

Contents

The packages needed for building a web app	2
Making the section to add and search for items in the freezer.....	2
Setting up the configuration etc	4
Making the user management section	5
What the pages look like.....	7
Deployment	8
Maintenance	10
Next steps	10

The packages needed for building a web app

- python with packages installed using pip

```
1 from flask import Flask, render_template, redirect, url_for, request
2 from flask_sqlalchemy import SQLAlchemy
3 from flask_login import LoginManager, UserMixin, login_user, login_required, logout_user, current_user
4 from datetime import datetime
5 from sqlalchemy import and_
6 from pws import hash_password, verify_password
7
```

- 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

```
26
27 class Freezer(db.Model):
28     id = db.Column(db.Integer, primary_key=True)
29     content = db.Column(db.String(200), nullable=False)
30     date_created = db.Column(db.DateTime, default=datetime.utcnow)
31     shelf_name = db.Column(db.String(200), nullable=True)
32     owner_id = db.Column(db.Integer, db.ForeignKey('user.id'))
33
34
```

the app.py

```
65
66 @app.route('/freezer_contents', methods=['GET', 'POST'])
67 @login_required
68 def freezer_contents():
69
70     if request.method == 'POST':
71
72         content = request.form['content']
73         content = content.lower()
74         shelf = request.form['shelf_name']
75         shelf = shelf.lower()
76         new_record = Freezer(content=content, shelf_name=shelf, owner_id=current_user.id)
77
78         try:
79             db.session.add(new_record)
80             db.session.commit()
81             return redirect('/freezer_contents')
82
83         except:
84             return 'There was an issue adding your task'
85
86     elif request.method == 'GET':
87         s = str(request.args.get('s')).lower()
88         c_t = str(request.args.get('c_t')).lower()
89         num = len(current_user.freezers)
90         num = int(num-1)
91
92         if c_t == '':
93             shelves = []
94             if s == 'top':
95                 i = int(0)
96                 for each in current_user.freezers:
97
98                     shelf_name = current_user.freezers[i].shelf_name
99                     if shelf_name == 'top':
100                         shelves.append(each)
101                     i=i+1
102
103             elif s == 'middle':
104                 i = int(0)
105                 for each in current_user.freezers:
106
107                     shelf_name = current_user.freezers[i].shelf_name
108                     if shelf_name == 'middle':
109
```

```
109         if shelf_name == 'middle':
110             shelves.append(each)
111             i=i+1
112         elif s == 'bottom':
113             i = int(0)
114             for each in current_user.freezers:
115
116                 shelf_name = current_user.freezers[i].shelf_name
117                 if shelf_name == 'bottom':
118                     shelves.append(each)
119                 i=i+1
120             else:
121                 shelves = current_user.freezers
122
123         else:
124             shelves = []
125             i = int(0)
126             for each in current_user.freezers:
127
128                 content = current_user.freezers[i].content
129                 content = content.lower()
130                 if content == c_t:
131                     shelves.append(each)
132                 i=i+1
133
134         return render_template('freezer_contents.html', shelves = shelves)
```

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

```
136
137 @app.route('/delete/<int:id>')
138 def delete(id):
139     task_to_delete = Freezer.query.get_or_404(id)
140
141     try:
142         db.session.delete(task_to_delete)
143         db.session.commit()
144         return redirect('/freezer_contents')
145     except:
146         return 'There was a problem deleting that task'
147
148 @app.route('/update/<int:id>', methods=['GET', 'POST'])
149 def update(id):
150     shelf = Freezer.query.get_or_404(id)
151     if request.method == 'POST':
152         shelf.content = request.form['content']
153
154         try:
155             db.session.commit()
156             return redirect('/freezer_contents')
157         except:
158             return 'there was an issue updating your shelf'
159     else:
160         return render_template('update.html', shelf=shelf)
161
```

```

1 {% extends 'base.html'%}
2
3 {% block head %}
4 <title>Freezer Contents</title>
5
6 {% endblock %}
7
8 {% block body %}
9 <div class = "content">
10
11     <h4 style="text-align: center"> {{current_user.user_name}}'s Freezer</h4>
12
13
14
15     {% if shelves|length == 0 %}
16     <p id = "info" style="text-align: center">Choose to Add or Search</p>
17     {% else %}
18
19 <table class = "tcenter">
20 <tr>
21     <th>Item</th>
22     <th>Shelf</th>
23     <th>Added</th>
24     <th>Actions</th>
25 </tr>
26 {% for shelf in shelves %}
27
28 <tr>
29     <td>{{ shelf.content }}</td>
30     <td>{{ shelf.shelf_name }}</td>
31     <td>{{ shelf.date_created.date() }}</td>
32 <td>
33         <a href="/delete/{{shelf.id}}">Delete</a>
34         <br>
35         <a href="/update/{{shelf.id}}">Update</a>
36     </td>
37 </tr>
38 {% endfor %}
39
40 </table>
41 {% endif %}
42 <p id = "info" style="text-align: center">For all items select search below</p>
43
44 <p id = "info" style="text-align: center">To see what is in a shelf, enter top, middle or bottom</p>
45
46

