APPLIED INFORMATION TECHNOLOGY AND COMPUTER SCIENCE
e-ISSN: 2773-5141

AITCS

Vol. 6 No. 1 (2025) 644-662
https://publisher.uthm.edu.my/periodicals/index.php/aitcs

A Ticket Booking and Verification App for the Al-Wehda Stadium using Facial Recognition Technology

Abdulqader Abdulrahman Ali[1], Shamsul Kamal Ahmad Khalid[1*]

[1] *Fakulti Sains Komputer dan Teknologi Maklumat*
Universiti Tun Hussein Onn Malaysia, Parit Raja, Batu Pahat, 86400, MALAYSIA

*Corresponding Author: feresa@uthm.edu.my
DOI: https://doi.org/10.30880/aitcs.2025.06.01.036

Article Info

Received: 16 February 2025
Accepted: 19 June 2025
Available online: 30 June 2025

Keywords

Facial Recognition, Raspberry pi, Local Authentication, One-Time Password, Booking, Stadium Management System, Security

Abstract

Stadium entry management, especially during major events like football matches, grapples with significant challenges amid rapid technological advancement. Traditional ticket presentation methods prove inefficient and prone to security breaches, such as fake tickets, e-tickets, and unauthorized usage. This project addresses these issues by developing a booking mobile application integrated with facial recognition technology, ensuring smooth and secure stadium entry. Utilizing Flutter for the interface, Firebase for data management, and Raspberry Pi for facial recognition, the application will bolster security with features like OTP and local authentication, reducing the risk of unauthorized entry. Several functional and security tests will be conducted with regular stadium-goers. By enabling ticket booking and stadium entry via facial recognition, the app promises a seamless and secure experience, elevating the stadium's status as a technology-driven event management leader.

1. Introduction

In today's digital age, the need for secure and efficient stadium entry systems has become essential. Traditional entry methods, such as physical or electronic tickets, face significant challenges, including counterfeit tickets, while e-tickets are vulnerable to unauthorized transfers, multiple uses due to system loopholes, and hacking attempts, and long queues that compromise both security and attendee experience. The urgency for a better system is highlighted by incidents like the Kanjuruhan disaster [1]. Emphasizing the need for advanced and reliable stadium management solutions.

This project introduces The Al-Wehda Stadium using facial recognition technology to tackle these issues. By leveraging a mobile application, the system securely verifies attendee identities, preventing unauthorized access and reducing entry delays. It incorporates advanced features such as facial recognition powered by Raspberry Pi, a Flutter-based interface, Firebase for data management, and additional security layers like One-Time Passwords (OTP) and local authentication.

The system targets two user groups: stadium attendees and administrators. Fans can register, book tickets, and gain entry through facial recognition, while administrators oversee operations like user management and event monitoring. The project is designed to enhance operational efficiency, improve security, and deliver a seamless entry experience.

Expected outcomes include reduced risks of unauthorized entry, faster access, and increased customer satisfaction. This project establishes a new standard for secure and efficient event management, transforming the stadium experience for attendees and organizers alike.

2. Literature Review

This section reviews relevant literature on The Al-Wehda Stadium System, focusing on facial recognition, OTP, local authentication, and comparisons with existing systems.

2.1 Facial Recognition Technology

Facial recognition identifies individuals by analyzing unique facial features through detection, alignment, encoding, and matching. Face detection isolates facial regions using methods like HOG to simplify brightness gradients into key facial features. Face alignment ensures consistent positioning through landmark estimation and transformations[2]. Encoding converts features into numerical embeddings, which are matched against a database using classifiers like SVMs for quick and accurate identification[3]. This process is efficient, scalable, and widely used in security and access control systems.

2.2 Security Features

The stadium management system incorporates advanced security features, including multi-factor authentication (MFA), one-time passwords (OTPs), and local authentication, to ensure robust protection against unauthorized access. MFA combines passwords, OTPs, and biometrics to enhance security. OTPs provide dynamic, single-use codes sent to the User's email, reducing the risk of reuse or interception. Local authentication enables offline verification using biometric data stored securely on the device, ensuring access even without an internet connection.

2.3 One-Time Password (OTP)

The OTP feature enhances application security by generating unique, time-sensitive codes sent to a user's registered email. Users enter the code in the application and communicate with a backend server for verification. If the input matches, the user is authenticated and granted access[4]. OTPs prevent unauthorized access by ensuring only users with access to the registered phone or email can complete authentication, adding an extra layer of protection.

2.4 Local Authentication

Local authentication secures access through biometric methods like fingerprint and face recognition, enabling reliable verification directly on the device. Using tools like Flutter's Face Recognition API, the application facilitates seamless biometric authentication via the device's front camera or fingerprint scanner[5]. This method ensures data security by eliminating reliance on external networks, providing a secure and user-friendly experience for both developers and users. Flutter's robust support for local authentication enhances the overall security and usability of the system.

2.5 Study of Related System

In this section, an examination of several existing systems related to authentication features security has been conducted with the aim of studying their features and functionalities, and selected systems have been implemented.

2.5.1 Existing System (Facial Recognition of the Osasuna Stadium System)

The Osasuna Stadium system enhances fan experience and security by integrating advanced technologies for seamless access. Developed using Kotlin for Android, Swift for iOS, and React Native for cross-platform compatibility, the mobile application allows fans to book match tickets and register with a password. Upon arrival, fans present a QR ID along with a selfie capture, processed through Python libraries compatible with OpenCV and TensorFlow, ensuring secure access. At the stadium, access control devices resemble turnstiles verify the fan's identity, granting entry upon successful authentication, as indicated by a green check mark on the device's screen. This system not only streamlines the entry process but also significantly reduces waiting times, enhancing overall fan satisfaction. By leveraging biometric verification, the system ensures that only authorized individuals gain entry, thereby bolstering security measures within the venue. Fig. 1 shows the booking interface of the Osasuna App, allowing fans to select and purchase match tickets seamlessly, while Fig. 2 illustrates the QR ID capture process with selfie verification, demonstrating the app's user-friendly approach to secure stadium entry[6].

Fig. 1 *Booking Interface of the Osasuna App*

Fig. 2 *QR ID Capture with Selfie*

2.5.2 Adrar Stadium System (Paper Tickets)

At Adrar Stadium, the ticketing system relies entirely on physical tickets, which are tangible paper documents containing critical information like seat numbers and barcodes to guide spectators to their assigned seats. These tickets are available either at the stadium's box office or through authorized vendors. While straightforward, the physical ticketing system presents several challenges. Printing and distribution incur logistical costs, and tickets are vulnerable to loss, theft, or damage, causing inconvenience for spectators. Additionally, physical tickets lack flexibility for last-minute changes or transfers and contribute to environmental concerns due to paper waste. Counterfeiting is a significant issue, exacerbating crowding and delays at entry points. Fig. 3 illustrates the traditional booking and ticketing process at Adrar Stadium in Morocco, highlighting the inefficiencies and security concerns inherent in this outdated system.

Fig.3 *The traditional way to book tickets at Adrar Stadium*

2.5.3 Local Authentication of Allianz Arena System

The Allianz Arena stadium in Germany offers a mobile application that enables fans to reserve match tickets while incorporating local authentication for enhanced security. This feature allows users to log in securely by providing biometric data, such as fingerprints or facial recognition, which is stored in the application's database. Fig. 4 illustrates the use of local authentication, where fans can conveniently access their myFCB accounts using Face ID or fingerprint scanning, ensuring secure and easy login. While local authentication enhances login security, the system still faces challenges in preventing the forgery of electronic tickets and unauthorized resale on the black market, especially for large events. These vulnerabilities highlight the need for additional measures to ensure secure and efficient entry management.

