



# Maintenance Manual

**Project name:** Lilac Ventures Website

**Team name:** Algo Avengers

David Enzler

Prasad Prabhu

Nathaniel Cruz

Justin Roome

Robert Mancuso

Ryan McCormick

Francisco Avalos

Sir Christian Orefe Casals

Date: 12/1/2023

# Table of Contents

<b>Table of Contents.....</b>	<b>2</b>
<b>1.0 Introduction.....</b>	<b>3</b>
<b>2.0 Implementation tools and technologies.....</b>	<b>4</b>
Some Words on Transitioning from Javascript to Typescript.....	5
<b>3.0 Runbook.....</b>	<b>9</b>
List what to do if services goes down.....	9
Database becomes corrupt.....	11
Third Party services go down (API's).....	11
How to configure a computer for development work.....	12
How to make a code change.....	17
Merging branches from pull request.....	33
How to check for errors in the code.....	36
<b>4.0 Deployment.....</b>	<b>38</b>
Overview.....	38
Amplify.....	38
Building React Files for Deployment.....	39
Elastic Beanstalk.....	41
Deploying NodeJS Code to Existing Environment.....	43
<b>5.0 User interaction.....</b>	<b>48</b>
<b>6.0 Database.....</b>	<b>63</b>



## 1.0 Introduction

Description of the application:

Lilac Ventures requires a software application to fulfill the business needs of Gail Tateyama's financial coaching service. The application, a user-friendly website, must be ADA-compliant and mobile-friendly, featuring an interactive customer account portal. This portal allows clients to submit forms, manage appointments through a calendar system, and communicate easily with the client. The website also incorporates a secure payment processing system, prioritizing the safety of financial data. Additionally, a dedicated company email address is set up, and a separate data management system captures website traffic analytics and generates reports for Gail Tateyama's exclusive access. The software aims to streamline financial coaching services, enhance customer engagement, and ensure the secure management of sensitive information.

## 2.0 Implementation tools and technologies

---

Programming languages Used, libraries, frameworks SDK's

- IDE Used
  - Visual Studio Code
- Programming Languages
  - Typescript, React, Express.js, Node.js
- Technologies, libraries and frameworks
  - Tailwind CSS
- Web Services
  - AWS Beanstalk and Amplify

Database

- [MongoDB](#)
  - Username: REDACTED
  - Password: REDACTED

Payment Services or other API's

- [Stripe - Payment API](#)
  - Username: REDACTED
  - Password: REDACTED
- [AWS](#)
  - Username: REDACTED
  - Password: REDACTED
- [GoDaddy](#) - Domain registration
  - Username: REDACTED
  - Password: REDACTED

## Some Words on Transitioning from Javascript to Typescript:

---

The purpose of this section is to assist an experienced web developer who may be well-versed in Javascript, but might not have much (if any) experience with Typescript. We have compiled this brief guide to how the code on the site pages functions. Via an example page, we will explain maintaining and modifying the AdminTable component, a React component written in Typescript. It focuses on Typescript-specific features and best practices for managing customer data display and interactions.

### Typescript vs Javascript

**Type Annotations:** Typescript extends Javascript by adding type annotations. For example, in this page the *Client* interface defines the shape of client objects. When working with Typescript, it is critical that you ensure variables and function parameters align with these defined types.

**Interfaces:** Unlike Javascript, Typescript allows declaring interfaces (e.g., *Client*, *AdminTableProps*). They describe object shapes, enforcing structure and providing IntelliSense in the Visual Studio Code IDE this project utilized.

**Type Safety in Functions:** Typescript checks static types at compile time - *this is very different from Javascript*. For example, *fetchClients* is typed to return *Promise<Client[]>*, ensuring the function resolves to an array of *Client*. While adding more work up front, this difference helps with error catching and debugging later in the development process.

**Error Handling:** Typescript's strict typing procedures help identify potential type-related errors during development, unlike Javascript's runtime errors, which may cause errors to go unnoticed until something breaks, or returns the wrong output, etc., which can make debugging more difficult or cause errors in deployment or afterwards.

### Maintenance Tasks

#### **Modifying Table Rows:**

To change displayed client attributes, update the *Client* interface and modify the table's *<thead>* and *<tbody>* sections in the JSX/TSX.

Whenever this is done, make sure to implement these changes in the *fetchClients* function; to ensure the backend provides necessary data.

```

11
12  interface Client {
13    id: string;
14    firstName: string;
15    lastName: string;
16    city: string,
17    street: string,
18    state: string,
19    zip: string,
20    phone: string;
21    email: string;
22    marital: string;
23    employment: string;
24    progress: string;
25    accountLink: string;
26  }
27

```

```

112 <table className="clients-table">
113   <thead>
114     <tr>
115       <th>First Name</th>
116       <th>Last Name</th>
117       <th>Address</th>
118       <th>Mobile Number</th>
119       <th>Email</th>
120       <th>Marital Status</th>
121       <th>Employment Status</th>
122       <th>Progress Step</th>
123     </tr>
124   </thead>
125   <tbody>
126     {clients.map(client => (
127       <tr key={client.id}>
128         <td>{client.firstName}</td>
129         <td>{client.lastName}</td>
130         <td>{formatAddress(client.street, client.city, client.state, client.zip)}</td>
131         <td>{client.phone}</td>
132         <td>{client.email}</td>
133         <td>{client.marital}</td>
134         <td>{client.employment}</td>
135         <td>
136           {client.progress}
137           <button onClick={() => handleIncrease(client.email, client.progress)} className="green-button-admin">
138             <button onClick={() => handleDecrease(client.email, client.progress)} className="red-button-admin">
139           </td>
140         </tr>
141     ))}
142   </tbody>

```

### Updating Row Contents:

Adjust the *Client* interface to add or remove properties.  
 In the JSX/TSX, map these properties to table cells.

### Increment/Decrement Buttons:

In our code, *handleIncrease* and *handleDecrease* control these actions. Modify them, if necessary, to change any logic needed, ensuring continued compliance with the Typescript types used in the project.

For UI changes, update the corresponding JSX/TSX in the map function within <tbody>.

```

68  const handleIncrease = async (email: string, progress: string) => {
69    //console.log(`Increased for email: ${email}`);
70    try {
71      const id = await getClientIDByEmail(email);
72      //console.log(id);
73
74      const currentProgress = progress;
75      await axios.put(`${baseURL}/customerProgress/${id}`, { "progress": currentProgress + 1 });
76      setRefreshFlag(prevFlag => !prevFlag);
77    } catch (error) {
78      // Handle errors here
79      console.error('Error in handleIncrease:', error);
80    }
81  };
82

```

### Backend Database Integration:

Modify *fetchClients*, *getClientIDByEmail*, *handleIncrease*, and *handleDecrease* for any backend changes. These functions interact with other pages/files on the site - be aware you might need to refer to those for making certain changes to the code on this page.

```

45  const fetchClients = async (): Promise<Client[]> => {
46    try {
47      const response = await axios.get(`${baseURL}:8080/clientInfoUpdate/`);
48      return response.data;
49    } catch (error) {
50      console.error('Error fetching clients:', error);
51      throw new Error('Failed to fetch clients');
52    }
53  };
54
55  async function getClientIDByEmail(email: any) {
56    try {
57      const response = await axios.get(`${baseURL}/customerProgress/getID/${email}`);
58      const id = response.data.id; // Assuming the response contains an "id" property
59      //console.log(id)
60      return id;
61    } catch (error) {
62      // Handle errors here
63      console.error('Error fetching client ID:', error);
64      //throw error; // Re-throw the error to be handled by the calling function
65    }
66  }

```

Ensure the data types in these functions align with the backend responses and the Client interface.

### Error Handling and Debugging:

TypeScript's static typing analysis helps catch type-related issues early, earlier than one might be used to in JavaScript. Use this alongside traditional debugging methods - one common method we used often were 'try-catch' statements.

```

try {
  // Enter your function or other code here
} catch (error) {
  if (error instanceof Error) {
    // log statement of the mistake, helpful for debugging:
    console.error('Error:', error.message);
  }
}

```

## 3.0 Runbook

### List what to do if services goes down

Note: More information on accessing the consoles for both AWS Amplify and AWS Beanstalk can be found in the 4.0 Deployment section.

#### Amplify

1. If AWS Amplify goes down, then you can check the portal for Amplify to check the health of the instance.

The screenshot shows the AWS Amplify Hosting environments console. At the top, there are tabs for 'Hosting environments' (which is selected) and 'Backend environments'. Below the tabs, a message says 'This tab lists all connected branches, select a branch to view build details.' On the right, there is an orange 'Add environment' button. The main area displays a card for the branch 'dev3273'. The card features a small icon of a computer monitor with an arrow pointing right. To the right of the icon, the text 'Deployment successfully completed.' is displayed above a green progress bar that is fully filled to 100%. Below the progress bar, there is a table with two rows. The first row contains 'Domain' and 'https://www.lilacfinancials.com' followed by a blue link icon. The second row contains 'Last deployment' and '11/30/2023, 11:11:28 PM'.

2. However, if there are issues with AWS, then you should reach out using the support link in the left side of the AWS Amplify console

The screenshot shows the AWS Amplify console interface. On the left side, there is a vertical sidebar with several links: 'Documentation' with a link icon, 'Support' with a link icon, and other options like 'Logs' and 'Metrics' which are currently disabled. The main content area is mostly blank, indicating the user is on the support page.

#### AWS Beanstalk

1. AWS Beanstalk deploys 2 machines: a load balancer and an EC2 instance. The status of both can be seen from the environment tab

The screenshot shows the AWS Elastic Beanstalk Environments page. At the top, there's a header with 'Environments (1) Info' and buttons for 'Actions' and 'Create'. Below the header is a search bar with the placeholder 'Filter environments'. The main table has columns for 'Environment name', 'Health', 'Application', 'Platform', 'Domain', and 'Running version'. A single row is selected, showing 'Lilacventuretest-env' with a red 'Severe' status icon, 'lilac\_vent...' application, 'Node.js 1...' platform, 'Lilacventuretest-env.eba-qpu...' domain, and '1.7-1' running version.

2. The above instance is listed as 'Severe'
3. Click on the environment name to get more information

The screenshot shows the 'Environment overview' page for 'Lilacventuretest-env'. It includes sections for 'Environment overview' (Health: Severe, View causes; Domain: Lilacventuretest-env.eba-qpugnevs.us-west-1.elasticbeanstalk.com), 'Platform' (Platform: Node.js 18 running on 64bit Amazon Linux 2023/6.0.3, Running version: 1.7-1, Platform state: Supported), and tabs for 'Events', 'Health', 'Logs', 'Monitoring', 'Alarms', 'Managed updates', and 'Tags'. The 'Events' tab is selected, showing 77 events. One event is highlighted: 'November 30, 2023 19:01:52 (UTC-8)' with a 'WARN' type, detailing a deployment failure where environment health transitioned from Degraded to Severe due to 100.0% error rate and ELB unhealthiness.

4. Information can be seen in the Events tab. Additionally, security events are found in the Monitoring and Alarms tabs.
  - a. In this example, an application failed to deploy.

**Instance deployment failed. For details, see 'eb-engine.log'.**

5. Download and check logs from the Logs tab

The screenshot shows the AWS CloudWatch Logs interface. At the top, there are tabs for Events, Health, Logs (which is selected), Monitoring, Alarms, Managed updates, and Tags. Below the tabs, there's a search bar with a 'Request logs' button and a dropdown for 'EC2 instance'. A message says 'No logs' and 'No logs to display'. To the right, a dropdown menu titled 'Actions' lists several options: Load configuration, Save configuration, Swap environment domain, Clone environment, Abort current operation, Restart app server(s), Rebuild environment, Terminate environment, and Restore environment.

## Database becomes corrupt

Currently, a backup is not set up for the database. However, MongoDB Cloud offers snapshot features that can be configured on their clusters.

More information on conjuring regular snapshots can be found on their site:  
<https://www.mongodb.com/docs/atlas/backup/cloud-backup/overview/>

## Third Party services go down (API's)

Stripe will list issues on their dashboard. Generally, if there are issues with any of the Stripe payment APIs, the responses to the endpoints will return 5xx statuses.

All endpoints using the Stripe API are under the `./customerBilling` endpoints. The routes file for the services is `./routes/customerBilling.ts`.

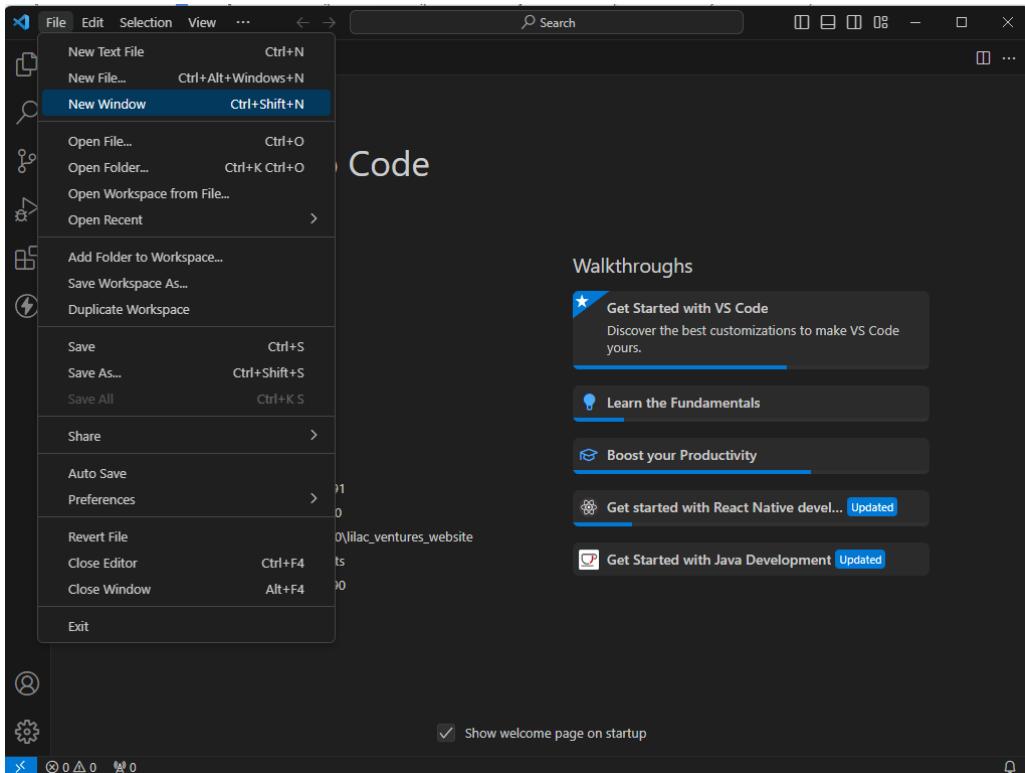
If there are issues with payments being processed or invoices not being created, then the first place to check is the Stripe Dashboard. They will list any service outages there. If it is not clear from the dashboard where the issues are stemming from, then set listeners on the endpoints.

Generally, anything returning a 5xx status code indicates an issue on Stripe's servers.

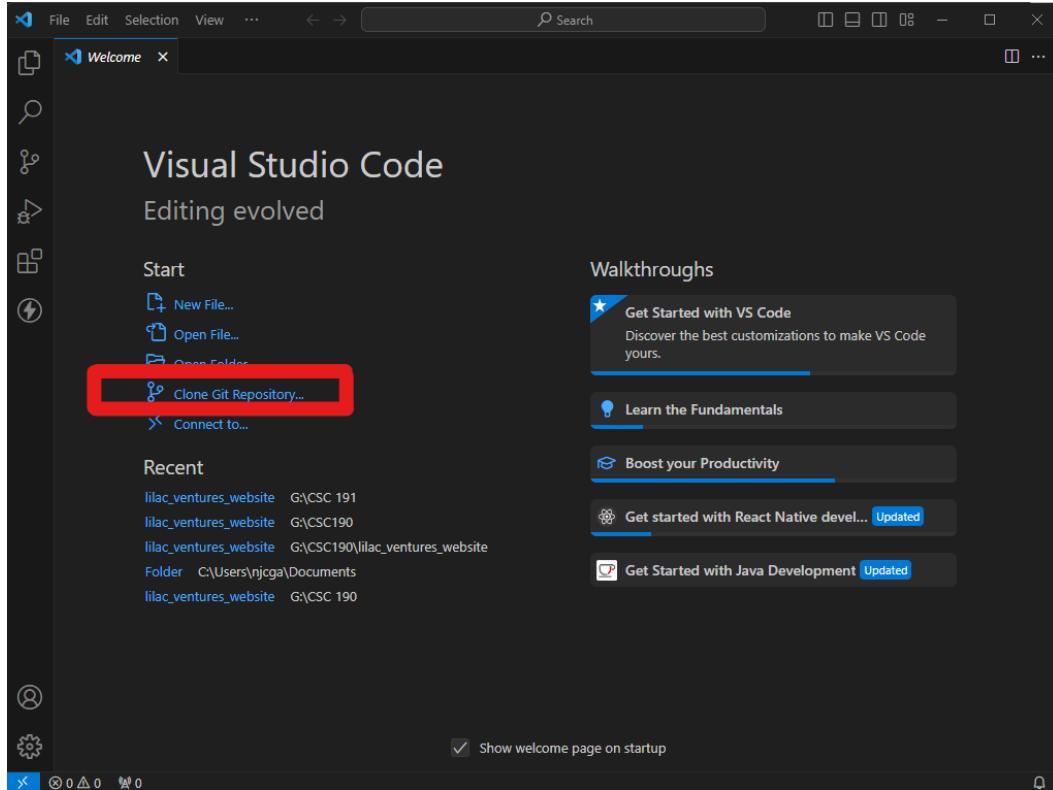
Generally, a status code 4xx indicates a configuration issue. Make sure to check the environmental variables of the NodeJS Elastic Beanstalk server to ensure all tokens are up to date.

## How to configure a computer for development work

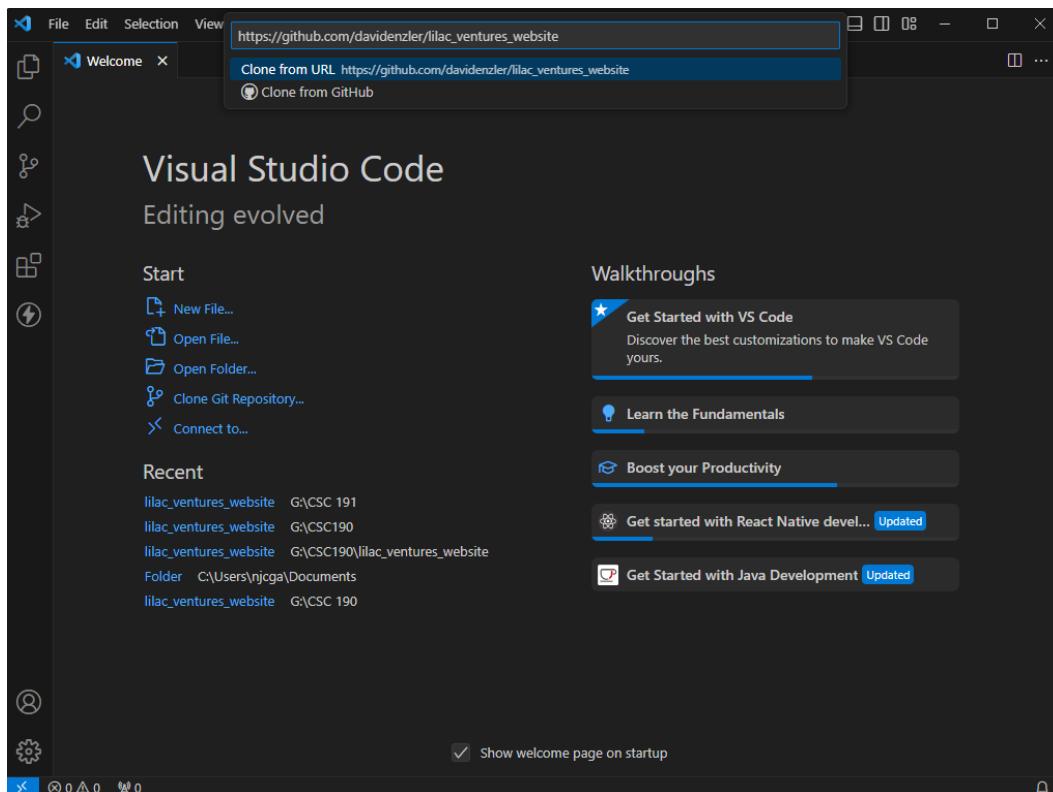
- Install Visual Studio Code, Node.js, and Git.
- In Visual Studio code, Click File in the top left and Open a new Window.



