# Step 1

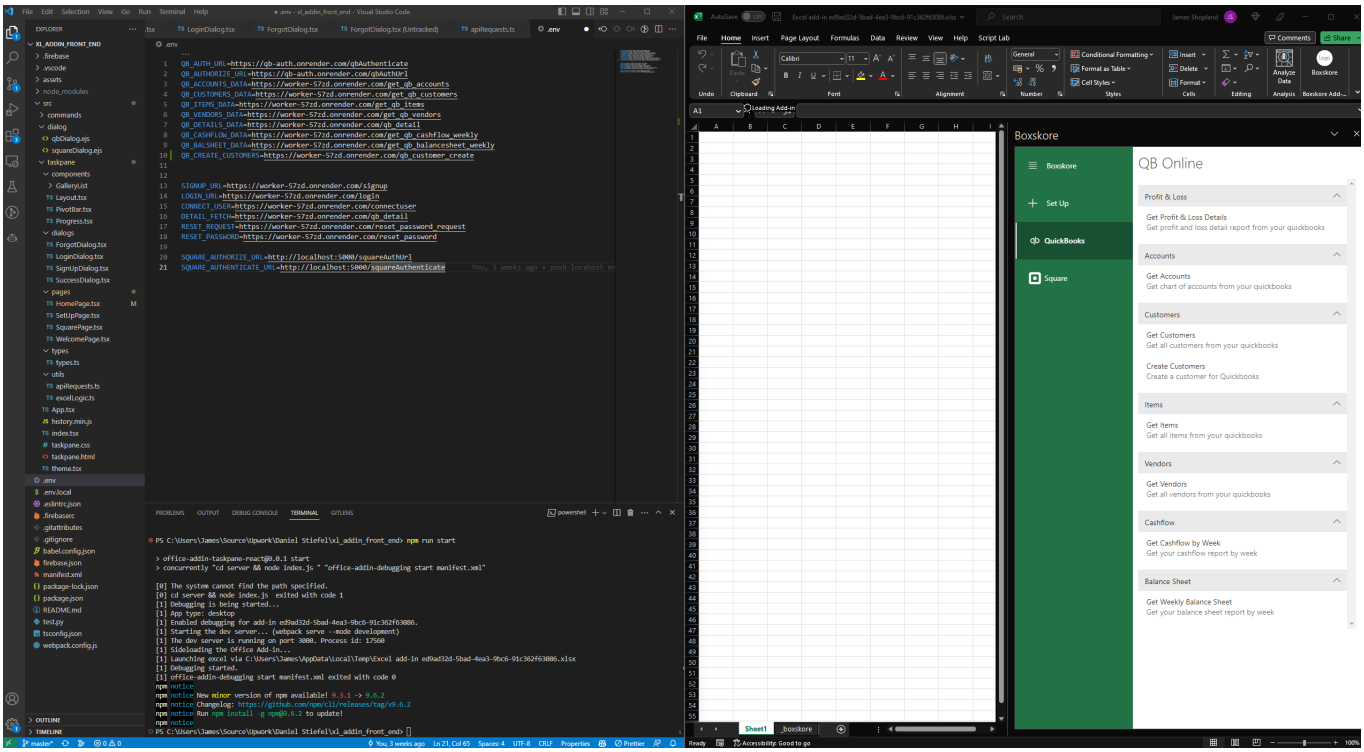
I recommend opening the excel app and VS code.

To do, so I assume you have run `npm install` in the folder. In a vs code terminal, from the `xl_addin_front_end` run:

```
npm run start
```

This should open a node server in a terminal window, and launch excel with the application already sideloaded.

You should see a screen like this:




# Step 2

Now you have the application running, go to the `QuickBooks` page. This is where you will be adding a button, so you should see any changes we make here.

