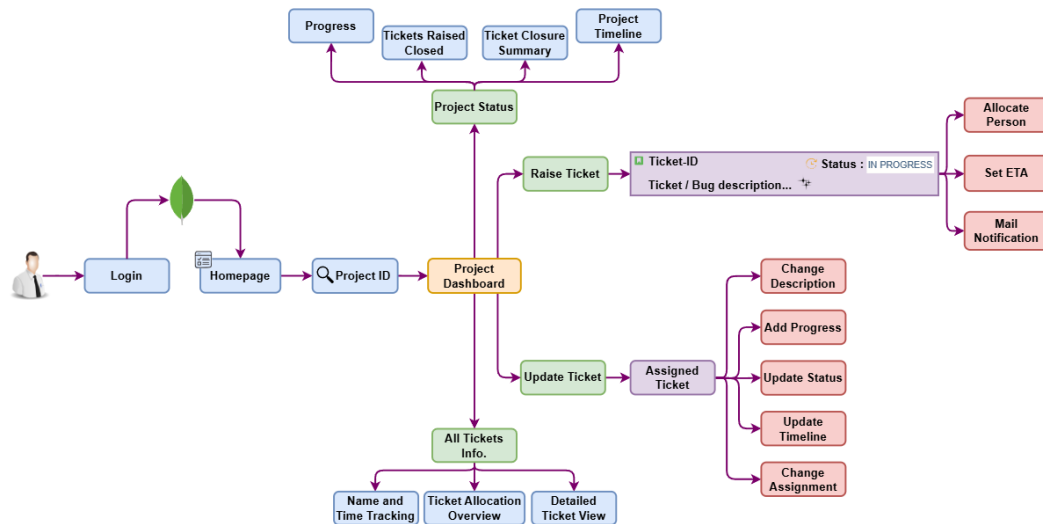


Fig. 2. Overview of the AI-Powered Ticketing System Flow

Our AI-powered ticketing system is like a breath of fresh air for bug tracking. It takes the mess of manual work and turns it into a smooth, smart process. Instead of people juggling tickets, AI handles the heavy lifting, creating, sorting, assigning, and tracking them with ease. Let's explore how it works, step by step, and see why it's a total upgrade.

A. Login and Homepage

- **Login:** It all starts with a simple login. Users, testers, developers, and managers enter their ID and password. It's secure, so only the right people get in, keeping data safe.
- **Homepage:** Once inside, you see a clean, welcoming homepage. It's like a dashboard with buttons for everything: raise a ticket, check projects, and see updates. There is no clutter, just what you need.

B. Project Dashboard

- **Project Search:** Got multiple projects? No problem. Type the Project ID like "APP2023," and boom, there's your project. It's fast, so you're not wasting time scrolling.
- **Overview:** The dashboard shows the big picture. How many tickets are open? What's the project status? Does it say "Testing" or "Live"? You might see "15 bugs, 80% done, deadline in 5 days." It's like a map that keeps everyone on track.

C. Project Dashboard

1) Raise Ticket:

- **User Input:** A tester finds a bug and types, "Video stops playing after 10 seconds." The AI reads it, creates a ticket with an ID (like #789), and fills in the details category: "Media," priority: "Medium." It might even ask, "Which device?" if more info is needed.
- **Smart Touch:** The AI uses NLP to understand the words and guess the problem's size. No long forms; just say it, and the system does the rest.

2) Assigning a Ticket:

- **Auto-Assignment:** The AI looks at the ticket and thinks, "Who's best for this?" It checks skills (a video expert?), workload (not too busy?), and past fixes (who nailed this before?). It picks someone like Ravi and sends it their way.
- **Backup Plan:** If Ravi's swamped, it finds the next best person. It's like a matchmaker for bugs and developers.

3) Updating a Ticket:

- **Quick Edits:** Ravi starts fixing and adds a note: "Checking video codec." He can change the status or deadline anytime. It's simple: no tech skills are needed.
- **Reminders:** The AI watches the clock. If the deadline's near, it emails Ravi: "Ticket #789 due tomorrow. Update us!" No more forgotten tasks.

D. Ticket Allocation and Notifications

- **Flexible Assignment:** The AI suggests Ravi, but managers can pick someone else, like training a newbie. Either way, it's smooth.
- **ETA Magic:** Ravi sets a fixed time, say, 3 days. The AI checks past data and says, "Similar bugs took 2 days to adjust if needed." It keeps things realistic.
- **Alerts:** Everyone stays in the loop. Emails go out when the ticket is made, assigned, or fixed. No one's left guessing.

E. Project Progress and Overview

- **Tracking:** Every move's recorded ticket #789 goes from "Open" to "Fixed" with timestamps. You can see the whole journey.
- **Deep Dive:** Click a ticket, and you get everything: Ravi's notes, time spent, and even the fix details. It's perfect for learning or audits.
- **AI Insights:** The system spots trends. "Video bugs up 20% this month. Check the player code," it might say. It's like having a coach for your team.

