Project: QUICKSILVER
AWS Lambda Function Skeleton Using JavaScript

-The following includes function skeletons for the four major functions of this app, sign up, sign in, edit profile, process payment, navigate main menu.

✅✅✅
Sign Up Skeleton:

```
// signup.js
const AWS = require('aws-sdk');
const dynamoDB = new AWS.DynamoDB.DocumentClient();
const cognito = new AWS.CognitoIdentityServiceProvider();

exports.handler = async (event, context) => {
    try {
        // Parse the user data from the request body
        const userData = JSON.parse(event.body);

        // Example: Create a new user in Cognito
        const params = {
            ClientId: process.env.COGNITO_CLIENT_ID, // Your Cognito App Client ID
            Username: userData.email,
            Password: userData.password,
            UserAttributes: [
                { Name: 'email', Value: userData.email }
            ]
        };

        // Create the user in Cognito
        await cognito.signUp(params).promise();

        // Optionally, store additional user details in DynamoDB
        const dbParams = {
            TableName: process.env.USERS_TABLE,
            Item: {
                userId: userData.email,
                name: userData.name,
                createdAt: new Date().toISOString()
            }
        };

        // Save user details in DynamoDB
        await dynamoDB.put(dbParams).promise();
```

```
        // Prepare the response
        const response = {
            statusCode: 201,
            body: JSON.stringify({
                message: "User registration successful"
            })
        };
        return response;
    } catch (error) {
        console.error("Error during user sign-up:", error);
        return {
            statusCode: 500,
            body: JSON.stringify({ message: "User registration failed" })
        };
    }
};
```

✅✅✅
Sign In Skeleton:

```
// signin.js
const AWS = require('aws-sdk');
const cognito = new AWS.CognitoIdentityServiceProvider();

exports.handler = async (event, context) => {
    try {
        // Parse the sign-in details from the request body
        const signInData = JSON.parse(event.body);

        const params = {
            AuthFlow: 'USER_PASSWORD_AUTH',
            ClientId: process.env.COGNITO_CLIENT_ID, // Your Cognito App Client ID
            AuthParameters: {
                USERNAME: signInData.email,
                PASSWORD: signInData.password
            }
        };

        // Initiate the authentication flow
        const authResult = await cognito.initiateAuth(params).promise();

        // Prepare the response with authentication tokens
        const response = {
```

```
            statusCode: 200,
            body: JSON.stringify({
                message: "Sign-in successful",
                idToken: authResult.AuthenticationResult.IdToken,
                accessToken: authResult.AuthenticationResult.AccessToken,
                refreshToken: authResult.AuthenticationResult.RefreshToken
            })
        };
        return response;
    } catch (error) {
        console.error("Error during sign-in:", error);
        return {
            statusCode: 401,
            body: JSON.stringify({ message: "Invalid credentials" })
        };
    }
};
```

✅✅✅
Edit Profile Skeleton:

```
// editProfile.js
const AWS = require('aws-sdk');
const dynamoDB = new AWS.DynamoDB.DocumentClient();
const cognito = new AWS.CognitoIdentityServiceProvider();

exports.handler = async (event, context) => {
    try {
        // Parse the updated profile data from the request body
        const profileData = JSON.parse(event.body);

        // Update user profile in DynamoDB
        const dbParams = {
            TableName: process.env.USERS_TABLE,
            Key: {
                userId: profileData.email // Assuming email is used as userId
            },
            UpdateExpression: 'set #name = :name, #phone = :phone, #address = :address', // Add
more attributes as needed
            ExpressionAttributeNames: {
                '#name': 'name',
                '#phone': 'phone',
                '#address': 'address'
            },
```

```
            ExpressionAttributeValues: {
                ':name': profileData.name,
                ':phone': profileData.phone,
                ':address': profileData.address
            },
            ReturnValues: 'UPDATED_NEW'
        };

        // Execute the update
        await dynamoDB.update(dbParams).promise();

        // Optionally, update user attributes in Cognito if needed
        if (profileData.name || profileData.phone) {
            const cognitoParams = {
                UserPoolId: process.env.COGNITO_USER_POOL_ID,
                Username: profileData.email,
                UserAttributes: []
            };

            if (profileData.name) {
                cognitoParams.UserAttributes.push({ Name: 'name', Value: profileData.name });
            }
            if (profileData.phone) {
                cognitoParams.UserAttributes.push({ Name: 'phone_number', Value:
profileData.phone });
            }

            // Update attributes in Cognito
            if (cognitoParams.UserAttributes.length > 0) {
                await cognito.adminUpdateUserAttributes(cognitoParams).promise();
            }
        }

        // Prepare the response
        const response = {
            statusCode: 200,
            body: JSON.stringify({ message: "Profile updated successfully" })
        };
        return response;
    } catch (error) {
        console.error("Error updating profile:", error);
        return {
            statusCode: 500,
            body: JSON.stringify({ message: "Profile update failed" })
```

```
        };
    }
};
```