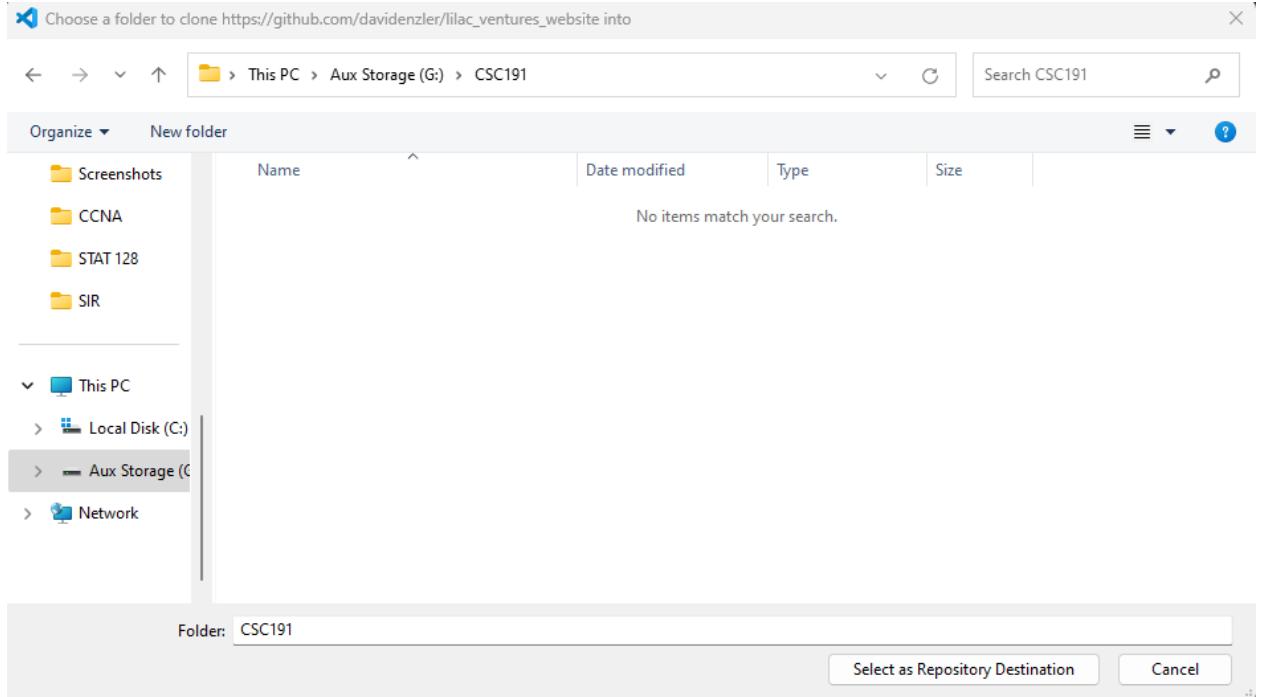
- Select Clone Git Repository



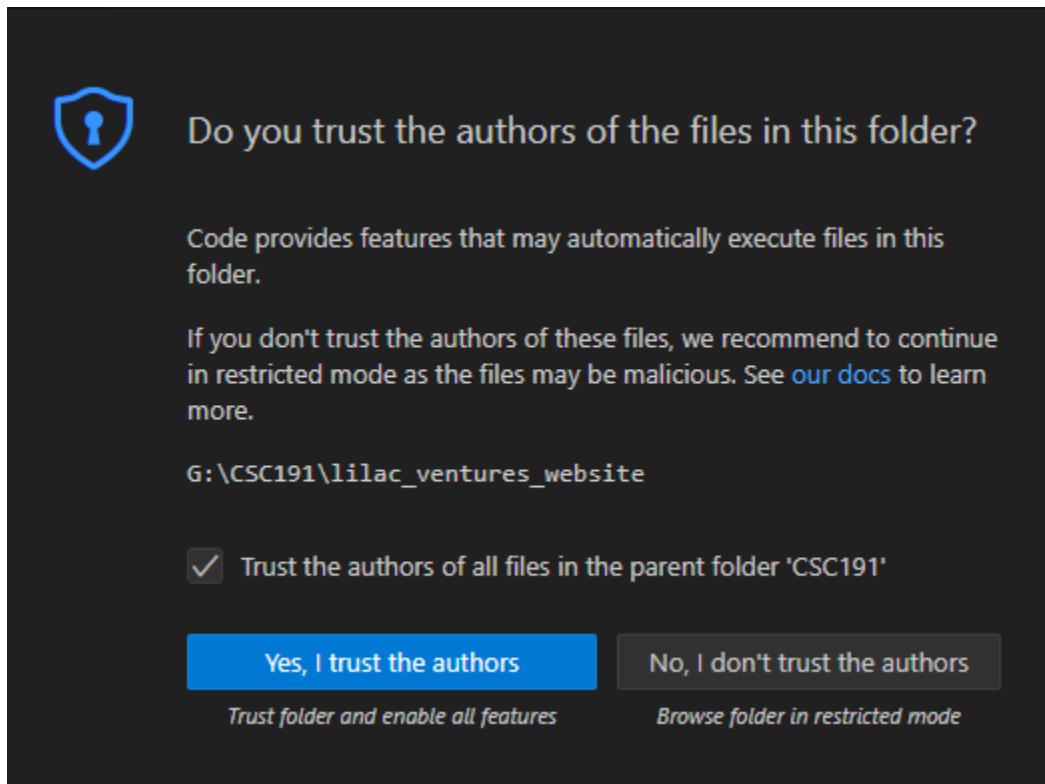
d. Enter the link for the for the github Lilac Ventures github repository



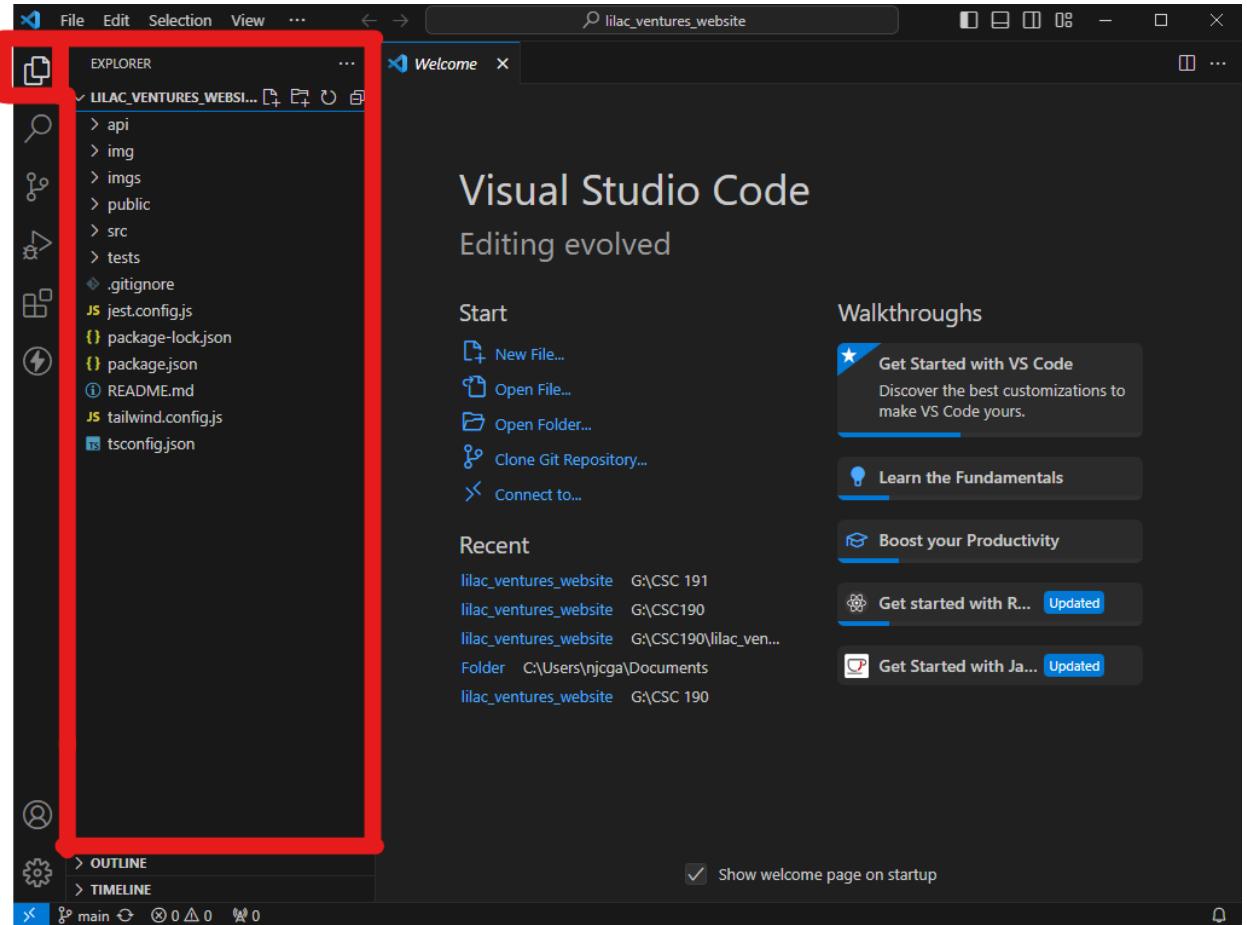
e. Select where to download the cloned repository



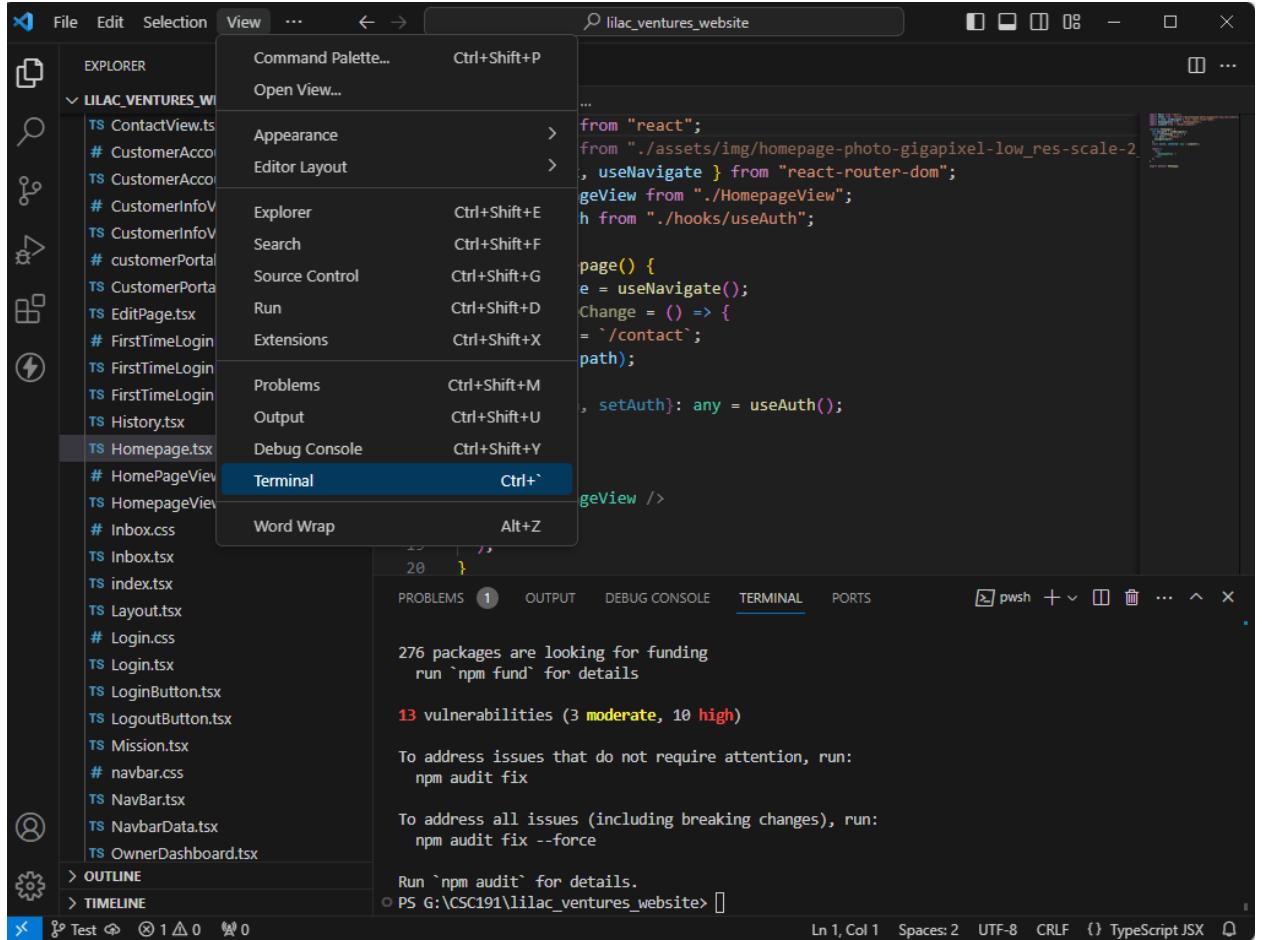
- f. In the next popup, select Open to open the repository, then select Yes, I trust the authors



- g. The files in the repository can now be opened and edited from the explorer tab



h. Press **Ctrl+`** or go to **View->Terminal** to open a new terminal tab



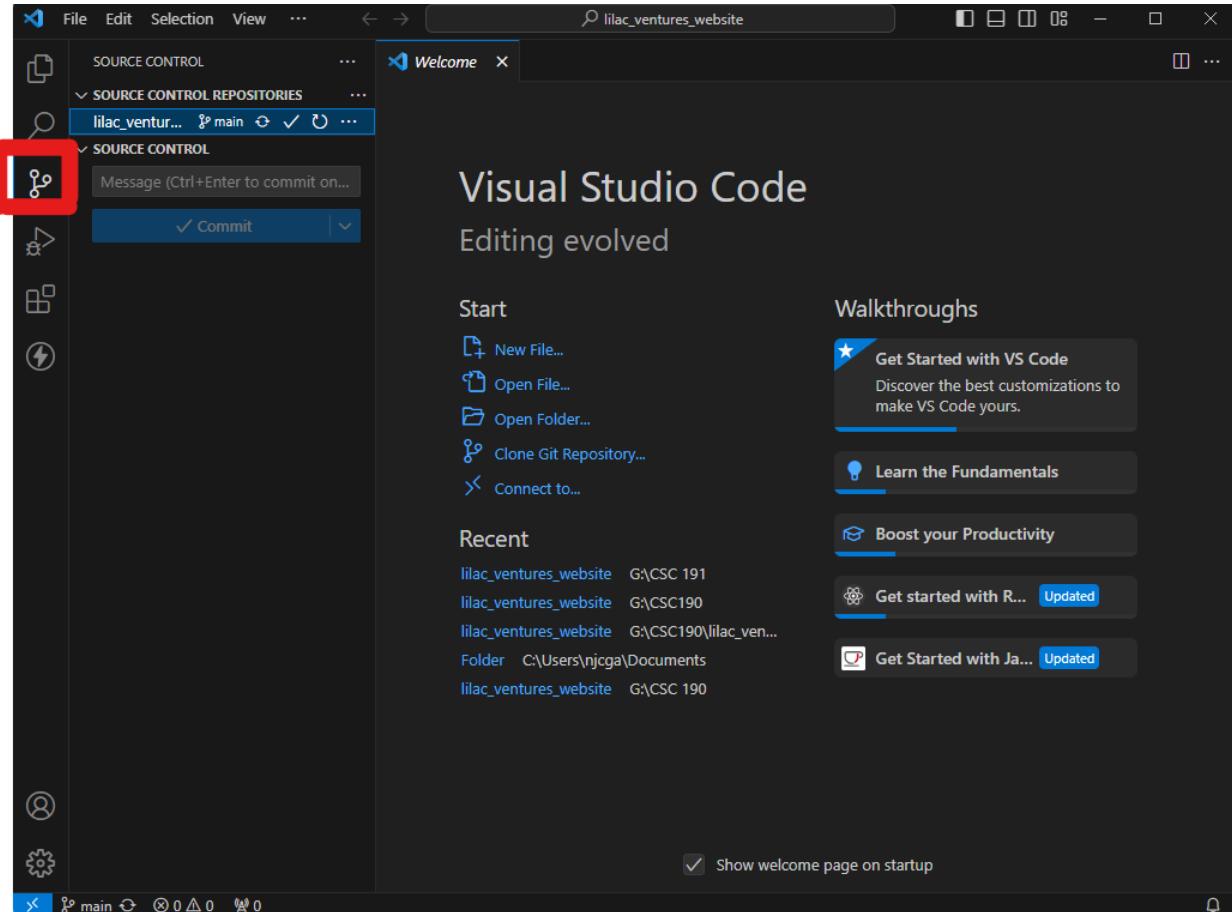
- In the terminal that appears at the bottom of the screen type `npm install` and press enter to install any missing dependencies

The screenshot shows the Visual Studio Code interface with the following details:

