INTRODUCTION TO CLOUD

Submitted By:

Hamood Ayoob Khan(23BBCACD209)

Ashfaque Muhammed M(23BBCACD143)

Fidhin Fadhil(23BBCACD195)

Godwin Kuriyakose(23BBCACD201)

Mohammed Nubhaan Shameer(23BBCACD270)

Project Report: Cloud-Based Todo List Web Application

Abstract

This report presents a cloud-based Todo List application built using Flask and AWS DynamoDB. The system provides a robust platform for managing daily tasks with features such as user authentication, task categorization, due dates, and reminders. The goal is to offer users an interactive, responsive, and secure way to organize and track their responsibilities online from any device.

Introduction

Task management is essential in today's fast-paced digital environment. Traditional paper-based lists are inefficient and inaccessible remotely. This project solves that by creating a cloud-integrated Todo application where tasks are persistently stored, users can manage deadlines and reminders, and the interface is optimized for modern usage with dynamic feedback.

Aim of the Application

To develop a full-stack web application that allows users to:

- Register and securely log in.
- Add, edit, and delete personal tasks.
- Set due dates and reminders.
- Store data in the cloud using AWS DynamoDB for high availability and scalability.

Objectives

- Build a secure authentication system with password hashing.
- Design an intuitive interface for task creation and filtering.
- Enable reminders for tasks with real-time notifications.
- Store and retrieve tasks in a scalable cloud database (DynamoDB).
- Implement dark mode for enhanced user experience.

Key Features of the Application

- **User Registration/Login:** Sessions are managed via Flask with hashed credentials.
- **Task Management:** Users can add, complete, edit, and delete tasks.
- **Reminders:** Tasks can be assigned reminders using datetime inputs.
- **Due Dates:** Tasks can be sorted by upcoming deadlines.

- **Persistent Cloud Storage:** All data is stored using AWS DynamoDB.
- **Dark Mode UI:** Toggleable interface with responsive design.
- **Client-Side Interactions:** Built-in JavaScript validations and dynamic updates.

What Has Been Done in This Project

Backend:

- Created Flask routes for user login, registration, and task CRUD operations.
- Integrated AWS DynamoDB with tables for users, todos, and reminders.
- Added session handling and password hashing using Werkzeug.

Frontend:

- Developed HTML pages for login, registration, and the main dashboard.
- Styled with CSS (`styles.css`, `auth.css`) for both light/dark modes.
- Implemented `script.js` for fetching and displaying tasks dynamically, handling reminders with `setTimeout`, and applying filters.

JavaScript Features:

- Reminder notifications with sound alerts.
- Task filtering by status (active, completed, due soon).
- Modal editing of tasks with live updates.

System Architecture

Frontend

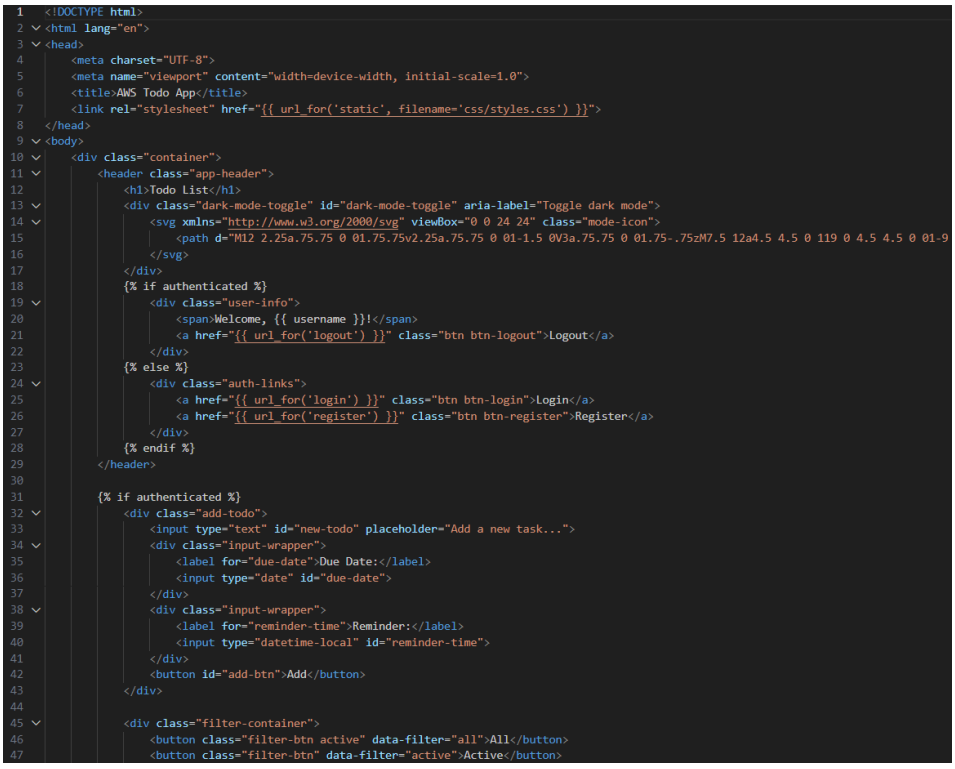
Technologies:

- HTML, CSS, JavaScript

Responsibilities:

- Provide responsive UI for login, registration, and task management
- Display tasks dynamically with filters (All, Active, Completed, Due Soon)
- Accept user input for due dates and reminders
- Trigger notifications with sound and visual alerts
- Support dark mode toggle and real-time UI updates

Frontend Source Code



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>AWS Todo App</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='css/styles.css') }}">
</head>
<body>
    <div class="container">
        <header class="app-header">
            <h1>Todo List</h1>
            <div class="dark-mode-toggle" id="dark-mode-toggle" aria-label="Toggle dark mode">
                <svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 24 24" class="mode-icon">
                    <path d="M12 2.25a.75.75 0 01.75.75v2.25a.75.75 0 01-1.5 0V3a.75.75 0 01.75-.75zM7.5 12a4.5 4.5 0 119 0 4.5 4.5 0 01-9
                </svg>
            </div>
            {% if authenticated %}
                <div class="user-info">
                    <span>Welcome, {{ username }}!</span>
                    <a href="{{ url_for('logout') }}" class="btn btn-logout">Logout</a>
                </div>
            {% else %}
                <div class="auth-links">
                    <a href="{{ url_for('login') }}" class="btn btn-login">Login</a>
                    <a href="{{ url_for('register') }}" class="btn btn-register">Register</a>
                </div>
            {% endif %}
        </header>

        {% if authenticated %}
            <div class="add-todo">
                <input type="text" id="new-todo" placeholder="Add a new task...">
                <div class="input-wrapper">
                    <label for="due-date">Due Date:</label>
                    <input type="date" id="due-date">
                </div>
                <div class="input-wrapper">
                    <label for="reminder-time">Reminder:</label>
                    <input type="datetime-local" id="reminder-time">
                </div>
                <button id="add-btn">Add</button>
            </div>

            <div class="filter-container">
                <button class="filter-btn active" data-filter="all">All</button>
                <button class="filter-btn" data-filter="active">Active</button>
```

Backend

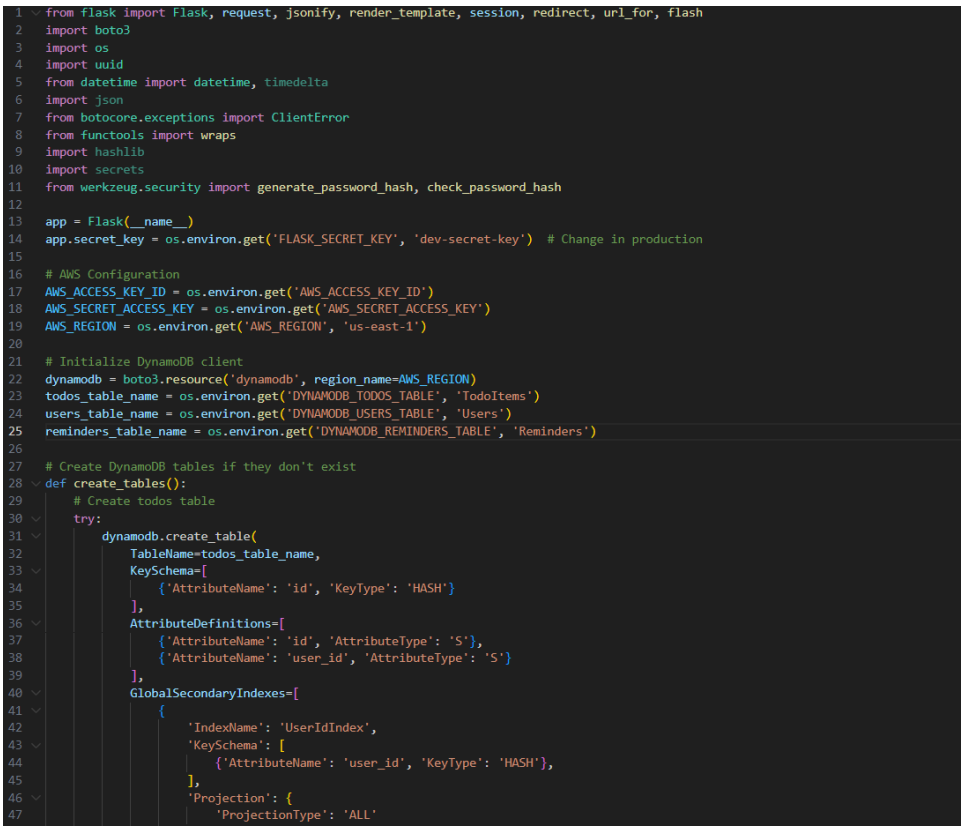
Technologies:

- Python (Flask framework)
- Werkzeug (for password hashing)
- Boto3 (AWS SDK for Python)
- DynamoDB (via AWS SDK)

Responsibilities:

- Handle user authentication (registration, login, logout)
- Manage user sessions securely
- Expose RESTful APIs for CRUD operations on todos and reminders
- Validate and store task/reminder data
- Connect to and interact with AWS DynamoDB tables

Backend Source Code



