Mosques Automation system Mobile App report

## **1. Introduction**

**1.1 Purpose**

The Mosque Management System is a comprehensive Flutter-based mobile application designed to streamline the management and monitoring of mosques. It serves as a central platform for government officials, imams, and worshippers, integrating Internet of Things (IoT) capabilities with religious information management.

**1.2 Scope**

This system covers multiple aspects of mosque management, including:

* Real-time monitoring of mosque environments
* Prayer time management
* Lesson scheduling and management
* Worshipper information and services
* Government-level oversight of multiple mosques

**1.3 Intended Audience**

* Government officials responsible for mosque oversight
* Mosque administrators and imams
* IT personnel maintaining the system
* Worshippers using the application's services

## **2. System Architecture**

**2.1 Technology Stack**

* Frontend: Flutter
* Backend: Firebase Realtime Database
* External APIs: Aladhan API for prayer times
* Machine Learning: TensorFlow Lite

## **3. Modules**

**3.1 Main Application (main.dart)**

**3.1.1 Purpose**

Serves as the entry point of the application and sets up the primary navigation structure.

**3.1.2 Key Features**

* Firebase initialization
* Home screen with dynamic navigation options
* Custom hover effects for improved user experience

**3.1.3 Code Structure**

* MyApp class: Root widget of the application
* MyHomePage class: Implements the main navigation interface
* HoverCard widget: Custom widget for interactive navigation buttons

**3.1.4 Dependencies**

* firebase\_core for Firebase initialization
* Custom LocalNotifications class for handling local notifications

**3.2 Government Interface (government.dart)**

**3.2.1 Purpose**

Provides government officials with an overview and management capabilities for all mosques in the system.

**3.2.2 Key Features**

* List of all managed mosques with quick access
* Navigation to detailed statistics and real-time data for each mosque
* Access to system-wide reports and analytics

**3.2.3 Code Structure**

* GovernmentPage class: Main widget for the government interface
* Integration with Firebase for real-time mosque data
* Navigation to AlbabtainPage and other specific mosque pages

**3.2.4 Data Flow**

* Fetches mosque list and summary data from Firebase
* Updates in real-time as mosque data changes

**3.3 Worshippers Interface (worshippers.dart)**

**3.3.1 Purpose**

Offers worshippers easy access to prayer times, nearby mosques, and lesson information.

**3.3.2 Key Features**

* Display of daily prayer times
* List of nearby mosques with distance information
* Access to lesson schedules for each mosque

**3.3.3 Code Structure**

* WorshippersPage class: Main widget for the worshippers interface
* fetchPrayerTimes() method: Retrieves prayer times from Aladhan API
* Integration with device location services for nearby mosque functionality

**3.3.4 API Integration**

* Aladhan API used for fetching accurate prayer times
* Format: http://api.aladhan.com/v1/timingsByAddress/[DATE]?address=[LOCATION]&tune=[ADJUSTMENTS]

**3.4 Al Babtain Mosque Management (albabtain.dart)**

**3.4.1 Purpose**

Provides a comprehensive management interface for the Al Babtain Mosque, serving as a template for individual mosque management.

**3.4.2 Key Features**

* Real-time sensor data display (temperature, motion, light, flow)
* Actuator controls for mosque equipment
* Lesson management interface
* Automated notifications for suspicious activities

**3.4.3 Code Structure**

* AlbabtainPage class: Main widget for mosque management
* SensorDataCard widget: Displays individual sensor readings
* ActionSwitchCard widget: Controls for actuators
* LessonsWidget widget: Manages lesson information

**3.4.4 Sensor Integration**

* Temperature Sensor: Monitors mosque environment
* Motion Sensor: Detects movement during prayer times
* Light Sensor: Manages lighting efficiency
* Flow Sensor: Monitors water usage

**3.4.5 Actuator Control**

* Fan Control: Manages mosque ventilation
* RGB LED Control: Adjusts mood lighting
* WS2812 LED Control: Manages decorative lighting

**3.4.6 Machine Learning Integration**

* TensorFlow Lite models for predictive analysis of sensor data
* getFutureFlow(), getFutureTemp(), getFutureLight() methods for ML predictions

**3.5 Lessons Management (lessons.dart)**

**3.5.1 Purpose**

Manages the scheduling, display, and administration of religious lessons within the mosque.