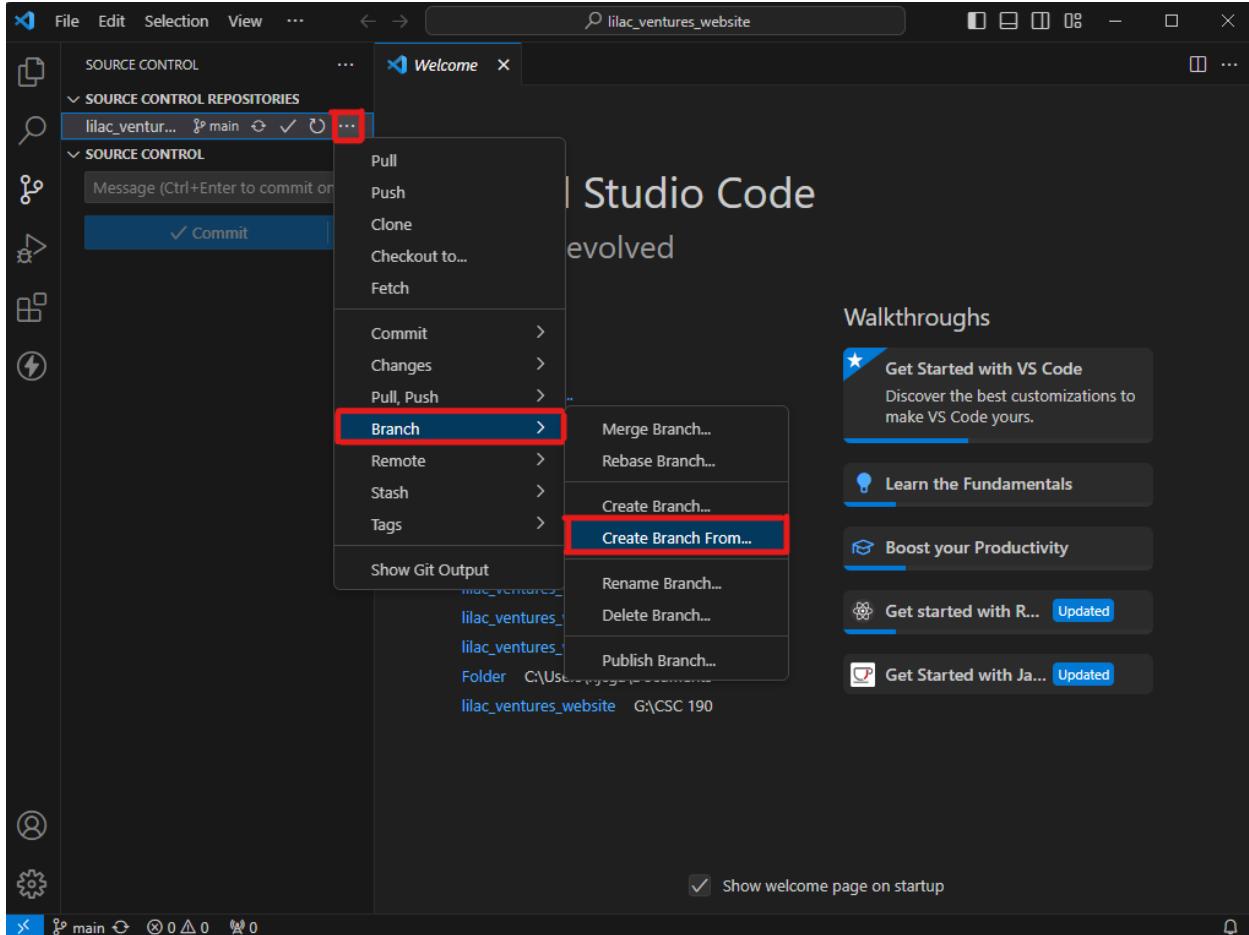
- File Explorer (Left):** Shows a tree view of files under the folder "LILAC\_VENTURES\_WEBSITE". The file "Homepage.tsx" is selected.
- Editor (Top Center):** Displays the code for "Homepage.tsx". The code imports React, money from a CSS file, and various hooks and components from "react-router-dom" and "hooks/useAuth". It defines a function "Homepage" that uses the "useNavigate" hook to handle route changes and the "HomepageView" component.
- Terminal (Bottom):** Shows the command "PS G:\CSC191\lilac\_ventures\_website> npm install" entered in the terminal tab. This command is highlighted with a red box.

## How to make a code change

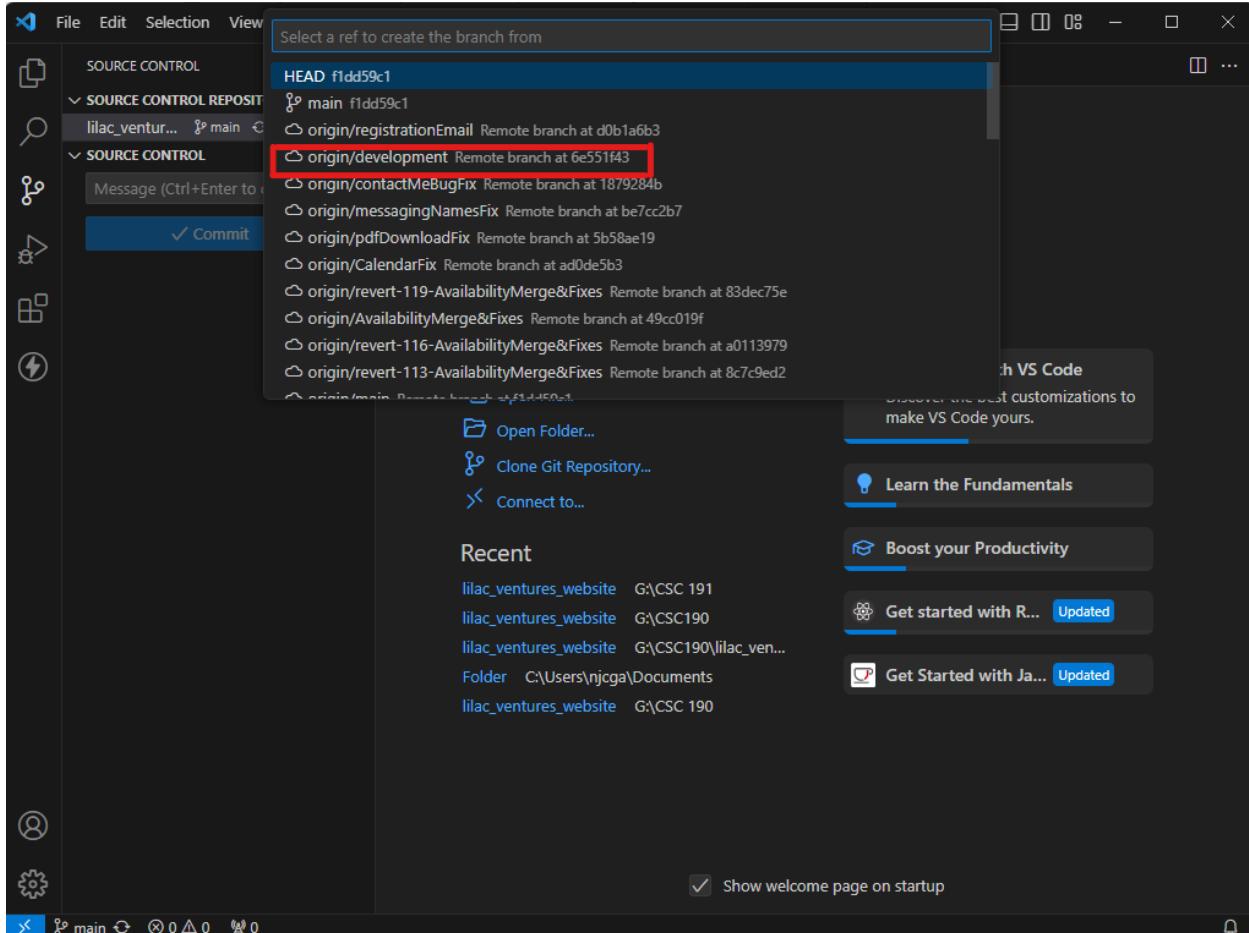
- To make changes, first, you must create a new branch. This can be done from the source control tab.



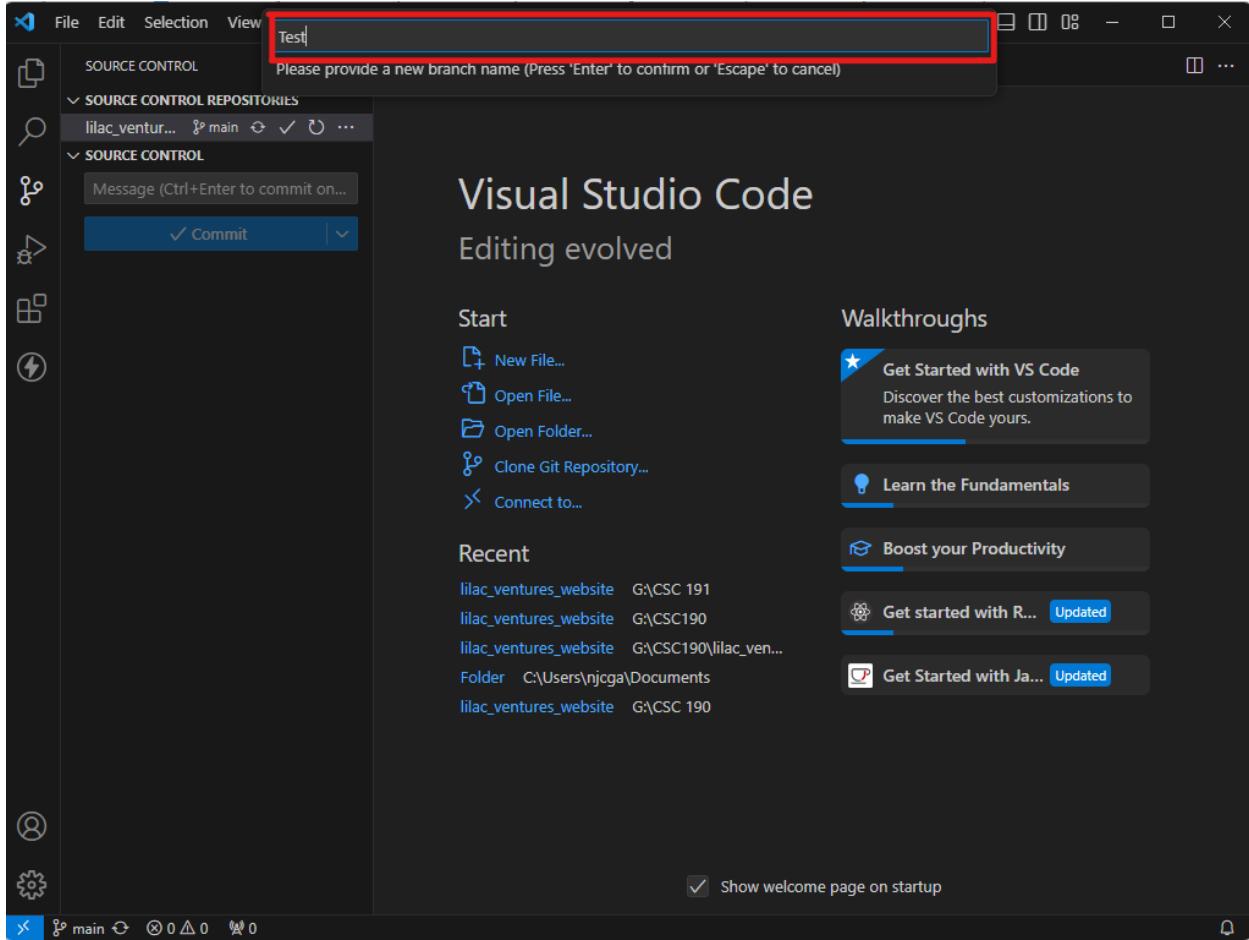
- b. Click on the ellipsis on the line highlighted in blue, and from the drop-down menu, navigate to branch then select create branch from



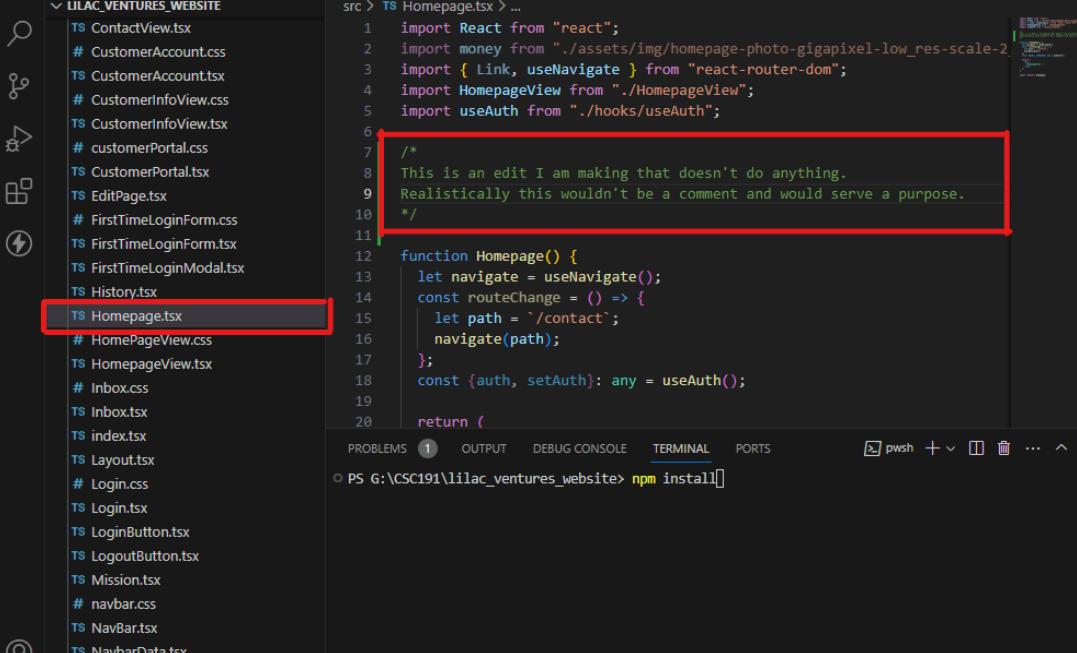
- c. In the popup menu, select the origin/development branch



- d. Enter the name of the new branch. Once done, hit the enter key



- e. Open and edit any necessary file or right click on the appropriate folder and select new file to add new files from the Explorer Tab



The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows the project structure under "LILAC\_VENTURES\_WEBSITE". The file "Homepage.tsx" is selected and highlighted with a red box.
- Editor:** Displays the content of "Homepage.tsx". A multi-line comment is present, starting with a block comment /\* and ending with \*/. The text inside the comment is:

```
/*
This is an edit I am making that doesn't do anything.
Realistically this wouldn't be a comment and would serve a purpose.
*/
```
- Terminal:** Shows the command PS G:\CSC191\lilac\_ventures\_website> npm install[]
- Bottom Status Bar:** Shows the current file is "Homepage.tsx", line 9, column 68, with 2 spaces, CRLF, and TypeScript JSX.

The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, ..., <, >, <ilac\_ventures\_website>
- Search Bar:** <ilac\_ventures\_website>
- Sidebar:** EXPLORER, LILAC\_VENTURES\_WEBSITE (expanded), src (selected), > api, > img, > imgs, > node\_modules, > public.
- Active Editor:** TS example.tsx
- Bottom Status Bar:** Ln 1, Col 1, Spaces: 4, UTF-8, CRLF, {}, TypeScript JSX

A context menu is open over the file 'example.tsx' in the 'src' folder. The menu items are:

- New File...
- New Folder...
- Reveal in File Explorer Shift+Alt+R
- Open in Integrated Terminal
- Share >
- Find in Folder... Shift+Alt+F
- Cut Ctrl+X
- Copy Ctrl+C
- Paste Ctrl+V
- Copy Path Shift+Alt+C
- Copy Relative Path Ctrl+K Ctrl+Shift+C
- Rename... F2
- Delete Delete
- Create Maven Project

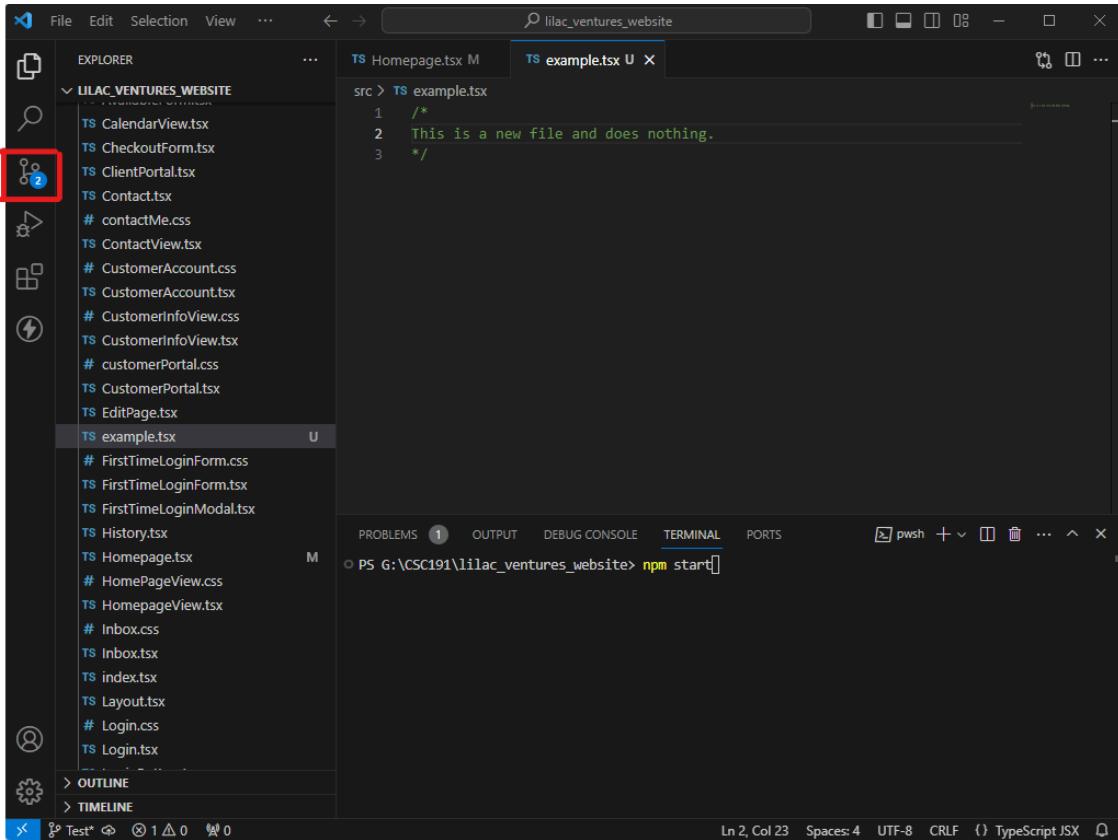
The 'TERMINAL' tab is active, showing the command 'npm install'.

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer (Left):** Shows a tree view of files under the folder "LILAC\_VENTURES\_WEBSITE". The file "example.tsx" is selected and highlighted with a red box.
- Editor (Center):** Displays the content of "example.tsx". The code is:

```
/*  
This is a new file and does nothing.  
*/
```
- Terminal (Bottom):** Shows the command "PS G:\CSC191\lilac\_ventures\_website> npm install" entered in the terminal.
- Status Bar (Bottom):** Shows "Ln 2, Col 23" and "Spaces: 4" and other file status indicators.

f. Press Ctrl+s to save any changes. Save changes will be added to the source control tab.



