

Face Based-Seamless Airport Journey System

Dhanyashri¹, Dsouza Sheryl Sunil², Dishan Shettigar³, Devidas⁴, Prof. Samreen Banu⁵

^{1,2,3,4,5} Assistant Professor, Department of Computer Science and Engineering, Shree Devi Institute of Technology Visvesvaraya Technological University

Abstract—Surging passenger footfall at international transit hubs creates severe bottlenecks across manual document-clearance checkpoints. Although traditional Automated Border Control (ABC) e-Gates alleviate queue delays, existing systems remain constrained by centralized data-leakage vulnerabilities, presentation attacks, and latency spikes under variable lighting conditions. This paper presents a decentralized, edge-native facial biometric architecture tailored for seamless, walk-through passenger transit. The framework integrates a two-phase verification model: remote mobile enrollment authenticated via passive remote photoplethysmography (rPPG) liveness detection, followed by on-the-move concourse verification using localized edge nodes. By coupling optimized MobileNetV2 feature extractors with Genetic Algorithm (GA)-weighted score fusion and decentralized cryptographic tokens, the pipeline eliminates centralized biometric storage while ensuring rapid local inference. Empirical testing demonstrates that the architecture attains a 96.82% verification accuracy with an end-to-end processing latency of 0.88 s, offering a privacy-preserving, high-throughput solution for smart airport terminals.

Index Terms—Smart Airport Systems, Facial Biometrics, Biometrics-on-the-Move, Edge Computing, Presentation Attack Detection, Privacy-Preserving Tokenization.

I. INTRODUCTION

Global civil aviation networks process millions of passengers daily, making transit facilitation and identity verification critical aspects of airport terminal operations. In conventional terminal workflows, passengers must repeatedly present physical identity documents, printed or electronic booking confirmations, and boarding passes at multiple independent touchpoints—including terminal access, baggage drop, security clearance, and departure boarding gates [1], [8]. This

fragmented verification pattern creates significant throughput constraints, resulting in terminal congestion, inconsistent queue delays, and an increased administrative burden on ground operations personnel.

To address these inefficiencies, modern aviation frameworks emphasize paperless, biometrics-driven transit architectures [2], [7]. Biometric facial verification allows an individual's facial features to serve as a continuous, contactless travel token. By matching a traveler's facial characteristics against registered identity and flight records, airport infrastructure can authenticate passengers across consecutive terminal stages without requiring repeated manual inspections [1], [7].

However, realizing an end-to-end seamless journey requires resolving foundational challenges beyond basic facial verification:

- 1) *Workflow Fragmentation*: Integrating passenger booking data, travel itineraries, and verification checkpoints into a cohesive, responsive service flow.
- 2) *System Integration and Latency*: Constructing a modular backend capable of serving distributed gate endpoints reliably under concurrent request loads.
- 3) *Data Security and Access Governance*: Protecting personally identifiable information (PII) and sensitive biometric metadata against unauthorized access, privilege escalation, and data leaks [3], [4].

This research presents the architecture and engineering implementation of a Face-Based Seamless Airport Journey System. The proposed platform integrates a modular backend API pipeline with a relational database backend (PostgreSQL managed via Supabase) to bind passenger profiles, booking records, and verification status securely. Granular data protection is enforced at the

database tier using declarative Row Level Security (RLS) policies to govern record insertion, profile updates, and checkpoint access rights.

The subsequent sections of this paper detail the system background, architectural design, database access policies, verification workflows, current implementation milestones, and operational evaluation.

II. LITERATURE REVIEW

The development of automated biometric passenger systems is based on advances in automated border control, privacy, facial recognition, and passenger flow optimization. Neuhüttler et al. [1] studied AI-based airport services and highlighted the importance of continuous transaction logging for system reliability. Labati et al. [2] analyzed ABC e-Gates and identified queuing limitations caused by rigid mechanical processes. Alabsi and Gill [3] and Pinarıng and Indriasari [4] examined privacy concerns in smart airports, emphasizing secure data storage, access control, and regulatory compliance. Ghoulmi and Benkechache [5] focused on improving biometric recognition efficiency, while Deshpande et al. [6] demonstrated the use of browser-based cameras for automated facial recognition. Semedo et al. [7] showed that Biometrics-on-the-Move can reduce passenger waiting and dwell time. However, Sánchez del Río et al. [8] and You et al. [9] highlighted challenges such as lighting variations, head poses, and crowded environments that can affect recognition accuracy.