Applied Information Technology and Computer Science Vol. 6 No. 1 (2025) p. 644-662

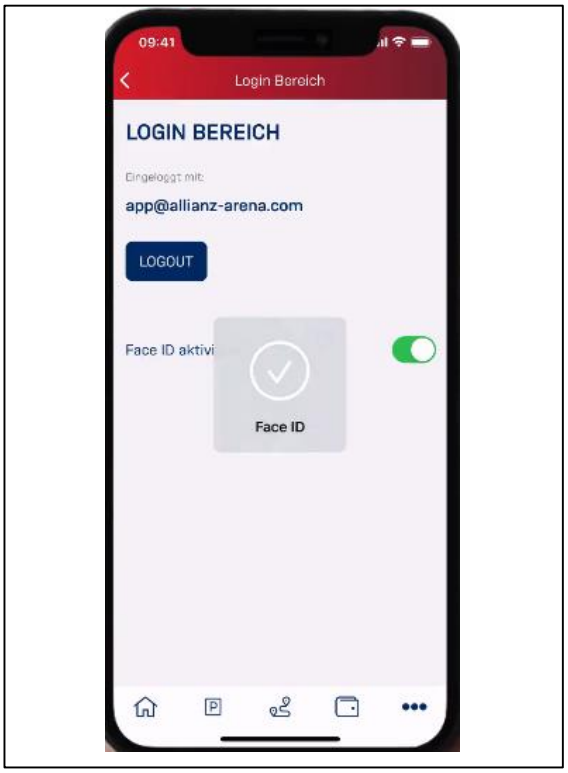


Fig. 4 *Secure Login via Face ID on Allianz Arena App*

2.5.4 Comparison Table with proposed system

Table 1 presents a comparison between the existing systems and the proposed system, evaluating their features and authentication mechanisms. The Osasuna System, Allianz Arena System, and the proposed system utilize app-based platforms, with React Native and Flutter for cross-platform compatibility, while the Adrar Stadium System relies on physical ticketing. For biometric security, the Osasuna System and the proposed system employ facial recognition, whereas the Allianz Arena System uses e-tickets, and the Adrar Stadium System depends on printed tickets. Data storage methods differ, with the Osasuna System utilizing SQLite, the Adrar Stadium System relying on physical ticket storage, and both the Allianz Arena System and the proposed system using Google Firebase for secure data management. While login mechanisms exist in the Osasuna System, Allianz Arena System, and the proposed system, only the proposed system incorporates password validation alongside OTP authentication for added security. Local authentication, missing in both the Osasuna System and Adrar Stadium System, is available in the Allianz Arena System and the proposed system, enhancing user convenience and security. The proposed system integrates advanced technologies and robust authentication methods, offering a comprehensive and secure stadium access solution.

Table 1 *Comparison table*

Criteria	Osasuna System	Adrar stadium System	Allianz Arena System	Alwehdah System
Platform	App-based React Native	Not Applicable	App-based Flutter	App-based Flutter
Type of biometrics	Facial Recognition	None	None	Facial Recognition
Data Storage	SQLite	Printed Tickets	Google Firebase	Google Firebase
Login	Yes	No	Yes	Yes
Password Validation	Yes	No	Yes	Yes
Local Authentication for the app	No	No	Yes	Yes

Login	Yes	No	Yes	Yes

Applied Information Technology and Computer Science Vol. 6 No. 1 (2025) p. 644-662

3. Methodology

Agile software development is a flexible approach emphasizing adaptability, collaboration, and customer satisfaction. Unlike traditional linear models, it divides projects into iterative cycles, covering planning, requirements analysis, design, development, testing, and deployment. This section outlines the methodology used for developing the Stadium Management System with Facial Recognition Technology, focusing on the Agile model's six structured phases. Each phase is detailed in subsequent sections, with a summary of the overall workflow. Fig. 5 shows the phases within Agile methodology, demonstrating how its iterative nature ensures responsiveness and continuous improvement.

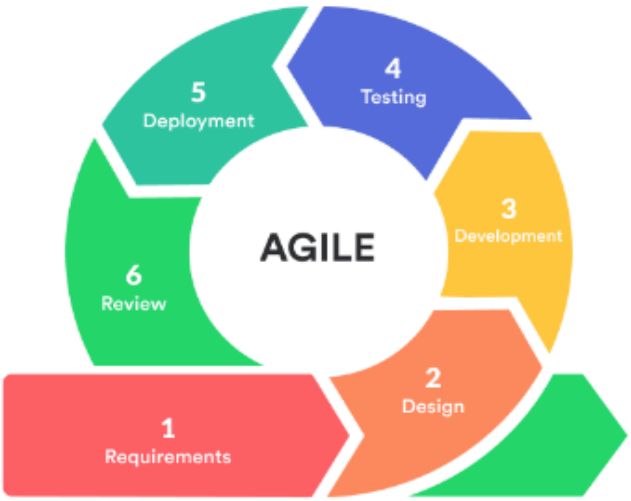


Fig. 5 *Phases within Agile Methodology[7]*

3.1 Requirements Phase

The requirements phase in the Agile methodology for the Stadium Management System involves gathering detailed requirements and setting priorities. This foundational stage emphasizes interactions with stakeholders to understand the challenges in managing stadium access and ensuring security. The goal is to develop a system that enhances operational efficiency and security using facial recognition technology. Thorough planning ensures a clear understanding of project requirements and fosters collaboration for a cohesive development journey.

3.2 Design Phase

In the design phase of developing the Stadium Management System with facial recognition, a detailed system architecture is created to align seamlessly with the gathered requirements. This involves designing the user interface for a seamless experience, planning the database schema, specifying algorithms for core functionalities, and integrating security measures. The phase establishes a clear and detailed plan, laying the foundation for the systematic implementation of the system in subsequent phases

3.3 Development Phase

The development phase involves building the Stadium Management System based on the design specifications. Key elements include implementing functionalities, features, and the facial recognition mechanism for enhanced security. The system will be developed using tools such as Visual Studio Code, with Raspberry Pi as the hardware platform, and programming languages like Python and libraries like OpenCV for robust and efficient development.

3.4 Testing Phase

In the Agile methodology for the Stadium Management System, the testing phase ensures system functionality through comprehensive testing. This includes unit testing, system testing, user acceptance testing, and security testing. The focus is on validating alignment with both functional and non-functional requirements, delivering reliable and user-friendly stadium management experience with enhanced security features.

3.5 Deployment Phase

In the deployment phase, the Stadium Management System is released and installed into the operational environment, involving server configuration, database uploading, and integration of facial recognition technology. Thorough testing precedes deployment, with user training and documentation ensuring effective navigation. Continuous monitoring addresses challenges, ensuring a smooth integration into the stadium management landscape.

3.6 Review Phase

The review phase involves assessing the system's performance post-deployment, collecting feedback from users, and identifying areas for improvement. This iterative review ensures that the system continues to meet the evolving needs of the stadium management team and attendees, fostering continuous improvement and adaptation.

4. System Analysis and Design

This section focuses on the specific functionality that users may anticipate from the Secure E-Voting System. It delineates the core features and capabilities that users can interact with, ensuring a comprehensive understanding of the system's behavior. Table 2 and Table 3 shows the functional requirement and non- functional requirement respectively with its description.

Table 2 *Functional requirement*

Requirement	Description
Login	Allow users to log in securely using OTP and local authentication.
Register	Enable users to register securely, including OTP verification, local authentication and facial recognition.
Facial Recognition	Identify authorized fans through facial recognition.
Booking	Facilitate booking of tickets for events.
Access Control	Ensure controlled access to events.
Event Management	Manage various aspects of events
Reporting	Generate reports on ticket sales and attendance
Security Measures	Implement security features including OTP, local authentication, and facial recognition for user verification

Table 3 *Non-functional requirement*

Requirement	Description
Performance	- Ensure fast response times for user interactions
	- Optimize system performance to handle concurrent users and heavy loads.
Usability	- Design an intuitive and user-friendly interface for easy navigation.
Security and Privacy	- Implement robust security measures to protect user data and privacy
	- Ensure compliance with relevant privacy regulations
	- Access to the system is always restricted to authenticated users.
Reliability	- Ensure the system operates reliably without frequent failures.
Availability	- Maintain high availability to allow users to access the system at all times.

4.1 Design Phase

System design entails defining a system's architecture, components, modules, interfaces, and data to meet specified requirements. It acts as a blueprint for implementation, ensuring cohesive system elements. This visualization aids in identifying and resolving issues early, streamlining development. Fig. 6 illustrates the system architecture for the Stadium Management System using Facial Recognition Technology.

Applied Information Technology and Computer Science Vol. 6 No. 1 (2025) p. 644-662

