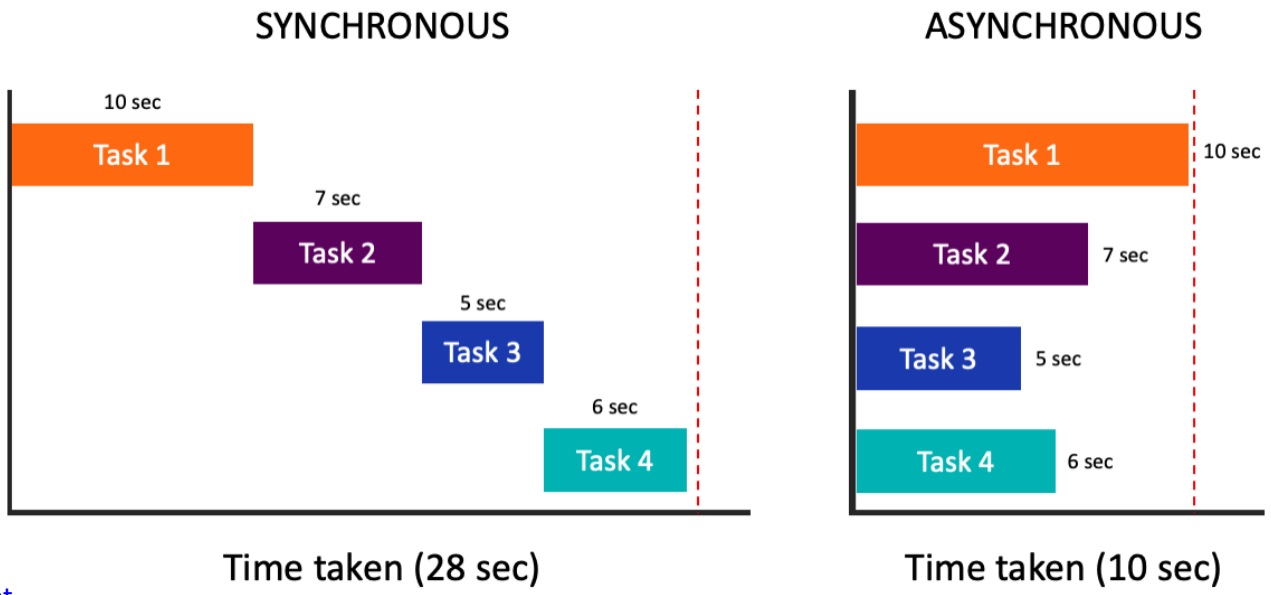
Chapter 3
Backend Application with Nodejs and Express

Instructor : Amonrat Prasitsupparote

Nodejs

- Formerly, JavaScript was designed for a static webpage on a browser only.
- Node.js is an open-source and cross-platform JavaScript runtime environment.
  - Allows JavaScript to be no longer required to run in the browser.
  - Using Asynchronous and event driven are the main concept.



Ref. https://scoutapm.com/blog/async-javascript

Getting started to Nodejs

- Install Nodejs
  - Download at https://nodejs.org
- Run Nodejs file
  - node <FILE_NAME>

```
const a = 10;
const b = 20;
console.log("a*(a+b)=" + a*(a + b));
```

Asynchronous Operations

- Callback is an asynchronous function
  - Called at the completion of a given task.
  - Equivalent to synchronous code

```
Function_Name(param1,param2,…, callback)
```

```
const myRequest = (data, callback) => {
  const response = 10 + data;
  const error = undefined;
  const result = callback(error, response);
  return result;
};
const result = myRequest(5, (err, res) => {
  if (err) return "Got error";
  else return res;
});
console.log("result is " + result);
```

Asynchronous Operations (Cont.)

- Promise is a way to deal with asynchronous code, without getting stuck in callback hell (complex nested callbacks).
  - Once a promise has been called, it will start in a pending state.
  - End in a resolved state, or in a rejected state (passed to then and catch)

```
New Promise((resolve, reject)=>{…})
```

```
const getStd = (id) => {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      let name = "James"; //undefined
      if (name) resolve({ id: id, name: name });
      else reject(new Error("Cannot get the result"));
    }, 2000);
  });
};
getStd(4)
  .then((res) => {
    console.log(res);
  })
  .catch((err) => {
    console.log(err);
  });
```

Asynchronous Operations (Cont.)

- Async & Await is a modern asynchronous code, built on promises
  - Reduce the complexity of promises
  - When call the async function, prepend await
    - The calling code will stop until the promise is resolved or rejected

```
async Function_Name(){
//code
}
                OR
const Function_Name = async () => {
//code
}
```

```
const checkAuth = (id, pass) => {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      console.log("User authenticated");
      resolve({ id: id, pass: pass });
    }, 1000);
  });
};

const getStd = (id) => {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      console.log("Retrive student data");
      resolve({ ...id, name: "James" });
    }, 2000);
  });
};

const getResult = async () => {
  const auth = await checkAuth(2, "mypass");
  const data = await getStd(auth);
  console.log(data);
};
getResult();
```

Module

- Same as JavaScript libraries.
- There are three types of modules
  - Build-in module or Core module, for example, http, fs, path, etc.
    - Usage: const VAR_NAME = require("MODULE_NAME");
  - Custom module or Local module – created locally in your Nodejs application
    - Use module.exports or exports to expose object, function or variable as a module in Node.js.
  - Third Party Modules – the module that available online using the Node Package Manager (NPM) to install first.
    - It can be installed in the project folder or globally

RESTful API

- API (Application Program Interface) allows two applications to communicate with each other.
- RESTful API is based on representational state transfer (REST)
  - Also referred to as a RESTful web service or REST API
- An architectural style for an application program interface (API) that uses HTTP requests to access and use data.
  - That data can be used to GET (reading), PUT (updating), POST (creating) and DELETE data types.
- Data formats the REST API supports include:
  - application/json
  - application/xml

HTTP response status codes

- Status codes are issued by a server in response to a client's request.
- HTTP response status codes are separated into five categories.
  - The first digit of the status code defines the category of response.
  - *1xx informational response* – the request was received, continuing process.
  - *2xx successful* – the request was successfully received, understood, and accepted.
    - 200 OK – the request succeeded.
  - *3xx redirection* – further action needs to be taken in order to complete the request.
  - *4xx client error* – the request contains bad syntax or cannot be fulfilled.
    - 401 Unauthorized – the authorization has been refused.
    - 404 Not Found – the server can not find the requested resource.
  - *5xx server error* – the server failed to fulfil an apparently valid request.

Postman

- An API tool for testing the RESTful API.

Methods	URLs	Actions
GET	https://api.publicapis.org/categories	List all categories
GET	https://api.publicapis.org/entries/:category	List all entries currently cataloged in the project

GET https://api.publicapis.org/categories Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	Bulk Edit
Key	Value	Description	

Body Cookies Headers (9) Test Results 200 OK 313 ms 1.07 KB Save Response

Pretty Raw Preview Visualize JSON


```
{
    "count": 51,
    "categories": [
        "Animals",
        "Anime",
        "Anti-Malware",
        "Art & Design",
        "Authentication & Authorization",
        "Blockchain",
        "Books",
        "Business",
        "Calendar",
        "Cloud Storage & File Sharing",
        "Continuous Integration",
        "Cryptocurrency",
        "Currency Exchange",
```


GET https://api.publicapis.org/entries?category=Photography Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	Bulk Edit
category	Photography		
Key	Value	Description	

Body Cookies Headers (9) Test Results 200 OK 318 ms 5.2 KB Save Response

Pretty Raw Preview Visualize JSON


```
{
    "count": 28,
    "entries": [
        {
            "API": "apilayer screenshotlayer",
            "Description": "URL 2 Image",
            "Auth": "",
            "HTTPS": true,
            "Cors": "unknown",
            "Link": "https://screenshotlayer.com",
            "Category": "Photography"
        },
        {
            "API": "APITemplate.io",
            "Description": "Dynamically generate images and PDFs from templates with a simple API",
```

Homework 2

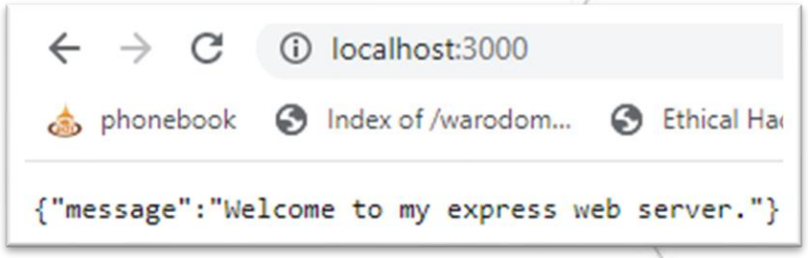
- Find out the free API and write the usage instruction in A4 format with the following topics:
  - What is the purpose of API?
  - How to call API?
  - Describe the required parameters.
  - Describe the optional parameters.
  - Show an example of API calls though Postman including the description.
- Submit the PDF file at https://bit.ly/465F3tV
- Due Date: Sep 4

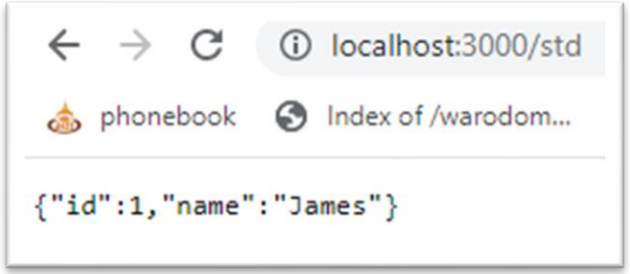
Express

- Express is Nodejs framework that designed for building web applications and APIs by using JavaScript.
- Installation – $ npm install express

```
const express = require("express");
const app = express();
app.get("/", (req, res) => {
  res.send({ message: "Welcome to my express web server." });
});
app.get("/std", (req, res) => {
  res.send({ id: 1, name: "James" });
});

const PORT = process.env.PORT || 3000;
app.listen(PORT, () => {
  console.log(`Server is running on port ${PORT}.`);
});
```



