



A Rule-Based Decision Support System for Bias in Automated Airport Security Checkpoints

Mrs. B. Priyanka¹, Mr. B. Karthik², R. Dheekshitha³, N. Uday Kiran⁴, P. Sunil⁵,
P. Anand Kumar⁶

^{1,2}Assistant Professor, Department of CSE, Teegala Krishna Reddy Engineering College, Telangana, India

^{3,4,5,6}B. Tech Students, Department of CSE, Teegala Krishna Reddy Engineering College, Telangana, India

Abstract- Airports require efficient systems for managing passenger information, flight bookings, luggage records, and security verification. Traditional procedures often depend on manual documentation and disconnected operations, which may cause delays, data duplication, communication gaps, and difficulties in monitoring passenger activities. This paper presents an Airport Passenger Security and Management System, a Django-based web application that integrates passenger registration, flight booking, luggage management, security verification, and administrative control into a centralized platform. Passengers can register, manage their profiles, view available flights, submit booking requests, and provide luggage details such as the number of bags, total weight, and description. Security officers can review passenger information, validate passport details, verify luggage conditions, select luggage tracking statuses, and approve or reject booking requests. The luggage tracking feature supports status updates such as Submitted, Checked, Dispatched, Loaded, In Transit, Delivered, and Missing. Administrators can manage flights, monitor bookings, and approve or reject security officer registrations through a dedicated dashboard. Email notifications provide updates regarding registration, booking, approval, rejection, and luggage status changes. The system uses Django for application development and SQLite for data storage. By combining passenger management, booking operations, security checking, and luggage tracking, the proposed system improves information organization, reduces manual effort, and supports more transparent airport management. The application also provides a foundation for future integration with biometric authentication, RFID-based luggage tracking, real-time flight information, and intelligent security assessment.

Keywords- Airport security, passenger management, flight booking, luggage tracking, security verification, Django, SQLite, web application.

I. INTRODUCTION

Airports are complex environments that manage passenger registration, flight bookings, luggage handling, identity verification, and security operations. The increasing number of passengers and flight activities creates a need for efficient, centralized, and reliable management systems. In many conventional airport processes, passenger information, booking records, luggage details, and security verification are maintained through separate procedures or manual documentation. These methods may result in data duplication, processing delays, communication gaps, and difficulties in tracking passenger and luggage activities.



Digital transformation in airport management can improve the coordination of different operational activities. A web-based application can provide a common platform through which passengers, security officers, and administrators can access information according to their responsibilities. Such a system can support online registration, flight booking, luggage recording, security verification, and timely communication. It can also maintain updated records and reduce the dependency on paperwork.

The proposed Airport Passenger Security and Management System is developed using the Django web framework and SQLite database. The system provides separate modules for passengers, security officers, and administrators. Passengers can register, log in, manage their profiles, view available flights, submit booking requests, and enter luggage details. Passport information is validated during registration, while booking requests are stored with an appropriate status for further review. Security officers can access passenger booking information after administrator approval. They can verify passport details, inspect luggage information, check luggage weight, and record security remarks. During the approval process, officers can also select a luggage tracking status such as Submitted, Checked, Dispatched, Loaded, In Transit, Delivered, or Missing. This feature allows luggage progress to be monitored along with booking information.

Administrators manage flights, passenger bookings, and security officer approvals through a dedicated dashboard. Email notifications are used to communicate registration, booking, approval, rejection, and luggage status updates. By integrating these functions, the proposed system supports organized airport operations, improves information accessibility, and reduces manual management efforts.

Background

Airports are complex environments that manage passenger registration, flight bookings, luggage handling, identity verification, and security operations. The increasing number of passengers and flight activities creates a need for efficient, centralized, and reliable management systems. Conventional airport processes often depend on separate procedures or manual documentation, which may result in data duplication, processing delays, communication gaps, and difficulties in tracking passenger and luggage activities.

Need for the System

Digital transformation can improve the coordination of airport operations by connecting passengers, security officers, and administrators through a common platform. A web-based application can support online registration, flight booking, luggage recording, security verification, and timely communication. It can also maintain updated records and reduce dependency on paperwork.

