**Securing a REST API with JWT Authentication**

**Introduction**

This project is a microservices-based authentication system using Node.js, Express, JWT (JSON Web Tokens), and Docker. It consists of two main services:

1. Auth Service: Handles user authentication and generates JWT tokens.
2. Resource Service: A protected service that validates JWT tokens before allowing access.

**What is JWT (JSON Web Token)?**

JWT (JSON Web Token) is a compact and self-contained way to transmit information between parties as a JSON object.

How JWT Works?

1. User logs in by providing their credentials.
2. The Auth Service verifies credentials and generates a JWT token.
3. The token is sent to the client and stored (e.g., in local storage or HTTP cookies).
4. When accessing a protected resource, the client sends the token in the Authorization header.
5. The Resource Service verifies the token before allowing access.

**Structure of a JWT Token**

A JWT consists of three parts:

HEADER.PAYLOAD.SIGNATURE

Example JWT: 

**Project Structure**

**A screenshot of a computer program

Description automatically generated**

**Setup & Installation**

Prerequisites: install Docker and Node.js

**API Endpoints**

Auth Service:

|  |  |  |
| --- | --- | --- |
| **Method** | **Endpoint** | **Description** |
| GET | /auth | Basic authentication route check |
| POST | /auth/login | Logs in a user and returns a JWT token |
| GET | /auth-routes | Loads additional authentication routes |

Resource Service:

|  |  |  |
| --- | --- | --- |
| **Method** | **Endpoint** | **Description** |
| POST | /resource | Access a protected resource (requires JWT in body) |
| GET | /resource-routes | Loads additional resource routes |