Sure, let's identify and describe each route in the provided Express.js routing module.

Routes Overview:

1. **Middleware: CheckLogged**
   - **Path:** Used in various routes
   - **Method:** Middleware
   - **Description:** This middleware checks if a user is already logged in by verifying the JWT token in the cookies. If the user is logged in, it responds with a status indicating the user is already logged in; otherwise, it allows the request to proceed.
2. **Update User Profile**
   - **Path:** `/update_profile`
   - **Method:** `POST`
   - **Description:** This route updates the user's profile information, such as first name, last name, and profile picture, in the database.
3. **Check If Logged**
   - **Path:** `/checkLogged`
   - **Method:** `GET`
   - **Description:** This route uses the `CheckLogged` middleware to check if the user is logged in. If not logged in, it returns a status indicating the user is not logged in.
4. **User Signup**
   - **Path:** `/signup`
   - **Method:** `POST`
   - **Description:** This route handles user signups, supporting both manual signups (with email and password) and OAuth-based signups. For manual signups, it sends a verification email to the user.
5. **Check Pending Signup**
   - **Path:** `/checkPending`
   - **Method:** `GET`
   - **Description:** This route checks if a user signup is pending by verifying the provided user ID. It's used to check the status of email verification.
6. **Finish Signup**
   - **Path:** `/signup-finish`
   - **Method:** `PUT`

- **Description:** This route completes the user signup process after email verification. It finalizes the user's registration in the system.

7. **User Login**

   - **Path:** `/login`
   - **Method:** `GET`
   - **Description:** This route handles user login, supporting both manual login (with email and password) and OAuth-based login. It verifies user credentials and returns a response indicating the login status.

8. **Request Password Reset**

   - **Path:** `/forgot-request`
   - **Method:** `POST`
   - **Description:** This route handles password reset requests. It generates a secret token and sends a password reset email to the user.

9. **Check Password Reset Request**

   - **Path:** `/forgot-check`
   - **Method:** `GET`
   - **Description:** This route checks the validity of a password reset request by verifying the provided user ID and secret token.

10. **Finish Password Reset**

    - **Path:** `/forgot-finish`
    - **Method:** `PUT`
    - **Description:** This route completes the password reset process by updating the user's password in the database.

Detailed Description of Each Route:

1. Middleware: CheckLogged

- **Purpose:** Verifies the user's JWT token to check if they are logged in.
- **Process:**
  1. Extracts the token from cookies.
  2. Verifies the token using `jwt.verify()`.
  3. If valid, fetches user data and sends a response indicating the user is already logged in.
  4. If invalid, clears the token and proceeds to the next middleware or route handler.

2. Update User Profile

- **Purpose:** Updates user profile information in the database.
- **Process:**
    1. Extracts `email`, `firstName`, `lastName`, and `profilePicture` from the request body.
    2. Updates the user document in the `USER` collection.
    3. Sends a success response or an error message if the update fails.

3. Check If Logged

- **Purpose:** Checks if the user is logged in.
- **Process:**
    1. Uses `CheckLogged` middleware.
    2. If not logged in, responds with a status indicating the user is not logged in.

4. User Signup

- **Purpose:** Handles user signups, both manual and OAuth-based.
- **Process:**
    1. For manual signups, verifies the password length and sends a verification email.
    2. For OAuth signups, verifies the OAuth token and email.
    3. Calls `user.signup()` to create a new user with pending status.
    4. Sends a verification email with a unique link for manual signups.
    5. Responds with a success message or an error message if signup fails.

5. Check Pending Signup

- **Purpose:** Checks if a user signup is pending verification.
- **Process:**
    1. Extracts `_id` from the query parameters.
    2. Calls `user.checkPending()` to check the signup status.
    3. Responds with the signup status or an error message if the check fails.

6. Finish Signup

- **Purpose:** Completes the user signup process after email verification.
- **Process:**
    1. Extracts user data from the request body.
    2. Calls `user.finishSignup()` to finalize the user's registration.
    3. Responds with a success message or an error message if the process fails.

7. User Login

- **Purpose:** Handles user logins, both manual and OAuth-based.
- **Process:**

1. For manual logins, verifies email and password.
2. For OAuth logins, verifies the OAuth token and email.
3. Calls `user.login()` to authenticate the user.
4. Responds with a success message or an error message if login fails.

8. Request Password Reset

- **Purpose:** Initiates a password reset process.
- **Process:**
    1. Extracts `email` from the request body.
    2. Generates a secret token for password reset.
    3. Calls `user.forgotRequest()` to create a password reset request.
    4. Sends a password reset email with a unique link.
    5. Responds with a success message or an error message if the request fails.

9. Check Password Reset Request

- **Purpose:** Verifies the validity of a password reset request.
- **Process:**
    1. Extracts `_id` and `secret` from the query parameters.
    2. Calls `user.forgotCheck()` to check the request validity.
    3. Responds with the request status or an error message if the check fails.

10. Finish Password Reset

- **Purpose:** Completes the password reset process.
- **Process:**
    1. Extracts user data from the request body.
    2. Calls `user.forgotFinish()` to update the user's password.
    3. Responds with a success message or an error message if the process fails.

This module provides a comprehensive set of routes for user authentication, including profile updates, signups, logins, and password resets, with robust error handling and support for both manual and OAuth-based operations.