TUNIS BUSINESS SCHOOL
UNIVERSITY OF TUNIS

Dar El Behy

Smart Home Control System

Technical Report

A JavaFX-based Smart Home Application with Voice Control,
Energy Management, and Islamic Prayer Times Integration

Supervised by:

Prof. Ameni Azzouz

Contributors:

Louey Dridi

Dhia Chedly

Achref Msekni

Mohamed Belgacem

Mohamed Amine Saoudi

Academic Year 2025-2026

December 2025

Contents

Chapter 1

Introduction

1.1 Project Overview

Dar El Behy (Arabic for "House of Comfort") is a comprehensive smart home control system developed in Java using the JavaFX framework. The application provides an intuitive interface for managing various smart home devices, monitoring energy consumption, and integrating Islamic prayer times based on the user's location.

1.2 Objectives

The primary objectives of this project are:

- Develop a centralized platform for controlling smart home devices
- Implement voice control capabilities for hands-free operation
- Provide real-time energy monitoring and goal tracking
- Integrate Islamic prayer times with automatic location detection
- Ensure safe access through Parent/Child mode with PIN protection
- Create an extensible architecture for future device additions

1.3 Scope

The system covers:

- 11 types of smart devices (lights, thermostats, TVs, doors, windows, etc.)
- Voice command processing using Web Speech API
- Energy consumption tracking with daily/weekly/monthly goals
- Prayer times fetched from Aladhan API
- User preferences and configuration persistence
- Parent/Child access control modes

Chapter 2

Technologies and Tools

2.1 Programming Language

- **Java 25** - Primary development language
- Object-Oriented Programming paradigm
- Strong typing and extensive standard library

2.2 Frameworks and Libraries

2.2.1 JavaFX 25.0.1

The graphical user interface is built using JavaFX, providing:

- Modern UI controls (Buttons, Labels, TextFields, etc.)
- WebView for embedded web content (speech recognition)
- CSS styling support
- Scene graph architecture
- Animation and effects

2.2.2 JavaFX Web Module

Used for:

- Embedding HTML5 content
- JavaScript bridge for speech-to-text
- HTTP server for local resources

2.3 External APIs

2.3.1 Aladhan Prayer Times API

- Endpoint: `http://api.aladhan.com/v1/timings`
- Provides accurate Islamic prayer times
- Supports multiple calculation methods

2.3.2 IP-API Geolocation Service

- Endpoint: `http://ip-api.com/json`
- Automatic location detection
- Returns latitude, longitude, city, and country

2.3.3 Web Speech API

- Browser-based speech recognition
- Integrated via WebView
- Supports multiple languages

2.4 Development Tools

- **Git** - Version control
- **GitHub** - Repository hosting
- **VS Code** - IDE
- **Batch Scripts** - Build automation (run.bat)

2.5 Data Storage

- **JSON** - Configuration and preferences storage
- **CSV** - Energy history logs
- File-based persistence (no external database required)

Chapter 3

System Architecture

3.1 Layered Architecture

The application follows a layered architecture pattern with clear separation of concerns:

1. **Presentation Layer** - UI components and user interaction
2. **Business Logic Layer** - Controllers, parsers, and analytics
3. **Domain Layer** - Core entities (Home, Room, Devices)
4. **Services Layer** - External API integrations
5. **Persistence Layer** - Data storage and retrieval

3.2 Package Structure

The codebase is organized into 13 packages:

Package	Description
`com.smarthome.core`	Core interfaces and base classes
`com.smarthome.devices`	Device implementations
`com.smarthome.home`	Home and Room entities
`com.smarthome.controller`	Central control and command parsing
`com.smarthome.ui`	Main UI and components
`com.smarthome.ui.components`	Reusable UI panels
`com.smarthome.ui.constants`	UI constants and styles
`com.smarthome.analytics`	Energy goals and tracking
`com.smarthome.ai`	AI-powered suggestions
`com.smarthome.automation`	Automation rules
`com.smarthome.services`	External API clients
`com.smarthome.persistence`	Configuration management
`com.smarthome.security`	Access control and authentication