Further, in VS code, open `HomePage.tsx` This is where the QuickBooks page logic and component is
(`src/taskpane/pages/HomePage.tsx`)

You should see the `Home` component (like below):

```
export default function Home() {
  const [loading, setLoading] = useState(false);
  const [loadingAccounts, setLoadingAccounts] = useState(false);
  const [loadingCustomer, setLoadingCustomers] = useState(false);
  const [loadingItems, setLoadingItems] = useState(false);
  const [loadingVendors, setLoadingVendors] = useState(false);
  const [loadingCashflow, setLoadingCashflow] = useState(false);
  const [balSheet, setBalSheet] = useState(false);
  const [loadingCreateCustomers, setLoadingCreateCustomers] = useState(false);
  const [hideDialog, { toggle: toggleHideDialog }] = useBoolean(true);
  const labelId: string = useId("dialogLabel");
  const subTextId: string = useId("subTextLabel");
  const today = useConst(new Date(Date.now()));       You, 2 weeks ago • add cas
  const [startDate, setStartDate] = useState<Date>();
  const [endDate, setEndDate] = useState<Date>(today);
  const [modalInitiator, setModalInitiator] = useState<string>("");
```

# Step 3

If you need to add a new header and a button, create a new `IGalleryListItem`. You will see examples in the `HomePage.tsx`, like below: (Explained later on how to fill in this object)

```
  const accountsItems: IGalleryListItem[] = [

    {
      key: "Get_Accounts",
      title: "Get Accounts",
      description: "Get chart of accounts from your quickbooks",
      onClick: () => {
        setLoadingAccounts(true);
        getAccountsData().finally(() => setLoadingAccounts(false));
      },
      showSpinner: loadingAccounts,
    },

  ];
```

If you are only adding to a already existing section, you just need to add a new object to the relevant array. You will see the GalleyList component that uses these objects at the bottom of the file, like so:

```
<GalleryList title="Profit & Loss" key="PnL" items={plItems} />
<GalleryList title="Accounts" key="accounts" items={accountsItems} />
<GalleryList title="Customers" key="customers" items={customerItems} />
<GalleryList title="Items" key="items" items={itemsItems} />
<GalleryList title="Vendors" key="vendors" items={vendorsItems} />
<GalleryList title="Cashflow" key="cashflow" items={cashflowItems} />
<GalleryList title="Balance Sheet" key="balsheet" items={balanceSheetItems} />
```

If you find the relevant items object, you can add an object to it, like so:

```
const customerItems: IGalleryListItem[] = [
  {
    key: "Get_Customers",
    title: "Get Customers",
    description: "Get all customers from your quickbooks",
    onClick: () => {
      setLoadingCustomers(true);
      getCustomers().finally(() => setLoadingCustomers(false));
    },
    showSpinner: loadingCustomer,
  },
  {
    key: "Create_Customers",
    title: "Create Customers",
    description: "Create a customer for Quickbooks",
    onClick: () => {
      setLoadingCreateCustomers(true);
      createCustomers().finally(() => setLoadingCreateCustomers(false));
    },
    showSpinner: loadingCreateCustomers,
  },
];
```

- Key should be unique across the app
- Title and Description are customisable to what is needed
- onClick - We will cover filling in this logic in a later step
- showSpinner - Cover this in Step 4

If you added a whole new section, Add a GalleryList component as well, like below.

```
<GalleryList title="Balance Sheet" key="balsheet" items={balanceSheetItems} />
```

# Step 4

We need to store whether or not the button is loading in a state. At the top of the Home component, add a new `useState` const similar to the ones below:

```
const [loading, setLoading] = useState(false);
const [loadingAccounts, setLoadingAccounts] = useState(false);
const [loadingCustomer, setLoadingCustomers] = useState(false);
const [loadingItems, setLoadingItems] = useState(false);
const [loadingVendors, setLoadingVendors] = useState(false);
const [loadingCashflow, setLoadingCashflow] = useState(false);
const [balSheet, setBalSheet] = useState(false);
const [loadingCreateCustomers, setLoadingCreateCustomers] = useState(false);
```

For example

```
const [loadingCreateAccounts, setLoadingCreateAccounts] = useState(false);
```

Making sure to set it to false by default.

Now in the object created in step 3, add the loading name to the `showSpinner` option, as well as setting it to `true` in the onClick. For example:

```
{
  key: "Create_Accounts",
  title: "Create Accounts",
  description: "Create accounts for your quickbooks",
  onClick: () => {
    setLoadingCreateAccounts(true);
  },      You, now • Uncommitted changes
  showSpinner: loadingCreateAccounts,
},
```

# Step 5

Now we need to add in the excel logic from the provided snippet.

