Open-Source Report

Proof of knowing your stuff in CSE312

Guidelines

Provided below is a template you must use to write your reports for your project.

Here are some things to note when working on your report, specifically about the **General Information & Licensing** section for each technology.

- **Code Repository**: Please link the code and not the documentation. If you'd like to refer to the documentation in the **Magic** section, you're more than welcome to, but we need to see the code you're referring to as well.
- **License Type**: Three letter acronym is fine.
- **License Description**: No need for the entire license here, just what separates it from the rest.
- **License Restrictions**: What can you *not* do as a result of using this technology in your project? Some licenses prevent you from using the project for commercial use, for example.

Also, feel free to extend the cell of any section if you feel you need more room.

If there's anything we can clarify, please don't hesitate to reach out! You can reach us using the methods outlined on the course website or see us during our office hours.

[Express.js]

General Information & Licensing

Code Repository	https://github.com/expressjs/express
License Type	MIT
License Description	This type of license allows users to reuse code for any purpose, even if it is part of proprietary software. Users are free to modify the code as long as they include the original copy of the MIT license in their distributions. The MIT License allows user to use the software for : ● Commercial use ● Modify ● Distribute ● Sublicense ● Private User The user of the Software with the MIT License must include: ● Copyright ● License

License Restrictions	Software that contains an MIT license restricts its user to hold the author of the software liable. This means any mistakes can not hold the author of the software for any mistakes.

[Node.js]

General Information & Licensing

Code Repository	https://github.com/nodejs/node
License Type	MIT
License Description	This type of license allows users to reuse code for any purpose, even if it is part of proprietary software. Users are free to modify the code as long as they include the original copy of the MIT license in their distributions. The MIT License allows user to use the software for : ● Commercial use ● Modify ● Distribute ● Sublicense ● Private User The user of the Software with the MIT License must include: ● Copyright ● License
License Restrictions	● Software that contains an MIT license restricts its user to hold the author of the software liable. This means any mistakes can not hold the author of the software for any mistakes.

[Mongoose]

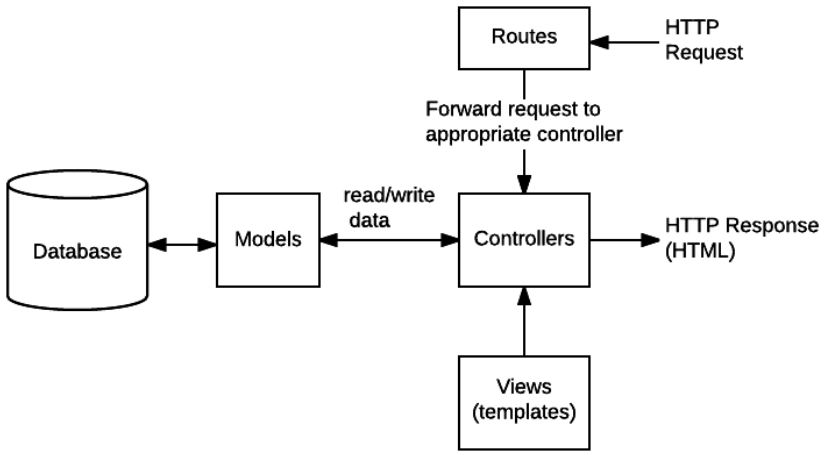
General Information & Licensing

Code Repository	https://github.com/Automattic/mongoose
License Type	MIT
License Description	This type of license allows users to reuse code for any purpose, even if it is part of proprietary software. Users are free to modify the code as long as they include the original copy of the MIT license in their distributions. The MIT License allows user to use the software for :

	● Commercial use ● Modify ● Distribute ● Sublicense ● Private User The user of the Software with the MIT License must include: ● Copyright ● License
License Restrictions	● Software that contains an MIT license restricts its user to hold the author of the software liable. This means any mistakes can not hold the author of the software for any mistakes.

This is an e-commerce technology that serves the purpose of online business between college students only. Like the big technologies, such as eBay, we can directly buy or sells student-owned goods, also we can have auctioning options as well.

Magic



Sample Image: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/routes#:~:text=%22Routes%22%20to%20forward%20the%20supported,to%20view%20in%20the%20browser.

Express.js:

Express provides a router method to help establish HTTP endpoints. For the GET request, we created instances of express and the router which can be seen in figure 2. The GET request is cached and then remains in the history of the browser. For the POST request, query parameters are sent by HTTP clients by forms to send the data to the node server which is then saved into the database. A method called .body helps extract the body from the request and allows the developers to extract information from it such as username, password, and/or university.

