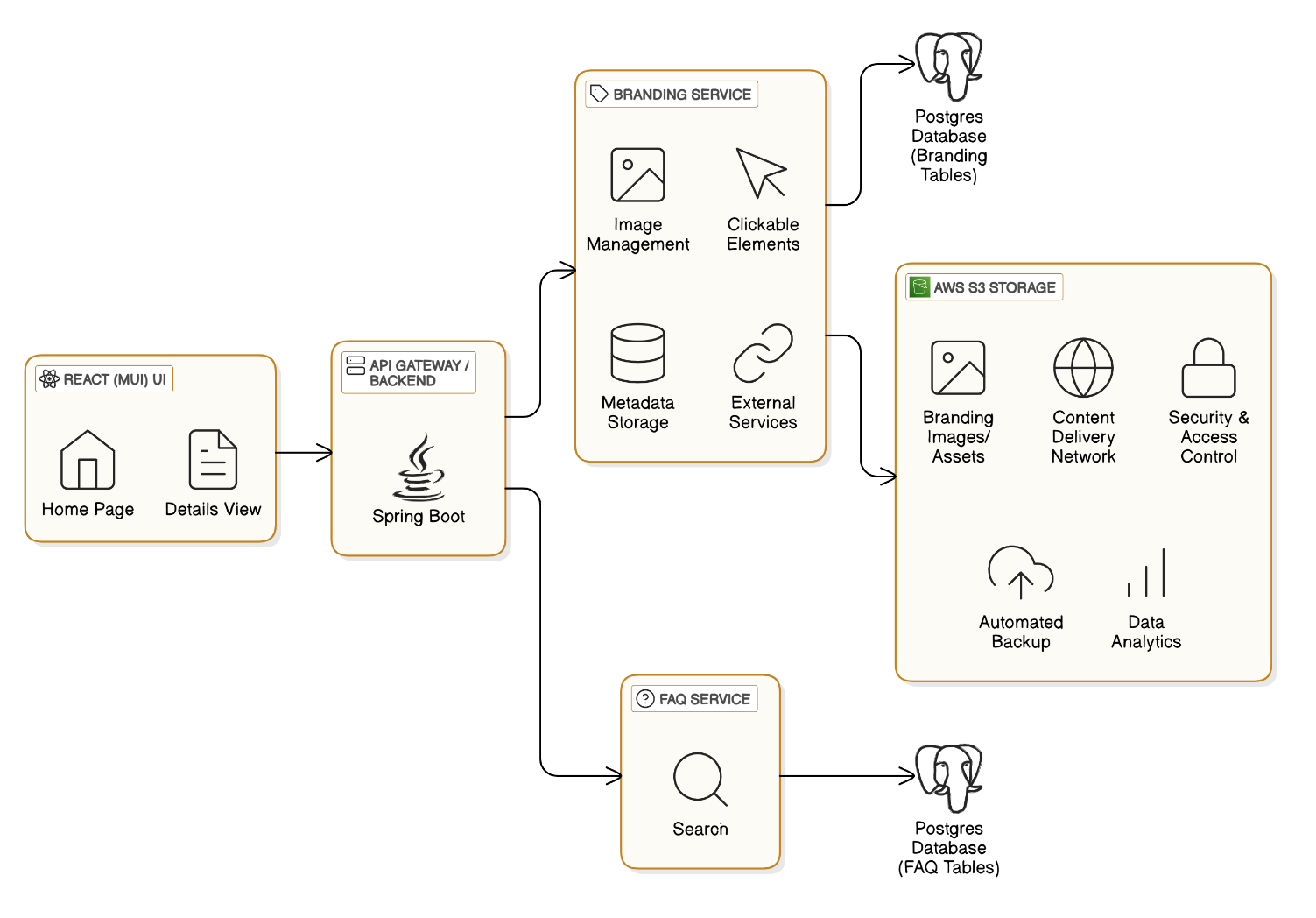
**1. Overview**

The **Branding Corner & FAQs** module comprises two major functionalities:

* **Branding Corner**:
  + **Purpose**: Showcases the latest events, promotions, and updates on the home page.
  + **Functionality**: Each branding element is clickable and redirects to a detailed page (either internal or external) with further information.
  + **Content Management**: Branding images and metadata (e.g., title, description, URL) are stored and managed centrally.
* **FAQs**:
  + **Purpose**: Provides answers to frequently asked questions for users to quickly resolve common queries.
  + **Functionality**: Users can browse, search, and filter FAQ items to find the relevant answers.

