Preplanning

Purpose

Create a more fun and initiative way to convert nonstandard measurement e.g. Cups, Teaspoons, or Pounds. Into more standard measurement like grams or mills.

Intended audience

Chefs or people interested in cooking who want a fun way to convert measurements. Or people who don't know the conversions between different measurements.

Relevant Implications

Accessibility

Explain: How easily people are able to use the program. Must be able to be easily used by anyone who may want to convert measurements.
Address:

- Choose a readable font so that anyone is able to read the website and see what it says.
- Display data in a format that makes it easy to see what, when, and how much has been converted
- Make sure all the part of the website and explained properly so the user can know how
- it work as well as how they can use it.
- Not have colors that can't be seen by everyone. E.g. color blind people who may not be able to see red or blue.

Usability

Explain: How users can interact with the program. Make sure that the program is able to be easily used by anyone who may want to use it.
Address:

- Make sure that the program is simple and isn't too complex
- Make sure that the user isn't able to create error or make situations that are impossible E.g. have a negative amount of measurements.
- Configure options to fit in with what the user wants to convert.

Functionality

Explain: The program should function as it is intended to. It needs to work as an efficient way to convert measurements for cooking.
Address:

- Make sure there is a list of units that is accessible to the user and is in the program.
- Create a dictionary containing the conversions between measurement and gram or milliliters.
- Make program convert user inputted measurement into grams or milliliters.
- Needs to correctly interact with database querying data and inserting it.

Ethical

Explain: Anything that is collected by the website must be done with the users permission and shouldn't be used for immoral or bad thing purposes.
Address:

- Only collect data which the user has agreed to.
- Only use data for displaying measurement conversions and for no other purpose except for that.
- Don't use the data collected for any immoral purposes i.e. using it to advertising products to.

Privacy

Explain: Data which is collected should be stored properly so that user's information is kept private and hidden from intruders.
Address:

- Keep data in SQLite3 database in tables.
- Make sure that database isn't directly accessible on the website.

Aesthetics

Explain: Needs to be visual appealing and or understand so user is able to understand what the program does and how it does it.
Address:

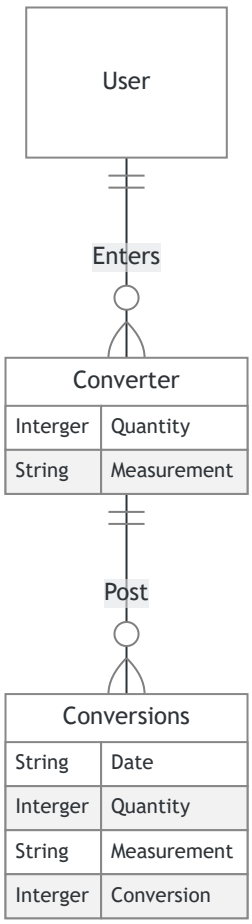
- Display data from conversions in a table in an order way so the user can easily understand the final conversion as well as see how much and what they converted at a certain date.
- Use formatting to change text in order to be more readable
- Make hover over text orange to make sure that the user can see what they are doing on the page.

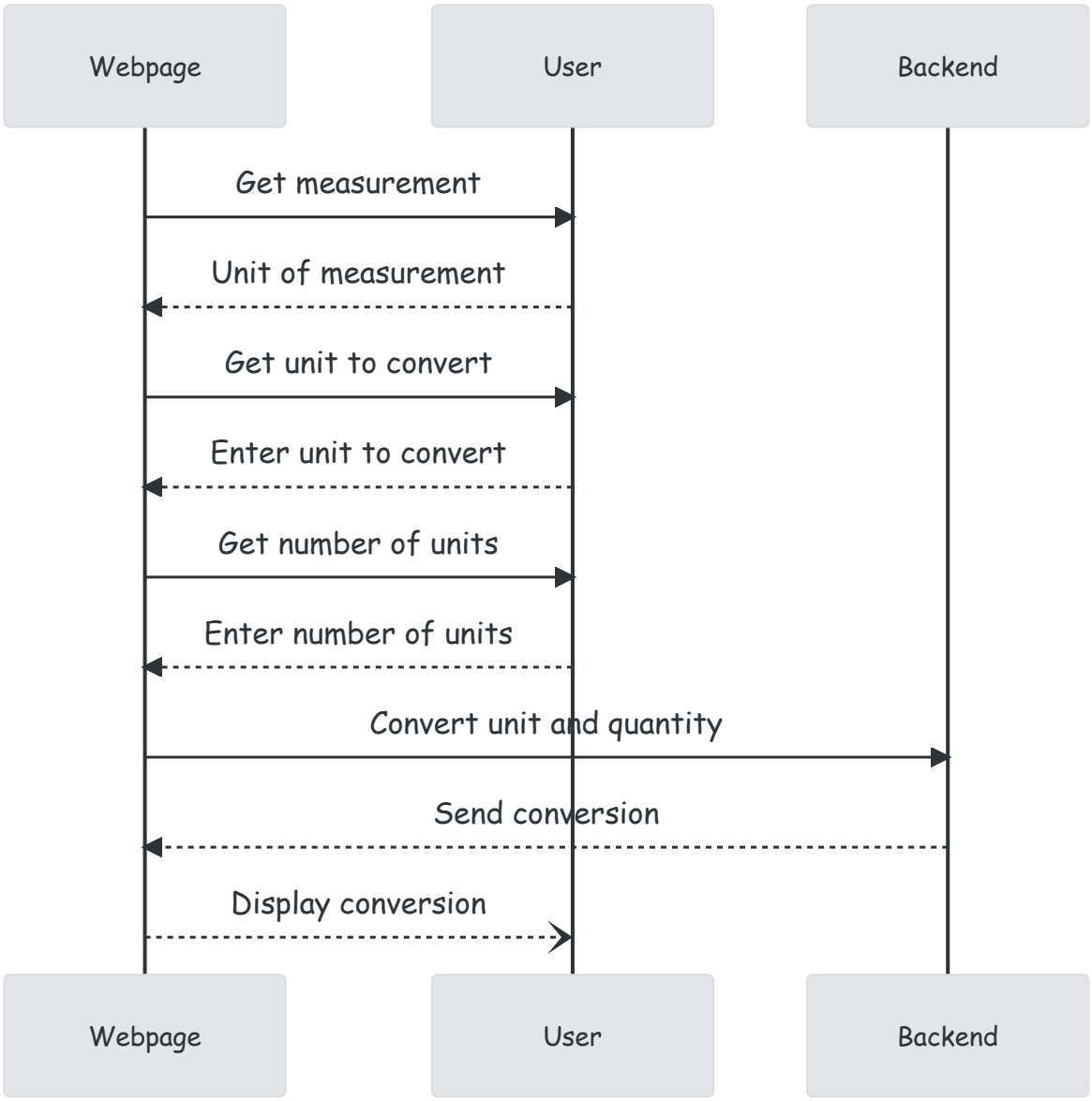
End-user considerations

Explain: Who the users are and what should be thought about in order to make a program which will be appropriate for them.
Address:

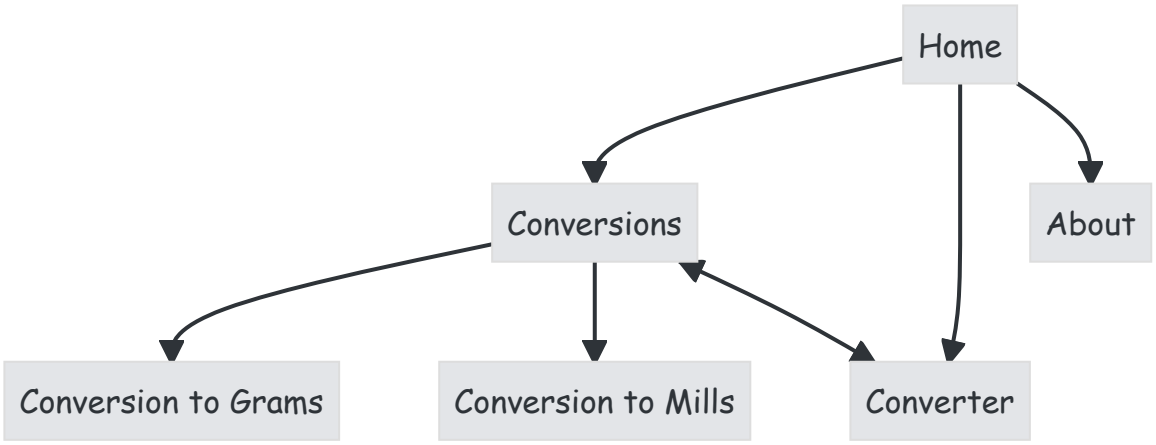
- Explain all features of website. As people who are interested in cooking are unlikely to be very technically savvy or advanced. The program needs to be written in a relatively simple and fun way so that the user can fully understand the purpose of the program.
- Fast and serve it's purpose well as a measurement convert
- Provide a more fun way to convert measurements with more personality and flare. Doing this with drawings and custom cursor to look more visually appealing.

Diagrams


User
Enters
Converter
Interger
Quantity
String
Measurement
Post
Conversions
String
Date
Interger
Quantity
String
Measurement
Interger
Conversion


Webpage
User
Backend
Get measurement
Unit of measurement
Get unit to convert
Enter unit to convert
Get number of units
Enter number of units
Convert unit and quantity
Send conversion
Display conversion
Webpage
User
Backend

Site map



Home page is what the user starts on and directs to other pages. User can then go to converter where once they fill out the form within the page will be redirected to conversions which will display recent conversions that user has entered. Conversion page can then redirect to converter if user wishes to convert any more measurement or direct to either conversion to grams or mills pages which have a table showing data entered into the database.

I managed project using GitHub

Design

Measureverto Color pallete
I decided to only use Comic sans across my website.
As well I decided that my primary colors would be F5F5F5and CDD0D0 , which would make up 60% of my website according to design principles. 000000 would make up 30% of my website. While only 10% of my website would be in FF4400 and FF7F50.
I did this to improve readability the the primary colors would be used for background and other elements as well as highlighting text. Black would be used only for text and other elements. While Coral would be used for accents across the website to add more color.
I decided to go for a simplistic color palette as I knew I wanted an image which would have more colors in it to be the main background of my website.

Sprints

Sprint 1

Outline

- Create routes for all pages
- Create database and make sure it is functional.
- Create form that user is able to interact with.
- Design a basic layout with header and footer.
- Change font style and add different background color.
- Insert test data from list into database.
- Insert test data into the database. .
- Set up Tables loop through data from database.

Process

Initial Setup of Web app

Reason for design: I wanted to create test code to make sure that the backend was routing correctly as well as displaying the right message when I debugged the program.

Code for route

```
from flask import Flask, render_template
app = Flask(__name__)

@app.route('/')
def test():
        return render_template('test.html')

app.run(debug=True)
```

Code for template

```
<h1>Hello world</h1>
```

Templates

Home

Reason for design:I Wanted to create a basic home page which would be display when the website was first accesses and display the correct message from the HTML template.