Express is a framework of Node that helps server side communication. Express makes it easy for the developer to get the request and send a response with minimal lines of code. As seen by the image above, Express for our applications will help take the HTTP request process the proper route it should take. From the routes it would go to a controller and determine the CRUD operation to use. As seen from the code below in Figure 1 which comes from loginController/login.js, the login controller looks to Read the database and find if the users request body by using .body on the req for a username, password and/or email is valid or already in use. In Figure 2 which comes from the .Routes/login.js, we utilize the Router to figure out which Controller to utilize which will determine what to do with the response. Currently, our only router is towards registration for a user which will take the user to the controller for login. Express also creates a listener to allow its developers to easily choose the port in which they want the website to listen to as seen in Figure 3 which comes from .models/User.js.

```
const Users = require('../models/Users.js')

const loginController ={
    register: async(req, res) =>{
        try{
            const {username, password, email, university} = res.body;
            const currentUser = await User.findOne({username})
            const currentEmail = await User.findOne({email})

            if (currentUser) return res.status(400).json({msg: "This username already exist"})
            if (password.length < 6) return res.status(400).json({msg: "Password must be greater than 6 characters"})
            if (currentEmail) return res.status(400).json({msg: "This email already exist"})

            res.json({msg:"Register Successful"})

        }catch(err){
            return res.status(500).json({msg:err.message})
        }

    }
}


module.exports = loginController
```

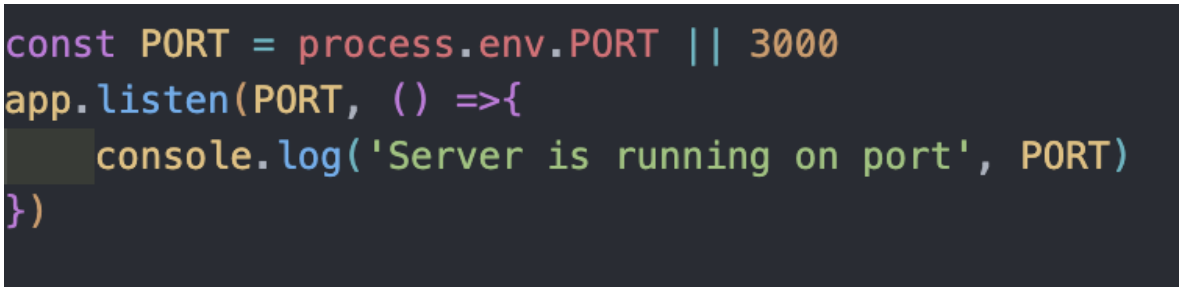
Figure 1:
https://github.com/brightcho5379/CSE312-TeamProject/blob/main/back-end/controllers/loginController.js - Lines: 1 - 24

```
const express = require('express');
const router = express.Router();
const loginController = require('..controllers/loginController')

router.post('/register', loginController)

module.exports = router      You, 2 days ago • Implment MongoDB
```

Figure 2:
https://github.com/brightcho5379/CSE312-TeamProject/blob/main/back-end/routes/login.js - Line 1 - 7



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

Figure 3:
https://github.com/brightcho5379/CSE312-TeamProject/blob/main/back-end/server.js - Lines 15 - 18

Mongoose:
Mongoose is an Object Data modeling library for Node for the use of MongoDB. Mongoose allows its developers to create specific schemas for their data in their database. As seen in Figure 4, our current schema for users are their username, password, and university. This structure will help organize the MongoDB database and add an application layer. It helps maintain type structures and makes sure that there is the

necessary data sent to the database.

```
const mongoose= require("mongoose");      You, 2 days ag

const userSchema = new mongoose.Schema({
    username:{
        type:String,
        required:true
    },
    password:{
        type:String,
        required:true
    },
    university:{
        type:String,
        required:true
    }
})

module.exports = mongoose.model('User',userSchema)
```

Figure 4

Node.js:
NodeJS is a runtime environment that is used to run JavaScript outside the browser. After the TCP socket is created, NodeJS allows us to run our JavaScript code to run outside of the browser such as running our controllers, routes, and models that hold Express.js and Mongoose.

We use NodeJS to run all of our back-end code, this is so we are able to create a database, and creating routes using express.js to handle HTTP Requests.