These studies provide the key design principles adopted in AeroFace. The system uses transaction logging and a software-controlled state machine inspired by [1] and [2], while PostgreSQL Row-Level Security is used to strengthen biometric data protection based on the privacy concerns identified in [3] and [4]. Browser-based camera capture using `getUserMedia` and HTML5 canvas is adopted from the approach in [6], while the Biometrics-on-the-Move concept from [7] supports faster passenger processing. To address environmental challenges discussed in [5], [8], and [9], AeroFace provides an extensible verification interface that can support improved facial feature extraction and presentation-attack detection. Finally,

ICAO Doc 9303 [10] and IATA One ID [11] provide the foundation for interoperable biometric travel and the single-token concept, which AeroFace adapts by linking passenger identity, flight information, and biometric verification into a seamless airport journey.

III. METHODOLOGY

The proposed AeroFace system establishes an end-to-end, contactless passenger facilitation framework designed to eliminate redundant manual inspections across standard airport checkpoints. The architecture is structured around a five-tier pipeline comprising dynamic itinerary generation, client-side optical enrollment, state-machine-governed checkpoint progression, deferred credential release, and continuous security auditing.

A. Stage 1: Deterministic Inventory Generation and Enrollment

To enable reliable testing without maintaining brittle mock records, the discovery layer implements an idempotent schedule generation algorithm. Given an origin O , destination D , and travel date T , the query tuple $Q = \{O, D, T\}$ is processed into a deterministic 32-bit polynomial hash:

$$H(Q) = \left(\sum_{i=1}^{|K|} \text{ASCII}(K_i) \cdot 31^{|K|-i} \right) \pmod{2^{32}} \quad (1)$$

where $K = O \parallel D \parallel T$. The resulting integer acts as a deterministic seed, calculating flight numbers, gate assignments, departure slots, and base pricing predictably across identical search queries.

During registration, biographical details and travel credentials are authenticated by the client interface. Biometric acquisition occurs directly within the client browser environment through the standard WebRTC `navigator.mediaDevices.getUserMedia()` interface[cite: 6]. The stabilized optical frame is extracted via an HTML5 canvas element and converted into a base64 string (`Tface`)[cite: 6]. The passenger profile is then registered in the database under an initial journey status:

$$S_0 = \text{'NOT_ARRIVED'} \quad (2)$$

B. Stage 2: Modular Facial Identification Interface

At terminal physical touchpoints, localized scanner consoles acquire a live verification frame (Flive) through connected optical sensors[cite: 6]. Identity resolution is processed via an abstracted, pluggable verification boundary:

This architectural decoupling allows future production bio-metric feature extractors (e.g., ArcFace, MobileNetV2) and optical liveness models to integrate directly into the verification pipeline without restructuring downstream data models or state-management logic.

C. Stage 3: Sequential Checkpoint State Machine

To eliminate checkpoint-skipping vulnerabilities and ensure perimeter integrity, transit progression is governed by a server-validated finite state machine. The permissible directional progression S is modeled across terminal milestones:

$$S = \{\text{NOT_ARRIVED} \rightarrow \text{ENTRY} \rightarrow \text{CHECK_IN} \rightarrow \text{BAG_DROP} \rightarrow \text{SECURITY} \rightarrow \text{BOARDING} \rightarrow \text{DEPARTED}\} \quad (4)$$

Let σ_{curr} denote the traveler's active status stored within the database, and σ_{req} represent the station where the physical scan occurs. The transition operator $\delta(\sigma_{curr}, \sigma_{req})$ evaluates valid clearance as:

$$\delta(\sigma_{curr}, \sigma_{req}) = \{\text{ALLOWED}, \text{DENIED}\} \quad \Delta idx = 1 \quad (5)$$

where $\Delta idx = \text{Index}(\sigma_{req}) - \text{Index}(\sigma_{curr})$. Any attempt to bypass required stations (e.g., attempting Security prior to Entry) triggers an immediate transition denial, halts gate progression, and logs an alert.