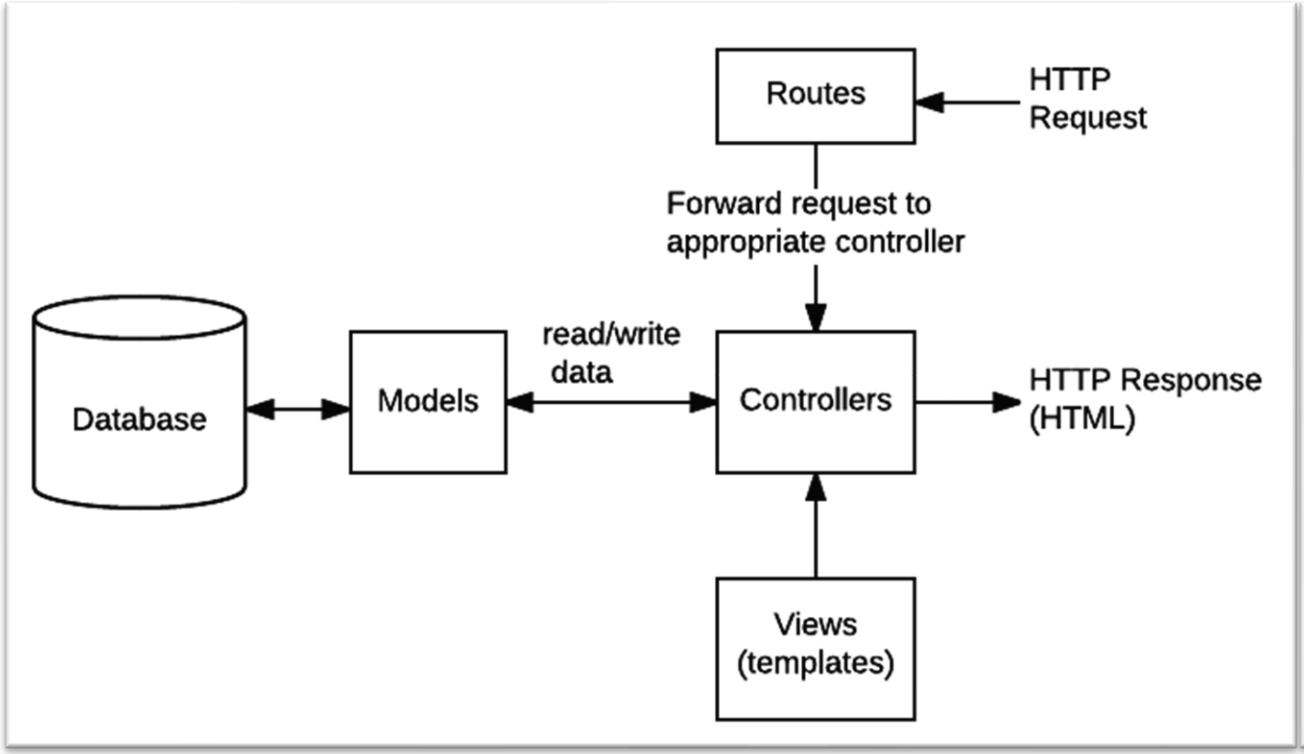


nodemon

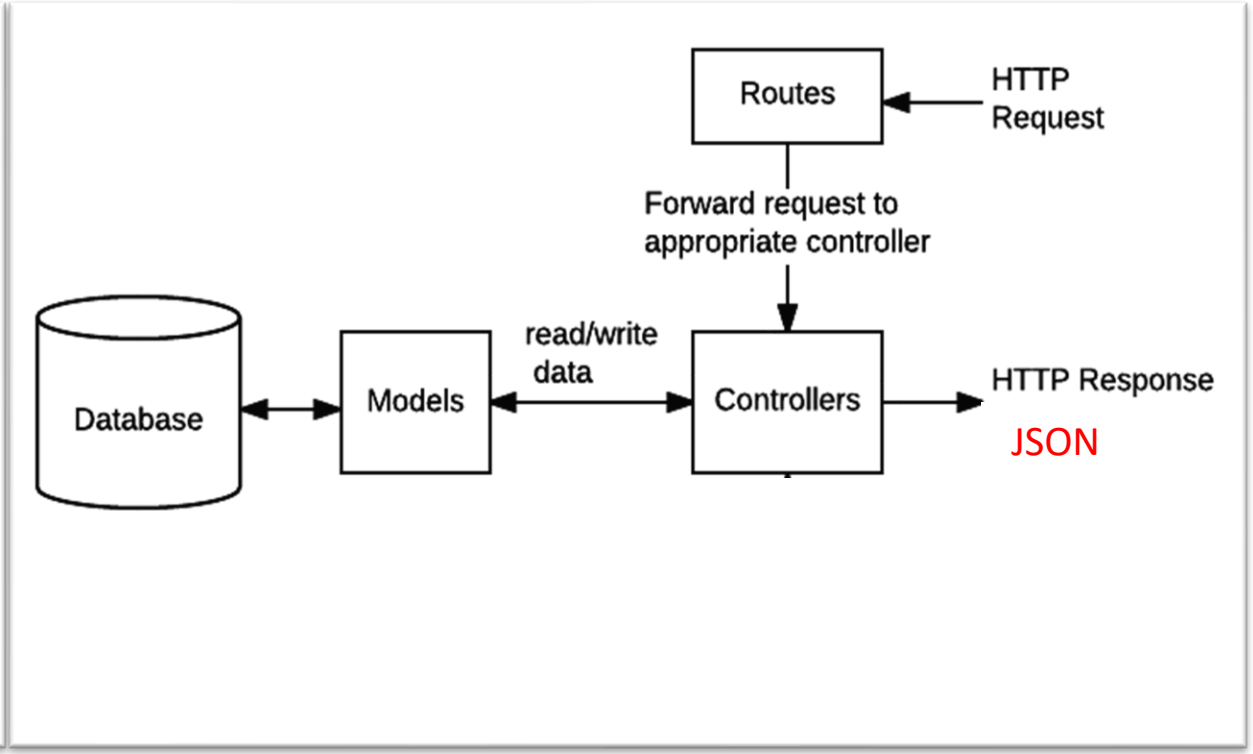
- Any change to the express file will not appear until restart the server (run file again).
- nodemon is a tool to solve this problem
- Installation – recommended dev dependency
  - Installed as globally – $ npm install –g nodemon
  - Installed as a development dependency – $ npm install –save-dev nodemon
- Usage
  - $ npx nodemon Nodejs_FILE.js
  - Update the scripts section in package.json file
    - Add “dev”: “nodemon Nodejs_FILE.js”
    - Run $ npm run dev

MVC

- MVC stands for Model, View, Controller.
- It is an architectural pattern.
- Three main logical components:
  - Model is the database interface that lets you interact with the database.
  - View used by the controllers to render the data into plain HTML.
    - It needs a Template Engine for the rendering process.
    - EJS is the famous Template Engine that works with Express.
  - Controller handles the HTTP Request and returns a response.



MVC

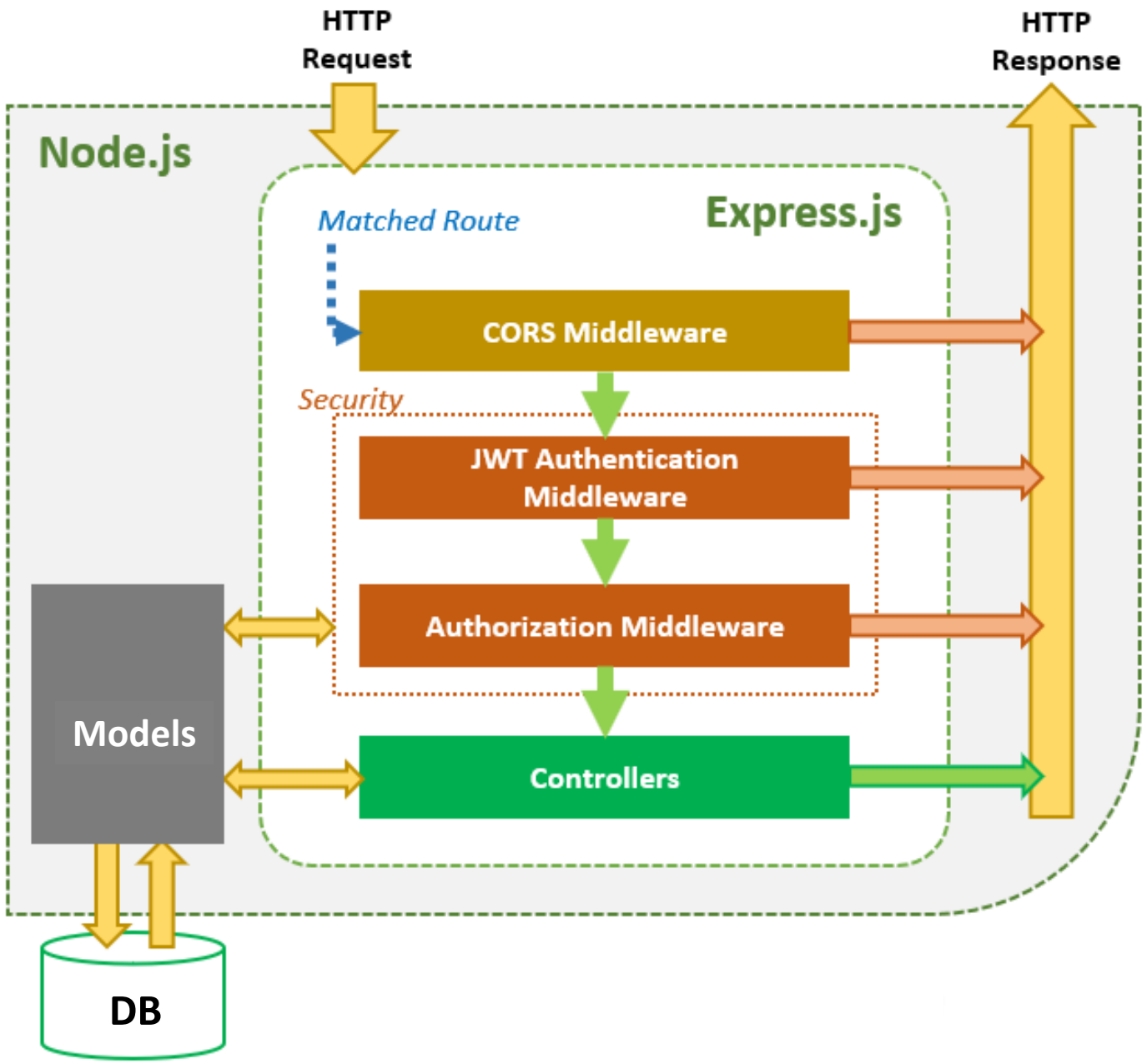


RESTful API

Ref. https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/routes

Middleware

- Middleware is a type of computer software that provides services to software applications.



RESTful API (Backend App) for Manage Accounts App

Functions Listing

- Check the exist username
- Upload an image
- Create a new account
- Login
- List all users
- Modify the specific user
- Remove the specific user

Production version

- Application
  - CoC (PSU Network): http://172.26.117.16:5000
  - Public: https://manage-account-frontend.vercel.app
- API
  - CoC (PSU Network): http://172.26.117.16:3000
  - Public: https://manage-account-api.vercel.app
    - Problem issue in the upload file due to Vercel is serverless.