Code for route

```
@app.route('/')
def home():
        return render_template('home.html')
```

Code for template

```
<p> Welcome to the home page</p>
```

About

Reason for design: I wanted to test the About/help page to make sure that the page was routed correctly via entering "/help" into the URL bar of the browser. Also that the template was also rendered correctly displaying "This is the about page".
Code for route

```
@app.route('/help')
def about():
        return render_template('about.html')
```

Code for template

```
<p> This is the about page. </p>
```

Converter

Reason for design: I wanted to test the converter page to make sure that the page was routed correctly via entering "/convert" into the URL bar of the browser. Also that the template was also rendered correctly displaying "This is the converter page".
Code for route

```
@app.route("/convert")
def converter():
        return render_template('converter.html')
```

Code for template

```
<p>This is the converter page<p>
```

Conversions

Reason for design: I wanted to test the conversions page to make sure that the page was routed correctly via entering "/" into the URL bar of the browser. Also that the template was also rendered correctly displaying "This is the conversion page".
Code for route

```
@app.route('/conversion')
def conversion():
    return ('conversions.html')
```

Code for template

```
<p> This is the conversion page</p>
```

Conversion to Mills

Reason for design: I wanted to test the conversion to mills page to make sure that the page was routed correctly via entering "/" into the URL bar of the browser. Also that the template was also rendered correctly displaying "This is the conversion to mills page".
Code for route

```
@app.route('/conversionmills')
def conversionmills():
    return render_template('conversionmills.html')
```

Code for template

```
<p> This is the conversion to mills page <p>
```

Conversion to Grams

Reason for design: I wanted to test the conversion to grams page to make sure that the page was routed correctly via entering "/" into the URL bar of the browser. Also that the template was also rendered correctly displaying "This is the Conversion to Grams page".
Code for route

```
@app.route('/conversiongrams')
def conversiongrams():
        return render_template('conversiongrams.html')
```

Code for template

```
<p> This is the conversion to grams page.<p>
```

Database setup

Reason for design:I wanted to make sure that SQLite worked correctly and that the database would be created as well as the table also created. I made a table containing two columns but I would only use it for testing purposes to make sure it was created correctly.

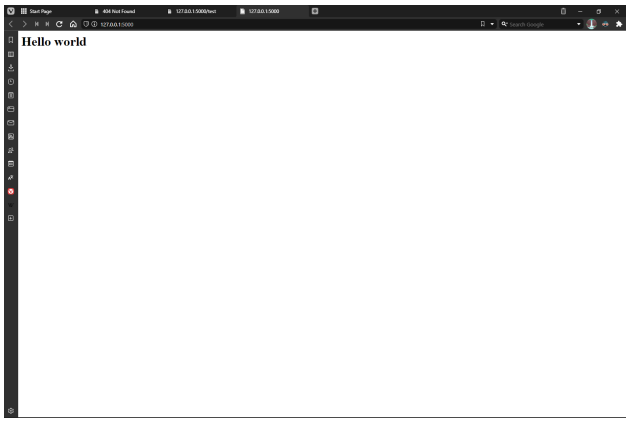
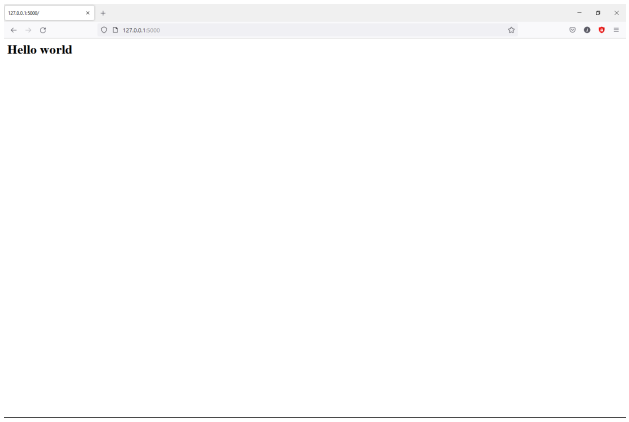
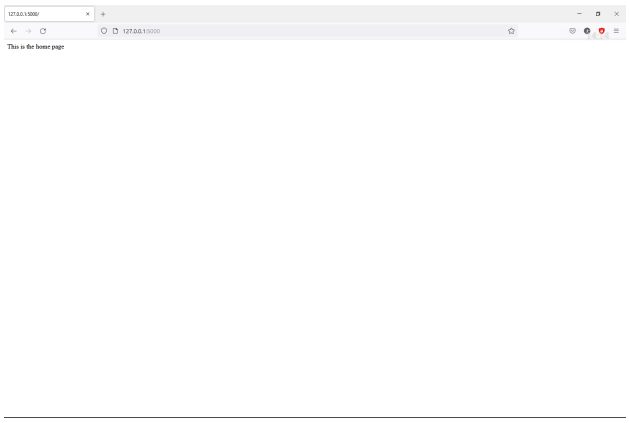
```
import sqlite3
connect = sqlite3.connect('Conversions')
c=connect.cursor()
try:
        c.execute("""CREATE TABLE millilitres(meausrment TEXT, quantity INTERGER
        , Conversion INTERGER)""")
        print('Table millilitres created')
except:
        pass

try:
        c.execute("""CREATE TABLE grams(meausrment TEXT, quantity INTERGER,
         Conversion INTERGER)""")
         print('Table grams created')
except:
        pass
connect.commit()
connect.close()
print('Database connected succesfully')
connect.commit()
connect.close()
```

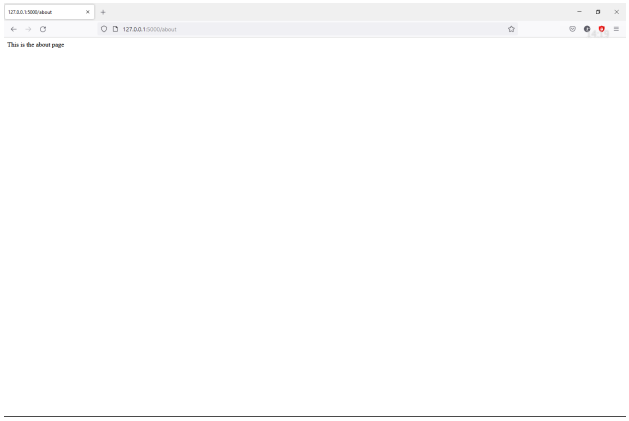
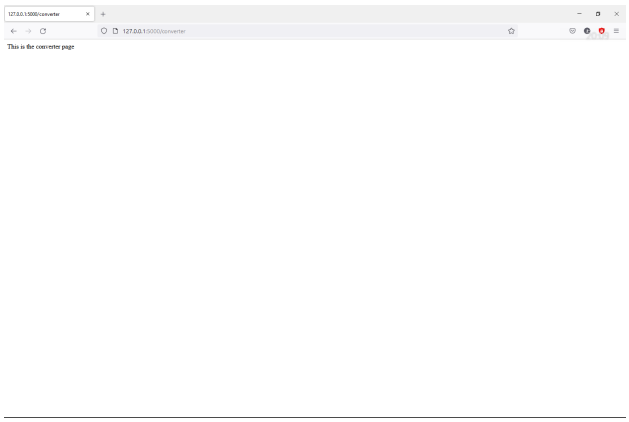
Basic Stylesheet

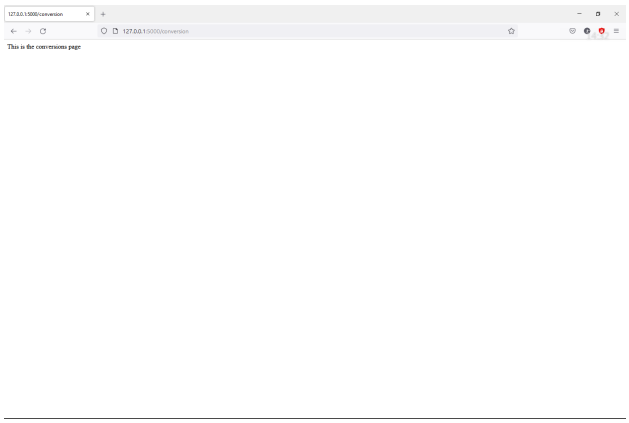
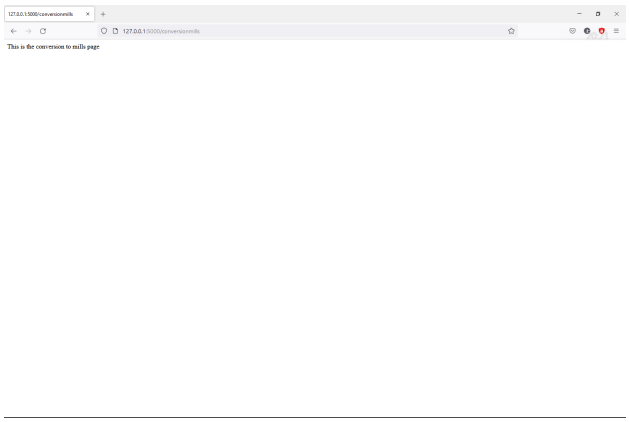
Reason for design:I wanted to apply styling which would be noticeable so I could see that the CSS had an effect on the templates. I would change this later but for testing purpose I would use it in order to see if there was a noticeable difference between the styled and unstyled pages.

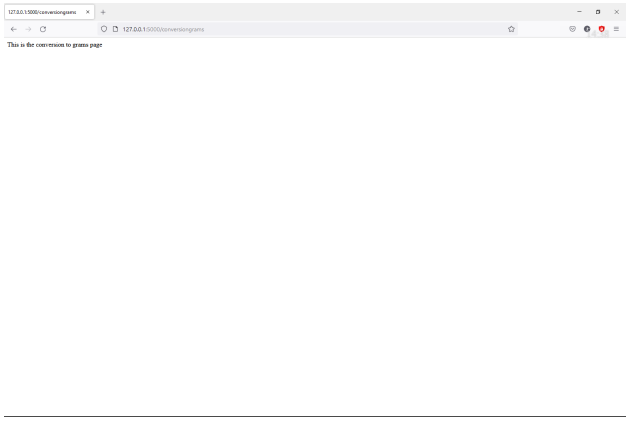
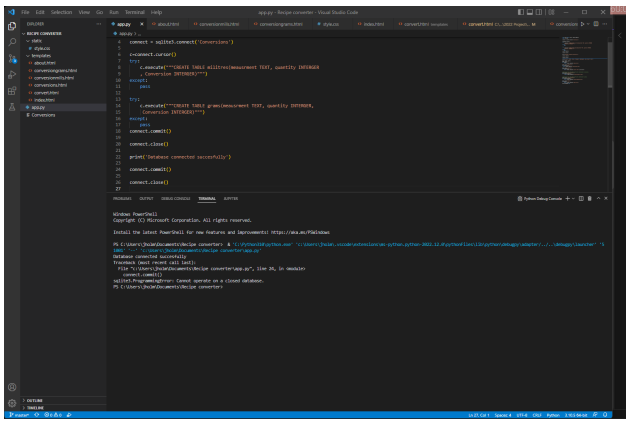
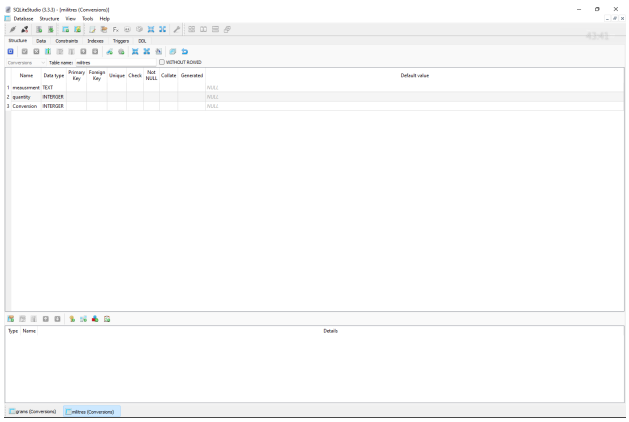
```
p {
    color:green;
    background-color: orange;
    font-size: 69px;
    font-family: 'Comic Sans MS';
}
```