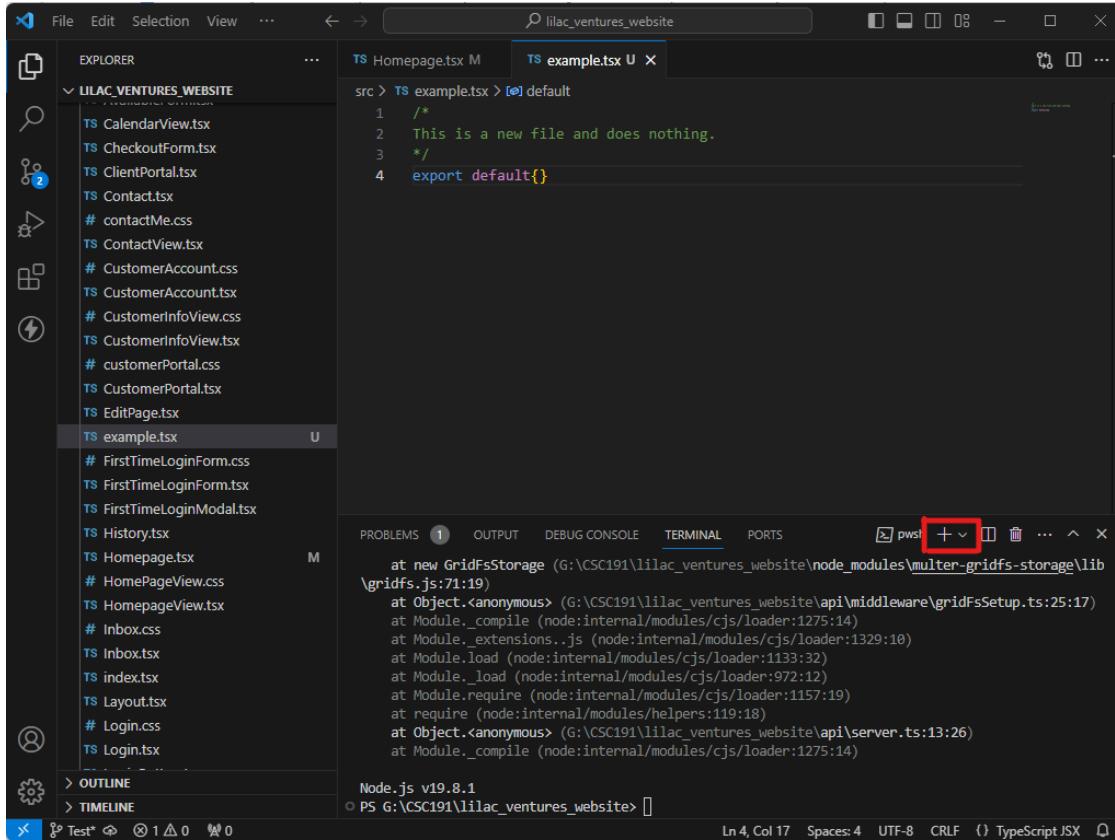
- g. To test the changes, in a terminal, type npm start and press enter. This will load the frontend components. Note: The site will not have full functionality until the backend is loaded, details are in the next step.

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer (Left):** Shows the project structure under 'LILAC\_VENTURES\_WEBSITE'. Files listed include: CalendarView.tsx, CheckoutForm.tsx, ClientPortal.tsx, Contact.tsx, contactMe.css, ContactView.tsx, CustomerAccount.css, CustomerAccount.tsx, CustomerInfoView.css, CustomerInfoView.tsx, customerPortal.css, CustomerPortal.tsx, EditPage.tsx, example.tsx (selected), FirstTimeLoginForm.css, FirstTimeLoginForm.tsx, FirstTimeLoginModal.tsx, History.tsx, Homepage.tsx, HomePageView.css, HomePageView.tsx, Inbox.css, Inbox.tsx, index.tsx, Layout.tsx, Login.css, and Login.tsx.
- Editor (Top Right):** Displays the content of 'example.tsx'. The code is:

```
1  /*
2  This is a new file and does nothing.
3  */
```
- Terminal (Bottom Right):** Shows a PowerShell terminal window with the command `PS G:\CSC191\lilac_ventures_website> npm start`. The terminal tab is highlighted with a red box.
- Status Bar (Bottom):** Shows 'Ln 2, Col 23' and other status information.

- h. To load the backend, click the plus icon in the top right of the terminal window to open a new terminal



The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows the project structure under 'LILAC\_VENTURES\_WEBSITE' with files like Homepage.tsx, CalendarView.tsx, and example.tsx.
- Editor:** The file 'example.tsx' is open, displaying the following code:
 

```
1  /*
2   * This is a new file and does nothing.
3   */
4  export default{}
```
- Terminal:** The terminal tab is active, showing a stack trace for a GridFsStorage error. The '+' button in the terminal toolbar is highlighted with a red box.
 

```
at new GridFsStorage (G:\CSC191\lilac_ventures_website\node_modules\multer-gridfs-storage\lib\gridfs.js:71:19)
at Object.<anonymous> (G:\CSC191\lilac_ventures_website\api\middleware\gridFsSetup.ts:25:17)
at Module._compile (node:internal/modules/cjs/loader:1275:14)
at Module._extensions..js (node:internal/modules/cjs/loader:1329:10)
at Module.load (node:internal/modules/cjs/loader:1133:32)
at Module._load (node:internal/modules/cjs/loader:972:12)
at Module.require (node:internal/modules/cjs/loader:1157:19)
at require (node:internal/modules/helpers:119:18)
at Object.<anonymous> (G:\CSC191\lilac_ventures_website\api\server.ts:13:26)
at Module._compile (node:internal/modules/cjs/loader:1275:14)
```
- Status Bar:** Shows Node.js v19.8.1, PS G:\CSC191\lilac\_ventures\_website, and other system information.

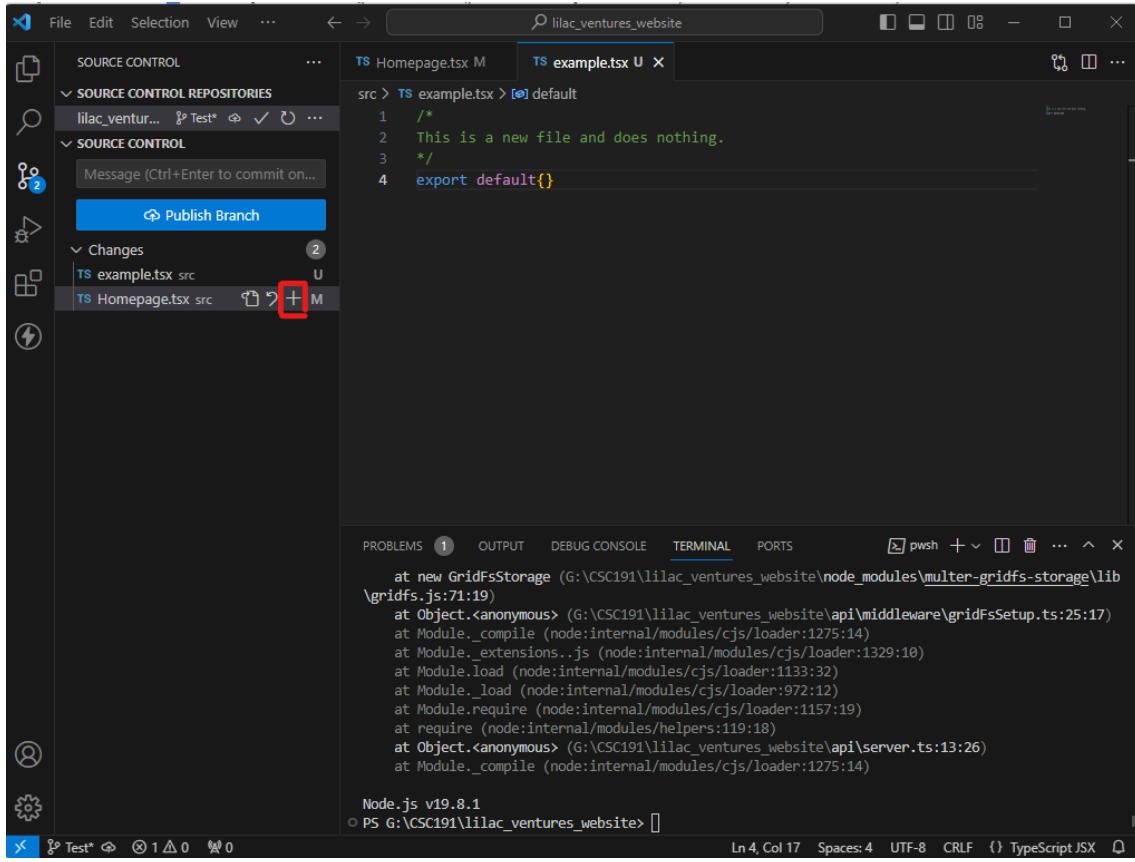
- In the new terminal, type `npm run server` and press enter. This will start a locally hosted server for API calls to the MongoDB database.

The screenshot shows the Microsoft Visual Studio Code interface. The code editor displays a file named 'example.tsx' with the following content:

```
src > TS example.tsx
1  /*
2  This is a new file and does nothing.
3  */
```

The terminal at the bottom shows the command `npm run server` entered and highlighted with a red box. The file explorer on the left shows a folder named 'LILAC\_VENTURES\_WEBSITE' containing numerous TypeScript files like 'Homepage.tsx', 'Contact.tsx', and 'Login.tsx'. The status bar at the bottom indicates the file is a TypeScript JSX file.

- j. Once a coding task has been completed and all changed or new files have been saved navigate to the Source Control tab. Click the + symbol next to each file name in the Changes section.



- k. Once the changes are staged, type a common message then click the Commit button.

The screenshot shows the Visual Studio Code interface with the following details:

- Source Control Sidebar:** On the left, under "SOURCE CONTROL REPOSITORIES", there is a repository named "lilac\_ventures\_website". Under "SOURCE CONTROL", there is a commit titled "Example Commit" which has been staged and is ready to be committed.
- Code Editor:** The main editor area displays a TypeScript file named "Homepage.tsx". The code includes imports for React, money, Link, useNavigate, HomepageView, and useAuth. It contains a comment explaining it's an edit that doesn't do anything. The function "Homepage" uses useNavigate to handle route changes.
- Terminal:** At the bottom, the terminal window shows the command "PS G:\CSC191\lilac\_ventures\_website> []" and the output "Node.js v19.8.1".
- Status Bar:** The status bar at the bottom indicates "Ln 10, Col 1" and "Spaces: 2".

- I. Click the Publish Branch button.

The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, ...
- Title Bar:** lilac\_ventures\_website
- Source Control Sidebar:**
  - SOURCE CONTROL
  - SOURCE CONTROL REPOSITORIES: lilac\_venture... (Test)
  - SOURCE CONTROL: Message (Ctrl+Enter to commit on...)
  - Publish Branch button (highlighted in blue)
- Code Editor:** Homepage.tsx
 

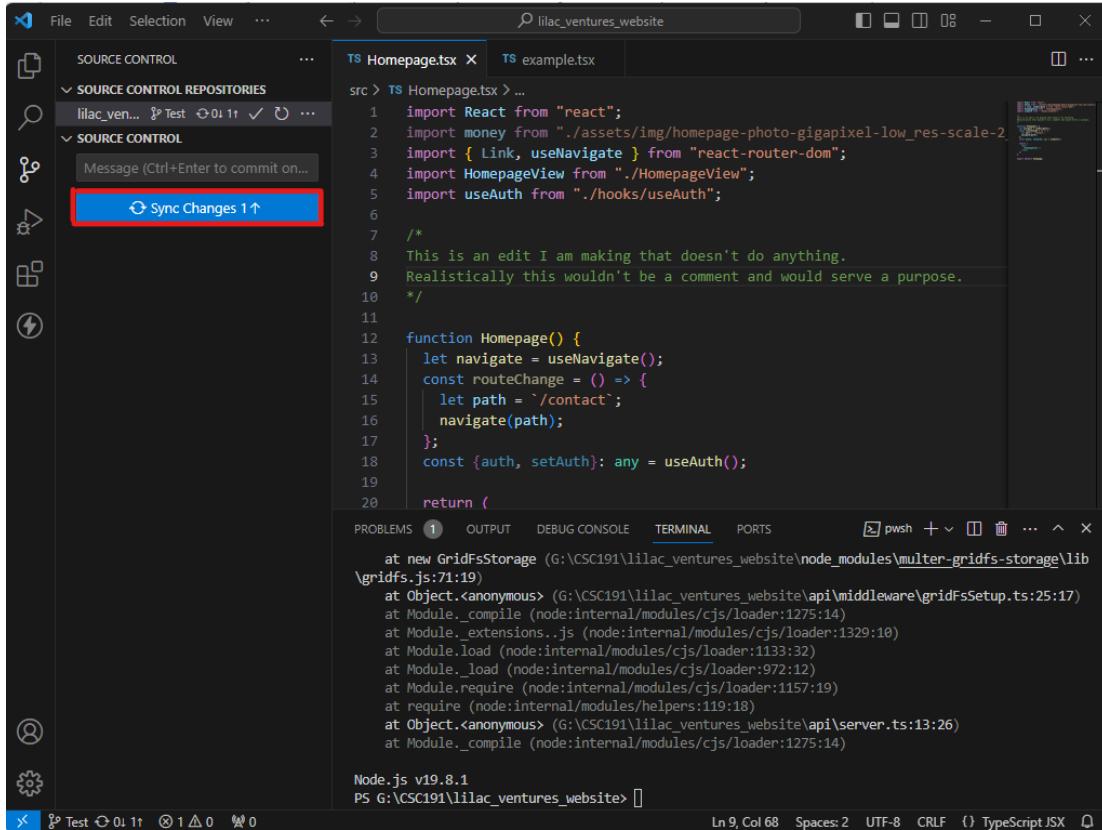
```

1 import React from "react";
2 import money from "./assets/img/homepage-photo-gigapixel-low_res-scale-2";
3 import { Link, useNavigate } from "react-router-dom";
4 import HomepageView from "./HomepageView";
5 import useAuth from "./hooks/useAuth";
6
7 /*
8 This is an edit I am making that doesn't do anything.
9 Realistically this wouldn't be a comment and would serve a purpose.
10 */
11
12 function Homepage() {
13   let navigate = useNavigate();
14   const routeChange = () => {
15     let path = '/contact';
16     navigate(path);
17   };
18   const {auth, setAuth}: any = useAuth();
19 }
```
- Terminal:** Node.js v19.8.1, PS G:\CSC191\lilac\_ventures\_website> []
 

```

at new GridFsStorage (G:\CSC191\lilac_ventures_website\node_modules\multer-gridfs-storage\lib\gridfs.js:7:19)
at Object.<anonymous> (G:\CSC191\lilac_ventures_website\api\middleware\gridFsSetup.ts:25:17)
at Module._compile (node:internal/modules/cjs/loader:1275:14)
at Module._extensions..js (node:internal/modules/cjs/loader:1329:10)
at Module.load (node:internal/modules/cjs/loader:1133:32)
at Module._load (node:internal/modules/cjs/loader:972:12)
at Module.require (node:internal/modules/cjs/loader:1157:19)
at require (node:internal/modules/helpers:119:18)
at Object.<anonymous> (G:\CSC191\lilac_ventures_website\api\server.ts:13:26)
at Module._compile (node:internal/modules/cjs/loader:1275:14)
```
- Status Bar:** Ln 10, Col 1, Spaces: 2, UTF-8, CRLF, {}, TypeScript JSX

- m. Repeat Steps e-f and j-k for each coding task. Additional Commits will only be synced with the repository once the Sync Branch button is pressed after commits are completed.



The screenshot shows a code editor interface with a dark theme. On the left, there's a sidebar with various icons for source control, file navigation, and search. The main area displays a TypeScript file named `Homepage.tsx`. The code is as follows:

```

1 import React from "react";
2 import money from "./assets/img/homepage-photo-gigapixel-low_res-scale-2";
3 import { Link, useNavigate } from "react-router-dom";
4 import HomepageView from "./HomepageView";
5 import useAuth from "./hooks/useAuth";
6
7 /*
8 This is an edit I am making that doesn't do anything.
9 Realistically this wouldn't be a comment and would serve a purpose.
*/
10
11 function Homepage() {
12   let navigate = useNavigate();
13   const routeChange = () => {
14     let path = `/contact`;
15     navigate(path);
16   };
17   const {auth, setAuth}: any = useAuth();
18
19   return (
20     ...

```

Below the code editor, the terminal window shows the command `node.js v19.8.1` and the path `PS G:\CSC191\lilac_ventures_website>`. The bottom status bar indicates the current line (Ln 9), column (Col 68), and encoding (UTF-8). The source control sidebar shows a repository named `lilac_ven...` with one commit pending.

- Once all changes are complete and thoroughly tested for bugs/errors, go to the GitHub repository in a web browser.

The screenshot shows a GitHub repository page for 'lilac\_ventures\_website'. The main branch is 'main'. The repository has 50 branches and 0 tags. The commit history shows several deployment changes, including one from 'LU-93: added jwt verification and token rotation' and another from 'LU-126 Finish Appointment Api Frontend integration'. The repository has 9 authors and 269 commits. It includes sections for About, Releases, Packages, Contributors, and Deployments.

**o. Change the branch from main to the new branch.**

The screenshot shows a GitHub repository page for 'lilac\_ventures\_website'. A dropdown menu in the top left corner is set to 'Test'. The repository has 50 branches and 0 tags. The commit history shows several deployment changes, including one from 'LU-93: added jwt verification and token rotation' and another from 'LU-126 Finish Appointment Api Frontend integration'. The repository has 9 authors and 269 commits. It includes sections for About, Releases, Packages, Contributors, and Deployments.

**p. Click on Compare and Pull Request**

The screenshot shows a GitHub repository page for 'lilac\_ventures\_website'. The 'Compare & pull request' button is highlighted in red. The page displays a list of commits from 'njcgamerplayer' and various repository details like releases, packages, contributors, and deployments.

- q. Change the destination branch from main to development. Then click Create Pull Request

The screenshot shows the 'Open a pull request' dialog box on GitHub. The 'base: main' dropdown is highlighted in red, indicating it's set to the 'main' branch. The 'compare: Test' dropdown is also highlighted in red. The dialog box contains fields for reviewers, assignees, labels, projects, and milestones, with a prominent 'Create pull request' button at the bottom.

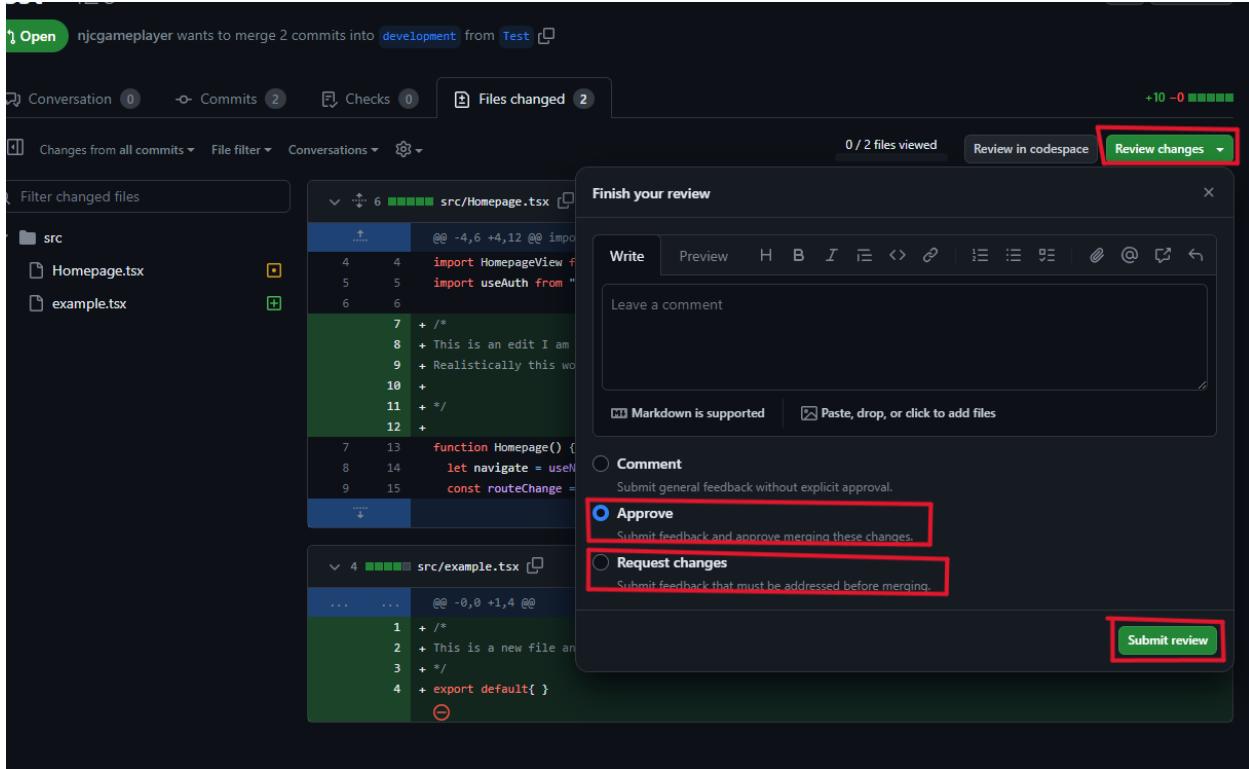
- r. A collaborator with writing privileges will be able to resolve any version conflicts created by merging the branches and then completing the merge. Afterwards, the pull request will be closed.