```python
from flask import Flask, request, jsonify, render_template, session, redirect, url_for, flash
import boto3
import os
import uuid
from datetime import datetime, timedelta
import json
from botocore.exceptions import ClientError
from functools import wraps
import hashlib
import secrets
from werkzeug.security import generate_password_hash, check_password_hash

app = Flask(__name__)
app.secret_key = os.environ.get('FLASK_SECRET_KEY', 'dev-secret-key')  # Change in production

# AWS Configuration
AWS_ACCESS_KEY_ID = os.environ.get('AWS_ACCESS_KEY_ID')
AWS_SECRET_ACCESS_KEY = os.environ.get('AWS_SECRET_ACCESS_KEY')
AWS_REGION = os.environ.get('AWS_REGION', 'us-east-1')

# Initialize DynamoDB client
dynamodb = boto3.resource('dynamodb', region_name=AWS_REGION)
todos_table_name = os.environ.get('DYNAMODB_TODOS_TABLE', 'TodoItems')
users_table_name = os.environ.get('DYNAMODB_USERS_TABLE', 'Users')
reminders_table_name = os.environ.get('DYNAMODB_REMINDERS_TABLE', 'Reminders')

# Create DynamoDB tables if they don't exist
def create_tables():
    # Create todos table
    try:
        dynamodb.create_table(
            TableName=todos_table_name,
            KeySchema=[
                {'AttributeName': 'id', 'KeyType': 'HASH'}
            ],
            AttributeDefinitions=[
                {'AttributeName': 'id', 'AttributeType': 'S'},
                {'AttributeName': 'user_id', 'AttributeType': 'S'}
            ],
            GlobalSecondaryIndexes=[
                {
                    'IndexName': 'UserIdIndex',
                    'KeySchema': [
                        {'AttributeName': 'user_id', 'KeyType': 'HASH'},
                    ],
                    'Projection': {
                        'ProjectionType': 'ALL'
```

Cloud Services

API Used:

- **AWS DynamoDB:** NoSQL database service used for storing:
  - User accounts (Users table)
  - Task items (TodoItems table)
  - Task reminders (Reminders table)

Features:

- Scalable and persistent data storage
- Secondary indexes for querying by user ID and reminder time
- Cloud-hosted storage for high availability
- Integration via Boto3 for seamless interaction with Python backend

Future Scope

- **Email/SMS Notifications:** Integrate AWS SNS or third-party services.
- **Progress Reports:** Weekly summary of tasks completed.
- **File Attachments:** Allow attaching files to tasks.
- **Mobile App:** Create a mobile version with the same backend.
- **Calendar View:** Visualize tasks and reminders on a calendar interface.
- **OAuth Integration:** Allow login with Google/GitHub accounts.

Conclusion

The development of this cloud-based Todo List web application successfully demonstrates the integration of modern web development practices with cloud-native technologies. By leveraging **Flask** for the backend and **AWS DynamoDB** for scalable data storage, the application ensures high performance, reliability, and user-specific task management.

This system not only provides users with essential functionality—like adding, editing, and deleting tasks—but also incorporates advanced features such as due date tracking, customizable reminders, and real-time notifications. The use of **client-side scripting** and dynamic UI feedback enhances the user experience, while **session-based authentication** and secure password storage ensure the protection of user data.

Furthermore, the project architecture is highly modular and adaptable. It supports future enhancements such as calendar integration, mobile compatibility, email/SMS reminders, and third-party logins via OAuth. The interface's responsiveness and dark mode also make the app more inclusive and accessible across devices.