GitHub-Asana Integration Service

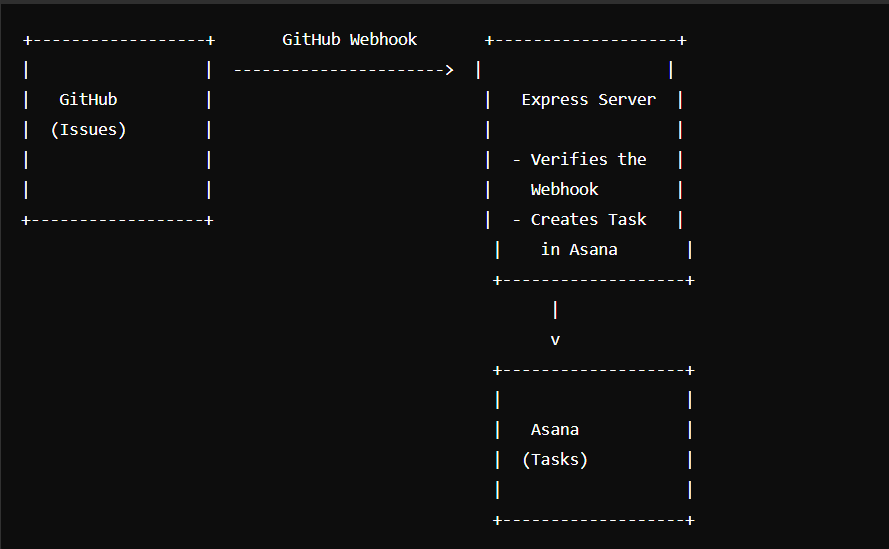
# Overview

This project demonstrates the creation of an integration service that links GitHub and Asana. The service automatically creates a task in Asana whenever a new issue is created on GitHub. The service is built using Node.js and makes use of Express.js for handling HTTP requests, Axios for making API calls, and the Crypto library for securing the webhook

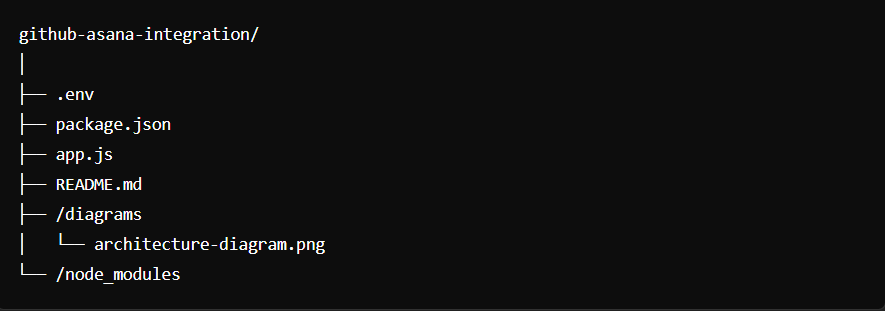
# Key Features

* **Automatic Task Creation**: When a new issue is created on GitHub, a corresponding task is created in Asana with relevant details such as the issue title, description, URL, and creator.
* **Webhook Validation**: The service includes validation to ensure that the requests received from GitHub are authentic and not tampered with.
* **No Pre-built Packages**: The service is implemented without using pre-built packages for Asana, showcasing manual integration via REST APIs

## Architecture Diagram



# Project Structure

* **app.js**: The main file where the Express server is initialized, and routes are defined.
* **.env**: Environment variables are stored here, including sensitive data like Asana Access Token, GitHub Secret, and Asana Project ID.

# Code Explanation

javascript

Copy code

require("dotenv").config();  
  
const express = require("express");  
const bodyParser = require("body-parser");  
const axios = require("axios");  
const crypto = require("crypto");  
  
const app = express();  
const PORT = process.env.PORT || 3000;  
  
const ASANA\_ACCESS\_TOKEN = process.env.ASANA\_ACCESS\_TOKEN;  
const GITHUB\_SECRET = process.env.GITHUB\_SECRET;  
const ASANA\_PROJECT\_ID = process.env.ASANA\_PROJECT\_ID;  
  
app.use(bodyParser.json());

Environment Setup:

* The dotenv package is used to load environment variables from a .env file. This ensures that sensitive data like API tokens are not hardcoded. Express is used for setting up the server, and bodyParser is used to parse incoming JSON requests.

