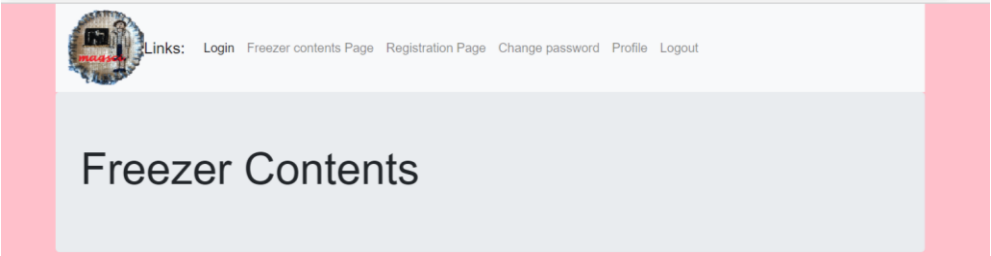


Contents

The packages needed for building a web app

- python with packages installed using pip

```
from flask import Flask, render_template, redirect, url_for, request
from flask_sqlalchemy import SQLAlchemy
from flask_login import LoginManager,UserMixin, login_user,login_required,logout_user,current_user
from datetime import datetime
from sqlalchemy  import and_
from pws import hash_password, verify_password

```

- sqlite database
- flask sqlalchemy with packages installed
- html and css with bootstrap libraries
- jinja template
- github and git bash
- postgres and heroku

Making the section to add and search for items in the freezer

```
class Freezer(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    content = db.Column(db.String(200), nullable=False)
    date_created = db.Column(db.DateTime, default=datetime.utcnow)
    shelf_name = db.Column(db.String(200),nullable=True)
    owner_id = db.Column(db.Integer,db.ForeignKey('user.id'))

```

the app.py

```
@app.route('/freezer_contents',methods=['GET','POST'])
@login_required
def freezer_contents():

    if request.method == 'POST':

        content = request.form['content']
        content = content.lower()
        shelf = request.form['shelf_name']
        shelf = shelf.lower()
        new_record = Freezer(content= content,shelf_name = shelf,owner_id = current_user.id)

        try:
            db.session.add(new_record)
            db.session.commit()

            return redirect('/freezer_contents')

        except:
            return 'There was an issue adding your task'

    elif request.method =='GET':
        s = str(request.args.get('s')).lower()
        c_t = str(request.args.get('c_t')).lower()
        num= len(current_user.freezers)
        num = int(num-1)

        if c_t =="":
            shelfs=[]
            if s=='top':
                i=int(0)
                for each in current_user.freezers:

                    shelf_name = current_user.freezers[i].shelf_name
                    if shelf_name == 'top':
                        shelfs.append(each)
                    i=i+1

            elif s=='middle':
                i=int(0)
                for each in current_user.freezers:

                    shelf_name = current_user.freezers[i].shelf_name
                    if shelf_name == 'middle':
                        shelfs.append(each)
                    i=i+1
            elif s=='bottom':
                i=int(0)
                for each in current_user.freezers:

                    shelf_name = current_user.freezers[i].shelf_name
                    if shelf_name == 'bottom':
                        shelfs.append(each)
                    i=i+1
            else:
                shelfs = current_user.freezers

        else:
            shelfs=[]
            i= int(0)
            for each in current_user.freezers:

                    content = current_user.freezers[i].content
                    content = content.lower()
                    if content == c_t:
                        shelfs.append(each)
                    i=i+1

    return render_template('freezer_contents.html',shelfs = shelfs)
```

the code to add or search is also found in the app.py :

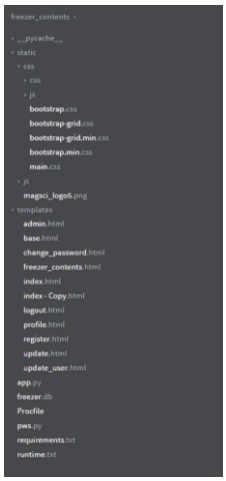
```
@app.route('/delete/<int:id>')
def delete(id):
    task_to_delete =Freezer.query.get_or_404(id)

    try:
        db.session.delete(task_to_delete)
        db.session.commit()
        return redirect('/freezer_contents')
    except:
        return 'There was a problem deleting that task'

@app.route('/update/<int:id>',methods=['GET','POST'])
def update(id):
    shelf = Freezer.query.get_or_404(id)
    if request.method == 'POST':
        shelf.content = request.form['content']

        try:
            db.session.commit()
            return redirect('/freezer_contents')
        except:
            return 'there was an issue updating your shelf'
    else:
        return render_template('update.html', shelf=shelf)