Proposed Approach

The proposed Airport Passenger Security and Management System is developed using the Django web framework and SQLite database. The system provides separate modules for passengers, security officers, and administrators. Passengers can register, log in, manage their profiles, view available flights, submit booking requests, and enter luggage details. Passport information is validated during registration, while booking requests are stored with an appropriate status for further review.

Security and Luggage Tracking

Security officers can access passenger booking information after administrator approval. They can verify passport details, inspect luggage information, check luggage weight, and record security remarks. During the approval process, officers can select a luggage tracking status such as Submitted, Checked, Dispatched, Loaded, In Transit, Delivered, or Missing. This feature allows luggage progress to be monitored along with booking information.



Administrative Management

Administrators manage flights, passenger bookings, and security officer approvals through a dedicated dashboard. Email notifications are used to communicate registration, booking, approval, rejection, and luggage status updates.

Contribution of the Work

By integrating passenger management, flight booking, security verification, and luggage tracking, the proposed system supports organized airport operations and improves information accessibility. It reduces manual management efforts and provides a foundation for future enhancements such as biometric authentication, RFID-based luggage tracking, real-time flight information, and intelligent security assessment.

II. PROPOSED SYSTEM

System Overview

The proposed Airport Passenger Security and Management System is a Django-based web application developed to integrate passenger management, flight booking, luggage verification, security approval, and administrative activities into a centralized platform. The system provides separate access interfaces for passengers, security officers, and administrators. It reduces manual documentation and improves the accessibility of passenger and booking information.

Passenger Registration and Management

Passengers can register by entering their personal information, contact details, passport number, and password. The system validates the passport number according to a predefined format, such as PASS123456. After successful registration and login, passengers can manage their profiles, view available flights, submit booking requests, and check booking updates.

Flight Booking and Luggage Details

Passengers can select available flights and submit details such as flight number, airline, source, destination, and travel date. They can also provide the number of luggage bags, total weight, and luggage description. Each booking is initially stored with a Pending status until it is reviewed by an authorized security officer or administrator.

Security Verification

Security officers can register through the application, but their accounts require administrator approval before login. Approved officers can view passenger details, booking information, passport records, and luggage details. The system performs rule-based checks to identify invalid passport formats, missing luggage information, and luggage weight exceeding the permitted limit of 23 kg.

Luggage Tracking

During security approval, the officer can select the current luggage status from a dropdown menu. The available statuses include Submitted, Checked, Dispatched, Loaded, In Transit, Delivered, and Missing. The selected status is stored in the database and displayed on the passenger dashboard. This feature supports better luggage monitoring and provides passengers with updated information about luggage movement.

Administrative Management

Administrators can manage flight information, monitor passenger bookings, approve or reject security officer registrations, and update booking decisions. The administrator dashboard provides summarized information about passengers, flights, bookings, and officers.



Notification and Database Management

The system uses SQLite to store passenger, flight, booking, luggage, and security verification records. Gmail SMTP sends registration, booking, approval, rejection, and luggage status notifications. The proposed system improves coordination, reduces manual work, and supports organized airport passenger security operations.

III. IMPLEMENTATION DETAILS

Development Environment

The Airport Passenger Security and Management System is implemented as a web-based application using Python and the Django framework. Django provides the core application structure, URL routing, database connectivity, authentication, and request-response handling. The system uses SQLite as the default database because it is lightweight, easy to configure, and suitable for academic and small-scale deployments. The application is developed using Python 3.11.2 and can be executed on Windows, Linux, or macOS systems.

The user interface is developed using HTML, CSS, and JavaScript. These technologies provide responsive forms, dashboards, tables, buttons, and status displays for passengers, security officers, and administrators.

B. System Architecture

The system follows a modular three-layer architecture. The presentation layer contains the HTML templates and user interface components. The application layer contains Django views, authentication functions, validation rules, booking operations, and security verification logic. The data layer uses SQLite and Django ORM to store passenger details, flight information, booking records, luggage details, officer approvals, and tracking status.