## Merging branches from pull request

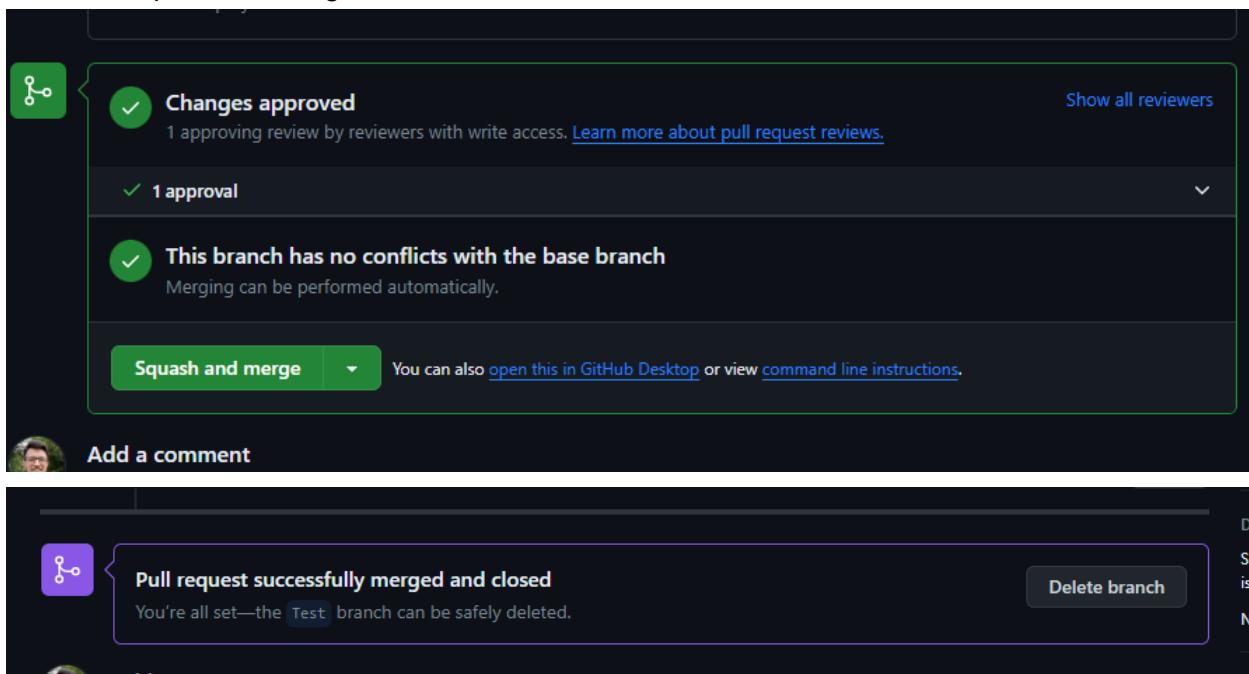
- a. As a collaborator with write privileges to the development branch, open the pull request

The screenshot shows a GitHub pull request interface. At the top, there's a navigation bar with filters for 'Open' (1) and 'Closed' (125). Below the navigation bar, a list item for 'Test' is shown, with a note that '#126 opened 16 hours ago by njcgameplayer • Review required'. The main content area is titled 'Test #126'. It includes a green 'Open' button and a message from 'njcgameplayer' wanting to merge two commits from the 'Test' branch into the 'development' branch. Below this, there are tabs for 'Conversation' (0), 'Commits' (2), 'Checks' (0), and 'Files changed' (2), with 'Files changed' highlighted with a red border. The commit list shows two commits from 'njcgameplayer' added 16 hours ago: 'Example Commit' (44d483b) and 'Example Commit' (62d2461). A note at the bottom says 'Add more commits by pushing to the [Test](#) branch on [davidenzler/lilac\\_ventures\\_website](#)'. Below the commit list, there are sections for deployment status ('This branch has not been deployed') and review requirements ('Review required' and 'Merging is blocked'). A checkbox for 'Merge without waiting for requirements to be met (bypass branch protections)' is present. At the bottom, there's a 'Squash and merge' button and a note about opening in GitHub Desktop or command line instructions. A 'Write' tab and a rich text editor toolbar are also visible.

- c. Review the changes and verify that the changes will not affect other functionality. Press the Review Changes button. If further changes need to be made, you can check the request changes box. Otherwise, the approve box can be checked. Press submit Review.

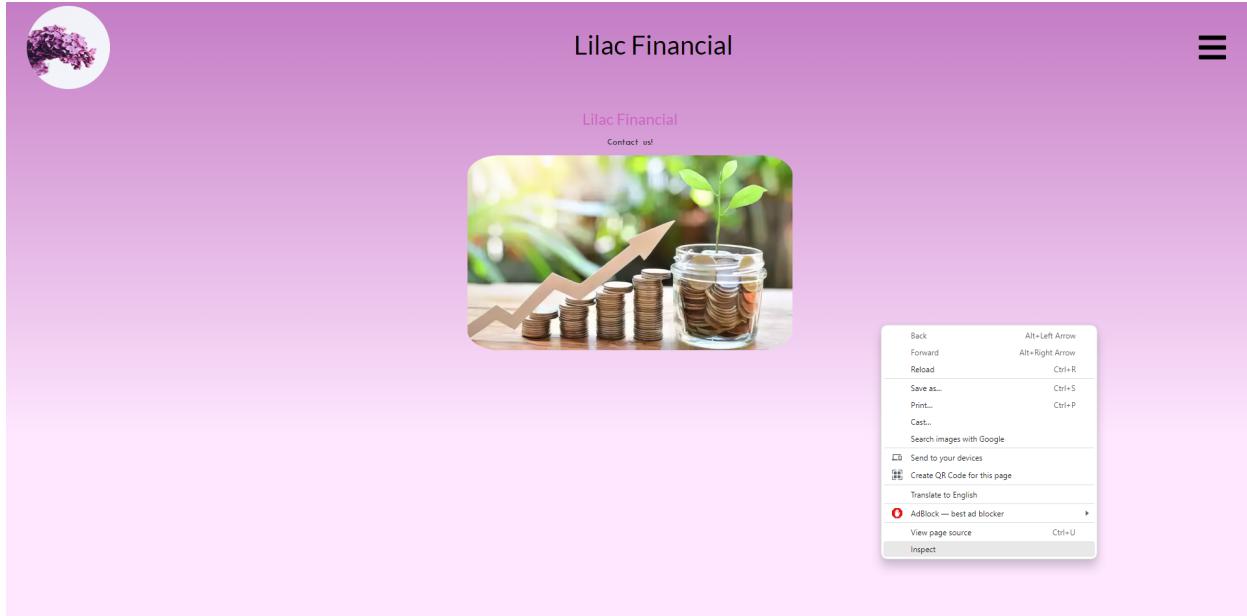


- d. Once the code is ready to be merged, press the squash and merge button. This will complete the merge.

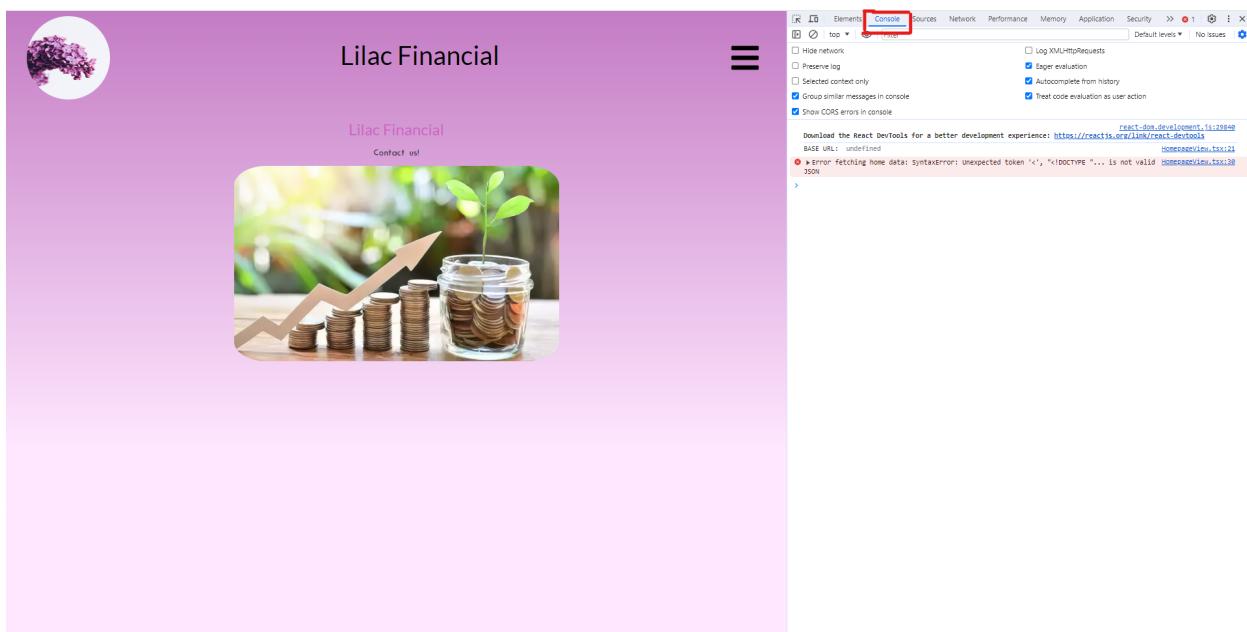


## How to check for errors in the code

- Errors typically are caught and logged into the browser's console. To access the console, with the site open, either press F12 or right click on the screen and select inspect



- Click the console tab



- Any errors or notable events will be sent to the console

Screenshot of the Chrome DevTools Console tab showing developer settings and a list of errors.

**Developer Settings:**

- Hide network
- Log XMLHttpRequests
- Preserve log
- Eager evaluation
- Selected context only
- Autocomplete from history
- Group similar messages in console
- Treat code evaluation as user action
- Show CORS errors in console

**Console Errors:**

- at Router (<http://localhost:3000/static/js/bundle.js:197262:15>)
  - at BrowserRouter (<http://localhost:3000/static/js/bundle.js:195377:5>)
    - at AuthProvider (<http://localhost:3000/static/js/bundle.js:12243:3>)
- ✖ ▶ Warning: validateDOMNesting(...): <tr> cannot appear as a child of <table>. Add [react-dom.development.js:86](#) a <tbody>, <thead> or <tfoot> to your code to match the DOM tree generated by the browser.
  - at tr
  - at ToggleItem (<http://localhost:3000/static/js/bundle.js:510:3>)
  - at table
  - at section
  - at AdminOverview (<http://localhost:3000/static/js/bundle.js:629:100>)
  - at RenderedRoute (<http://localhost:3000/static/js/bundle.js:196707:5>)
  - at Outlet (<http://localhost:3000/static/js/bundle.js:197241:26>)
  - at section
  - at section
  - at AdminComponent (<http://localhost:3000/static/js/bundle.js:1043:81>)
  - at RenderedRoute (<http://localhost:3000/static/js/bundle.js:196707:5>)
  - at Outlet (<http://localhost:3000/static/js/bundle.js:197241:26>)
  - at ProtectedRoute (<http://localhost:3000/static/js/bundle.js:10831:3>)
  - at RenderedRoute (<http://localhost:3000/static/js/bundle.js:196707:5>)
  - at Routes (<http://localhost:3000/static/js/bundle.js:197324:5>)
  - at div
  - at App (<http://localhost:3000/static/js/bundle.js:1318:66>)
  - at RenderedRoute (<http://localhost:3000/static/js/bundle.js:196707:5>)
  - at Routes (<http://localhost:3000/static/js/bundle.js:197324:5>)
  - at Router (<http://localhost:3000/static/js/bundle.js:197262:15>)
  - at BrowserRouter (<http://localhost:3000/static/js/bundle.js:195377:5>)
  - at AuthProvider (<http://localhost:3000/static/js/bundle.js:12243:3>)
- ✖ ▶ POST <http://localhost:3000/undefined/customerProgress/getCustomersAtStep> 404 (Not [ProgressService.tsx:4](#) [Found](#))
- ✖ ▶ POST <http://localhost:3000/undefined/customerProgress/getCustomersAtStep> 404 (Not [ProgressService.tsx:4](#) [Found](#))
- ✖ ▶ Uncaught (in promise) TypeError: Cannot read properties of undefined (reading ['length'](#))
  - at fetchData ([AdminDashboard.tsx:86:1](#))
- ✖ ▶ POST <http://localhost:3000/undefined/customerProgress/getCustomersAtStep> 404 (Not [ProgressService.tsx:4](#) [Found](#))
- ✖ ▶ POST <http://localhost:3000/undefined/customerProgress/getCustomersAtStep> 404 (Not [ProgressService.tsx:4](#) [Found](#))
- ✖ ▶ POST <http://localhost:3000/undefined/customerProgress/getCustomersAtStep> 404 (Not [ProgressService.tsx:4](#) [Found](#))
- ✖ ▶ POST <http://localhost:3000/undefined/customerProgress/getCustomersAtStep> 404 (Not [ProgressService.tsx:4](#) [Found](#))
- ✖ ▶ POST <http://localhost:3000/undefined/customerProgress/getCustomersAtStep> 404 (Not [ProgressService.tsx:4](#) [Found](#))
- ✖ ▶ POST <http://localhost:3000/undefined/customerProgress/getCustomersAtStep> 404 (Not [ProgressService.tsx:4](#) [Found](#))
- ✖ ▶ Uncaught (in promise) TypeError: Cannot read properties of undefined (reading ['length'](#))
  - at fetchData ([AdminDashboard.tsx:86:1](#))
- ✖ ▶ GET <http://127.0.0.1:8080/files/NewPDFUploads/step/1> [pdfUploadsService.tsx:16](#) [net::ERR\\_CONNECTION\\_REFUSED](#)

## 4.0 Deployment

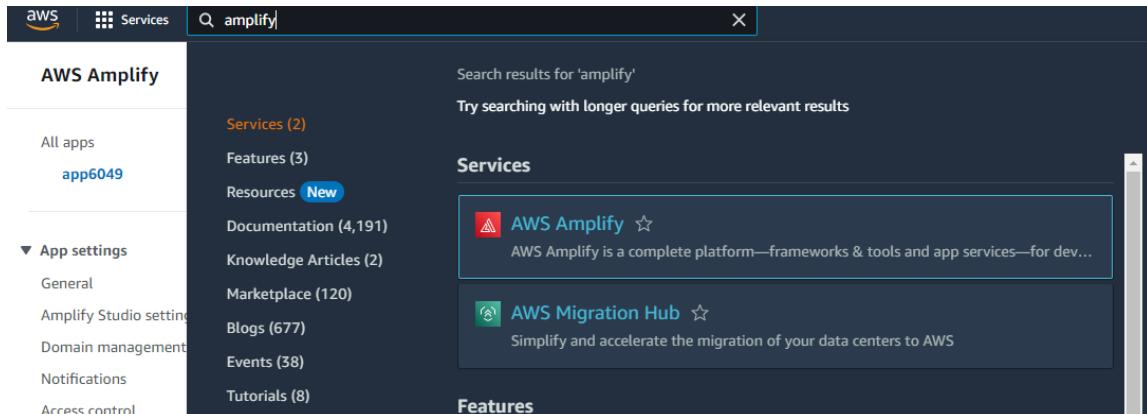
Describe how your application is currently deployed

### Overview

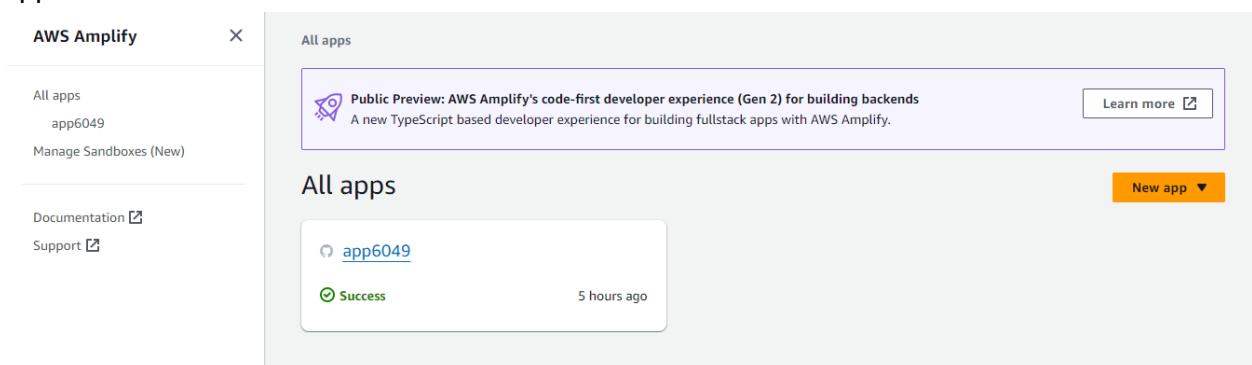
The application is currently deployed using AWS. The front end and back end are both hosted separately. The front-end React code is hosted using AWS Amplify. The backend NodeJS is hosted using AWS Elastic Beanstalk.

### Amplify

1. Login into the AWS console using the AWS credentials.
2. Search the navigation bar for “Amplify”. Click on “AWS Amplify”.



3. All currently running instances will appear in the console. Click on the running application.



4. The console for the application lists the health of the app, the domain of the app, and the current deployment status.

The screenshot shows the AWS Amplify console for the app app6049. On the left, there's a sidebar with 'All apps' (app6049 selected), 'App settings' (General, Amplify Studio settings, Domain management, Notifications, Access control, Monitoring, Rewrites and redirects, Custom headers), and links to Documentation and Support.

The main area is titled 'app6049' and shows the 'Hosting environments' tab selected. It says 'The app homepage lists all deployed frontend and backend environments.' Below this is a table for the 'dev3273' branch:

Deployment successfully completed.	Domain	Last deployment
	<a href="https://www.lilacfinancials.com">https://www.lilacfinancials.com</a>	11/30/2023, 5:48:57 PM

At the bottom, there's a section for dragging and dropping build files or a zip file, and a 'Choose files' button.