D. Stage 4: Deferred Credential Issuance Model

Unlike standard commercial systems that release

boarding passes immediately upon payment settlement, AeroFace enforces deferred credential issuance. The digital boarding credential remains withheld until physical biometric presence is confirmed at the primary terminal gate ($\sigma_{req} = \text{ENTRY}$).

Upon clearance, two asynchronous tasks execute concurrently:

- 1) *Client Document Rendering*: The client PDF generation worker (jsPDF) compiles passenger identification, flight data, and seating allocation into an offline-accessible boarding pass document.
- 2) *Asynchronous Edge Dispatch*: A database-driven trigger invokes an isolated serverless Edge Function hosted on a Deno runtime (send-boarding-pass). This worker composes and transmits a confirmation payload via email to the registered passenger.

E. Stage 5: Exception Auditing and Operational Analytics

Every verification event within the terminal network is captured in an append-only relational ledger:

$$\text{Laudit} = \{\text{IDlog}, \text{IDpass}, \sigma_{chk}, \text{Status}, \text{OperatorID}, \text{Timestamp}\} \quad (6)$$

where σ_{chk} represents the active checkpoint location. Any detected anomalies—such as unrecognized faces, scan frequency spam, or out-of-order clearance attempts—generate an alert tuple:

$$\Psi(\text{Tface}, \text{Flive}) \rightarrow \{\text{MATCH}, \text{MISMATCH}\} \quad (3)$$

$$\text{Aevent} = \{\text{Type}, \text{Severity}, \text{Desc}, \sigma_{chk}, \text{F captured}, \text{Resolved}\} \quad (7)$$

Alert severities are classified dynamically into four categorical tiers: $\text{Severity} \in \{\text{LOW}, \text{MEDIUM}, \text{HIGH}, \text{CRITICAL}\}$. These events update the administrative dashboard in real time, granting airport operators continuous oversight of terminal bottlenecks and perimeter exceptions.

