TEJAS Journal of Technologies and Humanitarian Science
ISSN-2583-5599
Vol.05, I.02 (2026)
https://www.tejasjournals.com/
https://doi.org/10.63920/tjths.52034

Design and Implementation of an NLP-Integrated Desktop Airline Management System

Ayush Yadav[1], Suyash Sharma[2], Nandini Agnihotri[3], Vaishali Gupta[4], Manish Kumar Srivastava[5]

[1,2,3,4]Department of Computer Science and Engineering, Shri Ramswaroop Memorial University, Lucknow, India

[5]Assistant Professor, Department of Computer Science and Engineering, Shri Ramswaroop Memorial University, Lucknow, India

ayushyadav20047@gmail.com, suyashs.0105@gmail.com, nandini.agnihotri270@gmail.com, vg5654005@gmail.com, srivastava.manish619@gmail.com

KEYWORD

Airline Management System, Python, Tkinter, MySQL, Natural Language Processing, Chatbot, GUI.

ABSTRACT

Air travel today depends heavily on digital systems, yet many booking platforms still feel unnecessarily complicated. Users are often required to go through multiple steps, fill in repetitive details, and navigate confusing interfaces just to complete a simple task like booking a ticket or checking flight information. These issues not only slow down the process but can also lead to mistakes and frustration. This project was developed with the intention of making such interactions simpler, smoother, and more user-friendly. The proposed Airline Management System is a desktop-based application built using Python, focusing on ease of use and practical functionality. The interface is designed using Tkinter, along with ttk widgets, to give it a cleaner and more organized look compared to basic GUI designs. Special attention has been given to keeping the layout simple so that even someone with minimal technical knowledge can use the system without difficulty. Since the application runs locally on a system, it does not depend heavily on continuous internet access, which makes it more reliable in certain situations. To handle data efficiently, a MySQL database is used to store all essential information such as passenger details, flight records, and booking data. This ensures that the data remains structured and easy to manage. The connection between the application and the database is established using PyMySQL, which allows the program to send and receive data through SQL queries. This setup helps maintain accuracy and ensures that every action performed by the user is properly recorded and reflected in the system.

1. INTRODUCTION

Over the past few years, the way people plan and book travel has quietly but completely changed. What once required standing in long queues or speaking to travel agents is now expected to happen instantly, often within a few clicks. While this shift to digital platforms has made travel more accessible, it has also exposed a different kind of problem—most booking systems are still not as smooth or dependable as users expect them to be [1], [2]. Something as simple as reserving a seat can turn frustrating when interfaces are confusing, inputs are repetitive, or the system fails at a crucial moment like payment confirmation.

Corresponding Author: Ayush Yadav, Department of Computer Science & Engineering, Shri Ramswaroop Memorial University, Lucknow, INDIA
Email: ayushyadav20047@gmail.com

TEJAS Journal of Technologies and Humanitarian Science
ISSN-2583-5599
Vol.05, I.02 (2026)
https://www.tejasjournals.com/
https://doi.org/10.63920/tjths.52034