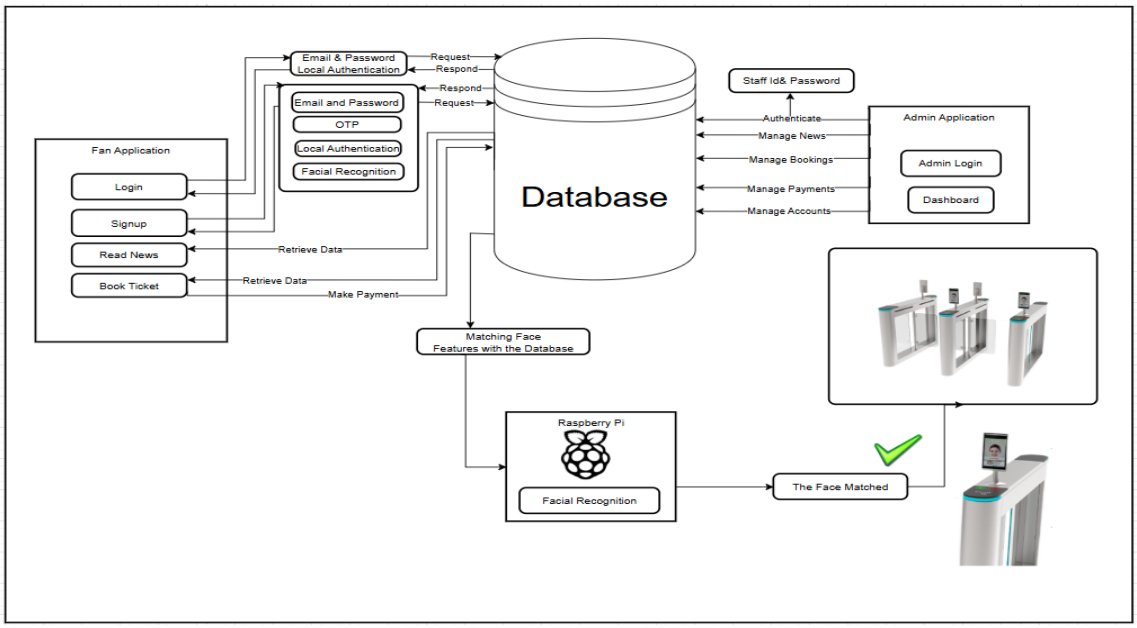


Fig. 6 *System Architecture*

4.2 Use-Case Diagram

The use case diagram outlines user-system interactions, identifying primary actors and actions within the Application for Booking Stadium Seating and Using Facial Recognition Technology. It showcases how users, including fans, administrators, and the system itself, engage with core functionalities such as registration, ticket booking, payments, account management, and entry validation. Fans access features like logging in, booking tickets, and verifying their entry through facial recognition, while new users register to gain access. Administrators manage bookings, payments, news updates, and user accounts to ensure smooth operations. The facial recognition system validates entries, enhancing security and streamlining access. Fig. 7 illustrates how fans, admins, and the facial recognition system interact with features like registration, login, ticket booking, payment processing, facial recognition entry, and account and booking management.

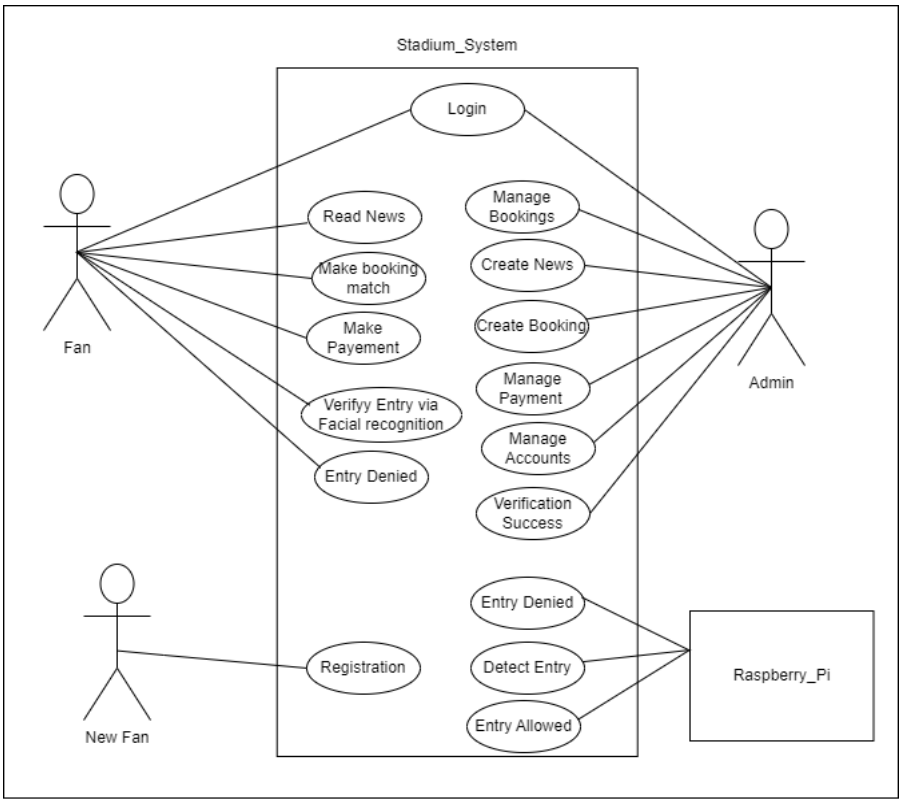


Fig. 7 *Use Case Diagram for the Stadium Management System*

Applied Information Technology and Computer Science Vol. 6 No. 1 (2025) p. 644-662

4.3 Sequence Diagram

A Sequence Diagram details interactions between actors and a system, outlining user interface, backend, and external system interactions for secure user authentication. These specifications demonstrate how fans and administrators engage with the system to ensure secure access. The Sequence Diagram in Fig. 8 illustrates the sequential steps for fan registration and login, showcasing interactions between the application, validation processes, and the database. The Sequence Diagram in Fig. 9 presents the admin login sequence, emphasizing the validation of credentials and secure access to the admin dashboard. Additionally, the Sequence Diagram in Fig. 10 highlights the facial recognition process for fan entry, detailing the interactions between the entry gate, Raspberry Pi system, camera detection, and the database to validate and grant secure access. Collectively, these diagrams emphasize the system's robust authentication and seamless operation for all user roles.

