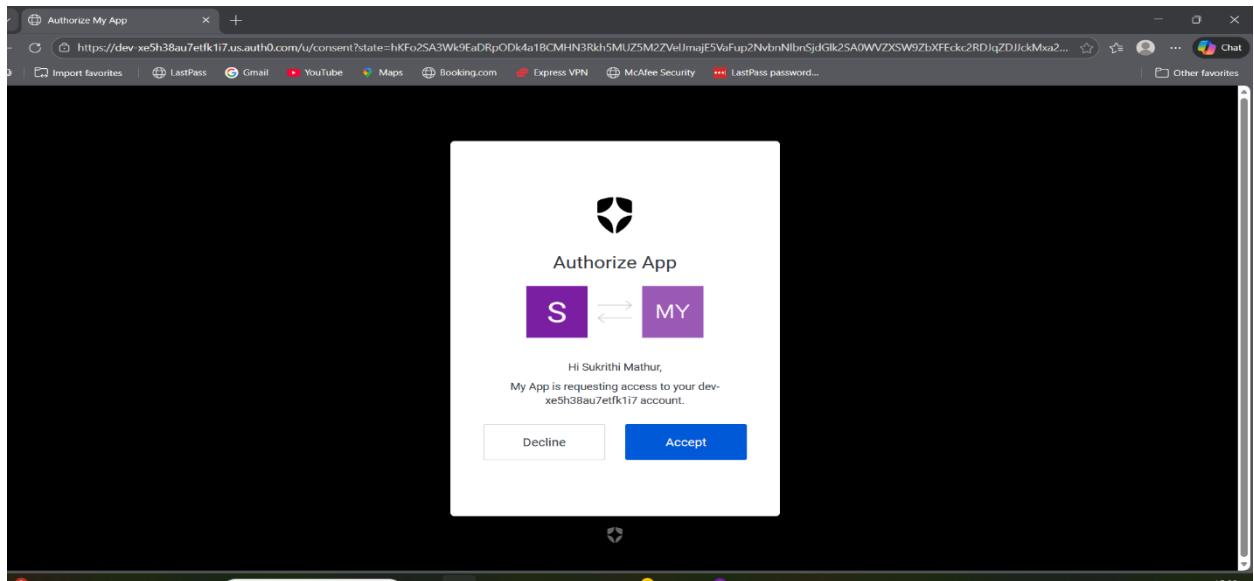
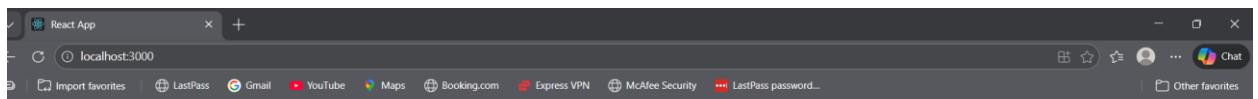
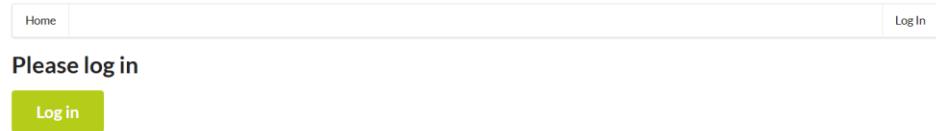
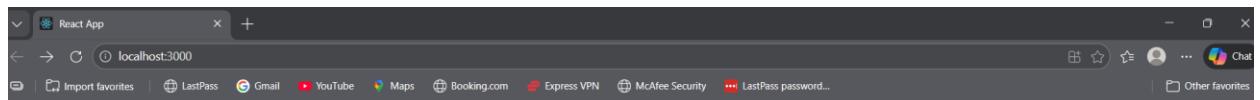


OUTPUTS SCREENSHOTS of REACT APP



Screenshots of successfully compiled Backend codes

The screenshot shows the VS Code interface with the following details:

- EXPLORER**: Shows the project structure for "CD12101-DEPLOY-APPLICATION-WITH-AWS-LAMBDA...".
- CODE**: Displays the content of `createTodo.js`:

```

1 import middy from 'middy';
2 import cors from '@middy/http-cors';
3 import { createTodo } from '../../../../../businessLogic/todos.mjs';
4 import { getUserId } from '../../../../../auth/utils.mjs';

5
6 export const handler = middy()
7   .use(cors({ credentials: true }))
8   .handler(async (event) => {
9     const userId = getUserId(event);
10    const newTodo = JSON.parse(event.body);

11    const item = await createTodo(userId, newTodo);

12    return {
13      statusCode: 201,
14      body: JSON.stringify(item)
15    }
16  })
17
18})
19

```
- TERMINAL**: Shows the command line output for AWS configuration and deployment commands.
- OUTPUT**: Shows logs related to the deployment process.
- PROBLEMS**: Shows no errors or warnings.
- PORTS**: Shows no active ports.
- POWER SHELL**: Shows three sessions: "powershell", "powershell backend", and "node client".

The screenshot shows the VS Code interface with the following details:

- EXPLORER**: Shows the project structure for "CD12101-DEPLOY-APPLICATION-WITH-AWS-LAMBDA...".
- CODE**: Displays the content of `createTodo.js`.
- TERMINAL**: Shows deployment logs:

```

DeleteTodo: todo-app-dev-DeleteTodo (7.1 MB)
GeneratedUploadUrl: todo-app-dev-GenerateUploadUrl (7.1 MB)

1 deprecation found: run 'serverless doctor' for more details

✓ Service deployed to stack todo-app-dev (84s)

endpoints:
GET - https://jm8ghi66x0.execute-api.us-east-1.amazonaws.com/dev/todos
POST - https://jm8ghi66x0.execute-api.us-east-1.amazonaws.com/dev/todos
PATCH [Follow link (ctrl + click)] execute-api.us-east-1.amazonaws.com/dev/todos/{todoId}
DELETE execute-api.us-east-1.amazonaws.com/dev/todos/{todoId}
POST - https://jm8ghi66x0.execute-api.us-east-1.amazonaws.com/dev/todos/{todoId}/attachment
functions:
Auth: todo-app-dev-Auth (7.1 MB)
GetTodos: todo-app-dev-GetTodos (7.1 MB)
CreateTodo: todo-app-dev-CreateTodo (7.1 MB)
UpdateTodo: todo-app-dev-UpdateTodo (7.1 MB)
DeleteTodo: todo-app-dev-DeleteTodo (7.1 MB)
GeneratedUploadUrl: todo-app-dev-GenerateUploadUrl (7.1 MB)

```
- OUTPUT**: Shows logs related to the deployment process.
- PROBLEMS**: Shows no errors or warnings.
- PORTS**: Shows no active ports.
- POWER SHELL**: Shows three sessions: "powershell", "powershell backend", and "node client".

Frontend (Client side codes)- successfull URL

.env (client side) URL'S

REACT_APP_AUTH0_DOMAIN=dev-xe5h38au7etfk1i7.us.auth0.com

REACT_APP_AUTH0_CLIENT_ID=RxKs84QnayvWDUMHJrPxn2ECrdn2mx89

REACT_APP_API_ENDPOINT=https://jm8ghi66x0.execute-api.us-east-1.amazonaws.com/dev

Compiled successfully!

The screenshot shows the VS Code interface with the following details:

- EXPLORER**: Shows the project structure under "CD12101-DEPLOY-APPLICATION-WITH-AWS-LAMBDA".
- CODE EDITOR**: Displays a file named "createTodo.js" with the following code:

```
3 import { createTodo } from '../../../../../businessLogic/todos.mjs'
4 import { getUserId } from '../../../../../auth/utils.mjs'
5
6 export const handler = middy()
7   .use(cors({ credentials: true }))
8   .handler(async (event) => {
9     const userId = getUserId(event)
10    const newTodo = JSON.parse(event.body)
11
12    const item = await createTodo(userId, newTodo)
13
14    return {
15      statusCode: 201,
16      body: JSON.stringify(item),
17    }
18  })
19
20 module.exports = handler
```

- TERMINAL**: Shows the command "npm start" being run, resulting in the message "webpack compiled successfully". It also lists several deprecation warnings related to fs.F_OK and setupMiddlewares.
- PORTS**: Shows ports 3000 and 142:3000 are in use.