Table 3.1: Package Structure Overview

3.3 Design Patterns Used

3.3.1 Observer Pattern

Used in `GoalTracker` for notifying UI components when goals are updated:

```java
public interface GoalListener {
    void onGoalProgress(EnergyGoal goal);
    void onGoalAchieved(EnergyGoal goal);
    void onGoalFailed(EnergyGoal goal);
    void onBudgetAlert(EnergyGoal goal);
}
```

3.3.2 Strategy Pattern

Different device types implement the `EnergyConsumer` interface with their own energy calculation strategies.

3.3.3 Facade Pattern

`CentralController` acts as a facade, providing a simplified interface to the complex subsystem of rooms and devices.

3.3.4 Template Method Pattern

`SmartDevice` abstract class defines the template for device operations:

```java
public abstract class SmartDevice implements Controllable {
    public abstract void turnOn();
    public abstract void turnOff();
    public abstract String getStatus();
}
```

Chapter 4

Core Components

4.1 SmartDevice Hierarchy

The base `SmartDevice` class is extended by 11 device types:

Device	Energy Consumer	Key Features
Light	Yes	Brightness, Color control
Thermostat	Yes	Temperature, HVAC control
SmartTV	Yes	Power, Standby modes
Oven	Yes	Temperature, Timer
Door	No	Open, Close, Lock, Unlock
GarageDoor	No	Open, Close, Lock
SmartWindow	No	Open percentage
SmokeDetector	No	Alarm threshold
GasDetector	No	Gas level detection
MotionSensor	No	Motion detection
Fridge	Yes	Temperature control

Table 4.1: Device Types and Capabilities

4.2 CentralController

The main orchestrator for device operations, providing:

- Room-based light control
- Thermostat management
- Energy consumption calculation
- Access control enforcement
- Device status reporting

4.3 CommandParser

Processes natural language voice commands using regex patterns:

```
// Pattern examples:
"turn on all lights in living room"
"set brightness in bedroom to 70"
"set thermostat in kitchen to 22"
"activate low power mode"
```

4.4 GoalTracker

Manages energy consumption goals with features:

- Daily, Weekly, Monthly goal periods
- Progress tracking and alerts
- Achievement notifications
- Budget alerts at 80% threshold

Chapter 5

User Interface

5.1 Main Layout

The application uses a `BorderPane` layout with:

- **Header (Top)** - Title, Parent/Child mode
- **Left Panel** - Voice control interface
- **Center Panel** - Device controls organized by room
- **Right Panel** - Energy dashboard, Goals, Suggestions, Prayer times
- **Bottom Panel** - Activity log console

5.2 Resizable Panels

The main content area uses a `SplitPane` allowing users to resize sections horizontally by dragging dividers.

5.3 Parent/Child Mode

- **Parent Mode** - Full access to all devices and settings
- **Child Mode** - Restricted access with child-friendly UI
- PIN protection (default: 1234) for Parent Mode access

5.4 UI Components

Component	Purpose
`EnergyDashboard`	Real-time energy consumption display
`GoalPanel`	Energy goal management with progress bars
`SuggestionPanel`	AI-powered energy saving suggestions
`PrayerTimePanel`	Islamic prayer times display
`DeviceControlPanel`	Individual device controls

Table 5.1: UI Component Overview

Chapter 6

Features

6.1 Voice Control

The application integrates Web Speech API through an embedded WebView:

1. User clicks microphone button or speaks wake word
2. Browser Speech Recognition converts speech to text
3. JavaScript bridge sends text to Java
4. CommandParser processes the command
5. CentralController executes the action
6. Result is displayed in console

6.2 Energy Management

6.2.1 Real-time Tracking

Energy consumption is tracked every 5 minutes:

- Devices implementing `EnergyConsumer` report their usage
- Total kWh is calculated and recorded
- Historical data is stored in CSV files

6.2.2 Energy Goals

Users can create consumption goals:

- Daily, Weekly, or Monthly periods
- Target kWh setting
- Visual progress tracking
- Budget alerts at 80% threshold
- Goal deletion capability

6.2.3 Energy Reports

Exportable CSV reports include:

- Date range and total usage
- Average daily consumption
- Peak and lowest usage days
- Daily breakdown

6.3 Prayer Times Integration

Automatic Islamic prayer times based on user location:

1. Geolocation service detects user's location via IP
2. Coordinates sent to Aladhan API
3. Five daily prayer times (Fajr, Dhuhr, Asr, Maghrib, Isha)
4. Next prayer is highlighted in the UI

6.4 AI Suggestions

The `SuggestionEngine` provides intelligent recommendations:

- Energy-saving tips based on usage patterns
- Device optimization suggestions
- Peak usage alerts

6.5 Configuration Persistence

All settings are automatically saved and restored:

- Home configuration (rooms, devices, states)
- User preferences (theme, language, cost/kWh)
- Energy goals
- Auto-save every 5 minutes (if enabled)

Chapter 7

UML Diagrams

7.1 Use Case Diagram

The system supports four actors:

- **Parent User** - Full system access
- **Child User** - Limited device access
- **System/Timer** - Automated background tasks
- **External APIs** - Prayer times and geolocation

Key use cases include:

- Device Control (7 use cases)
- Voice Control (2 use cases)
- Energy Management (5 use cases)
- Configuration (4 use cases)
- Services (2 use cases)

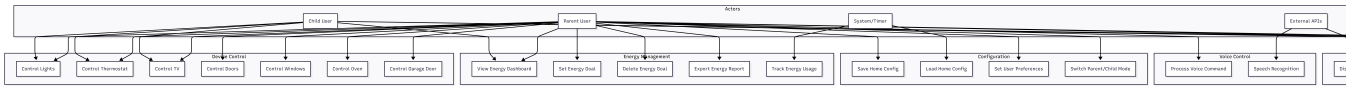

Figure 7.1: Use Case Diagram

7.2 Class Diagram

The class diagram shows:

- Core interfaces: `Controllable`, `EnergyConsumer`, `Schedulable`
- Abstract base class: `SmartDevice`
- 8 concrete device classes
- `Home` and `Room` composition
- Controller and service classes
- Persistence layer components