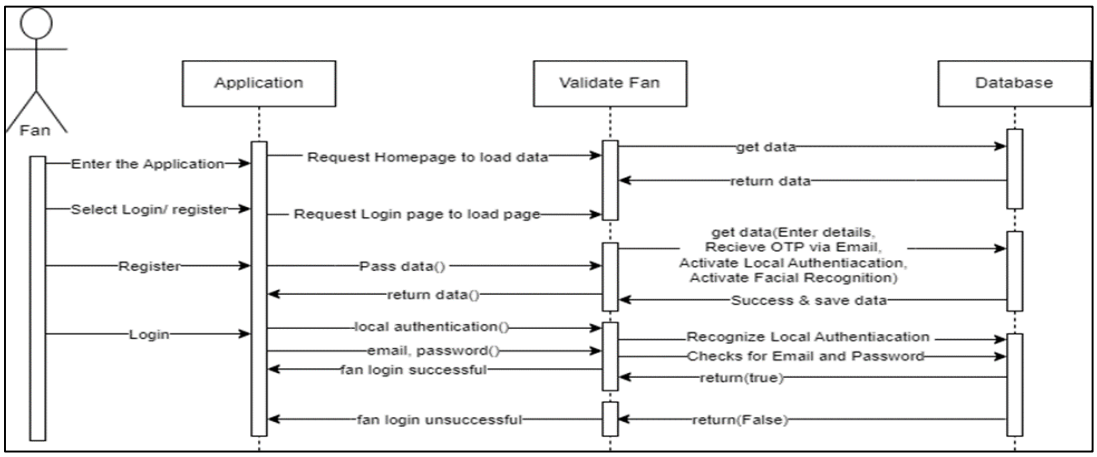


Fig. 8 *Registration & Login Fan Sequence Diagram*

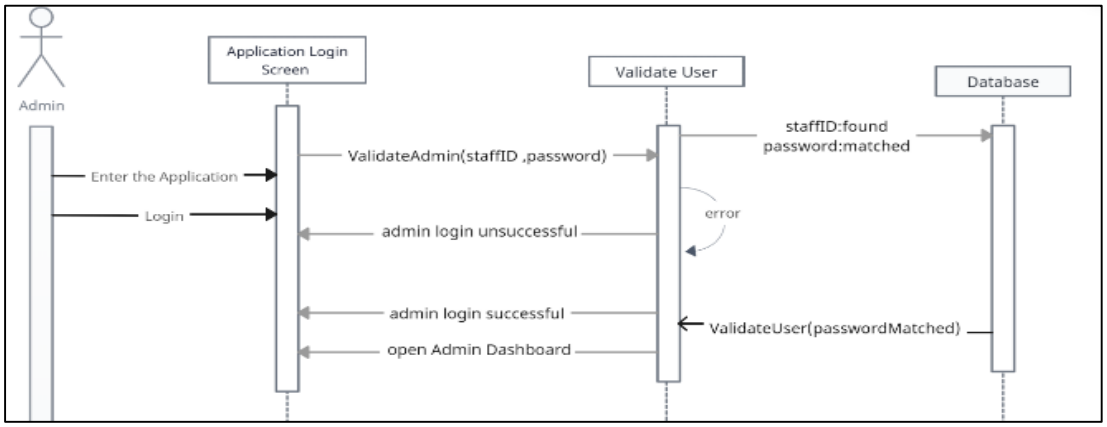


Fig. 9 *Sequence Diagram for Login Admin*

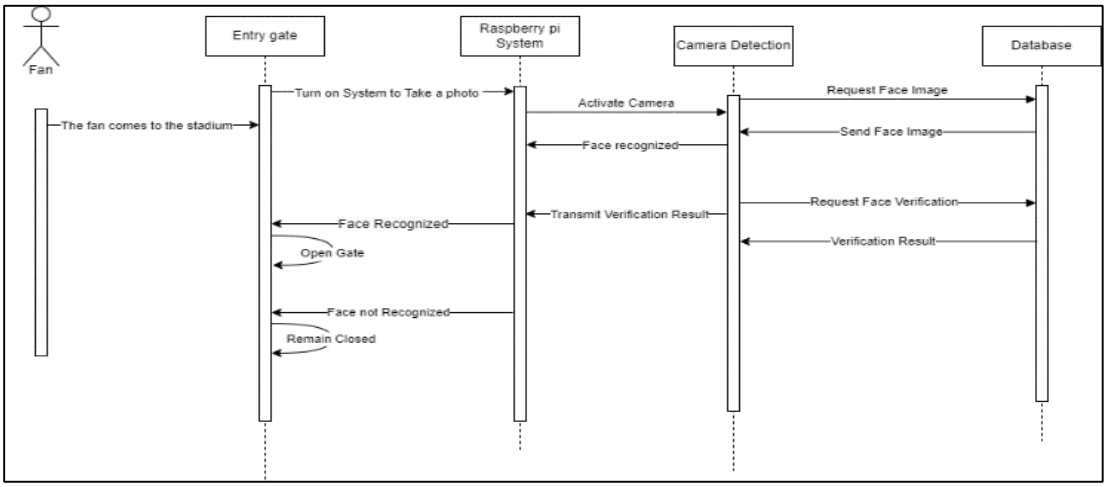


Fig. 10 *Sequence Diagram for Login Admin*

4.4 Activity Diagram

An activity diagram provides a detailed representation of system workflows, showcasing the sequence of actions, decisions, and interactions between actors and the system. Fig. 11 outlines the fan login and registration process, including credential input, local authentication, email/password verification, and successful login or registration. Fig. 12 expands on fan authentication, highlighting the steps for verifying local authentication, checking credentials, and redirecting users to the home page upon successful login. Fig. 13 depicts the admin login process, showing how credentials are validated to grant access to the admin dashboard for managing system operations. Fig. 14 illustrates the facial recognition process for stadium entry, detailing how the system captures images, processes facial data, compares identifiers, and determines entry eligibility. Together, these diagrams highlight the system's secure and streamlined workflows for both fans and administrators.

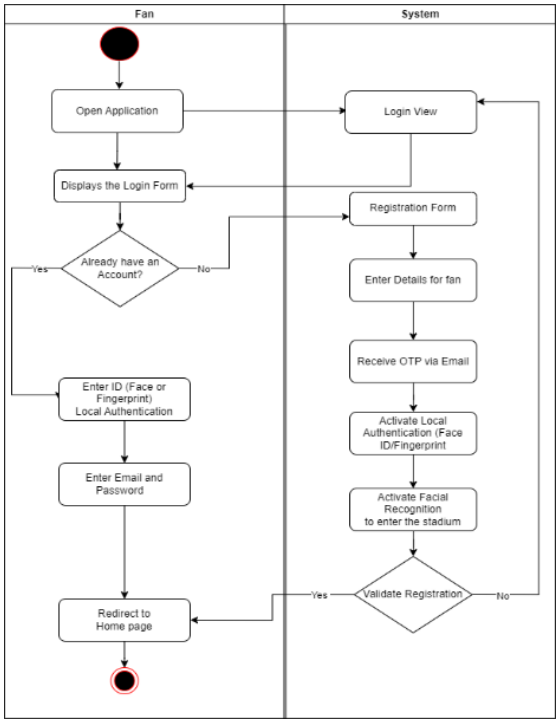


Fig. 11 *Fan Login and Registration Process*

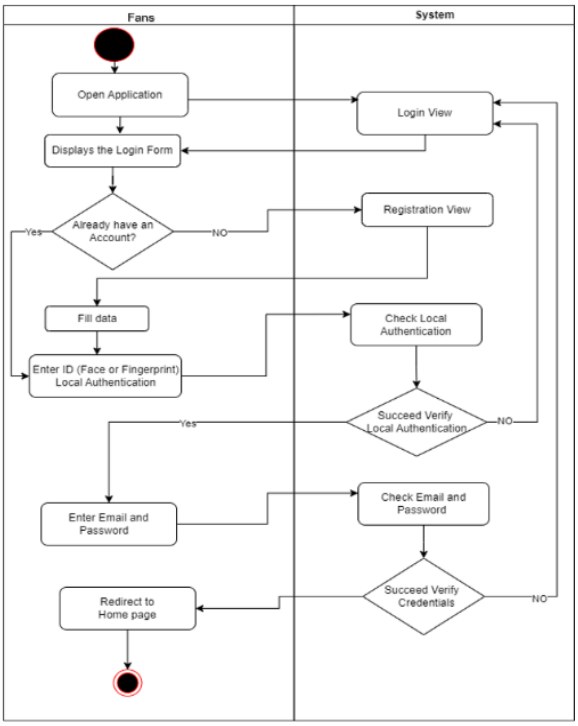


Fig. 12 *Fan Authentication Workflow*

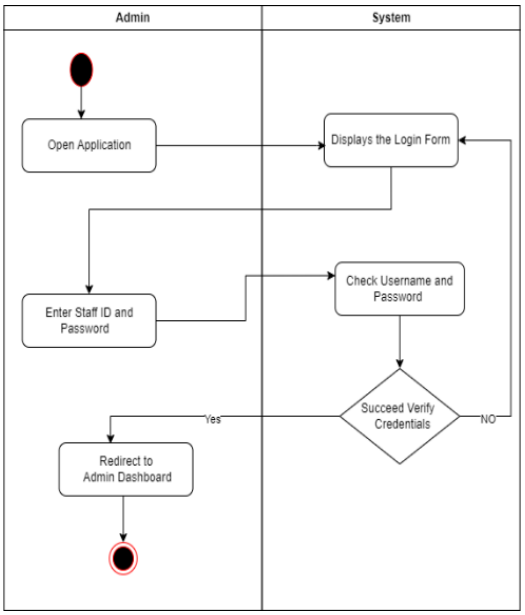


Fig. 13 *Admin Login Process*

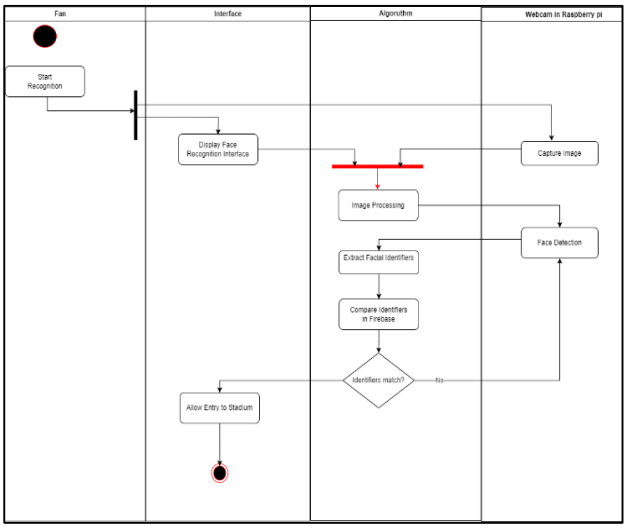


Fig. 14 *Facial Recognition Process for Stadium Entry*

5. Implementation and Testing

This section covers the implementation and testing of the Stadium Management System using Facial Recognition Technology, focusing on security and user accessibility. Technical challenges were resolved, and comprehensive testing was conducted to ensure the system's functionality, reliability, and readiness for deployment.

5.1 Security Implementation

This section highlights the integration of hardware and software to ensure reliability and user safety. It includes facial recognition, OTP verification, local biometric authentication, real-time security alerts, and controlled stadium access.

5.1.1 System Hardware Setup

The stadium management system hardware includes a Raspberry Pi 4 for processing, a high-definition webcam for facial recognition, and a relay module with an electric door lock for automated access control. USB-powered speakers provide audio feedback, and rechargeable batteries ensure uninterrupted operation. Fig. 15 illustrates the hardware components, showcasing the Raspberry Pi, webcam, relay module, and other peripherals in the setup.