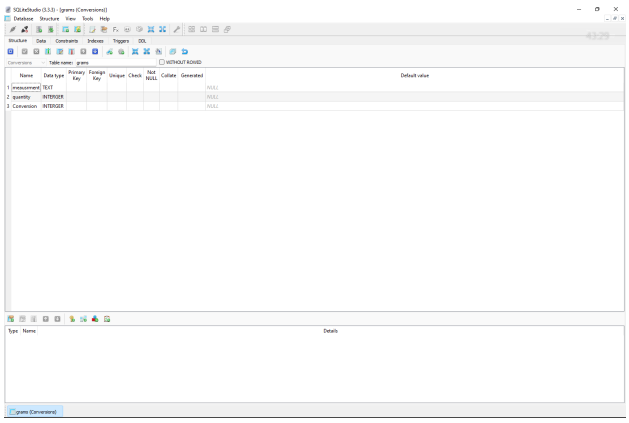
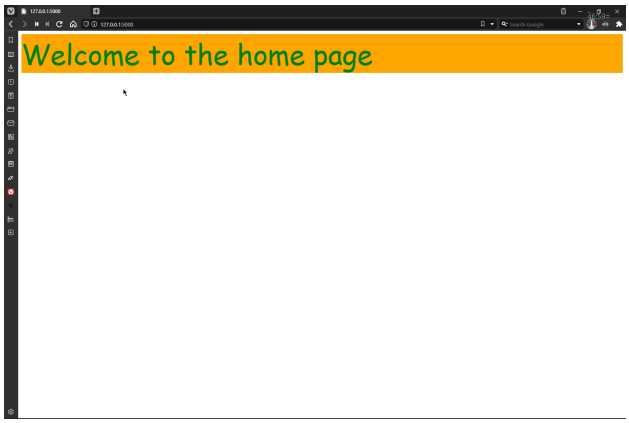
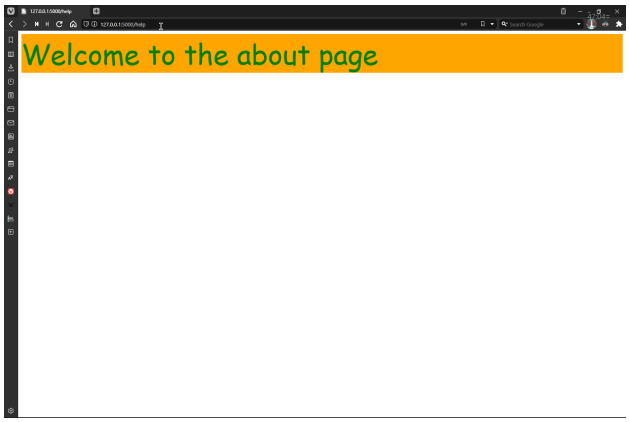
Testing

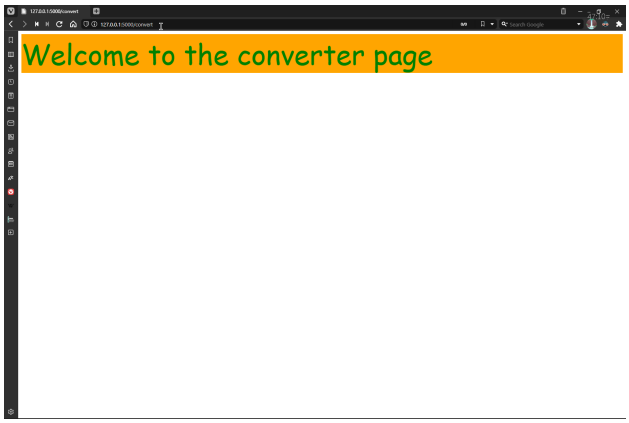
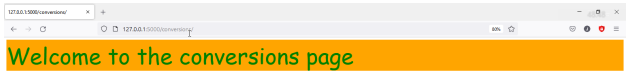
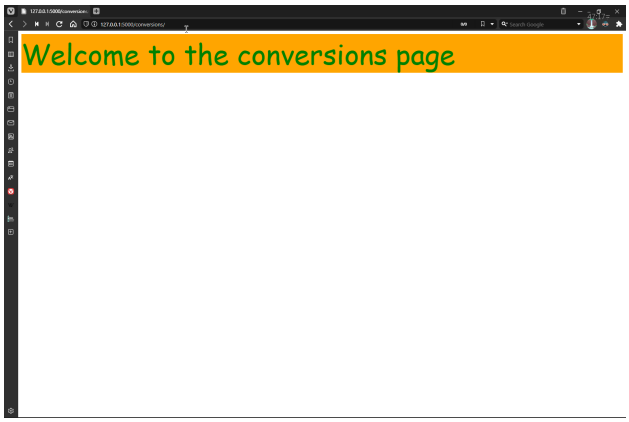
Test	Method	Result	Pass or Fail
See that route working on Chrome	Run program and see if any errors		Pass as I could see that the test was working correctly on Chrome.
See that route working on Firefox	Run program and see if any errors		Pass as I could see that the test was working correctly on Firefox
See that home page working properly on Chrome	Debug program and see that page renders correct template		Pass as template rendered correctly and was route correct on Chrome
See that home page working properly on Firefox	Debug program and see that page renders correct template		Pass as template rendered correctly and routed correct on Firefox

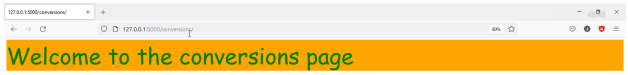
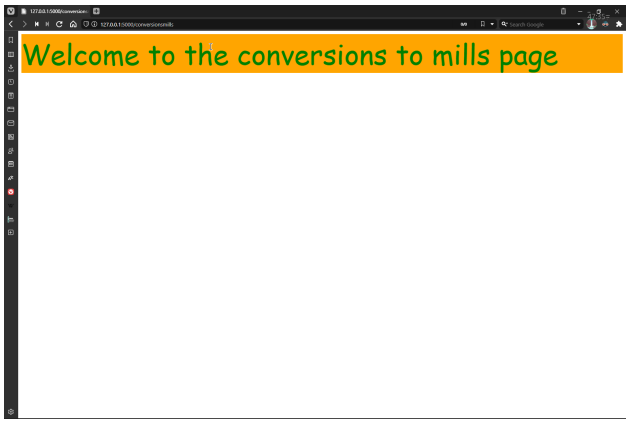
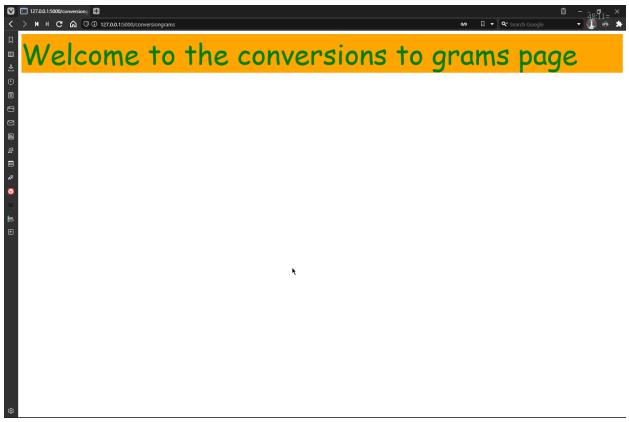
Test	Method	Result	Pass or Fail
See that About page is routed and rendered correctly on Chrome	Debug program enter "/about" into URL bar see that correct template is returned		Pass on Chrome as enter /about into URL bar rendered template and routed to correct page
See that About page is routed and rendered correctly on Firefox	Debug program enter "/about" into URL bar see that correct template is returned		Pass on Firefox as when '/about' entered into URL bar correct template is rendered and routed.
See if converter page rendering correctly on Chrome	Type '/converter' into URL bar and see that routes correctly and renders correct template		Pass on Chrome as when '/convert' entered into URL bar correct template is rendered and routed.
See if converter page rendering correctly on Firefox	Type '/converter' into URL bar and see that routes correctly and renders correct template		Pass on Firefox as when '/convert' entered into URL bar correct t

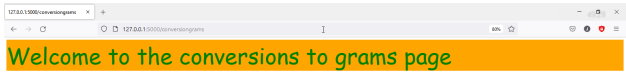
Test	Method	Result	Pass or Fail
See that Conversion page renders and routes correctly on Chrome	Enter '/conversion' will return correct template that should display This is the conversion page		ass on Chrome as when '/convert' entered into URL bar correct template is rendered and routed.
See that Conversion page renders and routes correctly on Firefox	Enter '/conversion' will return correct template that should display This is the conversion page		Pass on Firefox as when '/conversions' entered into URL bar correct template is rendered and routed.
See if Conversion to Millilitres page route and renders correctly on Chrome	Debug program enter "/conversionmills" into URL bar and see if page routes and renders correct template		Pass on Chrome as when '/conversionmills' entered into URL bar correct template is rendered and routed.
See if Conversion to Millilitres page route and renders correctly on Firefox	Debug program enter "/conversionmills" into URL bar and see if page routes and renders correct template		Pass on Firefox as when '/conversionmills' entered into URL bar correct template is rendered and routed.

Test	Method	Result	Pass or Fail
See that conversion to gram render and routed correctly on Chrome	Debug program then enter '/conversiongrams' into URL bar and see that it returns the correct template		Pass on Chrome as when '/conversiongrams' entered into URL bar correct template is rendered and routed.
See that conversion to gram render and routed correctly on Firefox	Debug program then enter '/conversiongrams' into URL bar and see that it returns the correct template		Pass on Firefox as when '/conversiongrams' entered into URL bar correct template is rendered and routed.
See if connected correctly	Create print statement that when database is created print message		Pass, as database was created properly via the program. Is a database in SQLite3.
See if table to grams created correctly	Debug program see open SQLite studio See if Table to grams created		Pass, table mills created correctly in program. Display in database of SQLite3.

Test	Method	Result	Pass or Fail
See if table to millilitres created correctly	Debug program see open SQLite studio See if Table to millilitres created		Pass, table grams created correctly in program. Display in database of SQLite3.
Test Styling applied to index page on Chrome	Enter whether "/" is entered into the URL bar see if styling has been applied to page		Pass on Chrome as when '/convert' entered into URL bar correct template is rendered and routed.
Test Styling applied to index page on Firefox	Enter whether "/" is entered into the URL bar see if styling has been applied to page		Pass on Firefox as when '/convert' entered into URL bar correct template is rendered and routed. With different styling applied.
Test Styling applied to about page on Chrome	Enter "/about" is entered into the URL bar see if styling has been applied to page		Pass on Chrome as when '/about' entered into URL bar correct template is rendered and routed.

Test	Method	Result	Pass or Fail
Test Styling applied to about page on Firefox	Enter "/about" is entered into the URL bar see if styling has been applied to page		Pass on Firefox as when '/convert' entered into URL bar correct template is rendered and routed. With different styling applied.
Test Styling applied to converter page	Enter "/convert" enter URL bar see if styling has been applied to page		Pass on Chrome as when '/convert' entered into URL bar correct template is rendered and routed. With different styling applied
Test Styling applied to converter page	Enter "/convert" enter URL bar see if styling has been applied to page		Pass on Firefox as when '/convert' entered into URL bar correct template is rendered and routed. With different styling applied.
Test Styling applied to conversion page on Chrome	Enter "/conversion" is entered URL bar see if styling has been applied to page		Pass on Chrome as when '/convert' entered into URL bar correct template is rendered and routed. With different styling applied

Test	Method	Result	Pass or Fail
Test Styling applied to conversion page on Firefox	Enter "/conversion" is entered URL bar see if styling has been applied to page		Pass on Firefox as when '/convert' entered into URL bar correct template is rendered and routed. With different styling applied.
Test Styling applied to conversion to millilitres page on Chrome	Enter "/conversiongrams" enter URL bar see if styling has been applied to page		Pass on Chrome as when '/convert' entered into URL bar correct template is rendered and routed. With different styling applied
Test Styling applied to conversion to millilitres page on Firefox	Enter "/conversiongrams" enter URL bar see if styling has been applied to page		Pass on Firefox as when '/convert' entered into URL bar correct template is rendered and routed. With different styling applied.
Test Styling applied to conversion to grams page on Chrome	Enter "/conversionmills" enter URL bar see if styling has been applied to page		Pass on Chrome as when '/convert' entered into URL bar correct template is rendered and routed. With different styling applied.