**3.5.2 Key Features**

* Display of current lesson schedule
* Interface for adding new lessons
* Real-time updates to lesson information

**3.5.3 Code Structure**

* LessonsPage class: Main widget for lesson management
* Integration with Firebase for real-time lesson data
* addLesson() method for creating new lesson entries

**3.5.4 Data Model**

* Lesson Title: String
* Lesson Time: String (formatted time)
* Lesson Date: Date object
* Instructor: String (optional)

## **4. Key Functionalities**

**4.1 Firebase Integration**

* Real-time data storage and retrieval
* Stores sensor data, actuator states, and lesson information

**4.2 Sensor Monitoring**

**4.2.1 Purpose**

Continuously monitors and analyzes data from various sensors installed in the mosque.

**4.2.2 Implementation Details**

* Sensor data fetched from Firebase in real-time
* Thresholds set for each sensor type to detect anomalies
* isWithinIntervals() method checks for suspicious activities during prayer times

**4.3 Actuator Control**

* Allows remote control of various mosque equipment for environment management.
* fanAction(), rgbLedAction(), ws2812Action() methods control respective equipment
* State of actuators stored and updated in Firebase

**4.4 Prayer Time Management**

**4.4.1 Purpose**

Ensures accurate and up-to-date prayer times are available to all users.

**4.4.2 Implementation Details**

* fetchPrayerTimes() method retrieves data from Aladhan API
* Prayer times cached locally for offline access
* Automatic updates daily or upon user request

**4.5 TensorFlow Lite Integration**

**4.5.1 Purpose**

Utilizes machine learning for predictive analysis of sensor data.

**4.5.2 Implementation Details**

* TensorFlow Lite models loaded for flow, temperature, and light intensity prediction
* getFutureFlow(), getFutureTemp(), getFutureLight() methods process sensor data through ML models
* Predictions used for proactive environment management and anomaly detection

**4.6 Local Notifications**

**4.6.1 Purpose**

Provides timely alerts and information to users.

**4.6.2 Implementation Details**

* Uses LocalNotifications class for managing notifications
* Triggered for prayer times, suspicious activities, and important announcements
* Customizable notification content and timing

## **5. Data Models**

**5.1 Sensor Data**

* Temperature
* Motion
* Light intensity
* Water flow

**5.2 Actuator States**

* Fan
* RGB LED
* WS2812

**5.3 Lesson Information**

* Lesson title
* Lesson time

## **6. API Integration**

**6.1 Aladhan API**

* Endpoint: http://api.aladhan.com/v1/timingsByAddress/
* Used for fetching daily prayer times

## **7. Machine Learning Models**

* Flow prediction model
* Temperature prediction model
* Light intensity prediction model

## **8. User Interfaces**

**8.1 Main Navigation**

* Government
* Worshippers
* Imam

**8.2 Government Dashboard**

* Overview of all managed mosques with key metrics
* Quick access buttons to individual mosque management pages
* Data visualization components for system-wide statistic

**8.3 Worshippers View**

* Prominently displayed prayer times for the current day
* Interactive map or list of nearby mosques
* Easy access to lesson schedules and mosque event information

**8.4 Mosque Management Interface**

* Real-time sensor data displayed in easy-to-read cards
* Toggle switches and sliders for actuator controls
* Expandable sections for lessons management and advanced settings

## **9. Security Considerations**

**9.1 Authentication**

* Implement Firebase Authentication for secure user login
* Role-based access control for different user types (government officials, imams, worshippers)

