

Introduction

This is the write up for Sup Invoice - an invoicing app designed for freelance creatives. It's simple and helps them get the job done and pay their bills.

We've tested this app with two users:

- one: a designer
- two: a video editor

The app is named 'Sup Invoice' as its a domain name that was short, cheap and a memorable word. People use the word sup to say hello, which is friendly and its short for super.

[T3A2-B-server Repo](#) [T3A2-B-front-end Repo](#) [T3A2-A Repo](#)

R1

At a minimum use the following technologies: MongoDB, Express, React & Node

Front-end technologies. Mainly React, with Tailwind & ShadCN component libraries.

```
"dependencies": {  
  "@headlessui/react": "^2.1.2",  
  "@heroicons/react": "^2.1.5",  
  "@radix-ui/react-dropdown-menu": "^2.1.1",  
  "@radix-ui/react-label": "^2.1.0",  
  "@radix-ui/react-scroll-area": "^1.1.0",  
  "@radix-ui/react-select": "^2.1.1",  
  "@radix-ui/react-slot": "^1.1.0",  
  "@radix-ui/react-switch": "^1.1.0",  
  "@radix-ui/react-tabs": "^1.1.0",  
  "class-variance-authority": "^0.7.0",  
  "clsx": "^2.1.1",  
  "framer-motion": "^11.3.20",  
  "lucide-react": "^0.418.0",  
  "react": "^18.3.1",  
  "react-dom": "^18.3.1",  
  "react-router-dom": "^6.26.0",  
  "tailwind-merge": "^2.4.0",  
  "tailwindcss-animate": "^1.0.7"  
},
```

Back-end technologies. Mainly MongoDB (with Mongoose), Express and Node (paired with Nodemon) to run everything.

```
"dependencies": {
  "bcrypt": "^5.1.1",
  "cookie-parser": "^1.4.6",
  "cors": "^2.8.5",
  "dotenv": "^16.4.5",
  "express": "^4.19.2",
  "jsonwebtoken": "^9.0.2",
  "mongodb": "^6.8.0",
  "mongodb-memory-server": "^10.0.0",
  "mongoose": "^8.5.1"
},
```

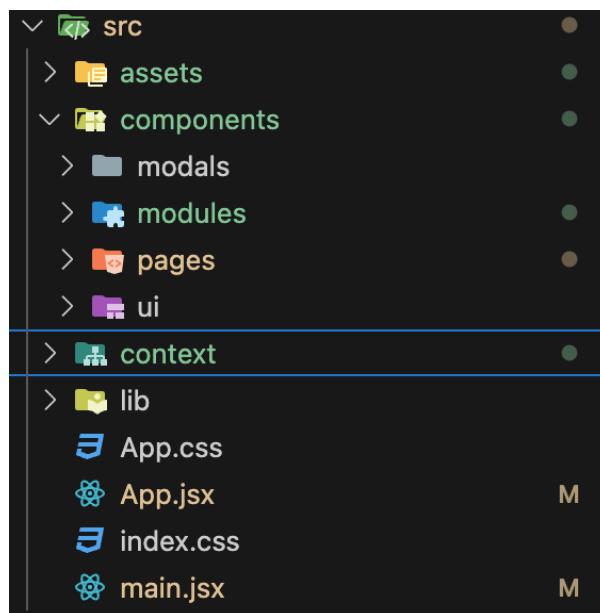
The rest can be seen in package.json files.

R2

Write well designed code that:

1. Separates the program into modules that each deal with one particular focus, or concern
2. Demonstrates DRY (Don't Repeat Yourself) coding principles

On the front-end my src folder houses all of the assets, components, context and lib files. Each file has its own focus and concern, allowing them to be easily applied and reused.



Example one

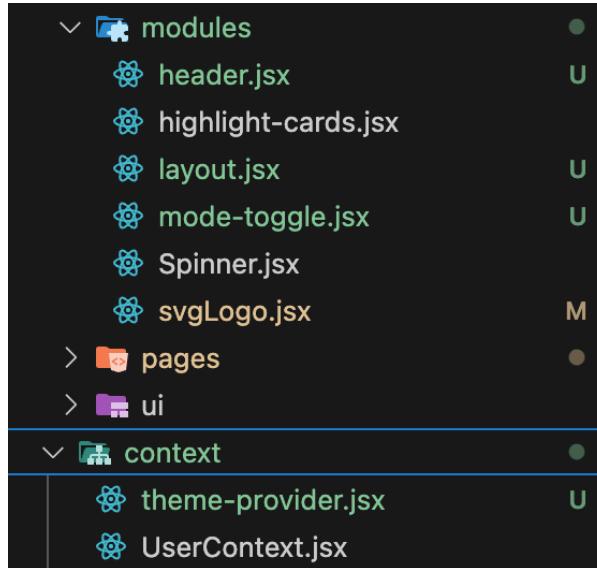
- o The header has a mode toggle which toggles dark and light mode - this means its only written once, which is super DRY. The theme is built into context, which provides it to the whole site. This allows for extremely swift and worry free developing. This makes updating my shadCN components a breeze.

Example two

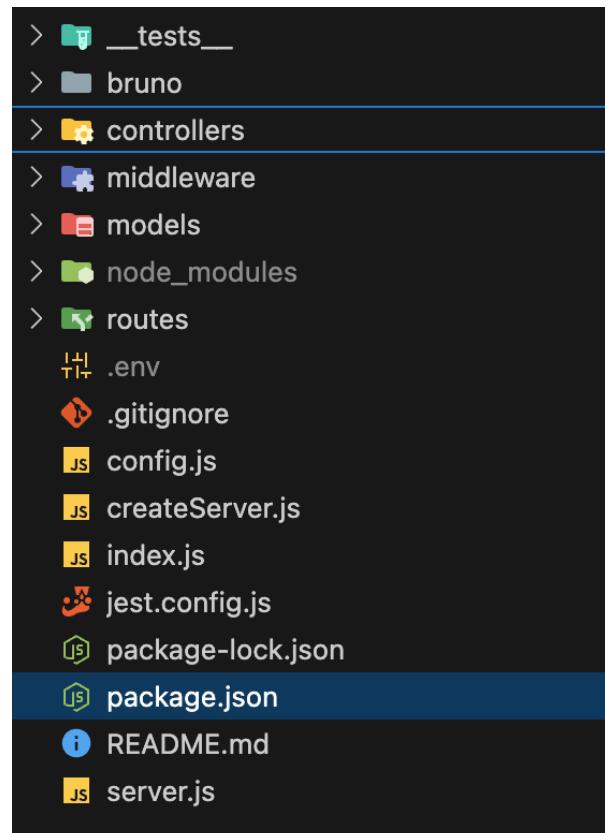
- The loading spinner is used throughout the site for loading components. Its one component which is in one file, but its called upon in many ways.

Example three

- UserContext supplies the cookies to authorise access throughout the site.



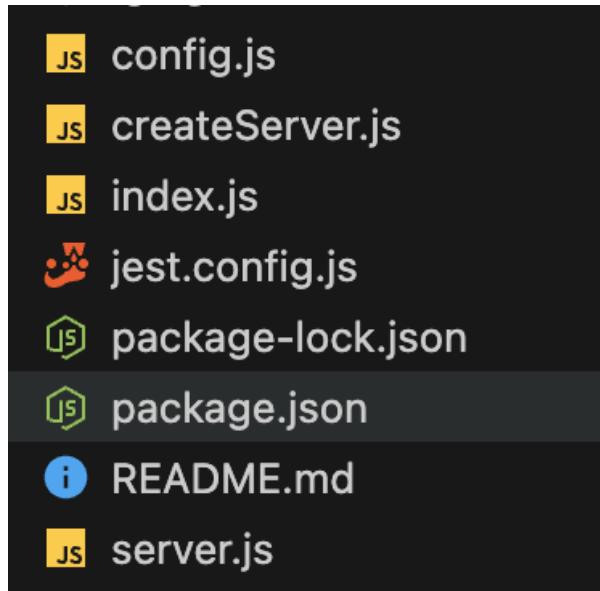
On the back-end the folders are organised into tests, bruno(API client) controllers, middleware, models and routes. Each file has its own focus and concern, allowing them to be easily applied and reused.



Example one

- Auth Middleware - on developing the backend, the auth checker for cookies, was copy and pasted many times. This lead to it being converted into middleware, meaning it was only

written once and could be passed to any route. This made the code very DRY.



Example two

- For testing purposes - the server was modularized. This meant the database could be detached and a mock database could provide tests for the routes. It's also made it incredibly neat.

Example three

- The router files became very clean and tidy, this allowed making new ones to be extremely swift and easy.

```
1 const express = require('express');
2 const router = express.Router();
3 const authController = require('../controllers/authController');

4

5 // routes are created to make it super neat
6 router.get('/', authController.home);
7 router.post('/register', authController.register);
8 router.post('/login', authController.login);
10 router.get('/profile', authController.profile);
11 router.post('/logout', authController.logout);
12
13 module.exports = router;
```

Uses appropriate libraries

Front-End Dependencies

- @headlessui/react: ^2.1.2 - really great UI library
- @heroicons/react: ^2.1.5 - really great icon library
- @radix-ui/react-dropdown-menu: ^2.1.1 - dropdown is a menu function used in the invoice creation
- @radix-ui/react-label: ^2.1.0 - react labels are for making labels
- @radix-ui/react-scroll-area: ^1.1.0 - this has been used in the drop down and on mobile with the invoice maker
- @radix-ui/react-select: ^2.1.1 - this allows us to select elements like drop downs.

- @radix-ui/react-slot: ^1.1.0 - allows us to pass elements to a component
- @radix-ui/react-switch: ^1.1.0 - the switch has been used to toggle
- @radix-ui/react-tabs: ^1.1.0 - react tabs has been used on the dashboard and on the login page
- class-variance-authority: ^0.7.0 - this allows control of the classes in CSS sheets
- clsx: ^2.1.1
- file-saver: ^2.0.5 - this was used to develop the screenshoting and file saving function
- html2canvas: ^1.4.1 - this was used to develop the screenshoting and file saving function
- lucide-react: ^0.418.0 - really great UI library
- react: ^18.3.1 - the great front end javascript library
- react-dom: ^18.3.1 - this allows us to change pages in the front end
- react-router-dom: ^6.26.0 - this allows us to change pages in the front end
- tailwind-merge: ^2.4.0 - this is the front end front end CSS replacement. It's great, I love it
- tailwindcss-animate: ^1.0.7 - this is the front end front end CSS replacement. It's great, I love it. The animate was used for the buttons and loading animations

Front-End Dev Dependencies

- tailwindcss: ^3.4.7 - this is the front end front end CSS replacement. It's great, I love it
- vite: ^5.3.4 - this app was first started and run on vite

Back-End Dependencies

- bcrypt: ^5.1.1 - this does all the auth/encryption for our users
- cookie-parser: ^1.4.6 - this allows us to pass the users from the backend to the browser and store it
- cors: ^2.8.5 - checks where the request comes from and validates it
- dotenv: ^16.4.5 - hides all of our secret info from github
- express: ^4.19.2 - express is what the server is built and deployed on
- jsonwebtoken: ^9.0.2 - JWT is what we used to log in and log out - it works with bcrypt to keep things very secure
- mongodb: ^6.8.0 - the database in which all of our data is saved too.
- mongoose: the package used to make models to send the code to our mongoDB database

Demonstrates good code flow control for user stories

The flow control of our users is fundamentally how this app works. The app allows our users to flow through the app based on the inputs they supply. Code is executed on their inputs and this achieves the users desired outcome.

Login/register

If a user has an account they can login. If a user wants an account they can register and add their own personal user/business details.

Invoice - view/download/create/delete

If user has login or has registered they can create, download, view and delete invoices. This is all accessible by the home screen

Users - view/create/delete

If user has login or has registered they can add, view and edit their own details. This is all accessible by the home screen

Clients - view/create/delete

If user has login or has registered they can create, add to invoice and delete clients. This is all accessible by the home screen

MANUAL USER STORY TESTS			
Primary - Designer - original user story	Flow 1	Flow 2	Flow 3
As a freelance designer I want a simple web app for managing my invoicing needs, including clients, projects, and invoices. So that I have more design work time and administrative tasks take less time.	If user has the right login credentials or has made an account: User logs in, see all invoices and clients on one page	Clicks on invoice, views it and can click to download and save it to computer	Can store all clients. They can be managed: create new or delete
As a freelance designer, I want to make and send minimalistic invoices to my clients, so that I can speed up my billing and sleep easily.	If user has the right login credentials or has made an account: User logs in, see all invoices and clients on one page	Once click to download	
As a freelance designer, I need to be able to see all of my invoices, so that I can follow up on payments and manage my clients effectively.	If user has the right login credentials or has made an account: User logs in, see all invoices and clients on one page	Clicks on invoice, views it and can click to download and save it to computer	
As a freelance designer, I need to be able to add or expense design assets, so that I can charge these expenses to my clients.	If user has the right login credentials or has made an account: User logs in, see all invoices and clients on one page	User can then add line items within the invoice modal, which can then be deleted if needed	
Secondary - Video editor - original user story	Flow 1	Flow 2	Flow 3
As a freelance video editor, I want a simple web app for managing my invoicing needs, including clients, projects, and invoices. So that I can spend more time on video editing and less time on administrative tasks.	If user has the right login credentials or has made an account: User logs in, see all invoices and clients on one page	Clicks on invoice, views it and can click to download and save it to computer	Can store all clients. They can be managed: create new or delete
As a Video Editor, I want to create invoices that include hour rates, day rates and project details, So that my clients clearly understand what they are paying for.	If user has the right login credentials or has made an account: User logs in, see all invoices and clients on one page	User can then add line items within the invoice modal, which can then be deleted if needed	
As a Video Editor, I want mobile access to create and look after invoices when I am out and about, So that I can make my job flexible and responsive to client needs.	If user has the right login credentials or has made an account: they can access the app from a mobile web browser		
As a Video Editor, I need to be able to add or expense stock footage or hard drive processing, so that I can charge these expenses to my clients.	If user has the right login credentials or has made an account: User logs in, see all invoices and clients on one page	User can then add line items within the invoice modal, which can then be deleted if needed	

STORY TESTS CONTROL FLOW [HERE](#)

Applies Object Oriented (OO) principles/patterns

Object Oriented (OO) principles have been heavily applied to mongoose schema structures.

```
const mongoose = require("mongoose");
const clientSchema = mongoose.Schema(
{
    userId: {
        type: mongoose.Schema.Types.ObjectId, ref: "User",
    },
    createdAt: {
        type: Date,
        default: Date.now
    },
    businessName: {
        type: String,
        required: true,
        unique: false
    },
    businessAbn: {
        type: String,
        required: true,
        unique: false
    },
    businessAddress: {
        type: String,
        required: true,
        unique: false
    },
    businessEmail: {
        type: String,
        required: true,
        unique: false
    },
    businessPhoneNumber: {
        type: Number,
        required: true,
        unique: false
    },
},
{
    timestamps: true
});

const ClientModel = mongoose.model("Client", clientSchema);

module.exports = {
    ClientModel
}
```

```
const mongoose = require("mongoose");
const lineItemSchema = mongoose.Schema(
{
    createdAt: {
        type: Date,
        default: Date.now
    },
    userId: {
        type: mongoose.Schema.Types.ObjectId,
        ref: "User",
        required: true,
        // unique: false
    },
    description: {
        type: String,
        required: true,
        unique: false
    },
    quantity: {
        type: Number,
        required: true,
        unique: false
    },
    unitPrice: {
        type: Number,
        required: true,
        unique: false
    },
    type: Number,
    required: true,
    unique: false
},
{
    invoice: {
        type: mongoose.Schema.Types.ObjectId,
        ref: "Invoice",
        required: true,
    },
},
{
    timestamps: true
});

const LineItemModel = mongoose.model("LineItem", lineItemSchema);

module.exports = {
    LineItemModel,
    lineItemSchema
}
```

```
const mongoose = require("mongoose");
const { lineItemSchema } = require("./LineItemModel")

const invoiceSchema = mongoose.Schema(
{
    createdAt: {
        type: Date,
        default: Date.now
    },
    lineItems: [
        {
            type: mongoose.Schema.Types.ObjectId, ref: "LineItem",
            required: false,
        }
    ],
    lineItemsTotal: {
        type: mongoose.Schema.Types.ObjectId, ref: "LineItemsTotal",
        type: Number,
        required: false,
    },
    client: {
        type: mongoose.Schema.Types.ObjectId, ref: "Client",
        required: false,
    },
    clientName: {
        type: mongoose.Schema.Types.ObjectId, ref: "ClientName",
        type: String,
        required: false,
    },
    userId: {
        type: mongoose.Schema.Types.ObjectId, ref: "User",
        required: false,
    },
    dueDate: {
        type: mongoose.Schema.Types.ObjectId, ref: "DueDate",
        type: String,
        required: false,
    },
    invoiceCount: {
        // type: mongoose.Schema.Types.ObjectId, ref: "InvoiceCount",
        type: Number,
        required: false,
    },
},
{
    timestamps: true
});

const InvoiceModel = mongoose.model("Invoice", invoiceSchema);

module.exports = {
    InvoiceModel
}
```

```
const mongoose = require("mongoose");
const { Schema, model } = mongoose;

const UserInfoSchema = new Schema(
{
    userId: {
        type: mongoose.Schema.Types.ObjectId,
        ref: "User",
        required: true,
        // unique: true
    },
    createdAt: {
        type: Date,
        default: Date.now
    },
    businessName: {
        type: String,
        required: false,
        unique: false
    },
    businessAbn: {
        type: String,
        required: false,
        unique: false
    },
    businessAddress: {
        type: String,
        required: false,
        unique: false
    },
    businessEmail: {
        type: String,
        required: false,
        unique: false
    },
    businessPhoneNumber: {
        type: Number,
        required: false,
        unique: false
    },
},
{
    timestamps: true
});

const UserInfoModel = model('UserInfo', UserInfoSchema);

module.exports = UserInfoModel;
```

```
const mongoose = require("mongoose");
const {Schema, model} = mongoose;

const UserSchema = new Schema(
{
    username: {
        type: String,
        required: true,
        min: 4,
        unique: true
    },
    password: {
        type: String,
        required: true
    },
    createdAt: {
        type: Date,
        default: Date.now
    },
},
{
    timestamps: true
}
);

const UserModel = model('User', UserSchema);

module.exports = UserModel;
```

Object Oriented (OO) principles have been heavily applied to all controller routes.

```

const dotenv = require('dotenv');
dotenv.config();

const User = require('../models/UserModel');
const bcrypt = require('bcrypt');
const jwt = require('jsonwebtoken');

const salt = bcrypt.genSaltSync(10);
const secret = process.env.JWT_SECRET;

exports.home = ('/', (request, response) => {
  response.send('Hello World! Welcome to Sup Invoice API')
});

// register user
> exports.register = async (request, response) => { ... };

// login as user
> exports.login = async (request, response) => { ... };

// user profile
> exports.profile = async (request, response) => { ... };

// logout user
> exports.logout = (request, response) => { ... };