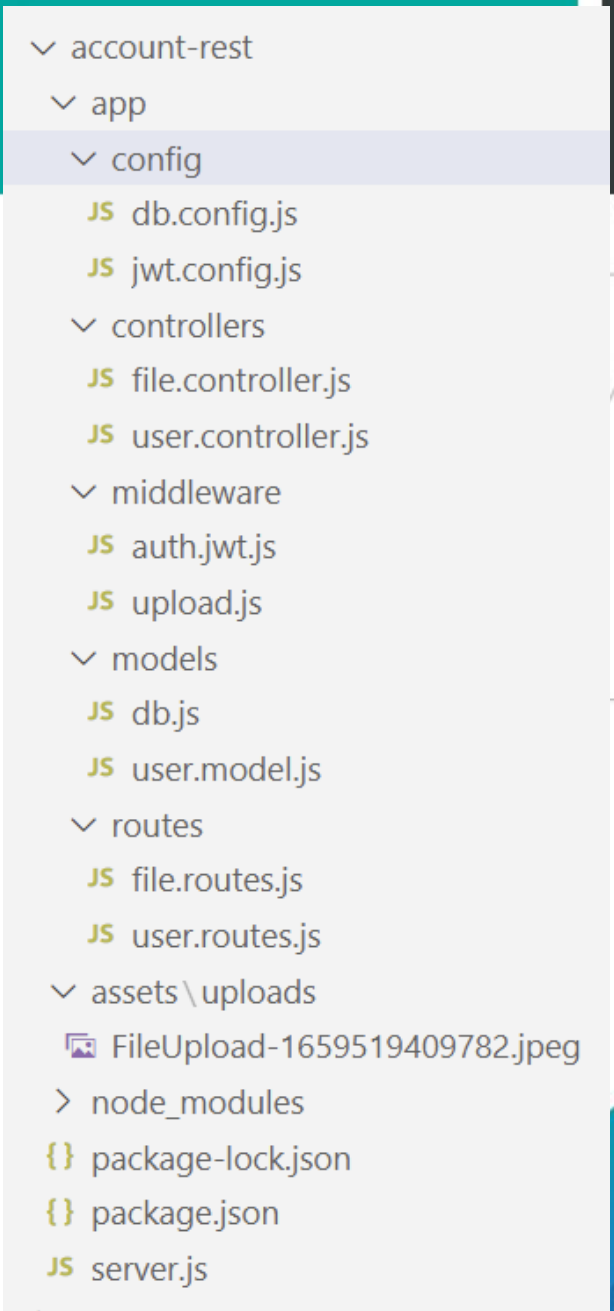
Create a node project

- $ npm init OR Create *package.json* file
- Download DB https://bit.ly/3BuNx0x

```
package name: (account-rest)
version: (1.0.0)
description: A RESTful API for authentication and accounts management
entry point: (index.js)
test command:
git repository:
keywords:
author: Amonrat P.
license: (ISC)
About to write to D:\Gdrive\COC\Sem1-2565\977-110 WEB\Source\ch04\account-rest\package.json:

{
  "name": "account-rest",
  "version": "1.0.0",
  "description": "A RESTful API for authentication and accounts management",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Amonrat P.",
  "license": "ISC"
}


Is this OK? (yes) yes
```



Install the necessary modules


```
$ npm install express mysql2 body-parser cors
jsonwebtoken bcryptjs multer nodemon
```

- CORS (Cross-origin resource sharing) is an HTTP-header based mechanism that allows a server to indicate any origins (domain, scheme, or port) other than its own permit loading resources.
  - Restrict access only specific URL
    - Access-Control-Allow-Origin: https://foo.example
  - Allow any origin
    - Access-Control-Allow-Origin: *
- JSON Web Token (JWT) is securely transmitting information between parties as a JSON object.
  - Using a public/private key pair to sign a token

Install the necessary modules (Cont.)

- bcrypt.js is used to encrypt a plain text password with a hash algorithm.
  - It is one-way encryption and cannot decrypt the cipher to plain text.
  - Salt is random data that is used as an additional input to a one-way function
  - To hash a password

```
const saltRounds = 10;
const salt = bcrypt.genSaltSync(saltRounds);
const hash = bcrypt.hashSync(myPlaintextPassword, salt);
```

- Modify *package.json*

```
"scripts": {
  "test": "echo \"Error: no test specified\" && exit 1",
  "dev": "nodemon server.js",
  "start": "node server.js"
},
```

```
"dependencies": {
  "bcryptjs": "^2.4.3",
  "body-parser": "^1.20.0",
  "cors": "^2.8.5",
  "express": "^4.18.1",
  "jsonwebtoken": "^8.5.1",
  "multer": "^1.4.5-lts.1",
  "mysql2": "^2.3.3",
  "nodemon": "^2.0.19"
}
```

Setup Express web service

- Create *server.js* at the root folder
  - For execute the REST API
- Create *app* folder at the root folder
  - Keep all application files

```
const express = require("express");
const bodyParser = require("body-parser");
const cors = require("cors");
const app = express();

global.__basedir = __dirname;
var corsOptions = {
  origin: "*"
};
app.use(cors(corsOptions));
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));
app.get("/", (req, res) => {
  res.json({ message: "Welcome to my REST." });
});

const PORT = process.env.PORT || 3000;
app.listen(PORT, () => {
  console.log(`Server is running on port ${PORT}.`);
});
```

Config DB & JWT

- Create *config* folder under *app* folder to store the configuration files.
- Create *db.config.js* under *config* folder for the DB configuration.

```
module.exports = {
    HOST: "localhost",
    USER: "root",
    PASSWORD: "",
    DB: "testdbapi"
  };
```

- Create *jwt.config.js* under *config* folder for keep the secret word.

```
module.exports = { secret: "my-secret-key" };
```

Define DB connection

- Create *model* folder under *app* folder to store the model files.
- Create *db.js* under *models* folder for DB connection.

```
const mysql = require("mysql2");
const dbConfig = require("../config/db.config.js");
// Create a connection to the database
const connection = mysql.createConnection({
  host: dbConfig.HOST,
  user: dbConfig.USER,
  password: dbConfig.PASSWORD,
  database: dbConfig.DB
});
// open the MySQL connection
connection.connect(error => {
  if (error) console.log("MySQL connection: " + error);
  else console.log("Successfully connected to the database.");
});
module.exports = connection;
```

Define Middleware

- Create *middleware* folder under *app* folder.
- Create *auth.jwt.js* under *middleware* folder for verify token.

```
const jwt = require("jsonwebtoken");
const scKey = require("../config/jwt.config");
const verifyToken = (req, res, next) => {
  const token = req.headers["x-access-token"];
  if (!token) {
    return res.status(403).send({ message: "No token provided." });
  }
  jwt.verify(token, scKey.secret, (err, decoded) => {
    if (err) {
      return res.status(401).send({ message: "Unauthorized." });
    }
    req.id = decoded.id;
    next();
  });
};
module.exports = verifyToken;
```

Define Middleware (Cont.)

- Create *upload.js* under *middleware* folder for upload file with *Multer*.
  - util.promisify() makes the exported middleware object can be used with async-await
- Create *assets* folder and *uploads* folder at the root for keep the upload files.

```
const util = require("util");
const multer = require("multer");
const storage = multer.diskStorage({
  destination: (req, file, cb) => {
    cb(null, __basedir + "/assets/uploads/");
  },
  filename: (req, file, cb) => {
    const extArray = file.mimetype.split("/");
    const extension = extArray[extArray.length - 1];
    const newFileName = `FileUpload-${Date.now()}.${extension}`;
    cb(null, newFileName);
  },
});
const uploadFile = multer({ storage: storage }).single("singlefile");
const uploadFileMiddleware = util.promisify(uploadFile);
module.exports = uploadFileMiddleware;
```

Define Routes

- Create *routes* folder under *app* folder.
- Create *file.routes.js* under *routes* folder.
- Create *user.routes.js* under *routes* folder.
- Add routes to *server.js*

```
// server.js
require("./app/routes/file.routes")(app);
require("./app/routes/user.routes")(app);
```

```
// file.routes.js
module.exports = app => {
    const file_controller = require("../controllers/file.controller");
    var router = require("express").Router();
    router.post("/upload", file_controller.upload);
    app.use("/api/file", router);
};
```

```
// user.routes.js
const authJwt = require("../middleware/authJwt");
module.exports = (app) => {
  const user_controller = require("../controllers/user.controller");
  var router = require("express").Router();
  router.post("/signup", user_controller.createNew);
  router.get("/:us", user_controller.validUsername);
  router.post("/login", user_controller.login);
  router.get("/", authJwt, user_controller.getAllUsers);
  router.put("/:id", authJwt, user_controller.updateUser);
  router.delete("/:id", authJwt, user_controller.deleteUser);
  app.use("/api/auth", router);
};
```

