DAD PROJECT DOCUMENTATION

How many apps involved

- Manage summon
- Pay summon

Brief explanation each apps

- Manage summon - serves as an administrative tool for summons management. Administrators can handle tasks like creating, reading, updating, and deleting summons records. The application directly interacts with the database to ensure real-time data management, ensuring accuracy in summons administration tasks.
- Pay summon - enables users to pay their summons. It features a Java Swing user interface where users can log in, view their summons, and process payments. This frontend interacts with a PHP backend, which handles fetching summons data from the database and processing payment transactions.

Architecture/Layer diagram for each of the apps including the middleware

Pay summon

1. **Presentation Layer (Java Swing UI):**
    - **User Interface Components:** Includes screens for login, viewing summons, and processing payments.
    - **Event Handlers:** Manage user interactions such as button clicks and form submissions.
2. **Application Layer (Java Backend):**
    - **Controller Classes:** Handle user input and orchestrate interactions between the UI and backend services.
    - **Service Classes:** Implement business logic for fetching summons data and processing payments.
3. **Middleware (PHP Backend):**
    - **API Endpoints:** Expose functionalities for login authentication, fetching summons data, and processing payment requests.
    - **Session Management:** Maintains user sessions and handles authentication tokens.

4. **Data Access Layer (Database):**
    - **Database Schema:** Tables for users, cases, and summons storing relevant data.
    - **Data Access Objects (DAOs):** Interface with the database to perform CRUD operations on summons and related entities.

Manage Summon

1. **Presentation Layer (Java Swing UI):**
    - **User Interface Components:** Administrative interface for managing summons, users, and cases.
    - **Event Handlers:** Handles user interactions and form validations.
2. **Application Layer (Java Backend):**
    - **Controller Classes:** Manage HTTP requests, route them to appropriate handlers.
    - **Service Classes:** Implement business logic for CRUD operations on summons, users, and cases.
3. **Middleware (PHP Backend):**
    - **API Endpoints:** Tables for storing summons, users, and cases information.
    - **Session Management:** Translate database records into objects and vice versa.
4. **Data Access Layer (Database):**
    - **Database Schema:** Direct interaction with the database using SQL queries or ORM frameworks.
    - **Data Access Objects (DAOs):** Encapsulate data access logic, providing an interface for CRUD operations.

List of URL end points middleware RESTful

paySummons.java

```
// Fetch summons data for the logged-in user from the database.
URL url = new URL("http://localhost/eSummonsSystem/fetch_summons.php");

// Update the status of a summon in the database.
URL url = new URL("http://localhost/eSummonsSystem/update_status.php");
```

manageSummons.java

```
      // Fetch cases
URL url = new URL("http://localhost/eSummonsSystem/fetch_cases.php");
```

```
// Add summons
URL url = new URL("http://localhost/eSummonsSystem/add_summons.php");

// Edit summons
URL url = new URL("http://localhost/eSummonsSystem/edit_summons.php");

// Delete summons
URL url = new URL("http://localhost/eSummonsSystem/delete_summons.php");

// Fetch all summons
URL url = new URL("http://localhost/eSummonsSystem/fetch_all_summons.php");
```

Functions/Features in the middleware

- ✓ fetch_summons.php

```
// Fetch summons data for the logged-in user from the database.
URL url = new URL("http://localhost/eSummonsSystem/fetch_summons.php");
```

Function: Fetches summons data for the logged-in user from the database.

Purpose: Specifically retrieves summons associated with the currently logged-in user, likely used to display summons relevant to that user

- ✓ update_status.php

```
// Update the status of a summon in the database.
URL url = new URL("http://localhost/eSummonsSystem/update_status.php");
```

Function: Updates the status of a summon in the database.

Purpose: Changes the status of a particular summon (e.g., from pending to paid) based on the summon ID and new status provided.

- ✓ fetch_cases.php

```
      // Fetch cases
URL url = new URL("http://localhost/eSummonsSystem/fetch_cases.php");
```

Function: Fetches all available case types from the database.

Purpose: Provides a list of case types for users to select when adding or editing summons.

- ✓ add_summons.php

```
// Add summons
URL url = new URL("http://localhost/eSummonsSystem/add_summons.php");
```

Function: Adds a new summon to the database.

Purpose: Inserts a new summon record with details such as summon ID, user ID, case type, and amount into the database.

- ✓ edit_summons.php

```
// Edit summons
URL url = new URL("http://localhost/eSummonsSystem/edit_summons.php");
```

Function: Updates an existing summon in the database.

Purpose: Modifies summon details (username, case type, amount) based on the summon ID provided.

- ✓ delete_summons.php

```
// Delete summons
URL url = new URL("http://localhost/eSummonsSystem/delete_summons.php");
```

Function: Deletes a summon from the database.

Purpose: Removes a summon record entirely from the database based on the summon ID provided.