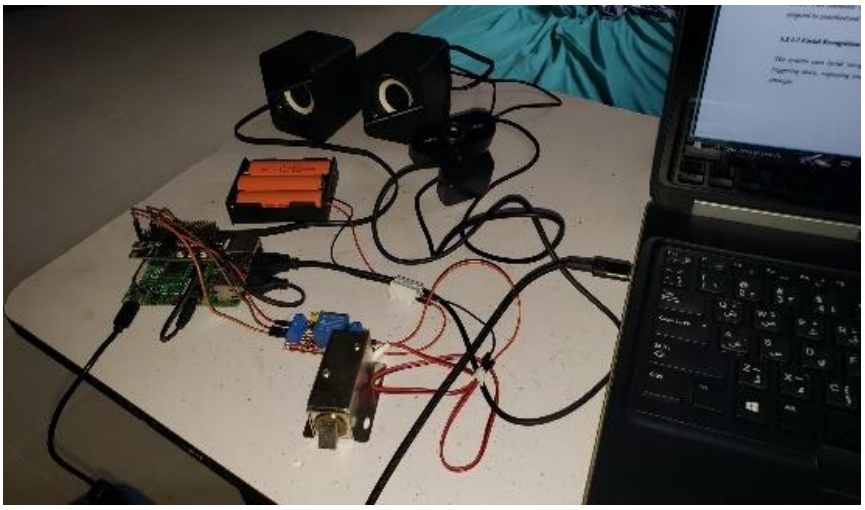


Fig. 15 *Hardware Components*

- **Database Connection**

The system integrates Firebase for database management, initializing credentials and connecting to a storage bucket for efficient data handling. Fig. 16 displays the Firebase initialization code, showing how the system connects to the database and manages data storage. This configuration ensures seamless communication between hardware and software. firestore.client() connects to Firestore, enabling database operations like reading, writing, updating, and deleting documents, allowing interaction with Firestore collections and documents.

```
# Firebase Initialization
cred = credentials.Certificate("/home/pi/Face Recognition/service_account.json")
firebase_admin.initialize_app(cred, {"storageBucket": "stadium-booking-c5ad5.appspot.com"})
db = firestore.client()
bucket = storage.bucket()
```

Fig. 16 *Firebase Initialization Code for Database Connection*

- ## Facial Recognition Data Retrieval and Matching

The system retrieves current match data and compares fan profiles to ensure secure and accurate access control. Fig. 17 shows the function for fetching match data from the database, streaming documents from the "currmatches" collection, and extracting relevant details. Fig. 18 illustrates how captured facial data is compared with stored profile images using OpenCV and facial recognition libraries to validate user identities.

```python
def fetch_currmatches_data():
    try:
        currmatches_ref = db.collection("currmatches")
        documents = currmatches_ref.stream()
        for doc in documents:
            match_data = doc.to_dict()
            match_data['id'] = doc.id
            return match_data
    except Exception as e:
        print(f"[ERROR] Unable to fetch currmatches data: {e}")
        return None
```

Fig. 17 *Fetch Current Matches Data*

```python
def compare_with_profile_images(fans, face_encoding):
    for fan in fans:
        try:
            profile_image_url = fan.get("profileImage", "")
            if not profile_image_url:
                print(f"[WARNING] No profile image for {fan.get('name', 'Unknown')}. Skipping...")
                continue

            response = urllib.request.urlopen(profile_image_url)
            profile_image = np.array(Image.open(response))

            rgb_profile = cv2.cvtColor(profile_image, cv2.COLOR_BGR2RGB)
            profile_encodings = face_recognition.face_encodings(rgb_profile)

            if not profile_encodings:
                print(f"[WARNING] No face found in profile image for {fan['name']}")
                continue

            match = face_recognition.compare_faces(profile_encodings, face_encoding, tolerance=0.5)
            if match[0]:
                return fan
        except Exception as e:
            print(f"[ERROR] Failed to process profile image for {fan['name']}: {e}")
    return None
```

Fig. 18 *Compare with Profile Images*

- ## Saving and Handling Recognized Data

Recognized faces are securely stored in the system to maintain data integrity and traceability. Fig. 19 demonstrates how recognized facial images are saved to Firebase storage, with public URLs generated for access and local files deleted to conserve space. Fig. 20 depicts the process of storing match details, including fan and match data, in the "historymatch" collection, ensuring a record of all successful entries.

```python
def save_recognized_image_to_storage(name, frame, face_location):
    top, right, bottom, left = face_location
    margin = 50
    height, width, _ = frame.shape
    top = max(0, top - margin)
    bottom = min(height, bottom + margin)
    left = max(0, left - margin)
    right = min(width, right + margin)
    face = frame[top:bottom, left:right]

    timestamp = int(time.time())
    local_file_path = f"{name}_{timestamp}.jpg"
    cv2.imwrite(local_file_path, face)

    try:
        blob = bucket.blob(f"recognized_faces/{name}/{timestamp}.jpg")
        blob.upload_from_filename(local_file_path)
        blob.make_public()
        public_url = blob.public_url

        os.remove(local_file_path)
        print(f"[INFO] Uploaded image to {public_url}")
        return public_url
    except Exception as e:
        print(f"[ERROR] Failed to upload image to storage: {e}")
        return None
```

Fig. 19 *Save Recognized Image to Storage*

```python
def save_to_historymatch(fan, real_face_url, match_id):
    try:
        historymatch_ref = db.collection("historymatch").where("matchId", "==", match_id).limit(1)
        historymatch_docs = historymatch_ref.get()

        timestamp = time.time()
        fan_data = {
            'fanId': fan.get('fanId', ''),
            'name': fan['name'],
            'profileImage': fan.get('profileImage', ''),
            'realFaceImage': real_face_url,
            'email': fan.get('email', ''),
            'paymentAmount': fan.get('paymentAmount', 0),
            'paymentImage': fan.get('paymentImage', ''),
            'timestamp': timestamp
        }

        if len(historymatch_docs) > 0:
            match_doc = historymatch_docs[0]
            match_ref = db.collection("historymatch").document(match_doc.id)
            match_data = match_doc.to_dict()
            fans_map = match_data.get('fans', {})
            fans_map[fan.get('fanId', fan['name'])] = fan_data
            match_ref.update({'fans': fans_map, 'lastUpdated': timestamp})
            print(f"[INFO] Updated historymatch for match {match_id} with fan {fan['name']}")
        else:
            doc_ref = db.collection("historymatch").document()
            doc_ref.set({'matchId': match_id, 'fans': {fan.get('fanId', fan['name']): fan_data}, 'created': timestamp, 'lastUpdated': timestamp})
            print(f"[INFO] Created new historymatch for match {match_id} with fan {fan['name']}")
    except Exception as e:
        print(f"[ERROR] Failed to save to historymatch: {e}")
```

Fig. 20 *Save to History Match*

- ## Saving and Handling Recognized Data

The system manages unauthorized access attempts and entry points for enhanced security. Fig. 21 shows how images of unrecognized faces are captured and uploaded to Firebase under "unauthorized_access," documenting all unauthorized attempts. Fig. 22 displays the door lock timer function, which unlocks the door for valid entries and relocks it automatically after a set duration to ensure secure operations.

```python
def handle_unauthorized_access(frame, face_location):
    play_sound(audio_messages["cannot_attend"])

    # Capture image of the unknown face
    top, right, bottom, left = face_location
    unknown_face = frame[top:bottom, left:right]
    timestamp = int(time.time())
    local_file_path = f"unknown_{timestamp}.jpg"
    cv2.imwrite(local_file_path, unknown_face)

    try:
        # Upload image to Firebase Storage
        blob = bucket.blob(f"unauthorized_access/{timestamp}.jpg")
        blob.upload_from_filename(local_file_path)
        blob.make_public()
        os.remove(local_file_path)
        print(f"[INFO] Unauthorized face image uploaded: {blob.public_url}")
    except Exception as e:
        print(f"[ERROR] Failed to upload unauthorized access image: {e}")
```

Fig. 21 *Handle Unauthorized Access*

```python
def door_lock_timer():
    GPIO.output(RELAY, GPIO.HIGH)  # Unlock the door
    print("[INFO] Door unlocked")
    time.sleep(5)  # Keep the door unlocked for 5 seconds
    GPIO.output(RELAY, GPIO.LOW)  # Lock the door
    print("[INFO] Door automatically locked")
```

Fig. 22 *Door Lock Timer*

- ## Real-Time Processing and Recognition Flow

The real-time processing loop integrates facial recognition, image capture, and access control mechanisms for seamless stadium entry. Fig. 23 illustrates the continuous processing of frames, facial recognition, validation of matches, and activation of access mechanisms for authorized users, ensuring efficient and secure entry.

```
while True:
    frame = vs.read()
    if frame is None:
        print("[ERROR] Failed to grab frame.")
        continue

    rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    face_locations = face_recognition.face_locations(rgb_frame, model="hog")
    face_encodings = face_recognition.face_encodings(rgb_frame, face_locations)

    match_data = fetch_currmatches_data()
    if not match_data:
        continue
    fans = match_data.get("fans", [])

    current_time = time.time()

    for face_encoding, face_location in zip(face_encodings, face_locations):
        matched_fan = compare_with_profile_images(fans, face_encoding)
        if matched_fan:
            real_face_url = save_recognized_image_to_storage(matched_fan['name'], frame, face_location)
            save_to_historymatch(matched_fan, real_face_url, match_data['id'])
            play_sound(audio_messages["welcome"])
            threading.Thread(target=door_lock_timer).start()
        else:
            handle_unauthorized_access(frame, face_location)

    cv2.imshow("Face Recognition Door Lock System", frame)
    if cv2.waitKey(1) & 0xFF == ord("q"):
        break
```

Fig. 23 *Facial Recognition Processing Loop*

- ## Facial Recognition: Authorized vs. Unauthorized Fans

The system leverages facial recognition to process authorized and unauthorized individuals while managing entry securely. It fetches match data and fan profiles, captures live images, and detects faces in real time. For authorized individuals, such as "abdulqader Abdulrahman" in Fig. 24, a match is found, their details are saved, the door unlocks, and feedback is provided with a green rectangle. For unauthorized individuals, shown in Fig. 25, the system labels them as "Unknown," triggers an alert, logs their data, keeps the door locked, and denies access, ensuring robust security and controlled entry.