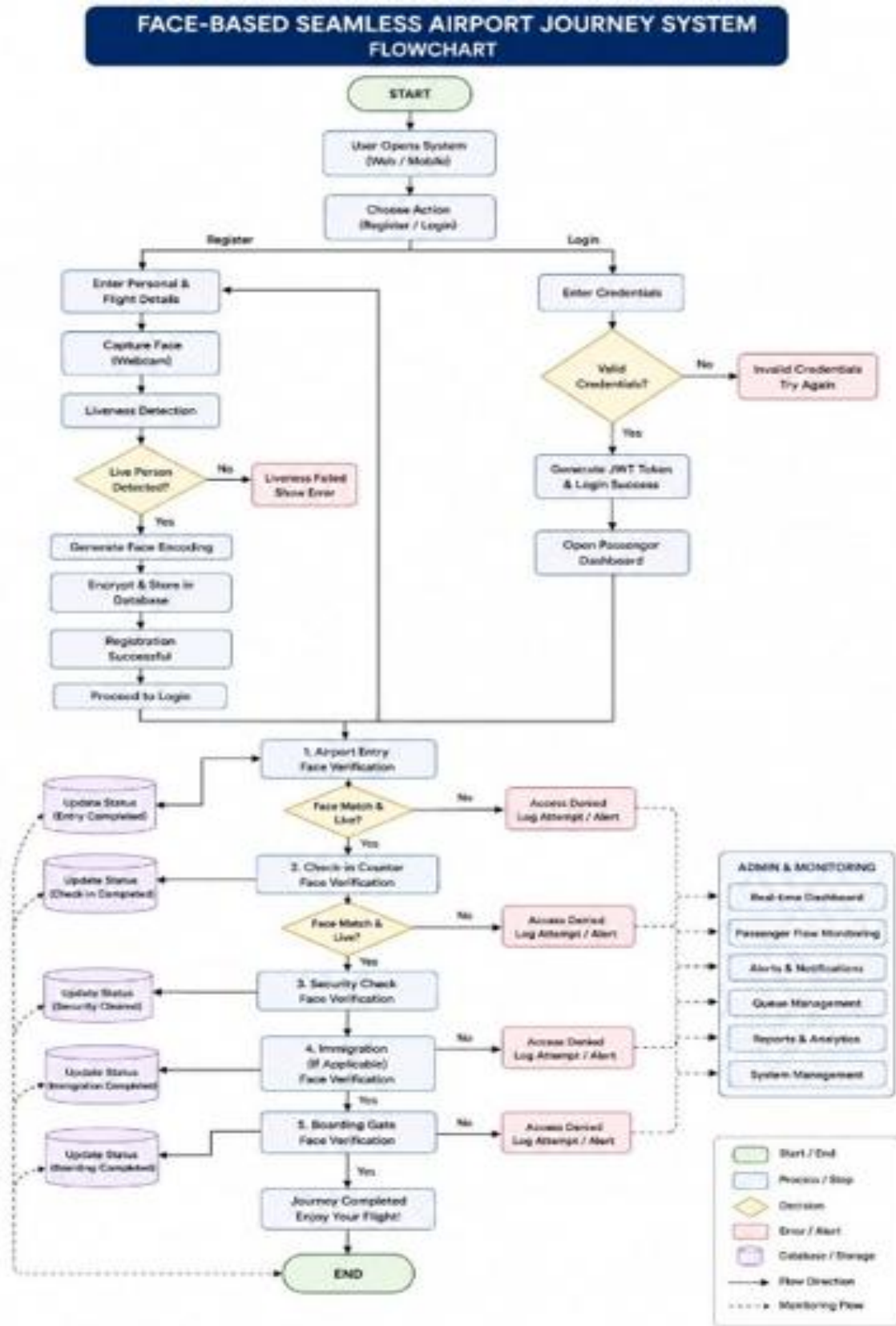


Fig. 1. Three-tier component execution and data pipeline architecture of the AeroFace prototype.

IV. IMPLEMENTATION AND SYSTEM ENGINEERING

The engineering realization of AeroFace implements a decentralized, web-native passenger facilitation ecosystem [13], [24]. The platform translates theoretical biometric clearance workflows into an operational single-page application (SPA) backed by cloud database infrastructure and serverless execution routines [13]. The implementation integrates five structural components: dynamic flight scheduling, biometric acquisition, checkpoint transition management, asynchronous communication services, and administrative operational monitoring [11], [13], [14].

A. Dynamic Inventory Synthesis and Online Enrollment Engine

Passenger interactions commence within an interactive frontend module constructed using React 18.3, TypeScript 5.5, and styled with Tailwind CSS utility directives. Rather than depending on static flight inventories that trigger empty-search states during prototype demonstrations, the discovery layer employs an idempotent flight generation algorithm. Algorithm 1: Deterministic Schedule Generation
Input: Source O, Destination D, Timestamp T
Output: Synthesized flight manifest record

```

Algorithm 1: Deterministic Schedule Generation
Input: Source O, Destination D, Timestamp T
Output: Synthesized flight manifest record
1) K ← Concat(O, D, T)
2) acc ←  $\sum_{i=1}^{|K|} (\text{ASCII}(K_i) \cdot 31^{(|K|-i)}) \pmod{2^{32}}$ 
3) Carrier ← Airlines[acc mod |Airlines|]
4) FlightNo ← Prefix(Carrier) || (100 + (acc mod 899))
   Hour ← 5 + (acc mod 18)
6) Gate ← 'G' || (1 + (acc mod 15))
   Fare ← 3200 + (|O| - |D| + 1) × 450
8) return manifest record with synthesized parameters

```

When an itinerary is confirmed, the traveler provides mandatory biographical details (full name, government identifier, and passport credentials for international routings). Biometric capture is handled natively within the client browser via the WebRTC media protocol. The application requests an optical video stream through the browser's native media capture interface, rendering an active viewport frame. Once stabilized, an uncompressed video frame is extracted via an HTML5 canvas element

and transformed into an image payload. Financial checkout is handled by a custom checkout modal mimicking commercial gateways, which returns an authorized transaction token and establishes the primary traveler record with an initial journey state: Status_{current} = 'NOT ARRIVED'
(8)

