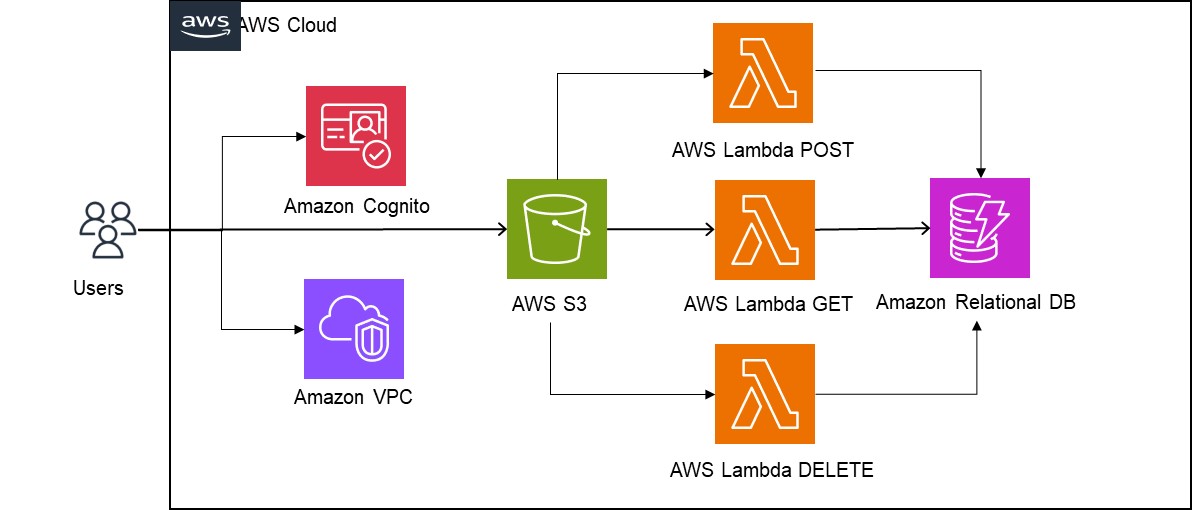
**AWS Personal Expense Tracker Application**

**1. Core Application Overview**

The application will:

* Validate user access
* Provide a landing web page to facilitate managing expense activities
* Depending on what activity is required, the landing page will call the following pages (each will have a return to landing page option):
  + Add expenses – {how will it be displayed is to be decided}
  + View expenses – {how will it be displayed is to be decided}
  + Delete expenses – {how will it be displayed is to be decided}
* Each web activity page will call the required function to update a database.
* The information captured and stored in the database will be:
  + User\_id
  + expense\_date
  + category
  + description
  + amount
* A menu for interaction, options for Add Expense, View Summary (between two dates), Delete Expenses (between two dates), Quit.

**2. Architecture**

**2.1 Diagram**

**2.2 Authentication & User Management**

* **Amazon Cognito**
  + Handles user sign-up, sign-in, and authentication.
  + Each user gets a unique identity (so their expenses are kept private).
  + Supports social logins (Google, Facebook) or username/password.

**2.3 Compute Layer**

* **The application GUI will be stored on web pages**
  + These will be hosted by AWS S3 and have the following pages:
    - Home page
    - Add expenses
    - View expenses
    - Delete expenses
* **The update functions will be serverless**
  + **The function will be enabled by AWS Lambda**. The required functions are:
    - Add expenses
    - View expenses
    - Delete expenses

**2.4 Data Storage**

* **Each user’s expenses will be stored privately:**
* This function will be enabled by RDS (using MySQL):
  + Each expense record has a user\_id (from Cognito) as the primary key + details (amount, category, description, expense\_date).
  + Easy to query by user and date range.

**2.5 IAM (Security & Access Control)**

* Amazon Cognito will handle sign-up & login.
* Amazon API Gateway + AWS Lambda will handle serverless API backend.
* IAM roles and policies control what your backend can access (e.g., write to S3/DynamoDB).
* Cognito, will give **per-user temporary credentials** so users only access their own data.

