SMART-JOB

1. Overview.

The **Smart-Job Application** is a full-stack web platform that allows job seekers and employers to interact through job listings, applications, interviews, and messaging. The system is designed to be scalable, modular, and secure using industry-standard design patterns and Spring Boot architecture principles.

The platform supports:

- User registration and login with role-based access.
- Job listing and advanced search.
- Real-time notifications and email alerts.
- Applying for available jobs.

2. Design Patterns Used

Below is a detailed list of design patterns applied in the application and their specific use cases:

Pattern	Applied In	Purpose / Problem Solved
Factory	Auth Service	Dynamically instantiate different types of users (Admin, Recruiter, Seeker) at runtime.
Strategy	Auth, Search, Job, Admin	Enable interchangeable login strategies, filtering algorithms, and report generation.
Chain of Responsibility	Auth Service	Break login into modular steps: validation, authentication, role-checking.
Singleton	JWT Utility, Security	Maintain one instance of stateless services (e.g., token utility) throughout the application.

Observer	Job Service, Notification	Decouple event publishing (job posted) from consumers (notifications, email).
Repository	All services (Persistence)	Centralize and abstract access to database using Spring Data JPA.
Adapter	Email Service	Wrap third-party email APIs like SendGrid into app-specific interfaces.
State	Application Service	Handle application lifecycle: Applied → In Review → Shortlisted → Hired/Rejected.

3. System Architecture

3.1 High-Level Architecture

$$\begin{tikzcd}
\text{Browser (Client)} \arrow[r, leftrightarrow] & \text{Spring Boot Controllers} \arrow[r, leftrightarrow] \arrow[d, leftrightarrow] & \text{Service Layer} \arrow[r, leftrightarrow] & \text{Repositories} \arrow[r, leftrightarrow] & \text{PostgreSQL} \\
& \text{Notification System} \arrow[d, leftrightarrow] & & & \\
& \text{External Services (Email)} & & &
\end{tikzcd}$$

3.2 Components

Component	Description
Frontend	Thymeleaf templates rendered by Spring Boot; provides the UI layer.
Backend	RESTful controllers + business services written in Java/Spring Boot.
Authentication	Spring Security + JWT for token-based stateless auth.
Database	PostgreSQL; relational schema for all core entities.
Notification System	Event-driven alert system for in-app and email alerts.
Email Integration	External email API (SendGrid or SMTP) for transactional messages.

3.3 Data Flow

User Registration & Login

1. User submits credentials (POST /auth/login)
2. AuthService verifies identity using Chain of Responsibility.
3. If valid, JWT is generated and returned.
4. JWT is included in subsequent requests (as Proxy guards).

Job Posting

1. Recruiter submits job post via UI.
2. JobService persists it via Repository.
3. Observer pattern triggers NotificationService.
4. EmailService (Adapter) sends external alerts.

Search Jobs

1. User applies filters (location, title, type).
2. Strategy pattern executes filtering algorithm.
3. Results are returned and displayed.

Apply to Job

1. ApplicationService creates application.
2. State Pattern manages lifecycle: Applied → Hired.
3. NotificationService alerts recruiter.

4. Component/Service Breakdown

Auth Service

- **Purpose**: Manage registration, authentication, and authorization.
- **Design Patterns**:
    - **Factory**: Create different user types.
    - **Strategy**: Allow OAuth or traditional login in future.
    - **Chain of Responsibility**: Break down login process.
    - **Singleton**: Token service maintains one instance across app.

Job Service

- **Purpose**: Manage CRUD for job posts, implement search and filtering.
- **Design Patterns**:
  - **Observer**: Notify users when new jobs are posted.
  - **Strategy**: Filter jobs using interchangeable filter strategies.

Notification Service

- **Purpose**: Real-time or asynchronous alerts on system events.
- **Design Patterns**:
  - **Observer**: Subscribed to Job, Interview, Application events.
  - **Repository**: Persist notification history.

Email Service

- **Purpose**: Send confirmation and notification emails.
- **Design Patterns**:
  - **Adapter**: Integrate external services (SMTP/SendGrid).
  - **Observer**: Triggered via NotificationService. **Search & Filter Module**
- **Purpose**: Provide flexible job search experience.
- **Design Pattern**:
  - **Strategy**: Filter by category, location, experience, etc.

Application Service

- **Purpose**: Allow job seekers to apply and track applications.
- **Design Pattern**:
  - **State Pattern**: Manages states such as Applied, Rejected, Hired.

5. Technology Stack

Layer	Technology / Tool
Backend	Java, Spring Boot, Spring Security
Frontend	Thymeleaf, HTML, CSS, Bootstrap
Database	PostgreSQL

ORM	Spring Data JPA, Hibernate
Authentication	JWT, Spring Security
Notifications	In-App Alerts
Email	JavaMailSender, SendGrid API
Logging	SLF4J + Logback
Build Tool	Maven or Gradle