Test	Method	Result	Pass or Fail
Test Styling applied to conversion to grams page on Firefox	Enter "/conversionmills" enter URL bar see if styling has been applied to page		Pass on Firefox as when '/convert' entered into URL bar correct template is rendered and routed. With different styling applied.

Improvement

- Add styling so pages look more aesthetically pleasing
- Make it easier to navigate to other webpages
- Change templates so each serve a purpose E.g. Converter page works as form. Conversions page has links to conversion millilitres and grams pages.
- Format table in HTML pages.
- Insert test data into database.
- Change so user isn't able to enter negative amount of measurements
- Change so different step for input so can enter 1.5 or .25 of measurement.

Sprint 2

Outline

- Link pages together with anchor tags
- Use formatting to get base template for all pages
- Step number input so user has to enter a minimum of 0.25 and can't try to convert a negative amount of any measurement
- Database test data to see that it is inserted correctly.

Process

Base Template

Reason for design: I wanted to set up a base template that all other HTML templates can use and then add body based on what they needed. The template would include a base header which would have a navigation bar to other pages including the converter, conversions, about/help, and home page. This would also make it easier to make sure all the pages would have the same formatting as well.

Code for template

```
<header>
    <ul>
        <li><a href="{{ url_for('home')}}"> Home</a></li>
        <li><a href="{{ url_for('conversions')}}"> Conversions</a></li>
        <li><a href="{{ url_for('converter')}}"> Converter</a></li>
        <li><a href="{{ url_for('help')}}"> About</a> </li>
    </ul>
    <h1> This is the header</h1>
</header>




{% block content %}



{% endblock %}



<footer>This is the footer</footer>
```

Reason for design:I wanted to then apply styling to the base template in order to make it more aesthically appealing. This would also mean that the website would be more usable as it would be more intuitively set out.
Code for style sheet

```
header{
    color:black;
    background-color: whitesmoke;
    font-family: 'Comic Sans MS';
    font-size: 18px;
    margin: 0px;
    padding: 0px;
    text-align: left;
}

body{
    color:black;
    background-color: white;
    font-size: 10px;
    padding: 1mm;
    margin: 0.5mm;
    font-family: 'Comic Sans MS';
}
footer{
    color:black;
    background-color: whitesmoke;
    font-family: 'Comic Sans MS';
    font-size: 9px;
}

ul{
    margin: 0px;
    padding: 0px;
    list-style: none;
}

li{
    font-family: "Comic San MS";
    font-size: 20px;
    color: black;
}
```

```css
ul li {
    float: right;
    width: 250px;
    height: 50px;
    text-align: center;
    background-color: whitesmoke;
    color:black;
}
```

Converter

Reason for design:I wanted to make sure all elements of the forms were interactable and that the user could use all the built in functions. Current user input would serve no purpose but it later would be used to convert measurements.

Code for Template

```
{% extends 'base.html' %}
{% block content%}
<body>

    <form action='\conversions' methods='post'>


            <label>

                <input type="radio"> Militres

            </label>

            <label>

                <input type="radio"> Grams

            </label>


       <div>

        <label id='units mills'> <h3>Select a unit to convert to militres</h3>

            <label>

                <select name="unit" >

                    {% for unit in units_mills%}

                    <option> {{ unit }} </option>

                    {% endfor%}

                </select>

            </label>

        </label>

       </div>

       <div>

        <label id='unit grams' >
        <h3>Select a unit to convert to grams</h3>

            <label>

                <select name="unit">

                    {% for unit in units_grams%}

                    <option> {{ unit }} </option>

                    {% endfor%}

                </select>

            </label>

        </label>

       </div>



        <div>

            <label>
```

```
                <h2>Please enter the number of units you would like to convert?</h2>
                <p><input name='number of units' type='number' min=0.25>
                </label>
        </div>


        <p><input type="submit"> </p>
    </form>
{% endblock%}
```

Database and Tables

Insert Data into Database

Reason for design:I wanted to test that the program could insert data into the database. I acheived this through the use of a loop and try statement which would create the table and then insert data into the database by looping through two seperate lists of measurements and number to convert it and also add all the elements to a list which would be insertered later on.

I deleted the orginal tables in the database as it used a try so if the tables already existed it wouldn't function so couldn't test to see if the data was inserted. I then added more to the orginal code for the original create tables. (Measureverto Documentation > Database setup) I added a for loop which would go through all the measurements in units grams and units mills. It would then make a conversions multiplying the conversion to grams or mills using the measurement as a dictionary key. Then adding the measurement, number and conversion to a tuple which would be inserted into the table using the c.excutemany insert INTO (table) values.

The last code would test to make sure that the values were in the table by querying the data and printing it. This helped me to see if there was any errors.

```
import sqlite3
#Database
connect=sqlite3.connect('Conversion')
c=connect.cursor()
#Intial setup of database
try:
    connect=sqlite3.connect('Conversion')
    c=connect.cursor()
    #Create table to store measurement entered in mills
    c.execute("CREATE TABLE millilitres( unit TEXT,quantity INTERGER,conversion INTERGER)")
    #Create table to store measurement entered in grams
    c.execute("CREATE TABLE grams(unit TEXT,quantity INTERGER,conversion INTERGER)")
    units=('ounce','pound','stick','teaspoon','tablespoon','cup','quart','pint')
    units_grams=('ounce','pound', 'stick')
    units_mills=('teaspoon','tablespoon','cup','quart','pint')
    conversion_to_grams = {'ounce':28.35,'pound':454,'stick':113}
    conversion_to_mills ={'cup':240,'cups':240,'teaspoon':5,
    'tablespoon':15,'pint':473,'quart':946}
    #This code snippet not working not what program looks like. >:(
    print("Database Setup")
    connect.commit()
    list_mills=[]
    list_grams=[]
    numbers=[1,2,3]
    numbers_mills=[1,2,3,4,5]
    for (measure,number) in zip(units_mills,numbers_mills):
        final_conversion_mills=conversion_to_mills[measure.lower()]*number
        add_to_table=(measure,number,final_conversion_mills)
        list_mills.append(add_to_table)
    for (measure,number) in zip(units_grams,numbers):
        final_conversion_grams=conversion_to_grams[measure.lower()]*number
        add_to_table=(measure,number,final_conversion_grams)
        list_grams.append(add_to_table)
    c.executemany("INSERT INTO grams VALUES (?,?,?)",list_grams)
    c.executemany("INSERT INTO millilitres VALUES (?,?,?)",list_mills)
    print('Connected to database and test data inserted')
    connect.commit()
    connect.close()
except:
    pass
finally:
    print("Database connected")
#Get data from tables in database
#Query data from grams table in conversions Database
try:
 c.execute("SELECT * FROM grams")
 DisplaytoTableGrams=tuple(c.fetchall())
 print("Data queried from Table grams")
 print(DisplaytoTableGrams)
except:
 pass
#data from mills table
try:
 c.execute("SELECT * FROM millilitres")
 DisplaytoTableMills=tuple(c.fetchall())
 print("Data queried from Table Mills")
 print(DisplaytoTableMills)
except:
 pass
```

Conversion Mills

Reason for design: I wanted to display the data which was added to table mills through the loop. I achieved this through querying the data assigning it to a tuple called "add_to_table_mills".

The elements 'heading_table_mills' stores all the values for the table headings as a tuple within python. Jinja would loop through these elements to create headings for these in the HTML table using a for loop. Elements "add_to_table_mills" would then be passed through to the template where Jinja to insert these as rows. Looping through the element that is a tuple to get the individual values for the data in the rows.

Code for route

```
add_to_table_mills=c.exceute("SELECT * FROM millilitres")

@app.route("/conversionmills")
def conversion_mills():
                return render_template('conversion_mill.html',
                heading_table_mills=heading_table_mills,
                add_to_table_mills=add_to_table_mills)

#Code snippet also not working all suposed to be one same line as return render template.
```

Code for template

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

{% block content%}

<body>

    This is a table showing all recent conversion to mililitres.

