Project Documentation: 99 Names of Allah Feature

Project Name: Qalb Connect
Module: 99 Names of Allah (Asma Al-Husna) Display
Date: June 9, 2025
Student: Abdul Rafay (Software Engineering Student)

1. Feature Overview

This module provides a dedicated section within the Qalb Connect application to display the 99 Names of Allah (Asma Al-Husna) along with their English meanings. The names are stored persistently in a MySQL database and are populated automatically on application startup if the database table is empty. Users can view these names in a visually appealing, responsive grid layout after authenticating and navigating from the home page.

Key Functionality:

- **Database Storage:** Names are persisted in a dedicated asma_al_husna table.
- **Automatic Population:** The database table is populated with the 99 names if it's empty, ensuring data availability on first run.
- **Data Retrieval:** Names are fetched from the database for display.
- **User Interface:** A dedicated web page (/namesofAllah) displays the names in a clear, interactive grid.
- **Navigation:** Users can navigate to and from this feature from the authenticated home page.

2. Components Developed/Modified

Below is a detailed description of each component involved in the "99 Names of Allah" feature.

2.1. AsmaAlHusna Entity (src/main/java/com/qalbconnect/qalbconnect/model/AsmaAlHusna.java)

- **Layer:** Domain Layer (Model)
- **Purpose:** To represent a single Name of Allah as a persistent entity in the database. It defines the structure of each record in the asma_al_husna table.
- **Functionality:**
  - Maps Java object fields (id, arabicName, englishMeaning) to database table columns.
  - id is an auto-generated primary key.
  - arabicName is defined as unique=true to ensure no two entries have the same Arabic name.
  - englishMeaning holds the translation.

- **Design Patterns Implemented:**
    - **Domain Model Pattern:** AsmaAlHusna is a rich domain object that encapsulates both data and (potentially) behavior related to a Name of Allah.

2.2. AsmaAlHusnaRepository Interface (src/main/java/com/qalbconnect/qalbconnect/repository/AsmaAlHusnaRepository.java)

- **Layer:** Data Access Layer (Repository)
- **Purpose:** To provide an interface for performing CRUD (Create, Read, Update, Delete) operations on the AsmaAlHusna entities in the database.
- **Functionality:**
    - Extends Spring Data JPA's JpaRepository, automatically inheriting methods like save(), saveAll(), findAll(), count(), etc.
    - Abstracts away the boilerplate code typically required for database interactions.
- **Design Patterns Implemented:**
    - **Repository Pattern:** AsmaAlHusnaRepository acts as a repository for AsmaAlHusna objects, mediating between the domain and data mapping layers. It encapsulates the set of objects retrieved from the database and provides a clear API for data access without exposing underlying database technologies.
    - **Singleton Pattern:** As a Spring @Repository, it is managed as a Singleton by the Spring IoC container.

2.3. AsmaAlHusnaService Class (src/main/java/com/qalbconnect/qalbconnect/service/AsmaAlHusnaService.java)

- **Layer:** Service Layer
- **Purpose:** To encapsulate the business logic related to the 99 Names of Allah, specifically for data population and retrieval.
- **Functionality:**
    - **populateNamesIfEmpty() method:** Annotated with @PostConstruct and @Transactional. This method runs immediately after the service is initialized and injects its dependencies. It checks if the asma_al_husna table is empty (asmaAlHusnaRepository.count() == 0). If empty, it populates the table with the 99 predefined names using asmaAlHusnaRepository.saveAll(). This ensures the database is pre-filled on the first run.
    - **getAllNames() method:** Retrieves all AsmaAlHusna entities from the database using asmaAlHusnaRepository.findAll().
- **Design Patterns Implemented:**

    - **Singleton Pattern:** Marked with @Service, making it a single instance throughout the application. This ensures consistent data population logic and efficient resource management.
    - **Command Pattern (Implicit - Data Initialization Command):** The populateNamesIfEmpty() method can be viewed as an implicit command. It encapsulates the "initialize 99 names" action, making it a self-contained operation executed at a specific lifecycle event (@PostConstruct).
    - **Null Object Pattern (Implicit via Empty List):** The getAllNames() method returns a List<AsmaAlHusna>. If no names are found (e.g., the table is empty and populateNamesIfEmpty hasn't run yet, though less likely now), it returns an empty list rather than null. This allows client code to process the result without explicit null checks, promoting more robust and cleaner code.

2.4. NamesOfAllahController Class (src/main/java/com/qalbconnect/qalbconnect/controller/NamesOfAllahController.java)

- **Layer:** Presentation Layer (Controller)
- **Purpose:** To handle web requests for the "99 Names of Allah" page, fetch the data, and prepare it for display in the HTML template.
- **Functionality:**
    - @RequestMapping("/namesofAllah"): Maps all requests for /namesofAllah to this controller.
    - @GetMapping: Maps GET requests to the displayNamesOfAllah method.
    - Injects AsmaAlHusnaService to retrieve the names.
    - Adds the list of names to the Thymeleaf Model under the attribute name "names".
    - Returns "namesofAllah", which tells Spring to render the namesofAllah.html template.
- **Design Patterns Implemented:**
    - **Singleton Pattern:** Marked with @Controller, making it a single instance throughout the application.
    - **Facade Pattern:** The controller interacts with the AsmaAlHusnaService which acts as a Facade. The controller doesn't need to know *how* the names are stored or retrieved (e.g., from a database, file, or cache); it simply calls a high-level method (getAllNames()) on the service, simplifying its logic and decoupling it from data access concerns.
    - **Command Pattern (Implicit):** The displayNamesOfAllah method implicitly represents a "Display 99 Names" command. It encapsulates the request to render the names, separating the request from its execution mechanism.

2.5. Frontend Views (src/main/resources/static/css/namesofAllah.css, src/main/resources/templates/namesofAllah.html)

- **Layer:** Presentation Layer (View)
- **Purpose:** To provide the visual representation of the 99 Names of Allah to the user.
- **Functionality:**
  - **namesofAllah.css**: Defines the styling for the page, including the responsive grid layout, card design for each name, and typography for Arabic and English text, ensuring a consistent and appealing user experience.
  - **namesofAllah.html**:
    - Uses Thymeleaf (xmlns:th="http://www.thymeleaf.org") to dynamically render the data.
    - Includes a "Back to Home" button for easy navigation using th:href="@{/home}".
    - Employs th:each="name : ${names}" to iterate through the list of AsmaAlHusna objects passed from the controller.
    - Uses th:text="${name.arabicName}" and th:text="${name.englishMeaning}" to display the specific data for each name.
    - Avoids client-side JavaScript for data population, relying on server-side rendering for robustness.
- **Design Patterns Implemented:**
  - **Model-View-Controller (MVC) Pattern:** These files form the "View" part of the MVC pattern. They are responsible solely for presenting data and receiving user input, while the Controller (NamesOfAllahController) acts as the "Controller" and the data from AsmaAlHusna entities forms the "Model." This separation of concerns is fundamental to the architecture.