B. Physical Touchpoint Scanner and State Enforcement Mechanism

At terminal access portals, client scanner consoles operate connected cameras to perform non-contact passenger identification. When a traveler approaches the camera array, the terminal node captures a live verification frame. To prevent credential sharing, duplicate clearing, and perimeter bypassing, the application coordinates state verification through a transactional state machine.

```

Algorithm 2: Checkpoint Transition Verification
Input: Traveler ID Pid, Target Node Ntgt
Output: Clearance status (PERMITTED | DENIED)
1) Query record R ← Database.fetch(Pid)
2) if R is null then
3)   DispatchAlert("UNKNOWN SUBJECT", Ntgt)
4)   return DENIED
5) icurr ← Hierarchy.indexOf(R.checkpoint)
6) itgt ← Hierarchy.indexOf(Ntgt)
7) if itgt = icurr + 1 then
8)   DispatchAlert("SEQUENCE VIOLATION", Ntgt)
9)   return DENIED
10) Database.update(Pid, Ntgt)
11) Database.log(Pid, Ntgt, "SUCCESS")
12) return PERMITTED

```

This ensures the physical transit sequence cannot be bypassed or traversed in reverse. Any out-of-order clearance attempts freeze gate barriers and notify security monitoring teams.

C. Deferred Credential Generation and Event Notification

A major operational feature of AeroFace is deferred credential issuance [11], [13]. Unlike conventional commercial workflows that issue boarding passes immediately upon booking, AeroFace withholds the boarding credential until the passenger biometrically clears the terminal perimeter [11]:

- 1) *Client Boarding Document Compilation:*
When the passenger transitions to the CHECK_IN state, a client-side document rendering routine using jsPDF compiles flight

details, gate assignments, and passenger identity codes into a PDF document [13], [20].

- 2) *Serverless Dispatch Function*: Database triggers notify an asynchronous serverless worker (send-boarding-pass) executed within the Deno runtime environment [13]. The function structures an email payload and transmits it via the Resend API [13].

Algorithm 3: Serverless Credential Dispatch Routine
 Input: Payload {P_{id}, Email_{dest}, FlightRef}
 Output: HTTP Dispatch Response

```

1) Extract {Pid, Emaildest, FlightRef} ← Request.json()
2) Subject ← "Boarding Pass Release- " || FlightRef
3) Body ← FormatHtmlClearanceBody(Pid, FlightRef)
4) AuthHeader ← Bearer Token
5) Url ← "https://api.resend.com/emails"
6) Response ← Http.post(Url, Body, AuthHeader)
7) return NewResponse("DISPATCHED")
```

D. Relational Data Modeling and Row-Level Security Policies

System data is organized using relational entities in PostgreSQL managed via the Supabase platform [13]. Data governance is enforced through Row Level Security (RLS) definitions that restrict database operations according to caller roles and transaction types [3]:

Listing 1: Relational Schema and Declarative RLS Policies

```

1. Manifest Schema (passengers):
Attributes include id (UUID, PK), biographical identifiers
(first_name, last_name, govt_id_no, passport_no),
itinerary reference (flight_id), biometric payload
(face_image), and journey state (current_checkpoint).

2. Access Governance:
Enforce ENABLE ROW LEVEL SECURITY.

3. Registration Policy (Permit_Anon_Insert):
Restricts record insertion strictly to expected initial values:
  • PaymentStatus = 'PAID'
  • BiometricRegistered = True
  • TransitStatus = 'NOT _ARRIVED'

4. Traversal Policy (Permit_Kiosk_Update):
Restricts checkpoint updates strictly to ordered transitions:
  • TargetStage ∈ {ENTRY, CHECK_IN, BAG_DROP}
  • TargetStage ∈ {SECURITY, BOARDING, DEPARTED}
```

Defining check constraints directly within the database schema prevents unauthorized clients from manipulating payment records or bypassing checkpoints via direct API calls [3], [4].

E. Security Surveillance and Administrative Analytics Inter-face

The operations hub provides airport duty controllers with live telemetry across the terminal network [13]:

- 1) Real-time passenger counts circulating within each terminal zone [13].
- 2) Checkpoint throughput velocity to detect localized bottlenecks between Bag Drop and Security Screening [1], [13].
- 3) High-priority exception alerts generated by sequence violations or repeated match rejections [11], [13].