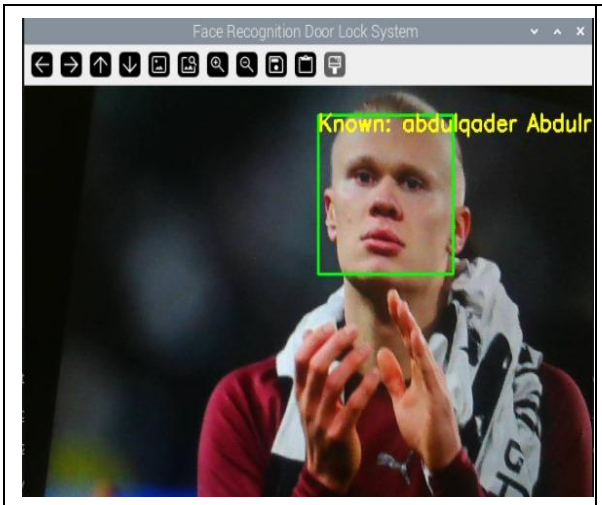


Fig. 24 *Camera Recognized Fan Face*

Fig. 25 *Camera Detected Unauthorized Person*

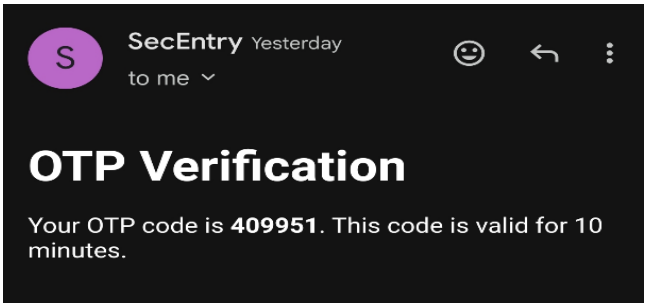
5.1.2 System Software Setup

The system software setup configures authentication mechanisms, facial recognition, and cloud integration with Firebase. Developed using Python, Flutter, and SendGrid API, it ensures secure and efficient operation of the stadium management system.

- **Email-based OTP**

Fig. 26 (a) shows the OTP module implementation using SendGrid API, which sends email-based OTPs with recipient details, sender information, and code validity. The function ensures reliable delivery, confirming success with a status code of 202. Fig. 26 (b) displays an OTP email sent by the system, branded with "SecEntry," containing the code "409951" valid for 10 minutes. Fig. 26 (c) illustrates the app's confirmation message, "OTP Verified!," signaling successful authentication.

(a)



(b)

(c)

Fig. 26 *OTP Module(a) OTP using SendGrid API; (b) OTP email sent by the system; (c) Successful OTP verification*

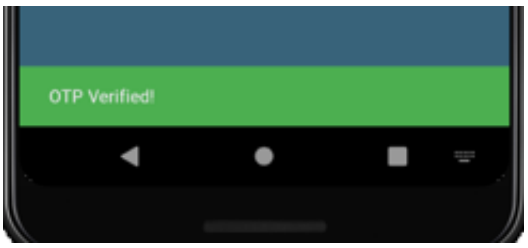
- **Local Authentication**

Local authentication enhances security by using biometric verification, such as fingerprints, to ensure secure access. Fig. 27(a) illustrates the process, starting with a device compatibility check, displaying a fingerprint dialog if supported, and providing feedback for success or failure. Fig. 27(b) highlights utility functions like showFingerprintAuthDialog for managing prompts and verification, with robust error handling and the isBiometricsSupported function to check device compatibility.

```
Future<bool> authenticateWithBiometrics(BuildContext context) async {
  try {
    // Check if biometrics are supported
    if (await LocalAuthUtils.isBiometricsSupported()) {
      // Attempt biometric authentication
      final didAuthenticate = await LocalAuthUtils.showFingerprintAuthDialog(context);
      if (didAuthenticate) {
        // Biometric authentication successful
        showCustomSnackBar(customSnackBarSuccess(successMessage: "Biometric authentication successful."));
        authenticated.value = true;
        return true;
      } else {
        // Biometric authentication failed
        showCustomSnackBar(customSnackBarError(errorMessage: "Biometric authentication failed."));
        return false;
      }
    } else {
      // Biometrics not supported
      showCustomSnackBar(customSnackBarError(errorMessage: "Biometric authentication is not supported on this device."));
      return false;
    }
  } catch (e) {
    // Handle any exceptions during biometric authentication
    showCustomSnackBar(customSnackBarError(errorMessage: "An error occurred during biometric authentication."));
    return false;
  }
}

void changeObscureText() {
  obscureText.value = !obscureText.value;
}
}
```

(a)



(b)

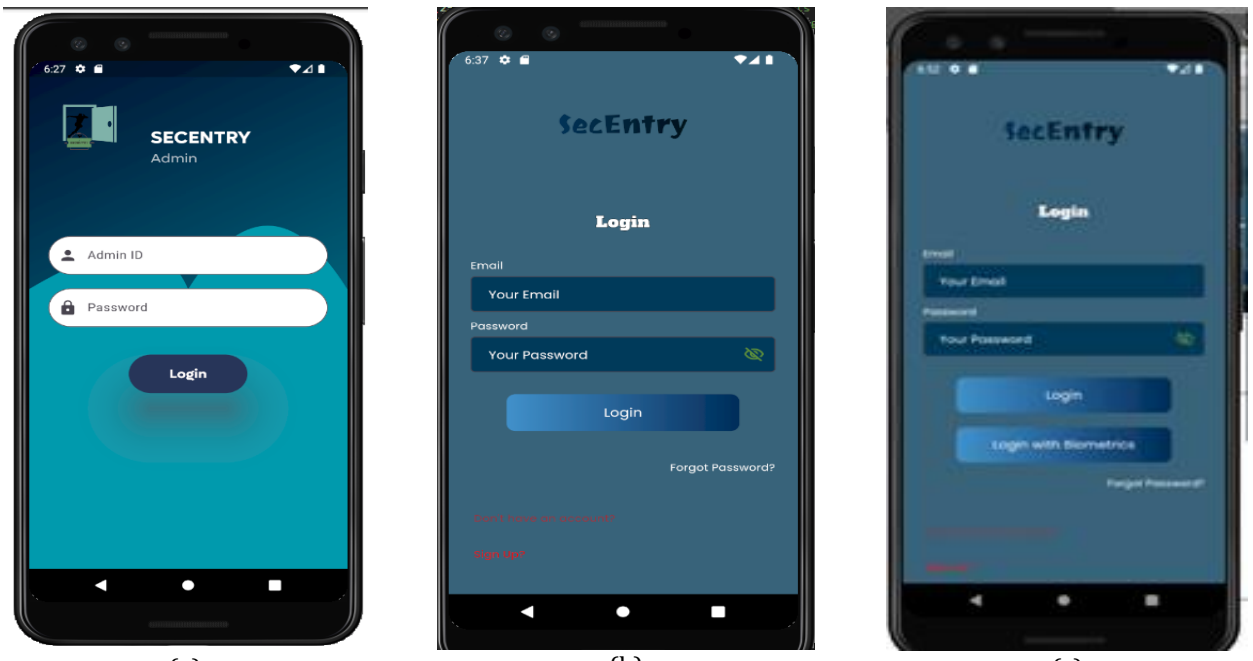
Fig. 27 *Local Authentication Module (a) Biometric authentication process; (b) Utility functions*

5.2 Module Implementation

The Module Implementation section details the development of key modules in the Al-Wehdah Booking System, focusing on enhancing efficiency, security, and user experience. Dividing the system into essential modules streamline development, testing, and maintenance, addressing critical functionalities such as authentication and reporting.

5.2.1 Login Module

The login module features screens for both administrators and users, ensuring secure access to the system. Fig. 28 (a) shows the admin login interface, where administrators authenticate using their Admin ID and password. Fig. 28 (b) presents the user login screen, offering email and password authentication with a "Forgot Password?" option for account recovery. Fig. 28 (c) enhances the user login process by integrating biometric authentication, providing an additional layer of security and convenience.