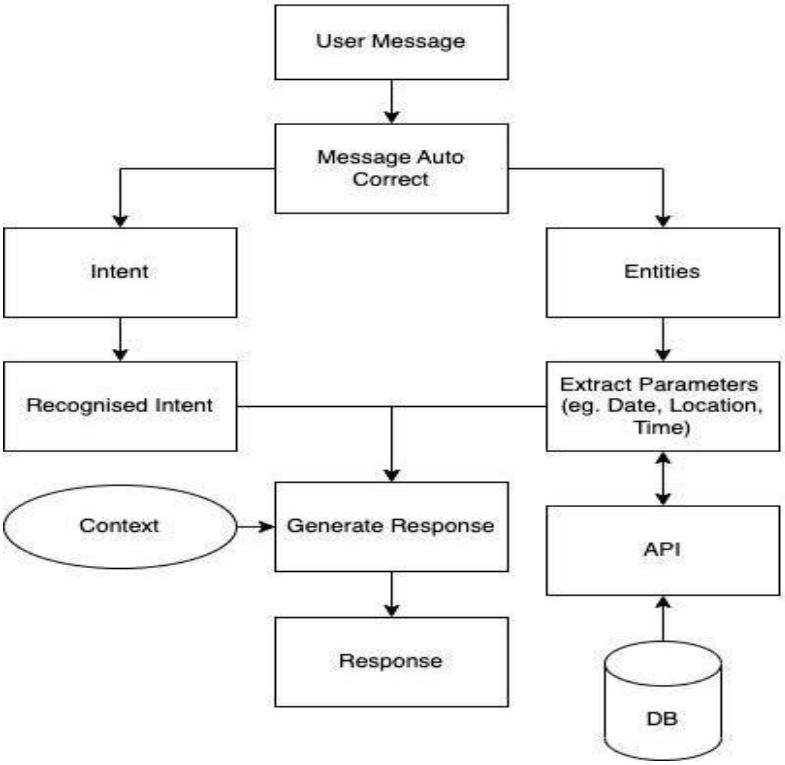

Fig. 1. High Level Design for Conversation AI Assitant

This gap between expectation and reality is what led to the development of this project. Instead of building yet another conventional booking platform filled with rigid forms and static inputs, the aim here was to design a system that feels more natural to use. The idea was simple: reduce unnecessary complexity and make the interaction feel closer to how people actually think and communicate. That is where the concept of integrating conversational elements into a desktop-based airline management system began to take shape.

At its foundation, a reservation system is not just about storing data—it acts as a connection point between the user and a highly dynamic backend where every second matters. Flight availability changes constantly, seats get booked in real time, and even a minor delay in updating this information can lead to serious issues like double booking or failed transactions. Keeping this in mind, the system in this project has been built using Python for its flexibility and MySQL for structured and reliable data handling [3], [9]. Rather than relying on web-based dependencies, the decision to create a desktop application was intentional, ensuring stability and consistent performance even in environments with limited connectivity.

consistent performance even in environments with limited connectivity.

One of the most important aspects explored in this work is how a system can better understand user intent without forcing them into strict input formats. In many traditional applications, users are required to follow a fixed pattern while entering details, which often leads to errors or incomplete submissions. To address this, this project introduces a simple, rule-based chatbot that allows users to interact using everyday language [4], [5]. By combining basic Natural Language Processing techniques with pattern matching through regular expressions, the system can interpret key details such as destinations, flight numbers, or booking requests. While the chatbot is not designed to handle open-ended conversations, it performs effectively within its defined scope and makes the interaction feel less mechanical.

Another important consideration during development was ensuring that every action performed by the user is accurately reflected in the system without delays or inconsistencies. This is achieved through an event-driven approach, where each step—whether it is entering passenger details, selecting a flight, or confirming a booking—triggers a corresponding update in the database. This method helps maintain synchro-nization between the interface and stored data, reducing the chances of errors and improving overall reliability.

TEJAS Journal of Technologies and Humanitarian Science
ISSN-2583-5599
Vol.05, I.02 (2026)
https://www.tejasjournals.com/
https://doi.org/10.63920/tjths.52034

The project also pays close attention to the user interface, which is often overlooked in technically focused systems. Using Tkinter along with ttk components, the design aims to strike a balance between simplicity and clarity [10]. Instead of overwhelming the user with too many options at once, the interface guides them step by step, providing clear feedback at every stage. This not only improves usability but also builds confidence in the system.

To make the experience more complete, a simulated pay-ment module has been included. This feature allows users to go through a process similar to real-world transactions, including options like card-based payments and QR-style interactions. Although no actual financial transactions take place, the simulation provides a realistic understanding of how such systems function while ensuring safety during testing and demonstration.

What makes this project meaningful is not just the tech-nologies used, but the intent behind how they are combined. The focus has been on creating a system that works reliably under practical conditions while keeping the user experience straightforward and stress-free. In a world where even a small technical failure can disrupt an entire travel plan, building systems that people can trust becomes essential.

In the end, this work is an attempt to move away from overly complicated designs and toward something more thoughtful and user-centered. By bringing together structured data man-agement, responsive design, and simple language-based inter-action, the project offers a grounded approach to improving how airline booking systems are built and experienced.