Navigate to `src/taskpane/utils/excelLogic.ts`

At the bottom of the file, copy in the entire `async function getData()` function from the provided gist. You do not have to copy in anything else, so ignore the try-catch at the bottom.

e.g:

```
async function getData() {
  await Excel.run(async (context) => {
    console.log("hi");
    const sheet = context.workbook.worksheets.getItem("Create Accounts");
    const expensesTable = sheet.tables.getItem("CreateAccounts");
    // This is the sheet for the access_token
    const codeSheet = context.workbook.worksheets.getItem("_boxskore").getRange("B1").load("v
    await context.sync();
    // This is the access_token value
    const access_token = codeSheet["values"][0][0];
    // console.log(JSON.stringify(access_token));
    const headerRange = expensesTable.getHeaderRowRange().load("values");
    const bodyRange = expensesTable.getDataBodyRange().load("values");
    await sheet.context.sync();
    const accounts = { accounts: [] };
    // console.log(bodyRange, headerRange);
```

Now rename the getData function to something more relevant, and export it.
e.g

```
export async function createAccounts() {
  await Excel.run(async (context) => {
    console.log("hi");
    const sheet = context.workbook.worksheets.getItem("Create Accounts");
    const expensesTable = sheet.tables.getItem("CreateAccounts");
    // This is the sheet for the access_token
    const codeSheet = context.workbook.worksheets.getItem("_boxskore").getRange("B1").loa
    await context.sync();
    // This is the access_token value
    const access_token = codeSheet["values"][0][0];
    // console.log(JSON.stringify(access_token));
    const headerRange = expensesTable.getHeaderRowRange().load("values");
    const bodyRange = expensesTable.getDataBodyRange().load("values");
```

# Step 6

Back in the HomePage.tsx we were in earlier. Go back to the relevant items onClick
e.g

```
{
  key: "Create_Accounts",
  title: "Create Accounts",
  description: "Create accounts for your quickbooks",
  onClick: () => {
    setLoadingCreateAccounts(true);
  },      You, 5 minutes ago • Uncommitted changes
  showSpinner: loadingCreateAccounts,
},
```

Now in the onClick, after the setLoading function, call the function we just created, and then add

```
.finally(() => setLoading(false))
```

replacing setLoading with the one we created earler. E.g:

```
    {
      key: "Create_Accounts",
      title: "Create Accounts",
      description: "Create accounts for your quickbooks",
      onClick: () => {
        setLoadingCreateAccounts(true);
        createAccounts().finally(() => setLoadingCreateAccounts(false));
      },
      showSpinner: loadingCreateAccounts,
    },
```

Your button should now be linked to the excel function and be working.

# Step 7

Back in excelLogic.ts, I recommend removing direct references to a url in the code, and placing it in the env file:

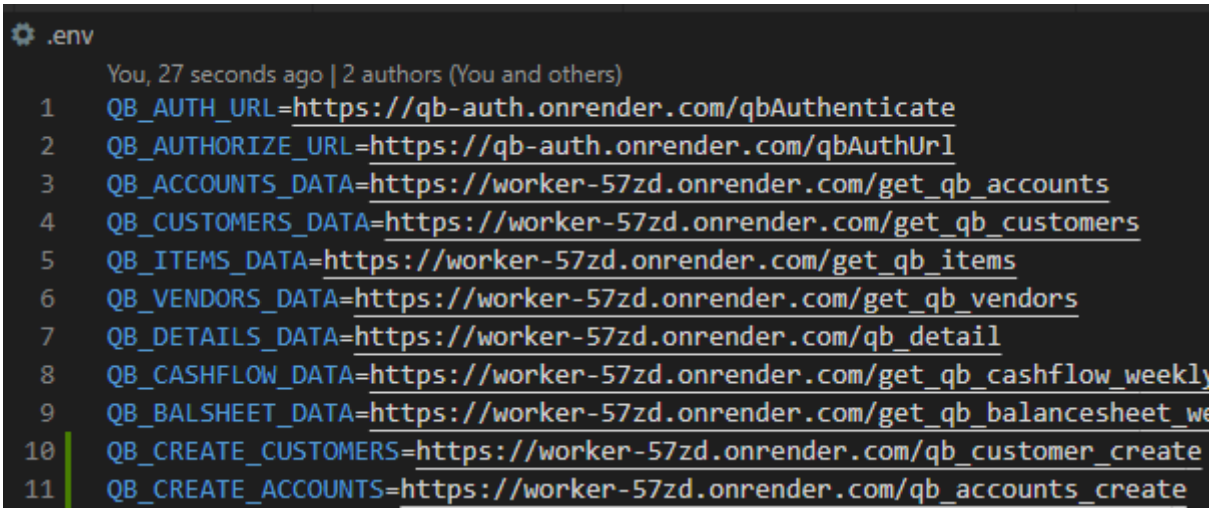
For example:

```
    fetch("https://worker-57zd.onrender.com/qb_accounts_create", options)
      .then((response) => response.json())
      .then((response) => console.log(response))
      .catch((err) => console.error(err));
```