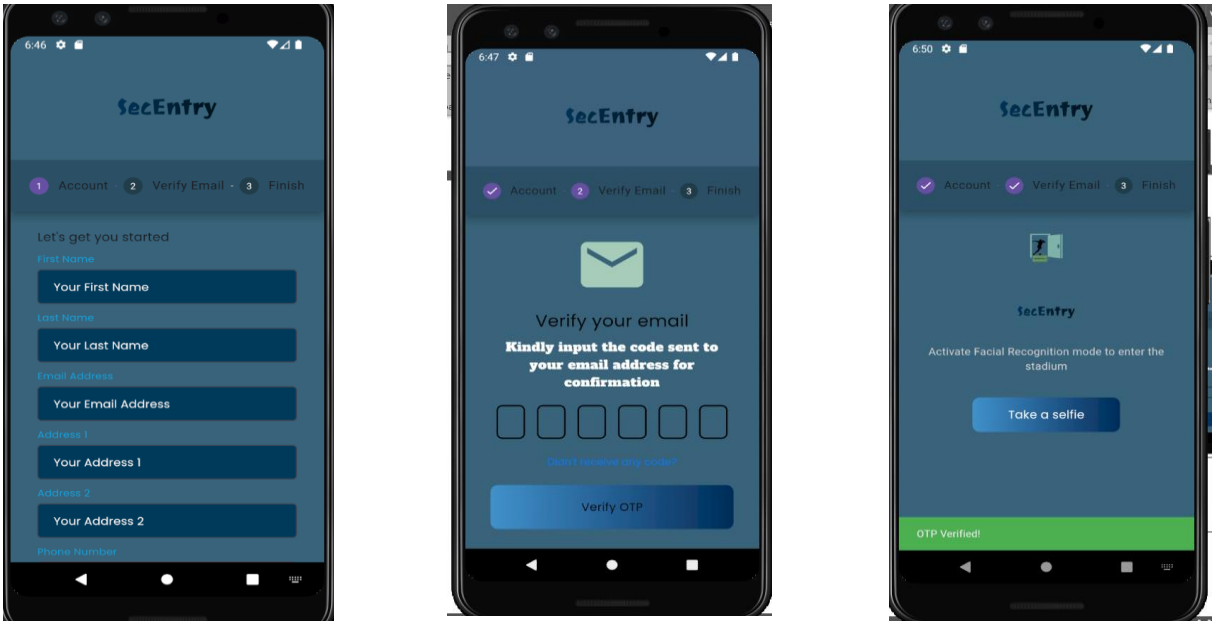
(a) (b) (c)

Fig. 28 Login module. *(a) Admin page; (b) Fan page; (c) Fan Local Auth page*

Applied Information Technology and Computer Science Vol. 6 No. 1 (2025) p. 644-662

5.2.2 Register Module

The register module streamlines the user registration process in the application. Fig. 29 (a) displays the fan sign-up form, where users input personal details like name, email, and contact information to create an account. Fig. 29 (b) illustrates the email verification step, where users enter the OTP sent to their registered email for confirmation. Fig. 29 (c) showcases the selfie capture process for activating facial recognition, ensuring secure access to the stadium. These steps collectively ensure a smooth and secure registration experience for users.

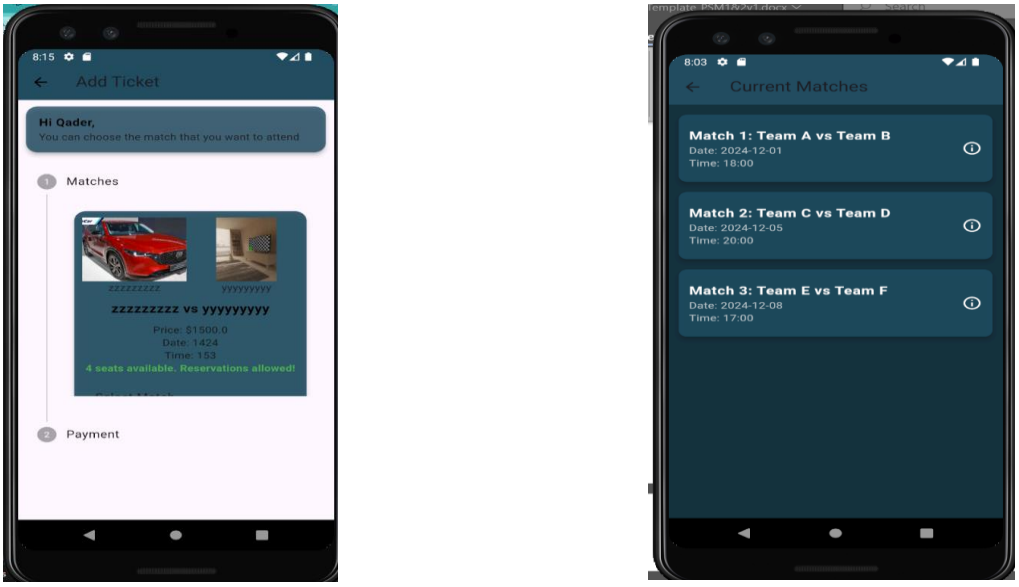


(a) (b) (c)

Fig. 29 *Register Module. (a) Fan Sign-Up Form; (b) Email Verification; (c) Selfie Capture for Facial Recognition.*

5.2.3 Booking Module

The booking module simplifies the ticket reservation process for fans. Fig. 30 (a) illustrates the "Add Ticket" interface, where users can select a match, view details such as date, time, price, and available seats, and proceed with reservations. Fig. 30 (b) showcases the "Current Matches" interface, providing an organized view of matches already booked by the fan. These features enhance the user experience by making the ticket booking process intuitive and efficient.

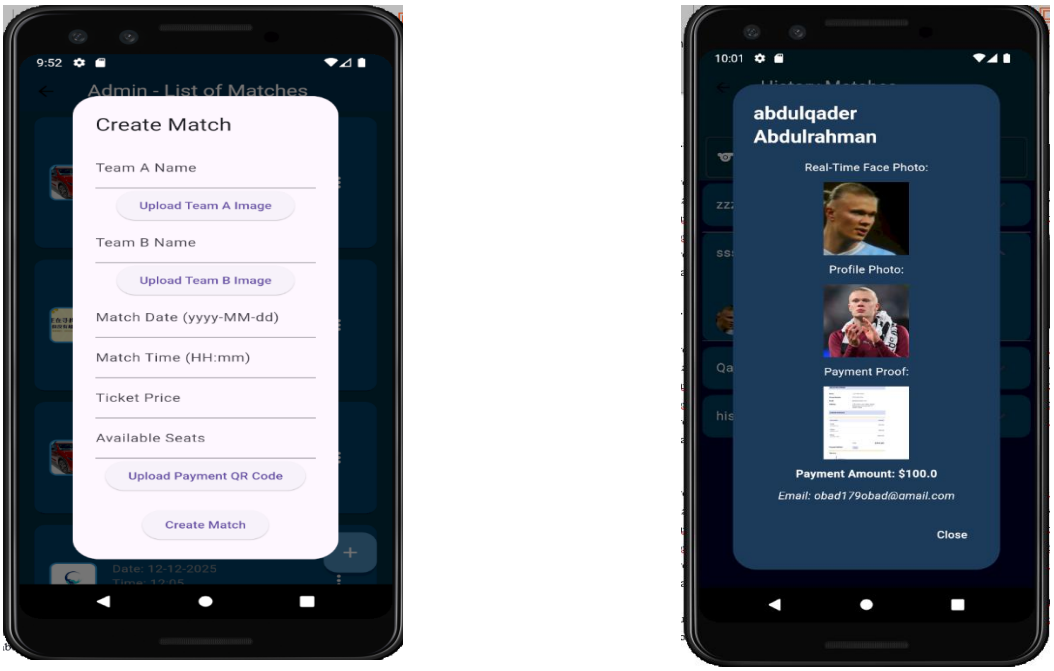


(a) (b)

Fig. 30 *Booking Module. (a) Add Ticket Interface; (b) Current Matches View*

5.2.4 Event Management and Reporting Modules

The event management and reporting module streamlines match creation and accountability for administrators. Fig. 31(a) illustrates the "Create Match" interface, enabling admins to input match details such as team names, match date, time, ticket price, and available seats. Admins can also upload team images and a payment QR code, with the "Create Match" button finalizing the setup for efficient event organization. Fig. 31(b) highlights the reporting module, displaying match history details for a fan. The interface provides comprehensive information, including real-time face photos, stored profile photos, and payment proof, along with the fan's name, payment amount, and email address. These features enhance transparency and accountability, ensuring accurate attendance verification.



(a) (b)

Fig. 31: *Event Management and Reporting Module. (a) Create Match Interface; (b) Match History Deta*

5.3 Testing

The testing phase evaluates the system's functionality and security through a structured test plan, including functionality testing and security checks. This section presents user acceptance testing outcomes, ensuring the system meets project requirements and provides robust security.