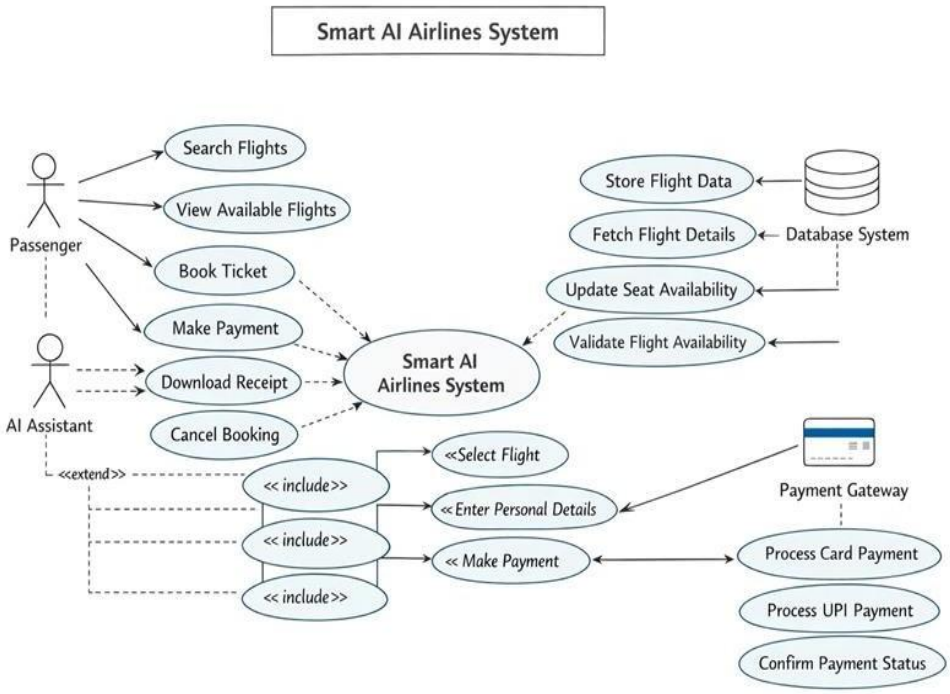

Fig. 2. Concept Flow Diagram

2. Related Work

The journey of airline reservation systems has not been a sudden transformation but rather a gradual shift shaped by both technological progress and changing user expectations. Earlier systems were built with a primary focus on function-ality—getting the job done—without paying much attention to how users actually experienced the process. Over time, as more people began relying on digital platforms for everyday tasks, it became clear that efficiency alone was not enough. Systems needed to be intuitive,

TEJAS Journal of Technologies and Humanitarian Science
ISSN-2583-5599
Vol.05, I.02 (2026)
https://www.tejasjournals.com/
https://doi.org/10.63920/tjths.52034

responsive, and capable of reducing human effort rather than adding to it. This section reflects on the key ideas and approaches that influenced the development of the current project.

A. Improving User Interaction Through Thoughtful Interface Design

One of the first areas that stood out during exploration was the role of interface design in shaping user behavior. Traditional desktop applications often presented all options at once, forcing users to figure out where to begin and what to do next. This frequently led to confusion, especially for first-time users.

To address this, modern design practices have gradually moved toward structured and visually separated layouts. While working on this project, tools like Tkinter were studied not just for their functionality but for how they could be used to guide users step by step. The use of themed widgets helped create a cleaner appearance, where different sections—such as searching for flights and entering passenger details—are clearly distinguished. This approach reduces cognitive load and makes the system feel more approachable.

Another important shift has been toward event-based in-teraction. Instead of waiting for users to complete an entire form before responding, the system reacts to each action as it happens. This makes the application feel more responsive and reduces the chances of errors going unnoticed.

B. Making Systems Understand Simple Human Language

A major limitation of many existing systems is their de-pendence on strict input formats. Users are often required to follow specific patterns while entering data, and even a small deviation can lead to errors. This rigid structure does not align well with how people naturally communicate.

