# **Lab Assignment Report – Aegis ID**

## **1. Roles and Responsibilities of Team Members**

### **Kavyan Hembrom (IIT2024245) – UI/UX Design & Wireframe Design**

* Create wireframes and prototypes using Figma
* Design user-friendly interfaces for login, dashboard, hostel outpass, and emergency features
* Ensure smooth navigation and consistent UI across the app

### **Afzal Raja (IIB2024031) – Backend Developer**

* Develop backend APIs using Express.js and Node.js
* Manage integration with MongoDB for data storage and retrieval
* Implement authentication, passkey generation, and secure communication (JWT)

### **Abdul Azeem (IIT202423) – User Schema Development**

* Design and implement MongoDB schemas for students, wardens, and security staff
* Define relationships between collections (students ↔ outpass requests ↔ logs)
* Ensure scalability and efficient queries

### **Khushi Shorey (IIT2024244) – Basic Backend Support**

* Assist in API development for secondary features (emergency calls, logging)
* Test backend endpoints with Postman
* Help integrate Firebase authentication if chosen

### **Vandita Jain (IIT2024228) – Frontend Developer**

* Develop mobile app frontend using React Native
* Implement screen navigation (login → dashboard → QR/NFC scan → outpass)
* Integrate APIs with backend services

### **Shon Ravindra Waghchoure (IIT2024247) – Blockchain & Security**

* Implement optional blockchain layer for transaction logs
* Work on cryptographic passkey generation (SHA-256)
* Ensure secure storage and transfer of student credentials

## **2. Frontend Design Layout & Tools**

### **Development Tools**

* **Primary Framework:** React Native
* **Design Tool:** Figma

### **Application Screens**

#### **Login/Registration Screen**

* **Inputs:** Email/Student ID, Password
* **Components:** Login button, registration link
* **Functionality:** User authentication and account creation

#### **Dashboard Screen**

* **Components:**
  + Generate Passkey button
  + Scan QR button
  + Request Outpass button
  + Emergency Help button
* **Navigation:** Central hub for all app features

#### **QR/NFC Passkey Screen**

* **Display:** Dynamic daily passkey as QR code
* **Functionality:** Real-time passkey generation and display
* **Security:** Time-based expiration

#### **Outpass Request Screen**

* **Inputs:** Date, Time, Reason for leave
* **Components:** Submit button, form validation
* **Process:** Request sent to warden for approval

#### **Emergency Screen**

* **Components:**
  + Call Security button
  + Call Ambulance button
* **Functionality:** Auto-location sharing with emergency contacts

#### **Profile Screen**

* **Display:** Student details and information
* **Components:** Edit profile option, account settings

### **Navigation Flow**

Login → Dashboard → (Profile / Outpass / Passkey / Emergency)

## **3. Backend Design Layout & Tools**

### **Technology Stack**

* **Runtime:** Node.js
* **Framework:** Express.js
* **Database:** MongoDB
* **Authentication:** JWT (JSON Web Tokens)
* **Optional:** Blockchain integration

### **Authentication Process Example (Login)**

1. User enters Student ID + Password
2. Backend verifies credentials against database (Firebase Auth / MongoDB)
3. If valid → JWT token generated
4. Token sent to frontend → Access granted
5. Subsequent requests authenticated via JWT

### **System Architecture**

**Frontend (React Native) ↔ Backend (Express.js APIs) ↔ Database (MongoDB + Blockchain logs)**

### **API Structure**

* **Authentication APIs:** Login, registration, token refresh
* **Passkey APIs:** Generate, validate, expire
* **Outpass APIs:** Create request, approve/reject, status check
* **Emergency APIs:** Log incidents, notify contacts
* **Profile APIs:** View, update student information

## **4. Database Design & Schema Description**

### **Database Management System: MongoDB**

### **Collections/Tables**

#### **Students Collection**

* **Attributes:**
  + student\_id (Primary Key) - Unique student identifier
  + name - Full name of student
  + email - Email address
  + password\_hash - Encrypted password
  + hostel\_id - Associated hostel identifier
  + device\_id - Registered device identifier