```

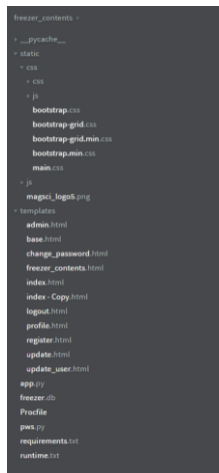
```

47 </table>
48
49 {% endif %}
50 <p id = "info" style="text-align: center">For all items select search below</p>
51
52 <p id = "info" style="text-align: center">To see what is in a shelf, enter top, mid
53
54
55 <form action="/freezer_contents" method='get' align = 'center'>
56 <label>To view single shelf enter the name here:</label>
57 <input type="text" name="s" type = "text">
58 <label>To search for an item enter the item name here:</label>
59 <input type="text" name="c_t" type = "text">
60 <!--<label>Shelf:</label>
61 <input type="text" name="shelf_name" type = text-->
62 <button type="submit" class="btn btn-success">Search</button>
63
64 </form>
65 <br>
66 <p id = "info" style="text-align: center">Add some items here:</p><br>
67 <form action="/freezer_contents" method='post' align = 'center'>
68
69 <label>New item:</label>
70 <input type="text" name="content" type = "text">
71 <label>Shelf:</label>
72 <input type="text" name="shelf_name" type = text>
73 <button type="submit" class="btn btn-success">Add</button>
74
75 </form>
76
77 </div>
78
79 {% 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:

```

1 <html lang="en">
2
3 <head>
4   <!-- Required meta tags -->
5   <meta charset="utf-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
7
8
9   <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
10  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css">
11  <link rel="stylesheet" href="{ { url_for('static', filename='css/main.css') }}">
12  <script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
13  <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"></script>
14  <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
15 </head>
16 <body>
17   <div class="bs-example" style="margin: 20px;">
18   </div>
19 </body>
20
21 <!-- Block head % -->
22 <!-- Endblock % -->
23 </head>
24
25 <body>
26
27 <div class="container">
28   <div class="navbar navbar-expand-lg navbar-light bg-light">
29     <a href="#" class="navbar-left"></a>
30     <a class="navbar-brand" href="#">Links: /a>
31     <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
32       <span class="navbar-toggler-icon"></span>
33     </button>
34
35     <div class="collapse navbar-collapse" id="navbarSupportedContent">
36       <ul class="navbar-nav mr-auto">
37         <li class="nav-item active">
38           <a class="nav-link" href="{ { url_for('index') }}">Home <span class="sr-only">(current)</span></a>
39         </li>
40
41         <li class="nav-item">
42           <a class="nav-link" href="{ { url_for('freezer_contents') }}">Freezer contents Page</a>
43         </li>
44
45         <li class="nav-item">
46           <a class="nav-link" href="{ { url_for('register') }}">Registration Page</a>
47         </li>
48
49       </ul>
50     </div>
51   </div>
52 </div>
53
54 </body>
55 </html>

```

Setting up the configuration etc

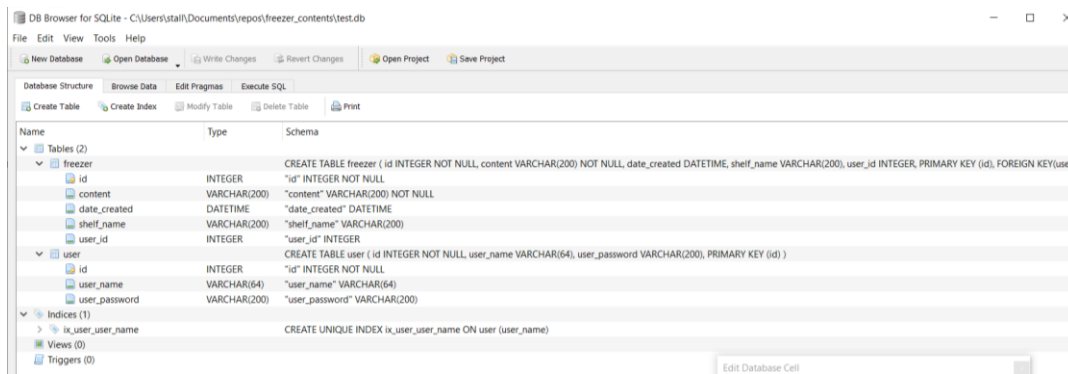
This is done in python in app.py