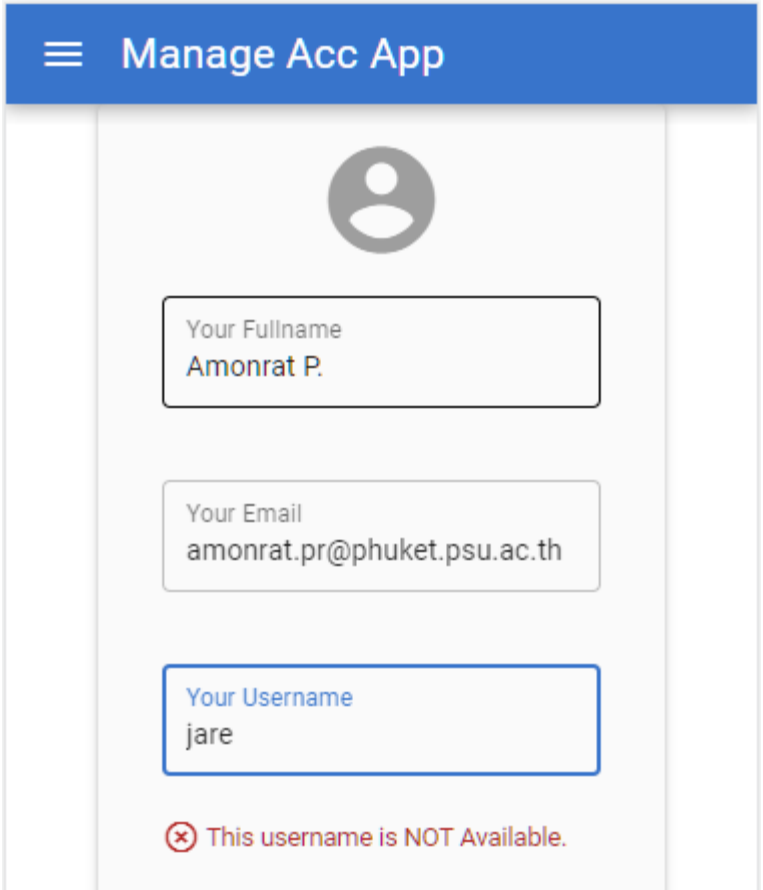
Define the model & controller

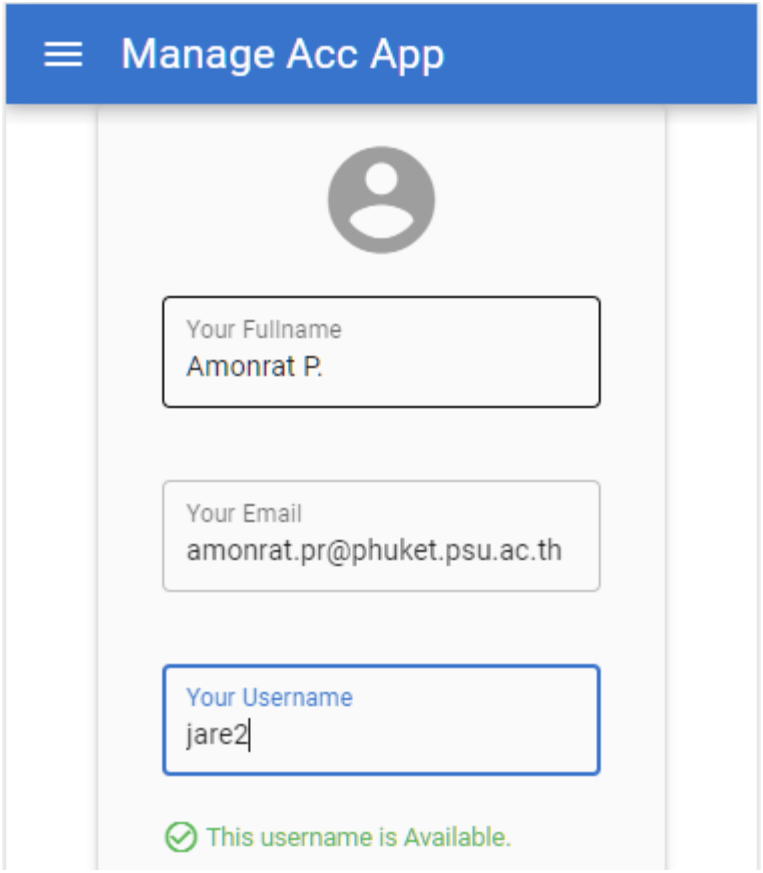
- Create *user.model.js* under *models* folder.
- Create *controllers* folder under *app* folder.
- Create *file.controller.js* under *controllers* folder
- Create *user.controller.js* under *controllers* folder

```
// file.controller.js
const uploadFile = require("../middleware/upload");
const upload = async (req, res) => {
  try {
    await uploadFile(req, res);
    if (req.file === undefined) {
      return res.status(400).send({ message: "Please upload a file!" });
    }
    res.status(200).send({
      message: "Uploaded the file successfully: " + req.file.filename,
      uploadFileName: req.file.filename,
    });
  } catch (err) {
    res.status(500).send({
      message: "Could not upload the file:" + err,
    });
  }
};
module.exports = { upload };
```

Check Exist Username

Method	URLs	Description
GET	*DOMAIN*/api/auth/:us	Check the exist username





Test Check Exist Username By Postman

GET http://172.26.117.16:3000/api/auth/v22

Params Authorization Headers (7) Body Pre-req

Query Params

KEY	
Key	

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON

```
{
    "message": "Not found v22",
    "valid": true
}
```

GET http://172.26.117.16:3000/api/auth/v23

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE
Key	Value

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON

```
{
    "record": {
        "id": 31,
        "fullname": "Test Ver23",
        "email": "v23@q.qqq",
        "username": "v23",
        "password": "$2a$10$bg2jYOjKF9mMi0TlIr0PpuzpkaEuS/0l/zNJTYzNd9tgcJcBAE2j2",
        "img": "FileUpload-1658085529952.png"
    },
    "valid": false
}
```

Upload an image

Method	URLs	Description
POST	*DOMAIN*/api/file/upload	Upload a single file to the server Required body: • singlefile

Test Upload an image By Postman

POST http://172.26.117.16:3000/api/file/upload Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
☑	singlefile	L7eNyA2c_400x400.jpg			
	Key	Value	Description		

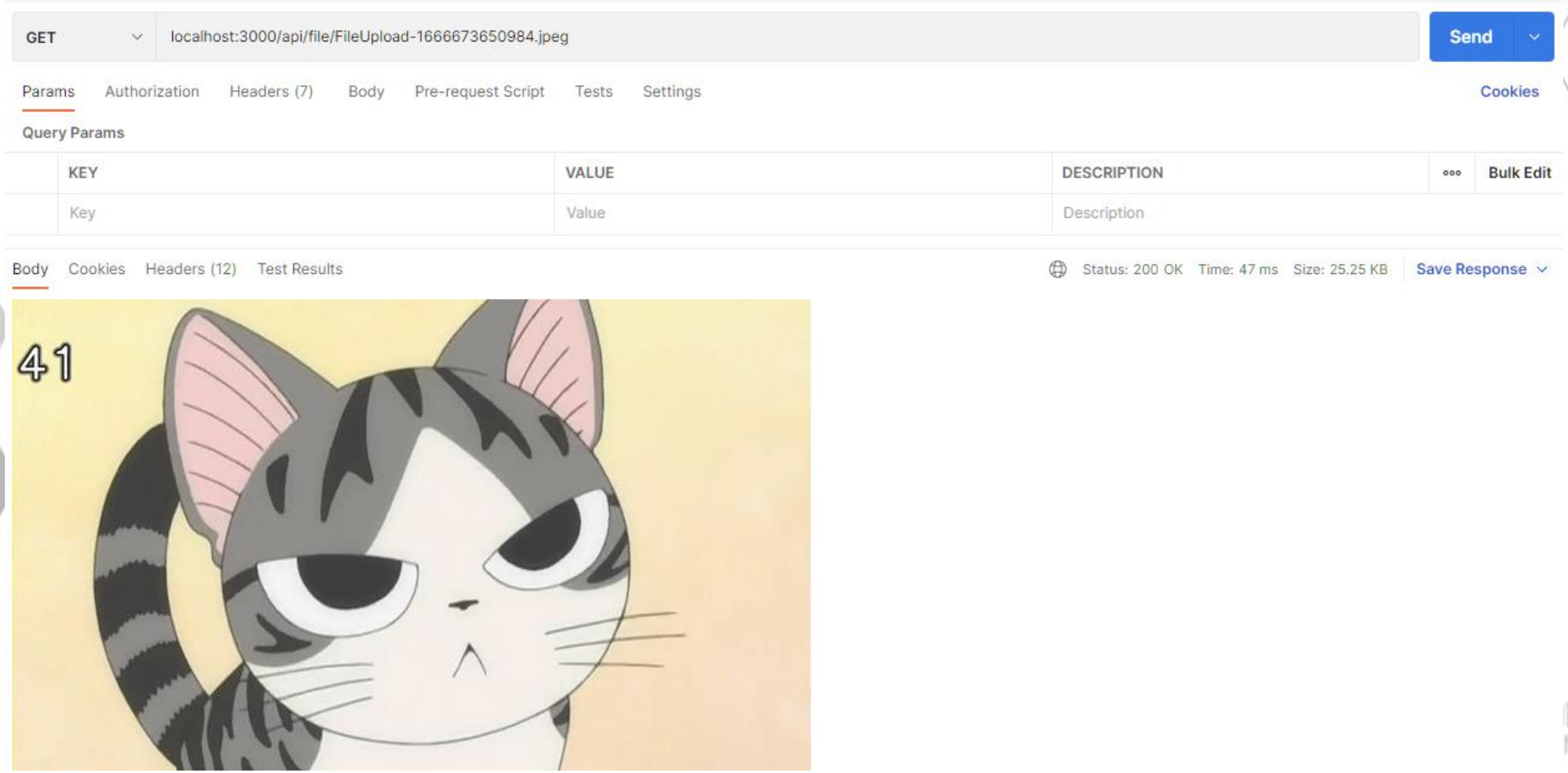
Body Cookies Headers (8) Test Results Status: 200 OK Time: 32 ms Size: 392 B Save Response

Pretty Raw Preview Visualize JSON

```
{
    "message": "Uploaded the file successfully: FileUpload-1659254660695.jpeg",
    "uploadFileName": "FileUpload-1659254660695.jpeg"
}
```

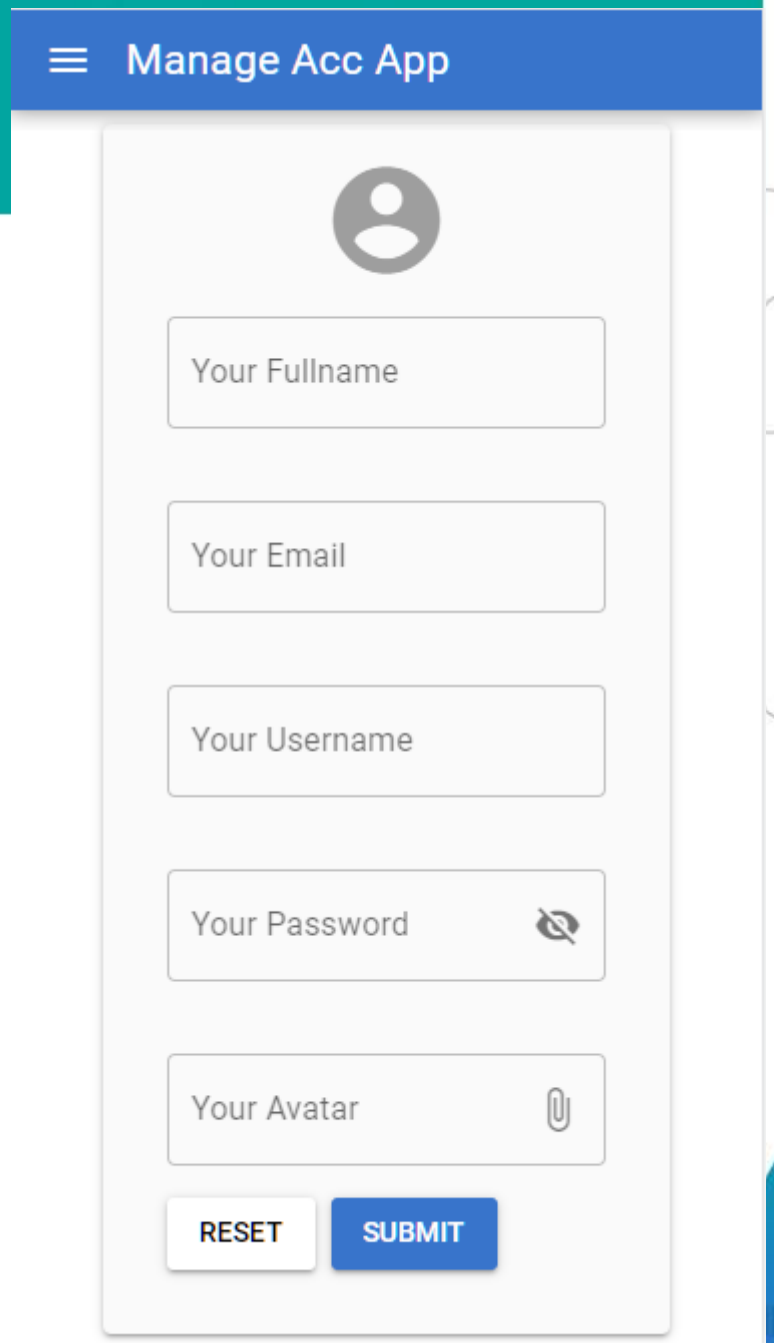
Download an image

Method	URLs	Description
GET	*DOMAIN*/api/file/:name	Download a specific file



Create New Account

Method	URLs	Description
POST	*DOMAIN*/api/auth/signup	Add a new user Required body: • fullname • email • username • password • img <optional>



