Movie Review Application Documentation

Overview

This application is a Java-based system that allows users to manage movie reviews. Users can register, log in, view movies, write and share reviews, and manage their profiles. Administrators have additional privileges, such as managing users and deleting any review. The application interacts with a MySQL database (movie_reviews_db) defined in MoviesDatabase.sql, which stores information about users, movies, reviews, and shared reviews.

Below is a detailed, structured documentation of each class and its public methods, explaining their functionality based on the provided source code.

Classes and Their Methods

DatabaseConnection

- **Purpose**: A utility class that manages connections to the MySQL database (movie_reviews_db).
- **Methods**:
    - **public static Connection getConnection() throws SQLException**
        - **Description**: Establishes and returns a connection to the movie_reviews_db database using the MySQL JDBC driver.
        - **Parameters**: None.
        - **Returns**: A Connection object for database operations.
        - **Throws**: SQLException if the connection fails (e.g., due to invalid credentials or database unavailability).
        - **Details**:
            - Uses predefined constants: URL (jdbc:mysql://localhost:3306/movie_reviews_db), USER (root), and PASSWORD (manager).
            - A static block loads the MySQL JDBC driver (com.mysql.cj.jdbc.Driver) during class initialization. If the driver is not found, the application exits with an error.

InputHandler

- **Purpose**: A utility class for handling and validating user input from the console.
- **Methods**:
    - **public InputHandler()**

        - **Description**: Constructor that initializes a Scanner object to read from System.in.
        - **Parameters**: None.
        - **Returns**: N/A (constructor).
    - **public String getString(String prompt)**
        - **Description**: Prompts the user with a message and reads a string input.
        - **Parameters**:
            - prompt (String): The message displayed to the user.
        - **Returns**: The trimmed input string; returns an empty string if the input is empty.
        - **Details**: Uses scanner.nextLine() to capture the full line of input.
    - **public int getInt(String prompt)**
        - **Description**: Prompts the user and reads an integer, handling invalid inputs with a retry loop.
        - **Parameters**:
            - prompt (String): The message displayed to the user.
        - **Returns**: The parsed integer value.
        - **Details**:
            - Continuously prompts until a valid integer is entered.
            - Catches NumberFormatException and displays an error message if the input is not an integer.
    - **public int getIntInRange(String prompt, int min, int max)**
        - **Description**: Prompts the user and reads an integer within a specified range.
        - **Parameters**:
            - prompt (String): The message displayed to the user.
            - min (int): The minimum acceptable value.
            - max (int): The maximum acceptable value.
        - **Returns**: An integer within the range [min, max].
        - **Throws**: IllegalArgumentException if min > max.
        - **Details**:
            - Uses getInt(prompt) internally and validates the range.
            - Displays an error message and retries if the input is outside the range.

- **public void close()**
  - **Description**: Closes the Scanner object to free system resources.
  - **Parameters**: None.
  - **Returns**: None.
  - **Details**: Checks if the scanner is not null before closing to avoid exceptions.

AuthService

- **Purpose**: Manages user authentication, registration, and profile operations.
- **Methods**:
  - **public boolean register(String firstName, String lastName, String email, String mobile, String birthDate, String password, String accountType)**
    - **Description**: Registers a new user with the provided details and stores them in the users table.
    - **Parameters**:
      - firstName (String): User's first name.
      - lastName (String): User's last name.
      - email (String): User's email (must be unique).
      - mobile (String): User's mobile number.
      - birthDate (String): User's birth date in YYYY-MM-DD format.
      - password (String): User's password (stored in plain text).
      - accountType (String): Either "Admin" or "Regular".
    - **Returns**: true if registration succeeds, false otherwise.
    - **Details**:
      - Validates inputs: all fields must be non-empty; email, mobile, and password must match specific formats; birth date must indicate the user is at least 13 years old; account type must be valid.
      - Checks for duplicate email in the database.
      - Inserts the user into the users table using a prepared statement.
      - Prints error messages for validation failures or SQL exceptions.
  - **public boolean login(String email, String password)**
    - **Description**: Authenticates a user and sets the currentUser if successful.

