**CS 628 Full-Stack Development II**

**HOS02B Analyzing a Web Application with Express Scaffold**

July 16, 2020

12/29/2020 Updated by Min

School of Technology & Computing (STC)

City University of Seattle (CityU)

**Before You Start**

* The directory path shown in screenshots may be different from yours.
* This HOS does not explain all steps in detail**.** If you are not sure what to do:
  1. Consult the resources listed below.
  2. If you cannot solve the problem after a few tries, ask a TA for help.

**Resources**

* Holmes, S. & Harber, C. (2019). [Getting MEAN with Mongo, Express, Angular, and Node, Second Edition](https://login.proxy.cityu.edu/sso/skillport?context=148121). Manning Publications. (ISBN 9781617294754)
  + The source code for this application is on GitHub  
    <https://github.com/cliveharber/gettingMean-2>
* Express – Using Middleware, <https://expressjs.com/en/guide/using-middleware.html>
* HTTP Status Code, <https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html>
* gitignore, <https://git-scm.com/docs/gitignore>

**Learning Outcomes**

Students will be able to:

* Understand how an Express project is structured, which is the outcome of   
  “$ express --view=pug --git”

**Activities**

* Preparing for a MEAN project environment
* **Analyzing an Express project**
* Pushing your work to GitHub

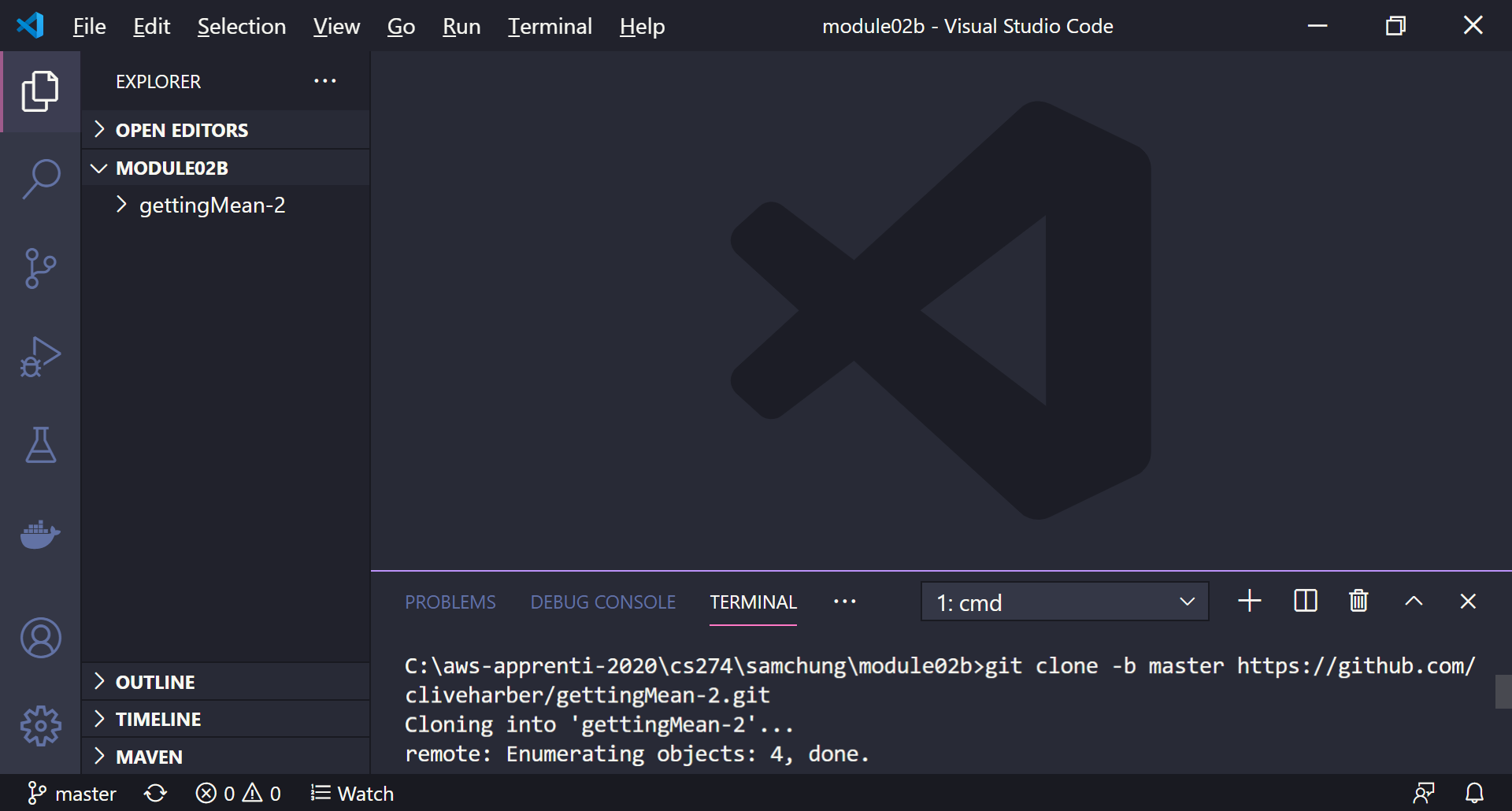
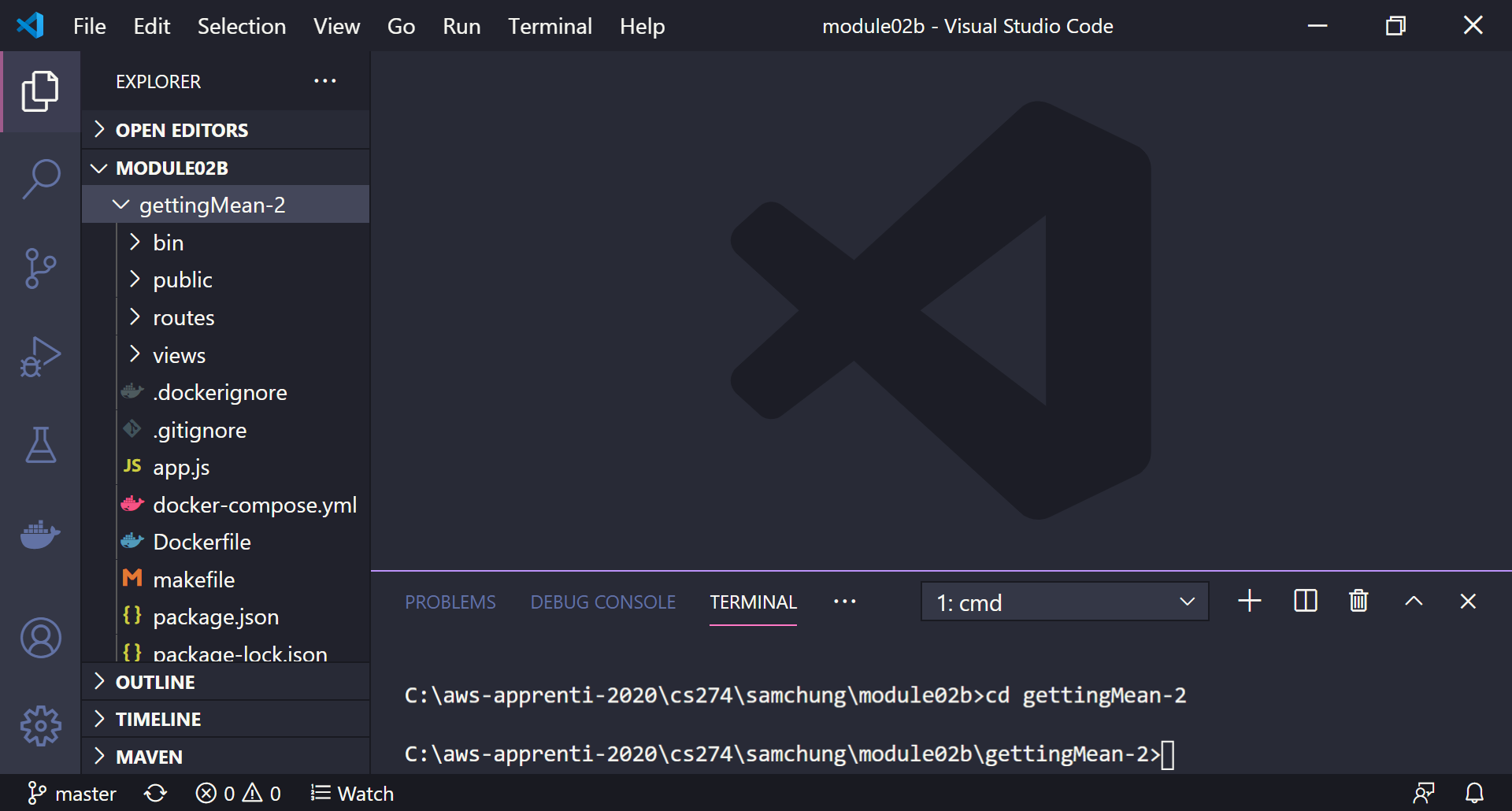
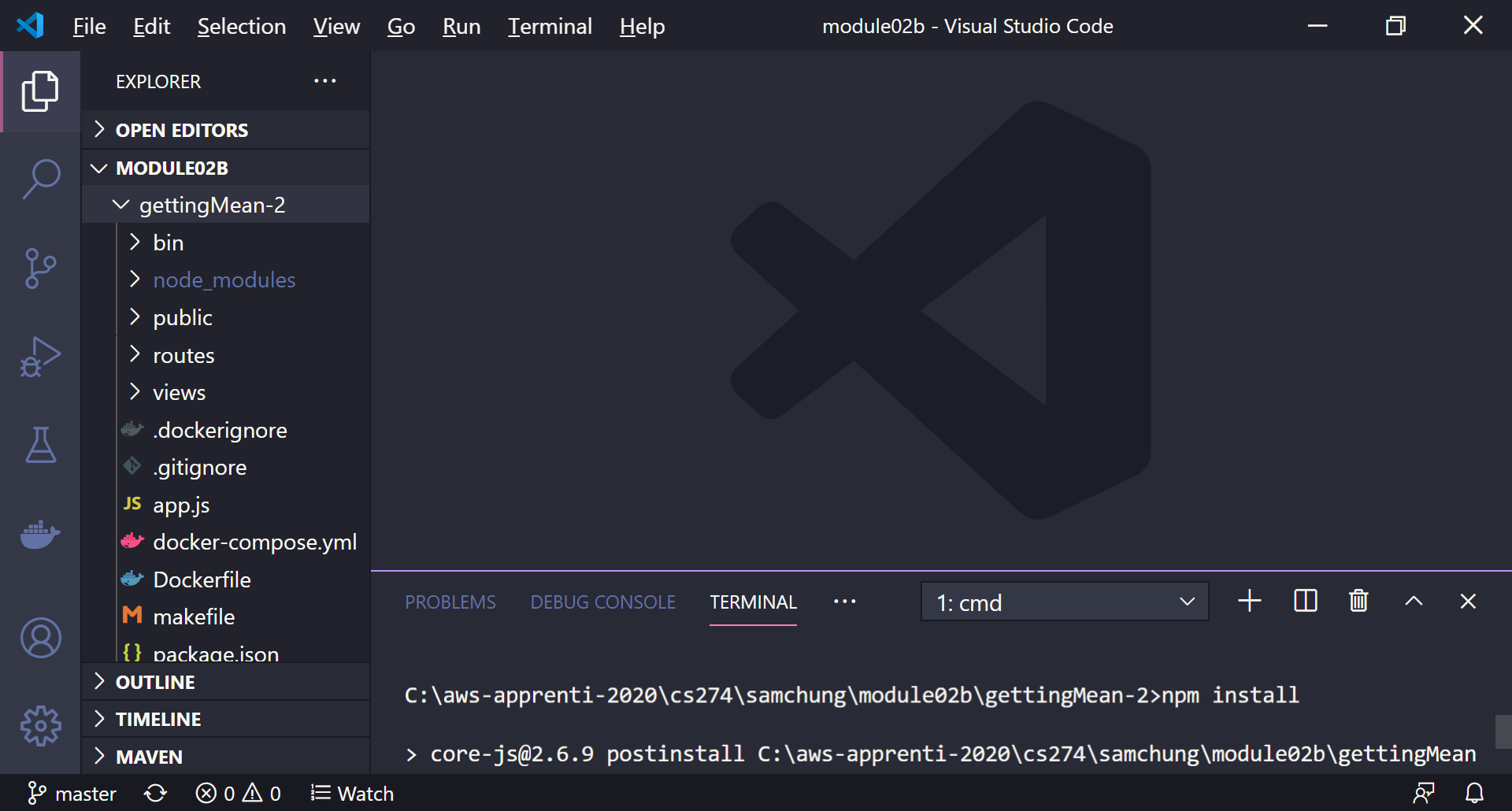
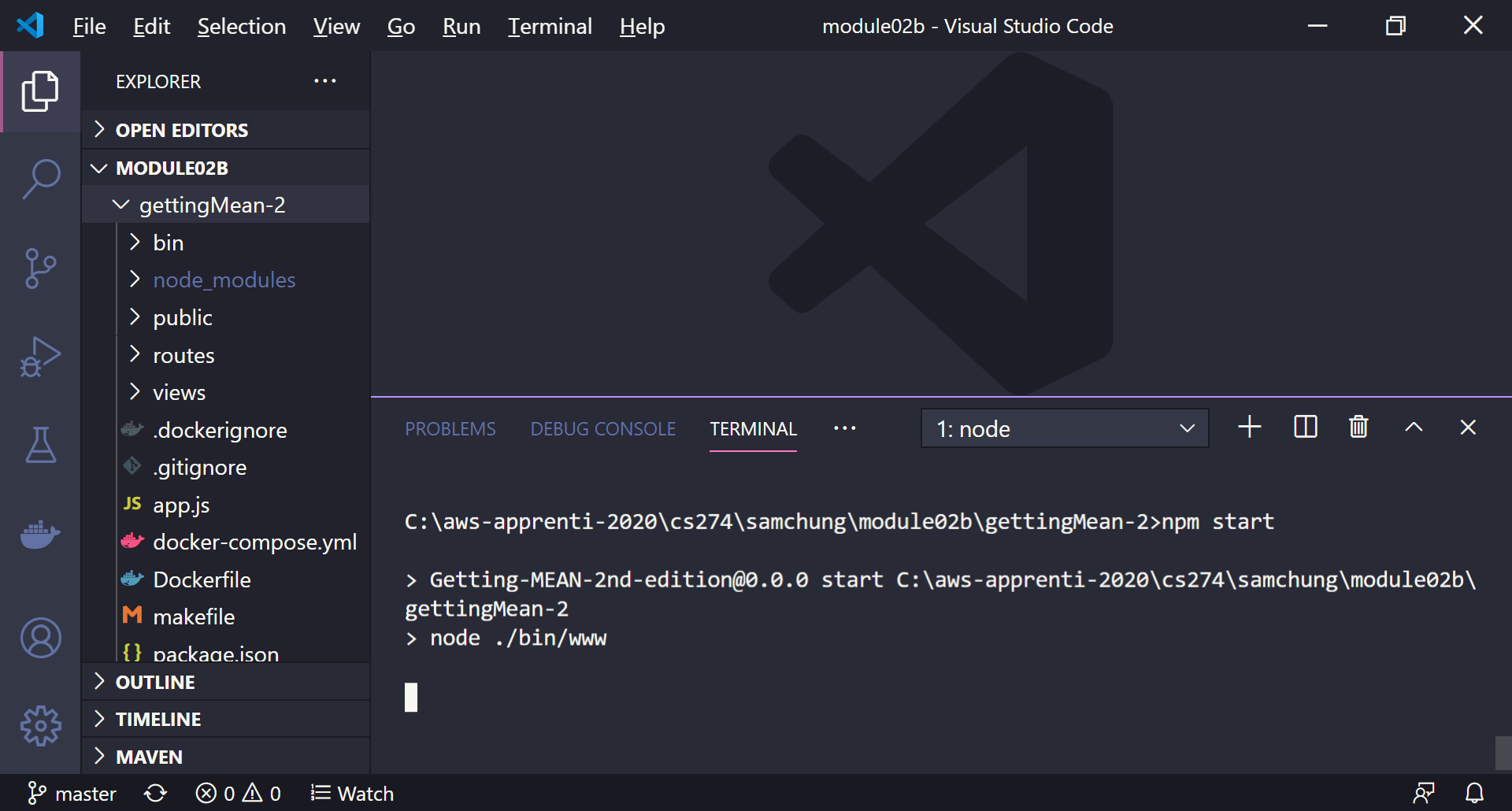
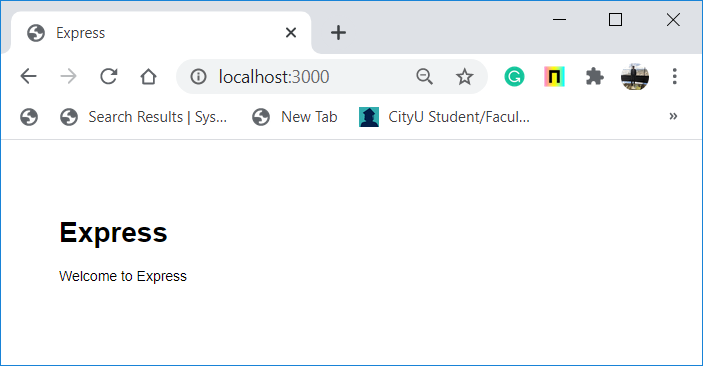
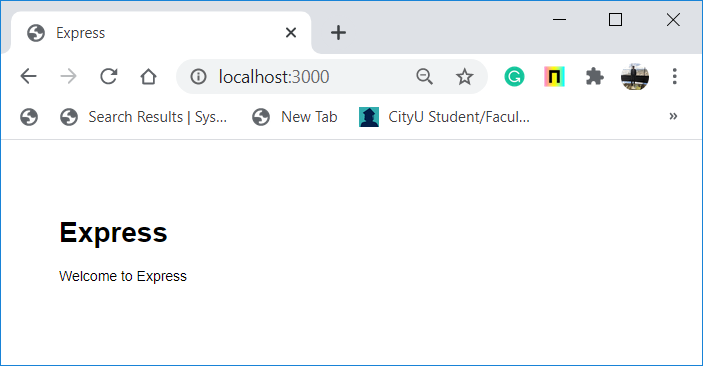
**Preparing for a MEAN project environment**