    <table>

        <tr>

            {% for heading in heading_table_mills%}

            <th> {{ heading }}</th>

            {% endfor%}

        </tr>

        {% for data_mills in add_to_table_mills%}

            <tr>

                {% for data in data_mills %}

                <td> {{ data }}</td>

                {% endfor%}

            </tr>

        {% endfor%}

    </table>

</body>



{% endblock%}
```

Conversion Grams

Reason for design: This followed the same design principles as Conversion mills Conversion Mills. Except instead of taking data from the "mililitres" table it would take data from the "Grams" table instead.

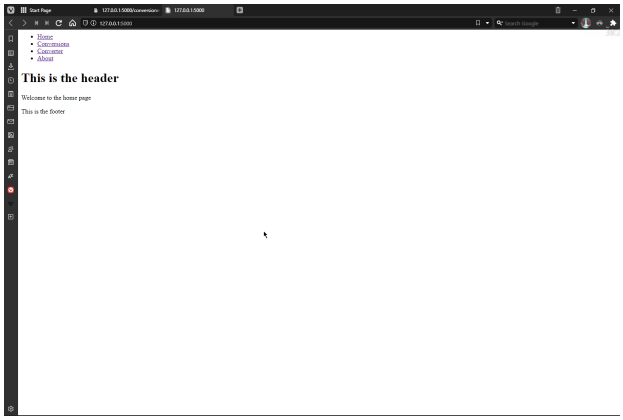
Code for route

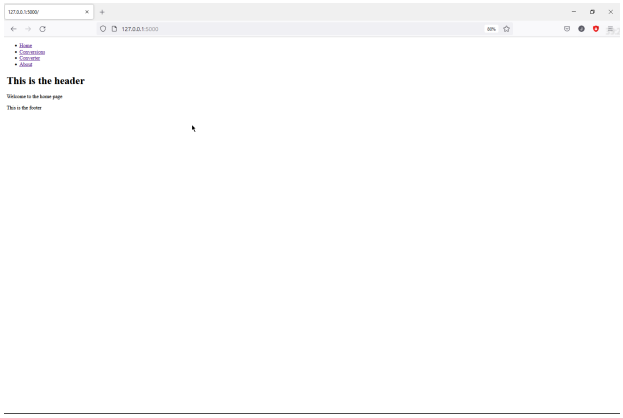
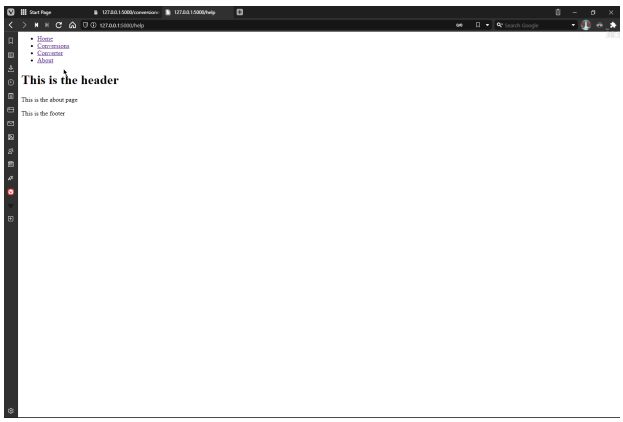
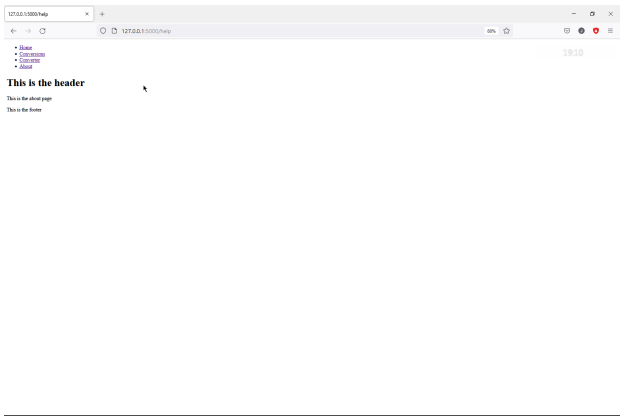
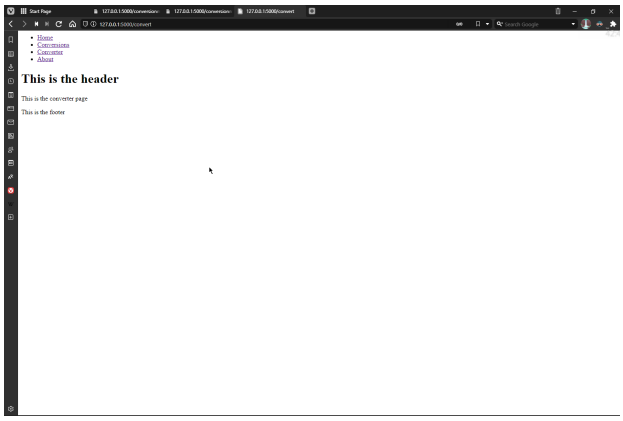
```
@app.route("/conversiongrams")
def conversion_grams():
    return render_template('conversion_grams.html',
    heading_table_grams=heading_table_grams,
    add_to_table_grams=add_to_table_grams)
    # Code snippet again not working all one row but won't display.
```

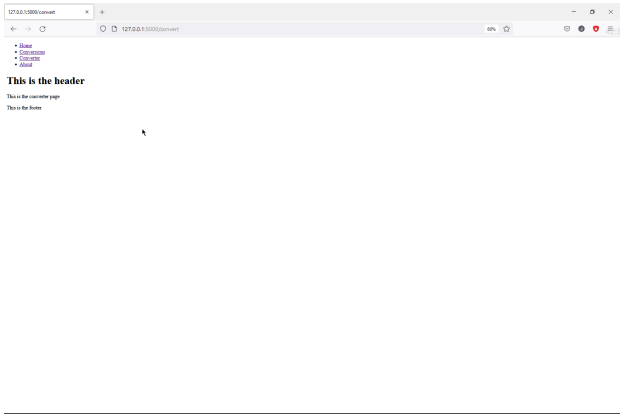
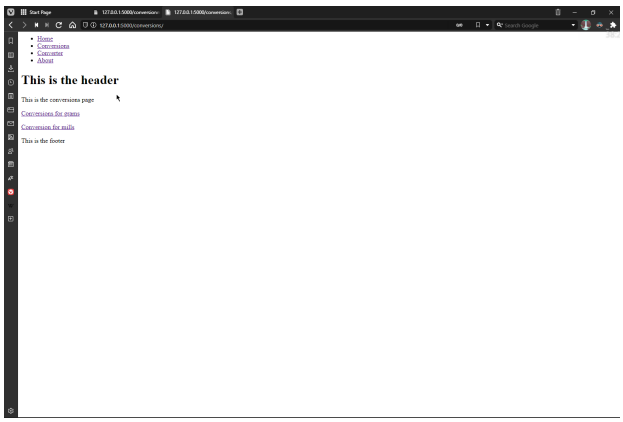
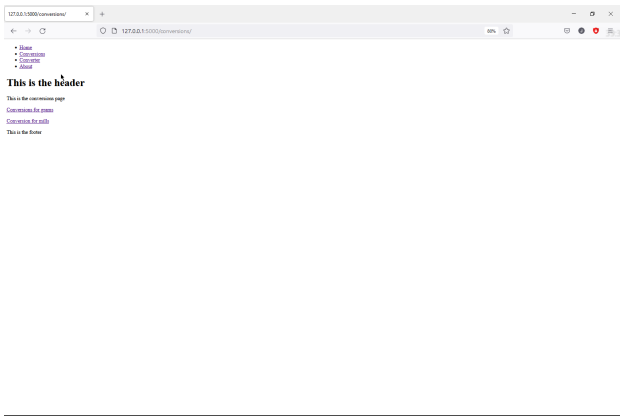
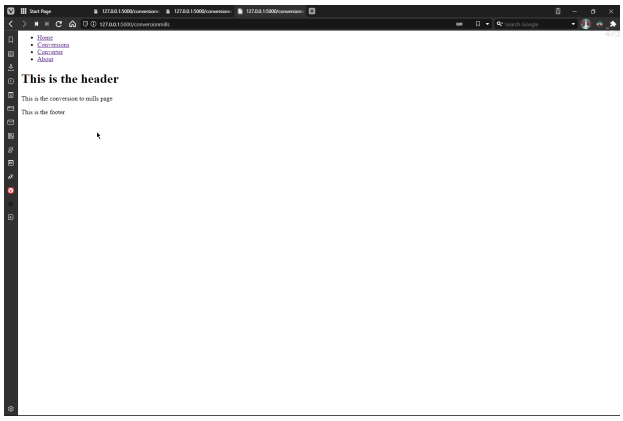
Code for template

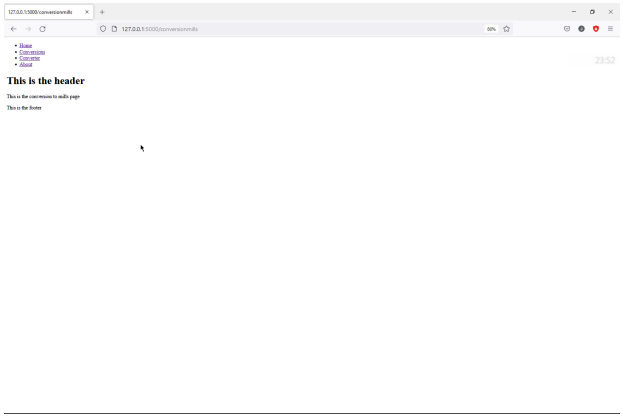
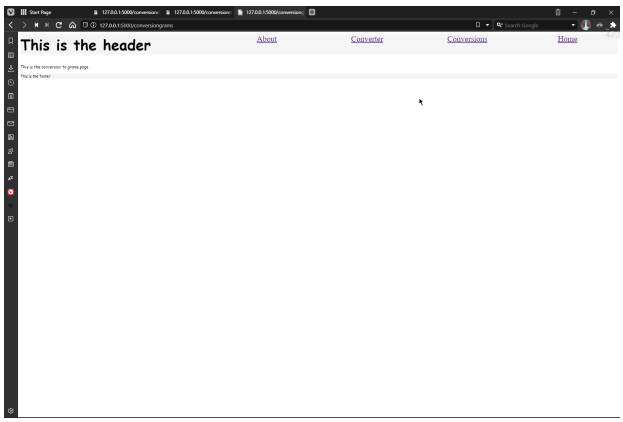
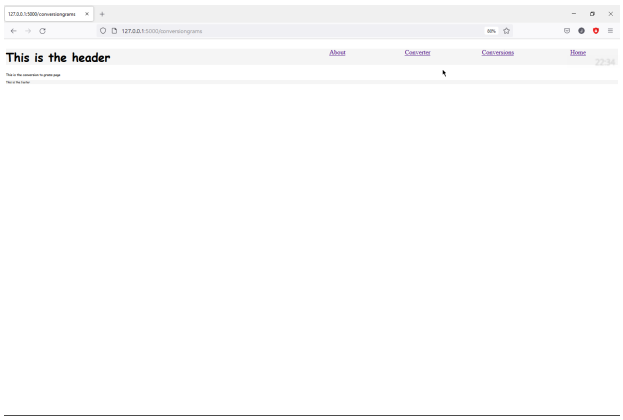
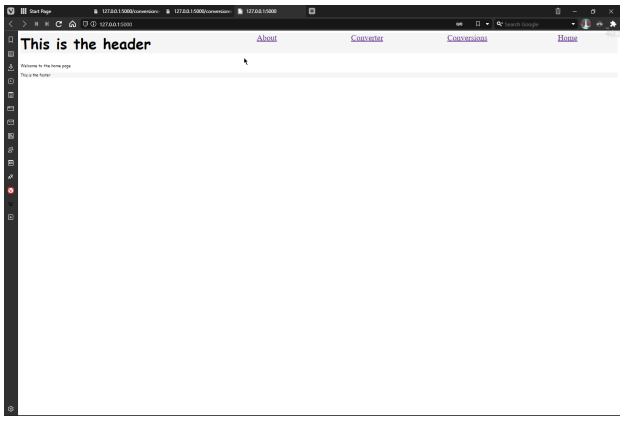
```
{% extends "base.html "%}

{% block content%}

<body>

    This is a table showing all recent conversion to grams.

