Electronic Ticketing System for Monuments and Museums

Atharva Pimple[1], Karan Satpute[2],Raman Pethe[3], Rohan Sonvane[4], Prof. Amar Chadchankar[5]

Department of Computer Engineering
Zeal College of Engineering And Research Pune, INDIA

***Abstract*: With the advancement of digital technologies, the modernization of public services, especially in tourism and cultural heritage, has become essential. This paper introduces a mobile-based, cross-platform e-ticketing application for museums and monuments using Flutter and Firebase. The system is designed to tackle issues associated with conventional paper ticketing, including long queues, manual checks, ticket fraud, and environmental waste. By integrating features such as QR-based digital tickets, real-time ticket status updates, and secure cloud storage, the system enhances convenience, efficiency, and administrative control. Field deployment at selected sites demonstrated improved processing speed, increased ticket sales, and better sustainability. This research provides a scalable digital model for managing cultural site access across India.**

I. INTRODUCTION

India's rich cultural heritage draws countless visitors to its monuments and museums each year. However, many of these heritage sites still operate with traditional paper ticketing systems, which result in long entry queues, manual verification delays, ticket duplication, and unnecessary paper use. These challenges affect visitor satisfaction and reduce the efficiency of operations at such high-traffic locations.

With widespread mobile usage and internet access across India, especially in semi-urban and rural regions, there's a clear opportunity to digitize these systems. To address this, we developed a mobile-based ticketing system using Flutter for app development and Firebase for backend services. This system covers the complete ticketing cycle—from booking to entry—via QR code-based verification.

Flutter supports both Android and iOS, ensuring wide device compatibility, while Firebase handles secure, real-time data storage and user authentication. Our model intentionally excludes Aadhaar and biometric data to ensure accessibility in low-connectivity areas and protect user privacy.

Administrators use a companion app to scan and verify tickets at entry points, reducing errors and speeding up entry. Real-time updates help prevent ticket reuse and fraud. The system was tested at heritage sites in Pune and proved effective in improving operational efficiency and user experience. This paper discusses the system's development, deployment, and the benefits it offers for scalable use at cultural locations.

II. LITERATURE REVIEW

In recent years, electronic ticketing has seen growing implementation across public domains like transport, entertainment, and tourism. QR code-based systems are widely acknowledged for their advantages, including quicker entry validation, reduced environmental footprint, and streamlined operations—critical aspects for heritage sites that manage large visitor volumes.

Rathore et al. (2022) proposed an Aadhaar-linked QR e-ticketing system for Indian museums, which emphasized entry authentication but raised privacy and infrastructural concerns, especially in rural areas where connectivity and Aadhaar services may be unreliable.

Anand et al. (2023) explored facial recognition for museum entry, aiming to minimize ticket fraud and streamline entry. While innovative, this system required significant financial investment and introduced ethical concerns regarding facial data storage.

A system introduced by Shinde et al. (2023) featured QR codes combined with biometric validation. Though it helped automate processes and reduce manual labor, its dependency on internet access limited effectiveness in areas with low network availability, and biometric integration added complexity and cost.

Kazi et al. (2020) implemented a smart bus ticketing system using GPS and QR verification. Although designed for public transit, the approach demonstrated the role of automation and mobile access in enhancing administrative control and user ease.

Periša and Kavran (2018) compared QR code generators and highlighted the need for optimized QR generation in high-traffic systems like museums, emphasizing memory and speed efficiency.

Ghosal et al. (2015) developed an Android ticketing app for suburban rail with GPS-based ticket expiry. Their model enhanced verification and fraud prevention but required continuous internet and location services.

Unlike these systems that heavily relied on biometrics or required constant connectivity, our solution focuses on accessibility, security, and simplicity. By omitting Aadhaar and biometric verification, the system remains inclusive and deployable in a broader

range of settings. It balances security with practicality, making it a scalable model suitable for public cultural sites.

III. METHODOLOGY

The development of the mobile-based e-ticketing system for monuments and museums was approached through a comprehensive software engineering lifecycle. The methodology integrates both frontend and backend development, cloud service deployment, and rigorous testing protocols to ensure that the system is scalable, secure, and easy to use for both visitors and site administrators.

1. Requirements Gathering

The first phase involved identifying the functional and non-functional requirements from two user perspectives: visitors (end-users) and site administrators. Functional requirements included user registration, ticket booking, QR code generation, payment integration, real-time QR validation, and ticket status management. Non-functional requirements focused on usability, performance, security, and system scalability to handle high volumes during peak tourist seasons.

Interviews were conducted with stakeholders such as monument staff, tech-savvy users, and less digitally literate individuals to ensure that the final design would be inclusive and practical in a real-world heritage setting.

2. System Architecture and Design

A modular and layered architecture was chosen to separate the core functionalities: presentation layer (Flutter UI), application logic (controllers and services), and data management (Firebase Firestore). The application's architecture was designed to support:

Cross-platform compatibility via Flutter

Real-time data updates through Firebase Realtime Database and Firestore

Stateless and scalable cloud infrastructureLoose coupling between frontend and backend components to support future updates

A flowchart and data flow diagram (DFD) were developed to map the interaction between the user interface, backend processes, and admin validation points.

3. UI/UX Development

The UI was crafted with a focus on clarity and responsiveness. Screens were designed for the main app features: login/sign-up, ticket search, booking interface, payment page, and QR display. The design followed Material Design principles and adaptive layouts to ensure responsiveness across devices.

Special consideration was given to accessibility—buttons were made large enough for easy tapping, color contrast was tested for visibility, and language was kept simple to accommodate non-technical users.

4. Backend Implementation

Firebase served as the complete backend-as-a-service (BaaS) provider, offering:

Firebase Authentication: For secure user registration and session management using email and password.

