Infosys | Springboard

VIRTUAL INTERNSHIP 6.0

SMART HOME ENERGY MANAGEMENT SYSTEM

Module 1 – Authentication & Role Management

BY Thanuja Vanjarapu
BATCH-10

MEANING OF THE TITLE

The Smart Home Energy Management System is a technology-driven platform designed to help homeowners intelligently monitor, control, and optimize the electricity usage of all electrical appliances inside the house.

It transforms a normal home into a smart, energy-efficient environment by providing features like real-time usage tracking, automation, device control, analytics, and smart recommendations.
The system ensures that users are always aware of how much electricity is being consumed, which devices are consuming the most power, and how they can reduce wastage.

It also automates many daily tasks—like turning devices ON/OFF at specific times—so that electricity is used only when required.
Additionally, it offers alerts for over-usage, safety risks, and high bill predictions, helping users save money and prevent energy wastage.

Overall, the project aims to provide a central intelligent platform that improves comfort, reduces electricity bills, and promotes smarter energy usage inside homes.

“Our project title reflects two key ideas:

1. Smart Home Energy – managing home energy consumption intelligently, using digital automation.
2. Management System – meaning we are building a software solution with user roles, authentication, dashboards, and controlled access.

Module 1 ensures that only valid users can log in and operate this system safely.”

PROBLEMS IN EXISTING SYSTEMS

1. No Real-Time Monitoring
People cannot see how much electricity each device is consuming.
They only know the total bill at the end of the month.

2. Devices Left ON Accidentally
Fans, lights, AC, geysers are often left ON unknowingly, leading to wastage of electricity.

3. No Centralized Control
Each appliance has to be operated manually.
There is no single dashboard to control all devices from one place.

4. No Usage Analytics
Existing systems do not show:

- Which device consumes the most
- Daily/weekly usage
- Peak hours
- Cost estimation

So users cannot understand their consumption patterns.

5. No Automation or Scheduling
Devices cannot be set to:

- Turn off automatically
- Turn on at a specific time
- Follow energy-saving rules

Everything is manual → increases wastage.

6. No Alerts or Notifications
Current systems do not warn the user when:

- Usage is too high
- A device is malfunctioning
- Bills may exceed the budget

This leads to surprise high bills.

7. No Recommendations for Energy Saving
Users do not receive smart suggestions like:

- “AC uses too much energy at 2 PM”
- “Try reducing heater usage”

Hence, energy efficiency is low.

8. No Role-Based Access
Everyone in the house has equal access.
No user authentication → low security.

PROPOSED SYSTEM (OUR DESIGN)

1. Secure Login & Role-Based Access
- Users log in with username & password.
- Role-based access for Admin and User.
- Ensures only authorized people can control devices.

2. Central Dashboard for All Devices
- Shows all connected appliances: AC, Fan, Lights, Heater, etc.
- Displays real-time ON/OFF status.
- Users can operate devices from one place.

3. Real-Time Energy Monitoring
- Tracks energy consumption of each device continuously.
- Shows live units used and total home usage.
- Helps users identify high-energy devices.

4. Analytics & Energy Reports

Provides detailed charts and insights:
- Daily/Weekly/Monthly usage
- Device-wise consumption
- Peak usage times
- Estimated bill
- This helps users control unnecessary usage.

5. Automation & Scheduling

Users can create rules like:
- “Turn off AC at 11 PM”
- “Turn on light at 6 PM”

The system executes these automatically → reduces wastage.

6. Alerts & Notifications
- High usage alert
- Device overheating alert
- Monthly bill warning
- Schedule reminders
- This improves safety and energy efficiency.