    <table>

     <tr>

            {% for heading in heading_table_grams%}

            <th> {{ heading }}</th>

            {% endfor%}

        </tr>

        {% for data_grams in add_to_table_grams%}

            <tr>

                {% for data in data_grams %}

                <td> {{ data }}</td>

                {% endfor%}

            </tr>

        {% endfor%}

    </table>


</body>

{% endblock%}
```

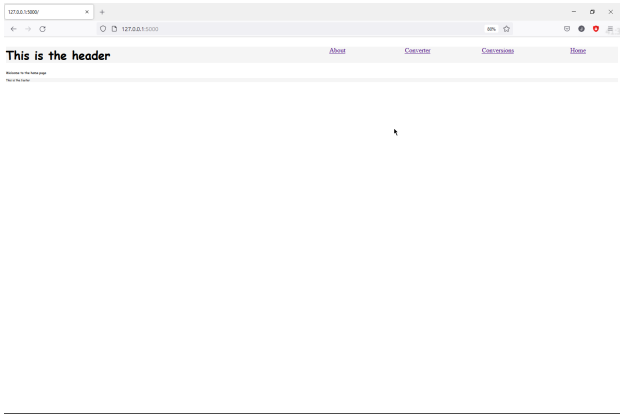
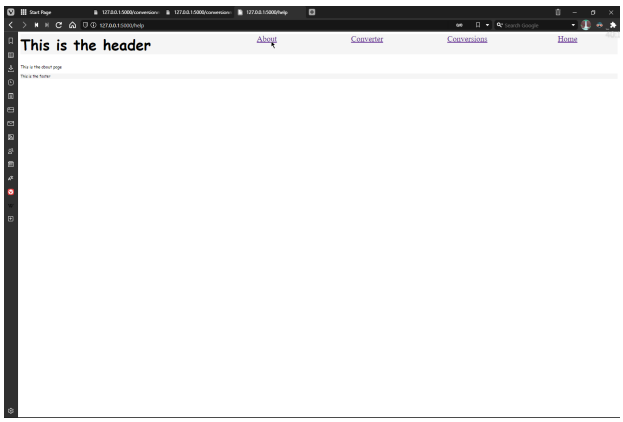
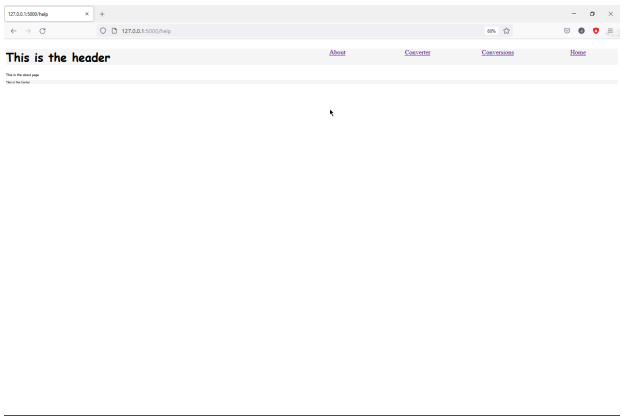
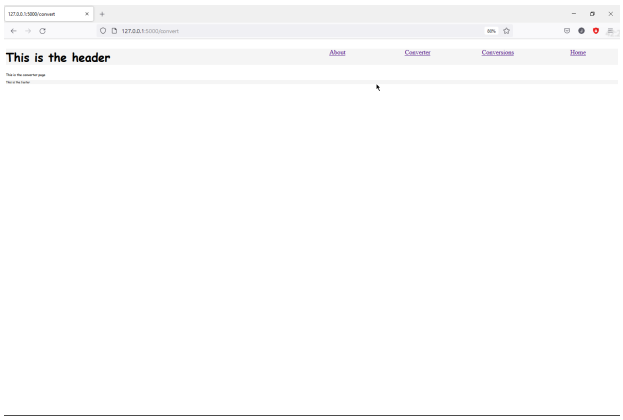
Testing

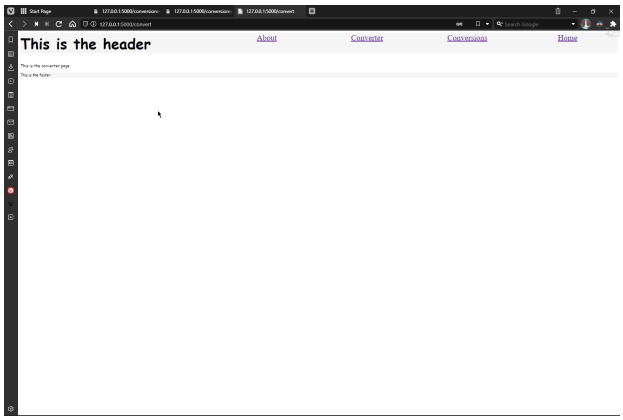
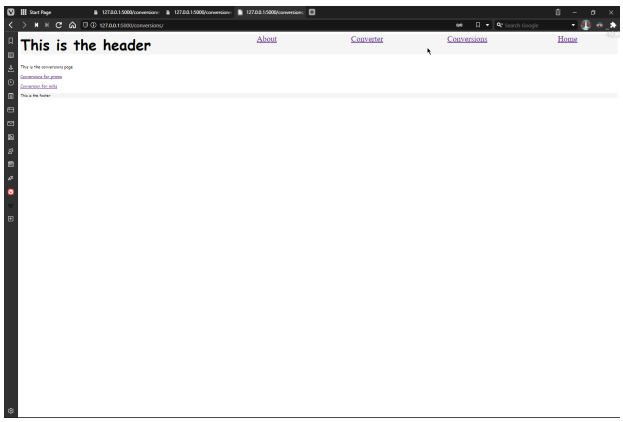
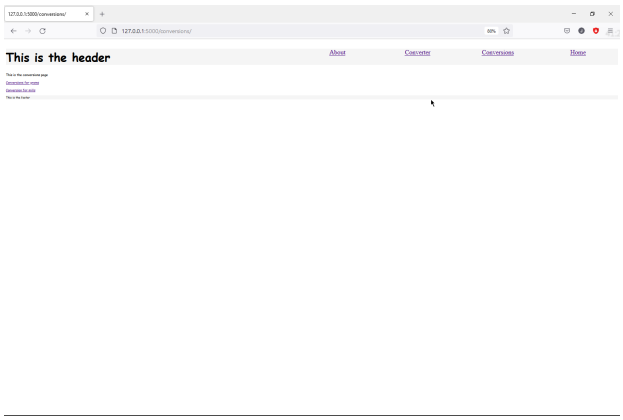
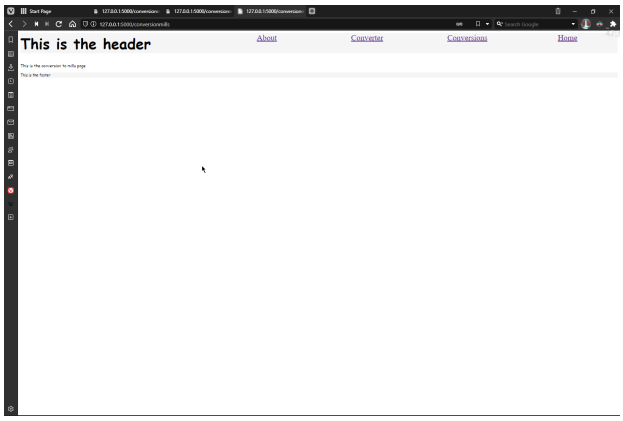
Test	Method	Result chrome	Result Firefox
See that base template applied Home page on Chrome	Debug program see that header and footer are there and thus base template working		Pass as new base template was rendered correctly on Home page in Chrome

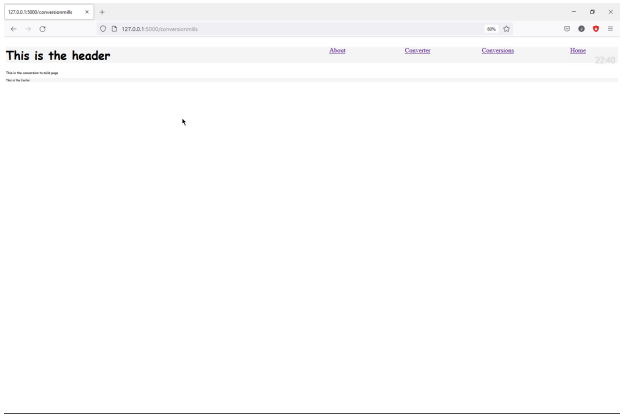
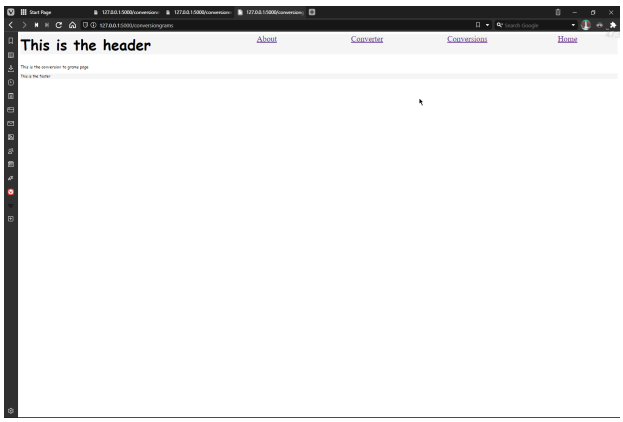
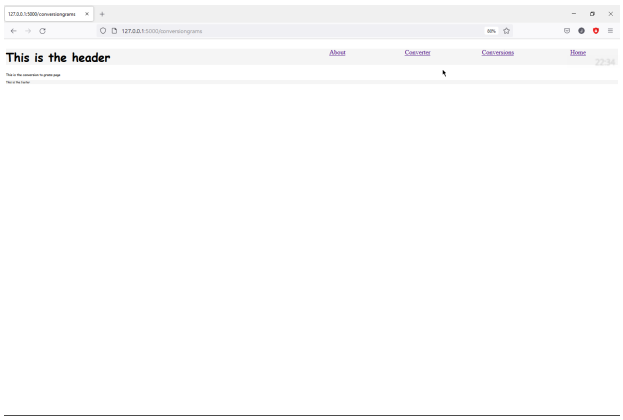
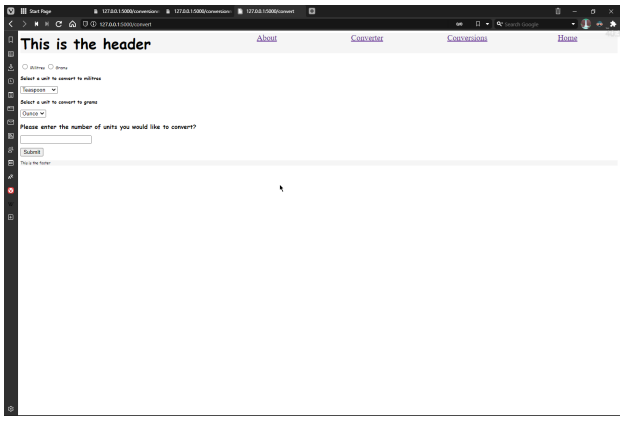
Test	Method	Result chrome	Result Firefox
See that base template applied Home page on Firefox	Debug program see that header and footer are there and thus base template working		Pass as new base template was rendered correctly on Home page in Firefox
See that base template applied About page on Chrome	Debug program see that header and footer are there and thus base template working		Pass as new base template was rendered correctly on about page in Chrome
See that base template applied About page on Firefox	Debug program see that header and footer are there and thus base template working		Pass as new base template was rendered correctly on about page in Firefox
See that base template applied Converter page on Chrome	Debug program see that header and footer are there and thus base template working		Pass as new base template was rendered correctly on converter page in Chrome

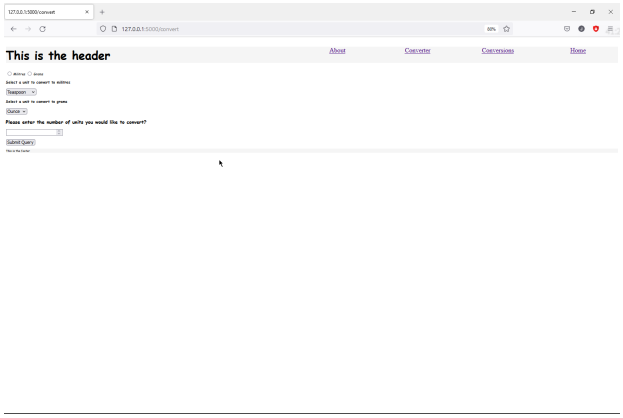
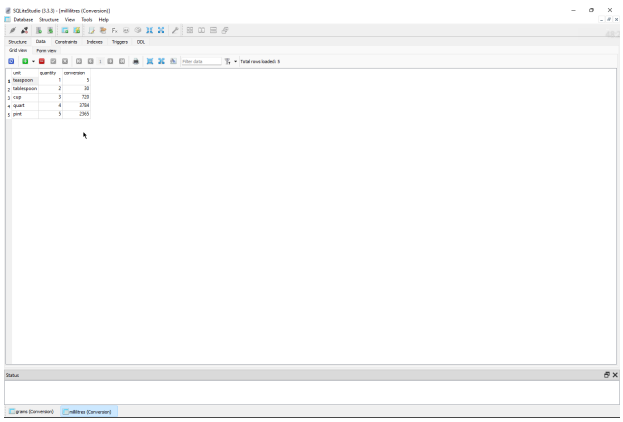
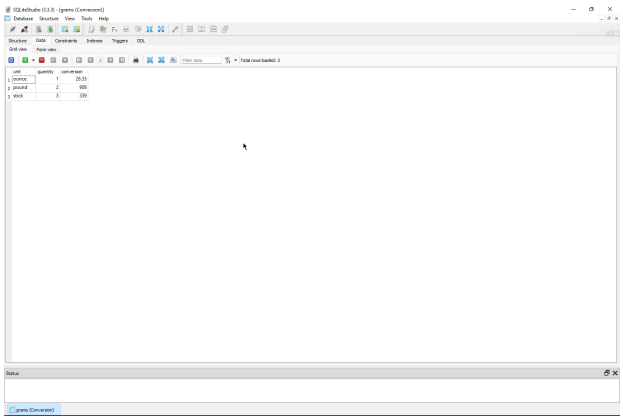
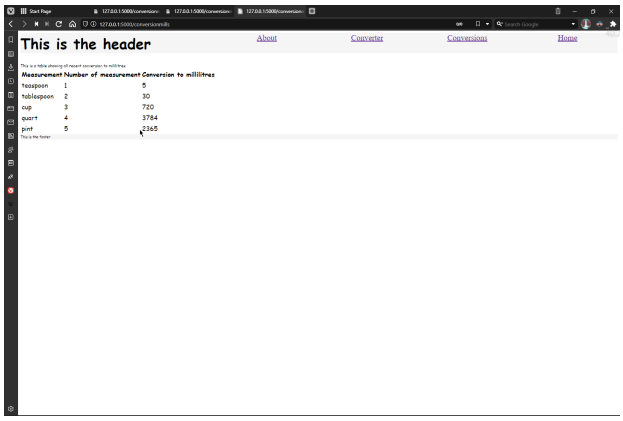
Test	Method	Result chrome	Result Firefox
See that base template applied Converter page on Firefox	Debug program see that header and footer are there and thus base template working		Pass as new base template was rendered correctly on converter page in Firefox
See that base template applied Conversions page on Chrome	Debug program see that header and footer are there and thus base template working		Pass as new base template was rendered correctly on conversions page in Chrome
See that base template applied Conversions page on Firefox	Debug program see that header and footer are there and thus base template working		Pass as new base template was rendered correctly on conversions page in Firefox
See that base template applied Conversion to mills page on Chrome	Debug program see that header and footer are there and thus base template working		Pass as new base template was rendered correctly on conversions to mills page in Chrome

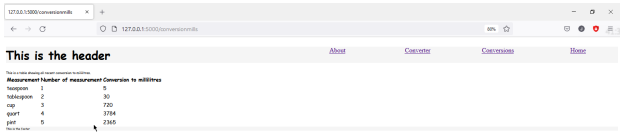
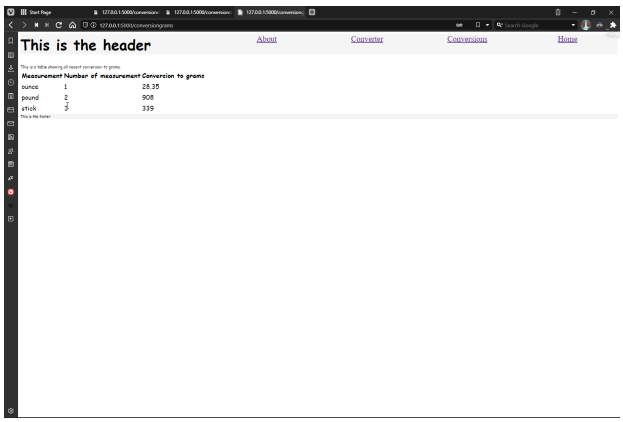
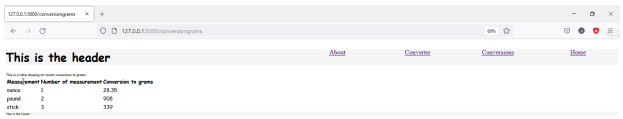
Test	Method	Result chrome	Result Firefox
See that base template applied Conversion to mills page on Firefox	Debug program see that header and footer are there and thus base template working		Pass as new base template was rendered correctly on conversion to mills page in Firefox
See that base template applied Conversion to grams page on Chrome	Debug program see that header and footer are there and thus base template working		Pass as new base template was rendered correctly on conversions to grams page in Chrome
See that base template applied Conversion to grams page on Firefox	Debug program see that header and footer are there and thus base template working		Pass as new base template was rendered correctly on conversion to grams page in Firefox
See that base template applied Home page with added styling on Chrome	Debug program see that header and footer are there and thus base template working		Pass as new base template was rendered correctly on home page in Chrome with styling applied.

Test	Method	Result chrome	Result Firefox
See that base template applied Home page with added styling on Firefox	Debug program see that header and footer are there and thus base template working		Pass as new base template was rendered correctly on home page in Firefox with styling applied.
See that base template applied About page with added styling on Chrome	Debug program see that header and footer are there and thus base template working		Pass as new base template was rendered correctly on about page in Chrome with styling applied.
See that base template applied About page with added styling on Firefox	Debug program see that header and footer are there and thus base template working		Pass as new base template was rendered correctly on about page in Firefox with styling applied.
See that base template applied Converter page with added styling on Chrome	Debug program see that header and footer are there and thus base template working		Pass as new base template was rendered correctly on converter page in Chrome with styling applied.

Test	Method	Result chrome	Result Firefox
See that base template applied Converter page with added styling on Firefox	Debug program see that header and footer are there and thus base template working		Pass as new base template was rendered correctly on converter page in Firefox with styling applied.
See that base template applied Conversion page with added styling on Chrome	Debug program see that header and footer are there and thus base template working		Pass as new base template was rendered correctly on conversions page in Chrome with styling applied.
See that base template applied Conversion page with added styling on Firefox	Debug program see that header and footer are there and thus base template working		Pass as new base template was rendered correctly on conversions page in Firefox with styling applied.
See that base template applied Conversion to Mills page with added styling on Chrome	Debug program see that header and footer are there and thus base template working		Pass as new base template was rendered correctly on conversion to mills page in Chrome with styling applied.

Test	Method	Result chrome	Result Firefox
See that base template applied Conversion to Mills page with added styling on Firefox	Debug program see that header and footer are there and thus base template working		Pass as new base template was rendered correctly on conversion to mills page in Firefox with styling applied.
See that base template applied Conversion to Grams page with added styling on Chrome	Debug program see that header and footer are there and thus base template working		Pass as new base template was rendered correctly on conversion to grams page in Chrome with styling applied.
See that base template applied Conversion to Grams page with added styling on Firefox	Debug program see that header and footer are there and thus base template working		Pass as new base template was rendered correctly on conversion to grams page in Firefox with styling applied.
See that new template is render and interactable on Chrome	Debug program enter "/convert" into URL bar		Pass on Chrome as form is interactable and all HTML elements are present

Test	Method	Result chrome	Result Firefox
See that new template is render and interactable on Firefox	Debug program enter "/convert" into URL bar		Pass on Firefox as form is interactable and all HTML elements are present
See that Data inserted into Table millilitres	Debug program then after debugged open SQLite studio. Connect to database and see data inside table.		Pass as test data from file has been successfully inserted into the SQLite3 database in the millilitres table.
See that Data inserted into Table grams	Debug program then after debugged open SQLite studio. Connect to database and see data inside table.		Passas test data from file has been successfully inserted into the SQLite3 database in the grams table.
See conversion to mills page working on Chrome	Debug program enter "/conversionmills" into URL see that table is rendered and that has heading and data from Database.		Pass on Chrome as all test data from millilitres table has been queried and displayed on the HTML.

Test	Method	Result chrome	Result Firefox
See conversion to mills page working on Firefox	Debug program enter "/conversionmills" into URL see that table is rendered and that has heading and data from Database.		Pass on Firefox as all test data from millilitres table has been queried and displayed on the HTML.
See that conversion to grams page render correctly on Chrome	Debug program see that runs enter '/conversiongrams' into URL bar see that correct template rendered		Pass on Chrome as all test data from grams table has been queried and displayed on the HTML.
See that conversion to grams page render correctly on Firefox	Debug program see that runs enter '/conversiongrams' into URL bar see that correct template rendered		Pass on Firefox as all test data from grams table has been queried and displayed on the HTML.

Improvement

- Make header fixed to top of page.
- Get real data from user instead of test data for millilitres and grams.
- Hide fields based on selection.
- Make background more aesthetically appealing.

Sprint 3

Outline

- Hide form fields based on what user radio button selection
- Format header and footer to fully extend to edge of page
- Post user responses from form into database
- Convert user responses from form using functions for converting to grams or mills
- Set a background image for the website.
- Create example text of what will be included on all pages in about page.
- Add pattern to table
- Add pattern to dropdown list
- Customize radio buttons

Process

Header and Footer

Reason for design: I wanted to make sure that the header was fixed to the top of the page as well as position it so that it would take up all available space without having any gaps. I wanted to do the same with the footer as well.