#### **Passkeys Collection**

* **Attributes:**
  + passkey\_id (Primary Key) - Unique passkey identifier
  + student\_id (Foreign Key) - Reference to Students collection
  + date - Generation date
  + encrypted\_key - SHA-256 encrypted passkey
  + expiry\_time - Time-based expiration

#### **Outpass Requests Collection**

* **Attributes:**
  + request\_id (Primary Key) - Unique request identifier
  + student\_id (Foreign Key) - Reference to Students collection
  + reason - Purpose of leave
  + date - Requested date
  + time - Requested time
  + status - Current status (Pending/Approved/Rejected)
  + warden\_id - Approving authority
  + created\_at - Request timestamp
  + updated\_at - Last modification timestamp

#### **Emergency Logs Collection**

* **Attributes:**
  + emergency\_id (Primary Key) - Unique incident identifier
  + student\_id (Foreign Key) - Reference to Students collection
  + type - Emergency type (Security/Ambulance)
  + timestamp - Incident time
  + location - GPS coordinates
  + status - Response status
  + description - Incident details

#### **Blockchain Logs Collection (Optional)**

* **Attributes:**
  + txn\_id (Primary Key) - Transaction identifier
  + student\_id - Student reference
  + event\_type - Type of logged event
  + hash - Cryptographic hash
  + timestamp - Block timestamp
  + previous\_hash - Previous block reference

### **Database Relationships**

* **Student ↔ Passkeys:** One-to-Many (1:M)
* **Student ↔ Outpass Requests:** One-to-Many (1:M)
* **Student ↔ Emergency Logs:** One-to-Many (1:M)
* **Student ↔ Blockchain Logs:** One-to-Many (1:M)

## **5. System Integration & Workflow Examples**

### **Outpass Request Workflow**

1. **Frontend (React Native):** Student fills outpass form and clicks Submit
2. **API Call:** POST request sent to /api/outpass/create with form data
3. **Backend (Express.js):** API validates request data and user authentication
4. **Database (MongoDB):** Request stored in Outpass Requests collection with status = "Pending"
5. **Notification:** Warden receives notification of new request
6. **Approval Process:** Warden reviews and approves/rejects via warden interface
7. **Status Update:** Database updated with warden decision
8. **Frontend Notification:** Student receives real-time status update in dashboard

### **QR/NFC Passkey Generation & Validation Workflow**

1. **Request:** Student clicks "Generate Passkey" in mobile app
2. **Backend Processing:**
   * Generate unique token using SHA-256 encryption
   * Set 24-hour expiration timestamp
   * Store in Passkeys collection
3. **Frontend Display:** Token converted to QR code and displayed on screen
4. **Validation Process:**
   * Security personnel scan QR code
   * Backend verifies token against database
   * Check expiration status
   * Grant/deny access based on validation
5. **Logging:** Access attempt logged for audit trail

### **Emergency Response Workflow**

1. **Trigger:** Student presses emergency button (Security/Ambulance)
2. **Location Capture:** App automatically captures GPS coordinates
3. **API Call:** POST request to /api/emergency/alert with location and type
4. **Database Logging:** Incident stored in Emergency Logs collection
5. **Notification System:**
   * Alert sent to security personnel
   * SMS/call initiated to emergency contacts
   * Real-time location shared with responders
6. **Response Tracking:** Status updates logged until incident resolution

### **Data Flow Architecture**

Mobile App (React Native)

↓ HTTP Requests (REST APIs)

Express.js Server

↓ Database Queries

MongoDB Database

↓ Optional Blockchain Logging

Blockchain Network (Immutable Audit Trail)

### **Security Measures**

* **Authentication:** JWT-based secure sessions
* **Data Encryption:** SHA-256 for passkeys, bcrypt for passwords
* **API Security:** Rate limiting, input validation, CORS configuration
* **Database Security:** Connection encryption, access controls
* **Blockchain Integration:** Immutable logging for critical transactions