```

```

const { InvoiceModel } = require('../models/InvoiceModel');
const mongoose = require('mongoose');

// create invoice
2_DanielCraig/T3A2-B-server/controllers/invoiceController.js · Contains emphasized items ...
};

// get all invoices
> exports.getInvoices = async (request, response) => { ... };

// get one invoice
> exports.getOneInvoice = async (request, response) => { ... };

// edit invoice
> exports.updateInvoice = async (request, response) => { ... };

// delete invoice
> exports.deleteInvoice = async (request, response) => { ... };

```

```

const UserInfoModel = require('../models/UserInfoModel');

// save User details
> exports.createUser = async (request, response) => { ... };

// get all User
> exports.getUser = async (request, response) => { ... };

// edit user
> exports.editUser = async (request, response) => { ... };

// delete user
> exports.deleteUser = async (request, response) => { ... };

```

```

const { ClientModel } = require('../models/ClientModel');
const mongoose = require('mongoose');

// save client details
> exports.createClient = async (request, response) => { ... };

// get all clients
> exports.getClients = async (request, response) => { ... };

// get one client
> exports.getOneClient = async (request, response) => { ... };

// edit client
> exports.editClient = async (request, response) => { ... };

// delete client
> exports.deleteClient = async (request, response) => { ... };

```

```

const { LineItemModel } = require('../models/LineItemModel');
const mongoose = require('mongoose');

// create line item
> exports.createLineItem = async (request, response) => { ... };

// get all line item by invoice id
> exports.getLineItemLinkedToInvoice = async (request, response) => { ... };

// get one line item
> exports.getOneLineItem = async (request, response) => { ... };

// edit line item
> exports.updateLineItem = async (request, response) => { ... };

// delete line item
> exports.deleteLineItem = async (request, response) => { ... };

```

Uses appropriate data structures

Functions

Functions have been used throughout to create reusable code. Here are a couple of examples:

This auth middleware was created and reused throughout the backend.

```
const dotenv = require('dotenv');
const jwt = require('jsonwebtoken');

dotenv.config();

const secret = process.env.JWT_SECRET;

const verifyUser = (request, response, next) => {
    // gets token from cookie
    const { token } = request.cookies;
    // makes sure its there
    if (!token) {
        return response.status(404).json({ message: "No auth token" });
    };
    // it then verifies it
    jwt.verify(token, secret, {}, (error, information) => {
        if (error) {
            // if user isn't logged in, we run this error but server keeps running
            return response.status(401).json("You need to be a valid user to login");
        } else {
            request.userId = information.id;
            // next, make it async
            next();
        }
    });
};

module.exports = verifyUser;
```

This header function was made once but used on every page on the app.

```
import { Link } from "react-router-dom"
// import logo from "../assets/logo.svg"
import ModeToggle from "@components/modules/mode-toggle"
import { useContext, useEffect } from "react"
import { Button } from "@components/ui/button"
import { UserContext } from "@context/UserContext"
import SvgComponent from "./svgLogo"

export default function Header() {
    const {setUserInfo, userInfo} = useContext(UserContext)
    useEffect(() => {
        fetch('http://localhost:4000/profile', {
            credentials: 'include',
        }).then(response => {
            if (response.status == 200) {
                response.json().then(data => {
                    setUserInfo(data.information)
                })
            }
        })
    }, [setUserInfo]);

    function logout() {
        fetch('http://localhost:4000/logout', {
            credentials: 'include',
            method: 'POST'
        });
        setUserInfo(null);
    }

    const username = userInfo?.username;
}

>     return [..]
      []
};
```

Ranges

Ranges have been used throughout to handle data. Here are a couple of examples:

Reducers and filters were used throughout to find and complete math logic, here are some examples:

```

const deleteLineItemById = (id) => {
  setLoading(true);
  fetch('http://localhost:4000/lineitems/${id}', {
    method: 'DELETE',
    credentials: 'include',
  })
  .then((response) => {
    setLoading(false);
    if (response.status === 200 || response.status === 204) {
      console.log('Line item deleted');
      setLineItems(lineItems.filter(lineItem => lineItem._id !== id));
      // return response.json();
    } else if (response.status === 400) {
      alert('Failed deleting line item')
    } else {
      alert('Something went wrong')
    }
  })
  .catch((error) => {
    console.log(error);
    setLoading(false);
  });
};

const overallTotal = (lineItems) => {
  return lineItems.reduce((sum, item) => sum + item.total, 0);
}

```

Classes (useState/useEffect etc)

Classes have been updated in recent years in react, therefore hooks which are a newer practice have been used throughout. The app is full of useState/useEffect.

useState and useEffect were used throughout the project to complete almost every useable function on the app, here are some examples:

```

useEffect(() => {
  setLoading(true);
  fetch('http://localhost:4000/create/invoice', {
    method: 'POST',
    credentials: 'include',
    headers: {
      'Content-Type': 'application/json'
    },
    body: JSON.stringify()
  })
  .then((response) => {
    setLoading(false);
    if (response.status === 201) {
      // alert('Success creating invoice')
      console.log('Great success');
      return response.json();
      // onClose();
    } else if (response.status === 400) {
      alert('Failed creating invoice')
    } else {
      alert('Something went wrong')
    }
  })
  .then(data => {
    setInvoice(data._id);
    console.log(data);
  })
  .catch((error) => {
    setLoading(false);
    alert('An error happened while creating a line item')
    console.log(error);
  })
}, []);

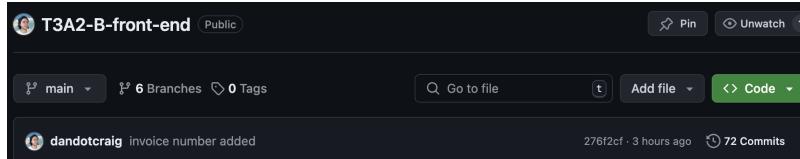
```

R3

Employ and utilise proper source control methodology (git)

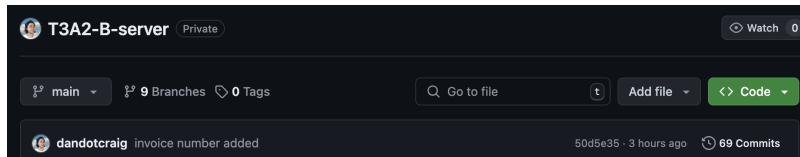
Front-End

At the time of writing this my front-end has 72 commits, with 6 completed feature branches. This was important for setting up new fetch routes, adding providers/context, refactoring the code to make it dryer and adding in line-items to invoices, which was an oversight in the original plan.



Back-End

At the time of writing this my back-end has 69 commits, with 9 completed feature branches. The branches were extremely important for features like middleware, line items and fragmenting it into files. I had it working and on making these major updates I didn't want to be adding these experiments to main.



Backend:

- Middleware setup branches
- line item experimentation branches (this was important as I wasn't sure if it would work)
- Router controller development.

R4

Demonstrate your ability to work in a team:

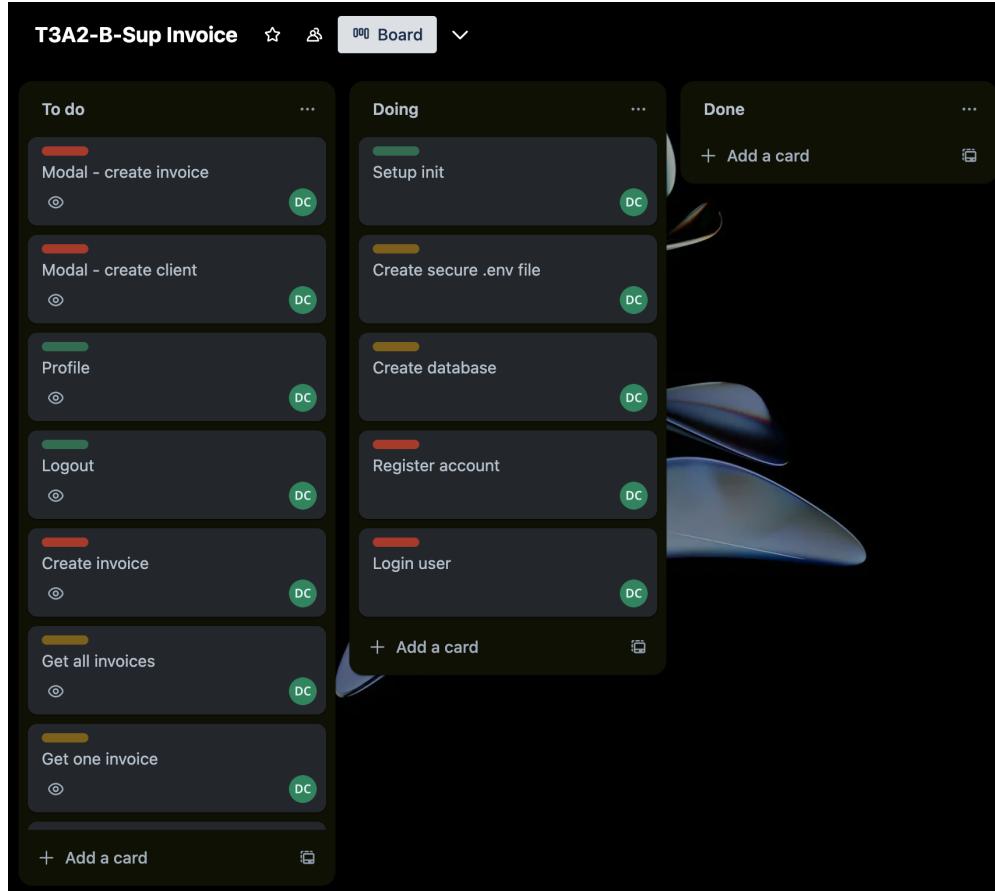
- Use a recognised project management methodology
 - Use a recognised task delegation methodology
-

We've used a Kanban framework to implement agile development processes to complete this first working version of Sup Invoice.

Trello was utilized to implement delegate tasks, plan the project and follow it through.

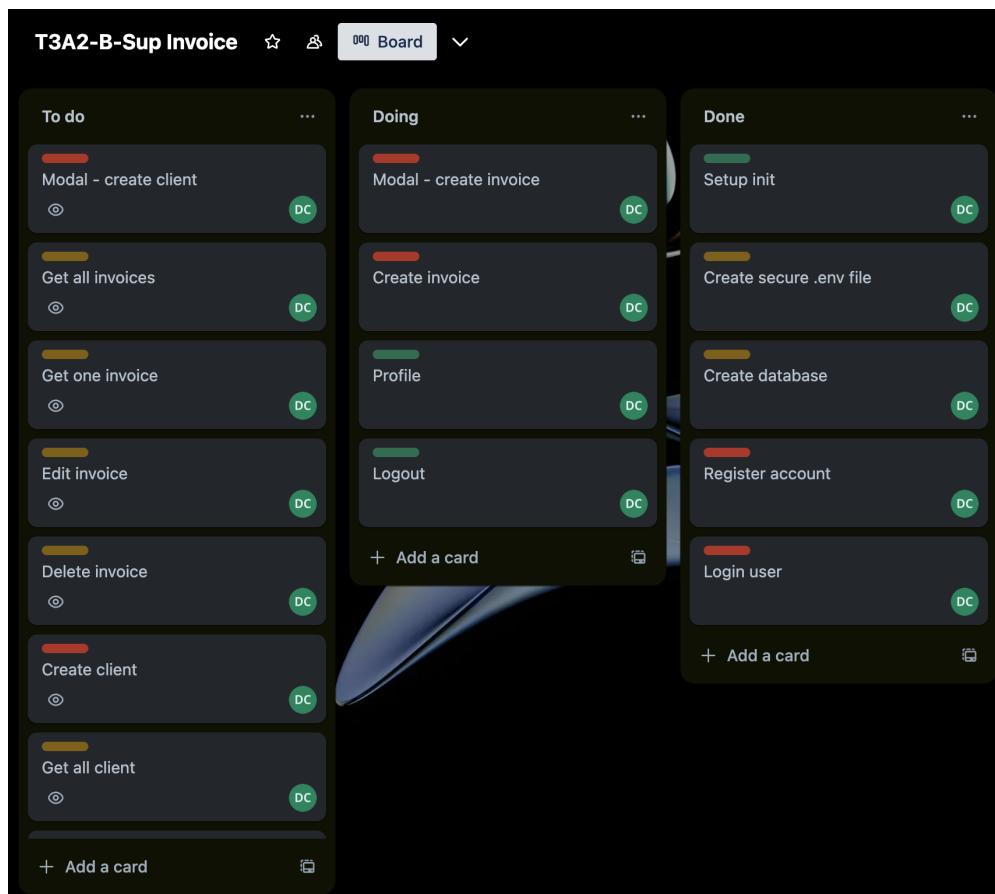
July 18

Init setup and auth setup



July 19

Create invoice modals and invoices - finish profile/login from Auth setup



July 20

Invoice routes setup

This Trello board is titled "T3A2-B-Sup Invoice". It has three columns: "To do", "Doing", and "Done".

- To do:**
 - Delete invoice
 - Create client
 - Get all client
 - Get one client
 - Edit client
 - Delete client
 - Create user info
- Doing:**
 - Modal - create client
 - Get all invoices
 - Get one invoice
 - Edit invoice
- Done:**
 - Create database
 - Register account
 - Login user
 - Modal - create invoice
 - Create invoice
 - Profile
 - Logout

Each card includes a "DC" button at the bottom right.

July 21

Client routes setup

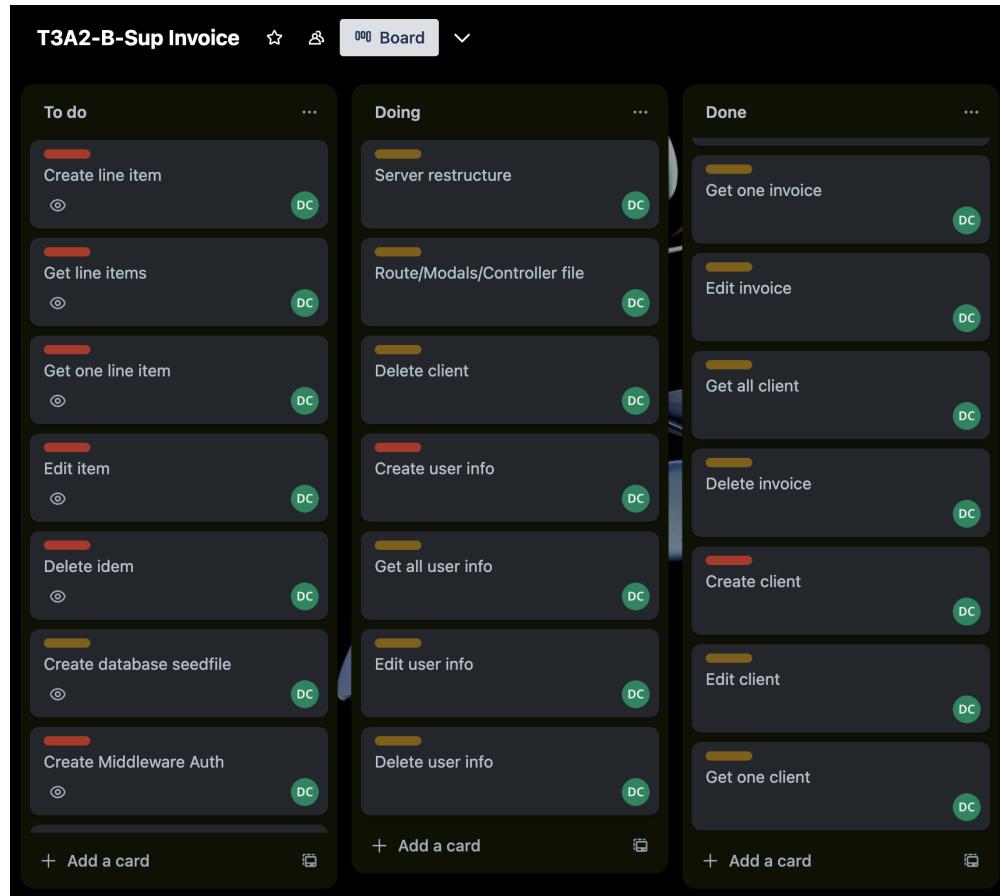
This Trello board is titled "T3A2-B-Sup Invoice". It has three columns: "To do", "Doing", and "Done".

- To do:**
 - Delete client
 - Create user info
 - Get all user info
 - Edit user info
 - Delete user info
 - Create line item
 - Get line items
- Doing:**
 - Route/Modals/Controller file
 - Get all client
 - Delete invoice
 - Create client
 - Edit client
 - Get one client
- Done:**
 - Create invoice
 - Profile
 - Logout
 - Modal - create client
 - Get all invoices
 - Get one invoice
 - Edit invoice

Each card includes a "DC" button at the bottom right.