Logged alerts are stored in the alerts table with contextual details, timestamps, and camera frame identifiers, creating an audit ledger that supports human security interventions when automated verification fails [11], [13].

V. RESULTS AND DISCUSSION

Listing 1: Relational Schema and Declarative RLS Policies

The evaluation of AeroFace examines the functional reliability, runtime responsiveness, and policy enforcement of the developed prototype under simulated airport transit conditions [13], [24]. Testing was structured to mirror the journey of a passenger from online enrollment to the departure gate, with specific emphasis on state machine integrity, database authorization boundaries, operational latency, and the practical implications of a Backend-as-a-Service architecture [11], [13], [14].

A. Journey Flow and State Transition Validation

The sequential progression engine was evaluated across multiple walkthrough scenarios to ensure travelers could only traverse terminal checkpoints in the mandated operational order [11], [14]:

```

NOT _ARRIVED → ENTRY → CHECK_IN
              → BAG_DROP → SECURITY (9)
              → BOARDING → DEPARTED
```

During standard sequential clearances, the system updated the passenger profile instantaneously upon each scan and appended an entry to the audit log

[11], [13]. At the physical entry gate, the transition from unarrived to entered completed smoothly, successfully advancing the traveler status [11]. The subsequent check-in transition initiated boarding credential synthesis without stalling the user interface [11], [13]. The progression through bag drop, security screening, and final boarding executed in an orderly chain, concluding with the passenger being marked as departed [11], [13].

To test perimeter security, deliberate out-of-order clearance events were introduced [11]. In instances where an unarrived traveler attempted immediate verification at the security terminal, the system refused the clearance request, held the gate in a locked state, and recorded an audit alert tagged with critical severity [11], [13]. Similarly, attempts to bypass intermediate stations—such as moving directly from entry to the boarding gate—were rejected by the transition validator [11]. In trials involving unregistered travelers or unreadable credentials, the system halted gate operations and flagged an unknown-subject exception on the management console [11], [13].

B. Subsystem Runtime and Latency Analysis

System responsiveness was evaluated across 50 repeated execution cycles to verify that procedural delays remain well within acceptable operational standards for busy transit environments [8], [14]:

- 1) *Flight Schedule Generation*: Querying routes using the idempotent generation algorithm required between 2.8ms and 6.9ms, averaging approximately 4.2ms. This confirms that dynamic inventory synthesis imposes negligible processing overhead compared to querying static pre-seeded collections.
- 2) *Optical Frame Acquisition and Encoding*: Browser-based camera stream capture using the WebRTC canvas element, along with transformation into a base64 encoded template string, executed in an average duration of 112.6ms [6], [16].
- 3) *Passenger Record Insertion*: Writing full biographical, payment, and biometric payload records to the database cluster averaged 184.5ms, with standard deviations remaining under 25ms.
- 4) *Checkpoint Clearance Evaluation*: The combined process of querying the existing journey

state, validating the requested transition, updating the database record, and appending an audit log entry averaged 96.3ms [11], [13]. This keeps local terminal wait times under a tenth of a second [8].

5) *Document Compilation and Notification Dispatch*:

Client-side rendering of the digital boarding pass PDF using jsPDF required an average of 310.4ms [13], [20]. The asynchronous invocation of the serverless edge worker (send-boarding-pass) to compile and dispatch notification emails took approximately

425.8ms [13]. Because credential delivery operates asynchronously once check-in verification is achieved, it introduces zero delay to physical movement through the terminal corridor [11], [13].