Django URL routing connects user requests to the appropriate views. Each user role is provided with a separate dashboard according to their permissions. This role-based structure prevents unauthorized access to administrative and security-related operations.

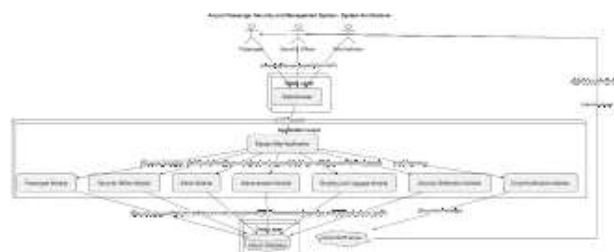


Fig 1: System Architecture Diagram of Proposed System

C. Passenger Registration and Authentication

The passenger module allows users to create an account by entering personal details, contact information, passport information, and login credentials. The passport number is validated according to the defined format. A valid passport number must begin with PASS followed by numeric characters, such as PASS123456.

After registration, the passenger can log in using the registered credentials. Django authentication is used to verify the username and password. Once authenticated, the passenger is redirected to the passenger dashboard, where available services such as flight viewing, booking, luggage entry, and booking status tracking are displayed.



D. Flight Booking Implementation

The administrator creates and manages flight records, including flight number, source, destination, date, time, and available seats. These records are displayed to passengers through the flight management interface.

A passenger selects a flight and submits a booking request. The booking record is initially stored with a Pending status. The system links the booking with the passenger and selected flight using database relationships. This enables the application to retrieve passenger-specific bookings and display the current booking status.

The system also checks whether the selected flight is available before accepting the booking request. After submission, the booking becomes available to the security officer for verification.

E. Luggage Details and Validation

During the booking process, passengers enter luggage-related information such as luggage quantity, luggage weight, and luggage description. These details are stored in the database and linked with the corresponding booking.

The system performs basic validation on the submitted luggage information. If luggage details are missing or the weight exceeds the defined limit of 23 kilograms, a warning is displayed to the security officer. These checks help identify possible luggage-related issues before final approval.

The validation messages are used as decision-support information. The security officer can examine the passenger details and make the final approval or rejection decision.

F. Security Officer Verification

Security officers are responsible for reviewing passenger bookings and verifying the submitted information. Officer accounts require administrative approval before the officer can access the security dashboard.

After login, the officer can view passenger information, passport details, flight information, luggage details, and booking status. The officer checks whether the passport format is valid, whether the luggage information is complete, and whether the luggage weight is within the permitted limit.

The officer can approve or reject a booking. If the booking is approved, its status changes from Pending to Booked. If the booking is rejected, the status changes to Rejected. The officer can also add remarks explaining the reason for rejection or any additional verification requirements.

G. Luggage Tracking Implementation

The system provides a luggage tracking feature to record the progress of passenger luggage. The security officer can update the luggage status using a predefined dropdown list.

H. Administrative Management

The administrator controls the main operations of the system. The administrator can view the dashboard, manage passenger records, approve or reject security officer registrations, create and update flight details, and monitor booking information.

The administrator also has access to booking records and can review the status of passenger requests. Administrative operations are protected through role-based access control so that passengers and security officers cannot access administrator functions.



Email Notification and Database Operations

Email notifications are implemented using Gmail SMTP configuration in Django. When a booking is approved or rejected, the system sends an email to the registered passenger. The email contains the booking status and, where applicable, the remarks entered by the security officer.

Django ORM is used to perform database operations such as inserting, updating, retrieving, and deleting records. The main database entities include passengers, security officers, flights, bookings, luggage details, and approval records. This centralized database structure reduces duplication and supports consistent access to system information.

IV. RESULTS AND PERFORMANCE ANALYSIS

Passenger Dashboard Results

The developed system provides a user-friendly passenger dashboard through which passengers can manage their profiles, view available flights, and monitor booking information. The dashboard displays the passenger's travel history in a structured table containing the flight number, airline, source, destination, travel date, and booking status.