To overcome this, several studies and practical implementations have explored the use of language-based interaction [4], [5]. While advanced AI models exist, they are not always necessary for smaller, well-defined tasks. In this project, a simpler approach has been adopted by combining pattern recognition with basic natural language processing techniques. By analyzing how users typically phrase their requests, it becomes possible to identify important details such as flight numbers or destinations without requiring perfectly structured input. Techniques like pattern matching allow the system to filter out unnecessary words and focus only on what matters.

This creates a smoother interaction where users do not feel restricted by the system's requirements.

Ensuring Reliability Through Strong Data Management

No matter how well-designed an interface is, the system ultimately depends on how accurately it handles data. In airline reservation systems, even a small inconsistency can lead to major problems such as duplicate bookings or incorrect passenger records.

Research into database management highlighted the impor-tance of maintaining a single, reliable source of truth. Rela-tional databases, particularly MySQL, have been widely used for such applications because they provide structured storage and allow precise control over data operations. By connecting the application to the database through a dedicated interface, every action—whether it is a booking or a cancellation—can be immediately recorded and verified.

An important concept explored here is real-time data con-sistency. When multiple users interact with the system simul-taneously, it is essential to ensure that changes are reflected instantly. This prevents conflicts and keeps the system depend-able even under higher usage.

Creating a Practical Approach to Payment Handling

The final step in any booking process is often the most sensitive—payment. In real-world applications, this stage re-quires a high level of security and user trust. While developing this project, the focus was not on integrating actual financial systems but on understanding how such processes work and how they can be represented safely.

Modern payment methods have set clear expectations for speed and simplicity. Users are now accustomed to quick transactions through digital wallets and QR-based systems. Keeping this in mind, a simulated payment process was designed to reflect these real-world practices. Instead of a single confirmation step, the process includes multiple stages, giving users a sense of control and clarity.

This simulation not only helps in demonstrating the com-plete booking cycle but also highlights the importance of structured workflows in handling sensitive operations.

Bringing It All Together

What stands out from studying existing approaches is that no single component defines a good system. It is the combi-nation of clear interface design, flexible interaction methods, reliable data handling, and

TEJAS Journal of Technologies and Humanitarian Science
ISSN-2583-5599
Vol.05, I.02 (2026)
https://www.tejasjournals.com/
https://doi.org/10.63920/tjths.52034

thoughtful process flow that creates a meaningful user experience.

The current project builds upon these ideas by focusing on simplicity and practical usability rather than unnecessary complexity. Instead of trying to replicate large-scale systems, it takes a more grounded approach—understanding common user challenges and addressing them through straightforward, well-integrated solutions.

Materials and Methods While working on this project, the main focus was not just on getting the system to function, but on making sure it behaves in a stable and predictable way when real users interact with it. Every part of the sys-tem—from storing data to handling user input—was designed with practicality in mind. Instead of depending on external services or live APIs, the entire setup was kept local so that the behavior of the system could be controlled, tested, and improved without outside interference.

C. Handling Data Through a Structured Database

At the core of the system lies a simple but well-organized database. Rather than pulling information from the internet, all flight-related details are stored locally. This includes flight numbers, destinations, pricing, and seat availability.

A relational database was chosen because it keeps every-thing structured and easy to manage. Each record is clearly defined, which reduces the chances of inconsistencies. When-ever a user searches for a flight or completes a booking, the system directly interacts with this database to fetch or update the required information.

To make this connection possible, a Python-based interface is used to communicate with the database. Every action performed by the user—whether it is viewing available flights or confirming a booking—results in a corresponding database operation. This ensures that the data shown on the screen always reflects the actual state of the system.