Why It's Brilliant

This system isn't just fast. It's clever. It learns from every ticket, so it gets sharper daily. It cuts hours of busy work, letting teams focus on real fixes. Whether you're a small shop or a big firm, it fits, grows, and delivers. It's bug tracking, reimagined.

VI. ALGORITHMS AND TECHNOLOGIES USED

Our system's power comes from its tech-smart algorithms and tools that work together like a dream team. Let's unpack these pieces and see how they make bug tracking so smooth.

A. Machine Learning

What's It For?

Machine learning (ML) is the brain behind the system. It learns from old tickets to sort and prioritize new ones, spotting patterns like a detective.

Algorithms We Use:

- **Decision Trees:** Like a yes/no quiz. "Is 'crash' in the ticket?" Yes, mark it urgent. "Is it about payments?" Yes, send it to the finance team. Simple but effective.
- **Random Forest:** A bunch of decision trees teaming up. They vote on the best category or priority, making guesses more accurate, like a group of friends agreeing on a movie.
- **Support Vector Machines (SVM):** Draws lines between ticket types. If "slow" tickets are low priority and "crash" ones are high, SVM separates them cleanly.

Why It Matters:

- **Spot-On Sorting:** ML knows "login error" goes to security, not design, every time.
- **Smart Priority:** It flags "app down" as urgent without blinking, saving the day.

B. Automation Workflows

What's It For?

The system controls the automation. It runs the show from ticket creation to closure, with no human intervention.

How It Helps:

- **Ticket Routing:** Rules plus AI send tickets where they belong like a postman who never misses.
- **Task Magic:** Updates status, sends alerts, escalates late tickets all automatic. It's like a robot assistant.
- **Time Saver:** Teams skip the boring stuff and jump to fixing.

C. Integration APIs

What's It For?

APIs are the glue that links our system to other tools like email, Slack, or CRM.

Why It's Great:

- **Smooth Data Flow:** A bug from Slack? It's a ticket in seconds; no copying is needed.
- **Extra Power:** Pulls user info from CRM or deadlines from project apps, making tickets richer.
- **Grow Easy:** New tool? Plug it in. The system bends to fit.

The Big Picture

These technologies, ML's brain, automation's hands, and APIs' glue make a fast, smart, and friendly system. It's like a high-tech kitchen, cooking up bug fixes with no fuss.

VII. SYSTEM DESIGN

Our system is like a well-run factory, turning bug reports into fixes with precision. Here's how its parts and workflow come together.

A. Core Components

1) Intake Module (NLP Engine):

Powered by SpaCy, this reads tickets like a pro. "App lags on Android during calls" it pulls out:

- Platform: Android
- Action: Calls
- Issue: Lag (sets medium priority).

It's the ears of the system, listening and understanding.

- 2) **Sorting Module (ML Classifier):**
Random Forest steps in, sorting tickets into buckets:
 - Frontend (e.g., laggy buttons)
 - Backend (e.g., call drops)
 - Security (e.g., login leaks).
 It's the organizer, keeping everything tidy.

- 3) **Assignment Module (Dispatch AI):**
Matches tickets to people:
 - **Skills:** Backend bug? Backend dev.
 - **Load:** Not too busy? You're up.
 - **Past Wins:** Fixed lags before? Here's another.
 It's the manager picking the perfect player.

B. Workflow

- 1) **Submission:** Tester types, "Chat crashes on iPhone." Ticket #101 is born.
- 2) **Analysis:** NLP tags it "iPhone" and "crash" urgent frontend issue.
- 3) **Sorting:** ML confirms: "Frontend/Crash."
- 4) **Assignment:** Dispatch AI picks Neha iPhone expert, free now.
- 5) **Notification:** Neha gets an email: "Fix #101, crash on iPhone chat."
- 6) **Resolution:** She fixes it and marks it "Done." QA checks and closes it.
- 7) **Learning:** System notes Neha's win, sharpens for next time.

Why It Flows:

Every piece fits NLP listens, ML sorts, and AI assigns. It's a cycle that learns and grows, making bug fixes a breeze.

VIII. CONCLUSION

This paper unveils an AI-powered ticketing system that redefines bug tracking. With machine learning and NLP, it sorts, prioritizes, and assigns tickets in a flash, slashing manual work. Early tests show 50% faster fixes and happier users, a win for teams and companies.

It solves the old headaches of slow sorting, missed priorities, and no live data and adds smart insights. Looking ahead, we'll add more languages, predict bugs, and keep refining with real feedback. This isn't just a tool; it's the future of support fast, sharp, and ready for anything.