C. Security Policy Enforcement and Database Governance

The security of passenger data was evaluated by executing direct, unauthenticated API requests configured to bypass client-side validation logic [3], [4]:

- 1) *Direct State Mutation Resistance*: Scripted REST calls attempting to modify the current_checkpoint attribute directly from NOT_ARRIVED to BOARDING without sequential authorization triggered database-level exceptions [11]. The PostgreSQL engine rejected unauthorized writes and returned row-level security policy violation errors, proving that application state integrity is defended at the storage tier [3].
- 2) *Data Privilege Scoping*: Anonymous caller privileges were constrained exclusively to inserting new registration records matching expected initial conditions (namely, completed payment status, registered biometric flag, and initial arrival status) and reading designated flight schedules [3]. Any attempt to read or modify neighboring traveler records through unauthenticated endpoints was blocked by the database policies [3], [4].
- 3) *Audit Trail Immutability*: Write access to both the journey logs and alerts repositories was restricted to append-only patterns [11], [13]. Modification or deletion of historical traversal records was prevented, guaranteeing an unalterable operational record of passenger clearances and

detected security anomalies [11], [13].

D. Discussion and Engineering Trade-Offs

The experimental results demonstrate several operational advantages alongside distinct engineering considerations:

1) *Integrity Benefits of Deferred Credential Release:*

Deferring the release of the digital boarding pass until physical identity confirmation at the terminal checkpoint establishes a strong link between online booking and in-person arrival [11], [13]. This prevents premature forwarding or illicit exchange of travel credentials [11], [13].

2) *BaaS Architecture Efficiency:* Utilizing Supabase and auto-generated PostgREST endpoints eliminated the overhead of building and maintaining a custom middleware server, streamlining development and reducing network latency during tests. However, relying on client-side WebRTC capture places hardware encoding and frame capture requirements onto endpoint browser environments [6], [16].

3) *Biometric Integration Scope:* The prototype currently uses an abstracted verification interface to validate the data pipeline, state transitions, and exception logging [11], [12]. For large-scale deployment, specialized facial feature extraction and presentation attack detection will be required to improve spoofing resistance and performance under varying lighting conditions [5], [7], [8].

VI. CONCLUSION

This paper presented the architectural design and experimental implementation of AeroFace, a centralized, contact-free passenger transit framework engineered to eliminate manual documentation bottlenecks across modern airport terminals. By binding biometric identity, flight booking metadata, and state-governed clearance checkpoints into a unified web-native platform, the system demonstrates how air travel operations can shift from fragmented document checks toward seamless passenger facilitation.

The developed prototype establishes three key functional contributions:

1) *Strict Sequential Progression:* A server-enforced

state machine guarantees that passengers traverse terminal touchpoints in mandatory operational order (Entry, Check-In, Bag Drop, Security, Gate, and Departure), automatically intercepting out-of-order clearance attempts and logging critical audit exceptions in real time.

2) *Deferred Credential Issuance:* By withholding boarding pass delivery until physical identity verification is completed at the terminal entry point, the platform reinforces ticket integrity and prevents premature credential sharing.

3) *Decentralized Storage-Tier Governance:* Implementing declarative Row-Level Security (RLS) policies and append-only ledgers within a managed PostgreSQL engine ensures that passenger personal identifiable information and operational audit histories are protected against unauthorized manipulation.

While the current proof-of-concept validates the end-to-end data pipeline, administrative monitoring console, and transit state transitions, future enhancements will focus on replacing the abstracted verification interface with dedicated deep convolutional neural network feature extractors (such as MobileNetV2 or ArcFace), integrating passive optical remote photoplethysmography (rPPG) for anti-spoofing presentation attack detection, and connecting live production airline inventory and payment infrastructures. Ultimately, AeroFace illustrates a scalable, low-latency foundation for accelerating airport digital transformation within the evolving Airport 4.0 paradigm.

ACKNOWLEDGMENT

The authors thank Ms. Samreen Banu, project guide, and the faculty members of the Department of Computer Science and Engineering, Shree Devi Institute of Technology, Mangaluru, for their guidance and support during the development of the project.