## 2. System Architecture

The module is designed as a self-contained component within the overall project. It is integrated with the front end, backend, and database layers using RESTful APIs and leverages AWS S3 for static content storage.   
 **Architecture Diagram**



## 3. Technology Stack

* **Frontend (FE):**
  + **Framework:** React.js
  + **UI Library:** Material-UI (MUI) for responsive and consistent design
* **Backend (BE):**
  + **Language:** Java
  + **Framework:** Spring Boot for building RESTful APIs and handling business logic
* **Database (DB):**
  + **Database System:** PostgreSQL for structured data storage (branding and FAQ content)
* **Storage:**
  + **Service:** AWS S3 for hosting and retrieving branding images and static assets
* **Source Code Management (SCM):**
  + **Tool:** GIT for version control and collaborative development

## 4. Functional Components

### 4.1 Branding Corner

* **Components:**
  + **Branding Carousel/Grid (FE):** Displays a series of branding elements (images, titles, brief descriptions).
  + **Branding Detail Page (FE):** Redirects users to detailed information when a branding element is clicked.
  + **Branding Service (BE):**
    - Retrieves branding data from PostgreSQL.
    - Manages links and redirection logic.
    - Serves branding images via URLs stored on AWS S3.
* **Data Attributes:**
  + ID, Title, Description, Image URL (from AWS S3), Redirection URL, Active Status, Display Priority

### 4.2 FAQs

* **Components:**
  + **FAQ List (FE):** Displays the list of frequently asked questions with an optional search or filter.
  + **FAQ Detail/Accordion (FE):** Allows users to expand and view the detailed answer.
  + **FAQ Service (BE):**
    - Handles CRUD operations (primarily read for public access) for FAQ items.
    - Processes search queries and filters.
* **Data Attributes:**
  + ID, Question, Answer, Category, Last Updated Timestamp, Active Status

## 5. Data Flow & Interaction

1. **Initial Load:**
   * When the home page loads, the React.js UI makes API calls to fetch the latest branding items and FAQ data.
   * Branding Service fetches a list of active branding elements from PostgreSQL. For images, it provides the AWS S3 URLs.
   * FAQ Service fetches active FAQ items with their associated details.
2. **User Interaction:**
   * **Branding Click:**
     + On clicking a branding element, the UI redirects the user to the branding detail page.
     + The backend validates the redirection URL (if required) and provides additional details.
   * **FAQ Interaction:**
     + Users can click on an FAQ to expand the answer or use the search/filter functionality to find specific topics.
3. **Content Updates:**
   * Admin tools or scheduled jobs update the branding and FAQ entries in the PostgreSQL database.
   * New images uploaded to AWS S3 are automatically integrated through updated S3 URLs in the branding table.

## 6. API Design

### Branding APIs

* **GET /api/branding:**
  + **Purpose:** Retrieve the list of active branding elements.
  + **Response:** JSON array of branding items containing id, title, imageUrl, redirectionUrl, description, etc.
* **GET /api/branding/{id}:**
  + **Purpose:** Retrieve detailed information for a specific branding item.
  + **Response:** Detailed JSON object with full branding details.

### FAQ APIs

* **GET /api/faqs:**
  + **Purpose:** Fetch the list of active FAQs.
  + **Response:** JSON array of FAQ items (id, question, answer, category).
* **GET /api/faqs/{id}:**
  + **Purpose:** Retrieve details for a specific FAQ item.
  + **Response:** JSON object with full FAQ details.
* **POST /api/faqs/search:** (Optional)
  + **Purpose:** Search FAQs by keywords or categories.
  + **Request:** JSON with search parameters.
  + **Response:** JSON array of matching FAQ items.

## 7. Security Considerations

* **Authentication & Authorization:**
  + Secure API endpoints using token-based authentication (JWT) to restrict administrative actions (like updates) to authorized personnel.
* **Data Protection:**
  + Use HTTPS to ensure secure data transmission between the frontend and backend.
  + Sanitize inputs to prevent SQL Injection and cross-site scripting (XSS) attacks.
* **Access Control:**
  + Role-based access to differentiate between public access for read operations (branding and FAQs) and admin operations (CRUD).

## 8. Scalability & Performance

* **Caching:**
  + Implement caching (e.g., using Spring Cache or Redis) for frequently accessed branding and FAQ data to reduce database load.
* **Asynchronous Data Loading:**
  + Use lazy loading or asynchronous requests on the frontend to enhance user experience during data fetching.
* **Load Balancing:**
  + Deploy multiple instances of the Spring Boot application behind a load balancer for high availability.
* **Static Content Delivery:**
  + Utilize AWS S3 with a CDN for fast delivery of branding images and static assets.

## 9. Source Code Management

* **Version Control:**
  + Use Git to manage code repositories.
  + Follow branch management strategies (e.g., Git Flow) to maintain a clear development and release process.