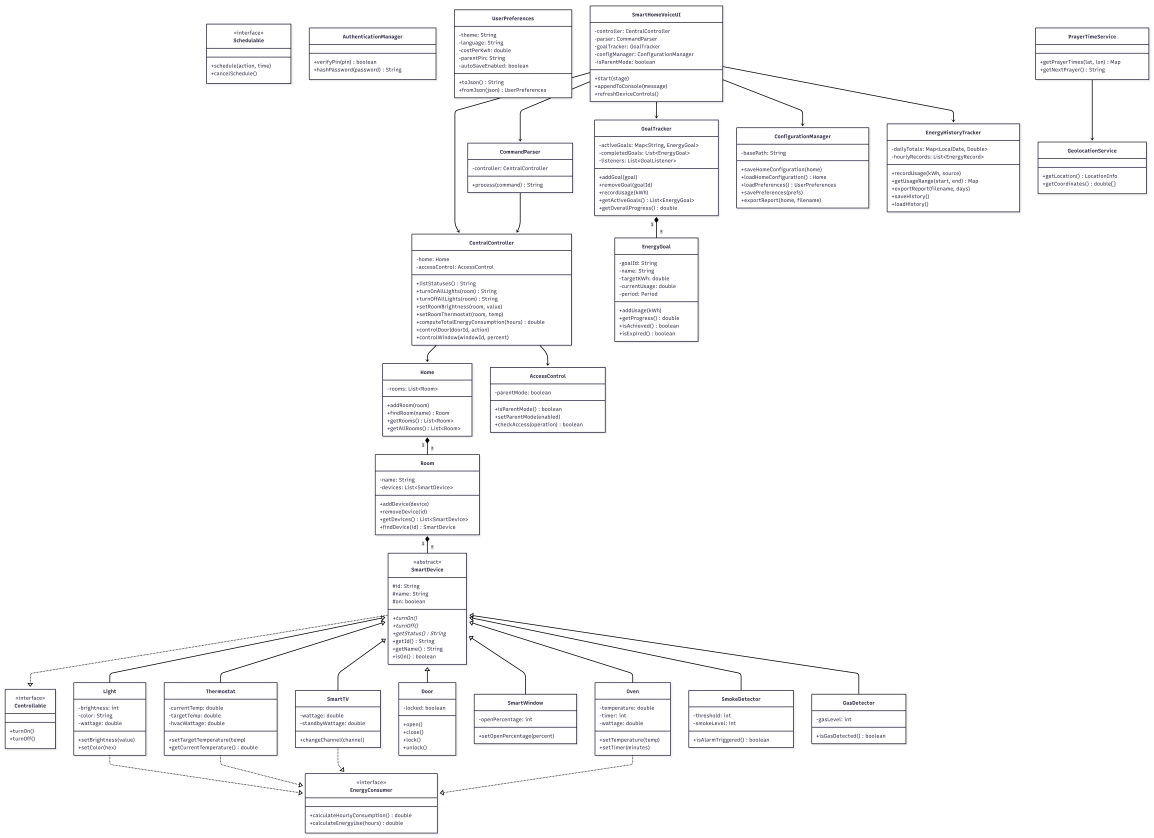

Figure 7.2: Class Diagram

7.3 Key Sequence Diagrams

7.3.1 Voice Command Processing

1. User speaks command
2. WebView processes speech-to-text
3. CommandParser matches pattern
4. CentralController executes action
5. Response displayed in console

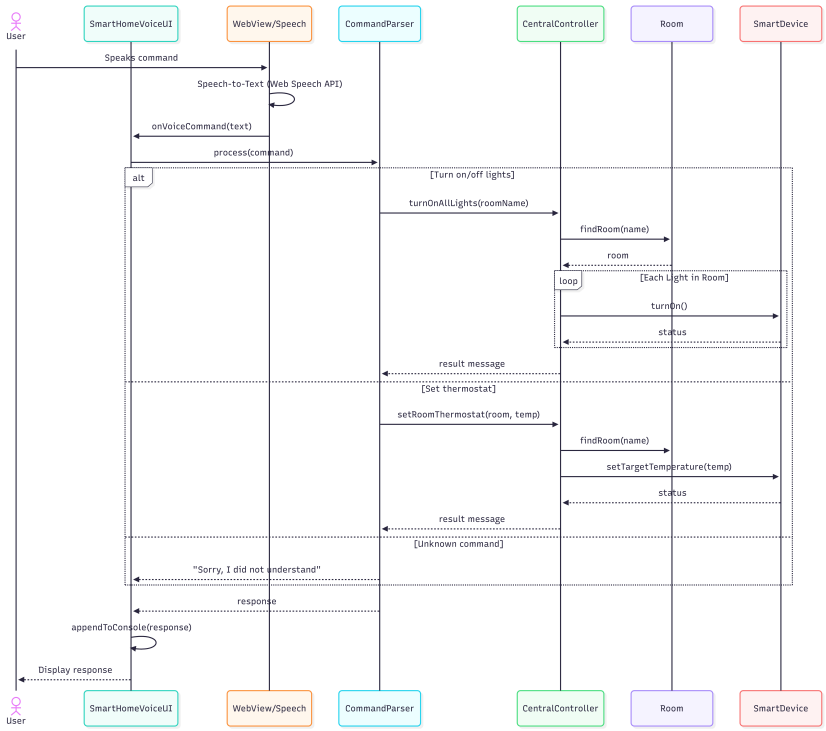

Figure 7.3: Voice Command Processing Sequence Diagram

7.3.2 Energy Tracking (5-minute cycle)

1. Timeline triggers onTick()
2. computeTotalEnergyConsumption() called
3. Each EnergyConsumer device reports usage
4. EnergyHistoryTracker records data
5. GoalTracker updates progress and checks alerts