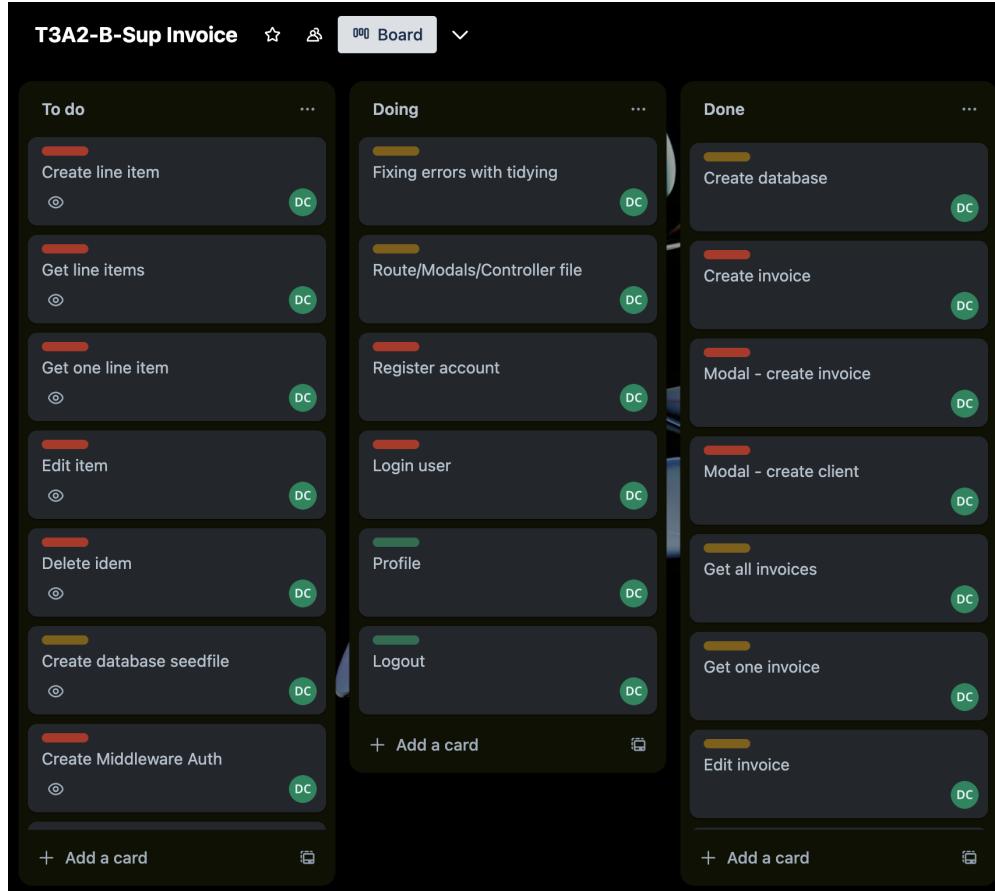
July 22

Users routes setup



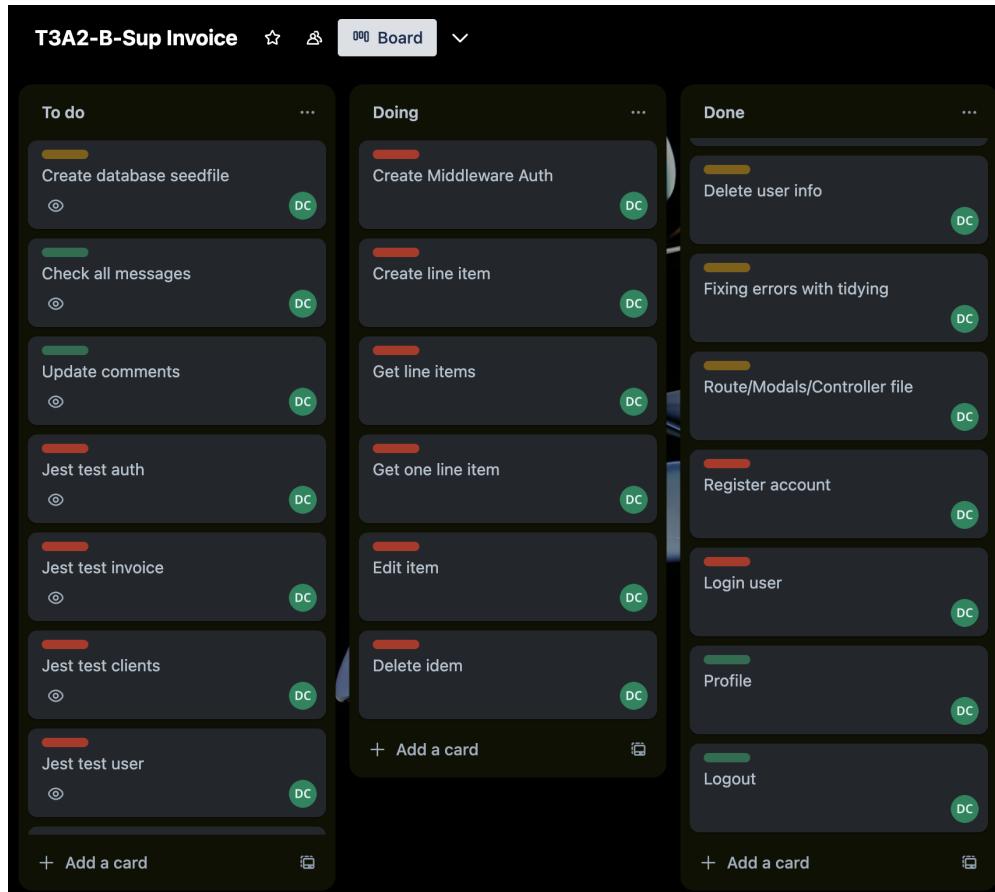
July 23

Restructure the file for DRY code



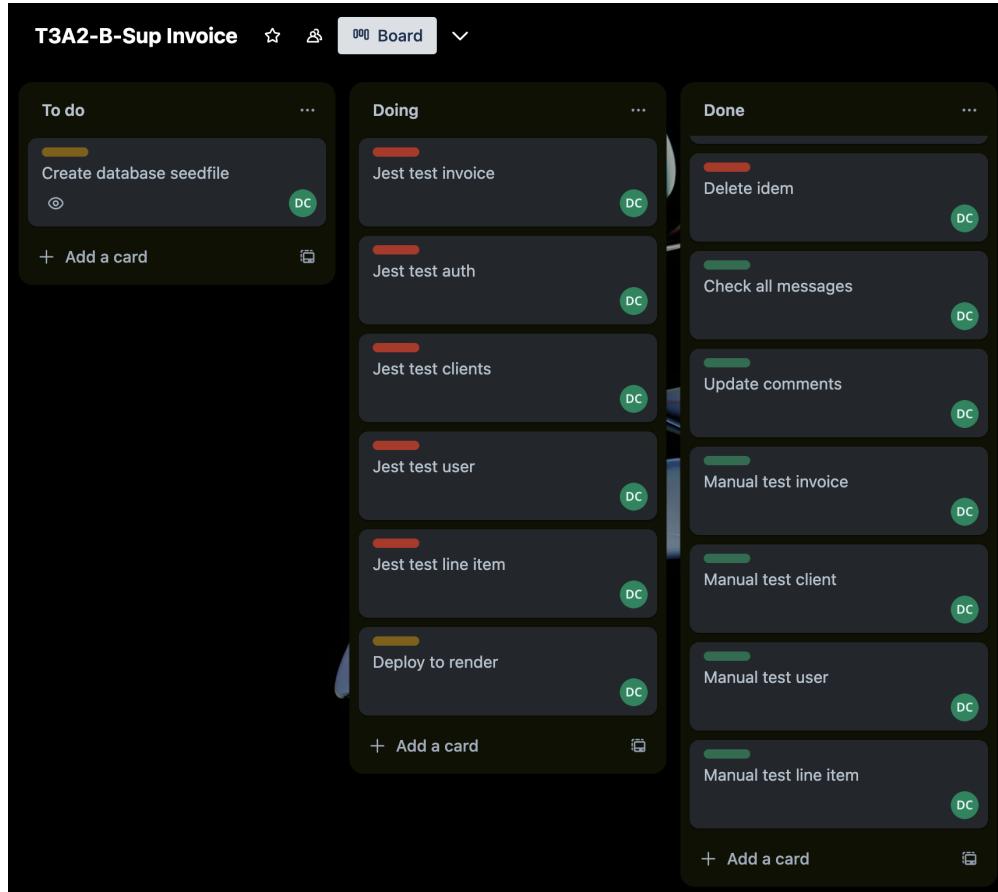
July 24

Create line items



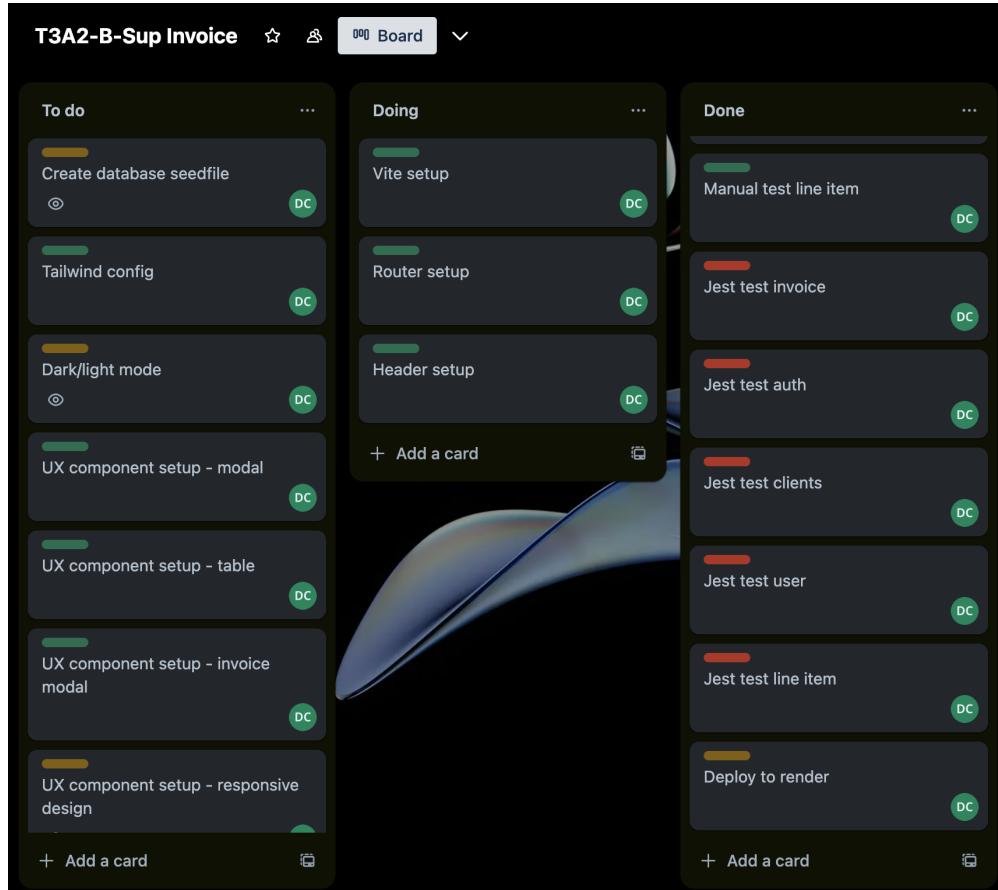
July 25-31

Testing



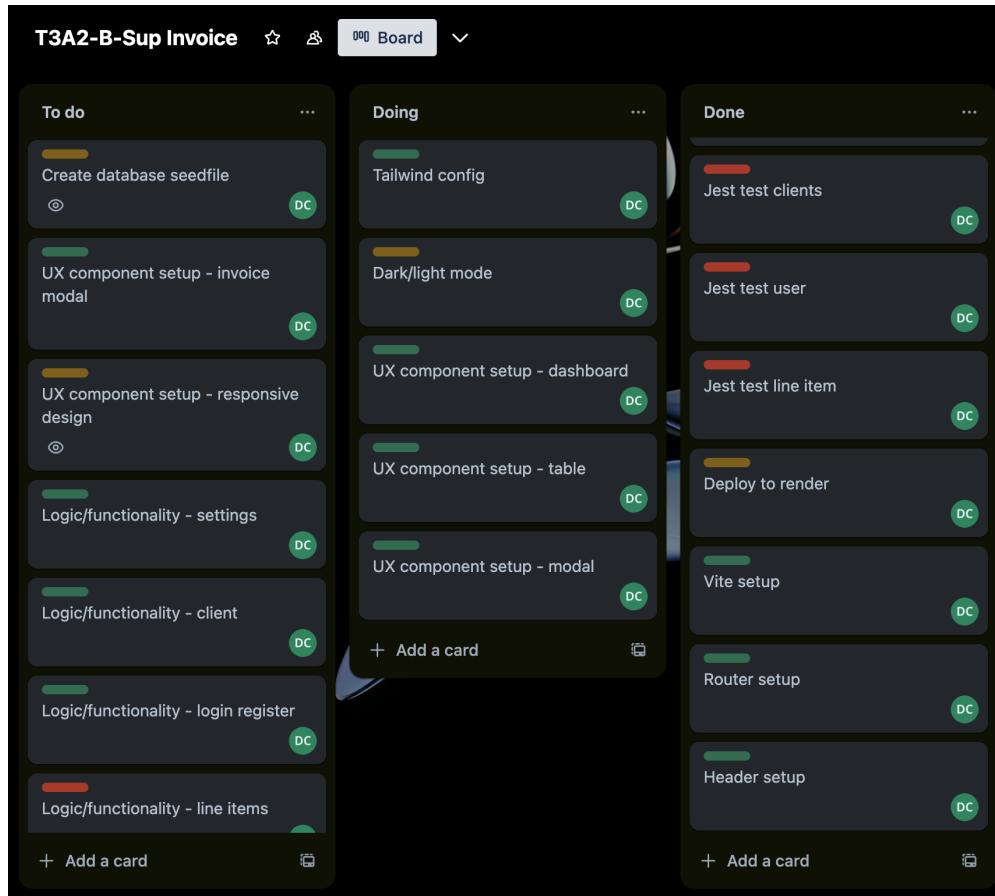
Aug 1

Setup vite, headers and router



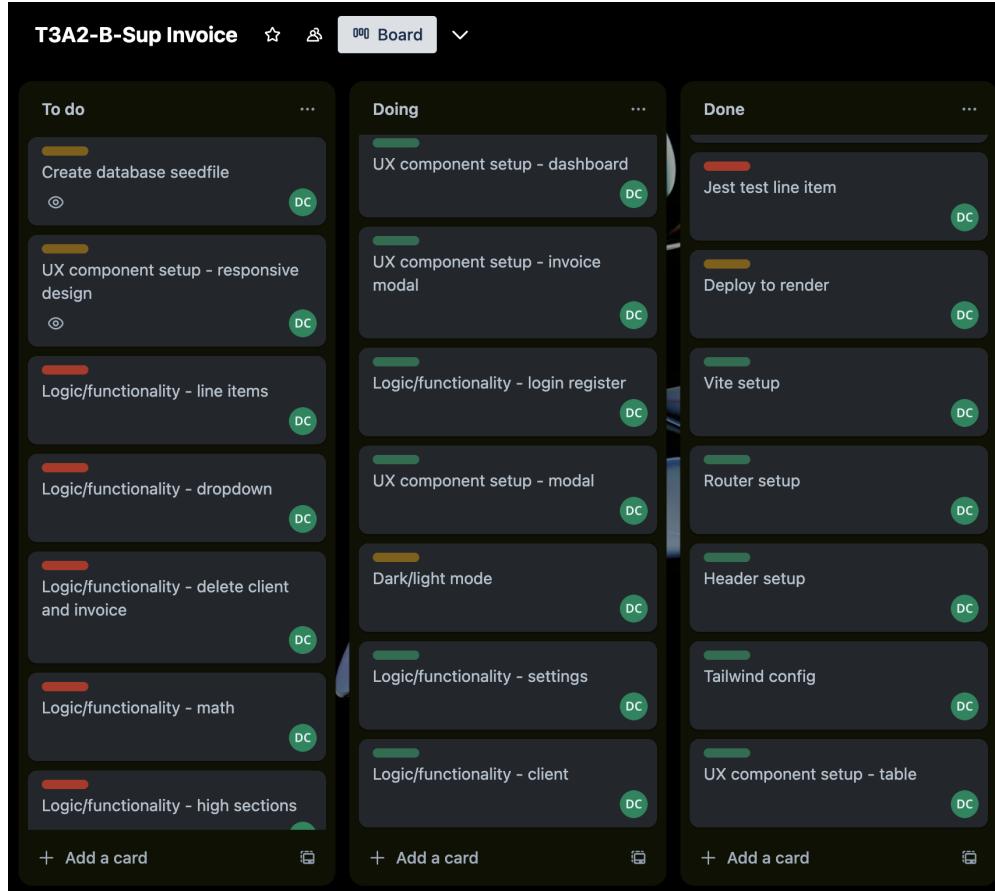
Aug 2

UX and component setup



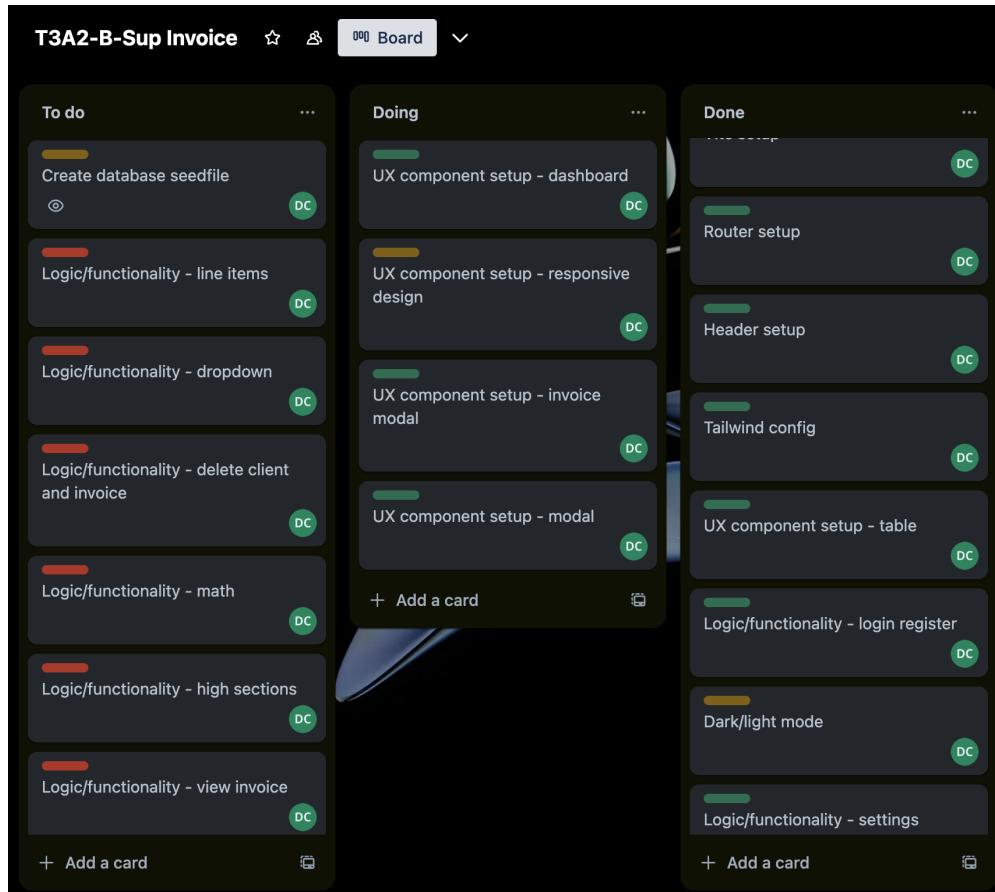
Aug 3

UX and component setup with some logic setup



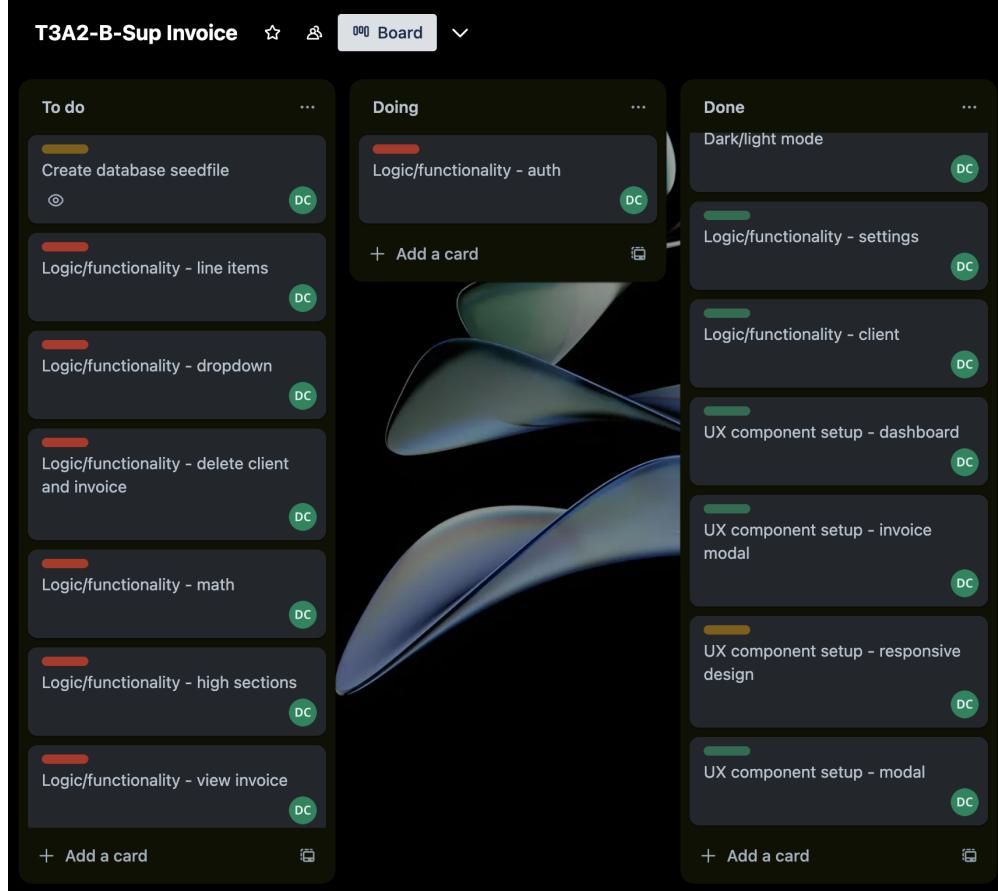
Aug 4

UX and component setup



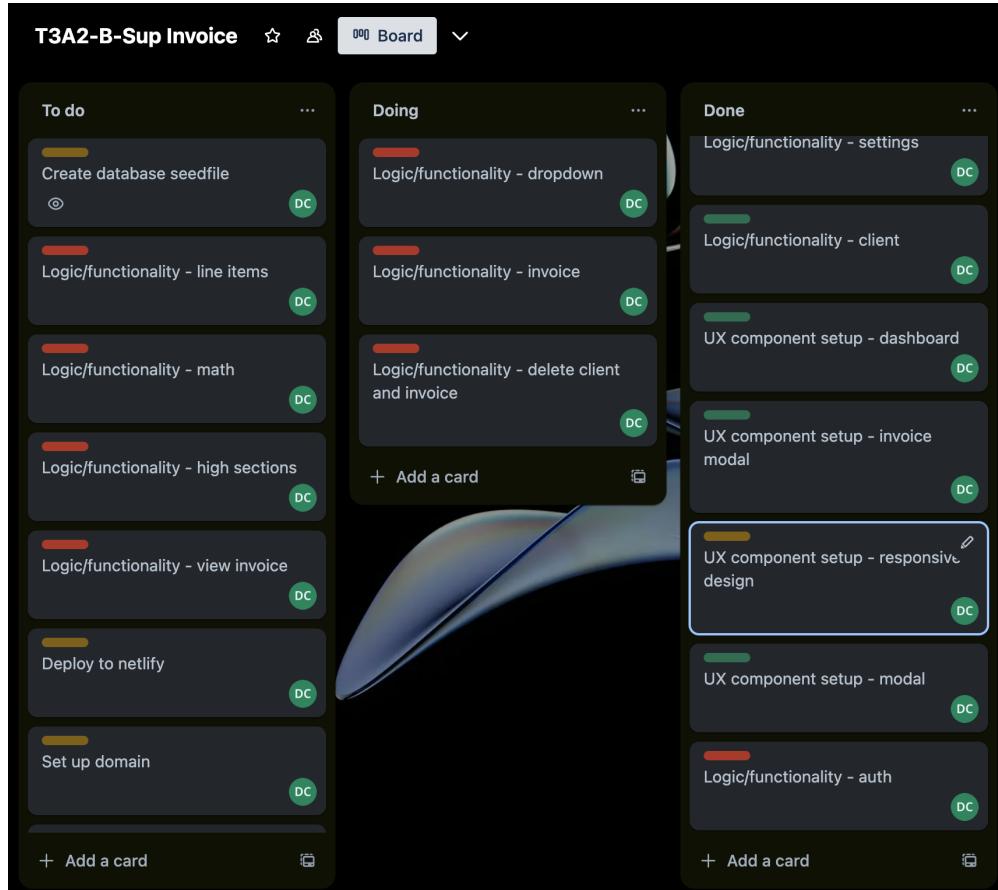
Aug 5

Auth logic applied to UX



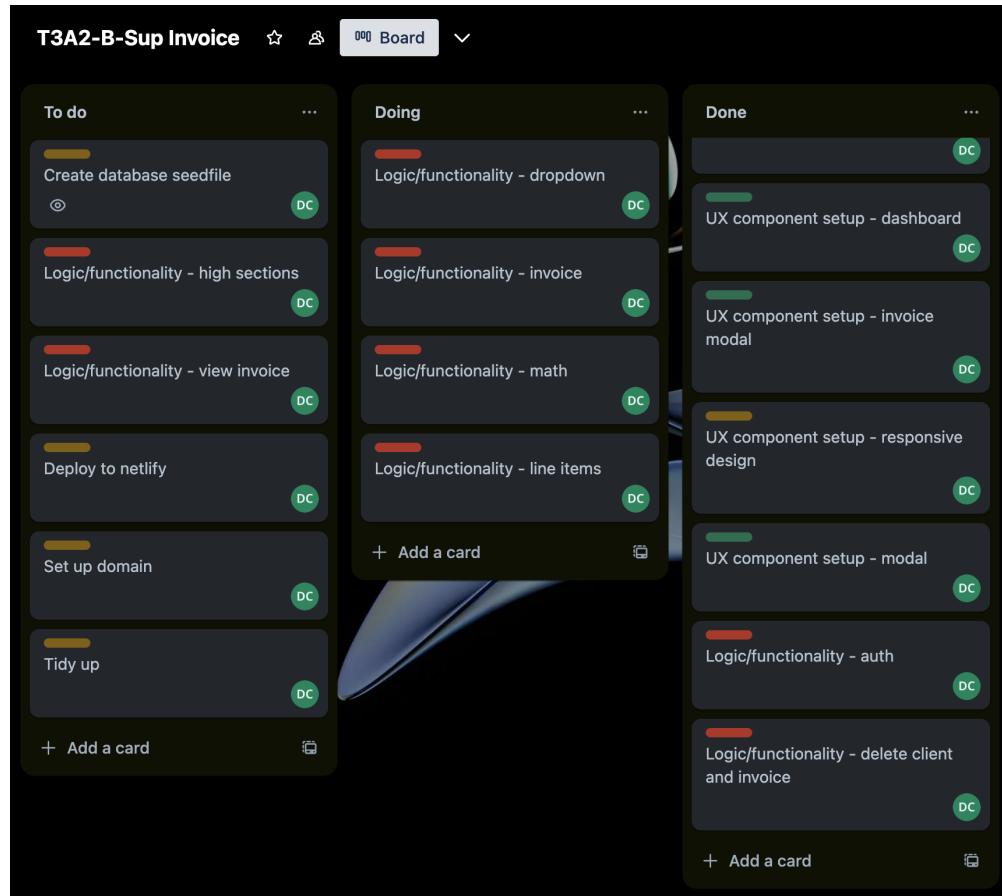
Aug 6

Invoice logic developed



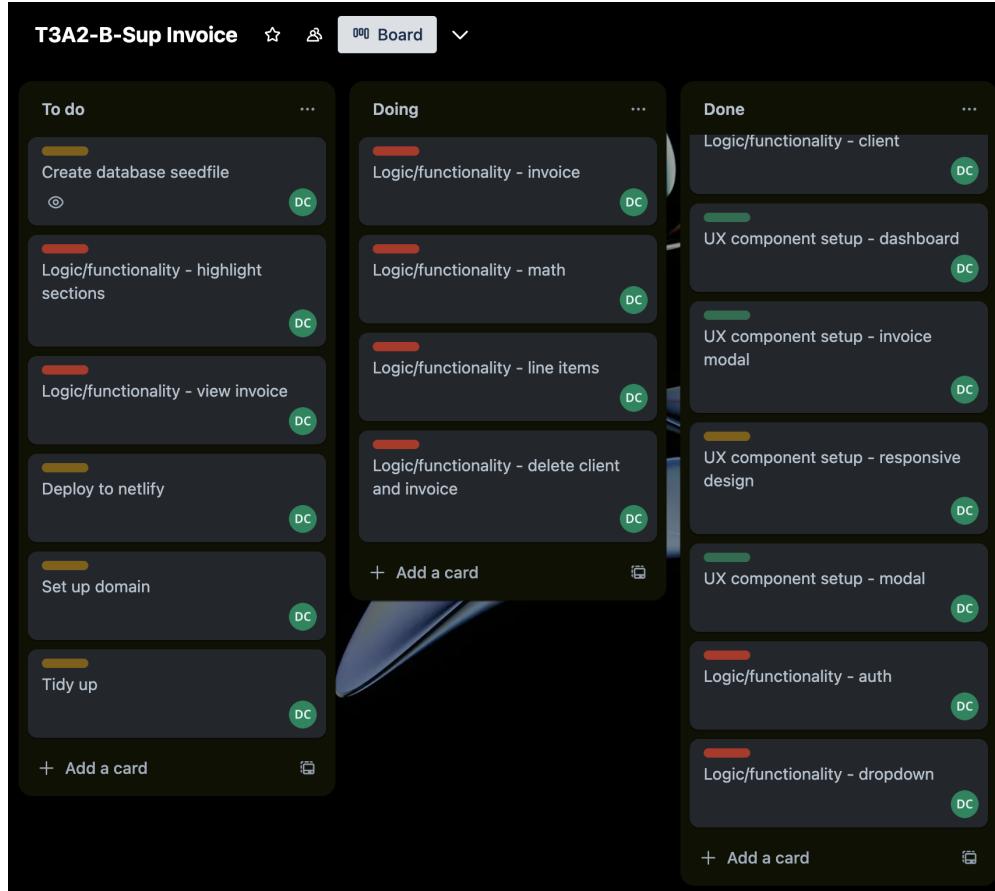
Aug 7

Invoice logic developed



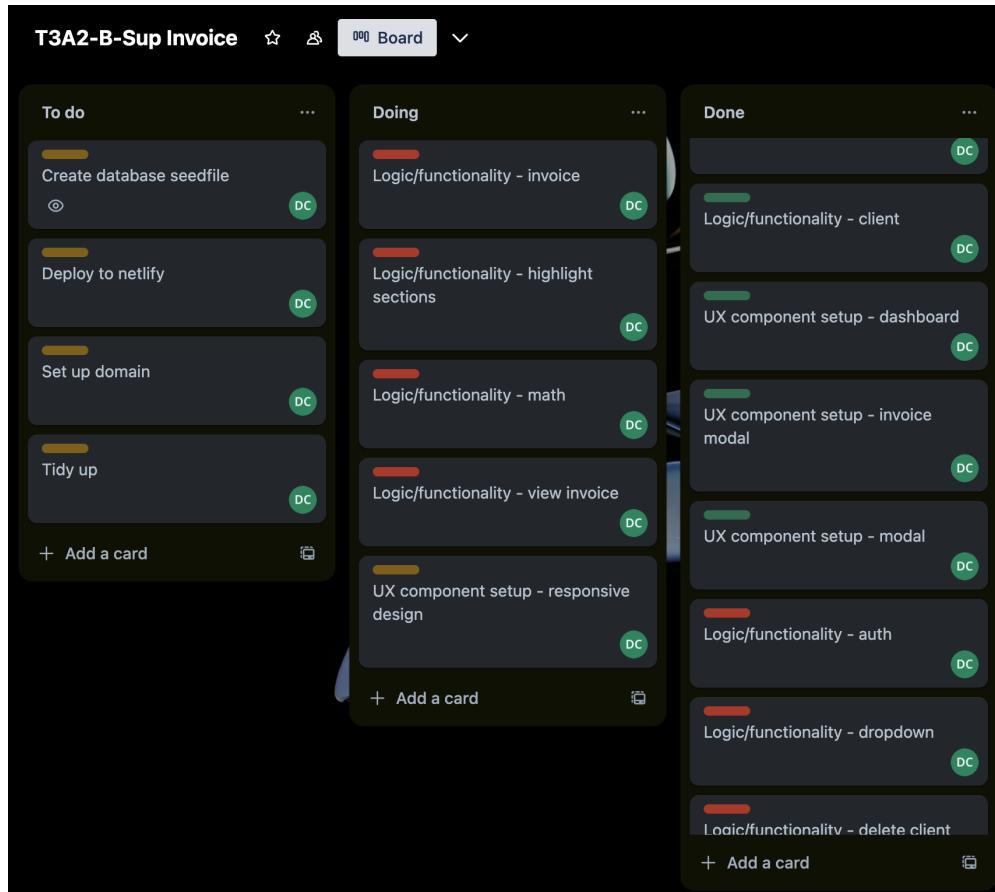
Aug 8

Invoice logic developed



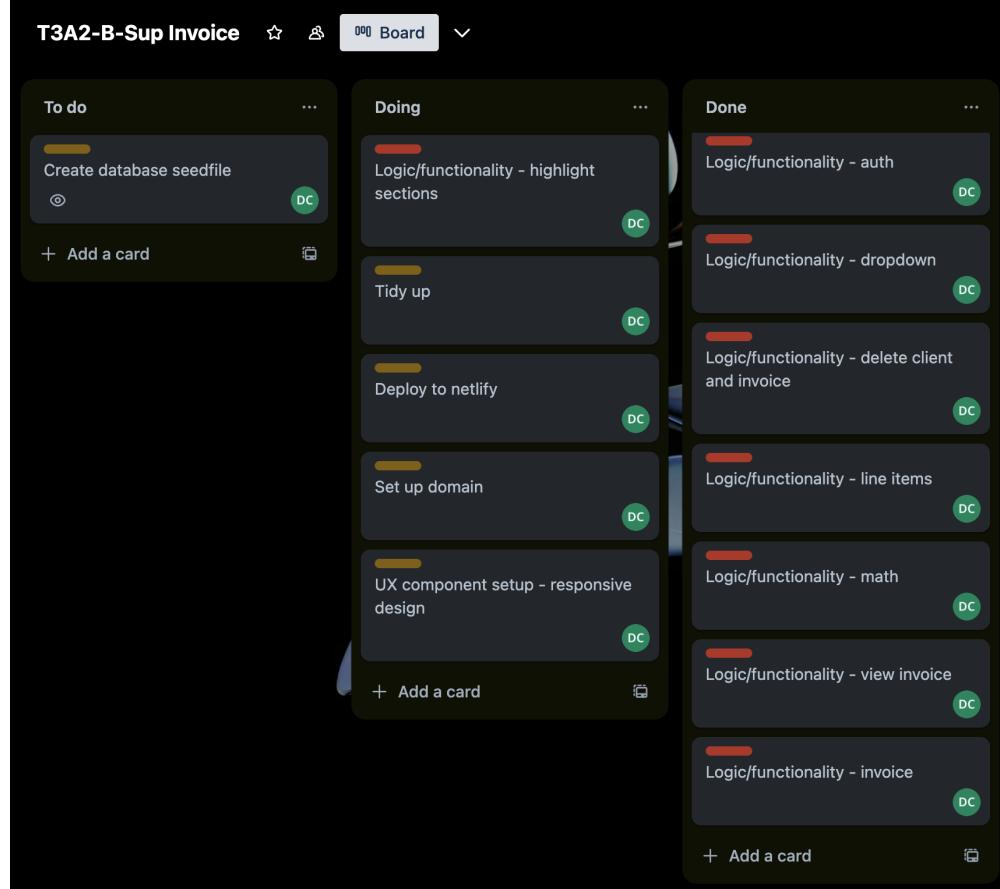
Aug 9

Invoice logic developed, tidy up and delete functions



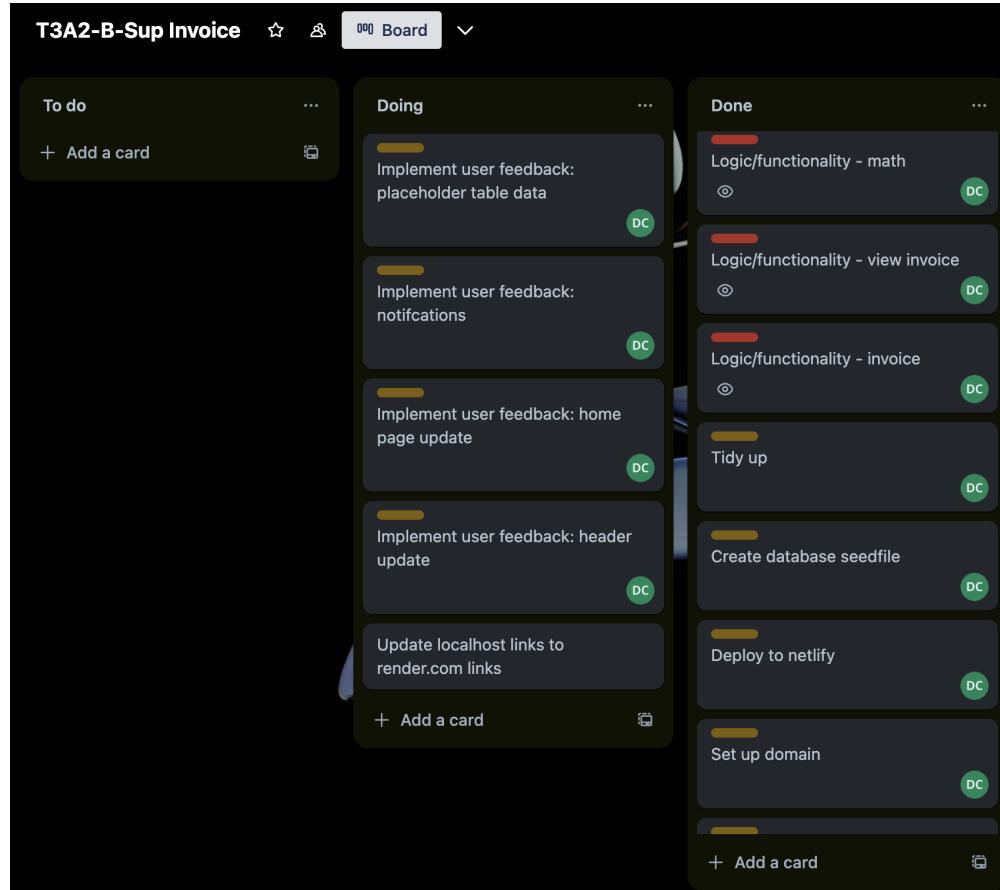
Aug 10

More tidy up and deployment



Aug 11

Final user feedback before launch



R5/ R6

Produce a working application that meets client and user needs

Deploy the application to a cloud hosting service

Deployed Sup Invoice can be found [HERE](#)

Netlify subdomain [HERE](#)

API can be found [HERE](#)

Feel free to use this testing account:

- User: manualtest
- Password: manualtest

R7

Produce an application with an intuitive user interface

This web application has utilized the ShadCN UI library, as well as tailwind. This mean't that the site is high user friendly and sleekly designed. Which is perfect for our itended users - professional freelance designers. We were tasked with developing an invoicing app. Below are five examples of great intuitive user interface design.

Utilizing tabs

Tabs are great as they don't feel like you're changing page, as the UX doesn't reload. This means you can display a lot of info and make it extremely easy to access. We used tabs on the home page to display the different pages. Overview, which shows all of the invoices, clients which displays all of the clients and settings which is a form to update the registers users details.

