This document is meant to help with the transition of the **Onboarding Checklist**!

The ‘Onboarding Checklist’ project is a web app that allows team leads to create and share checklists to help new employees onboard. The web app allows admin members to create task items that could have subtasks, links, and/or descriptions; and these lists can be shared using each list’s unique 6-digit code.

The application was created using React.js for the frontend and Python Flask for the backend. Here are the commands required to run it locally, I recommend using the VScode terminals:

Terminal 1

Path: OnboardingChecklist/onboarding-checklist

npm start

Terminal 2:

Path: OnboardingChecklist/onboarding-checklist/backend

python -m flask run

**Frequent errors when running the application:**

* npm ERR! Missing script: "start"
  + Means you are not in the correct folder, make sure when you run ‘npm start’ that you are in the ‘onboarding-checklist’ folder and not ‘OnboardingChecklist’
* ‘flask : the term ‘flask is not recognized…”

A screen shot of a computer

Description automatically generated with low confidence

* + Could either mean that you have not installed the regular imports, or there is an issue. Either way, mitigate it using the command “python -m flask run”, which will work and let you know if there are any un-installed imports.
    - If the problem still persists, try using ‘pip install xyz’ to resolve any missing imports or changing which terminal you are using. I’ve found this problem occurs less when I’m using the built in terminal for VScode.

**Frequent errors in browser**

Inspect the browser, and navigate to ‘Console’ to read browser errors. \*This is for the localhost version of the app\*. In order to test the application locally, make sure all fetch functions have the fetch link set to “localhost:5000/api/…”.

A screenshot of a computer

Description automatically generated

* CORS Error / ‘Failed to Fetch’
  + Example:

A screen shot of a computer

Description automatically generated with medium confidence

* + Usually this means that there is a problem with the backend, so first **check the terminal where you are running the flask server.**
  + Check that you’ve installed all of the necessary imports in your terminal, and that your flask server is running.
  + The problem is 99% of the time unrelated to CORS, so check the most recent error in the terminal, it could be an issue with what is being returned with fetch.
* SQLAlchemy Error
  + If you’ve added new aspects/models to models.py, a migration is needed for the database in order for it to work, otherwise you will get an error message like “xyz has no attribute abc”
  + You can migrate the database using the following commands:
    - Python -m flask db stamp head
    - Python -m flask db migrate
    - Python -m flask db upgrade
      * There is a possibility that there will be an error in this step if you are removing foreign keys.
      * Go to migration -> Versions and identify the .py file that has been created for this migration. This can be identified by the ‘Create Date’ field at the top of each file.
      * Comment out any command that has ‘None’ in it. Example:
        + #batch\_op.drop\_constraint(None, type\_=’foreignkey’).
      * Here’s a stack overflow that is similar to the one I used to fix this error: [python - Flask-Migrate creating foreign key with None for no apparent reason - Stack Overflow](https://stackoverflow.com/questions/63732400/flask-migrate-creating-foreign-key-with-none-for-no-apparent-reason)
      * Once you’ve changed this file, DO NOT DO THE FIRST COMMAND AGAIN. Just do migrate -> upgrade.
* Unable to login/No response
  + **Check the console!!!!**
  + If it is a CORS or fetch error, then check your flask terminal. This is where the issue will most likely be + more context of what the error is.
* Additional Debug Tools
  + **Postman** is a great debugging tool to determine whether the backend is running correctly, and can be used while running the localhost backend or to test a deployed backend.
* Additional notes
  + The ‘Onboarding Tasks’ button pulls pre-made onboarding tasks from app.py NOT the list included in todo.js. If you edit the tasks in todo.js, they will not appear when the button is clicked. If you want to edit these premade tasks, edit the list in app.py.
  + Set up of the python backend:
    - App.py
    - Models.py
      * Includes all information/models included in the database. If you want to change the attributes in the database, this is where you must do it. Includes everything that is saved.
      * Users
        + All profile information stored about a user
        + Includes the lists that a user owns
      * TaskList
        + Each individual list, which is attributed to a user.
        + Includes a list of the individual notes attributed to a user.
      * Note
        + Each individual note that is attributed to a Tasklist.
        + Includes a list of the subtasks that are attributed to a note.
      * Subtask
        + Each individual subtask that is attributed to a note.
      * Subtasks > Note > Tasklist > User
    - Config.py
      * Includes settings pertaining to the database and flask-sessions. Very important in deployment and storing cookies.
    - Wsgi.py
      * File that is run for the docker container for deployment.