Firestore Database: A scalable NoSQL database used to store user profiles, monument listings, ticket bookings, and QR metadata.

Firebase Functions (optional for future scaling): Potential backend logic for automated processes such as time-based ticket expiry.

The booking controller connects the frontend to Firestore, facilitating actions like ticket search, payment confirmation, and QR code storage.

5. QR Code Lifecycle

The core of the system lies in the lifecycle of a QR code. The steps include:

Upon payment (via Razorpay), the system generates a unique alphanumeric ticket ID.

The ID, along with user, monument, and visit details, is encoded into a QR format using the qr_flutter package.

The QR is stored in Firestore and displayed in the user's app.

At the monument gate, the QR is scanned using the admin's mobile app.

The system verifies the QR against the Firestore record and updates its status to "used" if valid, ensuring one-time entry only.

6. Admin Interface and Access Control

A dedicated admin panel (built with Flutter Web or React) allows monument staff to:

Log in securely

Scan QR codes

View entry logs and ticket history

Access analytics (visitor volume, peak hours, ticket type distribution)

Access control rules were implemented in Firestore to restrict unauthorized data manipulation and enforce role-based permissions.

7. Testing and Optimization

Extensive testing was performed at multiple stages:

Unit Testing: Ensured each module (e.g., booking, QR generation) worked independently.

Integration Testing: Checked data flow between frontend, backend, and cloud.

Load Testing: Simulated high traffic to test performance during holiday seasons.

Usability Testing: Assessed by users with different digital competencies to ensure the interface was intuitive.

Bug tracking tools and Firebase analytics were used to gather crash reports and usage data for further refinements.

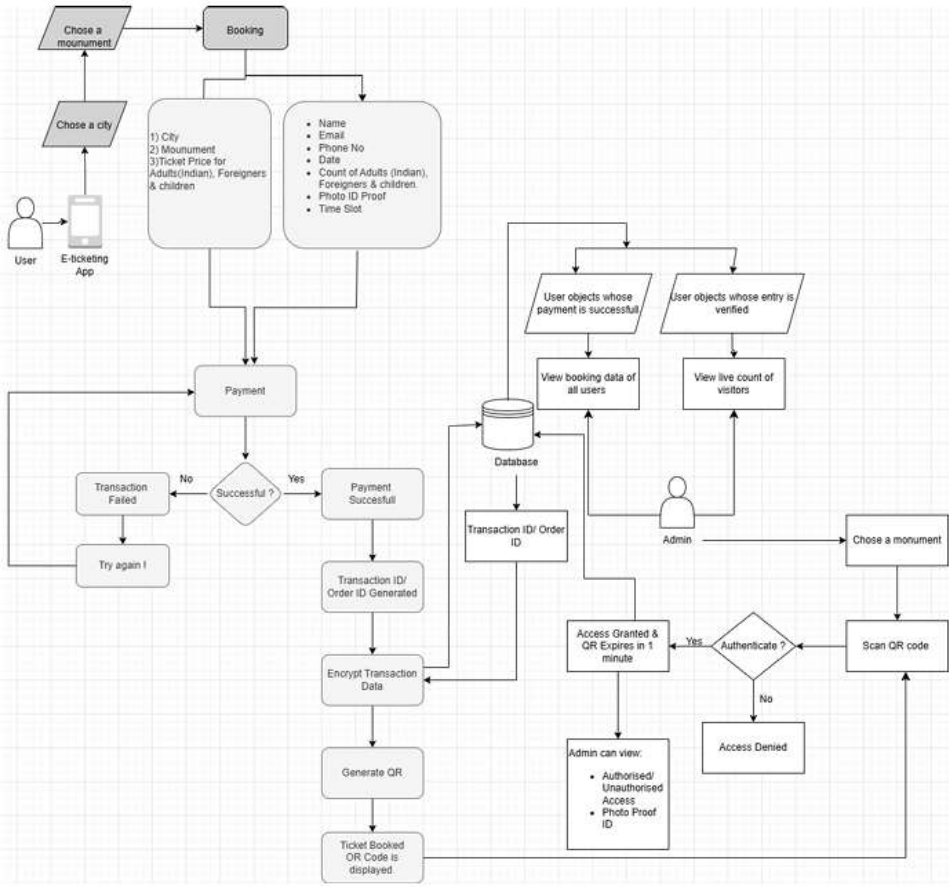
8. Deployment and Monitoring

The app was deployed for trial use at heritage sites in Pune. Firebase Hosting enabled fast deployment of the web admin dashboard. Firebase Analytics provided real-time insight into bookings, scan success rates, and peak hours, allowing administrators to respond quickly to on-site conditions.

Periodic updates and patches were pushed using Flutter's hot reload and Firebase's CI/CD integration for seamless feature rollouts.

System Design

4.1 SYSTEM ARCHITECTURE



IV. IMPLIMENTATION

The implementation phase translated the project's design into a fully functional application using Flutter for the frontend and Firebase for the backend. The architecture was structured for maintainability and performance, with a modular codebase and clearly defined responsibilities for each system component. The application was built in such a way that it could function reliably across various network conditions, platforms, and usage scenarios.

1. Frontend Development (Flutter)

The Flutter framework was used for developing the cross-platform application. Its widget-based architecture allowed for a responsive and dynamic user interface that adapted to both Android and iOS environments.

- **main.dart**: Served as the entry point, where Firebase was initialized, routing was configured, and navigation logic was set up for smooth transitions between screens.
- **Authentication screens**: Included login, registration, and password recovery interfaces that integrated with Firebase Authentication services.
- **Booking screens**: Allowed users to select monuments, dates, ticket categories, and view booking confirmation.
- **QR Code display**: After a successful booking, the generated QR code was displayed in the booking history for verification at the gate.