Test Create New Account By Postman

POST http://172.26.117.16:3000/api/auth/signup Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
{
    "fullname": "Amonrat Prasitsupparote",
    "email": "amonrat.pr@phuket.psu.ac.th",
    "username": "myus",
    "password": "myps"
}
```

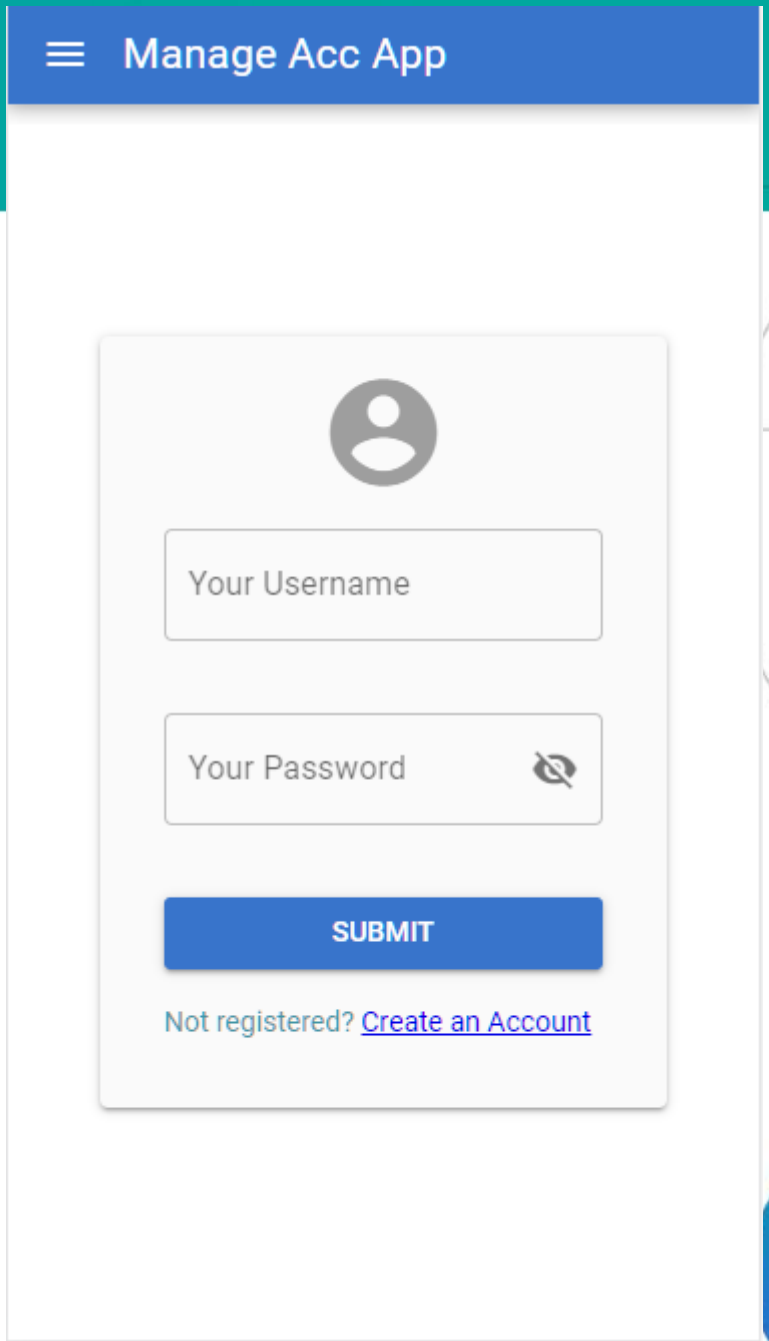
Body Cookies Headers (8) Test Results Status: 200 OK Time: 187 ms Size: 601 B Save Response

Pretty Raw Preview Visualize JSON

```
{
    "id": 40,
    "fullname": "Amonrat Prasitsupparote",
    "email": "amonrat.pr@phuket.psu.ac.th",
    "username": "myus",
    "password": "$2a$10$rDzpkI8wltfbxZvOuOeQweCUtWd95yw4gMiNHpmtVPcvtJjeimJB2",
    "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6NDAsImlhdCI6MTY1OTI1MjY2MCwiZXhwIjoxNjU5MjU5ODYwfQ.
        D9DFWArq-vFn7XG7MCBFjL5LrxmTMShlpuWuEj1MK58"
}
```

Login

Method	URLs	Description
POST	*DOMAIN*/api/auth/login	User authentication Required body: • username • password



Test Login By Postman

POST http://172.26.117.16:3000/api/auth/login Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
1 {
2     "username": "myus",
3     "password": "myps"
4 }
```

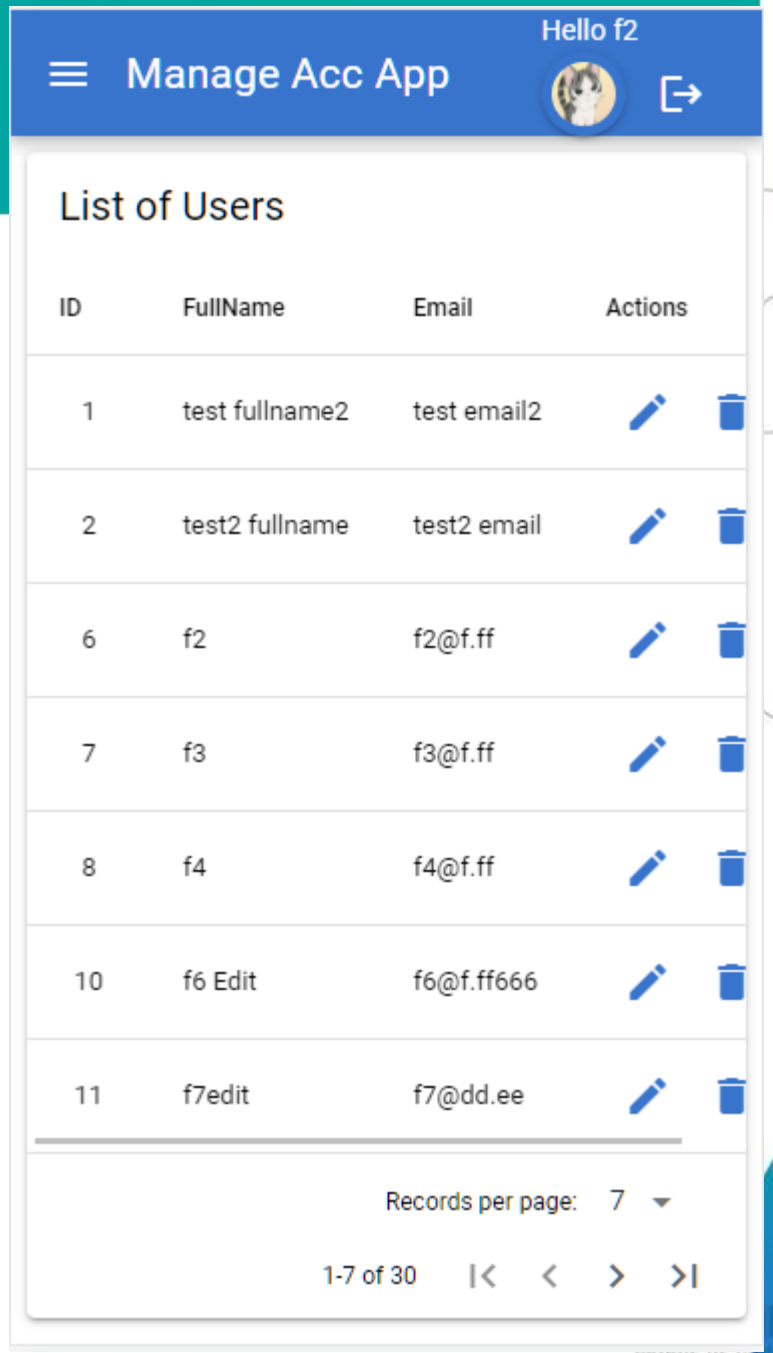
Body Cookies Headers (8) Test Results Status: 200 OK Time: 155 ms Size: 601 B Save Response

Pretty Raw Preview Visualize JSON


```
1 {
2     "id": 40,
3     "fullname": "Amonrat Prasitsupparote",
4     "email": "amonrat.pr@phuket.psu.ac.th",
5     "username": "myus",
6     "password": "$2a$10$rDzpkI8wltfbxZvOuOeQweCUtWd95yw4gMiNHpmtVPcvtJjeimJB2",
7     "img": null,
8     "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpYXQiOjE2NTkyNTI4NDksImV4cCI6MTY1OTI2MDA0OX0.Hv5QFqrPy8ohFEW8vDPoAvcibDnp9YskzmmVAL0cKbQ"
9 }
```

List All Users

Method	URLs	Description
GET	*DOMAIN*/api/auth	List all users Required JWT • Use `"x-access-token" in header`



Test List All Users By Postman

GET http://172.26.117.16:3000/api/auth Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Headers 7 hidden

	KEY	VALUE	DESCRIPTION
☑	x-access-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpYXQiOjE2NTkyNTI...	
	Key	Value	Description

Bulk Edit Presets

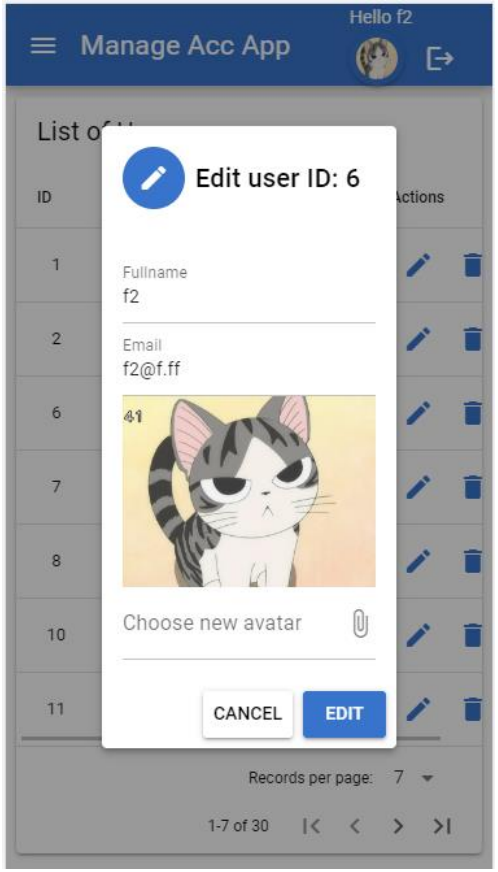
Body Cookies Headers (8) Test Results Status: 200 OK Time: 18 ms Size: 5.29 KB Save Response

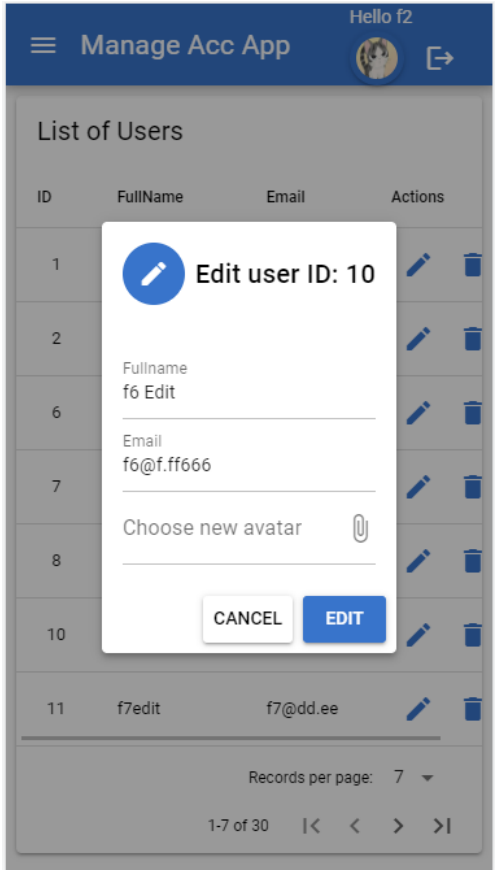
Pretty Raw Preview Visualize JSON


```
[
    {
        "id": 1,
        "fullname": "test fullname2",
        "email": "test email2",
        "username": "test username",
        "password": "$2a$08$T0B0i/96KE90jAYPOhpsN.vJGVPMfFw.FbxljzuQkkN4ZK3YauRLq",
        "img": "FileUpload-1658154816561.png"
    },
    {
        "id": 2,
        "fullname": "test2 fullname",
        "email": "test2 email",
        "username": "test2 username",
```