```

7
8
9 app = Flask(__name__)
10 app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///freezer.db'
11 app.config['SECRET_KEY'] = 'thisisasecret'
12
13 db = SQLAlchemy(app)
14 login_manager = LoginManager(app)
15 login_manager.init_app(app)
16
17

```

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:

```
18
19
20
21
22
23
24
25
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')
```

```
37 @login_manager.user_loader
38 def load_user(user_id):
39     return User.query.get(int(user_id))
40
41
42 @app.route('/', methods=['GET', 'POST'])
43 def index():
44     if request.method == 'POST':
45
46         name = request.form['name']
47         password = request.form['pw']
48         if name == 'administrator!':
49             user = User.query.filter_by(user_name=name).first()
50             s_pw = user.user_password
51             if s_pw == password:
52                 login_user(user)
53         else:
54             user = User.query.filter_by(user_name=name).first()
55             s_pw = user.user_password
56             if verify_password(s_pw, password):
57
58                 login_user(user)
59
60
61
62     return render_template('index.html')
```

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

```
1 {% extends 'base.html'%}
2
3 {% block head %}
4 <title>Index page</title>
5 {% endblock %}
6 {% block body %}
7
8 <div class = "content">
9     <p id = "info" style="text-align: center">login here, if not registered sign up on registration page<br><br>
10
11     <form actions="/" method = "post">
12         <label for="username">Username:</label>
13         <input type="text" id="username" name="name">
14         <label for="pass">Password (4 characters minimum):</label>
15         <input type="password" id="pass" name="pw">
16         minlength="4" required
17         <input type="submit" value="Sign in">
18     </form>
19     <br>
20     <p id = "info" style="text-align: center">Now select freezer_contents page from the links above<br>
21 {% endblock %}
22
```

The registration page has similar code

What the pages look like

index or login page:

The screenshot shows a web browser window with the URL `https://magscifreezer.herokuapp.com`. The page has a light blue header with a circular logo on the left and a navigation menu with links: [Links:](#), [Login](#), [Freezer contents Page](#), [Registration Page](#), [Change password](#), [Profile](#), and [Logout](#). Below the header, the main heading "Freezer Contents" is displayed in a large, bold, black font. Underneath this heading, a green text prompt reads: "Login here, if not registered sign up on registration page". At the bottom of the page, there is a login form with two input fields: "Username:" and "Password (4 characters minimum):". A "Sign in" button is positioned below the "Username:" field.

