Jobs API

Rohan Saraswat
12008941
Roll No.: 45
Submitted to : Balwinder Kaur Dhaliwal

Project Description

- Jobs API : multiple users can register themselves and add their jobs, only the jobs created by this particular user will to displayed
- User Authorization using JSON Web Token
- Create Read Update Delete Functionality
- GetAllJobs, get all the jobs created by all the users using get request
- Get Jobs created by a particular user using get request
- A particular User can create a Job using post request
- Job Details can be updated by a user using patch request
- User can Delete his/her Job using delete request
- API calls using Postman
- Implemented Custom Error Messages
- Used http codes to send the correct status of requests

Technologies Used

- **Node JS**: JS runtime environment, allows you to run JavaScript on the server.
- **Express JS:** back end web application framework for building RESTful APIs
- **MongoDB:** MongoDB is a document database. It stores data in a type of JSON format called BSON.
- **Mongoose:** JavaScript library that creates a connection between MongoDB and the Node.js JavaScript runtime environment
- **JSON Web Token:** a JSON payload containing a particular claim.

Technologies Used

- **Bcrypt JS**: Bcrypt turns a simple password into fixed-length characters called a hash. Before hashing a password, bcrypt applies a salt — a unique random string that makes the hash unpredictable
- **Express-async-errror**: Library to catch errors at runtime without using try/catch blocks in your async functions.
- **Http-status-codes:** to indicate status of request received by server
- **Dotenv:** Dotenv is a zero-dependency module that loads environment variables from a .env file into process.env.
- **Postman:** Postman is a software testing API (Application Programming Interface) platform to build, test, design, modify, and document.

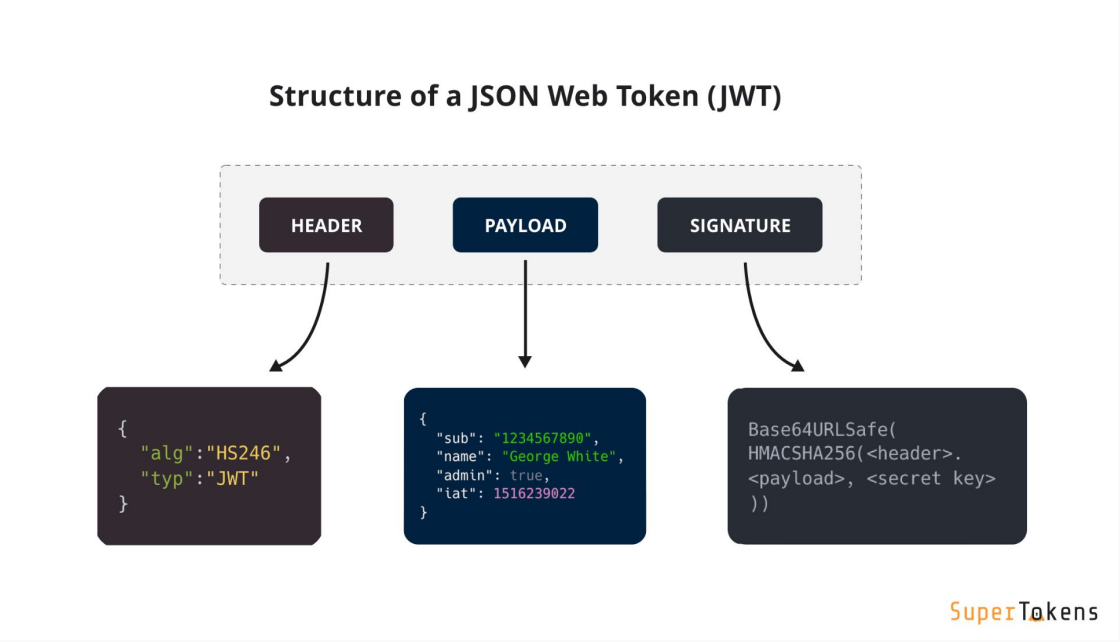
Application Workflow

- App.js
  - /routes/auth.js
    - /controllers/auth.js
      - register
      - login
  - /routes/jobs.js
    - /controllers/jobs.js
      - getAllJobs
      - getJob
      - createJob
      - updateJob
      - Delete Job

JWT Working

Authorization: This is the most common scenario for using JWT. Once the user is logged in, each subsequent request will include the JWT, allowing the user to access routes, services, and resources that are permitted with that token.

Information Exchange: JSON Web Tokens are a good way of securely transmitting information between parties. Because JWTs can be signed you can be sure the senders are who they say they are. Additionally, as the signature is calculated using the header and the payload, you can also verify that the content hasn't been tampered with.



- **Header** The header *typically* consists of two parts: the type of the token, which is JWT, and the signing algorithm being used, such as HMAC SHA256 or RSA.
- **Payload** The second part of the token is the payload, which contains the claims. Claims are statements about an entity (typically, the user) and additional data.
- **Signature** To create the signature part you have to take the encoded header, the encoded payload, a secret, the algorithm specified in the header, and sign that.

Thank You