1. Create the “module02b” folder under the repo:  
   >>>mkdir module02b

You will see like that:  


1. Open the “module02b” folder in the VSCode.  
   Close the “Welcome” window.  
   Open the terminal (Ctrl+`).

**Analyzing an Express project**

1. Get a blank Express project for analysis:  
   >>>git clone -b **master** <https://github.com/cliveharber/gettingMean-2.git>  
   
2. Move to the application – gettingMean-2.  
   >>>cd gettingMean2  
   Then, expand the ‘gettingMean’ directory.  
   
3. Install all dependencies.  
   >>>npm install  
   Check whether the “node\_modules” was created.  
   
4. Execute the application.  
   Before your execution, close any application using the port number 3000,  
   >>>npm start  
   Then, open your web browser with the “localhost:3000” URL.  
     
     
   
5. Open the REAM markdown file, README.md.
   1. Find the application name for this code. gettingMean-2
   2. Also, understand the purposes of all GitHub branches. Looks like different chapters of the book work on different parts of the MEAN stack. The branches are related to these.
6. Open the node project file, package.json.
   1. Find how many packages are used. 7
   2. Find what each package is from <https://www.npmjs.com/package/package>
   3. Check the “scripts” attribute and a script name “start.”   
      And then, explain why you used “npm start” to execute this project. Looks like we pass an argument when we call node.exe. And the argument we pass is the start. Start is a listed script in our project, so it seems to trigger www.js inside the directory that follows the start script.
   4. You can also see the “package-lock.json” file.   
      Why do you need it? It seems to contain the dependencies and where they were downloaded from.  
      How is it different from the “package.json” file. Package.json file seems to list main dependency library names aswell as other attributes about the project such as scripts
7. Open the “www” file under the “bin” directory.
   1. Explain how “npm start” calls “app.js” by mentioning the related statement with its line number. Line 6 contains a call to node.exe with the argument ./bin/www
   2. Explain why you used the port number 3000 by mentioning each related statement with its line number. Line 75 sets 3000 as port
8. Open the “app.js” file.
   1. How many external packages are used? 8  
      What are they? The required libraries such as cookie parser, body parser, logger.  
      Mention each related statement with its line number. 1-9
   2. What view engine is used? pug  
      Where the views directory is located? In the main directory of the project  
      Mention each related statement with its line number. 15
   3. How many routes are used? 2  
      Which path is used for each route? / and /users  
      What files are used for each route? ./routes/index' and ./routes/users  
      Mention each related statement with its line number. 8-9, 25-26.
   4. Which directory serves static assets? public  
      Mention each related statement with its line number. 23
   5. How many middleware functions are used? 2  
      What are they? Error handlers  
      Mention each related statement with its line number. 28-44
   6. How many HTTP status codes are used? 2  
      What are they? 404, 500  
      What does each code mean? No source, error  
      Mention each related statement with its line number. 31, 42
   7. Several operators are used in this code.   
      Which line shows the “identity” operator? How does it work? 39 checks the value and the type  
      Which line shows the “or” operator? How does it work? 42 returns true if either is true  
      Which line shows the conditional (ternary) operator? How does it work? 39 returns second operand if first operand is true. Returns third operand if first operand is false.
   8. Which module is exported? app  
      Find the application name to be exported?   
      Mention each related statement with its line number. 46
9. Source Code Management (SCM)
   1. Open the “.gitignore” file.
   2. Is the “node\_modules” package pushed to your GitHub?   
      Mention each related statement with its line number. 27
10. Directory Structure
    1. How many subdirectories exist under the Express project? 5
    2. What is the purpose of each directory? Routes includes javascript files. Public includes css, views include pugs
    3. Mention each directory name. public, routes, views, bin, node\_modules
11. Consider the output in the web browser.  
    
    1. Which files are related to the output? Index.pug
    2. What is the role of each file? Displaying the data dynamically

**Pushing your work to GitHub**

Run the following commands to push your work to the GitHub repository:

Open the terminal from the VSCode by hitting the control + ~ key and type the following command:

>>> git add .

>>> git commit -m “Submission for Module 2 --yourname”

>>> git push origin master

If you cannot remember your branch name, run the command “git status” to check.