App (local URL – React App)

You can now view serverless-todo-app-client in the browser

Local: <http://localhost:3000>

On Your Network: <http://192.168.0.142:3000>

webpack compiled successfully

serverless.yml code

service: todo-app

frameworkVersion: '3'

provider:

name: aws

runtime: nodejs18.x

stage: \${opt:stage, 'dev'}

region: us-east-1

environment:

TODOS_TABLE: Todos-\${self:provider.stage}-\${opt:uniqueId, '1234'}

TODOS_CREATED_AT_INDEX: CreatedAtIndex

ATTACHMENT_S3_BUCKET: todo-attachments-\${self:provider.stage}-\${opt:uniqueId, '1234'}

iam:

role:

statements:

- Effect: Allow

Action:

- dynamodb:Query
- dynamodb:PutItem
- dynamodb:UpdateItem
- dynamodb:DeleteItem

Resource:

- arn:aws:dynamodb:*:*:table/\${self:provider.environment.TODOS_TABLE}

-

arn:aws:dynamodb:*:*:table/\${self:provider.environment.TODOS_TABLE}/index/\${self:provider.environment.TODOS_CREATED_AT_INDEX}

- Effect: Allow

Action:

- s3:PutObject

Resource:

- arn:aws:s3:::\${self:provider.environment.ATTACHMENT_S3_BUCKET}/*

functions:

Auth:

handler: src/lambda/auth/auth0Authorizer.handler

GetTodos:

handler: src/lambda/http/getTodos.handler

events:

- http:
 - method: get
 - path: todos
 - cors: true
 - authorizer: Auth

CreateTodo:

handler: src/lambda/http/createTodo.handler

events:

- http:
 - method: post
 - path: todos
 - cors: true
 - authorizer: Auth

UpdateTodo:

handler: src/lambda/http/updateTodo.handler

events:

- http:
 - method: patch
 - path: todos/{todoId}
 - cors: true
 - authorizer: Auth

DeleteTodo:

handler: src/lambda/http/deleteTodo.handler

events:

- http:

```
method: delete  
path: todos/{todoId}  
cors: true  
authorizer: Auth
```

```
GenerateUploadUrl:  
  handler: src/lambda/http/generateUploadUrl.handler  
  events:  
    - http:  
      method: post  
      path: todos/{todoId}/attachment  
      cors: true  
      authorizer: Auth
```

resources:

Resources:

TodosTable:

Type: AWS::DynamoDB::Table

Properties:

AttributeDefinitions:

- AttributeName: userId

- AttributeName: S

- AttributeName: todoId

- AttributeName: S

- AttributeName: createdAt

- AttributeName: S

KeySchema:

- AttributeName: userId

- KeyType: HASH

- AttributeName: todoid
 - KeyType: RANGE
- BillingMode: PAY_PER_REQUEST
- TableName: \${self:provider.environment.TODOS_TABLE}
- LocalSecondaryIndexes:
 - IndexName: \${self:provider.environment.TODOS_CREATED_AT_INDEX}
 - KeySchema:
 - AttributeName: userId
 - KeyType: HASH
 - AttributeName: createdAt
 - KeyType: RANGE
 - Projection:
 - ProjectionType: ALL

AttachmentsBucket:

Type: AWS::S3::Bucket

Properties:

BucketName: \${self:provider.environment.ATTACHMENT_S3_BUCKET}

CorsConfiguration:

CorsRules:

- AllowedOrigins:
 - '*'
- AllowedHeaders:
 - '*'
- AllowedMethods:
 - PUT
 - GET