Modify Specific User

Method	URLs	Description
PUT	*DOMAIN*/api/auth/:id	Update the specific user by id Required JWT • Use `"x-access-token" in header`





Test Modify Specific User By Postman

PUT http://172.26.117.16:3000/api/auth/40

Params Authorization Headers (10) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
{
    "fullname": "Amonrat P.",
    "email": "amonrat.pr@coc.psu.ac.th",
    "img": null
}
```

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON

```
{
    "message": "No token provided."
}
```

PUT http://172.26.117.16:3000/api/auth/40

Params Authorization Headers (10) Body Pre-request Script Tests Settings

Headers 9 hidden

KEY	VALUE
x-access-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpY
Key	Value

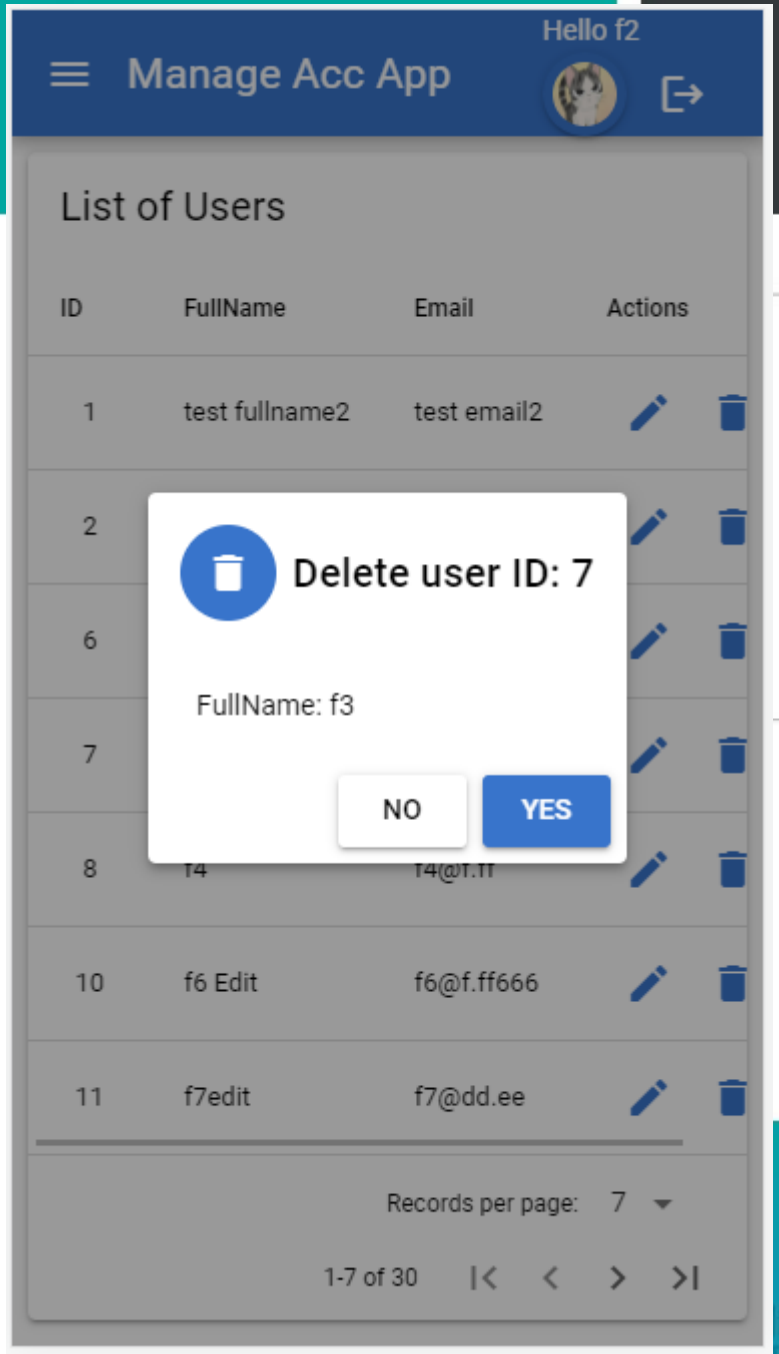
Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON

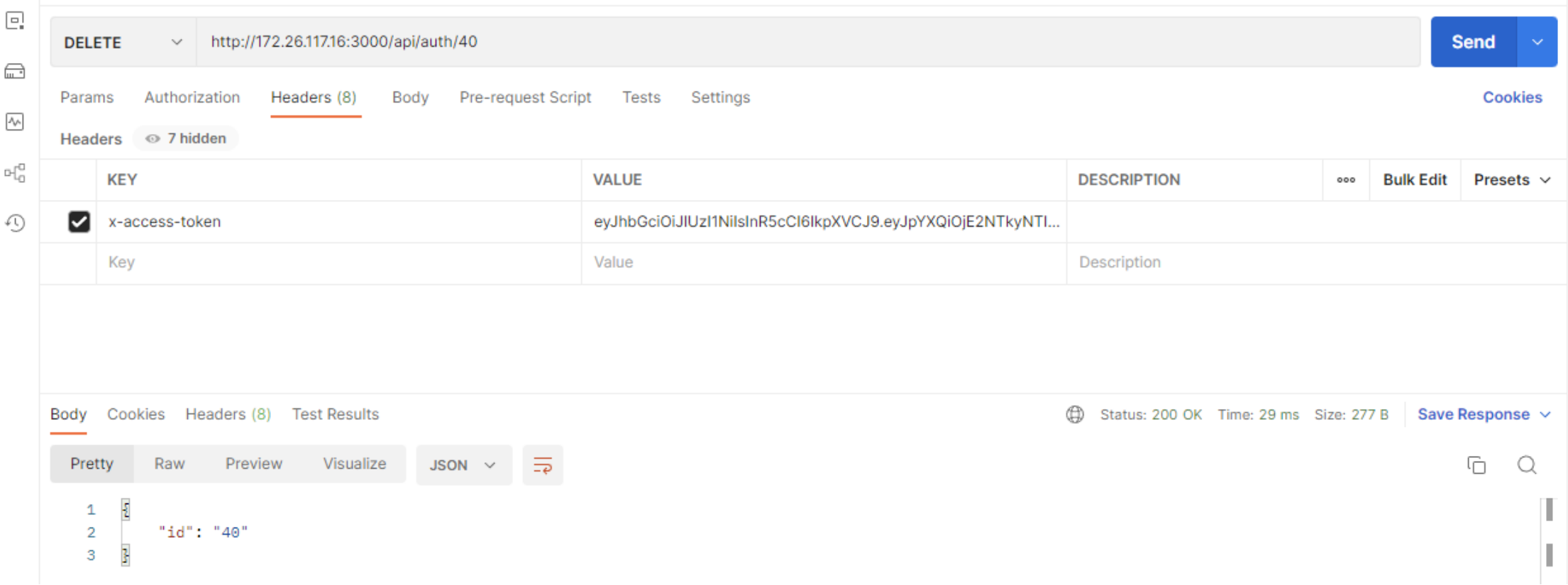
```
{
    "id": "40",
    "fullname": "Amonrat P.",
    "email": "amonrat.pr@coc.psu.ac.th"
}
```

Remove Specific User

Method	URLs	Description
DELETE	*DOMAIN*/api/auth/:id	Remove the specific user by id Required JWT • Use `"x-access-token" in header`



Test Remove Specific User By Postman



Deploy to Public Server

Deploy to Vercel

- Vercel is a platform for frontend developers
  - Install Vercel-cli
  - Create Vercel account

```
$ npm i -g vercel
```

- Git is a distributed version control system
  - git-scm at https://git-scm.com/download
  - Create Github account

Vercel Configuration

- Create *vercel.json* under the root directory of your project.
  - version >= 2
  - name [optional] – string name for the deployment
    - A maximum length of 52 characters
    - Only lower case alphanumeric characters or hyphens are allowed
    - Cannot begin or end with a hyphen, or contain multiple consecutive hyphens
- See more detail at https://vercel.com/docs/concepts/projects/project-configuration#project/