- **Parameters**:
  - email (String): User's email.
  - password (String): User's password.
- **Returns**: true if login succeeds, false otherwise.
- **Details**:
  - Queries the users table by email and compares the provided password with the stored plain-text password.
  - If successful, creates a User object with the retrieved data and assigns it to currentUser.
  - Prints error messages for invalid email, password, or database errors.

- **public boolean deleteUser(int adminId, int userIdToDelete)**
  - **Description**: Allows an admin to delete a user, with restrictions on deleting other admins.
  - **Parameters**:
    - adminId (int): ID of the admin performing the deletion.
    - userIdToDelete (int): ID of the user to delete.
  - **Returns**: true if deletion succeeds, false otherwise.
  - **Details**:
    - Checks if the user exists and their account type.
    - Prevents an admin from deleting another admin unless it's themselves.
    - Deletes the user from the users table, cascading to related tables (e.g., reviews, shares).
    - Prints error messages for invalid users or SQL errors.
- **public List<User> listAllUsers()**
  - **Description**: Retrieves a list of all users from the users table.
  - **Parameters**: None.
  - **Returns**: A List<User> containing all user objects.
  - **Details**:
    - Executes a SELECT query to fetch all user details.
    - Constructs User objects for each row and adds them to an ArrayList.
    - Prints an error message if the query fails but still returns an empty list.

- **public boolean updateProfile(int userId, String firstName, String lastName, String email, String mobile, String birthDate)**
  - **Description**: Updates the profile of the specified user.
  - **Parameters**:
    - userId (int): ID of the user to update.
    - firstName (String): New first name (optional).
    - lastName (String): New last name (optional).
    - email (String): New email (optional, must be unique).
    - mobile (String): New mobile number (optional).
    - birthDate (String): New birth date (optional).
  - **Returns**: true if the update succeeds, false otherwise.
  - **Details**:
    - Uses current values from currentUser if new values are null or empty.
    - Validates email, mobile, and birth date (must be at least 13 years old).
    - Ensures the new email isn't already in use by another user.
    - Updates the users table with a prepared statement.
    - Prints error messages for validation failures or SQL errors.
- **public boolean changePassword(int userId, String newPassword)**
  - **Description**: Changes the password for the specified user.
  - **Parameters**:
    - userId (int): ID of the user.
    - newPassword (String): New password.
  - **Returns**: true if the update succeeds, false otherwise.
  - **Details**:
    - Validates the new password (at least 8 characters, with uppercase, lowercase, digit, and special character).
    - Updates the password field in the users table (stored in plain text).
    - Prints error messages for invalid password or SQL errors.
- **public User getCurrentUser()**
  - **Description**: Returns the currently logged-in user.

        - **Parameters**: None.
        - **Returns**: The User object representing the current user, or null if no user is logged in.
    - **public void logout()**
        - **Description**: Logs out the current user by setting currentUser to null.
        - **Parameters**: None.
        - **Returns**: None.

MovieService

- **Purpose**: Provides methods for displaying movie information from the movies and reviews tables.
- **Methods**:
    - **public void displayAllMovies()**
        - **Description**: Prints a list of all movies in the database, sorted by title.
        - **Parameters**: None.
        - **Returns**: None.
        - **Details**:
            - Queries the movies table and displays each movie's ID, title, release date, and genre.
            - Prints "No movies found" if the result set is empty.
            - Handles SQL exceptions by printing an error message.
    - **public void displayMovieDetails(int movieId)**
        - **Description**: Prints detailed information about a specific movie, including its reviews and average rating.
        - **Parameters**:
            - movieId (int): ID of the movie to display.
        - **Returns**: None.
        - **Details**:
            - Joins movies, reviews, and users tables to fetch movie details and associated reviews.
            - Calculates the average rating using calculateAverageRating(movieId).
            - Displays movie ID, title, release date, genre, average rating, and each review with reviewer name.