The dashboard features a top navigation bar with tabs: Overview (selected), Clients, and Settings. Below this is a summary section with four boxes:

- Total revenue**: \$144.00 (Total revenue, not including costs)
- GST**: +\$14.40 (10% of each invoice)
- Income tax**: +\$46.80 (32.5% of overall revenue)
- This year:** +3 (Total number of invoices)

Below the summary is a table of invoices:

Date	Client	Modify	Amount
08/10/24	coles express 2	edit trash	250
08/10/24	coles express 2	edit trash	32
08/10/24	coles express 2	edit trash	144
Total			\$144.00

Modals

Modals are another great UX trick, it makes the user feel like they are still on the home page. Making the pages minimal makes it feel less complex. The fact the user can see the app behind it (blurred) means it doesn't feel hard to get back to the main interface. Modals were utilized twice for client creation, invoice creation and invoice viewing.

The modal is titled "Invoice details" and contains fields for "Add client details below". It includes sections for "Date due" (Date due), "Select client" (Select a client), and "Line item" (Description, quantity, Unit price). A "Create line item" button is at the bottom left.

The main "Invoice" section shows:

- Invoice date**: 10/7/2024
- Due date**: Type in a date
- Invoice number**: 4
- To**: Select a client
- From**: manualUX
1234567654321
12 manualUX st Potts point
87654422523

A table for "Line item" details shows:

Description	Delete	Amount
Subtotal		\$0.00
GST		\$0.00
Total		\$0.00

At the bottom are "Close" and "Save invoice" buttons.

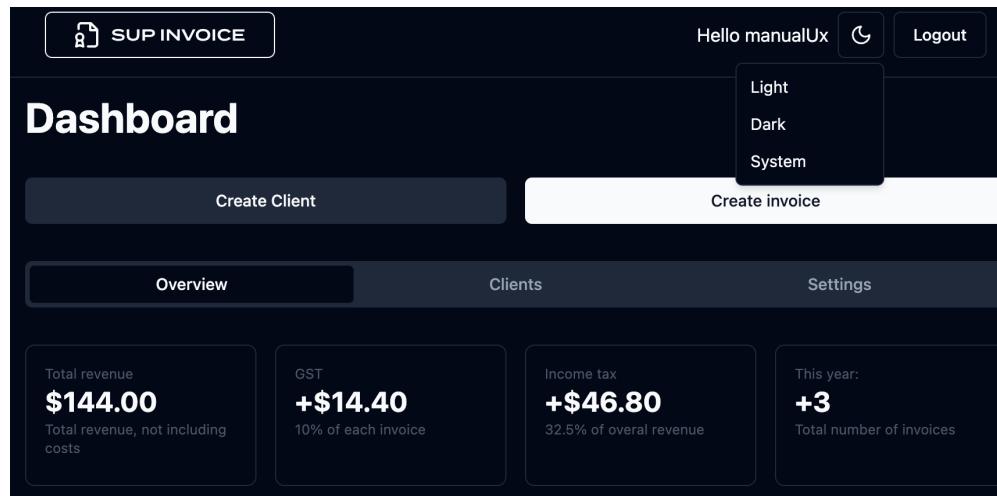
Dark and light mode

With more and more users getting large 4k screens with strong blue light - light mode is becoming less and less. This is due to it putting a lot of strain on the eyes due to its brightness. The ShadCN library made it really easy to implement this.

Select client <div style="border: 1px solid #ccc; padding: 5px; width: 150px; margin-bottom: 10px;"> coles express 2 </div> <div style="border: 1px solid #ccc; padding: 5px; width: 150px; background-color: #f0f0f0;"> Clients <div style="border: 1px solid #ccc; padding: 2px; display: inline-block;">✓ coles express 2</div> </div>	To coles express 2 23435465342312 12 potts point st potts point 123454321
Line item	Description

Dropdowns

A user could have hundreds of clients. That's why we've used dropdowns to display the clients. This means it will appear in a scrollable list, easy for the user to find them.



Register flow

The registration flow consists of two screens, one the registration section, which takes an email and password. Then once it's been completed, the cookies are stored - meaning the user can be logged straight in. A pop up is added which gets the user's details and stores them. Once complete it redirects to the home page so users can start making some invoices!

R8 / R9

Provides evidence of user testing:

- In the development environment
- In the production environment

Utilises a formal testing framework

User Story testing

MANUAL USER STORY TESTS - sheet [HERE](#)

Primary - designer

- As a freelance designer I want a simple web app for managing my invoicing needs, including clients, projects, and invoices. So that I have more design work time and administrative tasks take less time.

Manual user test

3	As a freelance designer I want a simple web app for managing my invoicing needs, including clients, projects, and invoices. So that I have more design work time and administrative tasks take less time.	How has it been implemented?	Clients and invoices can be saved and called upon	Simple and easy user interface. Easy to login. A URL that is simple to remember.	
---	---	------------------------------	---	--	--

Solution

Dashboard

The dashboard features a top navigation bar with 'Create Client' and 'Create invoice' buttons. Below is a navigation menu with 'Overview' (selected), 'Clients', and 'Settings'. Four summary boxes show: Total revenue (\$144.00), GST (+\$14.40), Income tax (+\$46.80), and This year: +3 total number of invoices. A table lists three invoices from 'coles express 2' with amounts 250, 32, and 144, totaling \$144.00.

Date	Client	Modify	Amount
08/10/24	coles express 2		250
08/10/24	coles express 2		32
08/10/24	coles express 2		144
Total			\$144.00

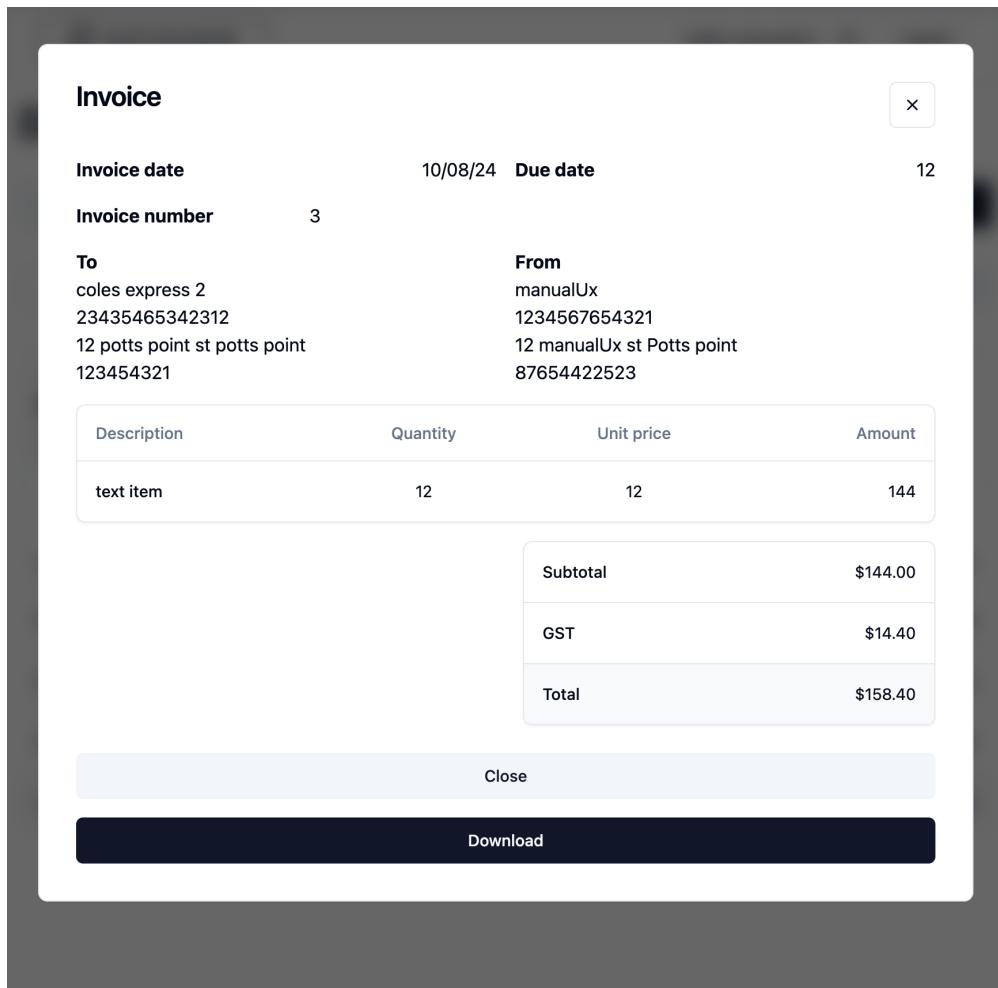
User interface is simple and easy to use. Everything that a user could want is on the home page, with easy access. This is exactly inline with our users needs

- As a freelance designer, I want to make and send minimalistic invoices to my clients, so that I can speed up my billing and sleep easily.

Manual user test

As a freelance designer, I want to make and send minimalistic invoices to my clients, so that I can speed up my billing and sleep easily.	How has it been implemented?	The invoices have all the info required. The design of the app is sleek and professional. Looks better than anything from a spreadsheet template.	Dates and remembering client details are saved and automated to you only need to add the line items.	
---	------------------------------	---	--	--

Solution



The invoice is all there and easy to access. It has a download button for easy and simplistic sharing.

3. As a freelance designer, I need to be able to see all of my invoices, so that I can follow up on payments and manage my clients effectively.

Manual user test

Dashboard

The dashboard features a top navigation bar with 'Create Client' and 'Create invoice' buttons. Below this are three tabs: 'Overview' (selected), 'Clients', and 'Settings'. The 'Overview' section displays four key metrics in boxes: 'Total revenue \$144.00' (not including costs), 'GST +\$14.40' (10% of each invoice), 'Income tax +\$46.80' (32.5% of overall revenue), and 'This year: +3' (total number of invoices). A table below lists invoices with columns for Date, Client, Modify (with delete icons), and Amount. The table shows three entries for 'coles express 2' with amounts 250, 32, and 144, totaling \$144.00.

Date	Client	Modify	Amount
08/10/24	coles express 2		250
08/10/24	coles express 2		32
08/10/24	coles express 2		144
Total			\$144.00

The dashboard has everything this designer needs. The date is the first thing, so users can look, see when they made it and make any follow ups or checks with their clients.

- As a freelance designer, I need to be able to add or expense design assets, so that I can charge these expenses to my clients.

Manual user test

As a freelance designer, I need to be able to add or expense design assets, so that I can charge these expenses to my clients.	How has it been implemented?	Line items can be added to the invoices and so can negative line items to expense assets etc.	Invoice can be downloaded and saved.	
--	------------------------------	---	--------------------------------------	--

Solution

The solution interface shows a left sidebar for 'Line item' creation with fields for 'line item' and '12'. On the right, a table lists an existing line item 'line item' with amount '144'. Below this are summary rows for 'Subtotal' (\$144.00), 'GST' (\$14.40), and 'Total' (\$158.40). A 'Create line item' button is located at the bottom left of the sidebar.

Description	Delete	Amount
line item		144

Subtotal	\$144.00
GST	\$14.40
Total	\$158.40

A simple yet simplistic way to add new line items to the invoice. They are added on the left and users can see them on the right side. They automatically sum up and they're easy to delete.

Secondary - Video editor

- As a freelance video editor, I want a simple web app for managing my invoicing needs, including clients, projects, and invoices. So that I can spend more time on video editing and less time on

administrative tasks.

✓ Manual user test

As a freelance video editor, I want a simple web app for managing my invoicing needs, including clients, projects, and invoices. So that I can spend more time on video editing and less time on administrative tasks.	How has it been implemented?	Clients and invoices can be saved and called upon	Simple and easy user interface. Easy to login. A URL that is simple to remember.	<input checked="" type="checkbox"/>
--	------------------------------	---	--	-------------------------------------

✓ Solution

Date	Client	Modify	Amount
08/10/24	coles express 2	<input type="button" value="edit"/> <input type="button" value="delete"/>	250
08/10/24	coles express 2	<input type="button" value="edit"/> <input type="button" value="delete"/>	32
08/10/24	coles express 2	<input type="button" value="edit"/> <input type="button" value="delete"/>	144
Total			\$144.00

User interface is simple and easy to use. Everything that a user could want is on the home page, with easy access. This is exactly inline with our users needs

2. As a Video Editor, I want to create invoices that include hour rates, day rates and project details, So that my clients clearly understand what they are paying for.

✓ Manual user test

As a Video Editor, I want to create invoices that include hour rates, day rates and project details, So that my clients clearly understand what they are paying for.	How has it been implemented?	The line items can add any information to them	They are displayed neat and clearly for clients to understand	<input checked="" type="checkbox"/>
--	------------------------------	--	---	-------------------------------------

✓ Solution

Description	Delete	Amount
line item	<input type="button" value="delete"/>	144

Subtotal	\$144.00
GST	\$14.40
Total	\$158.40

A simple yet simplistic way to add new line items to the invoice. They are added on the left and users can see them on the right side. They automatically sum up and they're easy to delete.

- As a Video Editor, I want mobile access to create and look after invoices when I am out and about, So that I can make my job flexible and responsive to client needs.

Manual user test

As a Video Editor, I want mobile access to create and look after invoices when I am out and about, So that I can make my job flexible and responsive to client needs.	<i>How has it been implemented?</i>	The interface is responsive and works well on mobile	
---	-------------------------------------	--	---

Solution

12

Create line item

Invoice

Invoice date 10/7/2024

Due date Type in a date

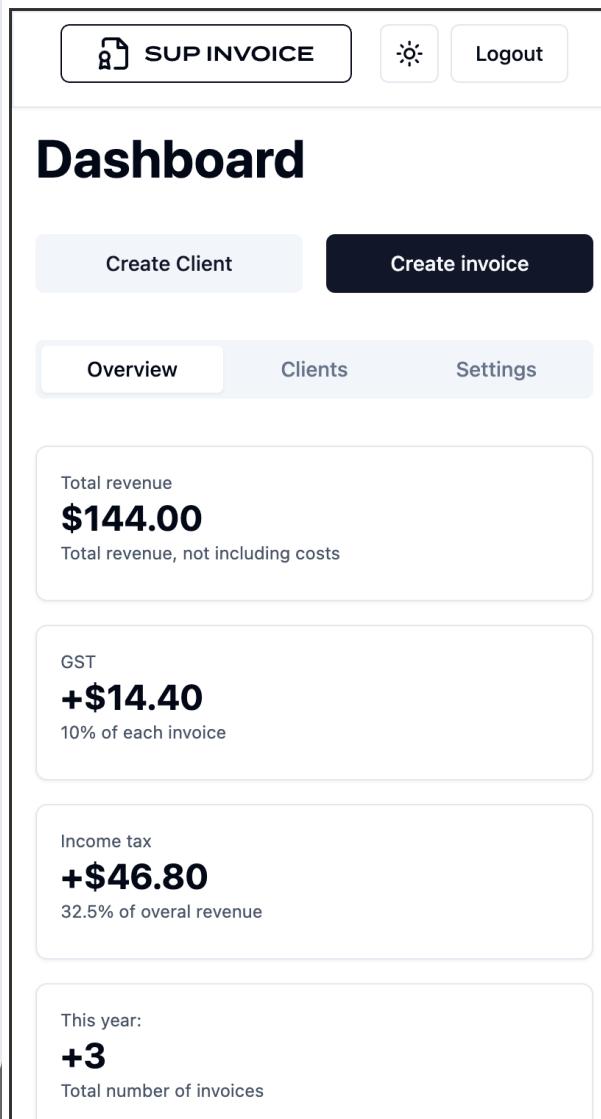
Invoice number

4

To
Select a client

From
manualUx
1234567654321
12 manualUx st Potts point
87654422523

Description	Delete	Amount
line item	刪	144
Subtotal		\$144.00



Invoice

Invoice date 10/08/24

Due date 12/12/24

Invoice number

1

To
coles express 2
23435465342312
12 potts point st potts point
123454321

From
manualUx
1234567654321
12 manualUx st Potts point
87654422523

Description	Amount
new item 2	20
new item 1	230

The design is completely responsive and even works on devices with small screens - down to 300px in width. Which by todays standards - is small.

4. As a Video Editor, I need to be able to add or expense stock footage or hard drive processing, so that I can charge these expenses to my clients.

✓ Manual user test

As a Video Editor, I need to be able to add or expense stock footage or hard drive processing, so that I can charge these expenses to my clients.	How has it been implemented?	Line items can be added to the invoices and so can negative line items to expense assets etc.	Invoice can be downloaded and saved.	✓
---	------------------------------	---	--------------------------------------	---

✓ Solution

Line item	Description	Delete	Amount
line item	line item	✗	144
12			
12			
Create line item	Subtotal		\$144.00
	GST		\$14.40
	Total		\$158.40

A simple yet simplistic way to add new line items to the invoice. They are added on the left and users can see them on the right side. They automatically sum up and they're easy to delete.

Overall manual testing ✓

MANUAL BRUNO TESTING - sheet [HERE](#)

Manual testing was done with Bruno to insure it would work perfectly when implemented while also a result to compare it too in the browser.

AUTH						
Register	http://localhost:4000/register	Registers user with username and password - while also set cookies	{"username": "brunotest", "password": "brunotest"}	200 OK	200 OK	
login	http://localhost:4000/login	Logs user in with username and password	{"username": "brunotest", "password": "brunotest"}	200 OK	200 OK	
get profile	http://localhost:4000/profile	Gets profile cookies to run the header	0	200 OK	200 OK	
logout	http://localhost:4000/logout	Clears the set-cookies	0	200 OK	200 OK	
CLIENTS						
create client	http://localhost:4000/create/client	Creates clients details	{"clientName": "brunotest", "clientEmail": "brunotest@gmail.com", "clientAddress": "123 Main St", "clientPhone": "123-456-7890", "clientWebsite": "http://www.brunotest.com", "clientStatus": "Active"}	201 OK	201 OK	
get all clients	http://localhost:4000/clients	Gets them for the drop down in the invoice modal		200 OK	200 OK	
get one client	http://localhost:4000/clients/{client_id}	Gets one to display invoice when viewing it		200 OK	200 OK	
edit client by id	http://localhost:4000/clients/{client_id}	Edit client details	{"clientAddress": "1234567890", "clientEmail": "brunotest@gmail.com", "clientAddress": "123 Main St", "clientPhone": "123-456-7890", "clientWebsite": "http://www.brunotest.com", "clientStatus": "Active"}	200 OK	200 OK	
delete client by id	http://localhost:4000/clients/{client_id}	delete client, so users can click the bin in the table to delete invoice		200 OK	200 OK	
INVOICE						
create invoice	http://localhost:4000/create/invoice	Create an invoice with id so all the details can be PUT in on save	0	201 OK	201 OK	
get one invoice	http://localhost:4000/invoices/{invoice_id}	Gets invoice via ID - this is for viewing on at a time		200 OK	200 OK	
get all invoice	http://localhost:4000/invoices	Gets all invoices for the dashboard table		200 OK	200 OK	
edit invoice by id	http://localhost:4000/invoices/{invoice_id}	Invoice via ID - this is for viewing on at a time		200 OK	200 OK	
delete invoice by id	http://localhost:4000/invoices/{invoice_id}	Deletes on invoice via ID - this is for viewing on at a time		200 OK	200 OK	
LINE ITEMS						
create line item	http://localhost:4000/create/lineitem	Creates one individual line item	{"description": "Line Item", "quantity": 1, "unitPrice": 100, "taxRate": 0.05, "total": 105}	201 OK	201 OK	
get all line item	http://localhost:4000/lineitems	Gets all of them		200 OK	200 OK	
get one line item by invoice	http://localhost:4000/lineitems/invoice/{invoice_id}	This gets the line items associated with the invoice		200 OK	200 OK	
get one line item by id	http://localhost:4000/lineitems/{line_item_id}	This gets the line item by id for display		200 OK	200 OK	
edit line item by id	http://localhost:4000/lineitems/{line_item_id}	This edits the line item by id for display	{"description": "Line", "quantity": 2, "unitPrice": 100, "taxRate": 0.05, "total": 210}	200 OK	200 OK	
delete line item by id	http://localhost:4000/lineitems/{line_item_id}	This deletes the line item by id for display		200 OK	200 OK	
MANUAL BRUNO TESTING						
NAME	ROUTE	ROLE	DATA SENT	EXPECTED	RECEIVED	DATA RESPONSE
USER INFO						
create user info	http://localhost:4000/create/user	Adds user info to so it can be accessed by invoice	{"username": "brunotest", "password": "brunotest", "email": "brunotest@gmail.com", "address": "123 Main St", "phone": "123-456-7890", "website": "http://www.brunotest.com", "status": "Active"}	201 OK	201 OK	
get all users info	http://localhost:4000/user/	Gets the user info to populate the address	0	200 OK	200 OK	
edit user info	http://localhost:4000/user/	Edit user info incase details need to change	{"username": "brunotest", "password": "brunotest", "email": "brunotest@gmail.com", "address": "123 Main St", "phone": "123-456-7890", "website": "http://www.brunotest.com", "status": "Active"}	200 OK	200 OK	
delete user info	http://localhost:4000/user/{user_id}	Deletes details to start fresh		200 OK	200 OK	

Web interface end points:

MANUAL UX TESTING - sheet [HERE](#)

Following on from the bruno testing, we tested everything hard in the browser. Some routes weren't used and might be used down the track.