Flutter's state management was handled using Provider, ensuring efficient data updates and UI refreshes without unnecessary re-rendering.

2. Backend Integration (Firebase)

Firebase provided a scalable backend-as-a-service environment, eliminating the need for server maintenance and reducing development complexity.

- **Authentication**: Firebase Authentication enabled secure user login, registration, and session management.
- **Cloud Firestore**: This NoSQL database handled real-time storage and synchronization of users, bookings, and QR code metadata.
- **Firebase Storage** (optional): Provisioned for storing additional resources such as ticket PDFs or user-uploaded documents (if required in future phases).

The backend was organized into the following modules:

- **auth_service.dart**: Managed all authentication logic including sign-in, sign-up, logout, and password reset.
- **booking_controller.dart**: Managed ticket selection, payment processing (via Razorpay), booking confirmation, and QR generation.
- **qr_service.dart**: Encoded user and ticket data into a secure QR format using the qr_flutter package.
- **firestore_service.dart**: Encapsulated all Firestore read/write operations for users, bookings, and ticket statuses.

3. Admin-side Application

A separate Flutter Web app (or optionally a React-based interface) was developed for administrators and gatekeepers. It included the following core features:

- **QR Scanner Module**: Implemented using the mobile_scanner package to read and validate QR codes at the entry gate.
- **Booking Validation Logic**: Compared scanned QR data with Firestore records to check ticket validity, usage status, and visit date.
- **Status Update**: Automatically updated ticket status to "used" on successful validation, preventing duplicate usage.
- **Real-time Logs**: Provided visibility into entries processed, invalid scans, and ticketing history.

The admin panel ensured a quick and accurate verification process, helping reduce entry gate congestion during peak times.

4. QR Code Generation and Validation

The QR code acted as the digital equivalent of a ticket. Upon successful booking and payment, the system encoded the following data into a QR:

- Ticket ID
- User ID

- Monument ID
- Visit date and time
- Ticket type (Adult, Child, etc.)

QRs were stored in Firestore and displayed to the user in their app. During scanning, the QR content was decoded and checked against backend records to verify eligibility. This real-time validation process was critical in maintaining a secure, one-time-use system.

5. Payment Integration

Payments were integrated using the Razorpay SDK:

- Users were redirected to Razorpay's secure checkout during booking.
- Payment success or failure callbacks were handled in real-time.
- Only after a confirmed transaction was a booking marked as valid and a QR generated.

This ensured that only verified, paid bookings resulted in usable tickets.

6. Reusable Components and Clean Code

To maintain code readability and reduce redundancy:

- The ui_components/ directory housed reusable widgets like buttons, dialogs, QR tiles, and ticket cards.
- All widgets followed Material Design guidelines and supported responsive behavior for different screen sizes and orientations.

7. Firebase Collections and Structure

The Firestore database was structured as follows:

- **users**: Stores user profiles, account info, and roles.
- **bookings**: Contains booking metadata, QR strings, payment status, and visit history.
- **sites**: Lists all available monuments/museums with details like timing, location, and ticket pricing.

Data access was governed by Firestore security rules, ensuring privacy and role-based permissions for both users and admins.

8. Deployment and Monitoring

The final version of the application was deployed in a live environment at select heritage locations in Pune. Firebase Hosting was used for deploying the admin dashboard, and Crashlytics and Firebase Analytics were activated to monitor app usage, crash reports, and user engagement.

Admins could access:

- Real-time visitor analytics
- Ticket scan rates
- Hourly traffic trends
- Alerts for repeated scan attempts or expired QR codes

V. ALGORITHM DETAILS

The efficiency and reliability of the e-ticketing system largely depend on its underlying algorithms for QR code generation, validation, payment confirmation, and real-time data synchronization. While the system does not use complex AI or machine learning models, its logic is optimized for responsiveness, minimal latency, and high accuracy in a dynamic environment such as public monument entry points.

1. QR Code Generation Algorithm

The generation of QR codes is a crucial part of the ticketing process. The algorithm performs the following steps:

- **Input Collection**: Upon successful booking and payment, the system gathers necessary details including:
    - Ticket ID (auto-generated)
    - User ID

    - Monument ID
    - Date and Time of Visit
    - Ticket Type (Adult, Child, etc.)
- **Data Structuring**: These values are serialized into a standardized string format (e.g., JSON or delimited string).
- **Optional Encryption**: For additional security, the payload can be encrypted using a symmetric encryption method such as AES or encoded using Base64. This ensures that ticket data cannot be easily read or altered if intercepted.
- **QR Code Creation**: The structured (and optionally encrypted) string is passed to the qr_flutter package, which renders it into a scannable QR code image.
- **Storage**:
    - Displayed in the user's mobile app under their booking history.
    - Persisted in Firestore under the corresponding booking document.

This one-time QR code is tightly bound to a single booking, preventing reuse and ensuring traceability.

2. QR Code Validation Algorithm

At the monument entrance, the QR code validation algorithm ensures ticket authenticity and checks usage conditions:

- **Scan Trigger**: When the QR code is scanned using the admin-side mobile application, its content is immediately decoded.
- **Backend Check**:
    - Retrieve the record from Firestore using the embedded Ticket ID.
    - Validate that:
        - The ticket exists.
        - The date of visit matches the current date.
        - The ticket has not already been used.
- **Status Update**:
    - If valid, the ticket's status in the Firestore database is updated to "used."
    - A confirmation message is shown on the admin's screen.
    - If any condition fails, access is denied and a relevant error is displayed (e.g., "Already Used" or "Invalid Ticket").

This algorithm ensures that every QR ticket is single-use and prevents unauthorized entries.