- ✓ fetch_all_cases.php

```
// Fetch all summons
URL url = new URL("http://localhost/eSummonsSystem/fetch_all_summons.php");
```

Function: Fetches all summons stored in the database.

Purpose: Retrieves a list of all summons, including details like summon ID, user ID, case type, amount, and status, for display in the application.

The database and tables involve in the projects

Table cases

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	case_id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
2	case_type	varchar(50)	utf8mb4_general_ci		No	None			Change Drop More

Table users

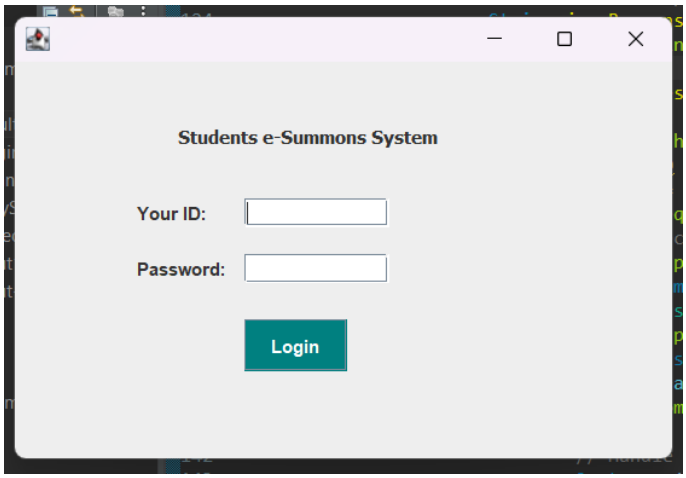
#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	user_id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
2	username	varchar(50)	utf8mb4_general_ci		No	None			Change Drop More
3	password	varchar(50)	utf8mb4_general_ci		No	None			Change Drop More
4	role	enum('admin', 'user')	utf8mb4_general_ci		No	None			Change Drop More

Table summons

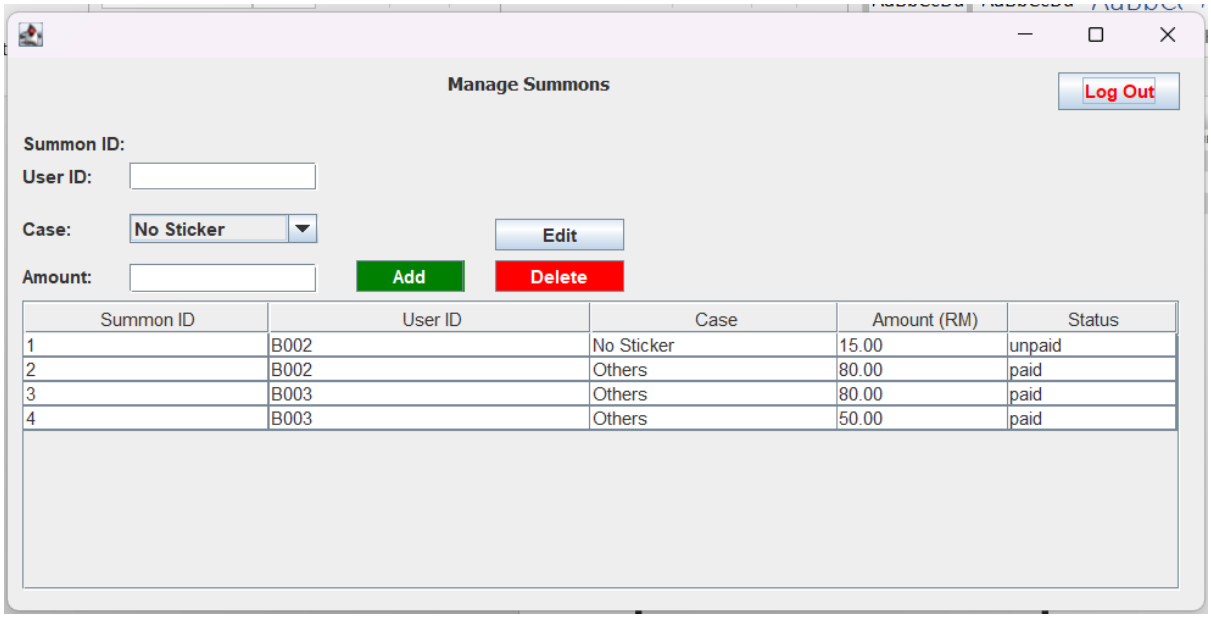
#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	summon_id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
2	user_id	int(11)			Yes	NULL			Change Drop More
3	case_id	int(11)			Yes	NULL			Change Drop More
4	amount	decimal(10,2)			Yes	NULL			Change Drop More
5	status	enum('paid', 'unpaid')	utf8mb4_general_ci		No	unpaid			Change Drop More

INTERFACES

login.java



manageSummons.java (admin)



paySummons.java (user)