5. In the bottom of the screen, the build files for React can be dragged and dropped or uploaded from a local machine. Amplify will ingest the build files and automatically deploy them after a couple minutes.
6. The domain name and certs are managed from the “Domain Management” option in the left panel

The screenshot shows the 'Domain management' section of the AWS Amplify console. The sidebar is identical to the previous screenshot. The main area has a heading 'Domain management' and a sub-section 'Custom domain: www.lilacfinancials.com'.

Domain	Status
www.lilacfinancials.com	Available

Below this is another section for 'amplifyapp.com':

Domain	Status
amplifyapp.com	Available

7. The certs for this domain will automatically update, however, if there are routing issues, then the certs should be checked.

## Building React Files for Deployment

1. In a Vscode terminal (or terminal at the root directory of the project), run the command `npm run build`

```
PS C:\Users\David\Documents\SacState\Spring 2023\190_senior\source\repos\lilac_ventures_website> npm run build

> my-app@0.1.0 build
> react-scripts build

Creating an optimized production build...

```

2. The script will take a couple minutes to complete

```
The build folder is ready to be deployed.
You may serve it with a static server:

  serve -s build

Find out more about deployment here:

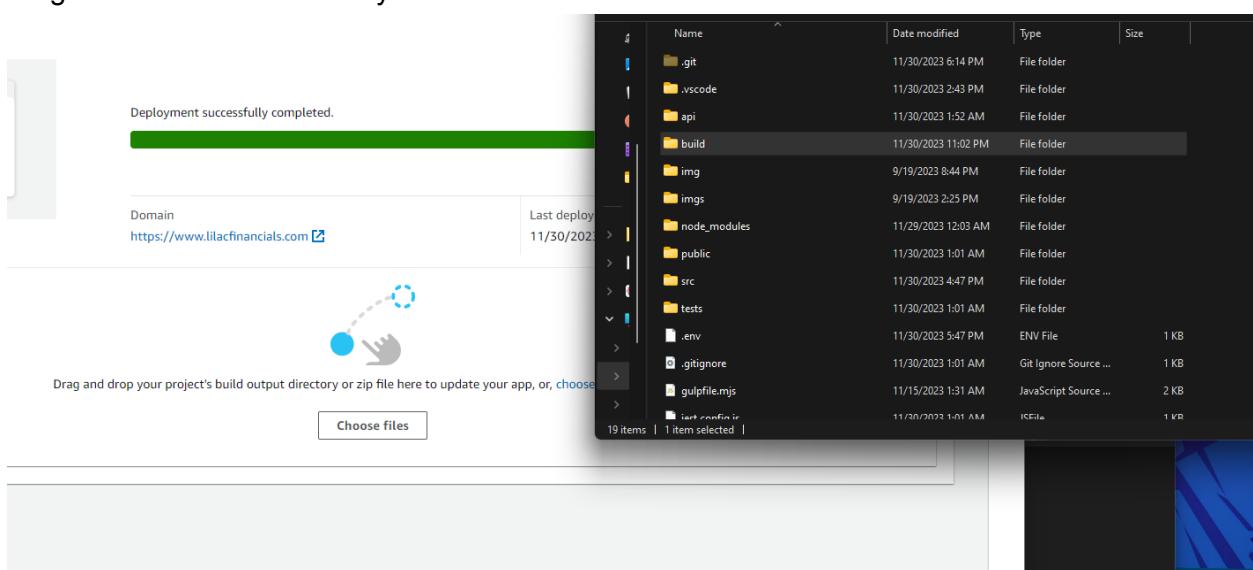
  https://cra.link/deployment

PS C:\Users\David\Documents\SacState\Spring 2023\190_senior\source\repos\lilac_ventures_website>
```

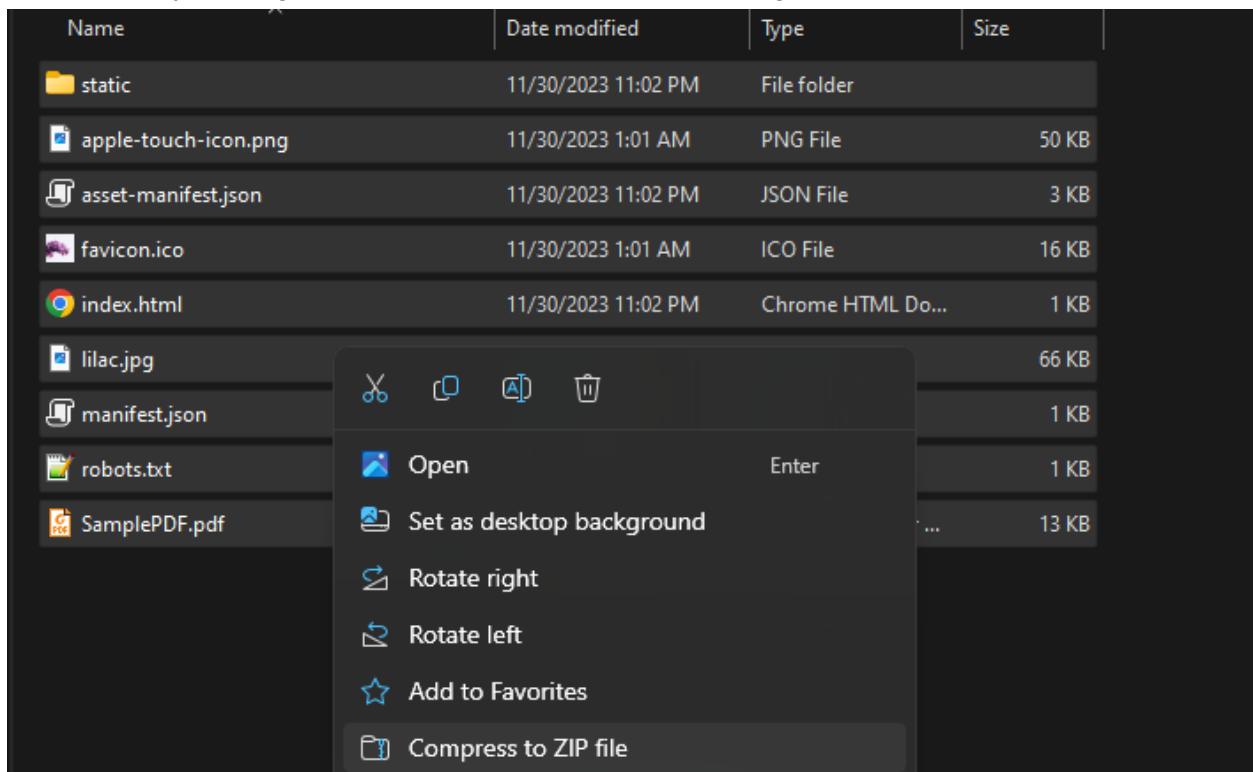
3. When it is finished, the project root will have a new build directory. Upload this build directory to Amplify to deploy a new version of the site.



4. There are two options for uploading the code: either upload a .zip file or drag the whole build directory onto the console screen.
5. Drag the whole build directory to the console window



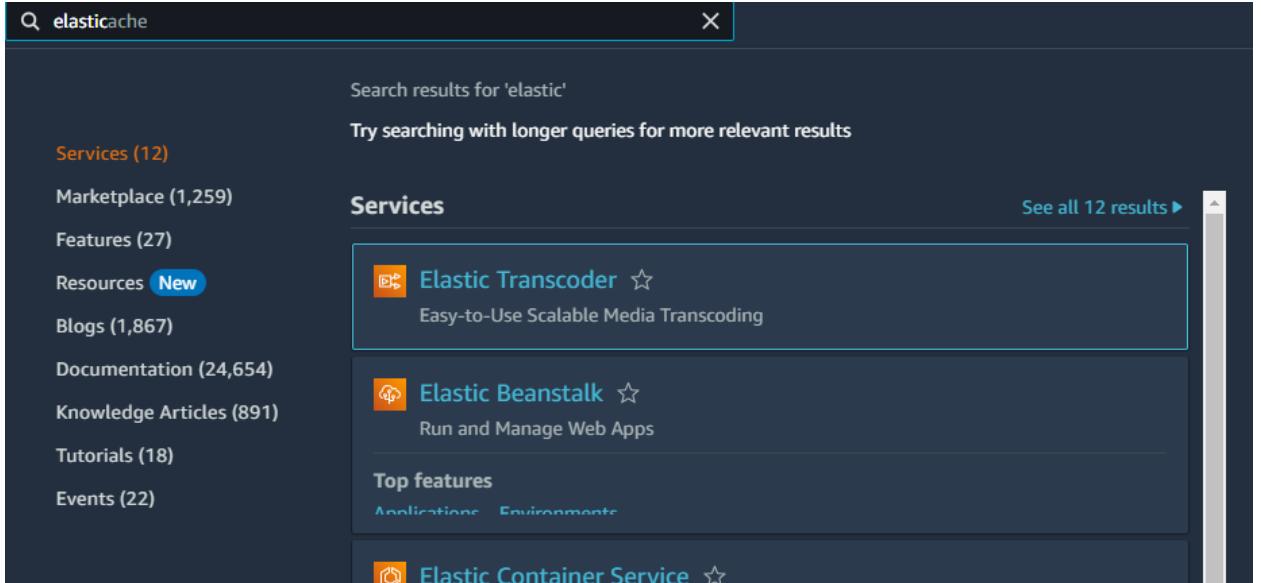
6. Alternatively, select the contents of the build directory and add to a zip archive. Upload this .zip file by clicking the ‘Choose File’ button and selecting the zip file.



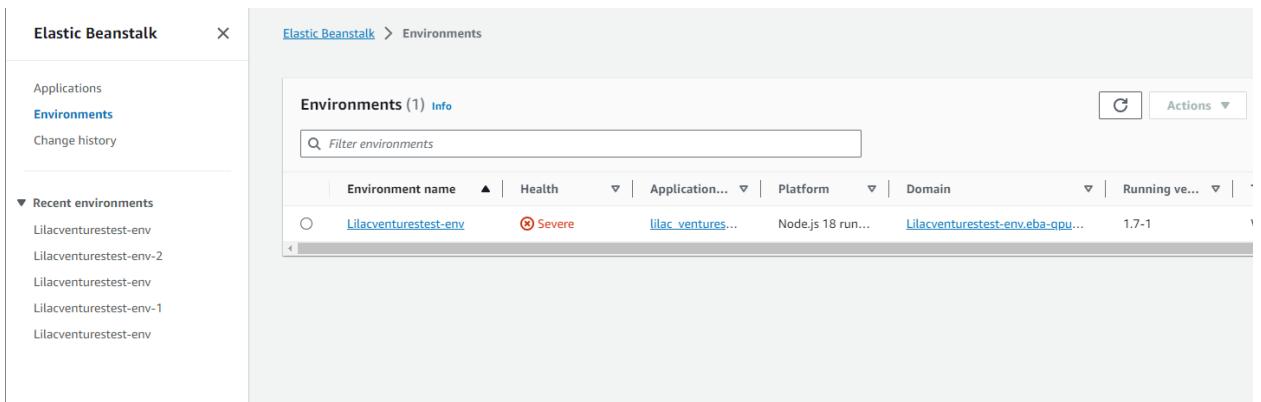
7. AWS has ample documentation on running applications on Amplify. For more information check out the link
- <https://docs.aws.amazon.com/amplify/>

## Elastic Beanstalk

- Log in to the AWS console with the AWS credentials.
- Navigate to the ELB (Elastic Beanstalk) console by searching for ‘Elastic’ in the search menu and selecting ‘Elastic Beanstalk’

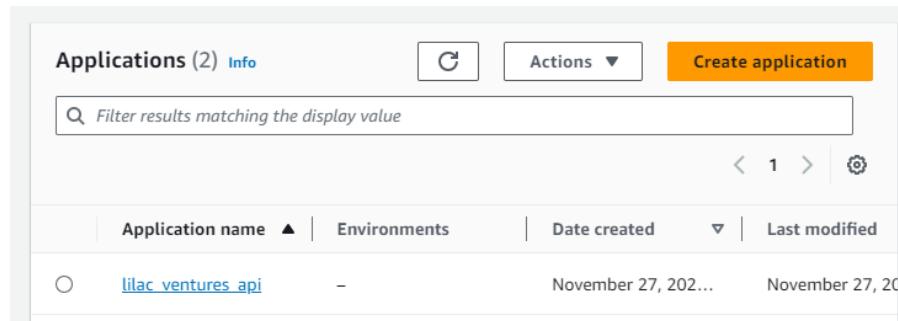


3. The link will drop you into an overview of either the Application or the Environment. ( The severe warning can be ignored in this example. The endpoints used for Health Checks was disabled at the time of this screenshot).



#### 4. A Note on Application vs Environment

- Application: The NodeJS source code. Anything referring to application will refer to the code deployed by ELB. Applications can be individually configured and assigned resources.



- Environment: This is the set of resources deployed by ELB to host the Application. Depending on the configuration, Beanstalk will stand up one or more

EC2 instances and a load balancer. The configuration for this site uses a single EC2 instance and a load balancer. This provides a cost effect solution while allowing for scaling as the site grows.

The screenshot shows the AWS Lambda Environments console. At the top, there's a header with 'Environments (1) Info' and a 'Create environment' button. Below the header is a search bar labeled 'Filter environments'. Underneath the search bar is a table with columns: Environment name, Health, Application..., and Platform. The table contains one row for 'Lilaventurestest-env', which is marked as 'Severe' with a red cross icon. The application name is 'lilac\_ventures...' and the platform is 'Node.js 18 run...'. There are navigation arrows and a settings gear icon at the bottom right of the table area.

## Deploying NodeJS Code to Existing Environment

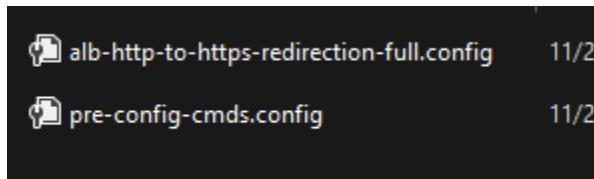
1. The following files will need to be zipped into an archive:
  - a. The contents of `./api/`
  - b. The package.json
  - c. The tsconfig.json
  - d. The .ebextensions folder and its contents
2. IMPORTANT: The structure of the zipped archive matters. Failing to follow the correct structure may result in a failed deployment
3. The following image is of the unzipped directory with all the appropriate files and directories

Name	Date modified	Type	Size
.ebextensions	11/30/2023 5:34 PM	File folder	
config	11/30/2023 5:33 PM	File folder	
controllers	11/30/2023 5:33 PM	File folder	
middleware	11/30/2023 5:33 PM	File folder	
model	11/30/2023 5:33 PM	File folder	
routes	11/30/2023 5:33 PM	File folder	
package.json	11/30/2023 5:35 PM	JSON File	3 KB
server.ts	11/30/2023 5:30 PM	VLC.ts	3 KB
tsconfig.json	11/30/2023 1:01 AM	JSON File	1 KB

4. WARNING: `package.json` must be reconfigured for the ELB deployment process. Please ensure that the `scripts` field of package.json contains a start script with the following format

```
},
"scripts": {
  "start": "node ./api/server.ts",
},
"build": {
  "analyze": true
}
```

5. Depending on the development environment, the package.json may need to be modified before it is zipped.
6. Important: please make sure that the .ebextensions folder has the following files:



7. These are used to configure SSL and HTTP redirects in new environments.
8. Once all the files are in place, select all the files and directories and zip them together.

A screenshot of a file explorer context menu. The menu items include: Open, Open in new window, Pin to Quick access, Compress to ZIP file (which is highlighted), Copy as path, Properties, PowerRename, and Open in Terminal. Below the menu, there is a list of files: 'tsconfig.json' (modified 11/30/2023 1:01 AM, JSON File) and 'prod-build.zip' (modified 11/30/2023 11:36 PM, Compressed (zipp...)).

9. In the AWS Beanstalk console, navigate to the application overview

The screenshot shows the 'Applications' page with a sidebar on the left containing links for 'Applications', 'Environments', and 'Change history'. Below this is a section titled 'Recent environments' listing 'Lilacventurestest-env' and 'Lilacventurestest-env-2'. The main area displays a table with columns for 'Application name', 'Environments', 'Date created', and 'Last modified'. A single row is visible for 'lilac\_ventures\_api'.

Application name	Environments	Date created	Last modified
<a href="#">lilac_ventures_api</a>	-	November 27, 202...	November 27, 20...

10. Select the application and select the 'Actions' dropdown menu. Select 'Application versions'

The screenshot shows the details for the 'lilac\_ventures\_api' application. The 'Actions' dropdown menu is open, showing options: 'Create environment', 'Delete application', 'View application versions', 'View saved configurations', and 'Restore terminated environment'. Below the application details, there are buttons for 'Create environment', 'Delete application', 'View application versions', and 'View saved configurations'.

11. In the following screen, select 'Upload', give the new code base an updated version number, choose the zip file from the previous step, and hit upload.

**Upload application version**

Version label  
Unique name for this version of your application code.

Description

Upload application  
  
File name: **prod-build.zip**  
File must be less than 500MB max file size

Tags — *optional*  
A tag is a key-value pair that you assign to an AWS resource.  
No tags associated with the resource.

You can add 50 more tags.