✅✅✅
Receive Toll Info & Process Payment Skeleton:

```
// paymentProcessor.js
const AWS = require('aws-sdk');
const dynamoDB = new AWS.DynamoDB.DocumentClient();
const paymentGateway = require('payment-gateway-sdk'); // Replace with actual SDK

exports.handler = async (event, context) => {
    try {
        // Parse payment details from the event
        const paymentData = JSON.parse(event.body);

        // Step 1: Verify the toll amount from the database
        const tollParams = {
            TableName: process.env.TOLL_HISTORY_TABLE,
            KeyConditionExpression: 'vehicleId = :vehicleId',
            ExpressionAttributeValues: {
                ':vehicleId': paymentData.vehicleId
            }
        };

        // Query the database for the toll amount
        const tollResult = await dynamoDB.query(tollParams).promise();
        if (tollResult.Items.length === 0) {
            throw new Error('Toll information not found.');
        }

        const expectedTollAmount = tollResult.Items[0].amount;

        // Step 2: Verify that the provided amount matches the expected toll amount
        if (paymentData.amount !== expectedTollAmount) {
            throw new Error('Payment amount does not match the expected toll amount.');
        }

        // Step 3: Create the payment charge using the payment gateway
        const charge = await paymentGateway.createCharge({
            amount: paymentData.amount,
            currency: 'USD',
            source: paymentData.paymentToken,
```

```
            description: `Toll payment for vehicle ID: ${paymentData.vehicleId}`
        });

        // Step 4: Update the toll history in the database
        const updateParams = {
            TableName: process.env.TOLL_HISTORY_TABLE,
            Key: { vehicleId: paymentData.vehicleId },
            UpdateExpression: 'set paymentStatus = :status, paymentTransactionId =
:transactionId',
            ExpressionAttributeValues: {
                ':status': 'Paid',
                ':transactionId': charge.id
            }
        };

        await dynamoDB.update(updateParams).promise();

        // Step 5: Prepare the response
        const response = {
            statusCode: 200,
            body: JSON.stringify({
                message: "Payment successful",
                transactionId: charge.id
            })
        };
        return response;
    } catch (error) {
        console.error("Error processing payment:", error);
        return {
            statusCode: 500,
            body: JSON.stringify({ message: "Payment processing failed" })
        };
    }
};
```

✅✅✅
Main Menu Skeleton:

```
// mainMenu.js
exports.handler = async (event, context) => {
    try {
        // Define the main menu options
        const menuOptions = {
            statusCode: 200,
```

```
            body: JSON.stringify({
                message: "Main Menu",
                options: [
                    { name: "View Toll History", action: "viewTollHistory" },
                    { name: "Edit Profile", action: "editProfile" },
                    { name: "Sign Out", action: "signOut" }
                ]
            })
        };
        return response;
    } catch (error) {
        console.error("Error displaying main menu:", error);
        return {
            statusCode: 500,
            body: JSON.stringify({ message: "Failed to load main menu" })
        };
    }
};
```