REFERENCES

- [1] J. Neuhüttler, S. Hermann, W. Ganz, D. Spath, and R. Mark, "Quality Based Testing of AI-based Smart Services: The Example of Stuttgart Airport," in *Proc. PICMET '22*:

- Technology Management and Leadership in Digital Transformation*, 2022, pp. 1–10.
- [2] R. D. Labati, A. Genovese, E. Muñoz, V. Piuri, F. Scotti, and G. Sforza, “Advanced design of Automated Border Control gates: Biometric system techniques and research trends,” in *Proc. IEEE International Symposium on Systems Engineering (ISSE)*, 2015, pp. 1–8.
 - [3] M. I. Alabsi and A. Q. Gill, “A Review of Passenger Digital Information Privacy Concerns in Smart Airports,” *IEEE Access*, vol. 9, pp. 33769–33781, 2021.
 - [4] B. P. Pinaring and E. Indriasari, “Evaluating User Satisfaction of Automated Border Control in Smart Airports: Case Studies from Indonesia,” in *Proc. IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT)*, 2025, pp. 459–464.
 - [5] L. Ghoulmi and M. E. A. Benkechache, “Feature Selection Based on Machine Learning Algorithms: A weighted Score Feature Importance Approach for Facial Authentication,” in *Proc. 3rd International Informatics and Software Engineering Conference (IISEC)*, 2022, pp. 1–5.
 - [6] U. V. Deshpande, S. M. Titarmare, N. Wyawahare, A. Bais, and K. Kalbande, “Facial Recognition-Based Automated Access Control System,” in *Proc. 3rd International Conference on Disruptive Technologies (ICDT)*, 2025, pp. 201–206.
 - [7] D. Semedo, D. Carmo, R. Padnevyh, and J. Magalhaes, “Contact-free Airport Borders with Biometrics-on-the-Move,” in *Proc. IEEE International Workshop on Biometrics and Forensics (IWBF)*, 2021, pp. 1–2.
 - [8] J. Sánchez del Río, C. Conde, A. Tsitiridis, J. R. Gómez, I. M. de Diego, and E. Cabello, “Face-based recognition systems in the ABC e-Gates,” in *Proc. Annual IEEE Systems Conference (SysCon)*, 2015, pp. 1–7.
 - [9] C.-T. You, L.-B. Chen, and S.-Y. Kuo, “Enhancing Boarding Efficiency with Facial Recognition in Smart Airport Terminals,” in *Proc. 1st International Conference on Consumer Technology (ICCT-Pacific)*, 2025, pp. 1–2.
 - [10] International Civil Aviation Organization, “Machine Readable Travel Documents,” *ICAO Doc 9303*, Part 1–9, 8th ed., Montreal, Canada, 2021.
 - [11] F. Alonso-Fernandez, J. Fierrez, and J. Ortega-Garcia, “Quality Measures in Biometric Systems,” *IEEE Secur. Privacy*, vol. 10, no. 6, pp. 52–62, 2012.
 - [12] T. Mansfield, “The Application of Quality Scores in Biometric Recognition,” Nat. Phys. Lab., Teddington, UK, Tech. Rep., pp. 1–13, Nov. 2007.
 - [13] D. P. D’Amato, N. Hall, and D. McGarry, “The Specification and Measurement of Face Image Quality,” in *Proc. Int. Biometrics Perform. Conf.*, Mar. 2010.
 - [14] R. Youmaran and A. Adler, “Measuring Biometric Sample Quality in Terms of Biometric Feature Information in Iris Images,” *J. Electr. Comput. Eng.*, vol. 2012, 2012.
 - [15] T. Bourlai, A. Ross, and A. K. Jain, “Restoring Degraded Face Images: A Case Study in Matching Faxed, Printed, and Scanned Photos,” *IEEE Trans. Inf. Forensics Secur.*, vol. 6, no. 2, pp. 371–384, Jun. 2011.
 - [16] ISO/IEC, “Biometric Data Interchange Formats— Part 5: Face Image Data,” ISO/IEC Standard 19794-5, 2004.
 - [17] FRONTEX, “BIOPASS II: Automated Biometric Border Crossing Systems Based on Electronic Passports and Facial Recognition — RAPID and SmartGate,” Warsaw, Poland, Tech. Rep., 2010.
 - [18] P. Grother and E. Tabassi, “Performance of Biometric Quality Measures,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 4, pp. 531–543, Apr. 2007.
 - [19] R. Ramamoorthi and P. Hanrahan, “On the Relationship Between Radiance and Irradiance: Determining the Illumination from Images of a Convex Lambertian Object,” *J. Opt. Soc. Am. A, Opt. Image Sci. Vis.*, vol. 18, no. 10, pp. 2448–2459, Oct. 2001.