5.3.1 User Acceptance Form

The user acceptance section evaluates the system's usability and functionality from fans' and admins' perspectives. Feedback ensures practicality, user-friendliness, and effectiveness. Table 4 captures fan feedback, while Table 5 reflects admin evaluations, addressing their specific needs.

Table 4 *Acceptance functional Form for Fan*

Description	Pass	Fail
User can login without problem	12	0
Display messages are easy to understand	12	0
User can receive OTP via email	12	0
User can enter the correct OTP	12	0
User can use OTP without problem	12	0
User can set up local authentication (Face ID/Fingerprint ID)	12	0
User can use local authentication without problem	12	0
User can register facial recognition data	12	0
User can use facial recognition for entry	12	0

Table 4 *(Cont.)*

Description	Pass	Fail
Facial recognition works seamlessly at the stadium gate	12	0
User can read news	12	0
User can book tickets	12	0
User can view booking history	12	0
Each user can use the system without problems	12	0
The interface is easy to understand	12	0
The system is easy to understand	12	0

Table 5 *Acceptance functional Form for Admin*

Description	Pass	Fail
Admin can login without problem	2	0
Display messages are easy to understand	2	0
Admin can manage news	2	0
Admin can manage bookings	2	0
Admin can manage payments	2	0
Admin can navigate through the system without problem	2	0
The interface is intuitive for admin	2	0
The system functionalities are clear	2	0

6. Conclusion and Future Works

The Al-Wehdah Stadium Booking System successfully achieved its objectives, integrating advanced technologies like facial recognition and OTP verification to provide secure and user-friendly stadium management. The system features a robust architecture using Flutter, Firebase, and Raspberry Pi, with rigorous testing validating its functionality across login, registration, booking, and reporting modules. Limitations include reliance on stable internet, limited scalability during peak events, and hardware sensitivity to environmental conditions. Future enhancements focus on improving scalability, introducing offline functionality, upgrading facial recognition accuracy, and expanding reporting capabilities to further optimize system performance and usability.

Acknowledgment

The authors would like to thank the Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia, for its support.

Conflict of Interest

Authors declare that there is no conflict of interests regarding the publication of the paper.

Author Contribution

The authors confirm contribution to the paper as follows: ***study conception and design****: A. A Ali, S.K. Ahmad Khalid;* ***data collection****: A. A Ali, S.K. Ahmad Khalid;* ***analysis and interpretation of results****: A. A Ali, S.K. Ahmad Khalid;* ***draft manuscript preparation****: A. A Ali, S.K. Ahmad Khalid. All authors reviewed the results and approved the final version of the manuscript.*

References

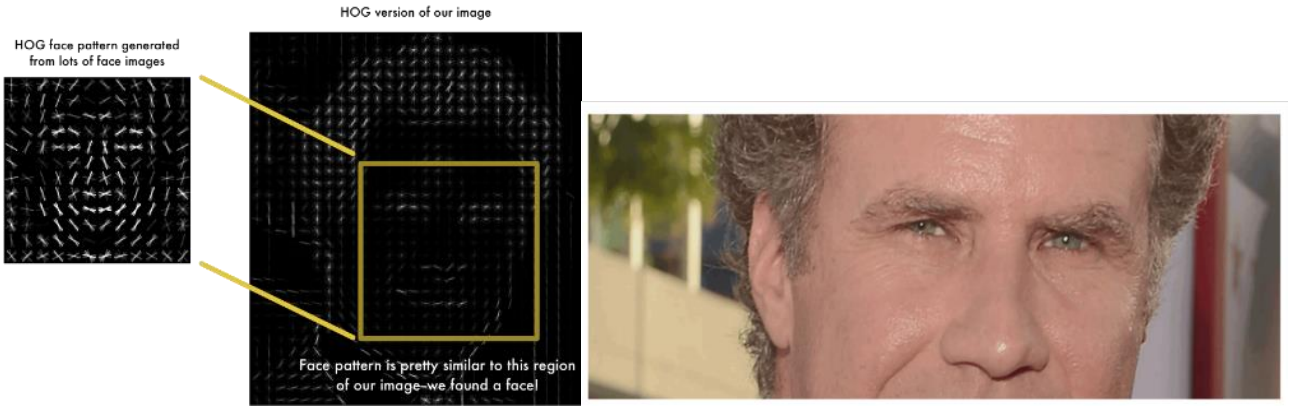
[1] F. Junaedi, F. G. Sukmono, and A. Fuller, "Kanjuruhan Disaster, Exploring Indonesia Mismanagement Football Match," *E3S Web of Conferences*, vol. 440, Nov. 2023, doi: 10.1051/E3SCONF/202344003010.

[2] L. Li, X. Mu, S. Li, and H. Peng, "A Review of Face Recognition Technology," *IEEE Access*, vol. 8, pp. 139110–139120, 2020, doi: 10.1109/ACCESS.2020.3011028.

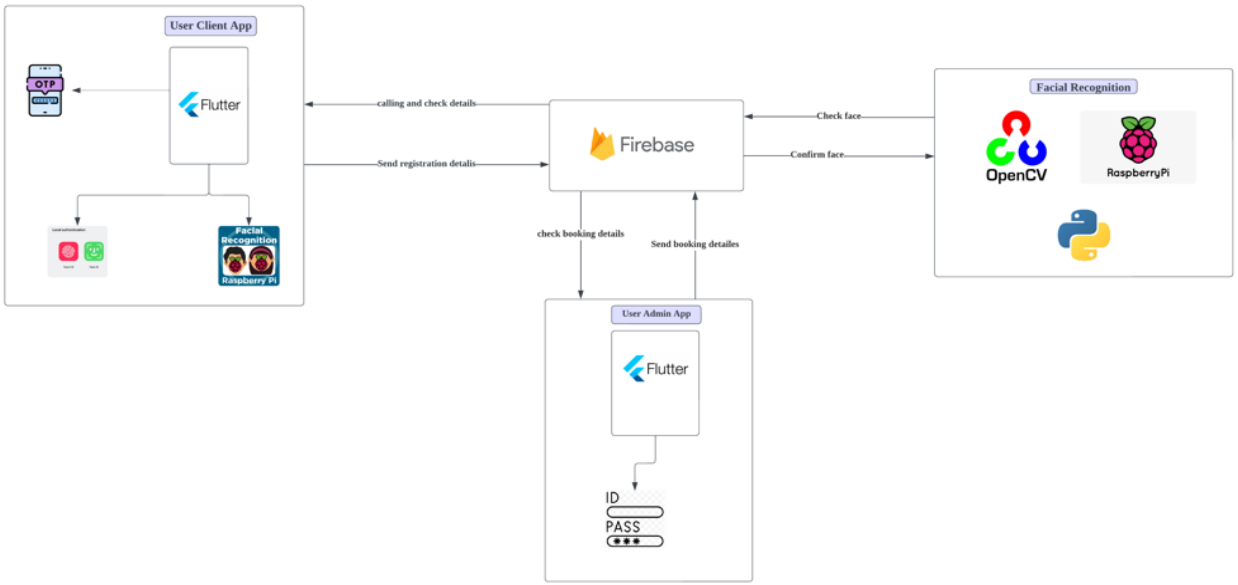
[3] V. Margapuri, N. Penumajji, and M. Neilsen, "PiBase: An IoT-based Security System Using Google Firebase and Raspberry Pi," *Proceedings of the 2021 IEEE International Conference on Internet of Things and Intelligence Systems, IoTaIS 2021*, pp. 79–85, 2021, doi: 10.1109/IOTAIS53735.2021.9628513.

[4] "OTP View Flutter. What is OTP (One Time Password)? | by Marvel Apps | Medium." Accessed: Apr. 24, 2024. [Online]. Available: https://medium.com/@MarvelApps_/otp-view-flutter-ed737f687922
[5] D. Sharma, "LOCAL AUTHENTICATION IN FLUTTER," *www.irjmets.com @International Research Journal of Modernization in Engineering*, vol. 3120, Apr. 2023, doi: 10.56726/IRJMETS36348.
[6] "C.A. Osasuna, first LaLiga club with facial recognition access | Veridas." Accessed: Dec. 31, 2024. [Online]. Available: https://veridas.com/en/success-stories/osasuna-access-facial-recognition/?utm_source=chatgpt.com
[7] M. Moloto, A. Harmse, and T. Zuva, "Impact of Agile Methodology Use on Project Success in Organizations - A Systematic Literature Review," *Advances in Intelligent Systems and Computing*, vol. 1294, pp. 267–280, 2020, doi: 10.1007/978-3-030-63322-6_21.

Appendix A



Appendix A1: Facial HOG Representation: Feature Capture



Appendix A2: Project Framework for Al-Wehdah Stadium