3. Payment Confirmation Workflow

To ensure that QR codes are only generated after successful payment, the system uses Razorpay's secure payment gateway with webhook-based callbacks:

- **Transaction Request**: When the user proceeds to pay, a transaction request is created with metadata like ticket cost, user email, and booking details.
- **Razorpay Checkout**: The user is redirected to the Razorpay interface for secure transaction processing.
- **Callback Verification**:
    - Upon payment success, Razorpay sends a callback to the app.
    - The app verifies the signature of the transaction and marks the booking as "paid."
    - The QR generation process is triggered immediately afterward.

This ensures that only verified users who have successfully completed the transaction receive an active QR ticket.

4. Real-Time Data Synchronization Algorithm

To maintain consistency across devices and interfaces (user app, admin app, and web dashboard), real-time synchronization is handled using Firebase Firestore:

- **Listeners and Snapshots**:
    - Both user and admin apps use snapshot listeners to automatically fetch and reflect data changes (e.g., a new booking or ticket used).
    - When a user books a ticket or an admin scans a QR code, all connected interfaces are updated in real-time without manual refresh.
- **Atomic Writes and Transactions**:
    - Firestore supports batched writes and transactions, ensuring that no inconsistent state is created (e.g., a ticket marked as "used" before it is generated).
- **Role-Based Access Control**:
    - Firestore security rules ensure that only users can read/write their bookings.
    - Admins can view only relevant visitor data and cannot alter user-specific data unrelated to ticket validation.