**9.2 Data Encryption**

* Ensure all sensitive data is encrypted both in transit and at rest
* Use secure HTTPS connections for all API communications

**9.3 Input Validation**

* Implement thorough input validation on all user inputs to prevent injection attacks
* Sanitize data before storing in Firebase to prevent stored XSS attacks

**9.4 API Security**

* Use API keys and implement rate limiting for the Aladhan API integration
* Keep API keys and sensitive configurations in secure, non-versioned files

**9.5 Regular Security Audits**

* Conduct periodic security audits of the codebase and infrastructure
* Keep all dependencies and Flutter SDK up to date to patch known vulnerabilities

## **10. Future Enhancements**

**10.1 Multi-language Support**

* Implement localization to support multiple languages
* Allow users to switch languages within the app

**10.2 Advanced Analytics Dashboard**

* Develop a comprehensive analytics dashboard for government officials
* Include predictive analytics using machine learning models

**10.3 Community Features**

* Add a community board for mosque announcements and events
* Implement a donation tracking and management system

**10.4 IoT Expansion**

* Integrate with more IoT devices for enhanced mosque management
* Implement automated routines based on sensor data and prayer times

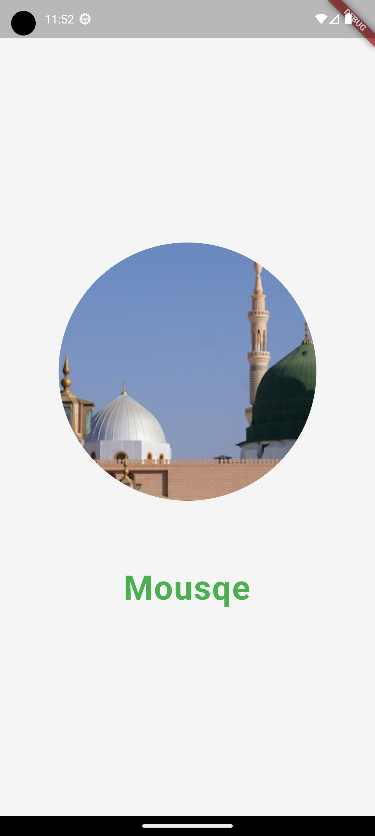
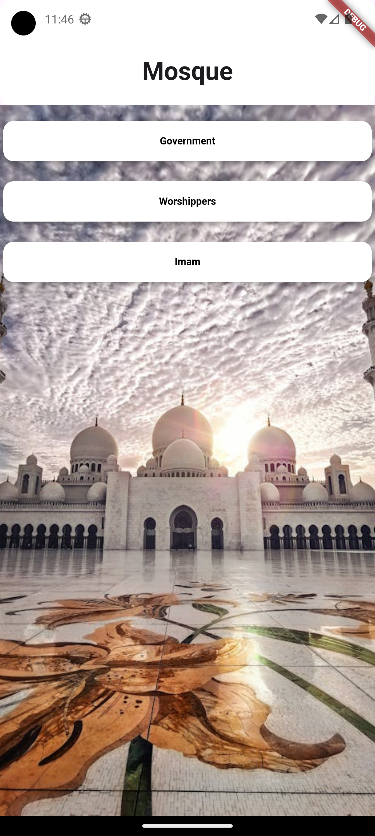
**10.5 Offline Mode**