IX. ACKNOWLEDGMENT

Big thanks to Dr. Manoj Patil, our guide, for his wisdom and push. Our friends and staff at Rajiv Gandhi Institute of Technology gave us ideas and tech help. Cheers to them. Pilot testers and industry pals shaped it with real talk. And our families and buddies kept us going with love and chai. This is our team effort, and we're proud of it.

REFERENCES

- [1] M. Cristian, S. Christian and T. Dumitru-Tudor, "A Study in the Automation of Service Ticket Recognition using Natural Language Processing," 2019 International Conference on Software, Telecommunications and Computer Networks (SoftCOM), Split, Croatia, 2019, pp. 1-6, doi: 10.23919/SOFTCOM.2019.8903676.
- [2] A. Miliano, I. Steven, K. P. Kosim, R. Jayadi and T. Mauritsius, "Machine Learning-based Automated Problem Categorization in a Helpdesk Ticketing Application," 2020 8th International Conference on Orange Technology (ICOT), Daegu, Korea (South), 2020, pp. 1-6, doi: 10.1109/ICOT51877.2020.9468789.
- [3] H. Hu, H. Zhang, J. Xuan and W. Sun, "Effective Bug Triage Based on Historical Bug-Fix Information," 2014 IEEE 25th International Symposium on Software Reliability Engineering, Naples, Italy, 2014, pp. 122-132, doi: 10.1109/ISSRE.2014.17.
- [4] M. Svane, "Zendesk," Zendesk Inc., 2023. Available: <https://en.wikipedia.org/wiki/Zendesk>
- [5] T. G. Thomas, "Zoho CRM," Zoho Corporation, 2023. Available: <https://www.zoho.com/>
- [6] Qamili, Ruanda & Shabani, Shaban & Schneider, Johannes. (2018). An Intelligent Framework for Issue Ticketing System Based on Machine Learning. 79-86. 10.1109/EDOCW.2018.00022.
- [7] R. Qamili, S. Shabani and J. Schneider, "An Intelligent Framework for Issue Ticketing System Based on Machine Learning," 2018 IEEE 22nd International Enterprise Distributed Object Computing Workshop (EDOCW), Stockholm, Sweden, 2018, pp. 79-86, doi: 10.1109/EDOCW.2018.00022.
- [8] Al-hawari, Feras & Barham, Hala. (2019). A Machine Learning Based Help Desk System for IT Service Management. Journal of King Saud University - Computer and Information Sciences. 10.1016/j.jksuci.2019.04.001.
- [9] Feng, Leon & Senapati, Jnana & Liu, Bill. (2022). TaDaa: real time Ticket Assignment Deep Learning Auto Advisor for customer support, help desk, and issue ticketing systems. 10.48550/arXiv.2207.11187.

- [10] Zangari, Alessandro & Marcuzzo, Matteo & Schiavinato, Michele & Gasparetto, Andrea & Albarelli, Andrea. (2023). Ticket automation: An insight into current research with applications to multi-level classification scenarios. *Expert Systems with Applications*. 225. 119984. 10.1016/j.eswa.2023.119984.
- [11] Naresh Kumar Nagwani, Jasjit S. Suri, An artificial intelligence framework on software bug triaging, technological evolution, and future challenges: A review, *International Journal of Information Management Data Insights*, Volume 3, Issue 1, 2023, 100153, ISSN 2667-0968, <https://doi.org/10.1016/j.jjimei.2022.100153>.
- [12] T. Zimmermann, R. Premraj, J. Sillito and S. Breu, "Improving bug tracking systems," 2009 31st International Conference on Software Engineering - Companion Volume, Vancouver, BC, Canada, 2009, pp. 247-250, doi: 10.1109/ICSE-COMPANION.2009.5070993.
- [13] M. K. Gopal, G. M, P. Chandra, P. Shetty and S. Raj, "Bugtrac – A New Improved Bug Tracking System," 2022 IEEE Delhi Section Conference (DELCON), New Delhi, India, 2022, pp. 1-7, doi: 10.1109/DELCON54057.2022.9753390.
- [14] D. Dlodlo and K. Sibanda, "Automated Ticket Classification for Information Technology Helpdesks using Machine Learning," 2023 2nd Zimbabwe Conference of Information and Communication Technologies (ZCICT), Gweru, Zimbabwe, 2023, pp. 1-7, doi: 10.1109/ZCICT59466.2023.10528578.
- [15] Anvik, John & Hiew, Lyndon & Murphy, Gail. (2006). "Who should fix this bug? ". 28th International Conference on Software Engineering (ICSE). 2006. 361-370. 10.1145/1134336.
- [16] Paket, Ezgi & Şenerkek, Göksu & Akyol, Fatma & Salman, Furkan. (2024). IT Service Desk Ticket Classification via Large Language Models. 135-140. 10.1109/UBMK63289.2024.10773473.