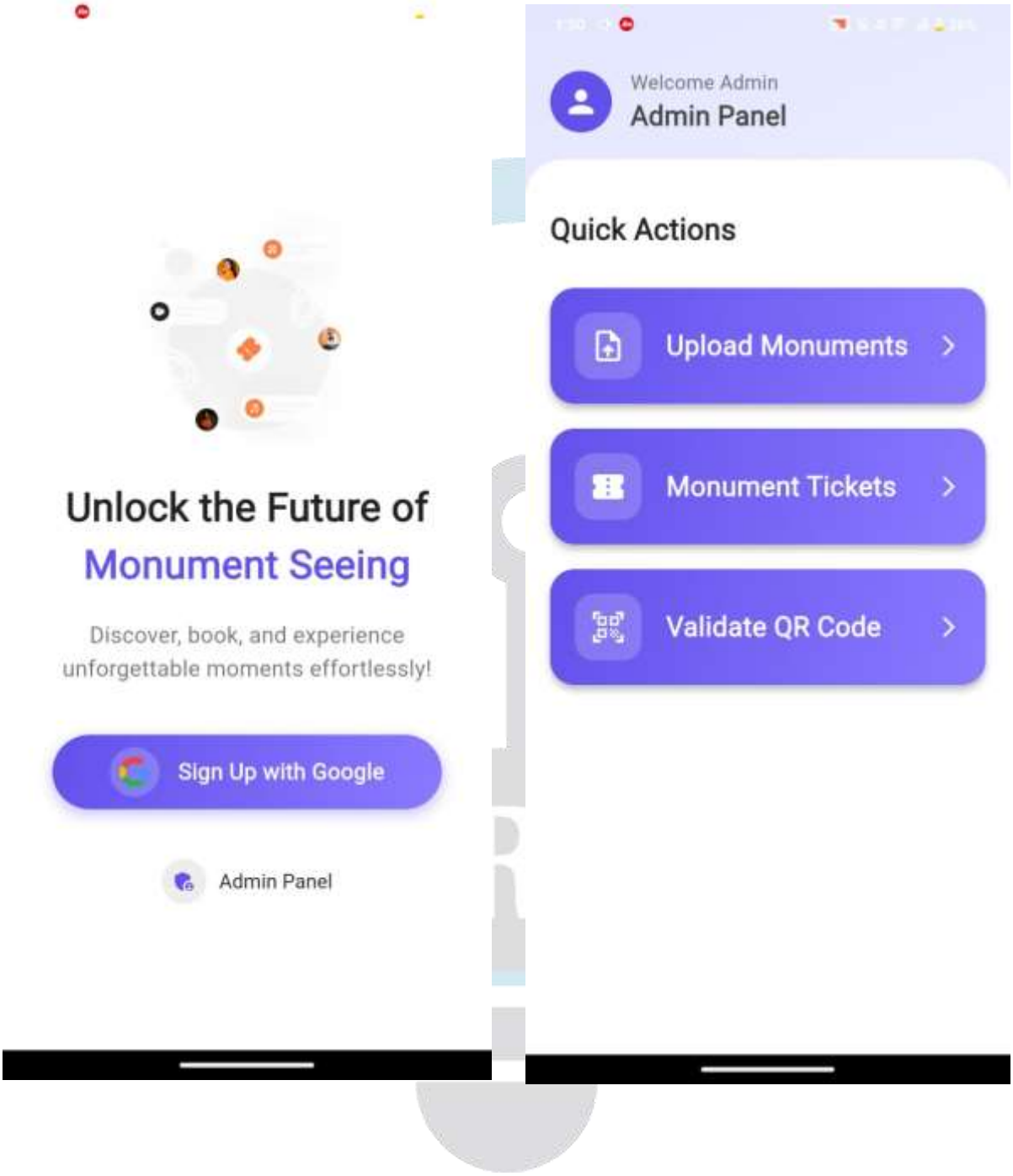
5. Data Security and Integrity

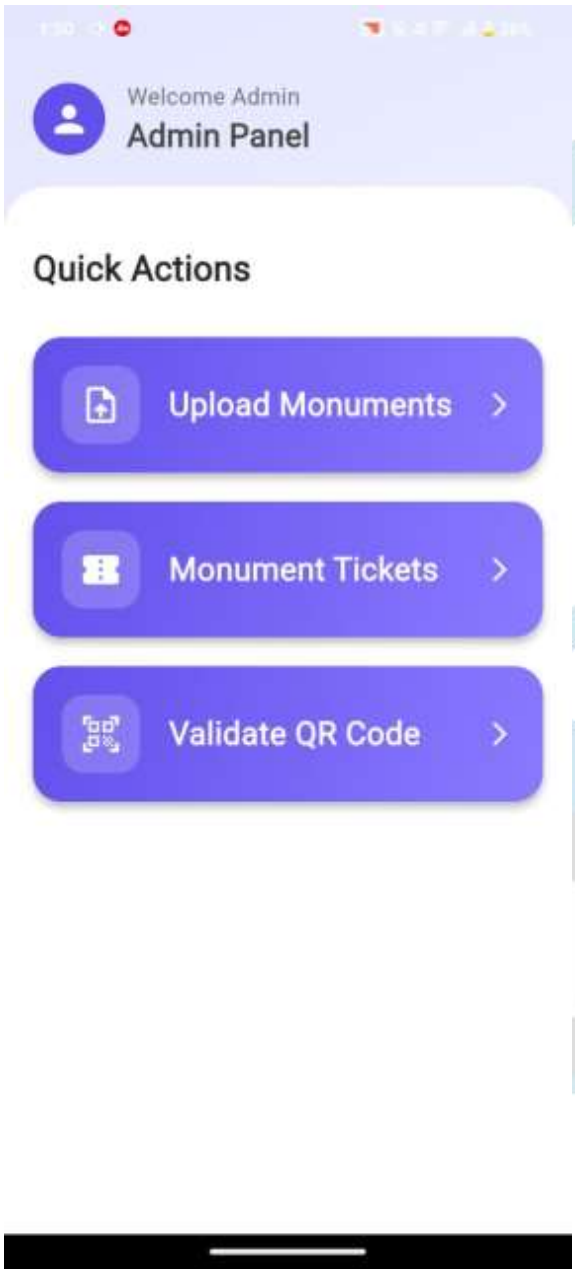
To ensure that data is handled securely at every step, the system includes:

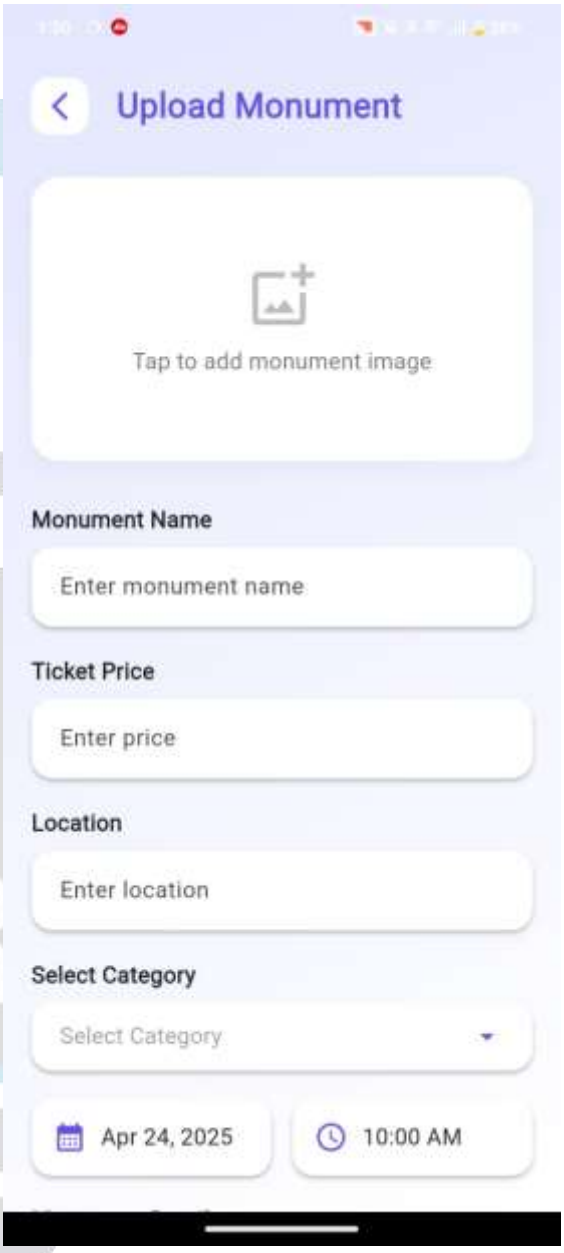
- **Firebase Authentication** for user login and identity management.
- **Encrypted QR Payloads** (optional) to secure ticket information.
- **Access Rules** defined in Firestore to restrict unauthorized access.
- **HTTPS Connections** for all network communications.