* Enhance offline capabilities

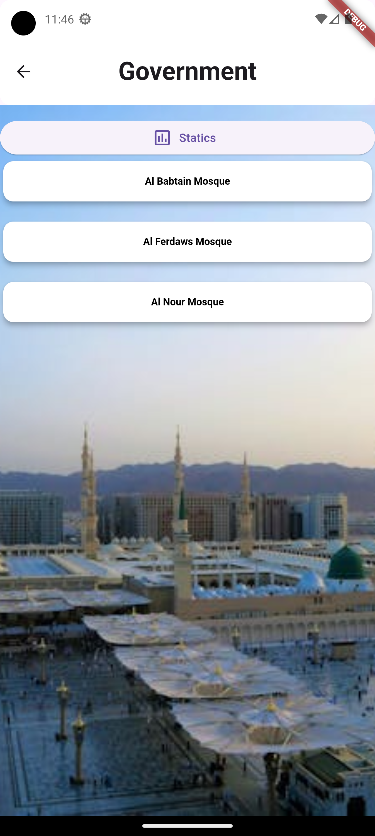
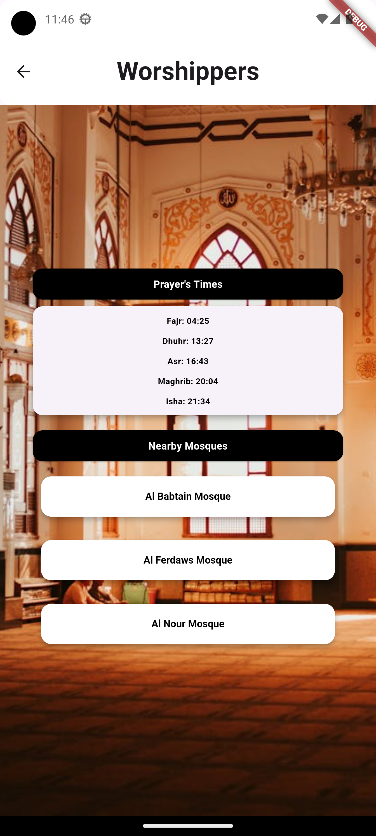
## **11. Troubleshooting**

* Check Firebase connection for real-time data issues
* Verify internet connection for prayer time fetching
* Ensure TensorFlow Lite models are correctly loaded