```

```
{% extends 'base.html'%}

{% block head %}
<title>Freezer Contents</title>

{% endblock %}

{% block body %}
<div class = "content">

    <h4 style="text-align: center"> {{current_user.user_name}}'s Freezer</h4>



    {% if shelfs|length == 0 %}
    <p id = "info" style="text-align: center">Choose to Add or Search<p>
    {% else %}

    <table class = "tcenter">
        <tr>
            <th>Item</th>
            <th>Shelf</th>
            <th>Added</th>
            <th>Actions</th>
        </tr>
        {% for shelf in shelfs %}

            <tr>
                <td>{{ shelf.content }}</td>
                <td>{{ shelf.shelf_name }}</td>
                <td>{{ shelf.date_created.date() }}</td>
                <td>
                    <a href="/delete/{{shelf.id}}">Delete</a>
                    <br>
                    <a href="/update/{{shelf.id}}">Update</a>
                </td>
            </tr>
        {% endfor %}

    </table>
    {% endif %}
    <p id = "info" style="text-align: center">For all items select search below!<p>

    <p id = "info" style="text-align: center">To see what is in a shelf, enter top, middle or bottom!<p>

```

```
    </table>
    {% endif %}
    <p id = "info" style="text-align: center">For all items select search below!<p>

    <p id = "info" style="text-align: center">To see what is in a shelf, enter top, mid


    <form action="/freezer_contents" method='get' align = 'center'>
        <label>To view single shelf enter the name here:</label>
        <input type="text" name="s" type = "text">
        <label>To search for an item enter the item name here:</label>
        <input type="text" name="c_t" type = "text">
        <!--<label>Shelf:</label>
        <input type="text" name="shelf_name" type = text>-->
        <button type="submit" class="btn btn-success">Search</button>


    </form>
    <br>
    <p id = "info"  style="text-align: center">Add some items here!<p><br>
    <form action="/freezer_contents" method='post' align = 'center'>

        <label>New item:</label>
        <input type="text" name="content" type = "text">
        <label>Shelf:</label>
        <input type="text" name="shelf_name" type = text>
        <button type="submit" class="btn btn-success">Add</button>


    </form>



</div>

{% endblock %}
```

html and css with jinja(?)

the app has a directory structure:



base.html has the code that is common to all pages, e.g. navbar:

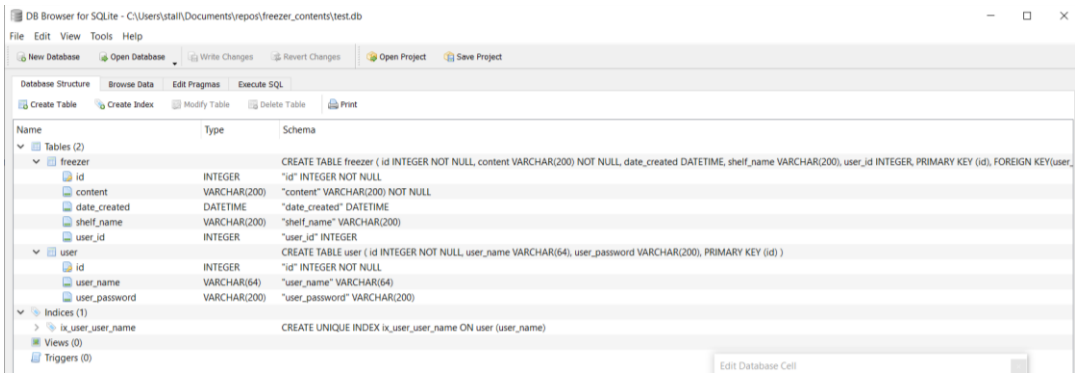
```
<html lang="en">

<head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">


    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css">
    <link rel="stylesheet" href="{{ url_for('static',filename ='css/main.css')}}">
    <script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
    <style>
    .bs-example{
        margin: 20px;
    }
</style>

    {% block head %}
    {% endblock %}
</head>

<body>

    <div class="container">
        <nav class="navbar navbar-expand-lg navbar-light bg-light">
            <a href="#" class="navbar-left"><img src="static/magsci_logo5.png"></a>
            <a class="navbar-brand" href="#">Links:</a>
            <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-
            expanded="false" aria-label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>

            <div class="collapse navbar-collapse" id="navbarSupportedContent">
                <ul class="navbar-nav mr-auto">
                    <li class="nav-item active">
                        <a class="nav-link" href="{{ url_for('index') }}">Home <span class="sr-only">(current)</span></a>
                    </li>

                    <li class="nav-item">
                        <a class="nav-link" href="{{ url_for('freezer_contents') }}">Freezer contents Page</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="{{ url_for('register') }}">Registration Page</a>
```