Remove the url and replace it with a relevantly named env variable:

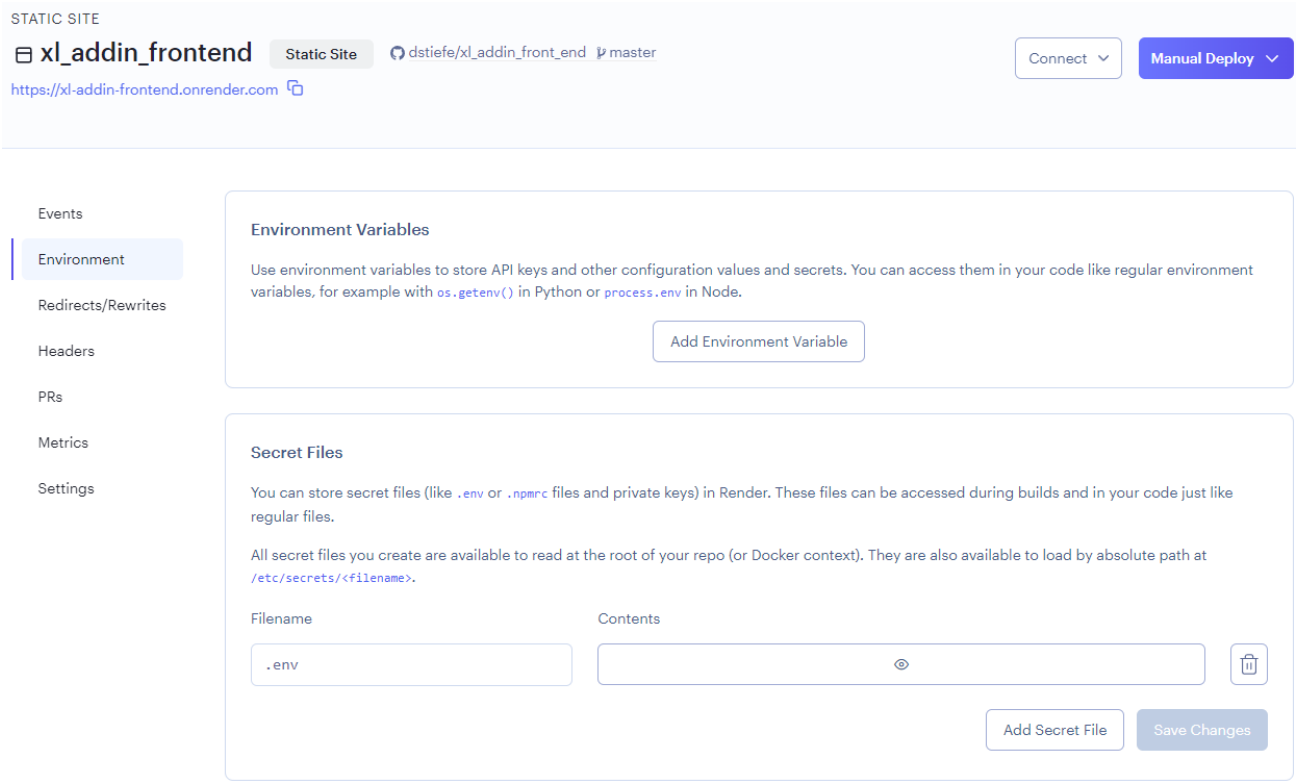
```
    fetch(process.env.QB_CREATE_ACCOUNTS, options)
      .then((response) => response.json())
      .then((response) => console.log(response))
      .catch((err) => console.error(err));
```