As shown in Fig. 2, the passenger can view multiple booking records with different statuses, such as Booked and Rejected. This allows passengers to understand the current condition of their travel requests without contacting the airport administration manually. The dashboard also provides direct access to profile management and available flight services.



Fig 2. Passenger Dashboard Showing Travel History and Booking Status

The dashboard design separates important functions into clearly visible sections. The welcome message, profile option, flight search option, and booking history improve navigation and accessibility. The status indicators help passengers quickly identify whether a booking has been confirmed or rejected.

Passenger Security Verification Results

The security verification interface allows security officers to review passenger information and booking details before approving or rejecting a booking request. The page displays important information such as passenger ID, full name, email address, phone number, passport number, flight number, airline, source, destination, travel date, booking date, and current booking status.

As shown in Fig. 3, the security officer can examine the passenger's personal details and flight information from a centralized interface. In the illustrated record, the passport number and booking information are displayed together with the booking status. This arrangement helps the officer perform verification using the available passenger and travel data.



Fig 3. Passenger Security Details Interface for Booking

Verification

The security details page supports the security verification workflow by presenting the information required for decision-making in a single view. It reduces the need to search through separate records and improves coordination between passenger management and security operations. The booking status displayed on the page also helps the officer identify whether further action is required.

C. Booking Status Management

The system successfully supports different booking states, including Pending, Booked, and Rejected. A new booking request is initially stored as Pending. After reviewing the passenger details and luggage information, the security officer can approve or reject the request.

When a booking is approved, the status changes to Booked and the updated information becomes visible on the passenger dashboard. If the booking is rejected, the status changes to Rejected. The screenshots demonstrate that the booking status is correctly reflected in both the passenger dashboard and the security verification interface.

D. User Interface and Usability Analysis

The application uses a consistent dashboard design with clearly separated sections, readable labels, status indicators, and navigation options. The passenger dashboard provides quick access to profile details and flight booking services, while the security interface organizes passenger and flight information into separate information panels.

The interface improves usability by presenting important data in a structured format. The use of status labels allows users to identify booking conditions quickly. The design also supports role-based interaction, where passengers view their own booking information and security officers access passenger records required for verification.

E. Overall Result

The implemented system successfully demonstrates the integration of passenger services and airport security verification. The screenshots confirm that passenger booking records are displayed correctly and that security officers can access the required passenger and flight details. The application provides a centralized platform for booking management, security assessment, and status communication.

The results indicate that the system can reduce manual record handling, improve information accessibility, and support faster communication between passengers, security officers, and administrators. Further performance evaluation with a larger number of users and real-time airport data can be carried out in future work.



VI. METHODOLOGY

The proposed Airport Passenger Security and Management System uses a structured web-based methodology developed using Django. The process begins with passenger registration and login, followed by flight selection and booking. The passenger provides the required personal, passport, and luggage details, which are stored in the SQLite database. Each new booking is initially assigned a Pending status. The system performs basic validation of passenger and luggage information, including passport format checking and luggage-weight verification. This centralized approach allows passenger, flight, booking, and luggage information to be maintained in an organized manner.

After a booking is submitted, the security officer reviews the passenger, flight, passport, and luggage details. The system identifies conditions such as an invalid passport format, missing luggage information, or luggage weight exceeding 23 kg and displays the corresponding warnings to the officer. The officer then makes the final approval or rejection decision. An approved booking is marked as Booked, while a rejected booking is marked as Rejected and remarks can be recorded. The system also provides luggage tracking through statuses such as Submitted, Checked, Dispatched, In Transit, Delivered. Updated booking information is displayed to the passenger, and email notifications are sent when the booking status changes.

VII. FUTURE WORK

Future work will focus on improving the functionality, security, scalability, and intelligence of the Airport Passenger Security and Management System. Biometric authentication, such as facial recognition and fingerprint verification, can be integrated to strengthen passenger identity validation. The system can also be connected with real-time flight information APIs to provide updated flight schedules, delays, cancellations, and gate details. RFID or barcode-based luggage tracking may be introduced to monitor baggage movement more accurately throughout the airport.