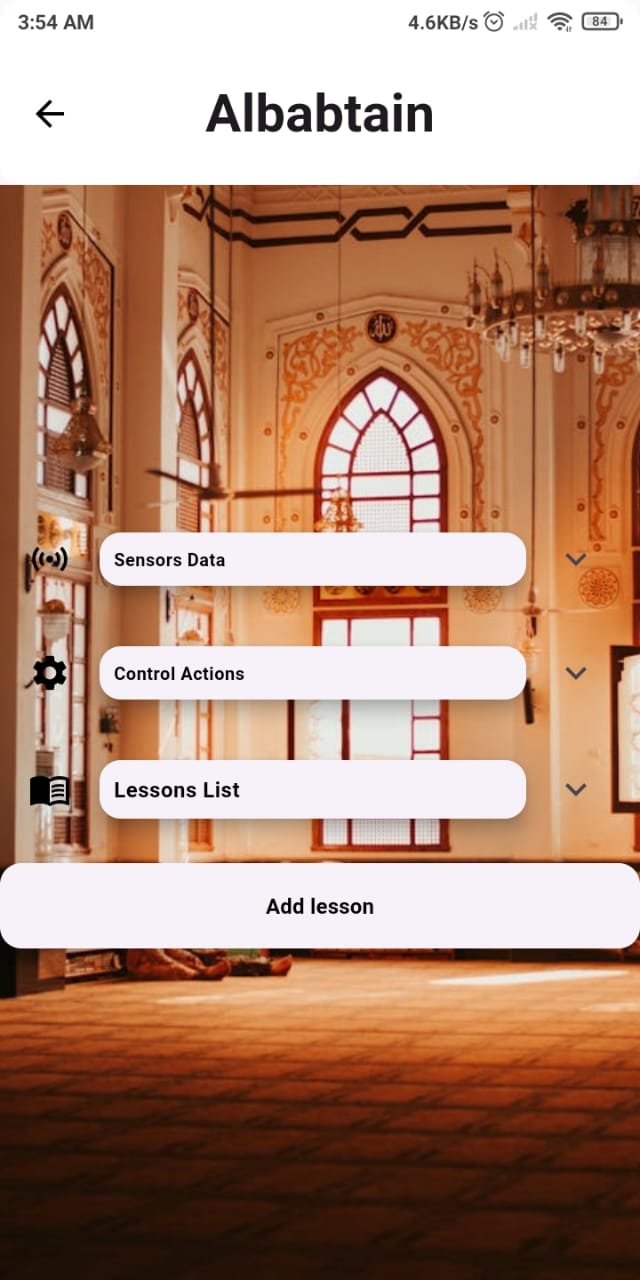
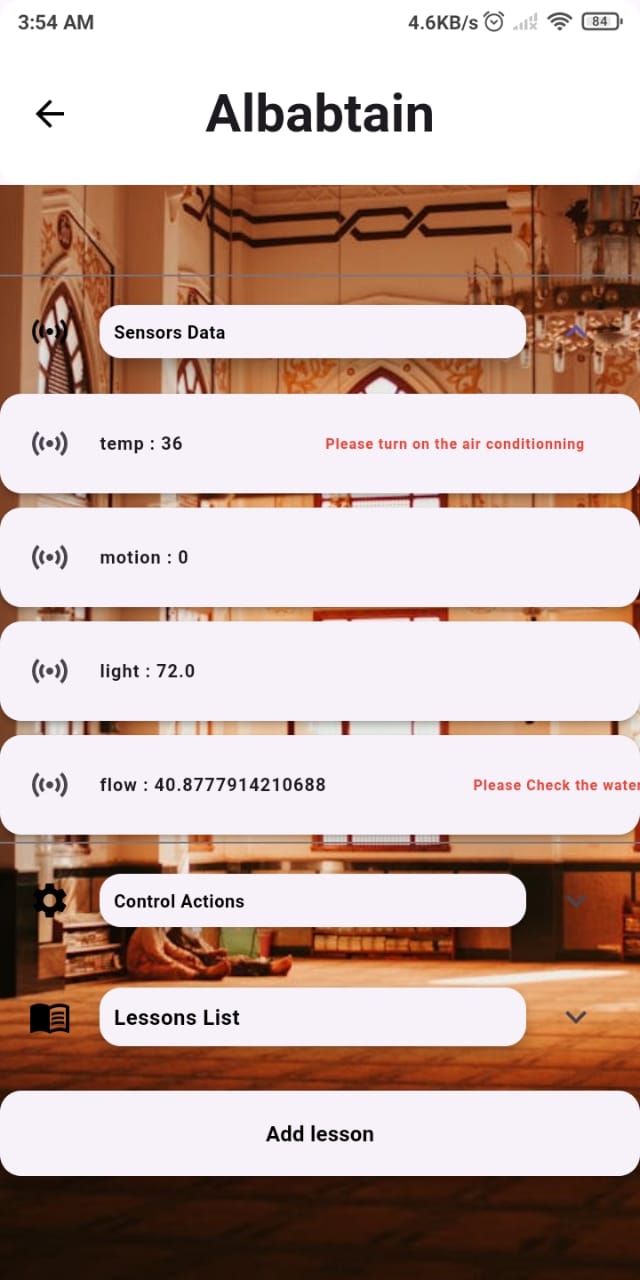
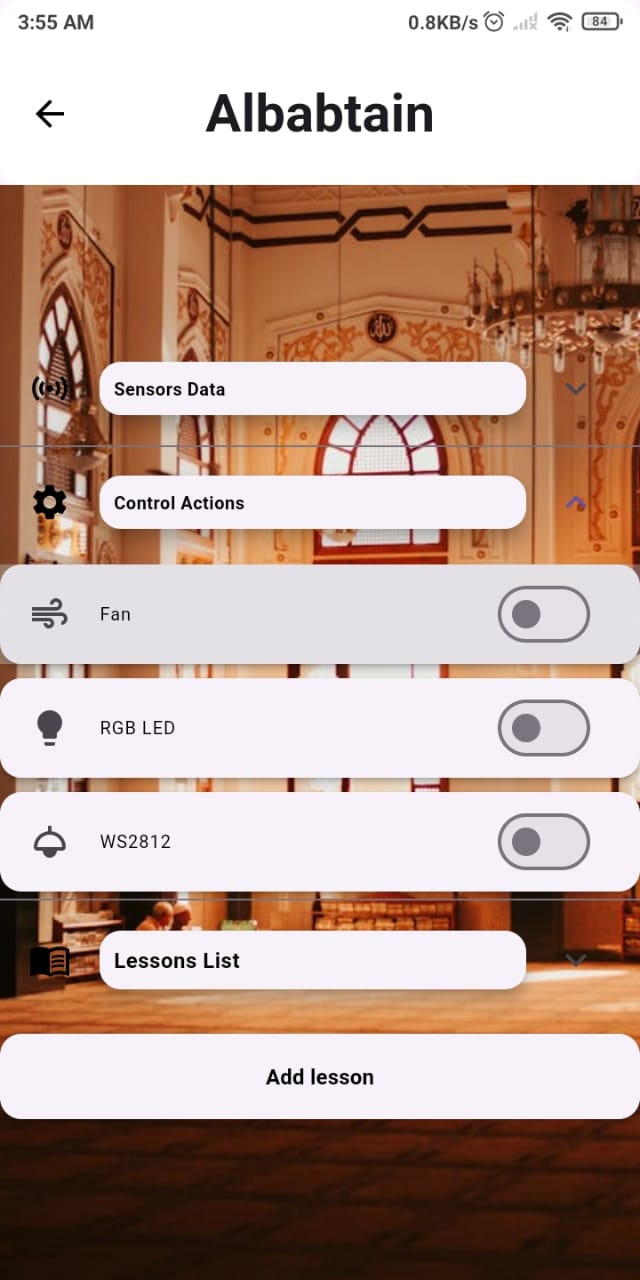
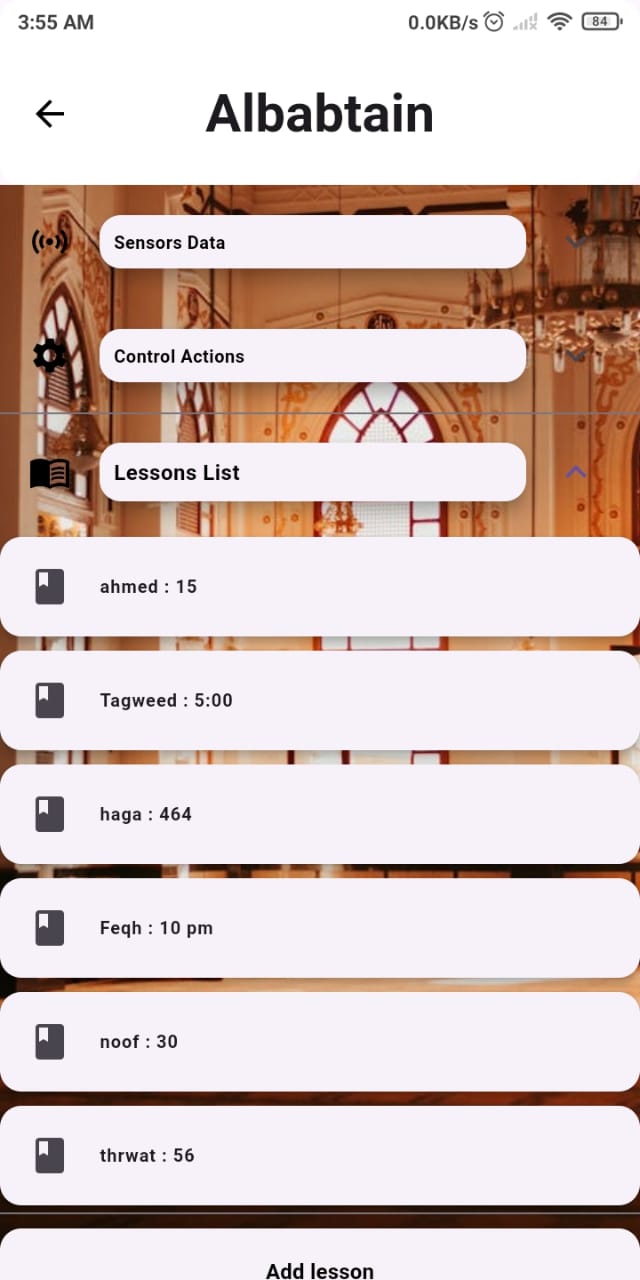
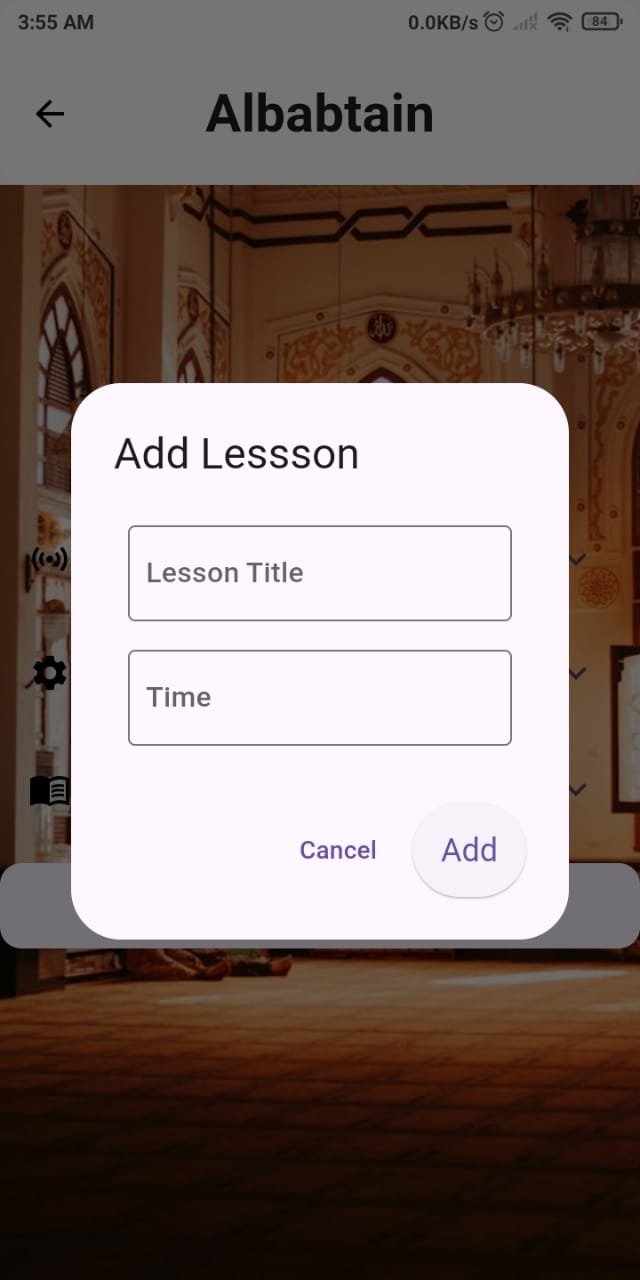
## **Appendix A: Images from the Mobile Application**



**Two images from the application the first one on the left is the intro to the application and the second one is the home page.**



**Two images from the application the first one on the left is the government view and the second one is the worshipers view**



**Five images from the Imam prospective. The imam has full control of the system**