Then add the URL to the .env file e.g

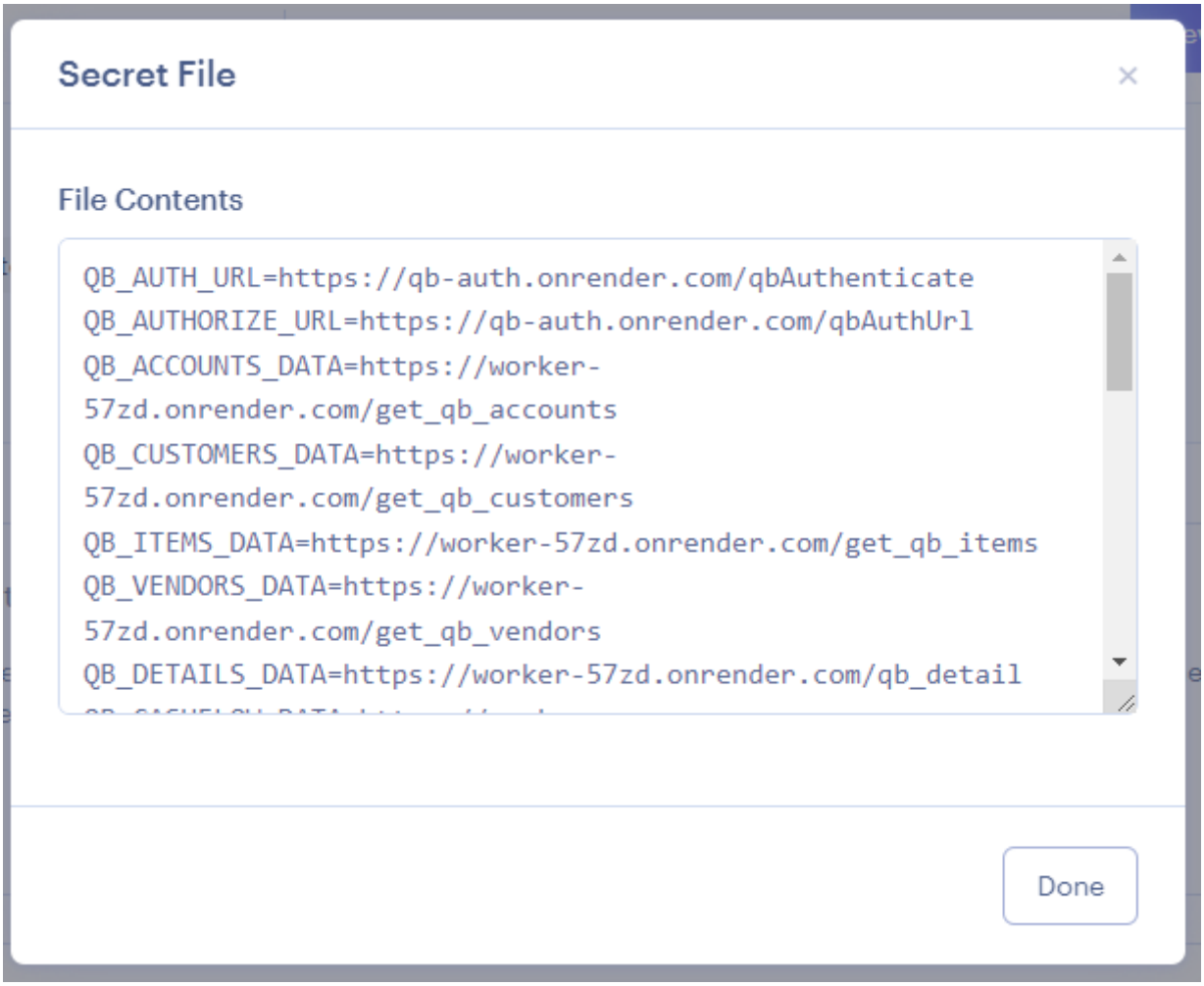



You will need to restart your local server for the env to be seen in the code. Further, you will need to make sure to add the new line to the onRender environment like so:

1. Click into the xl_addin_frontend project
2. Click environments in the sidebar



3. Click the contents button next to the .env name. You should get a pop-uo




4. Add the new line into this window and click done.
5. Click Save Changes