Setting up the configuration etc

This is done in python in app.py

```
app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///freezer.db'
app.config['SECRET_KEY']='thisisasecret'

db = SQLAlchemy(app)
login_manager = LoginManager(app)
login_manager.init_app(app)
```

this works fine for sqlite in development mode



pip3 install virtualenv

virtualenv env

\env\Scripts\activate.bat

this created virtual environment for app

python app.py

open browser 127.0.0.1:5000 as specified in cmd

open python shell: from app import db

db.create_all()

exit()

This makes tables according to schema in app.py

Making the user management section

the flask python code:

```python
class User(UserMixin,db.Model):
    id = db.Column(db.Integer, primary_key=True)
    user_name = db.Column(db.String(64), unique=True)
    user_password = db.Column(db.String(200),unique=True)
    freezers = db.relationship('Freezer', backref='owner')
```

```python
@login_manager.user_loader
def load_user(user_id):
    return User.query.get(int(user_id))


@app.route("/", methods=['GET','POST'])
def index():
    if request.method == 'POST':

        name = request.form['nam']
        password = request.form['pw']
        if name =='administrator1':
            user = User.query.filter_by(user_name=name).first()
            s_pw = user.user_password
            if s_pw ==password:
                login_user(user)
        else:
            user =User.query.filter_by(user_name=name).first()
            s_pw = user.user_password
            if verify_password(s_pw,password):

                login_user(user)


    return render_template(('index.html'))
```

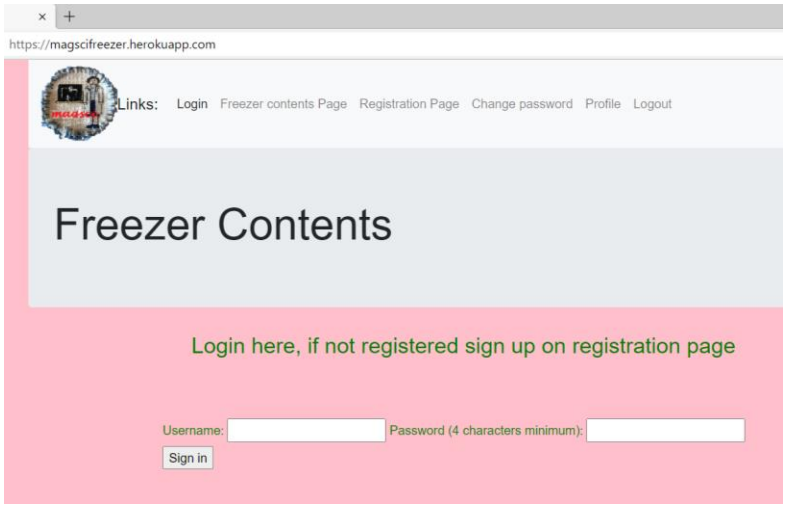
the Home pagen(index.html) has the login form:

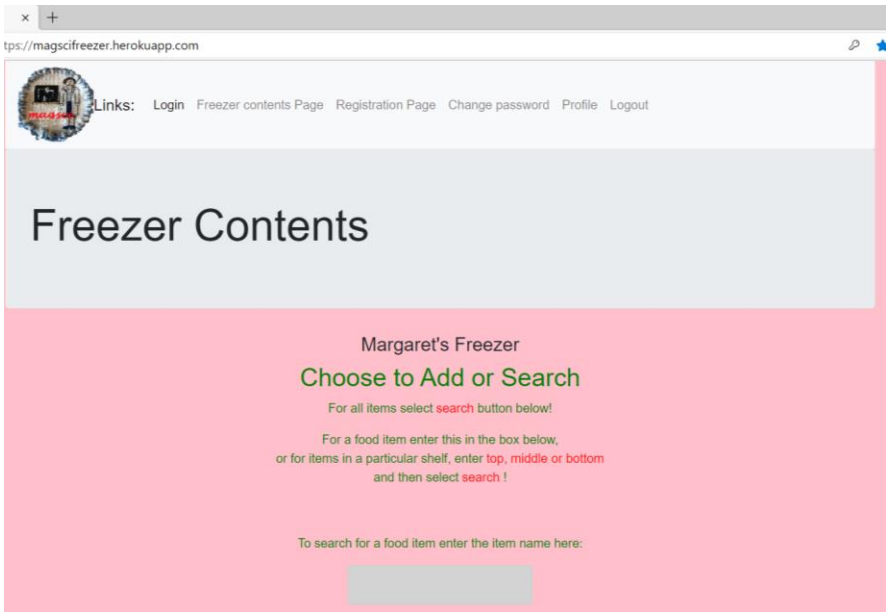
```
{% extends 'base.html'%}

{% block head %}
<title>Index page</title>
{% endblock %}
{% block body %}

<div class = "content">
    <p id = "info" style="text-align: center">login here, if not registered sign up on registration page<p><br><br>
     <form action="/"  method = "post">
        <label for="username">Username:</label>
        <input type="username" id="username" name="nam">

        <label for="pass">Password (4 characters minimum):</label>
        <input type="password" id="pass" name="pw"
               minlength="4" required>
        <input type="submit" value="Sign in">
     </form>
    <br>
    <p id = "info" style="text-align: center">Now select freezer_contents page from the links above<p>
{% endblock %}
```