```
{
  "version": 2,
  "name": “ANY_STRING",
  "builds": [{ "src": "server.js", "use": "@vercel/node" }],
  "routes": [{ "src": "/(.*)", "dest": "/server.js" }]
}
```

Free Database Server

- FREEDB.TECH (https://freedb.tech)
  - 1 MySQL Database with 50 MB Storage
  - Limited Queries
  - Need login before the usage, waiting for update service > 10 mins
- PlanetScale (https://planetscale.com)
  - NO phpMyAdmin
  - Required SSL connection
  - Connection String > create password
  - Connect with: General
  - 1 Database

MySQL Workbench

Setup New Connection

Connection Name:

Connection Method: Standard (TCP/IP)

Parameters | SSL | Advanced

Hostname: aws.connect.psdb.cloud

Port: 3306

Name or IP adc
TCP/IP port.

Username:

Name of the us

Setup New Connection

Connection Name: ManageAccount

Connection Method: Standard (TCP/IP)

Parameters | SSL | Advanced

Use SSL: Require

Turns on SSL e
is not available

Test MySQL connection from localhost

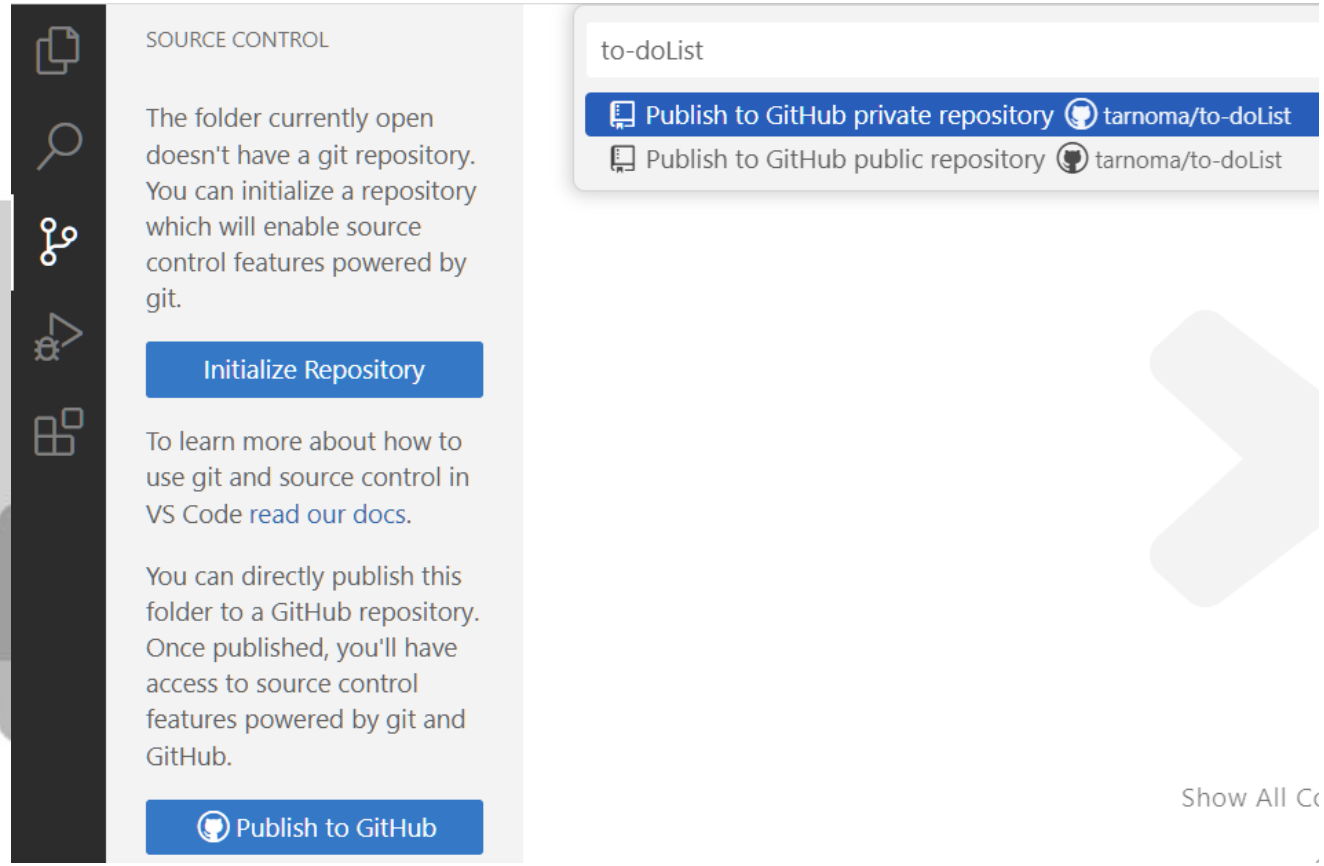
- PlanetScale
  - Connect with: Node.js
  - Install module dotenv
  - Create a file at the root directory, named ".env", for localhost only.
  - Create file ".gitignore"
  - Modify *models/db.js*

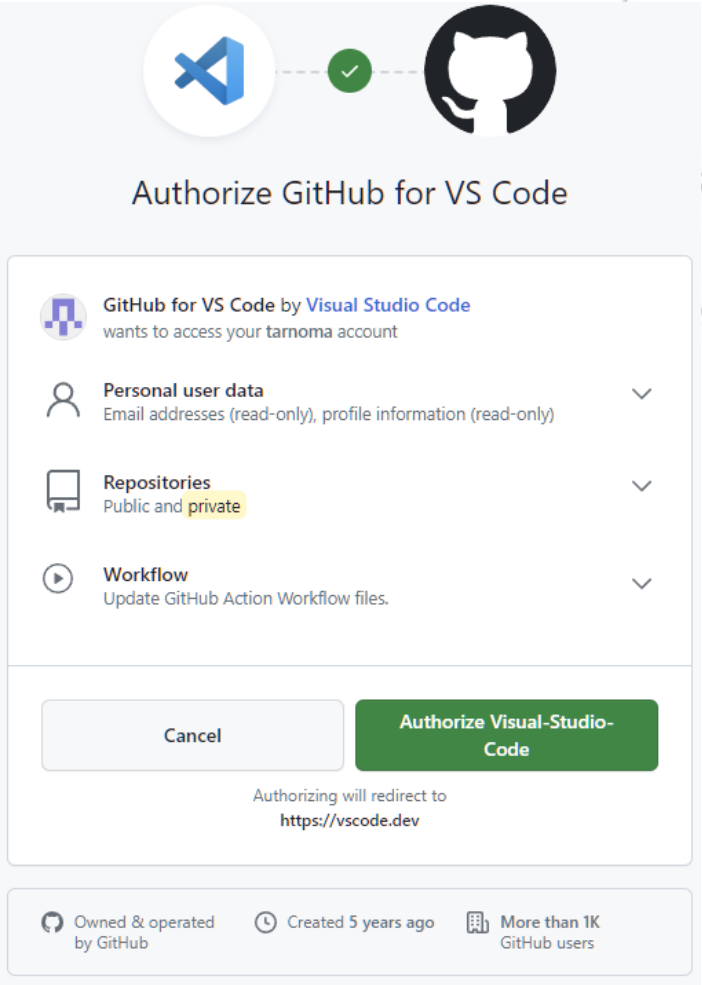
```
$ npm install dotenv
```