MANUAL UX TESTING						
USER INFO		STATUS	REASON	EXPECTED	RESULT	SUGGESTED IMPROVEMENTS
create user info	http://localhost:4000/create/user	Adds user info to so it can be accessed by invoice	Implemented	The screen after	T	
get all users info	http://localhost:4000/user/	Gets the user info to populate the address	Implemented	Get all user info on the invoice	✓	
edit user info	http://localhost:4000/user/	Edit user info incase details need to change	Implemented	User details are updated at the settings tab	✓	
delete user info	http://localhost:4000/user/{user id}	Deletes details to start fresh	Not applicable	Will come in future iterations	NA	NA
INVOICE						
create invoice	http://localhost:4000/create/invoice	Create an invoice with id so all the details can be PUT in on save	Implemented	Creates invoice	✓	
get one invoice	http://localhost:4000/invoices/{invoice id}	Gets on invoice via ID - this is for viewing on at a time	Implemented	Gets invoice for viewing	✓	
get all invoice	http://localhost:4000/invoices	Gets all invoices for the dashboard table	Implemented	Gets all invoices and puts them in the table	✓	
edit invoice by id	http://localhost:4000/invoices/{invoice id}	Edits on invoice via ID - this is for viewing on at a time	Implemented	This route only works to apply and replace data on creation. It cant edit the invoice as it stands.	✓	
delete invoice by id	http://localhost:4000/invoices/{invoice id}	Deletes on invoice via ID - this is for viewing on at a time	Implemented	Deletes invoice	✓	
CLIENTS						
create client	http://localhost:4000/create/client	Creates clients details	Implemented	Creates client	✓	
get all clients	http://localhost:4000/clients	Gets them for the drop down in the invoice modal	Implemented	Gets clients for client table	✓	
get one client	http://localhost:4000/clients/{client id}	Gets one to display invoice when viewing it	Implemented	Gets on client when selected inside invoice	✓	
edit client by id	http://localhost:4000/clients/{client id}	Edit client details	Not applicable	Will come in future iterations	NA	NA
delete client by id	http://localhost:4000/clients/{client id}	Deletes client	Implemented	Deletes invoice	✓	
LINE ITEMS						
create line item	http://localhost:4000/create/lineitem	Creates one individual line item	Implemented	Create line item	✓	
get all line item	http://localhost:4000/lineitems	Gets all of them	Implemented	not implemented or needed	not implemented	NA
get one line item by invoice	http://localhost:4000/lineitems/{invoice id}	This gets the line items associated with the invoice	Implemented	displays line item in table on invoice	✓	
get one line item by id	http://localhost:4000/lineitems/{line item id}	This gets the line item by id for display	Implemented	not implemented or needed	not implemented	NA
edit line item by id	http://localhost:4000/lineitems/{line item id}	This edits the line item by id for display	Not applicable	Will come in future iterations	NA	NA
delete line item by id	http://localhost:4000/lineitems/{line item id}	This deletes the line item by id for display	Implemented	Deletes line item	✓	
AUTH						
Register	http://localhost:4000/register	Registers users username and password - while also set cookies	Implemented	Creates new user - then opens modal	✓	
login	http://localhost:4000/login	Logs user in with username and password	Implemented	Logs user in	✓	
get profile	http://localhost:4000/profile	Gets profile cookies to run the header	Implemented	Gets profile for header	✓	
logout	http://localhost:4000/logout	Clears the set-cookies	Implemented	Logs out and clears cookies	✓	

Overall automated testing

AUTOMATED JEST TESTS - sheet [HERE](#)

AUTOMATED JEST TESTS			
NAME	ROUTE	ROLE	Results
USER INFO			
create user info	http://localhost:4000/create/user	Adds user info to so it can be accessed by invoice	Test Suites: 1 passed, 1 total Tests: 1 passed, 1 total Snapshots: 0 total Time: 1.158 s
get all users info	http://localhost:4000/user/	Gets the user info to populate the address	Test Suites: 1 passed, 1 total Tests: 1 passed, 1 total Snapshots: 0 total Time: 1.241 s
edit user info	http://localhost:4000/user/	Edit user info incase details need to change	Test Suites: 1 passed, 1 total Tests: 1 passed, 1 total Snapshots: 0 total Time: 1.116 s
delete user info	http://localhost:4000/user/{user id}	deletes details to start fresh	Test Suites: 1 passed, 1 total Tests: 1 passed, 1 total Snapshots: 0 total Time: 1.107 s
INVOICE			
create invoice	http://localhost:4000/create/invoice	Create an invoice with id so all the details can be PUT in on save	Test Suites: 1 passed, 1 total Tests: 1 passed, 1 total Snapshots: 0 total Time: 1.109 s
get one invoice	http://localhost:4000/invoices/{invoice id}	Gets on invoice via ID - this is for viewing on at a time	Test Suites: 1 passed, 1 total Tests: 1 passed, 1 total Snapshots: 0 total Time: 1.094 s
get all invoice	http://localhost:4000/invoices	Gets all invoices for the dashboard table	Test Suites: 1 passed, 1 total Tests: 1 passed, 1 total Snapshots: 0 total Time: 1.028 s
edit invoice by id	http://localhost:4000/invoices/{invoice id}	Edits on invoice via ID - this is for viewing on at a time	Test Suites: 1 passed, 1 total Tests: 1 passed, 1 total Snapshots: 0 total Time: 1.544 s, estimated 2
delete invoice by id	http://localhost:4000/invoices/{invoice id}	Deletes on invoice via ID - this is for viewing on at a time	Test Suites: 1 passed, 1 total Tests: 1 passed, 1 total Snapshots: 0 total Time: 1.134 s
CLIENTS			
create client	http://localhost:4000/create/client	Creates clients details	Test Suites: 1 passed, 1 total Tests: 1 passed, 1 total Snapshots: 0 total Time: 3.47 s
get all clients	http://localhost:4000/clients	Gets them for the drop down in the invoice modal	Test Suites: 1 passed, 1 total Tests: 1 passed, 1 total Snapshots: 0 total Time: 1.021 s
get one client	http://localhost:4000/clients/{client id}	Gets one to display invoice when viewing it	Test Suites: 1 passed, 1 total Tests: 1 passed, 1 total Snapshots: 0 total Time: 1.131 s
edit client by id	http://localhost:4000/clients/{client id}	Edit client details	Test Suites: 1 passed, 1 total Tests: 1 passed, 1 total Snapshots: 0 total Time: 1.159 s
delete client by id	http://localhost:4000/clients/{client id}	delete client, so users can click the bin in the table to delete invoice	Test Suites: 1 passed, 1 total Tests: 1 passed, 1 total Snapshots: 0 total Time: 1.148 s

LINE ITEMS			
create line item	http://localhost:4000/create/lineitem	creates one individual line item	Test Suites: 1 passed, 1 total Tests: 1 passed, 1 total Snapshots: 0 total Time: 1.108 s
get all line item	http://localhost:4000/lineitems	gets all of them	Test Suites: 1 passed, 1 total Tests: 1 passed, 1 total Snapshots: 0 total Time: 1.034 s
get one line item by id	http://localhost:4000/lineitems/{line item}	This gets the line item by id for display	Test Suites: 1 passed, 1 total Tests: 1 passed, 1 total Snapshots: 0 total Time: 1.124 s
edit line item by id	http://localhost:4000/lineitems/{line item}	This edits the line item by id for display	Test Suites: 1 passed, 1 total Tests: 1 passed, 1 total Snapshots: 0 total Time: 1.105 s
delete line item by id	http://localhost:4000/lineitems/{line item}	This deletes the line item by id for display	Test Suites: 1 passed, 1 total Tests: 1 passed, 1 total Snapshots: 0 total Time: 1.108 s
AUTH			
Register	http://localhost:4000/register	Registers users username and password - while also sets cookies	Test Suites: 1 passed, 1 total Tests: 1 passed, 1 total Snapshots: 0 total Time: 1.179 s, estimated 2 s
login	http://localhost:4000/login	Logs user in with username and password	Test Suites: 1 passed, 1 total Tests: 1 passed, 1 total Snapshots: 0 total Time: 1.166 s
get profile	http://localhost:4000/profile	Gets profile cookies to run the header	Test Suites: 1 passed, 1 total Tests: 1 passed, 1 total Snapshots: 0 total Time: 1.148 s
logout	http://localhost:4000/logout	Clears the set-cookies	Test Suites: 1 passed, 1 total Tests: 1 passed, 1 total Snapshots: 0 total Time: 1.108 s

✓ T3A2-B-SERVER

```

"scripts": {
  "start": "node server.js",
  "dev": "nodemon server.js",
  "test": "jest --detectOpenHandles",
  "test:createClients": "jest __tests__/clientTests/createClient.test.js --detectOpenHandles",
  "test:getAllClients": "jest __tests__/clientTests/getAllClient.test.js --detectOpenHandles",
  "test:getOneClient": "jest __tests__/clientTests/getOneClient.test.js --detectOpenHandles",
  "test:deleteClient": "jest __tests__/clientTests/deleteClient.test.js --detectOpenHandles",
  "test:editClient": "jest __tests__/clientTests/editClient.test.js --detectOpenHandles",
  "test:createInvoice": "jest __tests__/invoiceTests/createInvoice.test.js --detectOpenHandles",
  "test:deleteInvoice": "jest __tests__/invoiceTests/deleteInvoice.test.js --detectOpenHandles",
  "test:editInvoice": "jest __tests__/invoiceTests/editInvoice.test.js --detectOpenHandles",
  "test:getAllInvoice": "jest __tests__/invoiceTests/getAllInvoice.test.js --detectOpenHandles",
  "test:getOneInvoice": "jest __tests__/invoiceTests/getOneInvoice.test.js --detectOpenHandles",
  "test:createLineItem": "jest __tests__/lineItemTests/createLineItem.test.js --detectOpenHandles",
  "test:deleteLineItem": "jest __tests__/lineItemTests/deleteLineItem.test.js --detectOpenHandles",
  "test:editLineItem": "jest __tests__/lineItemTests/editLineItem.test.js --detectOpenHandles",
  "test:getAllLineItem": "jest __tests__/lineItemTests/getAllLineItem.test.js --detectOpenHandles",
  "test:getOneLineItem": "jest __tests__/lineItemTests/getOneLineItem.test.js --detectOpenHandles",
  "test:createUser": "jest __tests__/userInfoTests/createUserInfo.test.js --detectOpenHandles",
  "test:deleteUser": "jest __tests__/userInfoTests/deleteUserInfo.test.js --detectOpenHandles",
  "test:editUser": "jest __tests__/userInfoTests/editUserInfo.test.js --detectOpenHandles",
  "test:getUser": "jest __tests__/userInfoTests/getUser.test.js --detectOpenHandles",
  "test:login": "jest __tests__/userInfoTests/login.test.js --detectOpenHandles",
  "test:logout": "jest __tests__/userInfoTests/logout.test.js --detectOpenHandles",
  "test:profile": "jest __tests__/userInfoTests/profile.test.js --detectOpenHandles",
  "test:register": "jest __tests__/userInfoTests/register.test.js --detectOpenHandles"
}, "test:Run Script|Debug Script": "/login.test.js --detectOpenHandles",
"test:logout": "jest __tests__/logout.test.js --detectOpenHandles",
"test:profile": "jest __tests__/profile.test.js --detectOpenHandles",
"test:register": "jest __tests__/register.test.js --detectOpenHandles"
}

```

User feature implementation tests.

INVOICE MAKER EXPECTATIONS - sheet [HERE](#)

1	Expectationlist	
2	Jobs the app does.	
3	Create client	✓
4	Delete client	✓
5	Add client to invoice	✓
6	Add date to invoice	✓
7	Sequential invoice number	✓
8	Add line items	✓
9	Total	✓
10	GST total	✓
11	Total + GST	✓
12	Download invoice to send	✓
13	Save	✓
14	Close invoice modal	✓
15	Update accounts user info	✓
16	Delete invoice	✓
17	Delete line items	✓
18	Delete client	✓
19	Register	✓
20	Login	✓
21	Change light or dark mode	✓
22	Greeting	✓
23	Logout	✓
24	Total revenue overall	✓
25	GST Total overall	✓
26	Rough income tax overall	✓
27	Total invoice overall	✓

User testing and agile iteration

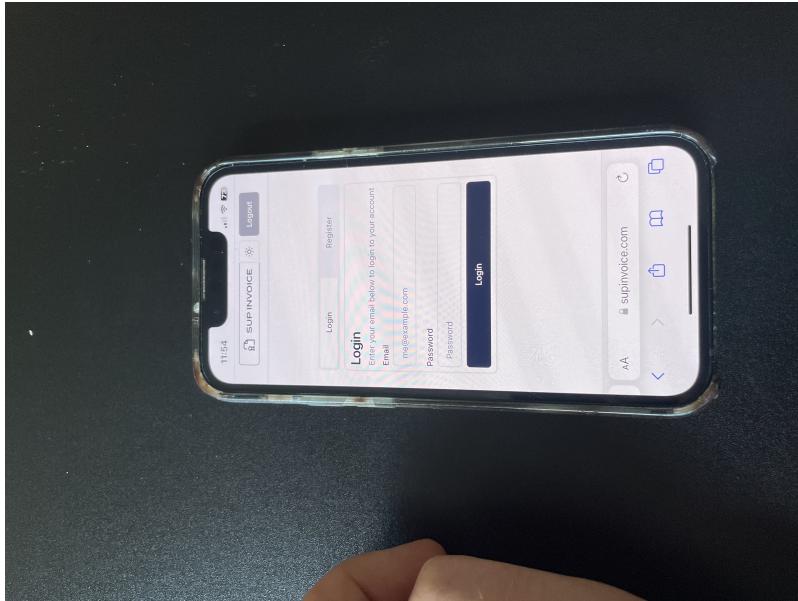
USER FEEDBACK - sheet [HERE](#)

User testing 1 & Feedback

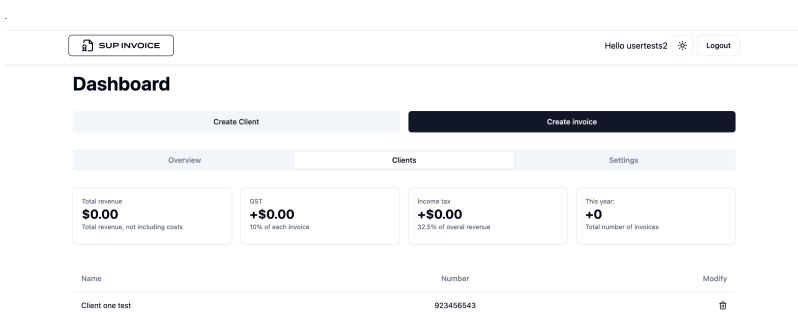
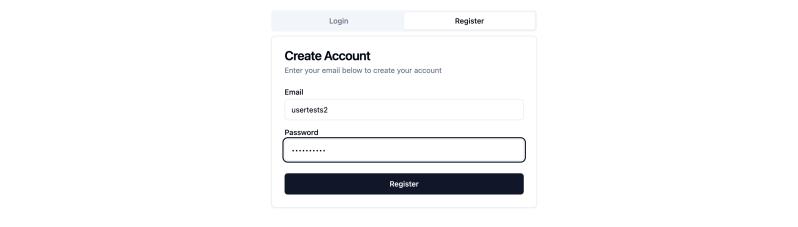
Feedback: "The heading doesnt have a background - a blurred background would look good"

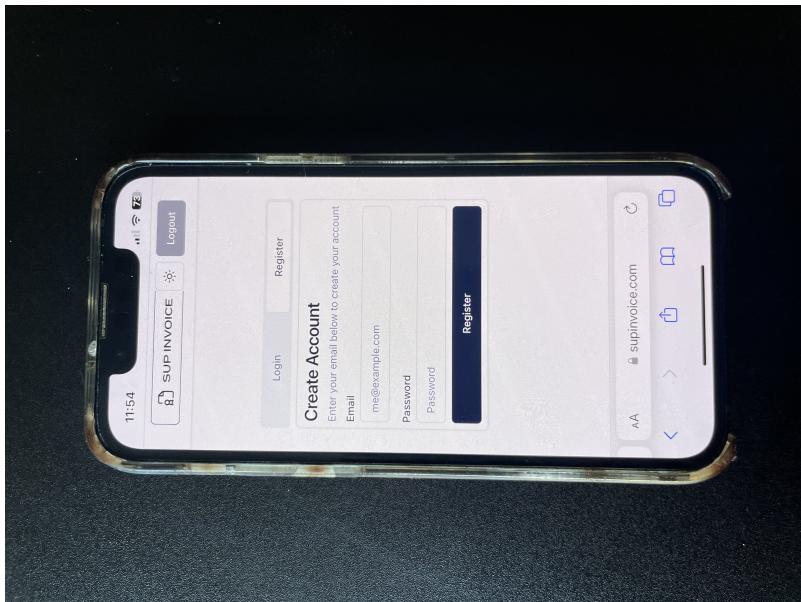
The screenshot shows the SUP INVOICE login interface. At the top, there's a header bar with a logo on the left and navigation links for 'Login', 'Register', and 'Logout' on the right. Below the header is a 'Login' form. The form has two input fields: 'Email' (containing 'me@example.com') and 'Password'. Below these fields is a large, solid black 'Login' button. The overall design is simple and functional.