7. Recommendation System

The system analyzes user behavior and gives suggestions:
- Best time to use appliances
- Tips to reduce consumption
- Identify high-consuming appliances
- Recommend energy-efficient options

8. Efficient Backend Architecture
- Controller Layer
- Service Layer
- Repository Layer
- Model Layer

SYSTEM CONFIGURATION

Software Requirements

- **Operating System:** Windows 10/11 or Linux
  Runs JDK, STS, MySQL smoothly
- **Backend:**
  - Java 17
  - Spring Boot (REST APIs, controllers, services)
  - Spring MVC
  - Hibernate/JPA
    Used for fast development and clean architecture
- **Frontend:**
  - HTML5, CSS3, Bootstrap
  - Thymeleaf
    Used for UI + binding backend data
- **Database:**
  - MySQL + MySQL Workbench
    Stores users, devices, logs, schedules
- **Tools:**
  - Maven (dependency management)
  - STS / IntelliJ (IDE)
  - Postman (API testing)
  - Chrome/Edge (browser)

Hardware Requirements

- **Client Device:** Laptop / Desktop / Smartphone
  Used to access the web app
- **Server/Developer Machine:**
  - Processor: Intel i3 or higher
  - RAM: 4–8 GB
  - Storage: 20+ GB
    Runs Spring Boot + MySQL

System Flow Summary

- User logs in → Controller
- Logic processed → Service
- Data stored/retrieved → Repository
- Data stored in MySQL → Model Entities
- Output shown in UI via Thymeleaf

ACTIVITIES OF MODULE 1

1. User Enters Login Details
- User types username and password on the login page.
- Input is sent to the backend through AuthController.

2. Controller Receives the Login Request
- AuthController collects login credentials.
- It forwards the data to AuthService for validation.

3**. Service Validates User Credentials**
- AuthService checks:
  - Whether the username exists
  - Whether the password is correct (after encryption check)
- If credentials do not match → Error message is returned.

4**. Password Encryption Check**
- The entered password is encrypted.
- It is compared with the stored encrypted password in the database.

5. Database Lookup
- UserRepository fetches the user record using:
  - findByUsername()
- It returns:
  - userId
  - encrypted password
  - role
  - isActive status

6. Role Verification
- After credentials match, the system checks the user's role:
  - Admin → gets full access
  - User/Resident → gets limited access

7. Session / Token Creation
- Upon successful login:
  - A secure session or token is generated
  - User is considered "authenticated"

8. Redirect to Dashboard (Role-Based UI)
- Admin → Dashboard with Device Management, Reports, Scheduling
- User → Limited dashboard (device control & usage only)

9**. Invalid Login Handling**

If:
- Username is incorrect
- Password is wrong
- Account is inactive

Then the system returns:
- "Invalid Credentials"
- "User Not Found"