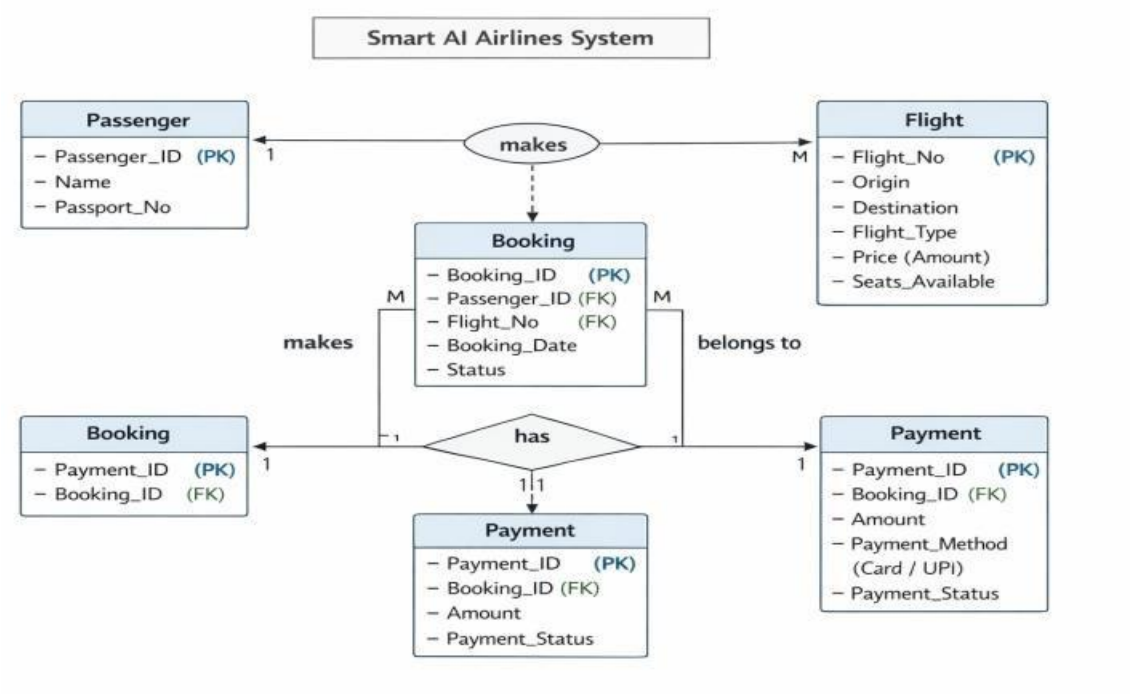

Fig. 3. ER Diagram of Smart AI Airlines System

D. Processing User Input in a Practical Way

One of the more interesting challenges in this project was dealing with how users type their requests. Unlike structured forms, chat-based input can vary a lot from person to person. Some users may type complete sentences, while others may only enter a few words.

TEJAS Journal of Technologies and Humanitarian Science
ISSN-2583-5599
Vol.05, I.02 (2026)
https://www.tejasjournals.com/
https://doi.org/10.63920/tjths.52034

To handle this variation, the input first goes through a simple cleaning process. Extra spaces are removed, and the text is converted into a consistent format so that it can be analyzed more easily. After that, the sentence is broken down into smaller parts, allowing the system to look for specific words that indicate what the user wants to do.

Instead of relying on complex models, a more direct ap-proach is used to pick out important details. For example, if a user mentions a number within a sentence, the system checks whether it could represent a flight number. Similarly, certain keywords are used to identify whether the user is trying to search for flights or make a booking.

This method may seem basic, but it works reliably within the defined scope of the system and avoids unnecessary complexity.

E. Understanding User Actions and System Flow

To keep the interaction smooth, the system keeps track of what the user is currently doing. For instance, if a booking process has already started, the system knows whether it should ask for a name, a contact detail, or confirmation. This step-by-step tracking helps avoid confusion and ensures that no important detail is skipped.

Another important aspect is how the system responds to user actions. Instead of waiting for everything to be completed at once, each click or input triggers an immediate response. This makes the application feel more active and reduces delays in feedback.

By combining this step-based tracking with responsive actions, the system maintains a clear flow from start to finish.

F. Building the Interface and System Structure

The overall structure of the application is kept simple so that it remains easy to understand and use. The interface is divided into clear sections, allowing users to move between different tasks without feeling lost.

The design focuses on clarity rather than decoration. Each part of the interface serves a purpose—whether it is displaying available flights, collecting user details, or guiding the booking process. This reduces unnecessary distractions and helps users complete their tasks more efficiently.

To make the experience more complete, a simulated pay-ment process has been included. This part of the system walks the user through steps similar to real transactions, such as entering details or scanning a code. Although no real payment is processed, the flow gives a realistic sense of how such systems operate.