Feedback: "The padding on the logo is causing it to cut off"

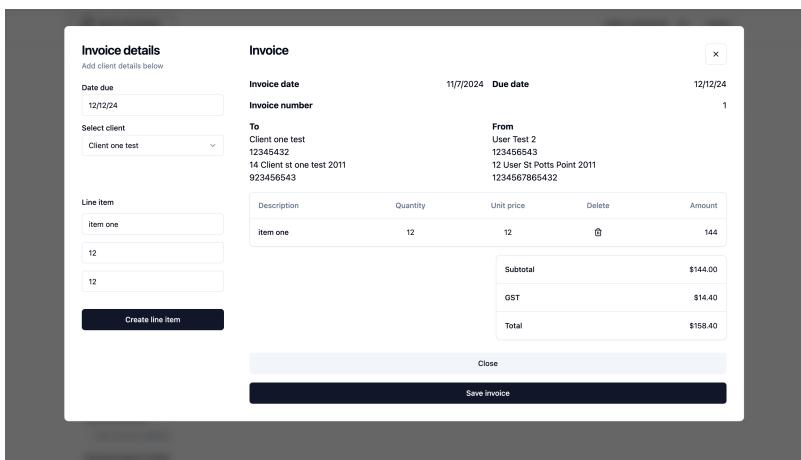


Feedback: "Changing between login and register makes the whole box move, it looks strange"

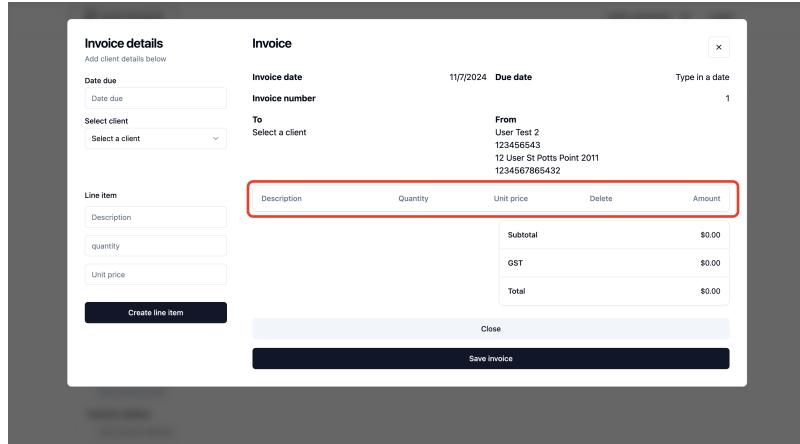




Feedback: "Empty table is confusing, it needs place holder text"



Feedback: "Empty table is confusing, it needs place holder text"

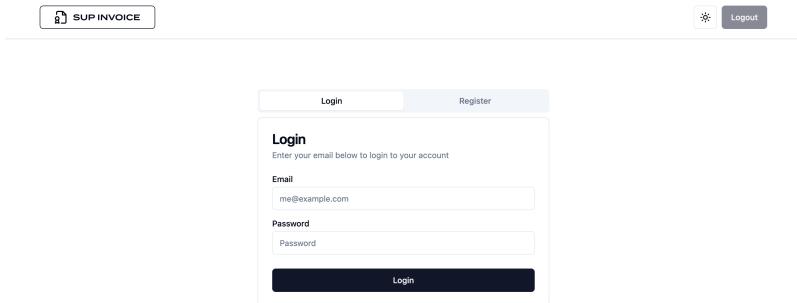


A screenshot of the application's dashboard. At the top, there are buttons for "Create Client" and "Create invoice" (which is highlighted in dark blue). Below these are three cards: "Overview" (Total revenue \$0.00, Total revenue, not including costs), "Clients" (GST +\$0.00, 10% of each invoice), and "Settings" (Income tax +\$0.00, 32.5% of overall revenue). A modal window titled "Dashboard" is open, containing a table for "Client" entries. A single row is highlighted with a red box. The table has columns for "Date", "Client", "Modify", and "Amount" (\$0.00). At the bottom of the modal are "Close" and "Save invoice" buttons.

Feedback: "these notifications don't look professional or premium"

Two screenshots of modal notifications from the localhost:5173 server. The top one, titled "localhost:5173 says", displays the message "Login failed" with an "OK" button. The bottom one, also titled "localhost:5173 says", displays the message "Success creating a User" with an "OK" button. Both notifications have a dark background and a light blue "OK" button.

Feedback: "Home page is not friendly, it feels very corporate"



User testing 2 & Feedback

Implemented feedback: Placeholder data implemented:

The screenshot shows an 'Invoice details' section with a dropdown for 'Select client' containing 'Select a client'. Below it is a 'Line item' table with a single row for 'Placeholder' having 0 quantity and \$0.00 amount. At the bottom are buttons for 'Close' and 'Save invoice'.

Invoice details

Add client details below

Date due: Date due

Select client: Select a client

Invoice

Invoice date	11/7/2024	Due date	Type in a date		
Invoice number	1				
To	From				
Select a client	usernewone usernewone usernewone 123454321				
Line item	Description	Quantity	Unit price	Delete	Amount
	Placeholder	0	0	✖	\$0.00
					Subtotal \$0.00
					GST \$0.00
					Total \$0.00

Create line item

Close

Save invoice

Summary Metrics

- Total revenue: **\$0.00** Total revenue, not including costs
- GST: **+\$0.00** 10% of each invoice
- Income tax: **+\$0.00** 32.5% of overall revenue
- This year: **+1** Total number of invoices

Date	Client	Modify	Amount
00/00/00	Placeholder	✖	\$0.00
Total			\$0.00

Implemented feedback: Blurred background added to header:

The screenshot shows a blurred background in the header area, which is a common design pattern for user testing to indicate a change in the user interface.

DASHBOARD

SUP INVOICE

Hello manualtest

Create Client

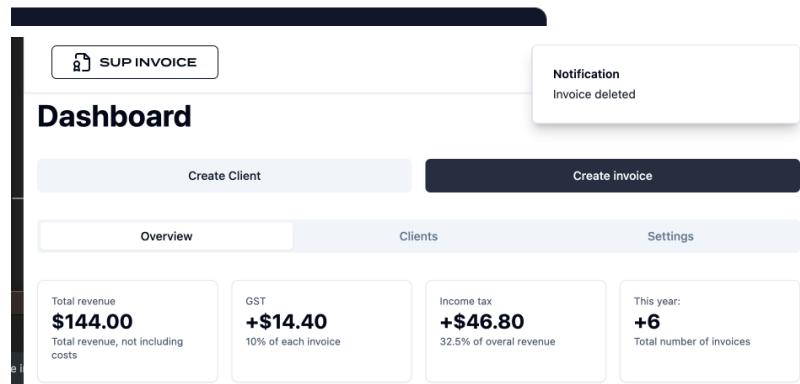
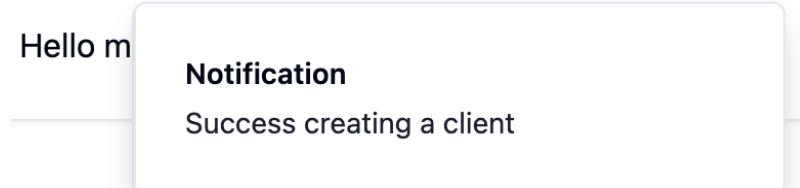
Create invoice

Overview **Clients** **Settings**

Summary Metrics

- Total revenue: **\$12.00** Total revenue, not including costs
- GST: **+\$1.20** 10% of each invoice
- Income tax: **+\$3.90** 32.5% of overall revenue
- This year: **+6** Total number of invoices

Implemented feedback: Alerts/notifcations/toast added:



Implemented feedback: Updated to have a friendly welcome message



Reference

- YouTube. Mocking JWT in Jest. Available at: <https://www.youtube.com/watch?v=HVPy-gNp5pc>.
- Stack Overflow. Mocking JSON Web Token Module with Jest. Available at: <https://stackoverflow.com/questions/64471965/mockng-jsonwebtoken-module-with-jest>.
- Medium. Express TypeScript: Properly Mocking JWT Verify in Unit Test. Available at: <https://zhifei-dev.medium.com/express-typescript-properly-mocking-jwt-verify-in-unit-test-b2dfd2e337a8>.
- Medium. Unit Tests with Jest. Available at: <https://medium.com/@fadime.ozdemir11/unit-tests-with-jest-364d61238420>.
- Jest. Using Matchers. Available at: <https://jestjs.io/docs/using-matchers>.
- YouTube. Testing with Jest and Supertest. Available at: https://www.youtube.com/watch?v=IDjF6-s1hGk&list=PL0X6fGhFFNTd5_wsAMasuLarx_VSkqYYX&index=6.
- YouTube. JWT Authentication in Node.js. Available at: <https://www.youtube.com/watch?v=mYy-d6BtqmU>.
- YouTube. Understanding Node.js Authentication. Available at: <https://www.youtube.com/watch?v=xKs2IZZya7c&t=5571s>.
- YouTube. Express.js Middleware Explained. Available at: <https://www.youtube.com/watch?v=-42K44A1oMA>.
- YouTube. Understanding Express.js Middleware. Available at: <https://www.youtube.com/watch?v=wcTzIJi2Oz4&list=PL4cUxeGkcC9h1NXLUuiAQ7c4UtdEInqma>.
- shadcn. Installation with Vite. Available at: <https://ui.shadcn.com/docs/installation/vite>.
- YouTube. Getting Started with Tailwind CSS. Available at: <https://www.youtube.com/watch?v=gXSC5eMw68o>.

- Aceternity UI. Components Library. Available at: <https://ui.aceternity.com/components>.
- YouTube. Advanced React Hooks Tutorial. Available at: https://www.youtube.com/watch?v=cCQZ0_y-4Os&list=PL4cUxeGkcC9h1NXLUuiAQ7c4UtdEInqma&index=4.
- YouTube. Introduction to Tailwind CSS. Available at: <https://www.youtube.com/watch?v=oidnyW71W0A>.
- YouTube. Tailwind CSS Tutorial. Available at: <https://www.youtube.com/watch?v=vGiYuaquCo4>.
- shadcn. UI Components. Available at: <https://ui.shadcn.com/docs/components>.
- YouTube. Flexbox Tutorial for Beginners. Available at: <https://www.youtube.com/watch?v=XkOXNIHJP6M&t=1017s>.
- YouTube. Advanced Flexbox Techniques. Available at: https://www.youtube.com/watch?v=UND-kib_iw4.
- Tailwind CSS. Flex Basis. Available at: <https://tailwindcss.com/docs/flex-basis>.
- Transform Tools. TypeScript to JavaScript. Available at: <https://transform.tools/typescript-to-javascript>.
- Stack Overflow. Converting TypeScript to JavaScript. Available at: <https://stackoverflow.com/questions/69696825/converting-typescript-to-javascript>.
- Tailwind UI. Modal Dialogs. Available at: <https://tailwindui.com/components/application-ui/overlays/modal-dialogs>.
- Flowbite React. Modal Components. Available at: <https://flowbite-react.com/docs/components/modal>.
- Crozat, B. Dialog Backdrop Styling with Tailwind CSS. Available at: <https://benjamincrozat.com/dialog-backdrop-styling-tailwind-css>.
- Creative Tim. Tailwind Starter Kit Documentation. Available at: <https://www.creative-tim.com/learning-lab/tailwind-starter-kit/documentation/react/modals/regular>.
- Lucide. X Icon. Available at: <https://lucide.dev/icons/x>.
- Tailwind CSS. Grid Template Columns. Available at: <https://tailwindcss.com/docs/grid-template-columns>.
- Medium. ReactJS MUI5 Color Themes with Dark/Light Modes. Available at: <https://medium.com/@facinick/reactjs-mui5-color-themes-with-dark-light-modes-theme-color-design-27039fd3d5de>.
- Pugazhenthi. Vite React Tailwind Shadcn. Available at: <https://www.pugazhenthi.in/blog/vite-react-tailwind-shadcn>.
- MDN Web Docs. Event.stopPropagation(). Available at: <https://developer.mozilla.org/en-US/docs/Web/API/Event/stopPropagation>.
- GeeksforGeeks. How to Use Calc in Tailwind CSS. Available at: <https://www.geeksforgeeks.org/how-to-use-calc-in-tailwind-css/>.
- Stack Overflow. React Open/Close Modal Component onClick. Available at: <https://stackoverflow.com/questions/67090762/react-open-close-modal-component-onclick>.
- Pluralsight. How to Open and Close React Bootstrap Modal Programmatically. Available at: <https://www.pluralsight.com/resources/blog/guides/how-to-open-and-close-react-bootstrap-modal-programmatically>.
- Medium. How to Get a React Bootstrap Modal to Open and Close Using a Button. Available at: <https://medium.com/nerd-for-tech/how-to-get-a-react-bootstrap-modal-to-open-and-close-using-a-button-of-your-choice-in-a-p-cde3ef6c0d4c>.
- YouTube. React Modal Tutorial. Available at: <https://www.youtube.com/watch?v=mhjbACbSeSU>.
- YouTube. Advanced React Modals. Available at: <https://www.youtube.com/watch?v=WkREeDy2WQ4>.
- React Router. Getting Started Tutorial. Available at: <https://reactrouter.com/en/main/start/tutorial>.

- GitHub. React Router Examples. Available at: <https://github.com/remix-run/react-router/tree/dev/examples>.
- Medium. Using useLocation Hook in React Router. Available at: https://medium.com/@alexanie_/https-ocxigin-hashnode-dev-uselocation-hook-in-react-router-758a0a711308.
- React Router. useLocation Hook. Available at: <https://reactrouter.com/en/main/hooks/use-location>.
- Medium. React Router: Get Current Route. Available at: <https://medium.com/codingbeauty-tutorials/react-router-get-current-route-9c2e6bd8d689>.
- Medium. Understanding React Outlet. Available at: <https://medium.com/@shruti.latthe/understanding-react-outlet-a-comprehensive-guide-b122b1e5e7ff>.
- Stack Overflow. React Router: Purpose of Using Outlet Alone. Available at: <https://stackoverflow.com/questions/73573071/react-router-what-is-the-purpose-of-using-outlet-alone-without-context>.
- React Router. Outlet Component. Available at: <https://reactrouter.com/en/main/components/outlet>.
- MUI. Material-UI Integrations with Routing. Available at: <https://mui.com/material-ui/integrations/routing/#tabs>.
- Medium. React JS Localhost with HTTPS Using mkcert. Available at: <https://medium.com/projectwt/react-js-localhost-with-https-using-mkcert-338927e94212>.
- YouTube. Understanding Node.js Authentication. Available at: <https://www.youtube.com/watch?v=xKs2IZZya7c&t=9120s>.
- YouTube. MongoDB Quick Start. Available at: <https://www.youtube.com/watch?v=-NfsmF-6BH0>.
- MongoDB. Quick Start: How to Get Connected to Your Database. Available at: <https://www.mongodb.com/blog/post/quick-start-nodejs-mongodb-how-to-get-connected-to-your-database>.
- YouTube. Express.js Middleware Explained. Available at: <https://www.youtube.com/watch?v=-42K44A1oMA>.
- YouTube. Node.js Authentication Tutorial. Available at: <https://www.youtube.com/watch?v=DZBGEVgL2eE&t=30s>.
- LoginRadius. Node.js Authentication Guide. Available at: <https://www.loginradius.com/blog/engineering/guest-post/nodejs-authentication-guide/>.
- Medium. How to Check if a String is a Valid MongoDB ObjectId in Node.js. Available at: <https://medium.com/@emmaw4430/how-to-check-if-a-string-is-a-valid-mongodb-objectid-in-node-js-283361445e10>.
- npm. bcrypt. Available at: <https://www.npmjs.com/package/bcrypt>.
- GitHub. node-jsonwebtoken. Available at: <https://github.com/auth0/node-jsonwebtoken>.
- YouTube. JWT Authentication in Node.js. Available at: <https://www.youtube.com/watch?v=mYy-d6BtqmU>.
- YouTube. Understanding JWT in Node.js. Available at: <https://www.youtube.com/watch?v=vgocTvahk18>.
- Medium. Things to Note When Signing and Verifying JWT Token in Node.js. Available at: <https://medium.com/@ardipurba/things-to-note-when-signing-and-verifying-jwt-token-in-nodejs-fad0ee4cbfad>.
- MDN Web Docs. async function. Available at: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/async_function.

- MongoDB. findOneAndUpdate. Available at:
<https://www.mongodb.com/docs/manual/reference/method/db.collection.findOneAndUpdate/>.
- GitHub. Contact Model Example. Available at: <https://github.com/AlexStormwood/oct23-t3w11-mongooseintro/blob/main/src/models/ContactModel.js>.
- Express.js. Session Middleware. Available at:
<https://expressjs.com/en/resources/middleware/session.html>.
- YouTube. Node.js User Authentication Example. Available at: <https://www.youtube.com/watch?v=MrEoixi8QY4&list=PL4cUxeGkcC9g8OhpOZxNdhXggFz2lOuCT&index=14>.
- Stack Overflow. How to Make a Page Only Accessible When Logged in with Express EJS. Available at: <https://stackoverflow.com/questions/61669346/how-to-make-a-page-only-accessible-when-logged-in-with-express-ejs>.
- Dev.to. Fix Cannot Set Headers After They Are Sent to the Client. Available at:
<https://dev.to/collegewap/fix-cannot-set-headers-after-they-are-sent-to-the-client-19gk>.
- MongoDB. BSON Type Date. Available at: <https://www.mongodb.com/docs/manual/reference/bson-types/#document-bson-type-date>.
- Mongoose. Mongoose Dates Tutorial. Available at: <https://mongoosejs.com/docs/tutorials/dates.html>.
- Stack Overflow. Difference Between Date.now() and new Date() in Mongoose. Available at:
<https://stackoverflow.com/questions/35501273/what-is-the-difference-between-date-now-and-date-now-in-mongoose>.
- Stack Overflow. How to Properly Embed Documents in Mongoose. Available at:
<https://stackoverflow.com/questions/43429500/how-do-i-properly-embed-documents-in-mongoose>.
- Medium. When to Use Embedded vs Reference Schema in Mongoose. Available at:
<https://medium.com/@truong.minh.michael/when-to-use-embedded-vs-reference-schema-in-mongoose-and-when-to-use-single-reference-vs-reference-36b757acab7e>.
- GeeksforGeeks. Mongoose insertMany Function. Available at:
<https://www.geeksforgeeks.org/mongoose-insertmany-function/>.
- GeeksforGeeks. Mongoose Document Model insertMany API. Available at:
<https://www.geeksforgeeks.org/mongoose-document-model-insertmany-api/>.
- GeeksforGeeks. How Query.prototype.map Works in Mongoose. Available at:
<https://www.geeksforgeeks.org/how-does-query-prototype-map-works-in-mongoose/>.
- Stack Overflow. Fetching Documents from Mongoose Inside a map Function Returns Promise Pending. Available at: <https://stackoverflow.com/questions/69780763/fetching-documents-from-mongoose-inside-a-map-function-returns-promise-pending-a>.
- YouTube. Mongoose with Node.js. Available at: <https://www.youtube.com/watch?v=lY6icfhap2o&t=309s>.
- Express.js. Writing Middleware. Available at: <https://expressjs.com/en/guide/writing-middleware.html>.
- MDN Web Docs. HTTP Status Codes. Available at: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Status>.
- Dev.to. Testing with MongoDB Memory Server. Available at: <https://dev.to/remrkabledev/testing-with-mongodb-memory-server-4ja2>.
- GitHub. MongoDB Memory Server Releases. Available at: <https://github.com/typegoose/mongodb-memory-server/releases>.
- npm. mongodb-memory-server. Available at: <https://www.npmjs.com/package/mongodb-memory-server>.