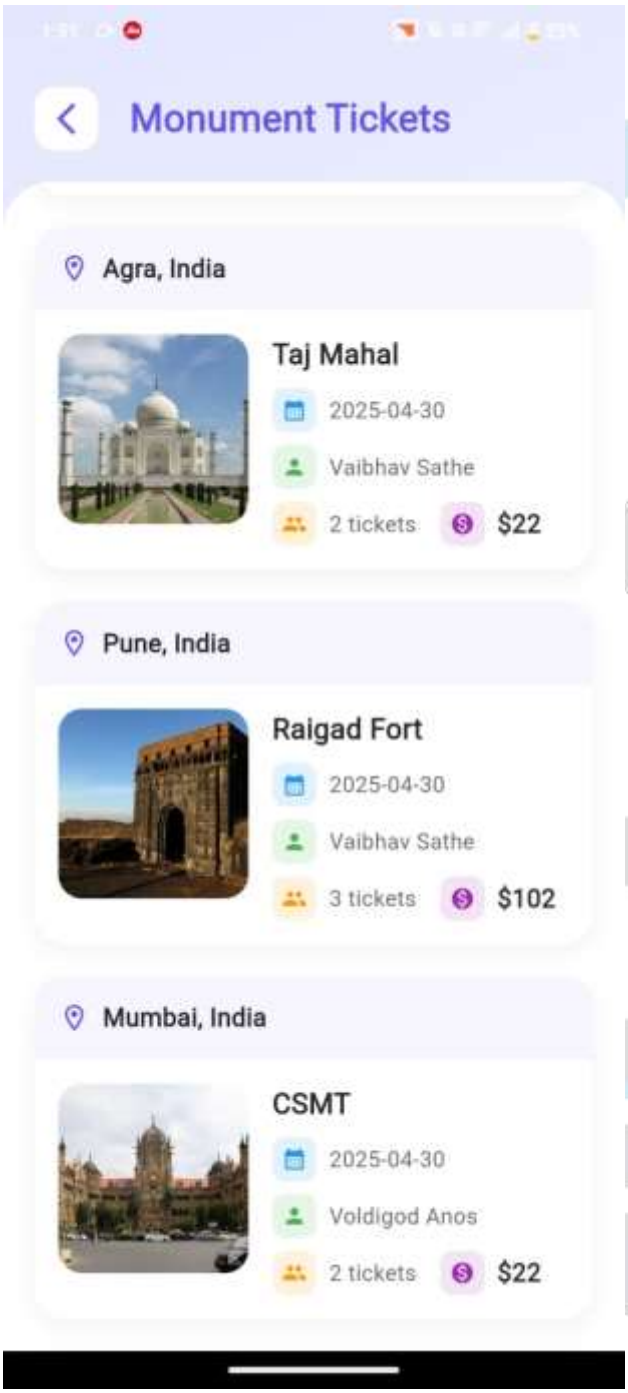
These safeguards ensure that the system adheres to basic data protection standards while offering a seamless and secure experience