```
// db.js
const mysql = require("mysql2");
require("dotenv").config()
const connection =
mysql.createConnection(process.env.DATABASE_URL)
```

Upload files to Github

- Create *.gitignore* file at root directory

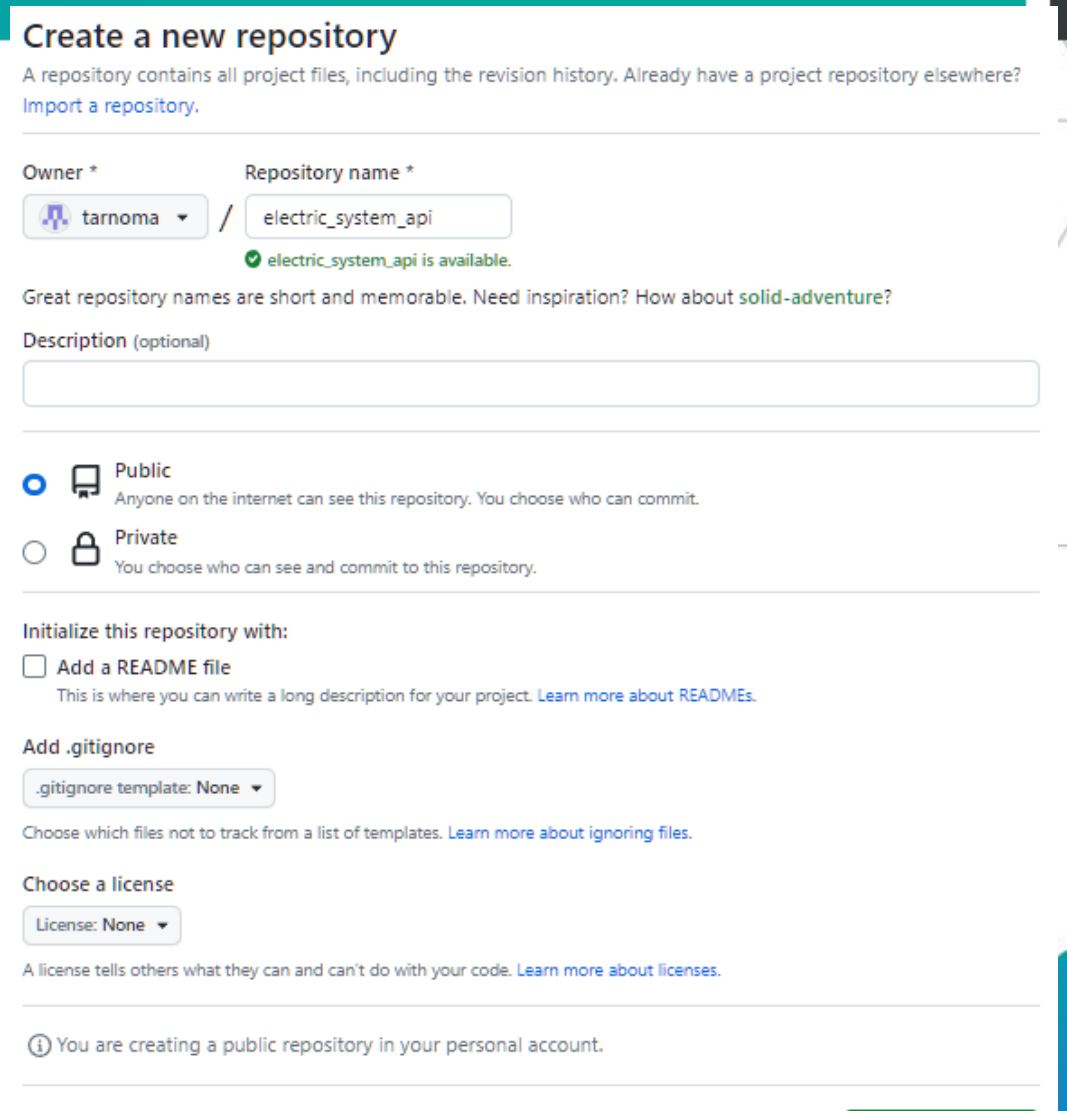




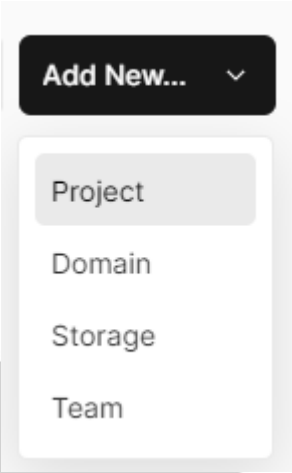
Upload files to Github (manually)

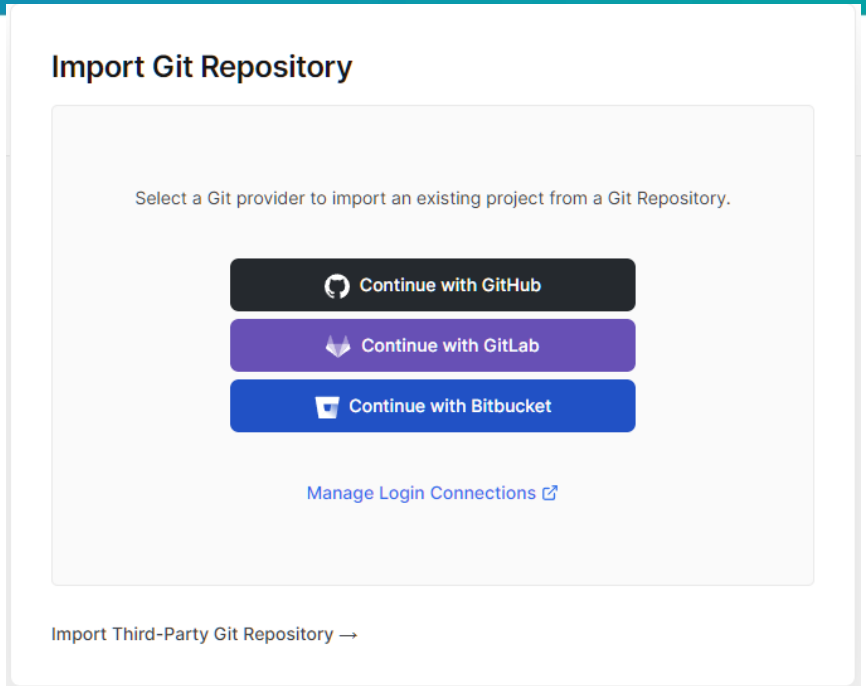
- Create a new repository

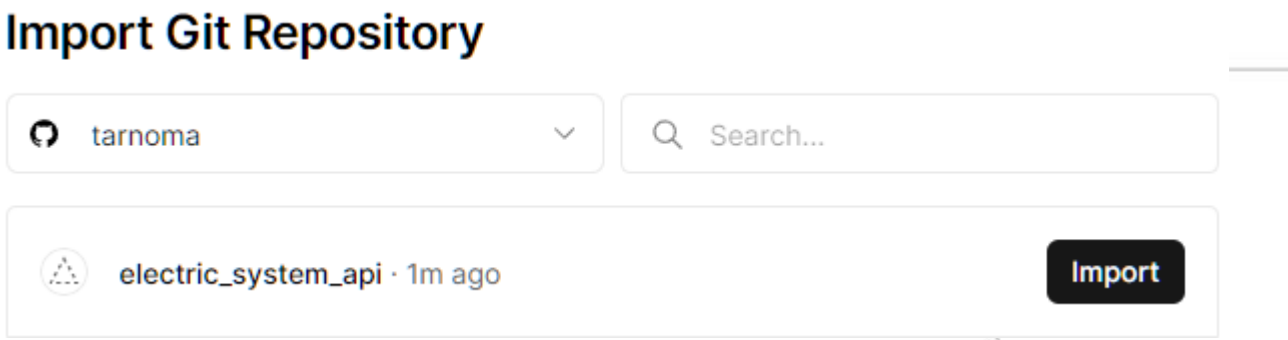
```
$ git init
$ git add .
$ git commit -m
  "YOUR_COMMIT_WORD"
$ git branch -M main
$ git remote add origin
  "GIT_REPOSITORY"
$ git push -u origin main
```

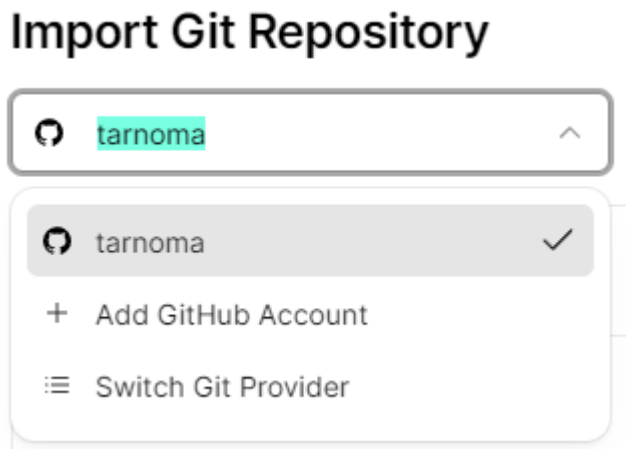


Create a new project on Vercel



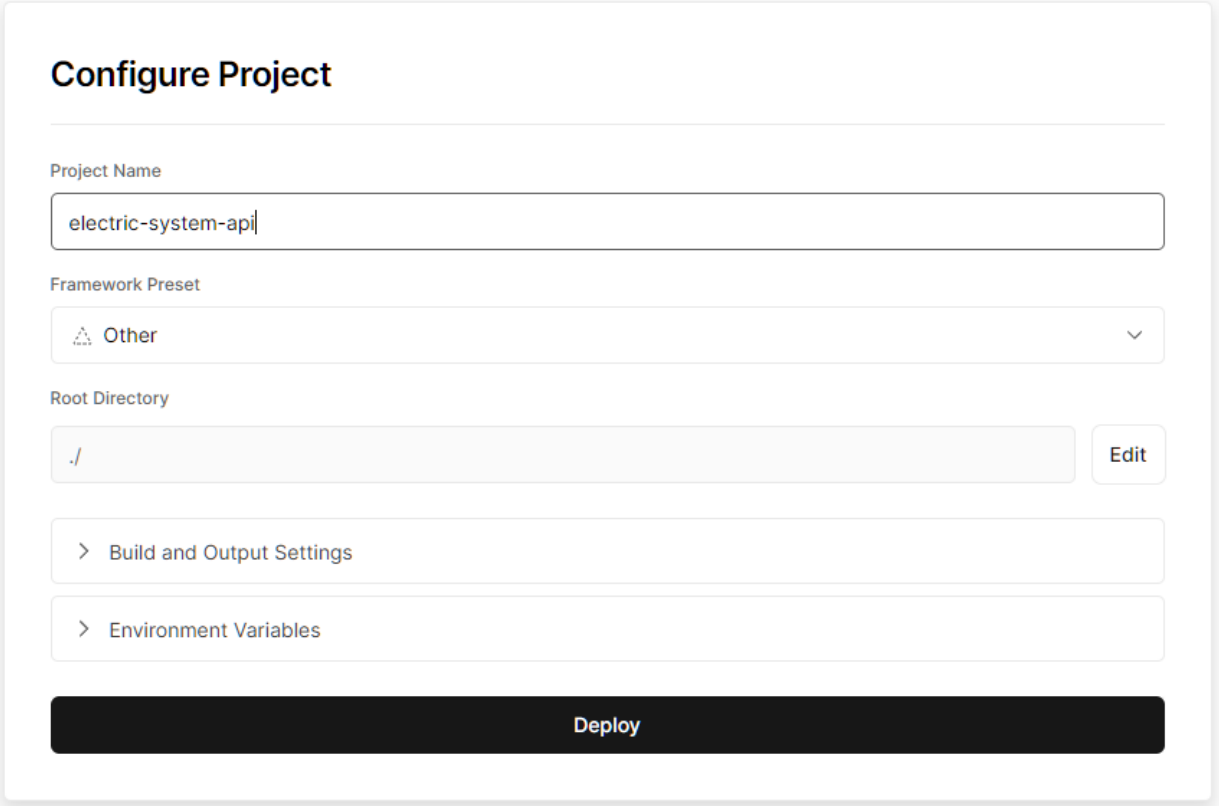


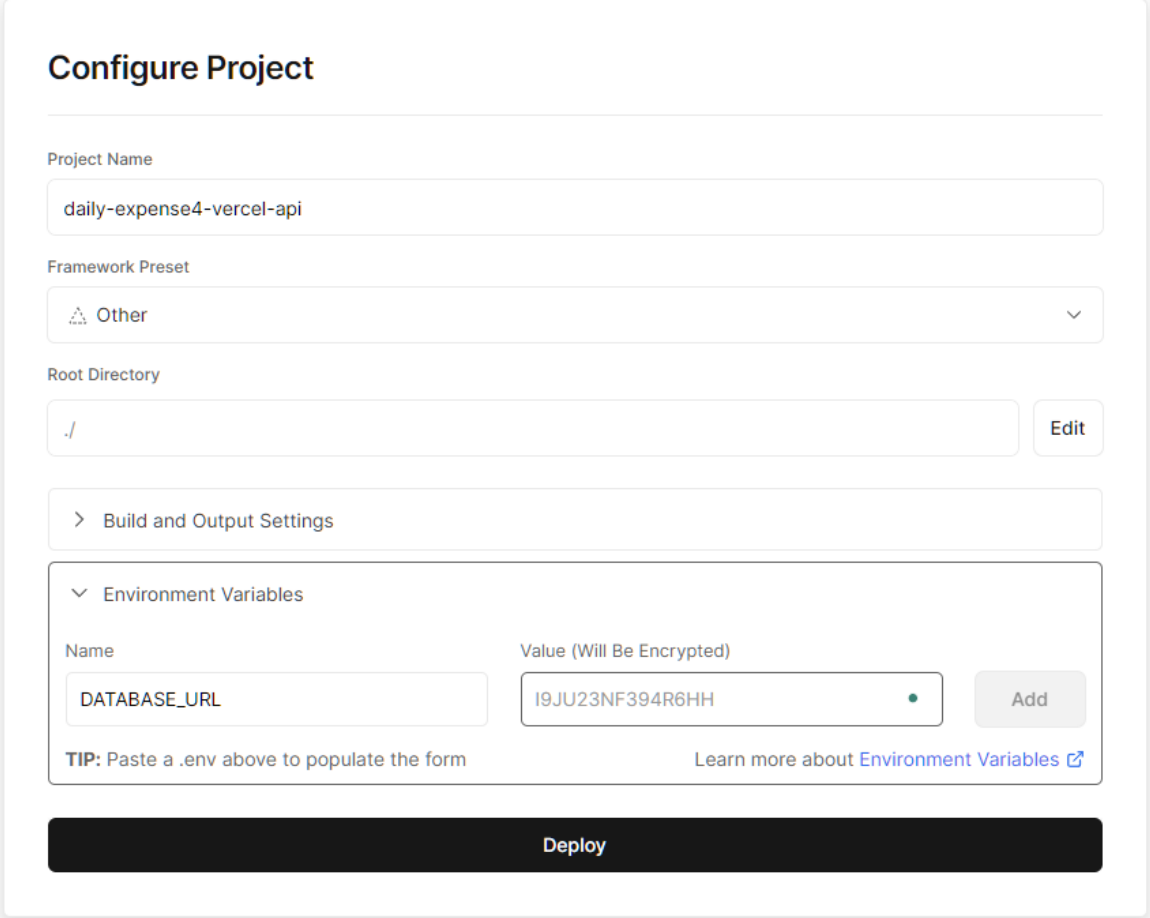




Create a new project on Vercel (Cont.)

- Environment variables
  - Only value, without single quote.





Config Environment Variable on Vercel

Environment Variables

In order to provide your Deployment with Environment Variables at Build and Runtime, you may enter them right here, for the Environment of your choice. Learn more

A new Deployment is required for your changes to take effect.

Key	Value
DATABASE_URL	xxxxx

Add another

Environment

- [x] Production
- [x] Preview Select custom branch
- [x] Development

Import TIP: Paste a .env above to populate the form

Save

Free Hosting

- cyclic.sh
- digitalocean.com
- railway.app
- render.com
- adaptable.io
- fly.io

Homework 3

- Deploy your Manage Accounts API to any hosting that can be accessed anywhere.
- After that fill your URL at https://bit.ly/46fTBHx
- Due Date: XXXX

References

- "Vue.js - The Progressive JavaScript Framework | Vue.js." https://vuejs.org/guide/introduction.html (accessed Jul. 19, 2022).
- "Quasar Framework - Build high-performance VueJS user interfaces in record time," *Quasar Framework*. https://quasar.dev/ (accessed Jul. 19, 2022).
- Node.js, "Documentation," *Node.js*. https://nodejs.org/en/docs/ (accessed Jul. 19, 2022).
- "HTML Tutorial." https://www.w3schools.com/html/default.asp (accessed Jul. 19, 2022).
- "HTML: HyperText Markup Language | MDN." https://developer.mozilla.org/en-US/docs/Web/HTML (accessed Jul. 19, 2022).
- "CSS Tutorial." https://www.w3schools.com/css/default.asp (accessed Jul. 19, 2022).