## Services Description

**AuthService**

* **Purpose**: Provides authentication services for the ROH application.
* **Key Method**: GenerateJwtToken(UserModel user)
* **Functionality**: Generates a JWT (JSON Web Token) for user authentication.

**GenerateJwtToken**

* **Parameters**: Accepts a UserModel object representing the user.
* **Process**:
  + Retrieves a security key from an environment variable or uses a default key.
  + Initializes a JwtSecurityTokenHandler.
  + Creates a SecurityTokenDescriptor with the user’s claims, token expiry, and signing credentials.
  + Generates a SecurityToken and returns its string representation.

**Security Considerations**

* **Key Management**: The security key should be stored securely and not hardcoded.
* **Token Expiry**: The token expires after 24 hours, which can be adjusted as needed.

**AccountService**

* **Class Overview**: AccountService is a service class in the ROH.Services.Account namespace, responsible for handling account-related operations.
* **Dependencies**:
  + **IExceptionHandler**: Handles exceptions that may occur during service operations.
  + **IAccountRepository**: Provides data access methods for account entities.
  + **IMapper**: Maps domain entities to models and vice versa.
* **Methods**:
  + **GetAccounByUserGuid**: Asynchronously retrieves an account by its user GUID and returns a DefaultResponse.
  + **UpdateAccount**: Asynchronously updates an account based on the provided AccountModel and returns a DefaultResponse.
* **Exception Handling**: Both methods implement try-catch blocks to handle exceptions using the injected IExceptionHandler.

**LoginService**

* **Class Definition**:
  + The LoginService class is part of the ROH.Services.Account namespace and implements the ILoginService interface.
* **Constructor Dependencies**:
  + It depends on IExceptionHandler, IValidator<LoginModel>, IUserService, and IAuthService for handling exceptions, validating login models, user services, and authentication services, respectively.
* **Methods**:
  + **Login**: Asynchronously validates the LoginModel and either returns errors, finds the user, or validates the password.
  + **ValidatePassword**: Checks if the provided password matches the user’s password.
  + **FindUser**: Attempts to find a user by email or username.
* **Return Types**:
  + All methods return a Task<DefaultResponse> which includes the HTTP status code and relevant messages or data.

**UserService**

* **Purpose**: Provides user-related operations and services.
* **Dependencies**:
  + IExceptionHandler: Handles exceptions that may occur during service operations.
  + IValidator<UserModel>: Validates user data.
  + IUserRepository: Interacts with the user data repository.
  + IMapper: Maps data between different object types.

**Public Methods**

* **NewUser(UserModel userModel)**:
  + **Description**: Creates a new user with the provided user model.
  + **Parameters**: UserModel userModel - The user data to create a new user.
  + **Returns**: Task<DefaultResponse> - The result of the user creation process, including status code and message.
  + **Exceptions**: Returns handled exceptions via IExceptionHandler.
* **FindUserByEmail(string email)**:
  + **Description**: Retrieves a user model based on the provided email.
  + **Parameters**: string email - The email address to search for.
  + **Returns**: Task<UserModel?> - The user model if found, otherwise null.
* **FindUserByUserName(string userName)**:
  + **Description**: Retrieves a user model based on the provided username.
  + **Parameters**: string userName - The username to search for.
  + **Returns**: Task<UserModel?> - The user model if found, otherwise null.
* **GetUserByGuid(Guid userGuid)**:
  + **Description**: Retrieves a user model based on the provided GUID.
  + **Parameters**: Guid userGuid - The GUID of the user to search for.
  + **Returns**: Task<UserModel> - The user model if found.
* **ValidatePassword(string password, Guid userGuid)**:
  + **Description**: Validates the provided password against the user’s stored password.
  + **Parameters**:
    - string password - The password to validate.
    - Guid userGuid - The GUID of the user whose password is being validated.
  + **Returns**: Task<bool> - True if the password is valid, otherwise false.

### **ExceptionHandler**

**Purpose**: Handles exceptions by logging them and providing a user-friendly response.

**Usage**:

* Instantiate with an ILogRepository object.
* Call HandleException method with an Exception object to process.

**Methods**:

* **HandleException(Exception exception)**: Processes the exception, logs it, and returns a DefaultResponse object. In **DEBUG** mode, the actual error message is returned to the user; otherwise, a generic error message is provided.