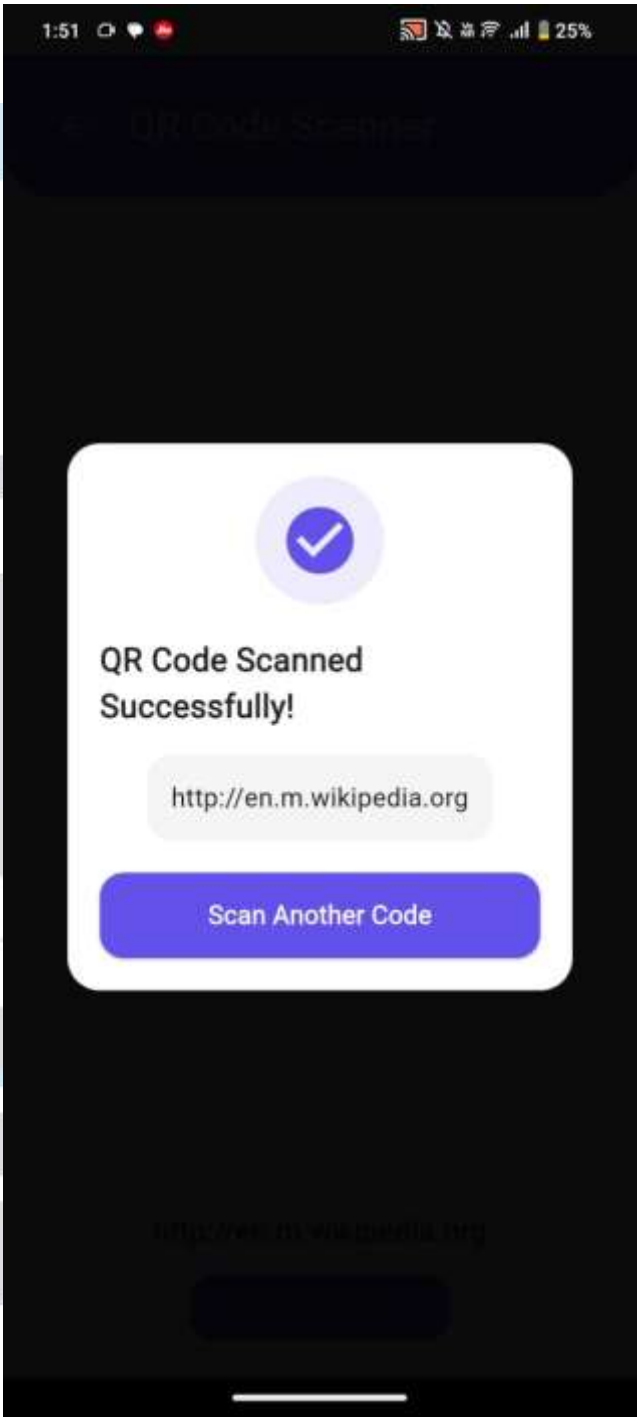
VI. SNAPSHOTS OF SYSTEM WORKING

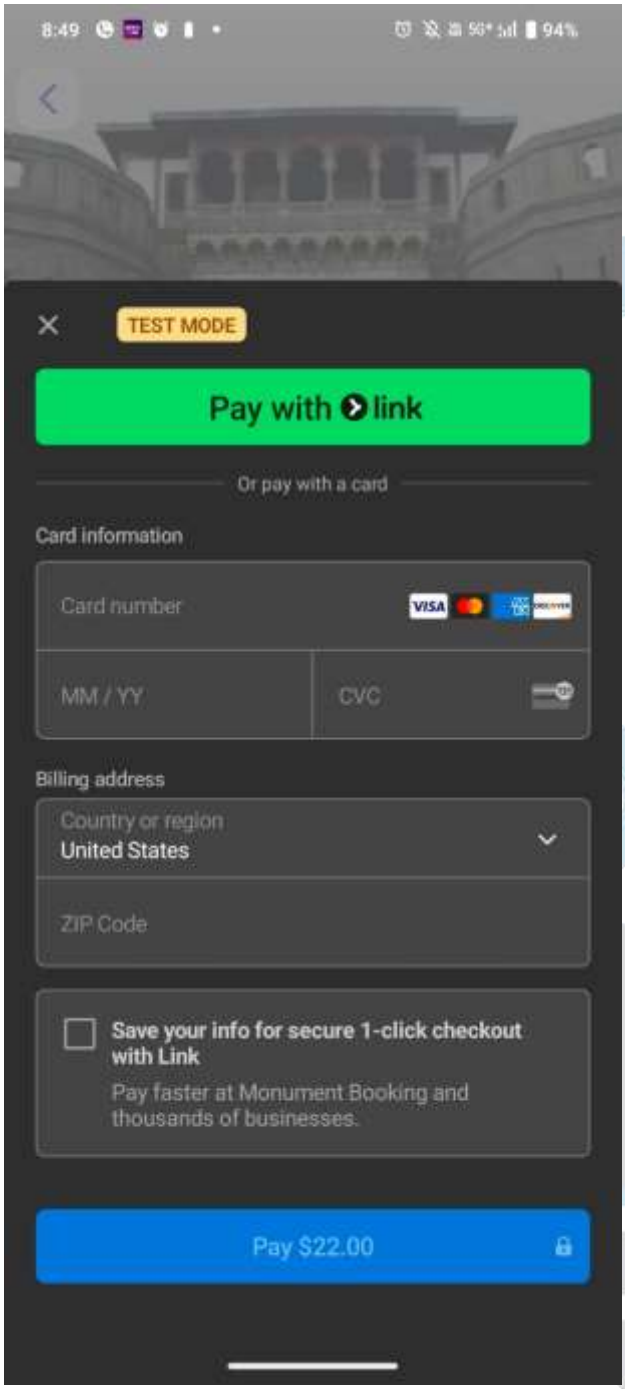


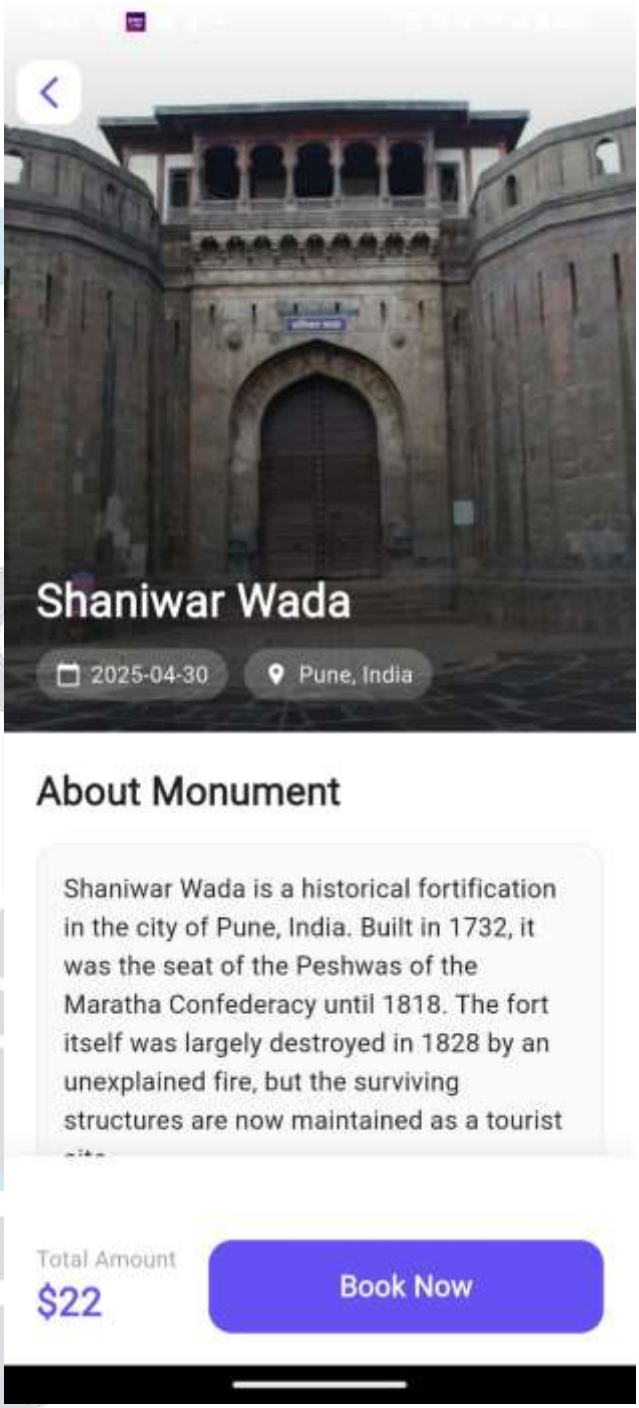

Welcome Admin
Admin Panel
Quick Actions
Upload Monuments
Monument Tickets
Validate QR Code


Upload Monument
Tap to add monument image
Monument Name
Enter monument name
Ticket Price
Enter price
Location
Enter location
Select Category
Select Category
Apr 24, 2025
10:00 AM


Monument Tickets
Agra, India
Taj Mahal
2025-04-30
Vaibhav Sathe
2 tickets
$22
Pune, India
Raigad Fort
2025-04-30
Vaibhav Sathe
3 tickets
$102
Mumbai, India
CSMT
2025-04-30
Voldigod Anos
2 tickets
$22


1:51
25%
QR Code Scanned Successfully!
http://en.m.wikipedia.org
Scan Another Code