```
header{
    color:black;
    background-color: whitesmoke;
    font-family: 'Comic Sans MS';
    font-size: 15px;
    margin: 0px;
    padding: 0px;
    text-align: left;
    left:0;
    right:0;
    width:100%;
    top:0;
    left:0;
    right:0;
    position:fixed;
}

footer{
    color:black;
    background-color: whitesmoke;
    font-family: 'Comic Sans MS';
    font-size: 9px;
    left:0;
    right:0;
    width:100%;
    position: fixed;
    bottom: 0;
    left:0;
    right:0;
}
```

Background image for website

Reason for design: I wanted to change the styling so that instead of the default white background I could have a unique image instead. I needed to also add padding to the body so that the text on the pages would be displayed properly and be readable as without it the header would cover the body text.

```
body{
  color:rgb(0, 0, 0);
  background-image:url('/static/images/logo.png');
  font-size: 20px;
  padding: 2mm;
  padding-top: 150px;
  margin: 8.5mm;
  font-family: 'Comic Sans MS';
  position: relative;
}
```

Redesigning pages

Reason for design:I wanted to remake both the home and about page to have a more significant purpose. As since the converter, conversions, and conversions to grams and mills page had additional features added I wanted more to be added to home and About page as well. I decided on the home page I would add a description of the website. And the about page I would describe what the website was made for as well as technical things like what measurements it could convert.

Template for Home

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

{%  block content%}

<body>

    <p> Welcome to Measureverto</p>

    <p> This is a website built using Flask, Sqlite3, and CSS. I hope you enjoy </p>

</body>

{% endblock %}
```

Template for About

```
{% extends "base.html" %}
{% block content%}
<p> This is Measureverto. A website I made for my school project for NCEA level 2</p>
<p> Measureverto is a website used to convert cooking measurements E.g. Cups, Quarts, Ounces, Pounds. To standard units
either grams or mililitres depending on the unit</p>
{% endblock%}
```

Reason for design: I wanted to apply styling to the table so that all data would be presented in a easy to read and understandable way for the user. This would imporve the usability of the website by being easier to determine the exact conversion that the program has made.

```
/*Table styling for tables in conversion to grams and mills pages. */
th, td {
    text-align: left;
   font-family: "Comic San MS";
   padding: 15px;
}
/*Table heading styling */
th {
    font-size: 25px;
}
/* Table data styling*/
td{
    font-size: 20px;
}
```

Post Data into Database

Reason for design: I wanted to finally add functionality to the converter page by posting the user responses from the form into the data base. I did this by importing the request module from flask. This allowed me to access the values which the user had enter in the form. As well I made sure to add Ids to all of the elements as well as values so that the program could correctly take the selected values from the user.

Code for route

```
@app.route('conversion')
conversions(methods='GET','POST'):
        if request.method=='POST':
                millsorgrams=request.form['millilitres or Grams']
                if millsorgrams == 'millilitres':
                    measurement_mills=request.form['unitmills']
                    # This gets the measurement that the user has selected from the form
                    print(measurement_mills)
                    #Print statement to check what the measurement is.
                    number_measurements=float(request.form['numberofunits'])
                     # This gets the number of measure that the user has enter in the form
                    print(number_measurements)
                    # Print statment to check what number is
                    conversion_mills=float(conversion_to_mills_dict[measurement_mills])*float(number_measurements)
        #This gets the usermeasurement from the form and then converts it  into mills using it as a key in a dictionary.
This is then multiplied by the number of measurements to get the final conversion.
                    #Print conversion to see if it is correct.
                  add_to_tale_mills(number_measurements,measurement_mills,conversion_mills)
                    #This is tuple which will be inserted into the Database.
                    print(add_to_tale_mills)
                    # See if it is correct.
                    connect=sqlite3.connect('Conversion')
                    c=connect.cursor()
                    c.execute("INSERT INTO millilitres VALUES(?,?,?)",add_to_tale_mills)
                    # Inserts form data into database.
                    connect.commit()
                    connect.close()
                    return render_template('conversions.html')
                else:
                    measurement_grams=request.form['unitgrams']
                    print(measurement_grams)
                    number_measurements=request.form['numberofunits']
                    print(number_measurements)
                    conversion_grams=float(conversion_to_grams_dict[measurement_grams])*float(number_measurements)
                    #This gets the usermeasurement from the form and then converts it into grams using it as a key in a
dictionary. This is then multiplied by the number of measurements to get the final conversion.
                  print(conversion_grams
                      add_to_tale_grams(number_measurements,measurement_grams,conversion_grams)
                    #This is tuple which will be inserted into the Database.
                    print(add_to_tale_grams)
                    connect=sqlite3.connect('Conversion')
                    c=connect.cursor()
                    c.execute("INSERT INTO grams VALUES(?,?,?)",add_to_tale_grams)
                    # Inserts form data into database.
                    connect.commit()
                    connect.close()
                    return render_template('conversions.html')
            else:
                return render_template('conversions.html',)
```

Write function to hide dropdown based on radio button

Reason for design:I wanted to create a function which would hide the dropdown list based on whether the use choose to convert millilitres or grams. I achieved this through creating a function in JavaScript which would display the block based upon which was selected.