## Environment Variables:

* ASANA\_ACCESS\_TOKEN: Used to authenticate requests to the Asana API.
* GITHUB\_SECRET: Used to verify the authenticity of GitHub webhooks.
* ASANA\_PROJECT\_ID: The ID of the Asana project where tasks should be created.

app.post("/github-webhook", (req, res) => {

const signature = req.headers["x-hub-signature-256"];

if (!verifyWebhook(req.body, signature)) {

return res.status(403).send("Invalid signature");

}

const issue = req.body.issue;

createAsanaTask(issue)

.then(() => res.status(201).send("Task created in Asana"))

.catch((err) =>

res.status(500).send("Error creating task in Asana: " + err.message)

);

});

## Webhook Endpoint:

* **POST /github-webhook**: This endpoint is triggered whenever a new issue is created on GitHub.
* **Webhook Verification**: The verifyWebhook function is called to validate the request using the signature provided by GitHub.
* **Task Creation**: If the webhook is verified, the createAsanaTask function is called to create a new task in Asana with the issue details.

function verifyWebhook(body, signature) {

const hmac = crypto.createHmac("sha256", GITHUB\_SECRET);

const digest = `sha256=${hmac.update(JSON.stringify(body)).digest("hex")}`;

if (signature !== digest) {

throw new Error("Invalid webhook signature");

}

return true;

}

## Webhook Verification:

* This function ensures that the request received is genuinely from GitHub. The HMAC (Hash-based Message Authentication Code) is generated using the GitHub secret, and it is compared with the signature provided in the request header.
* If the signatures don't match, an error is thrown, and the request is rejected.

async function createAsanaTask(issue) {

const taskData = {

data: {

name: issue.title,

notes: issue.body,

projects: [ASANA\_PROJECT\_ID],

assignee: issue.login,

},

};

try {

const response = await axios.post(

"https://app.asana.com/api/1.0/tasks",

taskData,

{

headers: {

Authorization: `Bearer ${ASANA\_ACCESS\_TOKEN}`,

"Content-Type": "application/json",

},

}

);

console.log("Task created in Asana:", response.data);

return response.data;

} catch (error) {

console.error(

"Error creating task in Asana:",

error.response ? error.response.data : error.message

);

throw new Error("Failed to create task in Asana");

}

}

## Asana Task Creation:

* **createAsanaTask Function**: This function is responsible for making a POST request to the Asana API to create a new task.
* **Task Data**: The task details, including the title, description, project ID, and assignee, are set using the data received from the GitHub issue.
* **Error Handling**: If the task creation fails, an error is logged, and the function throws an error to be handled by the calling function.

// Start the server

app.listen(PORT, () => {

console.log(`Server is running on http://localhost:${PORT}`);

});

**Server Initialization**:

* The server listens on the specified port (defaulting to 3000 if not set in the environment variables).

#### **Running the Service**

1. **Environment Setup**: Create a .env file in the root directory with the following variables

ASANA\_ACCESS\_TOKEN=your\_asana\_access\_token

GITHUB\_SECRET=your\_github\_webhook\_secret

ASANA\_PROJECT\_ID=your\_asana\_project\_id

1. **Install Dependencies**:

npm install express body-parser axios crypto dotenv

1. **Run the Server:**

node app.js

1. **Configure GitHub Webhook**:

* Set up a webhook in your GitHub repository that points to your server’s /github-webhook endpoint.
* Use the secret key you defined in the .env file for secure communication.

# Conclusion

This integration service efficiently automates the task creation process between GitHub and Asana, leveraging webhooks for real-time updates and REST APIs for task management. The service ensures secure and reliable communication by verifying webhooks and handling API responses carefully.