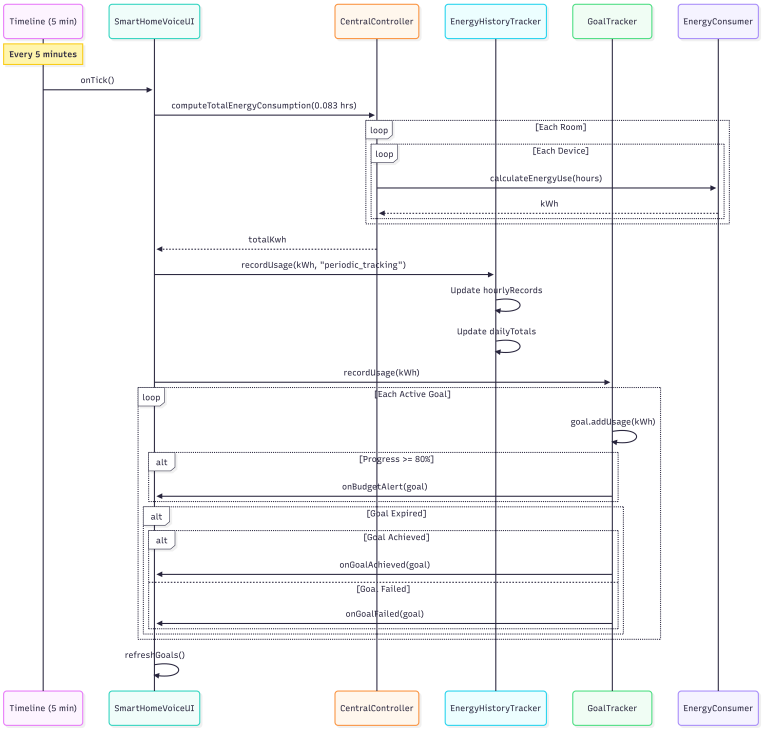

Figure 7.4: Energy Tracking Sequence Diagram

7.3.3 Prayer Time Loading

1. PrayerTimePanel requests location
2. GeolocationService fetches coordinates
3. PrayerTimeService queries Aladhan API
4. Prayer times displayed in UI