CD12101-DEPLOY-APPLICATION

EXPLORER

- > **starter**
 - > **backend**
 - > **serverless**
 - > **node_modules**
 - > **src**
 - > **auth**
 - JS utils.mjs**
 - > **businessLogic**
 - JS todos.mjs**
 - > **dataLayer**
 - JS todosAccess.mjs**
 - > **fileStorage**
 - JS attachmentUtils.mjs**
 - > **lambda**
 - > **auth**
 - JS auth0Authorizer.mjs**
 - > **http**
 - JS createTodo.js**
 - JS deleteTodo.js**
 - JS generateUploadUrl.js**
 - JS getTodos.js**
 - JS updateTodo.js**
 - JS utils.mjs**
 - > **utils**
 - .gitignore**
 - package-lock.json**
 - package.json**
 - serverless.yml**
 - > **client**
 - .gitignore**

JS todosAccess.mjs

```

1 import { decode } from 'jsonwebtoken'
2 import { createLogger } from '../utils/logger.mjs'
3
4 const logger = createLogger('utils')
5 /**
6  * Parse a JWT token and return a user id
7  * @param jwtToken JWT token to parse
8  * @returns a user id from the JWT token
9 */
10 export function parseUserId(jwtToken) {
11   const decodedJwt = decode(jwtToken)
12   return decodedJwt.sub
13 }
14

```

TERMINAL

```

PS C:\Users\hp\Downloads\cd12101-Deploy-Application-with-AWS-Lambda-main (1)\cd12101-Deploy-Application-with-AWS-Lambda-main\starter> npm start
recated. Please use the 'setupMiddlewares' option.
Starting the development server...
Compiled successfully!

```

You can now view **serverless-todo-app-client** in the browser.

Compiled successfully!

You can now view **serverless-todo-app-client** in the browser.

Local: <http://localhost:3000>
On Your Network: <http://192.168.0.142:3000>

CD12101-DEPLOY-APPLICATION

EXPLORER

 - > **starter**
 - > **backend**
 - > **serverless**
 - > **node_modules**
 - > **src**
 - > **auth**
 - JS utils.mjs**
 - > **businessLogic**
 - JS todos.mjs**
 - > **dataLayer**
 - JS todosAccess.mjs**
 - > **fileStorage**
 - JS attachmentUtils.mjs**
 - > **lambda**
 - > **auth**
 - JS auth0Authorizer.mjs**
 - > **http**
 - JS createTodo.js**
 - JS deleteTodo.js**
 - JS generateUploadUrl.js**
 - JS getTodos.js**
 - JS updateTodo.js**
 - JS utils.mjs**
 - > **utils**
 - .gitignore**
 - package-lock.json**
 - package.json**
 - serverless.yml**
 - > **client**
 - .gitignore**

JS todosAccess.mjs

```

1 import AWS from 'aws-sdk'
2
3 const docClient = new AWS.DynamoDB.DocumentClient()
4 const TODOS_TABLE = process.env.TODOS_TABLE
5 const INDEX = process.env.CREATED_AT_INDEX
6
7 export class TodosAccess {
8   async getTodos(userId) {
9     const result = await docClient.query({
10       TableName: TODOS_TABLE,
11       IndexName: INDEX,
12       KeyConditionExpression: 'userId = :userId',
13       ExpressionAttributeValues: {
14         ':userId': userId
15       }
16     }).promise()
17
18     return result.Items
19   }
20

```

TERMINAL

```

PS C:\Users\hp\Downloads\cd12101-Deploy-Application-with-AWS-Lambda-main (1)\cd12101-Deploy-Application-with-AWS-Lambda-main\starter> npm start
recated. Please use the 'setupMiddlewares' option.
Starting the development server...
Compiled successfully!

```

You can now view **serverless-todo-app-client** in the browser.

Compiled successfully!

You can now view **serverless-todo-app-client** in the browser.