In addition to this, once a booking is successfully com-pleted, the system generates a digital receipt. This serves as confirmation and gives the user a tangible output from the process.

G. Tools and Technologies Used

The entire system is developed using Python, mainly be-cause of its flexibility and ease of implementation for desktop applications. Supporting libraries are used wherever necessary to handle specific tasks like database communication or gen-erating output files.

The interface is built using a standard GUI framework, which allows the application to run smoothly on different systems without requiring heavy resources. Overall, the tools were selected based on how well they support the goal of building a simple and reliable application rather than adding unnecessary complexity.

H. Challenges Faced During Development

Like any practical project, this one also came with its share of difficulties. One of the main challenges was keeping the interface and the database in sync at all times. Even a small delay in updating information could create confusion, especially when dealing with seat availability.

Another issue was handling unexpected user input. Since users do not always follow a fixed pattern while typing, it was necessary to refine the input handling process multiple times to make it more adaptable.

Maintaining a smooth response time was also important. The system needed to process user actions quickly while still performing background operations like database updates. Balancing these tasks required careful structuring of how different parts of the system interact.

Overall, the methods used in this project are grounded in simplicity and practicality. Instead of aiming for overly complex solutions, the focus has been on building a system that works consistently, responds quickly,

TEJAS Journal of Technologies and Humanitarian Science
ISSN-2583-5599
Vol.05, I.02 (2026)
https://www.tejasjournals.com/
https://doi.org/10.63920/tjths.52034

and remains easy for users to understand and interact with.

3. DATA ANALYSIS

In this stage of the project, the focus was on understanding how user input travels through the system and eventually becomes meaningful, structured data. Instead of treating data analysis as just numbers and tables, the approach here was more practical—observing how people interact with the system and ensuring that their intent is captured correctly without causing confusion or errors. The aim was to make sure that what a user types is accurately reflected in what the system understands and stores.

A. Making User Input Usable

When users interact with the system through the chat interface, their input is rarely perfect. Some type full sentences, others use short phrases, and sometimes there are typing mistakes or unnecessary words. Because of this, the first step was to make the input easier for the system to process.

A simple cleaning process removes extra spaces and avoids inconsistencies that might affect matching. The text is then converted into a uniform format so that the system does not treat similar inputs differently just because of case differences. After that, the sentence is broken into smaller parts, which helps in identifying key action words.

Another important part of this step is identifying useful details hidden within the sentence. For example, if a user in-cludes a number, the system checks whether it could represent something important like a flight number. By focusing only on relevant pieces of information, the system avoids getting distracted by unnecessary text.

B. Understanding the Structure of Stored Data

Once the input is processed, the next step is to see how well it fits with the data stored in the system. The database contains information about flights, including destinations, pricing, and seat availability. During analysis, attention was given to how this data is organized and accessed.

By observing common patterns—such as frequently searched routes or typical booking behavior—it becomes easier to design queries that return results quickly and accurately. This also helps in ensuring that users are shown relevant options without delay.

Another important aspect is maintaining accuracy in seat availability. The system is designed in such a way that once a seat is booked, the change is immediately reflected everywhere. This prevents situations where multiple users might try to book the same seat at the same time.

C. Identifying What Really Matters in a Conversation

Not every part of a user's message is equally important. One of the key tasks during this phase was figuring out how to separate useful information from unnecessary words.

The system pays attention to certain keywords that indicate what the user wants to do, such as searching for flights or making a booking. At the same time, it keeps track of where the user is in the overall process. For example, if the system has already asked for a name, it expects the next input to be related to that, rather than starting the process again.

This step-by-step understanding makes the interaction feel more natural and prevents the user from getting lost in the process.

D. Checking How the System Performs in Real Situations

After setting up the logic, the system was tested under different scenarios to see how well it performs. Both the manual interface and the chat-based interaction were used to complete bookings and retrieve information.

It was observed that the guided nature of the chat system often helped users avoid mistakes, especially when they were unfamiliar with the process. By asking for one detail at a time, it reduced the chances of missing or incorrect inputs.

Error handling was also an important part of testing. Sit-uations such as entering invalid details or trying to book a full flight were carefully checked to ensure that the system responds correctly every time. The goal was to make sure that the system does not proceed unless all conditions are properly met.

