Use Case 1: User Registration

Actor: Visitor (unauthenticated)
Preconditions: Visitor has navigated to the "Sign Up" page.
Story:

1. The visitor opens the registration form and enters a unique **username**, **email**, and **password**.
2. Upon submission, the front end sends a POST to the **post_create_user** Lambda.
3. Cognito verifies the email uniqueness and creates a new user in the **Users** table with a generated UserId.
4. DynamoDB stores the record with fields like Username, DateJoined, and default empty arrays for Friends, Links, etc.
5. The system returns success; a confirmation email is dispatched via the **send_mail** function.
6. The visitor is redirected to the login page with a success notification.

Use Case 2: User Login

Actor: Registered User
Preconditions: User has a confirmed account (email+)
Story:

1. The user opens the login page and submits **email** and **password**.
2. Cognito authenticates credentials and returns tokens.
3. The front end saves the UserId and tokens in **localStorage**.
4. User is redirected to the main dashboard (index.html), now authenticated, seeing their personal data.

Use Case 3: Profile Management

Actor: Authenticated User
Preconditions: User is logged in.
Story:

1. User clicks "Profile" and is taken to his profile page.
2. Front end calls **get_user_by_id** (via users/get-user-by-id) passing both the profile's owner ID and logged-in user ID.
3. Lambda returns user info, achievements (from **UserAchievements**), and their links.
4. User edits fields (e.g. full name, country, picture).
5. On "Save," front end calls **update_user_profile**, which updates the **Users** table.

6. Confirmation popup appears upon success.

Use Case 4: Create a Short URL

Actor: Authenticated User
Preconditions: User is logged in.
Story:

1. User clicks “New Link,” opening new_item.html.
2. They enter the **long URL**, optional **name**, **description**, toggles for **private** or **password-protected**, and (if needed) a **password**.
3. On submit, front end calls **new_short_url** Lambda with JSON { url, userId, name, description, isPrivate, isPasswordProtected, password }.
4. Lambda validates inputs, generates a unique 6–8 character LinkId, writes a record into **Links** with fields like String (original URL), Date, IsActive=true, etc.
5. Lambda also updates the creator’s **Users.Links** array to include the new LinkId.
6. Front end displays the new shortened URL (redirect.html?code=<LinkId>) and navigates back to the dashboard.

Use Case 5: Accessing a Short URL (Redirection)

Actor: Any Visitor (guest or friend)
Preconditions: A valid LinkId is provided in redirect.html?code=.
Story:

1. Visitor opens the redirect page with ?code=<LinkId>.
2. Front end calls **track_click** Lambda, sending { code: LinkId, userId: loggedInUser? }.
3. Lambda:
    - Fetches the link record.
    - Increments NumberOfClicks.
    - Records click metadata (timestamp, geo-IP) into a separate “Clicks” log (if implemented).
4. Lambda returns a 200 with Location header pointing to the original URL.
5. Browser follows redirect to the long URL.

Use Case 6: Link Privacy & Password Protection

Actor: Link Owner & Visitor
Preconditions: Link has IsPrivate or IsPasswordProtected set.
Story (Visitor):

1. a **private** link does not appear in the public links table in the homepage.
2. For **password-protected** links: front end shows a password modal.
3. Visitor enters password; front end calls **verify_link_password**.
4. If correct, flows into the standard **track_click** + redirect; if incorrect, shows an error.
   Story (Owner managing privacy):
5. From dashboard, owner clicks a toggle next to a link.
6. Front end calls **toggle_link_privacy**, switching IsPrivate.
7. For password changes: **set_link_password**, **change_link_password**, or **remove_link_password** Lambdas update Password and IsPasswordProtected flags.

Use Case 7: Link Lifecycle (Delete & Restore)

Actor: Link Owner
Preconditions: Link exists.
Story (Delete):

1. Owner clicks “Delete” on a link in their dashboard.
2. Front end calls **delete_link**, setting IsActive=false.
3. Link disappears from active lists.
   Story (Restore):
4. Owner goes to “Archived Links” view.
5. Clicks “Restore”; front end calls **restore_link**, setting IsActive=true.
6. Link reappears in active dashboard.

Use Case 8: Click Analytics Overview

Actor: Admin
Preconditions: User has created at least one link.
Story:

1. On dashboard, each link row displays **NumberOfClicks** (populated by **get_link_details**).
2. Lambda **get_all_users_with_stats** can aggregate stats across all links for power users or admins.