10. Logout Activity
- User clicks Logout
- Session/token is cleared
- User is returned to the login page

ACTIVITY DIAGRAM

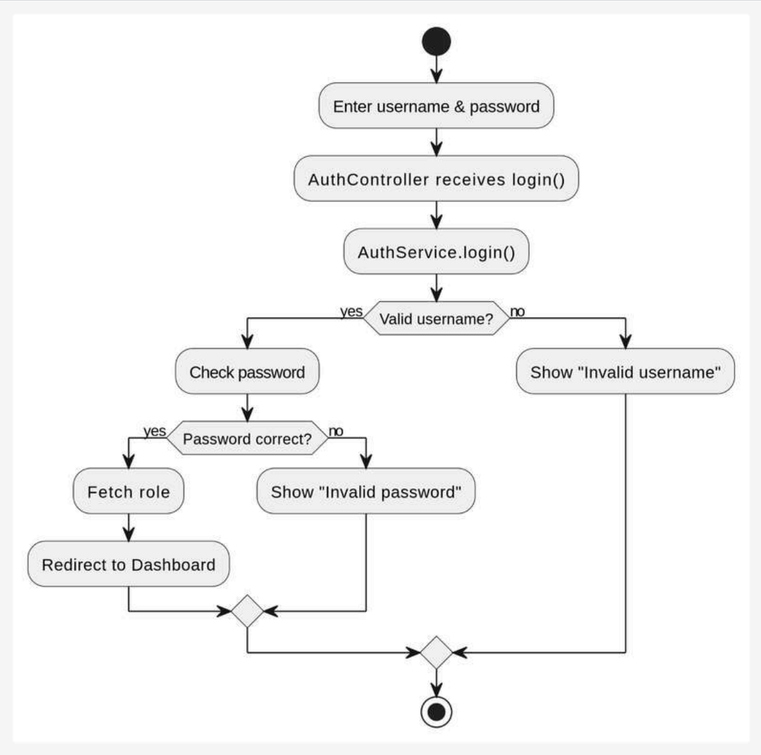


Figure (i) i

SEQUENCE DIAGRAM

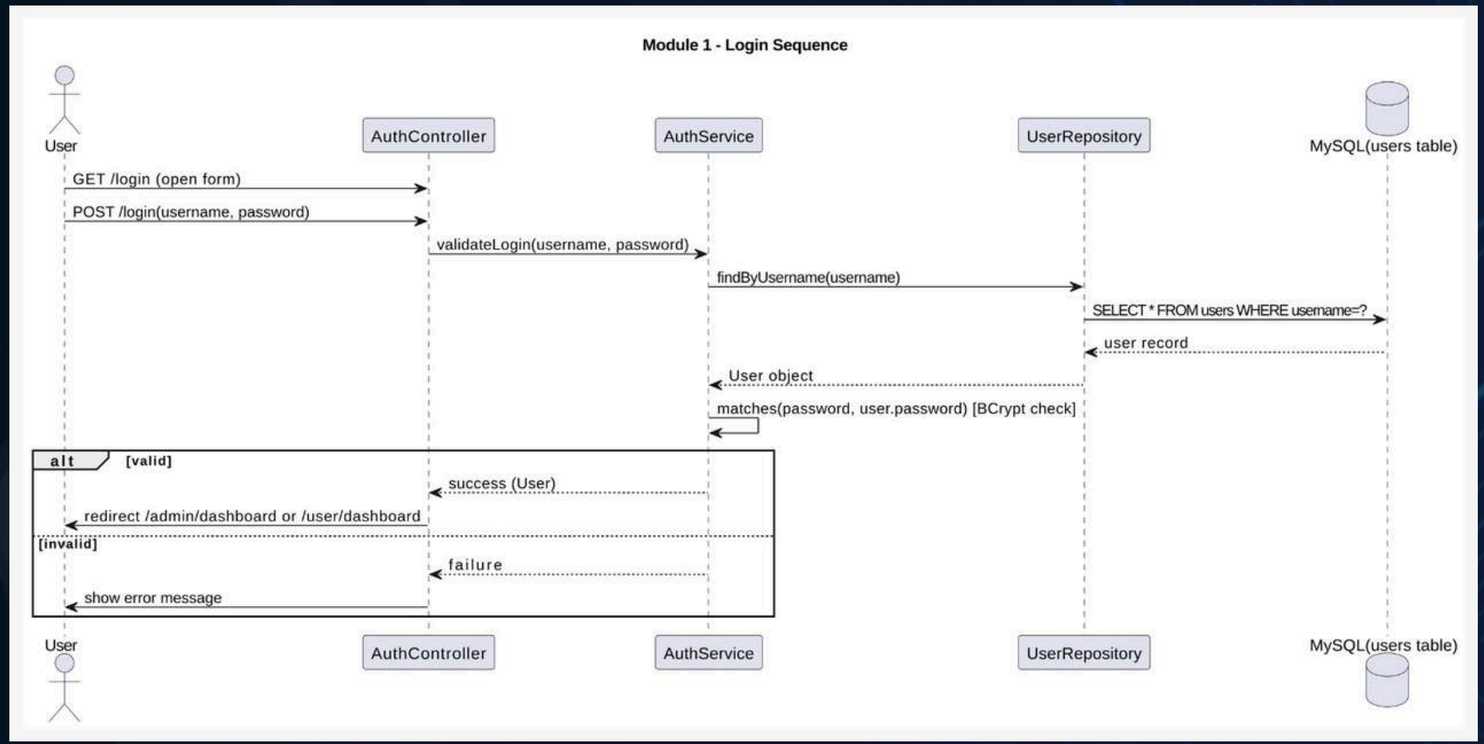


Figure (ii) i

CLASSES IN MODULE 1

1. User (Model / Entity Class)
Purpose: Represents the user record stored in the database.
Attributes
- userId : Long
- username : String
- email : String
- password : String
- role : String
- isActive : Boolean