12. In the versions console, select the recently uploaded version and then select 'Actions', and then 'Deploy'.

	Version label	Description
<input checked="" type="checkbox"/>	1.9	—
<input type="checkbox"/>	1.7-1	—

	<b>Actions</b> ▲	Settings
	Delete	
	<b>Deploy</b>	
	Manage tags	

13. Select the currently used environment from the drop down and select deploy. The deployment will take several minutes to complete. To view the status and get updates, go to the Environments console from the left pane and select the current environment

Applications

Environments

Change history

▼ Application: [lilac\\_ventures\\_test](#)

Environment name

Environment name
<a href="#">Lilacventurestest-env</a>

## 5.0 User interaction

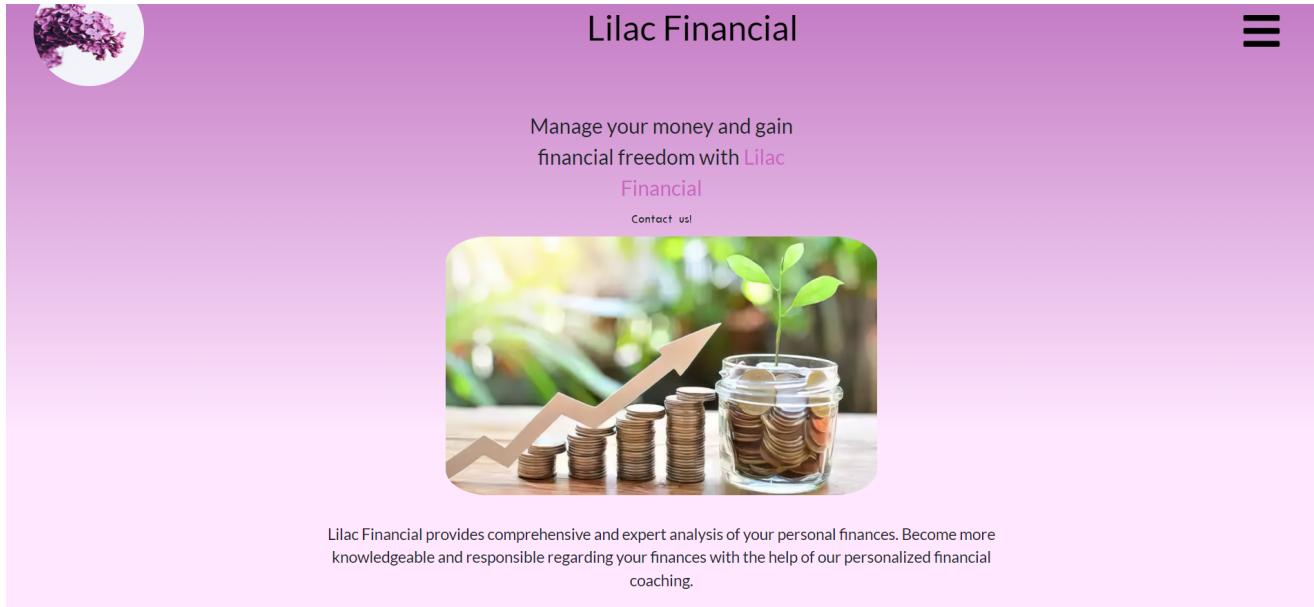
---

Description: Home Screen

What types of users see the screen (admin, customer, etc): Admin and customer

Related Code Files: HomePage.tsx, HomePageView.tsx, HomePageView.css

Screenshot:



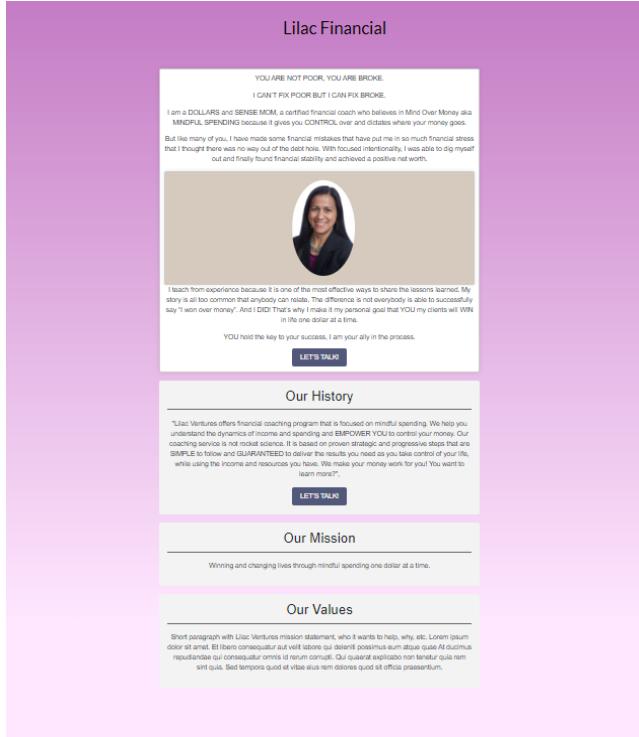
---

Description: About Screen

What types of users see the screen (admin, customer, etc): Admin and customer

Related Code Files: About.tsx, AboutView.tsx, About.css

Screenshot:

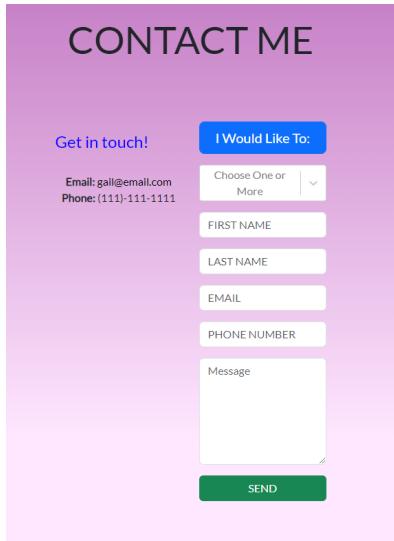


## Description: Contact Screen

What types of users see the screen (admin, customer, etc): Customer

Related Code Files: Contact.tsx, ContactView.tsx

Screenshot:

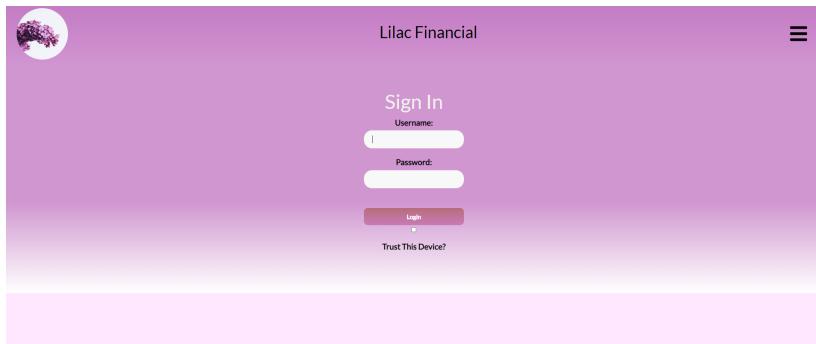


## Description: Login Screen

What types of users see the screen (admin, customer, etc): Customer

Related Code Files: LoginButton.tsx, LogoutButton.tsx, Login.tsx, Login.css, AuthProvider.tsx, logoutService.tsx

Screenshot:

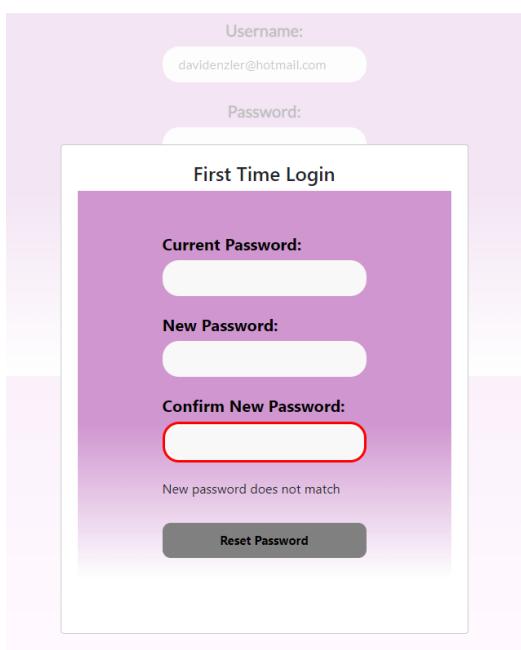


Description: First Time Login Screen

What types of users see the screen (admin, customer, etc): Customer

Related Code Files: FirstTimeLoginForm.css, FirstTimeLoginForm.tsx, FirstTimeLoginModal.tsx

Screenshot:

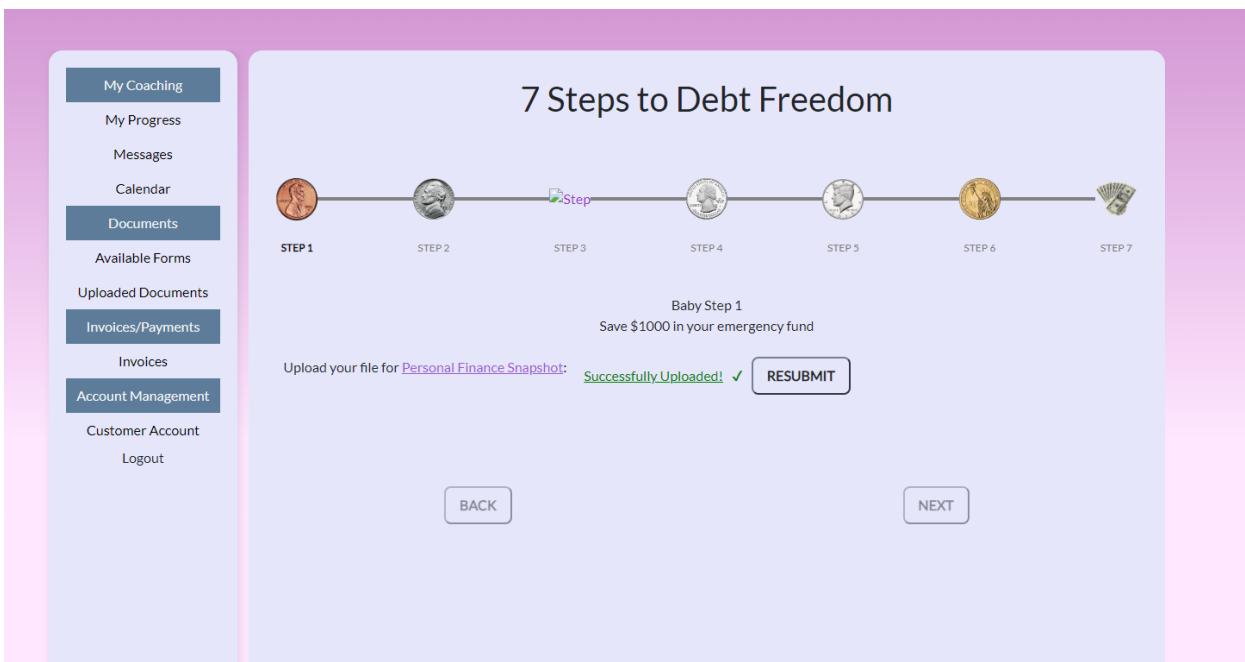


Description: Customer Portal Screen

What types of users see the screen (admin, customer, etc): Customer

Related Code Files: CustomerPortal.tsx, customerPortal.css

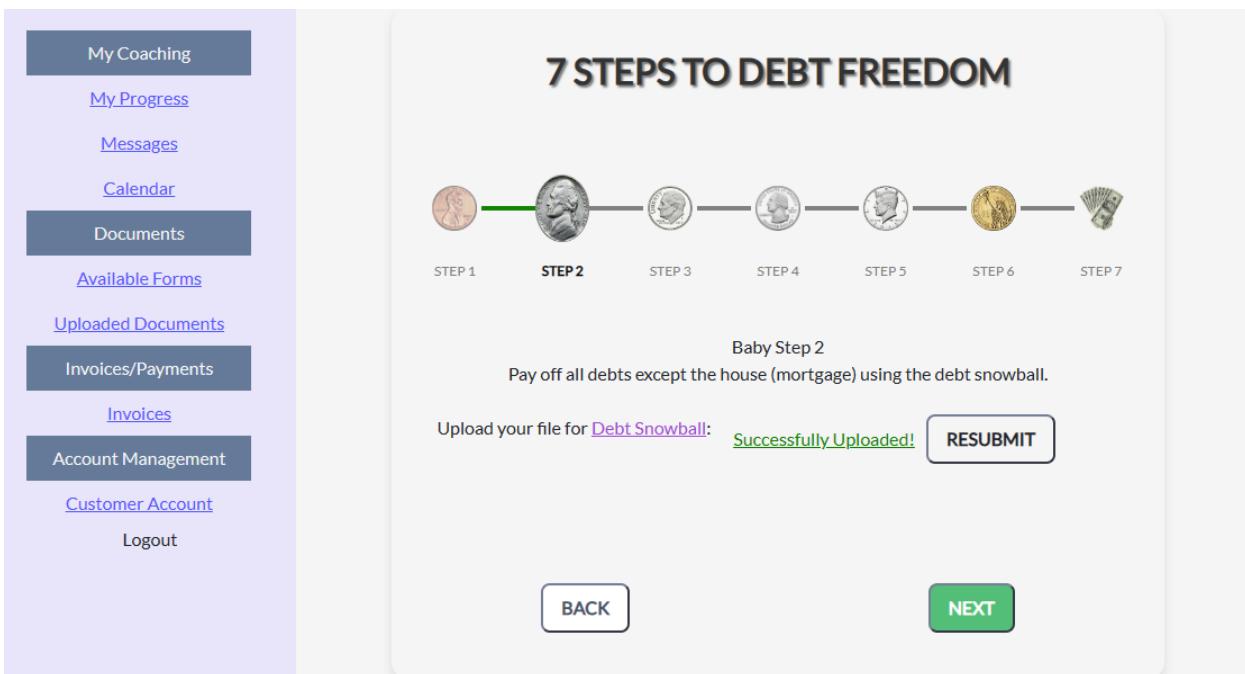
## Screenshot:



### Description: My Progress Screen

What types of users see the screen (admin, customer, etc.): Customer

Related Code Files: ProgressBar.tsx and all files under the folder “progress-bar-components”  
Screenshot:

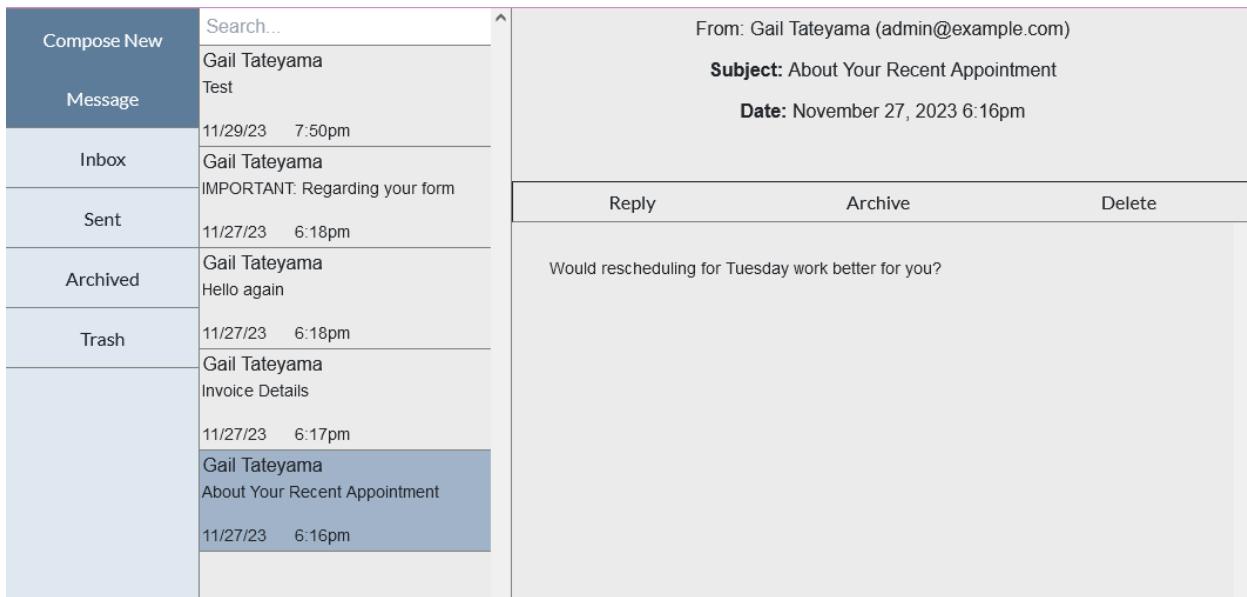


### Description: Messages Screen

What types of users see the screen (admin, customer, etc.): Customers and admin

Related Code Files: Inbox.tsx, Inbox.css

Screenshot:

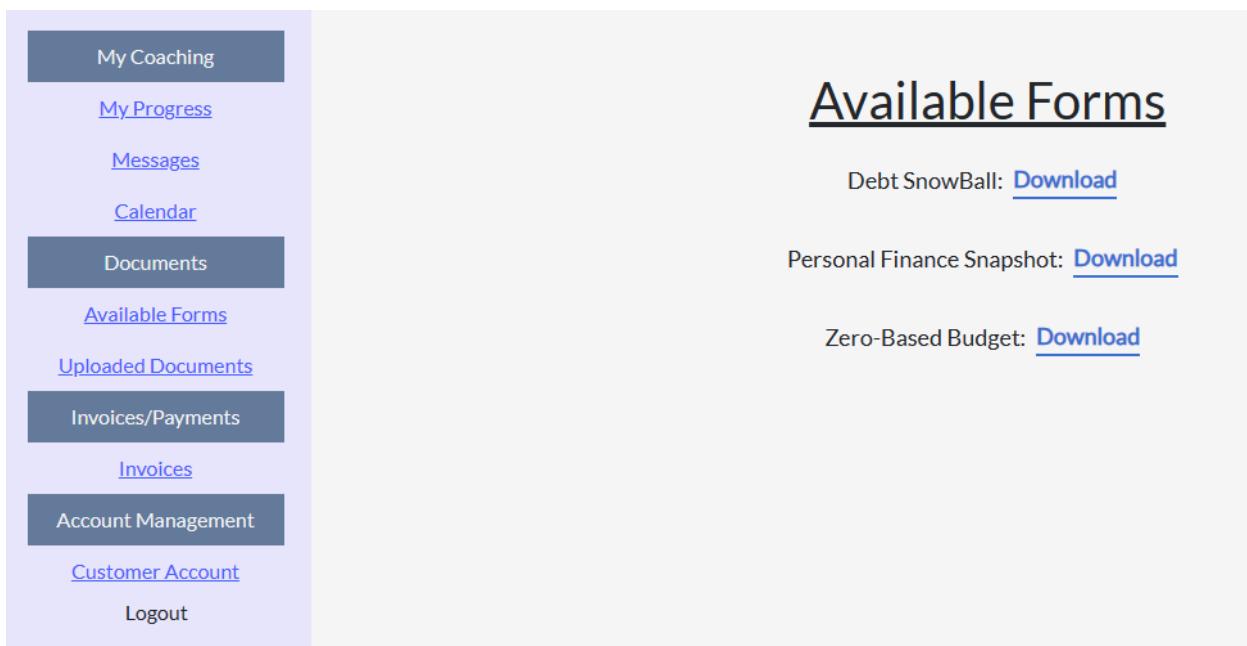


Description: Available Forms Screen

What types of users see the screen (admin, customer, etc): Customer

Related Code Files: AvailableForm.tsx

Screenshot:

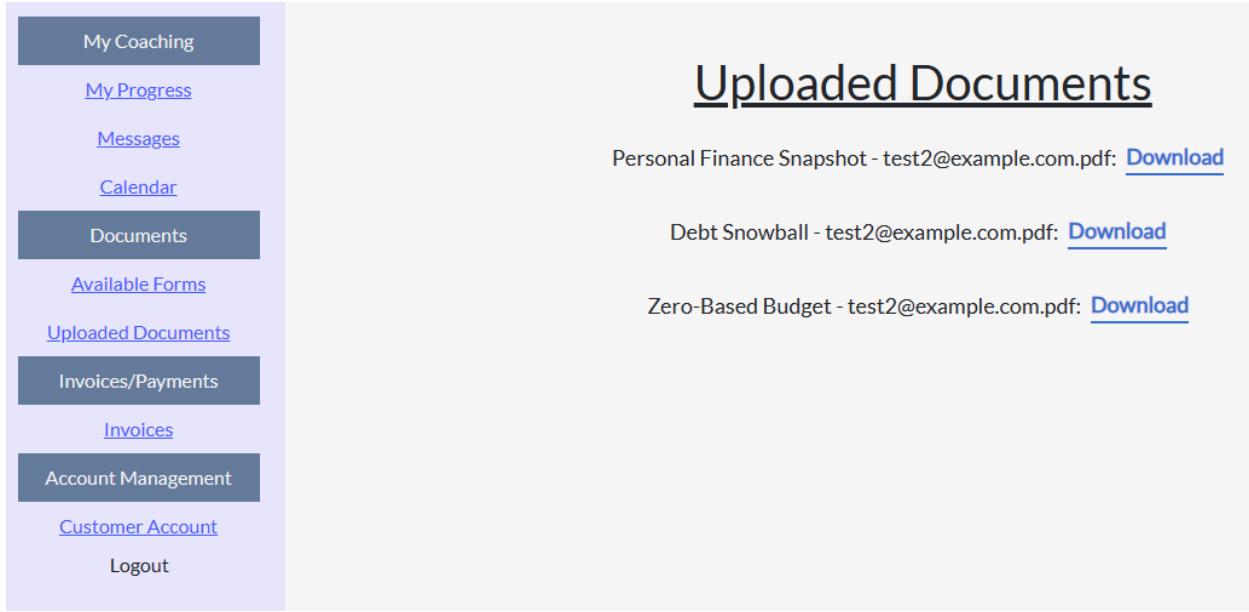


Description: Uploaded Documents Screen

What types of users see the screen (admin, customer, etc): Customer

Related Code Files: UploadedDocuments.tsx

Screenshot:



Description: Customer Account Screen

What types of users see the screen (admin, customer, etc): customer

Related Code Files:

Screenshot:

Description: Invoices Screen

What types of users see the screen (admin, customer, etc):

1. Customer

Related Code Files:

1. ./src/CustomerPortalInvoices/CustomerInvoices.css
2. ./src/CustomerPortalInvoices/CustomerInvoiceComponent.tsx
3. ./src/services/invoiceServices.tsx

Screenshot:

Invoices			
Amount Due	Due Date	Pay	Download PDF
\$5000	11/30/2023, 12:00:00 AM	<button>Pay</button>	<button>Download</button>
\$1000	11/30/2023, 12:00:00 AM	<button>Pay</button>	<button>Download</button>

Description: Admin Portal/ Admin Overview Screen

What types of users see the screen (admin, customer, etc):

1. Admin

Related Code Files:

1. /src/AdminDashboard/AdminDashboard.tsx
2. /src/AdminDashboard/adminDash.css
3. /src/services/pdfUploadServices.tsx
4. /src/services/ProgressService.tsx

Screenshot:

The screenshot shows a sidebar menu on the left with the following items: Overview (selected), Create Invoice, Customer List (selected), Register Client, Messages, Calendar, Edit Page, and Logout. The main content area is titled "Client Progress Overview" and displays a table with seven rows, each representing a step with a progress count and a downward arrow icon.