- Easy on the Web. In-Memory MongoDB for Testing. Available at: <https://easyontheweb.com/in-memory-mongodb-for-testing/>.
- Dev.to. Testing Node.js Mongoose with an In-Memory Database. Available at: <https://dev.to/paulasantamaria/testing-node-js-mongoose-with-an-in-memory-database-32np>.
- Medium. Jest Test Lifecycle and Test Environment. Available at: <https://medium.com/@yiqun.rong2/jest-test-lifecycle-and-test-environment-06cfbf90f612>.
- Medium. Integration and Unit Testing with Jest in Node.js and Mongoose. Available at: <https://medium.com/@tehvicke/integration-and-unit-testing-with-jest-in-nodejs-and-mongoose-bd41c61c9fbc>.
- Medium. Unit Tests in Jest with Supertest and MongoDB. Available at: <https://medium.com/@adammalej/unit-tests-in-jest-with-supertest-and-mongodb-e4d56e918ce8>.
- Jest. Mock Functions. Available at: <https://jestjs.io/docs/mock-functions>.
- GitHub. express_jest_and_mocks Repository. Available at: https://github.com/Sam-Meech-Ward/express_jest_and_mocks.
- GitHub. express_jest_and_mocks - app.test.js. Available at: https://github.com/Sam-Meech-Ward/express_jest_and_mocks/blob/master/app.test.js.
- YouTube. Jest and Supertest for Testing Node.js Apps. Available at: <https://www.youtube.com/watch?v=r5L1XRZaCR0>.
- DigitalOcean. Node.js JWT Express.js Tutorial. Available at: <https://www.digitalocean.com/community/tutorials/nodejs-jwt-expressjs>.
- LinkedIn. JWT Authentication Middleware in Express.js. Available at: <https://www.linkedin.com/pulse/jwt-authentication-middleware-expressjs-building-secure-tamjid-ahmed/>.
- Medium. Express.js Testing: Mocking MongoDB. Available at: <https://medium.com/weekly-webtips/express-js-testing-mocking-mongodb-46c3797a201>.
- GitHub. express_jest_and_mocks - app.js. Available at: https://github.com/Sam-Meech-Ward/express_jest_and_mocks/blob/master/app.js.
- Stack Overflow. Mocking Express Request with Jest and TypeScript Using Correct Types. Available at: <https://stackoverflow.com/questions/57964299/mocking-express-request-with-jest-and-typescript-using-correct-types>.
- Dev.to, 'ExpressJS: Handling Cross-Origin Cookies'. Available at: <https://dev.to/alexmercedcoder/expressjs-handling-cross-origin-cookies-38l9>
- Reddit, 'How to Avoid Getting CORS Error in Production'. Available at: https://www.reddit.com/r/reactjs/comments/15s76rw/how_to_avoid_getting_cors_error_in_production/
- Stack Overflow, 'Issue with Cross-Site Cookies: How to Set Cookie from Backend to Frontend'. Available at: <https://stackoverflow.com/questions/63010545/issue-with-cross-site-cookies-how-to-set-cookie-from-backend-to-frontend>
- Reddit, 'Set-Cookie Header Was Blocked Because it Had the "Secure" Attribute but Was Not Encrypted'. Available at: https://www.reddit.com/r/reactjs/comments/16z3e5d/help_setcookie_header_was_blocked_because_it_had/
- MDN Web Docs, 'Set-Cookie'. Available at: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie>
- PortSwigger, 'Bypassing SameSite Restrictions'. Available at: <https://portswigger.net/web-security/csrf/bypassing-samesite-restrictions>

- Stack Overflow, 'Node.js Express CORS Not Working, Pending Options Requests'. Available at: <https://stackoverflow.com/questions/46742605/node-js-express-cors-not-working-pending-options-requests>
- Chrome DevTools, 'Network Reference'. Available at: https://developer.chrome.com/docs/devtools/network/reference/?utm_source=devtools#timing-explanation
- Google Cloud Community, 'Session Cookie Won't Be Set Due to Domain Name Mismatch'. Available at: <https://www.googlecloudcommunity.com/gc/Serverless/Session-cookie-won-t-be-set-due-to-domain-name-mis-match-despite/m-p/490932>
- Stack Overflow, 'How Does the Access-Control-Allow-Origin Header Work?'. Available at: <https://stackoverflow.com/questions/10636611/how-does-the-access-control-allow-origin-header-work>
- Medium, '3 Ways to Fix the CORS Error and How Access-Control-Allow-Origin Works'. Available at: <https://medium.com/@dtkatz/3-ways-to-fix-the-cors-error-and-how-access-control-allow-origin-works-d97d55946d9>
- MDN Web Docs, 'Cross-Origin Resource Sharing (CORS)'. Available at: <https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>
- Reddit, 'Error: Not Allowed by CORS on Render'. Available at: https://www.reddit.com/r/learnprogramming/comments/18zxfg3/error_not_allowed_by_cors_on_renderer/
- Amazon Web Services, 'How to Enable CORS in API Gateway'. Available at: <https://docs.aws.amazon.com/apigateway/latest/developerguide/how-to-cors.html>
- Medium, 'React JS: Localhost with HTTPS Using mkcert'. Available at: <https://medium.com/projectwt/react-js-localhost-with-https-using-mkcert-338927e94212>
- YouTube, 'ReactJS: Handling CORS Issues'. Available at: <https://www.youtube.com/watch?v=l134cBAJCuc&t=131s>
- YouTube, 'ReactJS: Cross-Origin Cookies'. Available at: <https://www.youtube.com/watch?v=0XSDAup85SA>
- MUI, 'React Select Component'. Available at: <https://mui.com/material-ui/react-select/>
- YouTube, 'ReactJS: React Select Component'. Available at: <https://www.youtube.com/watch?v=oGq9o2Bxlal>
- YouTube, 'ReactJS: Get Current Date'. Available at: <https://www.youtube.com/watch?v=u-m0XINQUy&t=2s>
- MDN Web Docs, 'Array.prototype.find()'. Available at: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/find
- Medium, 'Get Current Date, Month, Year in React JS'. Available at: <https://web-tuts.medium.com/get-current-date-month-year-in-react-js-a45c067ab190>
- Stack Overflow, 'React JS: Get Current Date'. Available at: <https://stackoverflow.com/questions/43744312/react-js-get-current-date>
- Medium, 'Get Current Date in React JS'. Available at: <https://webtuts.medium.com/get-current-date-in-react-js-55283583186a>
- SheCodes, 'How to Get the Current Date in React'. Available at: [https://www.shecodes.io/athena/7466-how-to-get-current-date-in-react#:~:text=To%20get%20the%20current%20date%20in%20React%2C%20you,use%20the%20Date\(\)%20function.](https://www.shecodes.io/athena/7466-how-to-get-current-date-in-react#:~:text=To%20get%20the%20current%20date%20in%20React%2C%20you,use%20the%20Date()%20function.)

- Medium, 'How to Pre-populate Inputs When Editing Forms in React'. Available at: <https://medium.com/@vanthedeve/how-to-pre-populate-inputs-when-editing-forms-in-react-2530d6069ab3>
- Stack Overflow, 'How to Programmatically Fill Input Field Value with React'. Available at: <https://stackoverflow.com/questions/44671082/how-do-i-programmatically-fill-input-field-value-with-react>
- freeCodeCamp, '4 Methods to Search an Array'. Available at: <https://www.freecodecamp.org/news/4-methods-to-search-an-array/>
- DigitalOcean, 'React Select Component'. Available at: <https://www.digitalocean.com/community/tutorials/react-react-select>
- Mabl, 'Interacting with HTML Select Dropdowns'. Available at: <https://help.mabl.com/hc/en-us/articles/19078186418324-Interacting-with-HTML-select-dropdowns>
- W3C, 'ARIA: Accessible Rich Internet Applications'. Available at: <https://w3c.github.io/aria/#aria-hidden>
- Medium, 'Handling Browser Refresh in React: Best Practices and Pitfalls'. Available at: <https://medium.com/@stheodorejohn/handling-browser-refresh-in-react-best-practices-and-pitfalls-5d4451d579ff>
- LogRocket, 'Understanding React useEffect Cleanup Function'. Available at: <https://blog.logrocket.com/understanding-react-useeffect-cleanup-function/>
- Jason Watmore, 'Fetch HTTP DELETE Request Examples'. Available at: <https://jasonwatmore.com/post/2021/09/21/fetch-http-delete-request-examples>
- Dev.to, 'React Fetch Example: GET, POST, PUT, DELETE with API'. Available at: <https://dev.to/collegewap/react-fetch-example-getpostputdelete-with-api-3l00>
- Express.js, 'CORS Middleware'. Available at: <https://expressjs.com/en/resources/middleware/cors.html>
- React, 'Updating Arrays in State'. Available at: <https://react.dev/learn/updating-arrays-in-state>
- freeCodeCamp, 'How to Render Lists in React'. Available at: <https://www.freecodecamp.org/news/how-to-render-lists-in-react/>
- Scrimba, 'React List Array with map() Function'. Available at: <https://scrimba.com/articles/react-list-array-with-map-function/>
- Dev.to, 'Creating a Dynamic Table in React Using react-table v7.8.0'. Available at: <https://dev.to/hcoco1/creating-a-dynamic-table-in-react-using-react-table-v780-3823>
- GeeksforGeeks, 'How to Pass Data into Table from a Form Using React Components'. Available at: <https://www.geeksforgeeks.org/how-to-pass-data-into-table-from-a-form-using-react-components/>
- YouTube, 'ReactJS: Dynamic Tables in React'. Available at: <https://www.youtube.com/watch?v=xNWVkyDYKA4>
- YouTube, 'ReactJS: Append to State Array'. Available at: <https://www.youtube.com/watch?v=desktop&v=tkDym1sohI4>
- DhiWise, 'React Append to State Array: Understanding Immutability'. Available at: <https://www.dhiwise.com/post/react-append-to-state-array-understanding-immutability3>
- Stack Overflow, 'Call Multiple Functions onClick ReactJS'. Available at: <https://stackoverflow.com/questions/26069238/call-multiple-functions-onclick-reactjs>
- Stack Overflow. How to reload a component/part of page in ReactJS. Available at: <https://stackoverflow.com/questions/56649094/how-to-reload-a-component-part-of-page-in-reactjs>
- YouTube. Nxe-9PkP8Nw. Available at: <https://www.youtube.com/watch?v=Nxe-9PkP8Nw>.

- FreeCodeCamp Forum. React: Correct way to handle a component refresh on post request. Available at: <https://forum.freecodecamp.org/t/react-correct-way-to-handle-a-component-refresh-on-post-request/502426/2>.
- React Documentation. Lifting State Up. Available at: <https://legacy.reactjs.org/docs/lifting-state-up.html>.
- React.dev. Sharing State Between Components. Available at: <https://react.dev/learn/sharing-state-between-components>.
- Medium. Using useNavigate in React Router 6: A complete guide. Available at: <https://medium.com/@bobjunior542/using-usenavigate-in-react-router-6-a-complete-guide-46f51403f430>.
- FreeCodeCamp. Refresh the page in JavaScript (JS): Reload window tutorial. Available at: <https://www.freecodecamp.org/news/refresh-the-page-in-javascript-js-reload-window-tutorial/>.
- YouTube. xKs2IZZya7c&t=5286s. Available at: <https://www.youtube.com/watch?v=xKs2IZZya7c&t=5286s>.
- FreeCodeCamp Forum. React: Use state to toggle an element. Available at: <https://forum.freecodecamp.org/t/react-use-state-to-toggle-an-element/620947/2>.
- FreeCodeCamp. How to format a date with JavaScript: Date formatting in JS. Available at: <https://www.freecodecamp.org/news/how-to-format-a-date-with-javascript-date-formatting-in-js/>.
- MDN Web Docs. Date.prototype.toLocaleString(). Available at: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date/toLocaleString.
- Eduative. How to sum a JavaScript array with reduce. Available at: <https://www.educative.io/answers/how-to-sum-a-javascript-array-with-reduce>.
- FreeCodeCamp Forum. Why is there a 0 in the function and what does reduce do?. Available at: <https://forum.freecodecamp.org/t/why-is-there-a-0-in-the-function-and-what-does-reduce-do/487065>.
- TutorialsPoint. How to format a number with two decimals in JavaScript. Available at: <https://www.tutorialspoint.com/how-to-format-a-number-with-two-decimals-in-javascript>.
- React SVGR. Playground. Available at: <https://react-svgr.com/playground/>.
- Stack Overflow. How to save img to user's local computer using html2canvas. Available at: <https://stackoverflow.com/questions/31656689/how-to-save-img-to-users-local-computer-using-html2canvas>.
- HTML2Canvas. Getting Started. Available at: <https://html2canvas.hertzen.com/getting-started>.
- npm. file-saver. Available at: <https://www.npmjs.com/package/file-saver>.
- Expo. captureRef - Expo Documentation. Available at: [https://docs.expo.dev/versions/latest/sdk/captureRef/](https://docs.expo.dev/versions/latest/sdk/captureRef).
- GitHub Gist. HTML2Canvas and saving images. Available at: <https://gist.github.com/origamid/772d5b7edc2b18cfa4f34107612520dd>.
- LogRocket. Export React components as images using HTML2Canvas. Available at: <https://blog.logrocket.com/export-react-components-as-images-html2canvas/>.
- npm. html2canvas version 1.4.1. Available at: <https://www.npmjs.com/package/html2canvas/v/1.4.1>.
- MDN Web Docs. HTMLCanvasElement.toBlob(). Available at: <https://developer.mozilla.org/en-US/docs/Web/API/HTMLCanvasElement/toBlob>.
- Sonner. Sonner - Toast Notifications. Available at: <https://sonner.emilkowal.ski/>.
- YouTube. 8JTrY1dIXCw. Available at: <https://www.youtube.com/watch?v=8JTrY1dIXCw>.
- YouTube. -42K44A1oMA&t=1127s. Available at: <https://www.youtube.com/watch?v=-42K44A1oMA&t=1127s>.

- Radix UI. Toast Component Documentation. Available at: <https://www.radix-ui.com/primitives/docs/components/toast>.
- GeeksforGeeks. How to connect Node.js to MongoDB Atlas using Mongoose. Available at: <https://www.geeksforgeeks.org/how-to-connect-node-js-to-mongodb-atlas-using-mongoose/>.
- MongoDB Documentation. Getting Started with MongoDB Atlas. Available at: <https://www.mongodb.com/docs/atlas/getting-started/>.
- Medium. Mastering the connection: A step-by-step guide to linking Node.js and MongoDB Atlas with Mongoose. Available at: <https://luqmanshaban.medium.com/mastering-the-connection-a-step-by-step-guide-to-linking-node-js-and-mongodb-atlas-with-mongoose-3941cfbbc2c7>.
- Mongoose Documentation. Available at: <https://mongoosejs.com/docs/>.
- MongoDB Developer Hub. Getting started with MongoDB and Mongoose. Available at: <https://www.mongodb.com/developer/languages/javascript/getting-started-with-mongodb-and-mongoose/>.
- npm. dotenv. Available at: <https://www.npmjs.com/package/dotenv>.
- npm. bcryptjs. Available at: <https://www.npmjs.com/package/bcryptjs>.
- npm. bcrypt. Available at: <https://www.npmjs.com/package/bcrypt>.
- npm. jsonwebtoken. Available at: <https://www.npmjs.com/package/jsonwebtoken>.
- JWT.io. JSON Web Tokens - Introduction. Available at: <https://jwt.io/>.
- React Documentation. React DOM. Available at: <https://legacy.reactjs.org/docs/react-dom.html>.
- React Router. Main Documentation. Available at: <https://reactrouter.com/en/main>.
- npm. mongodb-memory-server. Available at: <https://www.npmjs.com/package/mongodb-memory-server>.
- npm. mongodb-memory-server-global. Available at: <https://www.npmjs.com/package/mongodb-memory-server-global>.
- Stack Overflow. How to use file-saver.js to save text files. Available at: <https://stackoverflow.com/questions/50126062/how-to-use-filensaver-js-to-save-text-files>.
- npm. framer-motion. Available at: <https://www.npmjs.com/package/framer-motion>.
- Framer Motion. Spring Transitions. Available at: https://www.framer.com/motion/transition/#spring?utm_source=motion-readme-docs.
- Lucide. Lucide React Guide. Available at: <https://lucide.dev/guide/packages/lucide-react>.
- npm. lucide-react. Available at: <https://www.npmjs.com/package/lucide-react>.
- Headless UI. Available at: <https://headlessui.com/>.
- Jest. Available at: <https://jestjs.io/>.
- Jest. Getting Started Documentation. Available at: <https://jestjs.io/docs/getting-started>.
- Testim Blog. Jest Testing: A Helpful Introductory Tutorial. Available at: <https://www.testim.io/blog/jest-testing-a-helpful-introductory-tutorial/>.
- FreeCodeCamp. How to test JavaScript code with Jest. Available at: <https://www.freecodecamp.org/news/how-to-test-javascript-code-with-jest/>.
- MDN Web Docs. Array.prototype.length. Available at: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/length.
- W3Schools. JavaScript Array length Property. Available at: https://www.w3schools.com/jsref/jsref_length_array.asp.
- React Documentation. Introduction to Hooks. Available at: <https://legacy.reactjs.org/docs/hooks-intro.html>.
- React.dev. Reference for React Hooks. Available at: <https://react.dev/reference/react/hooks>.

- Bitsrc Blog. Build a simple modal component with React. Available at: <https://blog.bitsrc.io/build-a-simple-modal-component-with-react-16dec111a6>.
- LogRocket. Creating a reusable pop-up modal in React. Available at: <https://blog.logrocket.com/creating-reusable-pop-up-modal-react/>.
- GeeksforGeeks. How to use Modal Component in ReactJS. Available at: <https://www.geeksforgeeks.org/how-to-use-modal-component-in-reactjs/>.
- Render. No 'Access-Control-Allow-Origin' header. Available at: <https://community.render.com/t/no-access-control-allow-origin-header/12947/3>
- Secure, HttpOnly, SameSite HTTP Cookies Attributes and Set-Cookie Explained. Available at: <https://medium.com/swlh/secure-httponly-samesite-http-cookies-attributes-and-set-cookie-explained-fc3c753dfeb6>
- Catch all redirect for create-react-app in netlify. Available at: <https://stackoverflow.com/questions/55990467/catch-all-redirect-for-create-react-app-in-netlify>
- Redirects and rewrites. Available at: <https://docs.netlify.com/routing/redirects/#history-pushstate-and-single-page-apps>
- Deploying single page applications. Available at: <https://answers.netlify.com/t/deploying-single-page-applications/14695>