        - Prints "Movie not found" if no matching movie is found.
        - Handles SQL exceptions by printing an error message.
    - **private double calculateAverageRating(int movieId)**
        - **Description**: Calculates the average rating for a movie based on its reviews.
        - **Parameters**:
            - movieId (int): ID of the movie.
        - **Returns**: The average rating as a double, or 0.0 if no reviews exist or an error occurs.
        - **Details**:
            - Uses an SQL AVG query on the reviews table filtered by movieId.
            - Returns 0.0 if the result is null or an SQL exception occurs.

User

- **Purpose**: A model class representing a user in the system with getters and setters for properties.
- **Properties**:
    - id (int, final): User's unique identifier.
    - firstName (String): User's first name.
    - lastName (String): User's last name.
    - email (String): User's email.
    - mobile (String): User's mobile number.
    - birthDate (String): User's birth date.
    - accountType (String, final): User's account type ("Admin" or "Regular").
- **Methods**:
    - **public User(int id, String firstName, String lastName, String email, String mobile, String birthDate, String accountType)**
        - **Description**: Constructor that initializes a User object.
        - **Parameters**: All properties as listed above.
        - **Details**: Sets default empty strings for null inputs except for id and accountType.
    - **public int getId()**
        - **Description**: Returns the user's ID.
        - **Returns**: int.

- **public String getFirstName()**
  - **Description**: Returns the user's first name.
  - **Returns**: String.
- **public void setFirstName(String firstName)**
  - **Description**: Sets the user's first name.
  - **Parameters**: firstName (String).
  - **Details**: Retains current value if input is null.
- **public String getLastName()**
  - **Description**: Returns the user's last name.
  - **Returns**: String.
- **public void setLastName(String lastName)**
  - **Description**: Sets the user's last name.
  - **Parameters**: lastName (String).
  - **Details**: Retains current value if input is null.
- **public String getEmail()**
  - **Description**: Returns the user's email.
  - **Returns**: String.
- **public void setEmail(String email)**
  - **Description**: Sets the user's email.
  - **Parameters**: email (String).
  - **Details**: Retains current value if input is null.
- **public String getMobile()**
  - **Description**: Returns the user's mobile number.
  - **Returns**: String.
- **public void setMobile(String mobile)**
  - **Description**: Sets the user's mobile number.
  - **Parameters**: mobile (String).
  - **Details**: Retains current value if input is null.
- **public String getBirthDate()**
  - **Description**: Returns the user's birth date.

        - **Returns**: String.
    - **public void setBirthDate(String birthDate)**
        - **Description**: Sets the user's birth date.
        - **Parameters**: birthDate (String).
        - **Details**: Retains current value if input is null.
    - **public String getAccountType()**
        - **Description**: Returns the user's account type.
        - **Returns**: String.

ReviewService

- **Purpose**: Manages the creation, editing, deletion, and sharing of movie reviews.
- **Methods**:
    - **public boolean createReview(int userId, int movieId, String reviewText, int rating)**
        - **Description**: Creates a new review for a movie by a user.
        - **Parameters**:
            - userId (int): ID of the user creating the review.
            - movieId (int): ID of the movie being reviewed.
            - reviewText (String): The review content.
            - rating (int): Rating from 1 to 5.
        - **Returns**: true if creation succeeds, false otherwise.
        - **Details**:
            - Validates: review text (non-empty, ≤1024 characters), rating (1-5), movie existence, and ensures the user hasn't already reviewed the movie.
            - Inserts into the reviews table with a prepared statement.
            - Prints error messages for validation failures or SQL errors.
    - **public boolean editReview(int reviewId, int userId, String reviewText, int rating)**
        - **Description**: Edits an existing review if it belongs to the user.
        - **Parameters**:
            - reviewId (int): ID of the review to edit.
            - userId (int): ID of the user editing the review.