8:49
94%
TEST MODE
Pay with link
Or pay with a card
Card information
Card number
MM / YY
CVC
Billing address
Country or region
United States
ZIP Code
Save your info for secure 1-click checkout with Link
Pay faster at Monument Booking and thousands of businesses.
Pay $22.00


Shaniwar Wada
2025-04-30
Pune, India
About Monument
Shaniwar Wada is a historical fortification in the city of Pune, India. Built in 1732, it was the seat of the Peshwas of the Maratha Confederacy until 1818. The fort itself was largely destroyed in 1828 by an unexplained fire, but the surviving structures are now maintained as a tourist
Total Amount
$22
Book Now

VII. ACKNOWLEDGMENT

I would like to take this opportunity to express my sincere gratitude to everyone who supported and guided me throughout the development of our project, *"E-Ticketing System for Monuments and Museums."*

First and foremost, I extend my deepest appreciation to our project guide, **Prof. Amar Chadchankar**, for his continuous encouragement, expert insights, and valuable feedback at every stage of this project. His deep understanding of software architecture and real-world applications was instrumental in shaping the success of this system.

I am also thankful to the **Department of Computer Engineering, Zeal College of Engineering and Research, Pune**, for providing us with the necessary resources, tools, and a collaborative environment that encouraged learning and innovation. Special thanks to our faculty members and peers for their valuable inputs, motivation, and technical suggestions, which played a significant role in refining our application.

We also acknowledge the open-source technologies and platforms such as **Flutter**, **Firebase**, and **Razorpay**, which made it possible to implement this system efficiently and securely. These tools helped us deliver a robust and scalable digital solution without the burden of infrastructure-heavy development.

Lastly, I am extremely grateful to my family and friends for their unwavering support, patience, and encouragement during this journey. Their constant belief in me helped me stay focused and committed, even during challenging times.

This project has been a meaningful and rewarding experience, blending technology with real-world social impact, and I am genuinely thankful to everyone who contributed to its realization..

VIII. CONCLUSIONS

The growing demand for digital transformation across public services has underscored the need for smarter, more efficient systems—especially in the domain of tourism and cultural heritage management. Monuments and museums often deal with high volumes of visitors, which can lead to overcrowded entry points, ticket fraud, and operational delays when traditional paper-based systems are used. Addressing these challenges, our project introduced a cross-platform mobile-based e-ticketing system, designed using Flutter and backed by Firebase, that leverages QR code technology for fast, secure, and paperless entry.

Throughout the design and implementation process, we prioritized accessibility, efficiency, and scalability. By removing the dependence on biometric and Aadhaar-based authentication, we ensured the system is inclusive and easier to deploy across different regions without the need for advanced infrastructure. Our approach focused on secure QR code generation, real-time validation, and seamless cloud integration, enabling a reliable system that both users and administrators could depend on.

Real-world testing demonstrated the system's effectiveness in improving visitor flow, reducing queue times, eliminating ticket reuse, and lowering the environmental impact by cutting down on paper usage. Moreover, administrators benefitted from real-time visitor analytics, enabling them to manage crowds and resources more efficiently.

In conclusion, this e-ticketing system presents a robust, scalable, and eco-friendly solution that meets the modern demands of heritage site management. It bridges the gap between technology and tourism by enhancing user experience while preserving cultural accessibility. This project lays the foundation for future enhancements such as multilingual support, offline ticketing capabilities, and AI-driven visitor predictions, ensuring that monuments and museums remain both digitally empowered and visitor-friendly.

VIIII. REFERENCES

1. Rathore, Aarohi, Gupta, Aayush, Gour, Abhay, & Nagar, Ankur. (2022). E-Ticketing System for Indian Museums & Heritage Sites. International Research Journal of Modernization in Engineering, Technology, and Science, 4(11), 512–515.

2. Anand, Gautam, Suthar, Jatin, Jain, Harshal, Minda, Devanshi, Dalsaniya, Nandani, & Kaushal, Jyoti. (2023). Ticketless Entry in Heritage Museums. International Advanced Research Journal in Science, Engineering and Technology, 10(Special Issue 2), 112–115.

3. Shinde, Sadashiv, Khule, Satyam, Abuj, Akash, & Janbhare, Aniket. (2023). Ticketless Entry in Heritage Museums. International Journal of Advanced Research in Science, Communication, and Technology, 4(5), 271–276.

4. Kazi, Sanam, Bagasrawala, Murtuza, Shaikh, Farheen, & Sayyed, Anamta. (2020). Smart E- Ticketing System for Public Transport Bus. IEEE International Conference on e-Education, Entertainment, and e-Management, 2(4), 89–94.

5. Sonkusale, Ankita, Chatap, Rashmi, Lulania, Sana, & Pande, Bhavana. (2018). Android Smart Ticketing System Using QR-code. International Journal of Scientific Research in Science and Technology, 6(3), 154–159.

6. Periša, Dino, & Kavran, Krešimir. (2018). Comparative Analysis of QR Code Generators. MIPRO 2018, 1(3), 321–328.

7. Ghosal, Subarnarekha, Chaturvedi, Shalini, Taywade, Akshay, & Jaisankar, N. (2015). Android Application for Ticket Booking and Ticket Checking in Suburban Railways. Indian Journal of Science and Technology, 8(S2), 171–178.

8. Chatterjee, Parag, & Nath, Ashoke. (2014). Smart Computing Applications in Railway Systems - A Case Study in Indian Railways Passenger Reservation System. International Journal of Advanced Trends in Computer Science and Engineering, 3(4), 276–284.

9. Kamalesh, VN, Ravindra, Vikram, Bomble, Pradeep P., MP, Pavan, BK, Chandan, & Srivatsa, S. K. (2011). Virtual Ticketing System: A Green Alternative. IEEE International Conference on e- Education, Entertainment, and e-Management, 1(4), 233–238.