This screenshot shows the same web browser window, but the page content has changed. The header and navigation menu remain the same. Below the "Freezer Contents" heading, the text "Margaret's Freezer" is displayed in a bold, black font. Underneath, the heading "Choose to Add or Search" is shown in a bold, green font. A green text prompt reads: "For all items select **search** button below!". Below this, another green text prompt reads: "For a food item enter this in the box below, or for items in a particular shelf, enter **top**, **middle** or **bottom** and then select **search** !". At the bottom of the page, there is a text prompt: "To search for a food item enter the item name here:", followed by a large, empty, light gray rectangular input box.

For a food item enter this in the box below,
or for items in a particular shelf, enter **top**, **middle** or **bottom**
and then select **search** !

To search for a food item enter the item name here:

To view single shelf enter the name here:

Search

Add some items here!

New item:

Shelf:

Add

Links: [Login](#) [Freezer contents Page](#) [Registration Page](#) [Change password](#) [Profile](#) [Logout](#)

Freezer Contents

Margaret's Freezer

Item	Shelf	Added	Actions
bread	bottom	2021-02-09	Delete Update

For all items select **search** button below!

For a food item enter this in the box below,
or for items in a particular shelf, enter **top**, **middle** or **bottom**
and then select **search** !

To search for a food item enter the item name here:

Deployment

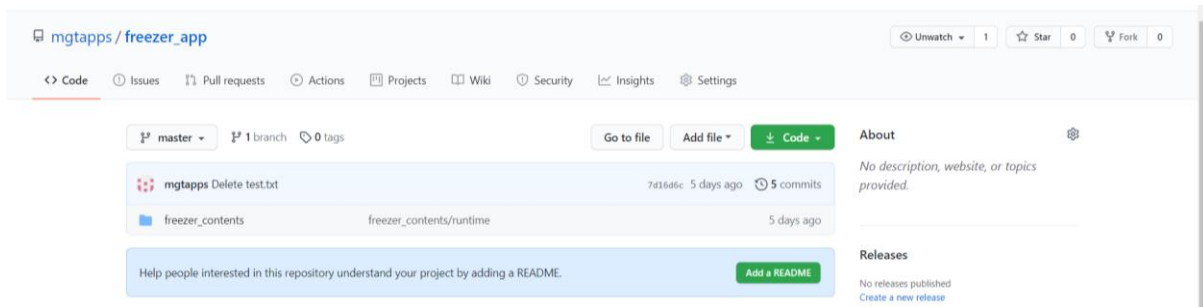
Github

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

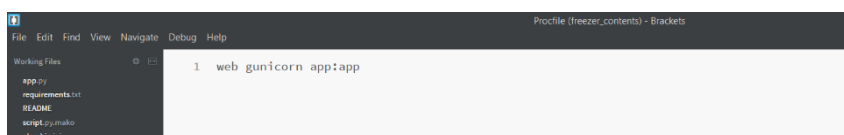
git config --global user.username mgtagps

git remote add origin <https://github.com>

git push origin master

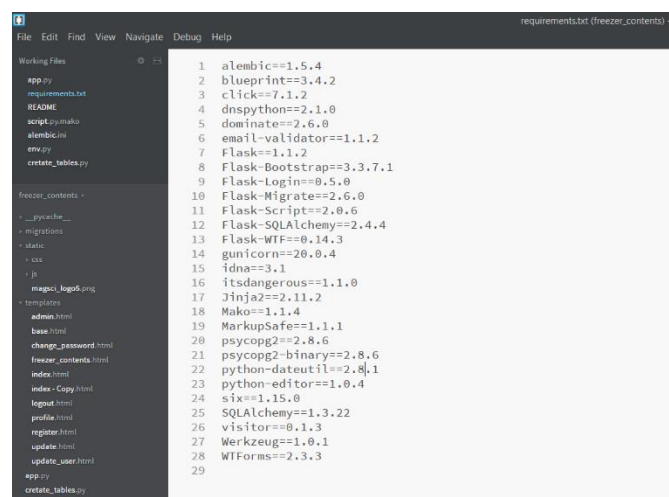
then configure email in a similar manner

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:

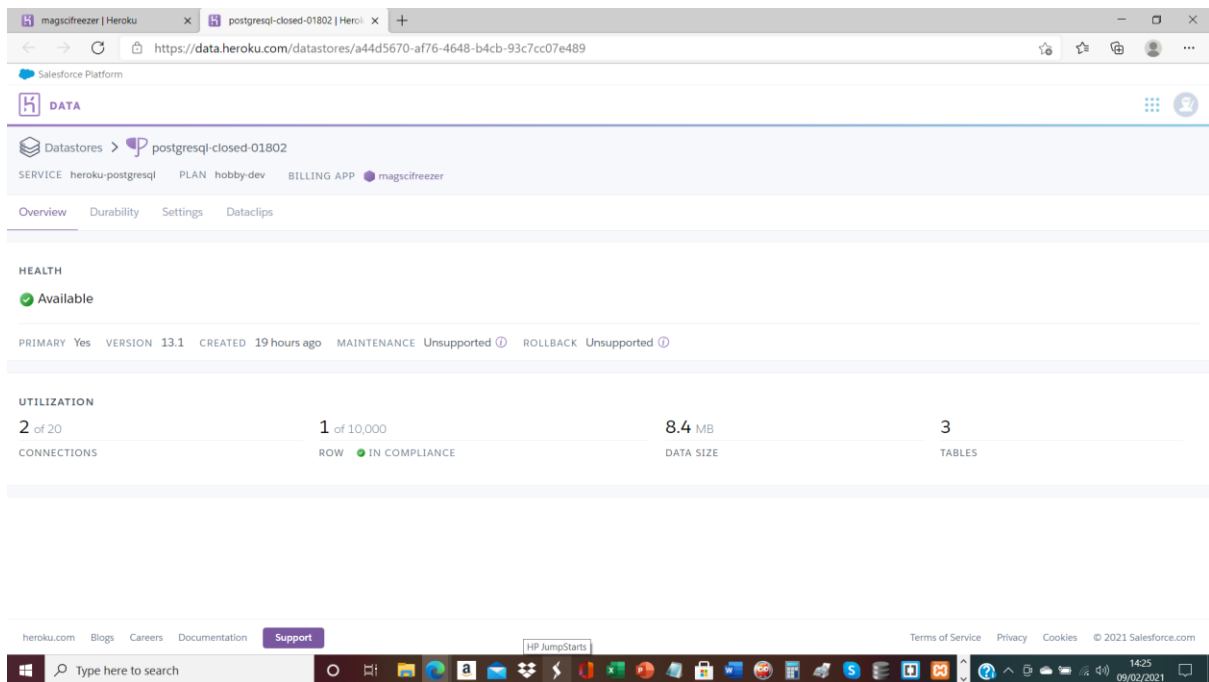
[Deploy a Flask App to Heroku With a Postgres Database \[2019\] - YouTube](#)

in local repository and then in Heroku:

flask db init

flask db migrate

flask 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