Use Case 9: Mailing List Creation & Emailing

Actor: Authenticated User
Preconditions: User is logged in.
Story (Create List):

1. User opens Homepage and clicks “Create Group.”
2. Enters a **Group Name** and selects contacts (by email) or from friends list.
3. Front end calls **create_mailing_list**, which writes to the **Mailing_List** table with ListId, RecipientsEmails, etc.
   Story (Send Email):
4. User selects a list or writes emails manually.
5. Front end calls **send_mail**, which iterates recipients and dispatches emails via google SMTP server.

Use Case 10: Friend Management & Social Sharing

Actor: Authenticated User
Preconditions: User is logged in.
Story (Send Request):

1. User searches for another by username or email in the “Friends” tab.
2. Clicks “Add Friend” on the user’s profile, triggering **send_friend_request** Lambda.
3. Recipient sees a notification.
   Story (Respond to Request):
4. Recipient opens notifications (get_all_notifications, check_unread_notifications).
5. Accepts or rejects via **respond_to_friend_request**, which updates both users’ Friends arrays.
6. On success, both appear in each other’s friends list (get_active_friends) and a notification is received by the user that sent the request.
   Story (Share Link Privately):
7. Owner marks link “private” and shares the direct code URL with a friend.
8. Friends can access that link.

Use Case 11: Notifications Center

Actor: Authenticated User
Preconditions: User is logged in.
Story:

1. On page load, front end polls **check_unread_notifications**.
2. If unread, badge count appears.
3. Clicking opens the full list via **get_all_notifications**, showing texts like "User X sent you a friend request."
4. Each item may link to the relevant resource (e.g. profile, link).

Use Case 12: Admin Dashboard & User Moderation

Actor: Admin (verified via **verify_admin_status**)
Preconditions: User has admin privileges.
Story:

1. Admin logs in and is routed to "Admin Panel."
2. Panel calls **get_all _users** and **get_all _links** for an overview.
3. Admin views usage stats (via **get_all_users_with_stats**).
4. To ban a user, demarks Is Active, invoking **ban_user**, which sets a user's IsActive=false and optionally deactivates their links.
5. System logs the action in notifications.

Use Case 13: S3 Image Upload & Management

Actor: Authenticated User
Preconditions: User is editing their profile

Story (Request Upload URL):

1. Front end calls **request_image_upload_url**, which uploads the image directly to S3 bucket.
2. After upload, front end calls **update_user_profile** (or link metadata) to save the S3 Picture URL in DynamoDB.

Use Case 14: Password Reset (Forgot Password)

Actor: Registered User (but currently locked out)
Preconditions: User is on the "Forgot Password" page.
Story:

1. User enters their **email** in the "Forgot Password" form.
2. Front end calls **AWS Cognito**, which:
    - Verifies that the email is associated with an active user.
    - Sends an email containing a reset link.

3. User clicks the link in their mailbox.
4. the user is shown a form to enter a new password.
5. Upon submission AWS Cognito App Client:
    - Validates strength of new password.
    - Updates the user's Password in the **User Pool**.
6. User is redirected to the login page with a success message.

Use Case 15: Browse All Public Links

Actor: Any Visitor or Authenticated User
Preconditions: Visitor is on the Homepage.
Story:

1. Front end calls **list_public_links** Lambda, which scans **Links** where IsPrivate=false and IsActive=true.
2. Lambda returns paginated results keyed by creation date or popularity.
3. Visitor can scroll the table to view the links.
4. Clicking a public link follows the same **track_click** + redirect flow.

Use Case 16: Search Links

Actor: Any Visitor or Authenticated User
Preconditions: Visitor is on any page with a search bar.
Story:

1. User enters a keyword (e.g., "github") and presses Enter.
2. Front end calls **search_links** Lambda with { query, filters… }.
3. User is sent to search results page.
4. Matching links are returned, ranked by relevance and click count.
5. Results display in a table.

Use Case 17: Achievements & Badges

Actor: Authenticated User
Preconditions: User is logged in.
Story:

1. On dashboard, user clicks "Achievements."
2. Front end calls **get_user_achievements** Lambda, which queries **UserAchievements** by UserId (sorted by DateEarned).

3. For each achievement, the front end fetches the master record from **Achievements** to display Name, IllustrationPicture, and Description.
4. When user reaches a new threshold (e.g. 10 links created), a background process or **track_click** Lambda writes a new **UserAchievements** entry.
5. upon new achievement, it triggers a discrete notification with the badge illustration.