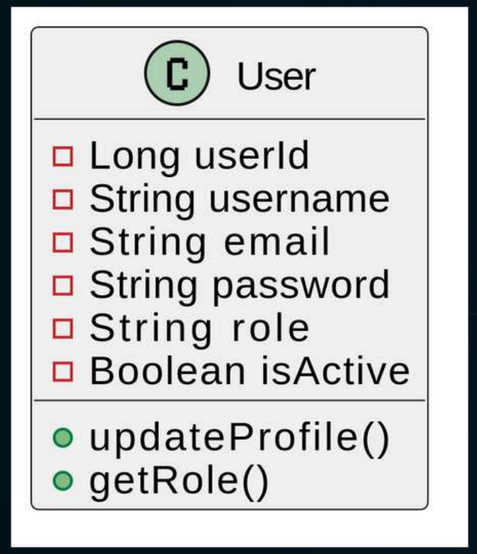
Methods
- updateProfile()
- getRole()

2. AuthController (Controller Class)
Purpose: Handles incoming requests from the UI.
Methods
- login(username, password)
- logout()
- register(User)

3. AuthService (Service Class)
Purpose: Contains the business logic for authentication.
Methods
- validateLogin(username, password) : boolean
- encryptPassword(password) : String
- generateToken(User) : String
- validateToken(token) : boolean

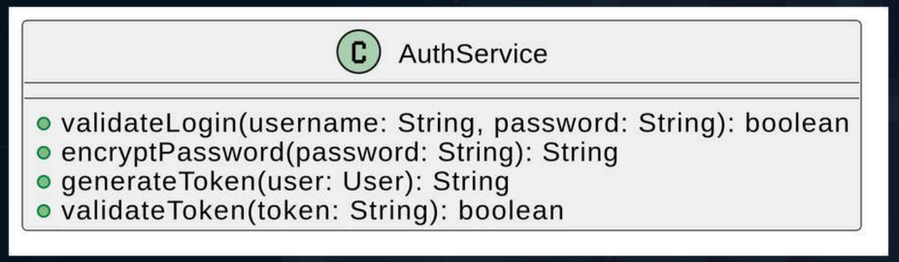
4. UserRepository (Repository Class)
Purpose: Interacts with the database using JPA/Hibernate.
Methods
- findByUsername(username) : User
- saveUser(User)
- updateUser(User)

CLASS DIAGRAMS

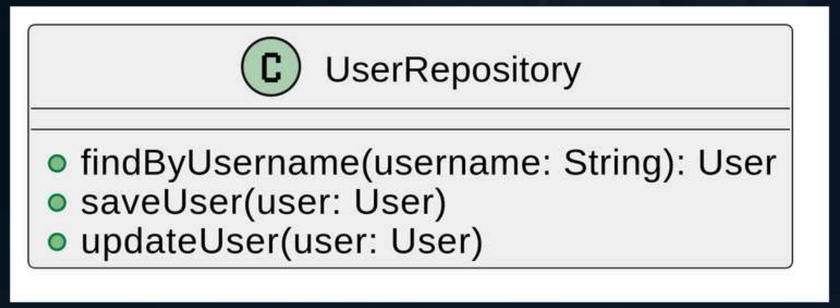


Figure(i)

Figure(ii)



Figure(iii)



Figure(iv)

Figure(v)

DB SCHEMA FOR MODULE 1

```
CREATE TABLE users (
    user_id BIGINT PRIMARY KEY AUTO_INCREMENT,
    username VARCHAR(50) UNIQUE NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    role VARCHAR(30) NOT NULL,      -- ADMIN / USER / MANAGER
    is_active BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Why only one table in Module 1?

Because in Spring Boot architecture:

✓ Only Model (Entity) classes create tables.

Thus only the User class becomes a table.


THANK YOU !