Advanced risk-assessment techniques can be developed to identify unusual passenger or booking patterns and support security officers during verification. Online payment integration can be added to support complete flight booking transactions. The application may also be deployed on cloud infrastructure to support larger numbers of users and improve availability. A mobile application, multilingual interface, automated reports, digital passport verification, and an AI-based passenger support chatbot can further improve accessibility and user experience. These enhancements can make the system more suitable for real-world airport operations.

REFERENCES

1. J. Skorupski and P. Uchroński, "Evaluation of the effectiveness of an airport passenger and baggage security screening system," *Journal of Air Transport Management*, vol. 66, pp. 53–64, 2018, doi: 10.1016/j.jairtraman.2017.10.006.
2. Q. Feng and H. Sahin, "A mathematical framework for sequential passenger and baggage screening to enhance aviation security," *Computers & Industrial Engineering*, vol. 57, no. 1, pp. 148–155, 2009, doi: 10.1016/j.cie.2008.11.005.
3. S. H. Jacobson and L. A. McLay, "A sequential stochastic security system design problem for aviation security," *Transportation Science*, vol. 41, no. 2, pp. 135–150, 2007, doi: 10.1287/trsc.1070.0198.
4. J. Skorupski, P. Uchroński, and A. Łach, "A method of hold baggage security screening system throughput analysis with an application for a medium-sized airport," *Transportation Research Part C: Emerging Technologies*, vol. 88, pp. 52–73, 2018, doi: 10.1016/j.trc.2018.01.009.



5. D. Anderson, "Optimising multi-layered security screening," *Journal of Transportation Security*, vol. 14, pp. 249–273, 2021, doi: 10.1007/s12198-021-00237-3.
6. K. J. Liang, J. B. Sigman, G. P. Spell, D. Strellis, W. Chang, F. Liu, T. Mehta, and L. Carin, "Toward automatic threat recognition for airport X-ray baggage screening with deep convolutional object detection," *arXiv preprint arXiv:1912.06329*, 2019.
7. S. Cao, A. Kasliwal, H. Zheng, M. Reihanifar, F. Robuste, and M. Hansen, "Robust management of airport security queues considering passenger non-compliance with chance-constrained optimization," *arXiv preprint arXiv:2505.05717*, 2025.
8. S. Cao, A. Kasliwal, M. Reihanifar, F. Robuste, and M. Hansen, "Effective management of airport security queues with passenger reassignment," *arXiv preprint arXiv:2407.00951*, 2024.
9. Y. Wang, F. Meng, X. Wang, and C. Xie, "Optimizing the passenger flow for airport security check," *arXiv preprint arXiv:2312.05259*, 2023.
10. R. Singh, "Safety and security management at international airports: A study of passenger perceptions and operational practices," *International Journal for Multidisciplinary Research*, vol. 8, no. 3, 2026.
11. J. Skorupski and P. Uchroński, "Managing the process of passenger security control at an airport using the fuzzy inference system," *Expert Systems with Applications*, 2016.
12. J. Skorupski and P. Uchroński, "The key design parameters of checked baggage security screening systems in airports," *Journal of Air Transport Management*, 2005.
13. K. Kierzkowski and J. Kisiel, "Evaluation of the effectiveness of an airport passenger and baggage security screening system," *Journal of Air Transport Management*, vol. 66, pp. 53–64, 2018.
14. D. C. Montgomery, *Introduction to Statistical Quality Control*, 8th ed. Hoboken, NJ, USA: Wiley, 2019.
15. I. Sommerville, *Software Engineering*, 10th ed. Boston, MA, USA: Pearson, 2016.
16. R. Pressman and B. Maxim, *Software Engineering: A Practitioner's Approach*, 9th ed. New York, NY, USA: McGraw-Hill, 2019.
17. Python Software Foundation, "Python documentation." Python Documentation
18. Django Software Foundation, "Django documentation." Django Documentation
19. Mozilla Developer Network, "HTML, CSS, and JavaScript documentation." MDN Web Docs
20. SQLite Consortium, "SQLite documentation." SQLite Documentation