**3.0 Pseudocode: AWS Expense Tracking Application**

BEGIN EXPENSE\_TRACKER\_APPLICATION

FUNCTION main():

INITIALIZE system resources

CONNECT to database

IF validate\_user\_access() == TRUE THEN

DISPLAY landing\_page()

ELSE

DISPLAY "Access Denied. Invalid user credentials."

TERMINATE program

END IF

END FUNCTION

FUNCTION validate\_user\_access():

DISPLAY "Enter username and password"

READ username, password

IF credentials\_exist\_in\_database(username, password) THEN

RETURN TRUE

ELSE

RETURN FALSE

END IF

END FUNCTION

FUNCTION landing\_page():

WHILE TRUE:

DISPLAY "---- Expense Tracker ----"

DISPLAY "1. Add Expense"

DISPLAY "2. View Expenses (Summary between dates)"

DISPLAY "3. Delete Expenses (between dates)"

DISPLAY "4. Quit"

READ user\_choice

SWITCH (user\_choice):

CASE 1:

CALL add\_expense\_page()

BREAK

CASE 2:

CALL view\_expense\_page()

BREAK

CASE 3:

CALL delete\_expense\_page()

BREAK

CASE 4:

DISPLAY "Goodbye!"

EXIT loop

DEFAULT:

DISPLAY "Invalid choice. Please try again."

END SWITCH

END WHILE

END FUNCTION

FUNCTION add\_expense\_page():

DISPLAY "---- Add Expense ----"

READ expense\_date, category, description, amount

user\_id = CURRENT\_USER\_ID

CALL add\_expense\_to\_database(user\_id, expense\_date, category, description, amount)

DISPLAY "Expense added successfully."

DISPLAY "Return to landing page? (Y/N)"

READ response

IF response == 'Y' THEN

RETURN

END IF

END FUNCTION

FUNCTION add\_expense\_to\_database(user\_id, date, category, description, amount):

EXECUTE SQL:

INSERT INTO expenses (user\_id, expense\_date, category, description, amount)

VALUES (user\_id, date, category, description, amount)

END FUNCTION

FUNCTION view\_expense\_page():

DISPLAY "---- View Expenses ----"

DISPLAY "Enter start date:"

READ start\_date

DISPLAY "Enter end date:"

READ end\_date

expenses = FETCH\_EXPENSES\_FROM\_DB(start\_date, end\_date, CURRENT\_USER\_ID)

DISPLAY "Expenses between " + start\_date + " and " + end\_date

DISPLAY expenses IN chosen\_format (table, list, chart, etc.)

DISPLAY "Return to landing page? (Y/N)"

READ response

IF response == 'Y' THEN

RETURN

END IF

END FUNCTION

FUNCTION FETCH\_EXPENSES\_FROM\_DB(start\_date, end\_date, user\_id):

EXECUTE SQL:

SELECT expense\_date, category, description, amount

FROM expenses

WHERE user\_id = user\_id

AND expense\_date BETWEEN start\_date AND end\_date

RETURN result\_set

END FUNCTION

FUNCTION delete\_expense\_page():

DISPLAY "---- Delete Expenses ----"

DISPLAY "Enter start date:"

READ start\_date

DISPLAY "Enter end date:"

READ end\_date

DISPLAY "Confirm deletion of all expenses between dates? (Y/N)"

READ confirm

IF confirm == 'Y' THEN

CALL delete\_expenses\_from\_database(start\_date, end\_date, CURRENT\_USER\_ID)

DISPLAY "Expenses deleted successfully."

ELSE

DISPLAY "Deletion cancelled."

END IF

DISPLAY "Return to landing page? (Y/N)"

READ response

IF response == 'Y' THEN

RETURN

END IF

END FUNCTION

FUNCTION delete\_expenses\_from\_database(start\_date, end\_date, user\_id):

EXECUTE SQL:

DELETE FROM expenses

WHERE user\_id = user\_id

AND expense\_date BETWEEN start\_date AND end\_date

END FUNCTION

-------------------------------------------------------

FUNCTION terminate\_application():

CLOSE database connection

DISPLAY "Session ended."

EXIT

END FUNCTION

END EXPENSE\_TRACKER\_APPLICATION