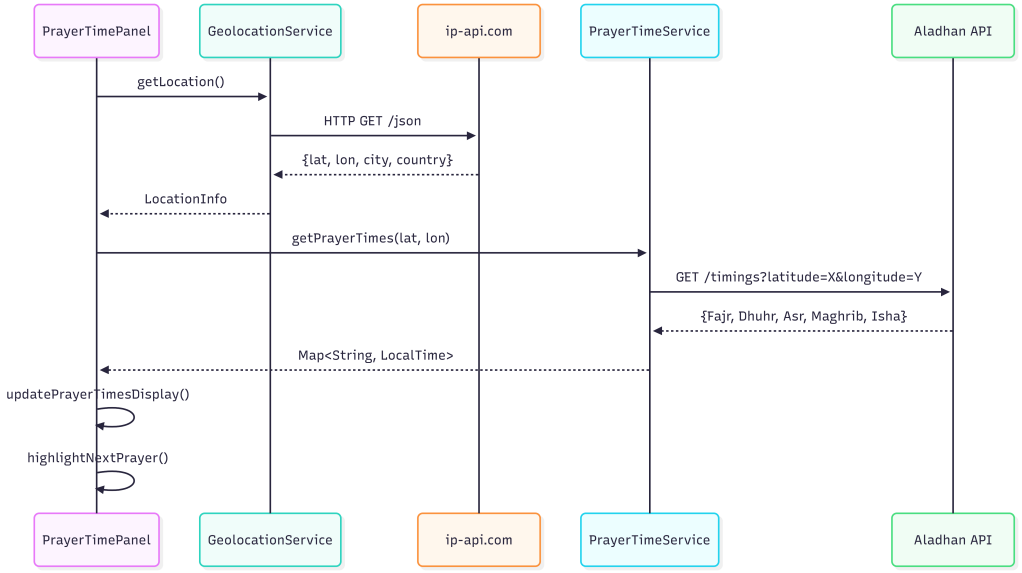

Figure 7.5: Prayer Time Loading Sequence Diagram

7.4 System Architecture Diagram

Five-layer architecture:

1. Presentation Layer (SmartHomeVoiceUI, Panels, WebView)
2. Business Logic Layer (CentralController, CommandParser, GoalTracker)
3. Domain Layer (Home, Room, SmartDevices)
4. Services Layer (PrayerTimeService, GeolocationService)
5. Persistence Layer (ConfigurationManager, EnergyHistoryTracker)

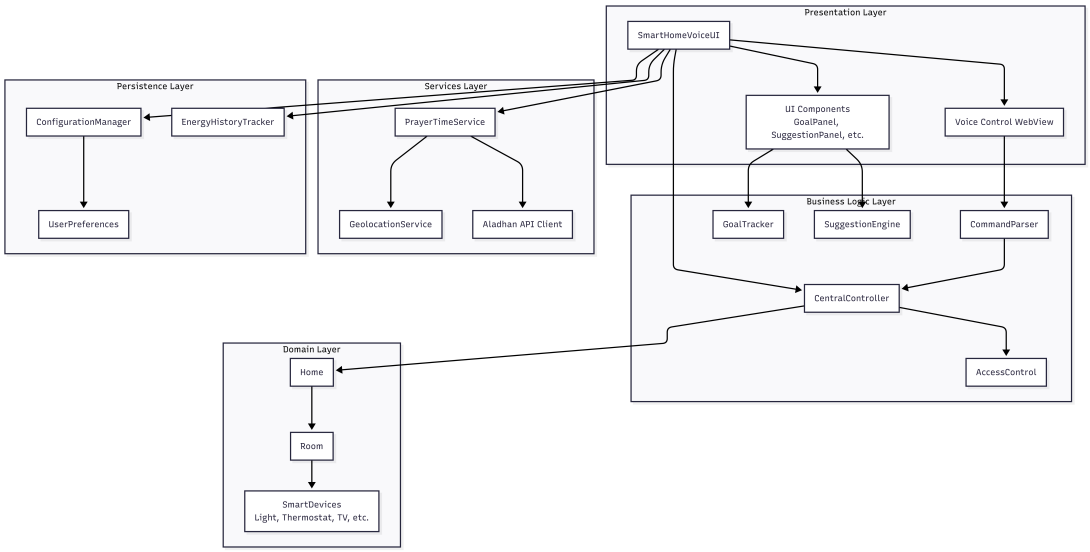

Figure 7.6: System Architecture Diagram

Chapter 8

Security Considerations

8.1 Access Control

- Parent/Child mode separation
- PIN-protected Parent Mode access
- Restricted device control in Child Mode
- Dangerous devices (oven, doors, windows) require Parent access

8.2 Rate Limiting

The `RateLimiter` class prevents abuse:

- Limits API calls per minute
- Prevents brute-force PIN attacks

8.3 Future Improvements

- PIN hashing (currently stored in plain text)
- Multi-user authentication
- Encrypted configuration files
- OAuth integration for external APIs

Chapter 9

Testing

9.1 Unit Tests

The project includes JUnit 5 test cases:

- `CentralControllerTest` - Controller operations
- `CommandParserTest` - Voice command parsing
- `RoomHomeTest` - Room and Home entity tests