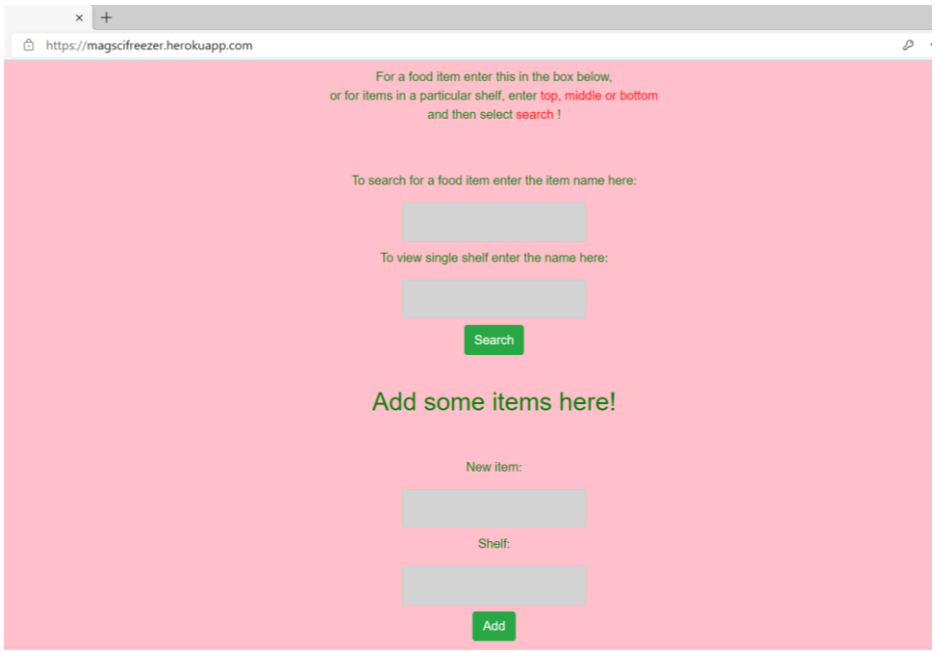
The registration page has similar code

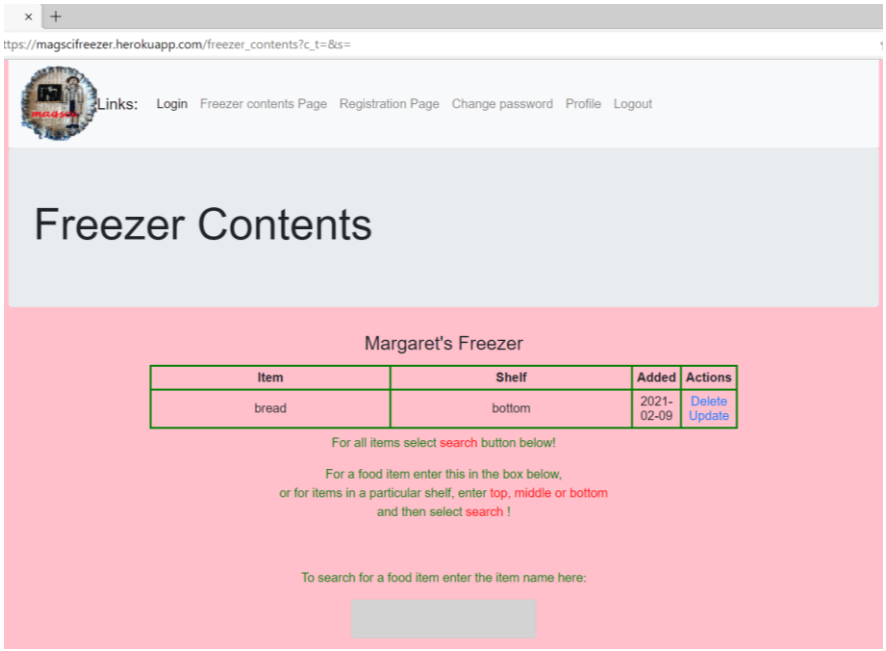
What the pages look like

index or login page:









Deployment

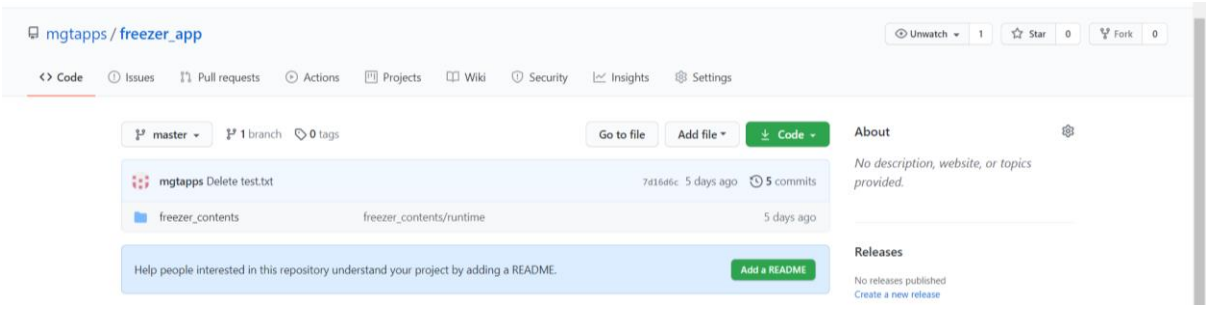
Github

in github I created a new repository :

git remote add origin https://github.com/remote_link (copied and psted from github)

