```

project\_root/

│

├── config/

│ └── database.php # Database connection configuration

│

├── includes/

│ ├── auth.php # Authentication functions

│ ├── db.php # Database connection function (getDbConnection)

│ ├── functions.php # General utility functions

│ └── User.php # User class for database operations

│

├── public/

│ ├── index.php # Entry point of the application

│ ├── dashboard.php # Main dashboard page for staff

│ ├── admin.php # Admin panel page

│ ├── login.php # Login page and processing

│ ├── logout.php # Logout processing

│ ├── reset\_password.php # Password reset page

│ ├── manage\_users.php # User management page

│ ├── edit\_user.php # Edit user page

│ ├── manage\_staff.php # Staff management page

│ └── edit\_staff.php # Edit staff page

│

├── assets/

│ ├── css/

│ │ └── styles.css # Main CSS file for styling

│ ├── js/

│ │ └── app.js # Main JavaScript file for client-side functionality

│ └── img/

│ └── silver\_system\_logo.png # Logo image file

│

└── .htaccess # Apache configuration file (if used)

```

File Descriptions and Logic:

1. `public/index.php`:

- Serves as the entry point of the application.

- Checks if the user is logged in using the `isLoggedIn()` function.

- Redirects to `dashboard.php` if the user is logged in, or to `login.php` if not.

2. `includes/auth.php`:

- Handles user authentication (login, logout).

3. `includes/db.php`:

- Provides `getDbConnection()` function to create and return a PDO database connection.

- Used throughout the application for database operations.

4. `includes/functions.php`:

- Contains utility functions like `sanitizeInput()`, `redirectTo()`, `isLoggedIn()`, etc.

- Used across multiple files for common operations.

5. `includes/User.php`:

- Defines the User class with methods for user-related database operations.

- Handles user creation, password resets, login logging, etc.

6. `public/dashboard.php`:

- Main page for staff members after login.

- Displays the main dashboard with various function buttons, including the 'ADMIN' button for all users (temporarily).

7. `public/admin.php`:

- Admin panel page, accessible only by logging in as the 'admin' user.

- Provides admin-specific functions like managing staff, users, and viewing logs.

8. `public/login.php`:

- Handles user login process.

- Authenticates users and sets appropriate session variables.

- Redirects to `admin.php` if the username is 'admin', otherwise to `dashboard.php`.

9. `public/logout.php`:

- Handles user logout process.

- Destroys the session and redirects to the login page.

10. `public/reset\_password.php`:

- Allows users to reset their password when required (e.g., after admin reset).

11. `public/manage\_users.php`:

- Displays a list of all users and allows adding new users.

12. `public/edit\_user.php`:

- Allows editing existing user information.

13. `public/manage\_staff.php`:

- Displays a list of all staff members and allows adding new staff.

14. `public/edit\_staff.php`:

- Allows editing existing staff information.

Application Logic Flow:

1. User visits the application root (which points to `index.php`).

2. `index.php` checks if the user is logged in:

- If logged in, redirects to `dashboard.php`.

- If not logged in, redirects to `login.php`.

3. At `login.php`, user enters credentials:

- After successful authentication:

- If the username is 'admin', they are redirected to `admin.php`.

- All other users are redirected to `dashboard.php`.

4. In `dashboard.php`, staff can access various functions, and the 'ADMIN' button is visible to all (temporarily).

5. In `admin.php`, the admin can manage staff, users, reset passwords, and perform other administrative tasks.

6. The `User` class and `auth.php` work together to handle all user-related operations and authentication.

7. Utility functions in `functions.php` are used throughout the application for common tasks.

8. All database operations are performed using the PDO connection provided by `db.php`.

This structure maintains a separation of concerns, with authentication, database operations, and utility functions in the `includes` directory, and public-facing pages in the `public` directory. The application uses session-based authentication, with a clear distinction between the 'admin' user and staff users. The role-based access control has been simplified, with plans to implement more complex permissions based on staff titles in the future.