Local: <http://localhost:3000>
On Your Network: <http://192.168.0.142:3000>

CD12101-DEPLOY-APPLICATION

EXPLORER

 - > **starter**
 - > **backend**
 - > **serverless**
 - > **node_modules**
 - > **src**
 - > **auth**
 - JS utils.mjs**
 - > **businessLogic**
 - JS todos.mjs**
 - > **dataLayer**
 - JS todosAccess.mjs**
 - > **fileStorage**
 - JS attachmentUtils.mjs**
 - > **lambda**
 - > **auth**
 - JS auth0Authorizer.mjs**
 - > **http**
 - JS createTodo.js**
 - JS deleteTodo.js**
 - JS generateUploadUrl.js**
 - JS getTodos.js**
 - JS updateTodo.js**
 - JS utils.mjs**

JS todos.mjs

```

1 import middy from '@middy/core'
2 import cors from '@middy/http-cors'
3 import { createTodo } from '../../../../../businessLogic/todos.mjs'
4 import { getuserId } from '../../auth/utils.mjs'
5
6 export const handler = middy()
7   .use(cors({ credentials: true }))
8   .handler(async (event) => {
9     const userId = getuserId(event)
10    const newTodo = JSON.parse(event.body)
11
12    const item = await createTodo(userId, newTodo)
13
14    return {
15      statusCode: 201,
16      body: JSON.stringify({ item })
17    }
18  })
19

```

TERMINAL

```

PS C:\Users\hp\Downloads\cd12101-Deploy-Application-with-AWS-Lambda-main (1)\cd12101-Deploy-Application-with-AWS-Lambda-main\starter> npm start
recated. Please use the 'setupMiddlewares' option.
Starting the development server...
Compiled successfully!

```

The screenshot shows the Okta Developer Platform interface. On the left, a sidebar navigation includes links like 'Getting Started', 'AI Agents', 'Activity', 'Applications' (selected), 'APIs', 'SSO Integrations', 'Authentication', 'Organizations', 'User Management', 'Branding', 'Security', 'Actions', 'Event Streams (EARLY)', 'Monitoring', 'Marketplace', 'Extensions', and 'Settings'. A 'Get support' button is also present.

The main content area has tabs for 'Quickstart', 'Settings' (selected), 'APIs', 'Addons', 'Connections', 'Login Experience', and 'Okta Integration Network (NEW)'. The 'Settings' tab contains sections for 'Basic Information' (Name: 'My App', Domain: 'dev-xe5h38au7etfk1i7.us.auth0.com', Client ID: 'RxKs84QnayvWIDUMHJrPxn2ECrdn2mx89', Client Secret: masked), 'Description' (Add a description in less than 140 characters), and a note that the Client Secret is not base64 encoded. Below these is a text box for a free text description of the application, with a character count limit of 140.

The bottom half of the screen shows a code editor for a 'todosAccess.mjs' file within a 'CD12101-DEPLOY-APPLICATION-WITH-AWS-LAMBDA' project. The code defines a Lambda function handler that uses middy and cors to handle HTTP requests, update a todo item in a database, and return a response. The code editor includes tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which shows a command prompt for npm start), and 'PORTS'. The terminal output indicates the application is running on localhost:3000. A sidebar on the right lists available tools: 'powershell', 'powerhell backend', and 'node client'.

Created of codes (folder as well as files)

The screenshot shows the VS Code Explorer sidebar with the following project structure:

- CD12101-DEPLOY-APPLICATION-WITH-AWS-LAMB...** (expanded)
 - .github
 - starter** (expanded)
 - backend** (expanded)
 - .serverless
 - node_modules
 - src** (expanded)
 - auth
 - businessLogic** (expanded)
 - todos.mjs
 - dataLayer** (expanded)
 - todosAccess.mjs
 - fileStorage** (expanded)
 - attachmentUtils.mjs
 - lambda** (expanded)
 - auth
 - http** (expanded)
 - createTodo.js
 - deleteTodo.js
 - generateUploadUrl.js
 - getTodos.js
 - updateTodo.js
 - utils.mjs
 - utils** (expanded)
 - .gitignore
 - package-lock.json
 - package.json
 - serverless.yml
 - client** (expanded)
 - .gitignore
 - prettierrc.json