        - reviewText (String): Updated review content.
        - rating (int): Updated rating (1-5).
    - **Returns**: true if the update succeeds, false otherwise.
    - **Details**:
        - Validates review text and rating, and checks ownership.
        - Updates the reviews table with a prepared statement.
        - Prints error messages for validation or permission issues.
- **public boolean deleteReviewByAdmin(int reviewId)**
    - **Description**: Deletes any review (admin-only functionality).
    - **Parameters**:
        - reviewId (int): ID of the review to delete.
    - **Returns**: true if deletion succeeds, false otherwise.
    - **Details**:
        - Verifies the review exists, then deletes it from the reviews table.
        - Prints error messages if the review isn't found or deletion fails.
- **public boolean deleteReview(int reviewId, int userId)**
    - **Description**: Deletes a review if it belongs to the user.
    - **Parameters**:
        - reviewId (int): ID of the review to delete.
        - userId (int): ID of the user requesting deletion.
    - **Returns**: true if deletion succeeds, false otherwise.
    - **Details**:
        - Deletes from reviews where both id and user_id match.
        - Prints an error if the review isn't found or the user lacks permission.
- **public void displayUserReviews(int userId)**
    - **Description**: Displays all reviews by a specific user.
    - **Parameters**:
        - userId (int): ID of the user whose reviews to display.
    - **Returns**: None.
    - **Details**:

        - Joins reviews and movies tables, orders by modification date descending.
        - Prints review ID, movie title, review text, rating, and modified timestamp.
        - Prints "No reviews found" if none exist.
    - **public void displayAllReviews()**
        - **Description**: Displays all reviews in the system.
        - **Parameters**: None.
        - **Returns**: None.
        - **Details**:
            - Joins reviews, movies, and users tables, orders by modification date descending.
            - Prints review ID, movie title, reviewer's name, review text, rating, and modified timestamp.
            - Prints "No reviews found" if none exist.
    - **public void displaySharedReviews(int userId)**
        - **Description**: Displays reviews shared with a specific user.
        - **Parameters**:
            - userId (int): ID of the user to show shared reviews for.
        - **Returns**: None.
        - **Details**:
            - Joins reviews, movies, users, and shares tables, filters by shares.user_id, orders by share date descending.
            - Prints review details similar to displayAllReviews.
            - Prints "No reviews shared with you" if none exist.
    - **public boolean shareReview(int reviewId, int userId, String sharedWithEmail)**
        - **Description**: Shares a review with another user via their email.
        - **Parameters**:
            - reviewId (int): ID of the review to share.
            - userId (int): ID of the user sharing the review.
            - sharedWithEmail (String): Email of the user to share with.
        - **Returns**: true if sharing succeeds, false otherwise.
        - **Details**:

        - Validates: review ownership, recipient email existence, prevents self-sharing and duplicate shares.
        - Inserts into the shares table with a prepared statement.
        - Prints error messages for validation or SQL errors.

Main

- **Purpose**: The entry point of the application, handling user interaction and menu navigation.
- **Key Methods**:
    - **public static void main(String[] args)**
        - **Description**: Launches the application and ensures the InputHandler is closed on exit.
        - **Parameters**: Command-line arguments (unused).
        - **Details**: Creates a Main instance and calls run() in a try-finally block.
    - **private void run()**
        - **Description**: The main loop that displays menus based on login status and account type.
        - **Details**:
            - Checks authService.getCurrentUser():
                - If null, shows showMainMenu().
                - If Admin, shows showAdminMenu().
                - If Regular, shows showSignedInMenu().
            - Runs indefinitely until the application exits.
    - **private void showAdminMenu()**
        - **Description**: Displays the admin menu and processes choices (1-14).
        - **Details**: Options include user management, profile editing, review management, and logout.
    - **private void showUserManagementMenu()**
        - **Description**: Displays the admin's user management submenu and processes choices (1-6).
        - **Details**: Options include adding users, updating regular users, deleting users, listing users, and returning to the admin menu.
    - **private void addAdminUser()**