**Deployment**

* I’ve been working on deploying the project using AWS. It’s a little above my skill set and knowledge, but here is what I’ve learned so far, as well as the tutorial I’ve been consulting:
  + Tutorial: [How to deploy a website on AWS with Docker, Flask, & React from scratch | by Adam Raudonis | Medium](https://adamraudonis.medium.com/how-to-deploy-a-website-on-aws-with-docker-flask-react-from-scratch-d0845ebd9da4)
  + It’s easiest to deploy the backend as a Docker Container. I tried other ways, but deploying as a Docker container allows you to specify the commands needed to run + the needed imports.
    - This is done in the Dockerfile, and since Flask is a development server, I needed to run it using gunicorn.
    - Essential commands:
      * Docker buildx build -t onboarding-backend .
        + -t specifies the tag, which is the name of the container/image (in this case, I’ve named it onboarding-backend)
      * Contents of the Dockerfile:
        + A screenshot of a computer

          Description automatically generated
        + WORKDIR

Specifies the folder with the contents of the container

* + - * + ENV SECRET\_KEY

Environment variable needed for sessions. It’s a fix for an error caused by “No secret key set”, despite it being set in app.py and other files.

* + - * + RUN pip install -r requirements.txt

Instructs the docker container to download all the needed imports. Requirements.txt has a list of the required imports to run the backend.

* + - * + EXPOSE 5000

Exposes the port that the docker container is being run on. If the application changes port, you will need to change it here.

* + - * + CMD […]

The run command. The wsgi.py file is what is run by the docker container (not app.py necessarily).

* + - Troubleshooting:
      * ERROR: failed to solve: failed to read dockerfile: open /var/lib/docker/tmp/buildkit-mount1217864649/Dockerfile: no such file or directory
        + Most likely means you are running the build command in the wrong folder. Make sure you are running the command in the same folder the ‘Dockerfile’ is in.
      * Mixed Content: The page at 'https://d392k517dcrs3l.cloudfront.net/' was loaded over HTTPS, but requested an insecure resource 'http://onboarding-checklist-env.eba-zmqvvmki.us-east-1.elasticbeanstalk.com/api/login'. This request has been blocked; the content must be served over HTTPS.
        + Unfortunately a complex issue I have not been able to solve myself. The frontend is being loaded over HTTPS and the backend (flask webapp) is loaded over HTTP.
        + The only workaround I have been able to do at the moment is run the application on Chrome, and going to my browser settings and set ‘load insecure website data’ as true.
        + Potentially can be fixed with a Load Balancer, but requires a certificate which requires a domain name.
      * No communication between frontend and backend
        + Can happen when the backend is running on a port other than the one the frontend is expecting. Make sure, before deployment, that the container can work with the local version of the frontend (Make sure to change the fetch links to localhost:5000/api/\* first).
    - Current Setup
      * The frontend is an s3 bucket deployed using CloudFront
        + S3 is where you will upload the build file of the react.js frontend.
        + CloudFront is where you will need to create a new Invalidation in order to see the most recent changes. You can do this by simply copying an existing invalidation (which forces CloudFront to refresh the files)
      * The backend is an Elastic Container Registry webapp
        + Click “View Push Commands” to view the commands required to update the backend. Just run those in your terminal in order to push a more recent version of the docker container.
      * There are some settings with Load Balancers, and the entire product I believe is an EC2 server?
      * I followed the above linked tutorial, which should give more insight to the set up.

**Areas for Improvement**

Here are some ideas I have for ways the application could be improved.

* Deletion of subtasks
  + Right now, subtasks cannot be deleted. It would be helpful to add functionality to delete subtasks.
  + How to tackle this:
    - Add a new function in app.py that allows for the deletion of subtasks
    - Create a new function in todo.js that uses ‘fetch’ and sends the desired subtask to be deleted so the python function can identify and delete it from the list
    - Create a ‘delete’ button in todo.js ‘render()’ that calls the fetch function.
* Undo button functionality with database
  + I’m not sure the undo button works correctly.
* Login using Enter key
  + Right now, users must click the login button to login, it would be nice if they could just hit the enter button.
* Unique Copy List Title
  + Right now, users can click ‘Copy List’ if they want to copy the contents of a list. However, this only works the first time, since I have only configured it to create a new list with the same title + copy. So if a user copies the same list twice, it would cause an error of two lists with the same name. It would be nice to keep track of how many lists were copied from the same parent list to avoid this problem (maybe make lists ‘Title + copy + copy number’)
* Edit existing tasks
  + It would be a good feature if users could edit the text or description of a task item. Currently, a user must delete a task if there is a typo or incorrect information.