TEJAS Journal of Technologies and Humanitarian Science
ISSN-2583-5599
Vol.05, I.02 (2026)
https://www.tejasjournals.com/
https://doi.org/10.63920/tjths.52034

E. Key Observations from the Analysis

One of the most important findings was that a clear sequence of steps makes the entire process more reliable. When users are guided from one stage to the next—such as selecting a flight before entering personal details—it becomes easier to manage the data without confusion.

Another observation was the importance of having a single source of truth. By linking every part of the system to the same database, inconsistencies were avoided, and all updates remained synchronized.

It was also noticed that immediate feedback plays a big role in user confidence. When the system confirms actions clearly—such as showing a successful booking message—it creates a sense of trust and completion.

F. How the System Handles Decisions Internally

Unlike systems that rely on probability or prediction, this project follows a clear and direct approach. Every decision is based on predefined conditions. If the input matches certain

criteria, the system performs the corresponding action. This reduces uncertainty and ensures that the output is always predictable.

The database plays a central role in this process by acting as a reference point for all operations. Whenever the system needs to verify something—such as whether a flight exists or if seats are available—it checks the stored data before proceeding.

The final stage of the process involves the payment sim-ulation, which acts as a checkpoint. Only after this step is completed does the system confirm the booking and update the records. This ensures that every reservation follows a complete and verified path.

G. Tools Supporting the Analysis

The analysis and overall system behavior are supported by Python, which handles the processing and logic. The database manages all stored information, while simple pattern-matching techniques help in interpreting user input. Together, these components create a system that is not only functional but also consistent in how it handles real-world interactions.

Overall, this phase was less about complex calculations and more about understanding behavior—both human and system-level. By focusing on clarity, consistency, and reliability, the project ensures that data flows smoothly from input to output without losing its meaning along the way.

4. RESULTS

After putting the system through multiple runs and observ-ing how it behaves on the actual interface, the results turned out to be quite practical and grounded. Instead of focusing only on theoretical performance, the evaluation was based on how smoothly a user could move from searching a flight to completing a booking without confusion or delay.

From the screenshots and the working flow, it is clear that the system is able to maintain a steady connection between what the user sees on the screen and what is actually hap-pening in the backend. The flight listing, booking section, and payment area all appear to stay in sync, which is one of the most important requirements for a system like this.

A. System Behaviour During Usage

While testing the application, one thing that stood out was how the interface responds to user actions. Whether it was selecting a flight, entering details, or moving to the payment section, the transitions felt continuous rather than broken. There were no visible freezes or delays, which shows that the internal flow has been handled carefully.

The chat-based input also worked in a predictable way [6]. It didn't try to overcomplicate things but instead focused on picking up clear instructions. In most cases, the system was able to understand what the user intended and guide them forward without requiring repeated inputs.

B. Flight Data and Display Accuracy

Looking at the flight details shown in the interface (like flight numbers, routes, and seat counts), the

TEJAS Journal of Technologies and Humanitarian Science
ISSN-2583-5599
Vol.05, I.02 (2026)
https://www.tejasjournals.com/
https://doi.org/10.63920/tjths.52034

data appears structured and consistently presented. The table format makes it easy to scan through available options, and there is no mismatch between displayed values and what is expected during booking. Another important observation is that once a booking step

C. Booking Flow and User Guidance

The booking process follows a clear order. Instead of asking for everything at once, the system moves step by step, which makes it easier for the user to follow. This is especially useful for someone using the system for the first time.

The assistant-like behavior helps in reducing mistakes. For example, the system does not allow users to skip essential details, which improves the overall reliability of the booking process. Compared to a fully manual form, this approach feels more controlled and less error-prone.

D. Payment Simulation Output

From the payment section shown in the output, the system successfully creates a realistic transaction environment. The amount displayed matches the selected flight, which indicates that the calculation and data transfer between modules are working correctly.

The presence of a UPI-style payment option adds to the realism. Even though it is a simulation, it gives a clear idea of how the system would behave in a real-world scenario. The transition from booking to payment also happens smoothly, without resetting or losing entered data.