9.2 Test Coverage Areas

- Device state management
- Energy consumption calculations
- Command pattern matching
- Room/Device lookups

9.3 Manual Testing

The following scenarios require manual testing:

- Voice command recognition
- UI responsiveness
- Prayer times accuracy
- Energy report generation

Chapter 10

Installation and Usage

10.1 Prerequisites

- Java Development Kit (JDK) 25
- JavaFX SDK 25.0.1
- Internet connection (for prayer times and geolocation)

10.2 Installation Steps

1. Clone the repository from GitHub
2. Download and extract JavaFX SDK 25.0.1
3. Update paths in `run.bat` if necessary
4. Run `./run.bat` to compile and launch

10.3 Configuration

Default paths in `run.bat`:

```
set JAVA_HOME=C:\Program Files\Java\jdk-25
set JAVAFX_PATH=C:\Program Files\Java\javafx-sdk-25.0.1\lib
```

10.4 Voice Commands Reference

Command	Action
"Turn on all lights in [room]"	Activates all lights in room
"Turn off lights in [room]"	Deactivates all lights in room
"Set brightness in [room] to [0-100]"	Adjusts light brightness
"Set thermostat in [room] to [temp]"	Sets temperature in degrees
"Total energy consumption"	Shows current energy usage
"Activate low power mode"	Reduces all lights to 30%
"List statuses"	Shows all device statuses

Table 10.1: Voice Commands Reference

Chapter 11

Conclusion

11.1 Summary

Dar El Behy successfully demonstrates a comprehensive smart home control system with:

- Intuitive JavaFX user interface
- Voice control via Web Speech API
- Real-time energy monitoring and goal tracking
- Islamic prayer times integration
- Robust Parent/Child access control
- Extensible architecture for future devices

11.2 Achievements

- 11 different smart device types implemented
- 13 well-organized packages
- Voice command processing with natural language
- External API integrations (Aladhan, IP-API)
- Persistent configuration and energy history

11.3 Future Work

Potential enhancements include:

- Mobile application companion
- Cloud synchronization
- Machine learning for usage prediction
- Integration with real IoT devices
- Multi-language support
- Dark mode theme (partially implemented)
- Enhanced security with password hashing

Appendix A

Project Structure

```
darelbehi-app/
+-- java/com/smarthome/
|   +-- ai/                 # AI suggestion engine
|   +-- analytics/          # Energy goals tracking
|   +-- automation/         # Automation rules
|   +-- controller/         # Central control logic
|   +-- core/               # Interfaces and base classes
|   +-- devices/            # Device implementations
|   +-- home/               # Home and Room entities
|   +-- persistence/        # Configuration storage
|   +-- scheduler/          # Task scheduling
|   +-- security/           # Access control
|   +-- services/           # External API clients
|   +-- ui/                 # JavaFX UI components
|       +-- components/     # Reusable panels
|       +-- constants/      # UI constants
+-- resources/
|   +-- icons/              # Application icons
|   +-- voiceCommand.html   # Speech recognition page
|   +-- styles.css          # UI styles
+-- config/                 # Saved configurations
+-- data/                   # Energy history
+-- docs/                   # Documentation
+-- run.bat                 # Build and run script
+-- README.md               # Project readme
```

Appendix B

Configuration File Formats

B.1 home_config.json

```
{
  "rooms": [
    {
      "name": "living room",
      "devices": [
        {
          "type": "Light",
          "id": "L1",
          "name": "Living Light 1",
          "on": false,
          "properties": {
            "wattage": "10.0",
            "brightness": "30",
            "color": "#FFFFFF"
          }
        }
      ]
    }
  ]
}
```

B.2 energy_goals.json

```
[
  {
    "goalId": "goal_1",
    "name": "Monthly Limit",
    "targetKWh": 100.0,
    "currentUsage": 45.5,
    "period": "MONTHLY",
    "startDate": "2024-12-01"
  }
]
```