        - **Description**: Registers a new admin user.
        - **Details**: Prompts for user details, confirms password, and calls authService.register() with "Admin".
    - **private void addRegularUser()**
        - **Description**: Registers a new regular user.
        - **Details**: Similar to addAdminUser() but uses "Regular".
    - **private void updateRegularUser()**
        - **Description**: Updates a regular user's profile and optionally resets their password.
        - **Details**:
            - Finds a regular user by email, prompts for new details, and calls authService.updateProfile() and authService.changePassword() if applicable.
    - **private void deleteUser()**
        - **Description**: Deletes a user after confirmation.
        - **Details**: Finds a user by email, confirms with "Y/N", and calls authService.deleteUser().
    - **private void listAllUsers()**
        - **Description**: Lists all users after confirmation.
        - **Details**: Calls authService.listAllUsers() and prints user details.
    - **private void deleteAnyReview()**
        - **Description**: Deletes any review (admin-only) after confirmation.
        - **Details**: Displays all reviews, prompts for a review ID, confirms, and calls reviewService.deleteReviewByAdmin().
    - **private void showMainMenu()**
        - **Description**: Displays the main menu for unauthenticated users (1-3).
        - **Details**: Options are sign up, sign in, or exit.
    - **private void register()**
        - **Description**: Handles user registration for regular users.
        - **Details**: Prompts for details, confirms password, and calls authService.register() with "Regular".
    - **private void login()**
        - **Description**: Handles user login.

        - **Details**: Prompts for email and password, calls authService.login(), and greets the user on success.
    - **private void showSignedInMenu()**
        - **Description**: Displays the menu for signed-in regular users (1-11).
        - **Details**: Options include profile editing, movie viewing, review management, and logout.
    - **private void editProfile()**
        - **Description**: Allows the current user to edit their profile.
        - **Details**: Prompts for optional new details and calls authService.updateProfile().
    - **private void changePassword()**
        - **Description**: Allows the current user to change their password.
        - **Details**: Prompts for new password, confirms it, and calls authService.changePassword().
    - **private void displayAllMovies()**
        - **Description**: Displays all movies.
        - **Details**: Calls movieService.displayAllMovies().
    - **private void createReview()**
        - **Description**: Allows the current user to create a review.
        - **Details**: Displays movies, prompts for movie ID, review text, and rating, then calls reviewService.createReview().
    - **private void editReview()**
        - **Description**: Allows the current user to edit their review.
        - **Details**: Displays user reviews, prompts for review ID, new text, and rating, then calls reviewService.editReview().
    - **private void deleteReview()**
        - **Description**: Allows the current user to delete their review.
        - **Details**: Displays user reviews, prompts for review ID, confirms, and calls reviewService.deleteReview().
    - **private void displayAllReviews()**
        - **Description**: Displays all reviews (admin-only).
        - **Details**: Calls reviewService.displayAllReviews().
    - **private void displayUserReviews()**

        - **Description**: Displays the current user's reviews.
        - **Details**: Calls reviewService.displayUserReviews().
    - **private void displaySharedReviews()**
        - **Description**: Displays reviews shared with the current user.
        - **Details**: Calls reviewService.displaySharedReviews().
    - **private void shareReview()**
        - **Description**: Allows the current user to share a review.
        - **Details**: Displays user reviews, prompts for review ID and email, then calls reviewService.shareReview().
    - **private void displayMovieDetails()**
        - **Description**: Displays details of a specific movie.
        - **Details**: Displays all movies, prompts for movie ID, and calls movieService.displayMovieDetails().

Additional Notes

- **Database Schema** (MoviesDatabase.sql):
    - Tables: users, movies, reviews, shares.
    - Includes sample data for users, movies, reviews, and shares.
    - Uses foreign keys with ON DELETE CASCADE for data integrity.
- **Security**: Passwords are stored in plain text, which is insecure. In a production environment, they should be hashed (e.g., using BCrypt).
- **Error Handling**: The application provides basic error messages but could benefit from more robust exception handling and logging.
- **Admin Privileges**: Admins can manage users and delete any review, while regular users are limited to their own data.