E. Consistency Between Interface and Backend

One of the strongest results observed is the consistency of data. The system ensures that every action taken by the user is reflected immediately. There is no visible delay between selecting a flight and seeing its effect in the next step.

This shows that the connection between the application and the database is stable [9]. It also reduces the risk of issues like duplicate bookings or incorrect seat availability, which are common problems in poorly designed systems.

F. Overall Observations

A few important points became clear after testing the system:

- The step-by-step flow makes the process easier to follow and reduces user mistakes.
- The interface remains stable even when multiple actions are performed quickly.
- The displayed data stays consistent throughout the pro-cess.
- The payment simulation adds completeness to the book-ing experience.
- The system avoids unnecessary complexity and focuses on doing basic tasks correctly.

G. Final Outcome

In practical terms, the system achieves what it was designed for—it provides a simple and reliable way to manage airline bookings in a desktop environment. It does not try to imitate large commercial platforms but instead focuses on clarity, control, and consistency.

The results show that even with a straightforward approach, it is possible to build a system that handles real-world scenar-ios effectively. By keeping the design focused and avoiding is initiated, the system does not lose track of the selected flight. This continuity is important because it avoids confusion and ensures that the correct details are carried forward to the final stages. overcomplication, the project delivers a working model that feels stable, understandable, and usable from a user's point of view.

5. CONCLUSION

Looking back at the entire process, this project turned out to be much more than just building a working application. It gave me a clearer understanding of how small design decisions can significantly affect how people interact with software.

What stands out the most is the shift from a purely form-based system to something that feels more interactive [4], [7]. Adding a conversational element, even in a basic form, changes the way users

TEJAS Journal of Technologies and Humanitarian Science
ISSN-2583-5599
Vol.05, I.02 (2026)
https://www.tejasjournals.com/
https://doi.org/10.63920/tjths.52034

approach the application. It removes some of the friction and makes the process feel more guided rather than forced.

At the same time, I learned that no matter how user-friendly an interface is, the backend has to be dependable [8]. Keeping the database in sync with every action was one of the most critical parts of this project. It ensured that the system remained accurate at all times, especially when handling bookings.

Working with Python and Tkinter also showed me that you don't always need complex frameworks to build something effective. With the right structure and logic, even simple tools can be used to create applications that are responsive and practical.

The payment module was another part that added depth to the project. Designing it in a way that reflects real-world systems made the application feel more complete and less like a basic prototype.

Of course, there's still a lot that can be improved. The assistant can be made more flexible, the system can be expanded to handle larger datasets, and the entire application could be moved to an online platform for better accessibility. But even in its current form, it serves as a strong starting point. In the end, this project reinforced a simple idea: a good application isn't just about features—it's about how naturally those features come together to create a smooth experience for the user.

REFERENCES

[1] P. Singh and A. Sharma, "Airline Reservation System," *International Journal of Innovative Science and Research Technology (IJISRT)*, vol. 6, no. 5, pp. 1234–1238, 2021.
[2] R. Kumar and S. Patel, "Airline Booking System Using Python," *EasyChair Preprint*, 2020.
[3] N. Verma and R. Gupta, "Creating a GUI-Based Train Ticket Booking System in Python," *ResearchGate*, 2025.
[4] D. Suhartanto et al., "AIRA Chatbot for Travel: Case Study of AirAsia," *Journal of Physics: Conference Series*, 2020.
[5] R. Pratama and Y. Nugroho, "Design of Travel Booking Application Using NLP Chatbot," *Merkurius Journal*, 2025.
[6] M. Henderson et al., "A Repository of Conversational Datasets for Conversational AI," *arXiv*, 2019.
[7] A. Kumar and R. Bhatia, "A Survey of Chatbot Systems and Their Applications," *arXiv*, 2022.
[8] Q. Zhang, H. Liu, and Y. Chen, "Scalable and Reliable Architecture for Travel Booking Systems," *arXiv*, 2024.
[9] A. Sweigart, *Automate the Boring Stuff with Python*, 2nd ed. San Francisco, CA: No Starch Press, 2019.
[10] M. Lutz, *Programming Python*, 4th ed. Sebastopol, CA: O'Reilly Media, 2013.