then

git push origin master



I copied the link of this new remote repository

This should have initialised c/app_folder as master

git config –global user.username mgtapps

git remote add origin https://github.com

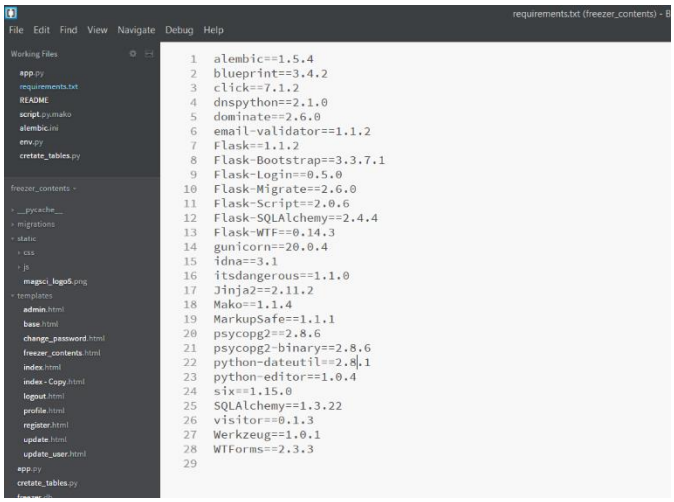
git push origin master

then configure email in a similar manner

The app folder needed Procfile with no file extension. This is achieved by saving in notepad but putting the name in the save as box in parentheses

the requirements.txt file needed to be generated:

pip freeze > requirements.txt



Heroku

I set up a free account in Heroku and used the steps in the video:

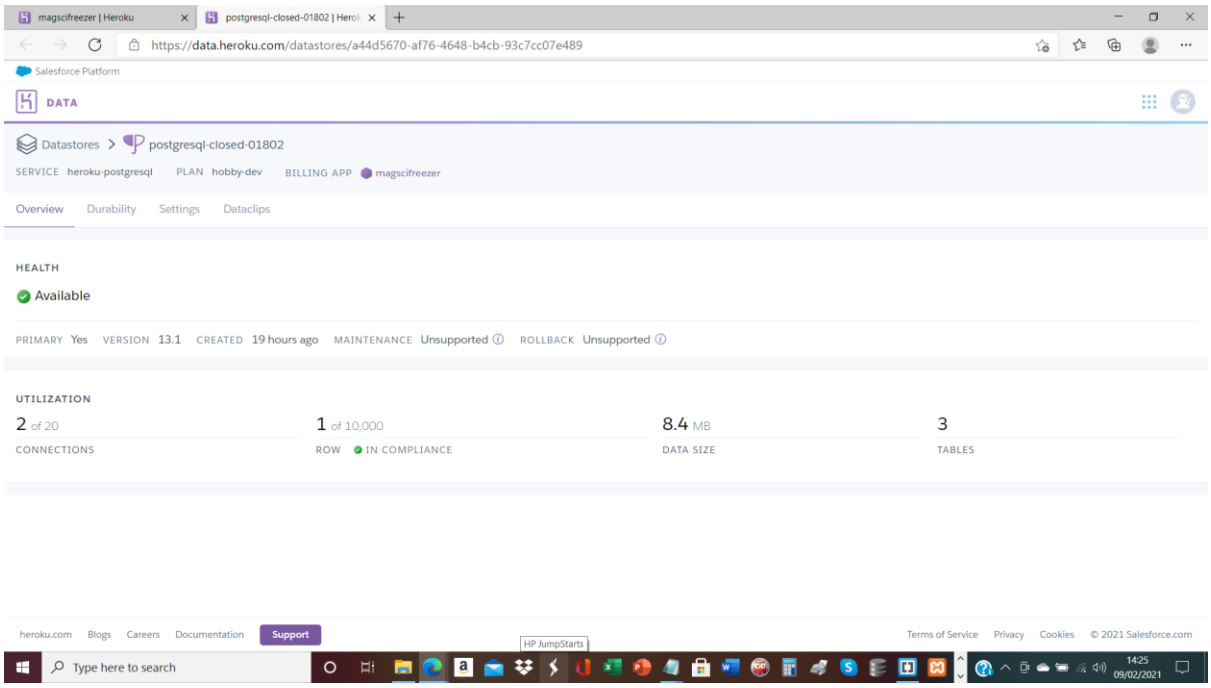
Deploy a Flask App to Heroku With a Postgres Database [2019] - YouTube

in local repository and then in Heroku:

flask d binit

flask db migrate

flsk db upgrade



Maintenance

I will need to check that the app stays available.

the add may need to be updated based on feedback:

updating is achieved:

using git bask in local repository:

git add filename

git commit -am filename

git push heroku master

Next steps

I may need to add a contact/help link to the app e.g. email

I may need to change the how the configuration setting have been added. I may experiment with a simpler system

may need to take out db.init_app(app) form app.py