Step	Count
Step 1	5 ▽
Step 2	1 ▽
Step 3	1 ▽
Step 4	1 ▽
Step 5	0 ▽
Step 6	0 ▽
Step 7	0 ▽

Client Progress Overview			
Step 1	5 △		
	Username	FirstName	LastName
	johndoe@example.com	John	Doe
	flob@example.com	Rick	McRick
	jsmith@email.com	john	smith
	jake@email.com	Jake	State
	davidenzler@hotmail.com	David	Enzler

Step 2	1 ▽
--------	-----

Description: Create Invoice Screen

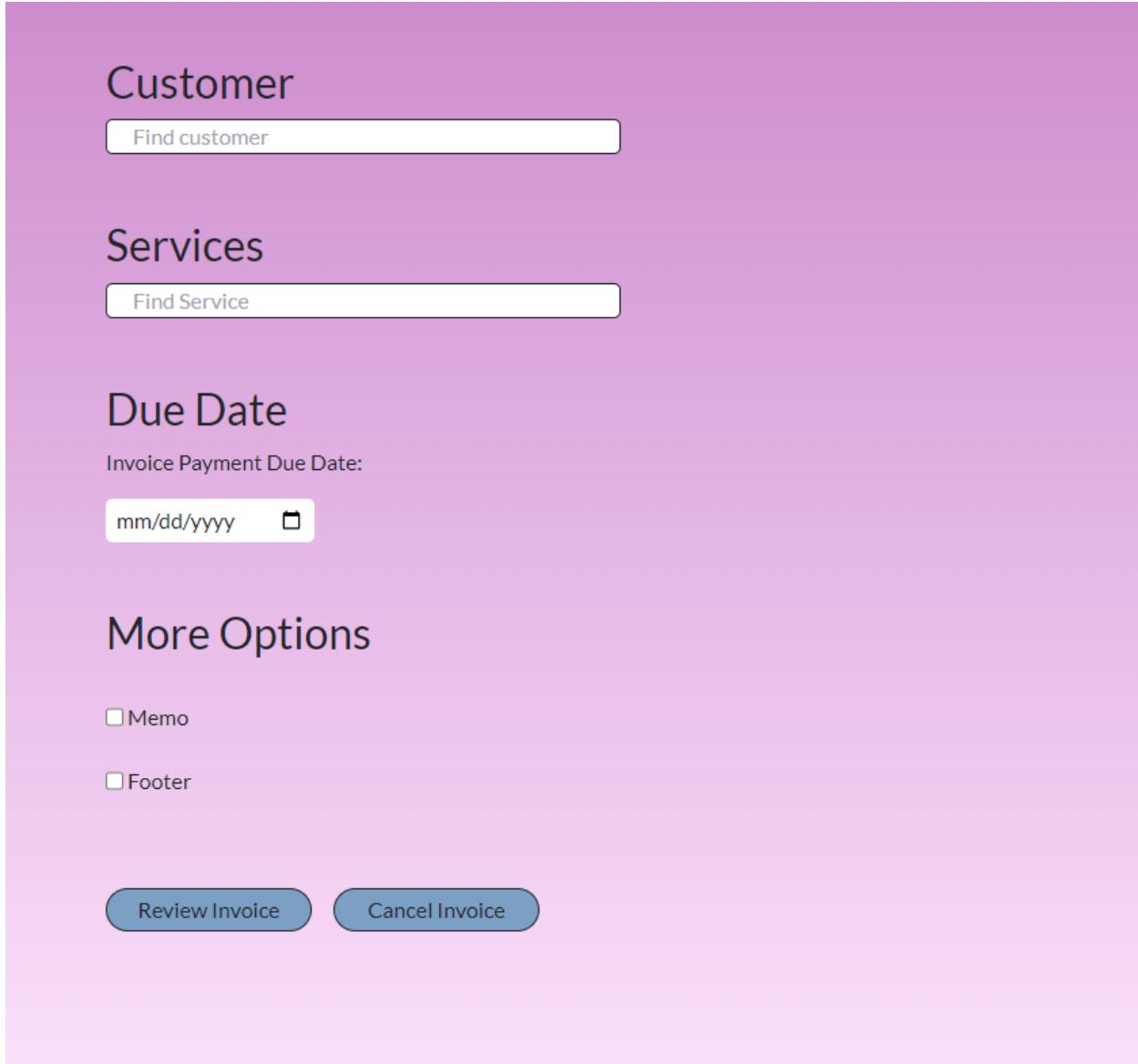
What types of users see the screen (admin, customer, etc):

1. Admin

Related Code Files:

1. /src/InvoiceCreation/CreateInvoice.tsx
2. /src/InvoiceCreation/Invoice.css
3. /src/services/customerServices.tsx
4. /src/services/productSearch.tsx
5. /src/services/invoiceServices.tsx

Screenshot:



---

Description: Customer List Screen

What types of users see the screen (admin, customer, etc): Admin

Related Code Files: /src/CustomerInfoView.tsx

Screenshot:

Current Clients								
First Name	Last Name	Address	Mobile Number	Email	Marital Status	Employment Status	Progress Step	
John	Doe	123 Main St, Anytown, CA 90001	1234567890	johndoe@example.com			1	 
Rick	McRick	123 Main St, Anytown, WA 90001	1234567890	flob@example.com			1	 
Test	Test-1	Incomplete Address		test@example.com			2	 
test2	test2Last	Incomplete Address		test2@example.com			1	 
	test3Last	Incomplete Address	93839070372	test3@example.com			3	 
john	smith	akmaja, chicago, california 797987	93839070372	jsmith@email.com		Employed	1	 
Jake	State	1500 street road, Folsom, California 95630	9168889208	jake@email.com		Employed	1	 
John	Smith	1234 Street way, Rancho Cordova, California 95630	916555555	jsmith@yahoo.com	Married	Single	1	 
Brock	Smith	123 Street way, Rancho cordova, California 95670	8378973827	bsmith@email.com	Married	Employed	1	 
Paul	Dean	13333 road street, sacramento, California 95670	873837894	pdean@email.com	Single	Employed	1	 
test	test	123 something street, city, state 12345	1234567890	testtest@example.com	married	employed	1	 
Gail	Tateyama	123 main street, my city, ca 11007	9163353390	gail.liasophia1@gmail.com	single	employed full time	1	 
David	Enzler	123 Street, Sacramento, California 12345	1234567890	davidenzler@hotmail.com	Married	Employed	1	 

### Description: Register Client Screen

What types of users see the screen (admin, customer, etc):

1. Admin

Related Code Files:

1. /src/RegistrationForm.tsx
2. /src/RegistrationForm.css
3. /src/services/RegisterService.tsx
4. /src/services/ClientService.tsx

Screenshot:

## CREATE USER FORM

First Name  Last Name

Email Address  Phone Number

Address  
Street

City  State (Please write whole state name)

Zip Code  Please fill out this field.

Marital Status  Employment Status

Contact Preference  
Please type Email or Phone

### Password Generator

Length:

Symbols  Numbers  LowerCase  UpperCase

**Generate Password**

**CREATE**

Description: Availability Screen

What types of users see the screen (admin, customer, etc):

1. Admin

Related Code Files:

1. Front End:
  - a. /src/AvailabilityCalendar.tsx
  - b. /src/CalendarComponents/calendar.tsx

- c. /src/CalendarComponents/calendar.css
  - d. /src/CalendarComponents/cn.tsx
  - e. /hooks/useAuth.tsx
2. API Calls:
- a. /api/models/Availability.ts
  - b. /api/routes/availability.ts
  - c. /api/availabilityController.ts

Screenshot:

November 2023							
	S	M	T	W	T	F	S
Overview	29	30	31	1	2	3	4
Create Invoice	5	6	7	8	9	10	11
Customer List	12	13	14	15	16	17	18
Register Client	19	20	<u>21</u>	<u>22</u>	<u>23</u>	<u>24</u>	<u>25</u>
Messages	26	27	<b>28</b>	<b>29</b>	30	1	2
Calendar	3	4	5	6	7	8	9
Availability							

Availability for Wed Nov 29 2023  
9:00 AM-2:45 PM  
[Set Availability](#)

---

Description: Calendar Screen

What types of users see the screen (admin, customer, etc):

1. Admin
2. Customer

Related Code Files:

1. Front End:
  - a. /src/CalendarView.tsx
  - b. /src/CalendarComponents/calendar.tsx
  - c. /src/CalendarComponents/calendar.css
  - d. /src/CalendarComponents/cn.tsx
  - e. /hooks/useAuth.tsx
2. API Calls:
  - a. /api/models/Appointments.ts
  - b. /api/routes/appointments.ts
  - c. /api/controllers/appointmentController.ts
  - d. /api/models/Availability.ts
  - e. /api/routes/availability.ts
  - f. /api/availabilityController.ts
  - g. /api/models/Message.ts

- h. /api/routes/messages.ts
- i. /api/messagesController.ts

Screenshot:

1. Admin

November 2023						
S	M	T	W	T	F	S
29	30	31	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	1	2
3	4	5	6	7	8	9

2. Customer

November 2023						
S	M	T	W	T	F	S
29	30	31	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	1	2
3	4	5	6	7	8	9

Description: Edit Page Screen

What types of users see the screen (admin, customer, etc): Admin

Related Code Files:

1. Front End:

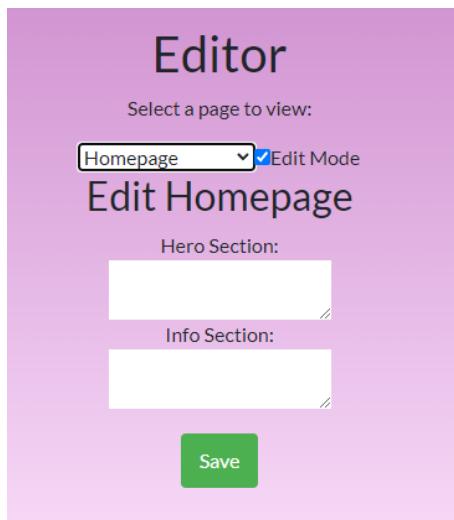
- a. /src/Edit/AboutEdit.tsx
- b. /src/Edit/ContactEdit.tsx
- c. /src/Edit/HomepageEdit.tsx
- d. /src/Edit/style\_edit.css

2. API Calls:

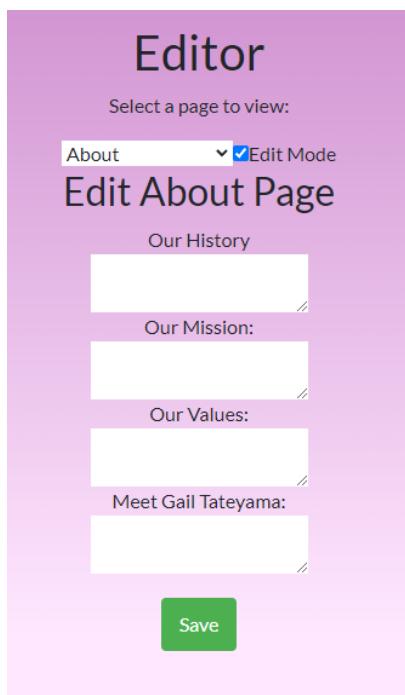
- a. /api/controllers/aboutController.ts
- b. /api/controllers/contactController.ts
- c. /abi/controllers/homeController.ts
- d. /api/model/About.ts
- e. /api/model/Contact.ts
- f. /api/model/Home.ts
- g. /api/routes/about.ts
- h. /api/routes/contact.ts
- i. /api/routes/home.ts

Screenshot:

1. HomepageEdit



2. AboutEdit



## 3. ContactEdit

The screenshot shows a user interface titled "Editor" with a pink header. Below the header, a dropdown menu says "Select a page to view:" followed by "Contact" and a checked "Edit Mode" checkbox. The main content area is titled "Edit Contact Page". It contains three input fields: "Call to Action" (empty), "Email" (empty), and "Phone Number" (empty). A green "Save" button is located at the bottom.

Select a page to view:

Contact  Edit Mode

## Edit Contact Page

Call to Action:

Email:

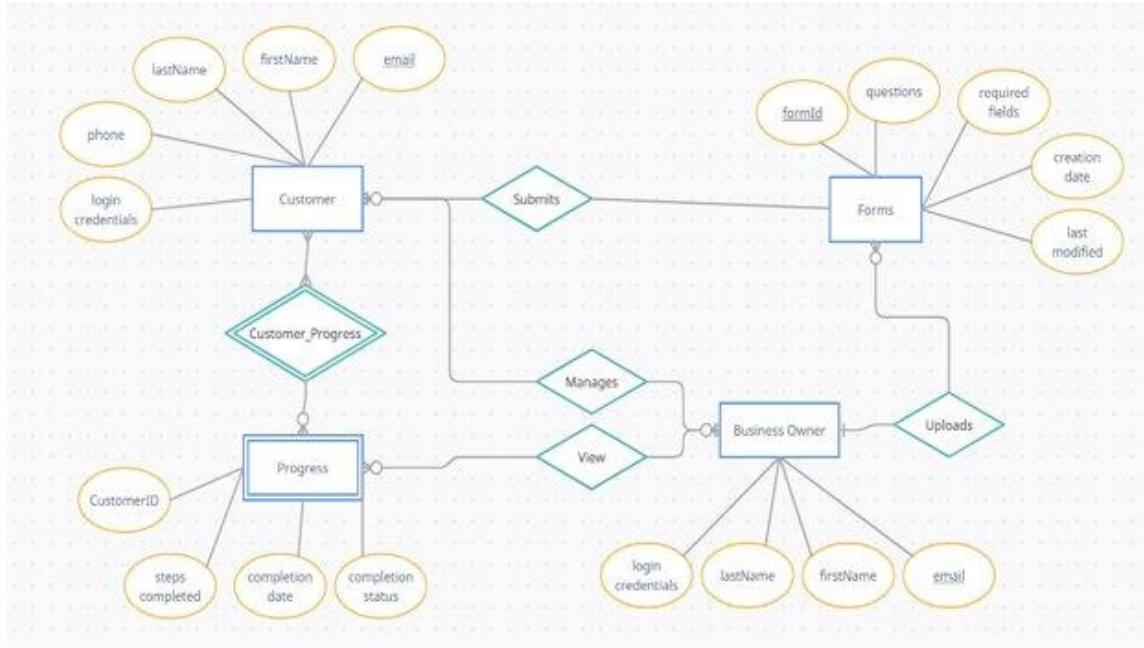
Phone Number:

Save

## 6.0 Database

---

- MongoDB - NoSQL database



Structure for various database items:

```
_id: ObjectId('655a80637faccdb50bb79bf')
firstName: "john"
lastName: "smith"
email: "jsmith@email.com"
phone: 93839070372
street: "akmaja"
city: "chicago"
state: "california"
zip: 797987
employment: "Employed"
cPreference: "Email"
__v: 0
```

## test.appointments

Documents Aggregations Schema Indexes Validation

Filter  Type a query: { field: 'value' }

 ADD DATA

 EXPORT DATA

```
_id: ObjectId('656951807e19d1bb0c6f20d3')
date: "Fri Dec 01 2023"
time: "3:30 PM"
user: "davidenzler@hotmail.com"
duration: 30
__v: 0
```

## test.availabilities

Documents Aggregations Schema Indexes Validation

Filter  Type a query: { field: 'value' }

 ADD DATA ▾

 EXPORT DATA ▾

```
_id: ObjectId('65680bcdef17b28046866e8a')
date: "Wed Nov 29 2023"
▶ time: Array (2)
  __v: 0
```

```
_id: ObjectId('65680f87adef33080f20b3d3')
date: "Fri Dec 01 2023"
▶ time: Array (4)
  __v: 0
```

```
_id: ObjectId('656847f382d2e7ab82570c04')
date: "Thu Nov 30 2023"
▶ time: Array (2)
  __v: 0
```

```
_id: ObjectId('65684a69b380faf7c81172d0')
date: "Tue Nov 28 2023"
▶ time: Array (2)
  __v: 0
```

## test.messages

Documents Aggregations Schema Indexes Validation

Filter 



Type a query: { field: 'value' }

 ADD DATA

 EXPORT DATA

```
_id: ObjectId('6524b31393427c1cab107746')
sender: "johndoe@example.com"
receiver: "flob@example.com"
subject: "Test Subject"
content: "Test content"
isArchivedBySender: false
isArchivedByReceiver: false
isReadBySender: false
isReadByReceiver: false
isDeletedBySender: false
isDeletedByReceiver: false
timestamp: 2023-10-10T02:12:35.374+00:00
__v: 0
```

```
_id: ObjectId('652a00bdb3abcddeb3a9c4e8')
sender: "flob@example.com"
receiver: "johndoe@example.com"
subject: "New Message"
content: "This is a new message"
isArchivedBySender: false
isArchivedByReceiver: false
isReadBySender: false
isReadByReceiver: false
isDeletedBySender: false
isDeletedByReceiver: false
timestamp: 2023-10-14T02:45:17.304+00:00
__v: 0
```

## test.pdfs

Documents Aggregations Schema Indexes Validation

Filter ⚙️ ⏱️ Type a query: { field: 'value' }

**ADD DATA** **EXPORT DATA**

```
_id: ObjectId('6521d7f41102d760313d7dd7')
fileId: ObjectId('6521d7f41102d760313d7dd5')
name: "Form A - JohnDoe.pdf"
seenByAdmin: false
__v: 0
```

- Viewer and Editor relationship