Hide input based on dropdown

Reason for design: I again following the same logic as the previous wanted to display an input to enter the number of measurements that the user wanted to convert once they selected a measurement from the dropdown list for either grams or millilitres. This would increase the functionality of the program by meaning that the form would follow a straight forward and linear process so that people who are less technically inclined can understand what to do. Also increase aesthics from not having the screen being cluttered with many unrelated options.

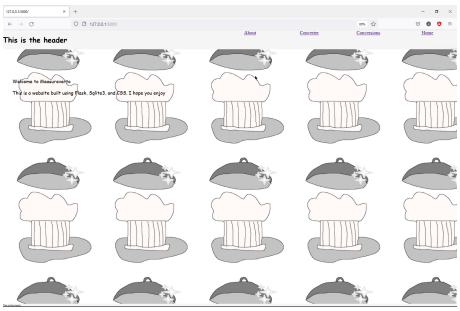
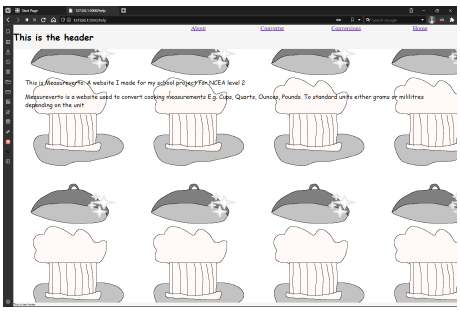
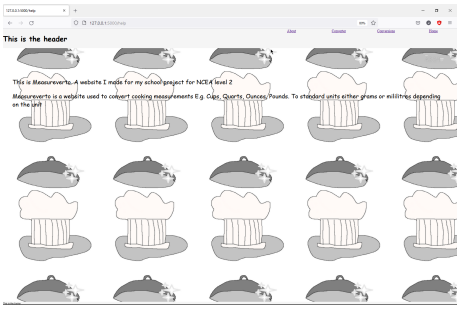
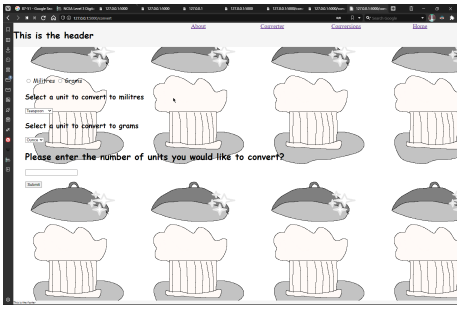
Hide submit function

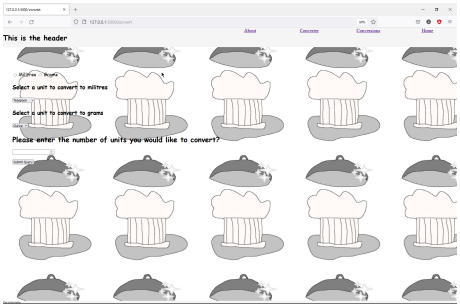
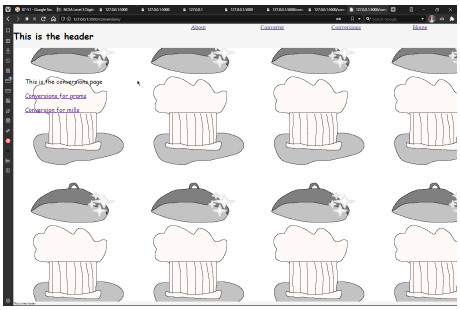
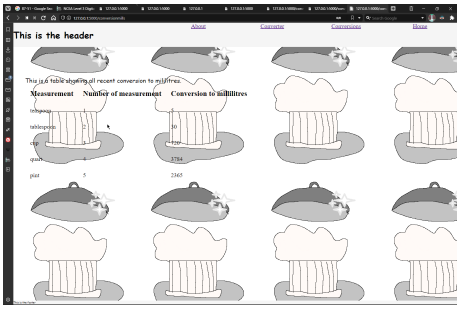
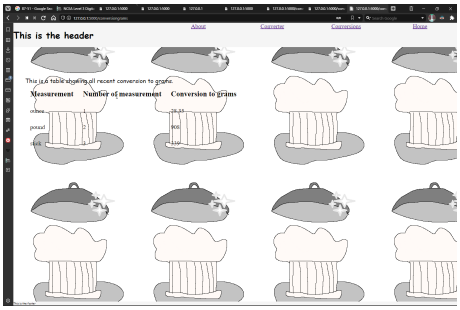
Reason for design:Following the same logic as before I would again repurpose the similar function to fit a new purpose. As when the user has interacted with the number input the sumbit button will appear. This would prevent errors with the database from submitting empty forms when the page eventually posted to the database. As well as again meaning that it would be easier for the end user to use the website to convert cooking measurments.

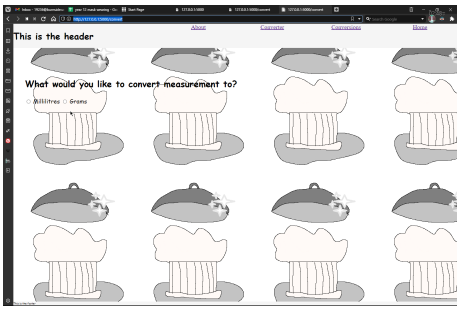
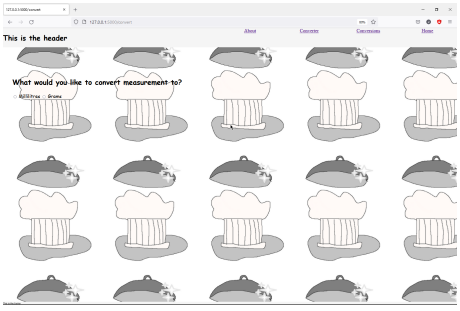
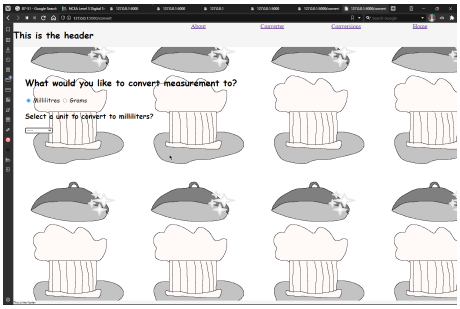
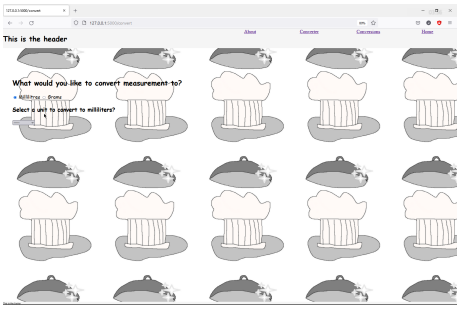
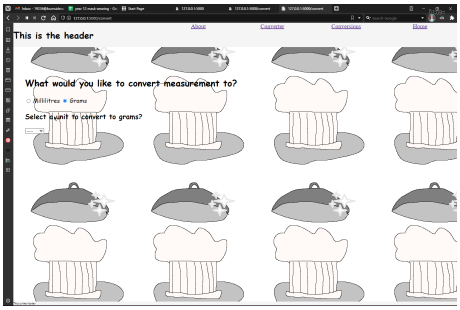
Code for template

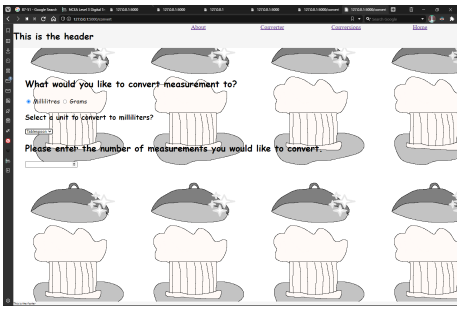
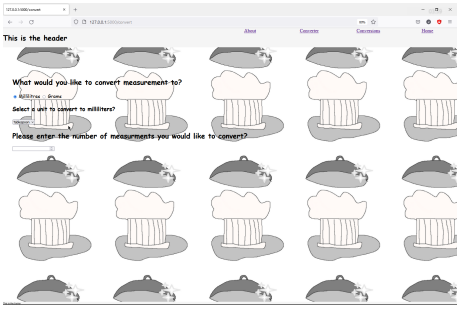
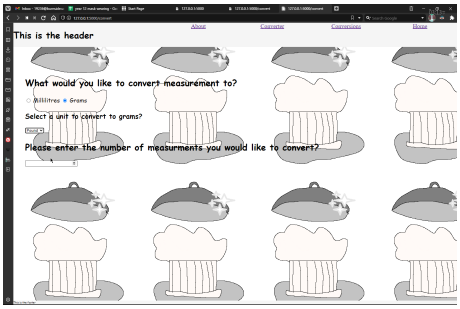
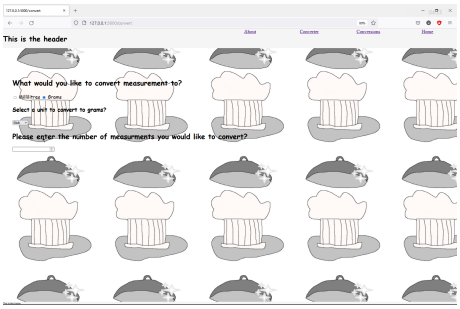
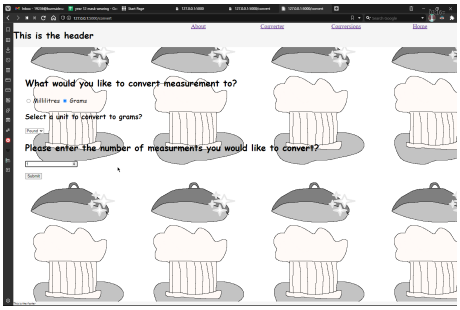
```
<script>
function hideform(y) {
        if (y=='Millilitres') {           document.getElementById('unitmills').style.display='block';//Will show dropdown
list for measurement to convert to mills
            document.getElementById('unitgrams').style.display='none';// Will hides measurment that convert to grams if
the user has previously selected it.
        }
        else  {
            document.getElementById('unitgrams').style.display='block';//Will show a dropdown list for measurement that
the user can convert to grams.
                                document.getElementById('unitmills').style.display='none';//Will not show dropdown list
for units to grams. Also hide this dropdown list if user changed from mills to grams.
        }
        return;
    }

function showsubmit() {
        document.getElementById('Submit').style.display='block'
    }

function showsubmit() {
        document.getElementById('Submit').style.display='block'
    }

</script>

<input type="radio"  name="Millilitres or Grams" id="Millilitres" value="Millilitres" onclick="hideform('Millilitres')">
Millilitres <!-- Radio button for millilitres which will run script if pressed.-->
<input type="radio"  name="Millilitres or Grams" id="Grams" value="Grams" onclick="hideform('Grams')"> Grams
<label hidden id="NumberMeasurements">
<h2>Please enter the number of <span id="quantity"></span>s you would like to convert?</h2>
<p><input name='numberofunits' type='number' min=0.25 step="0.25" onchange="showsubmit()">
</label>
<p hidden id="Submit"><input type="submit"></p>
```

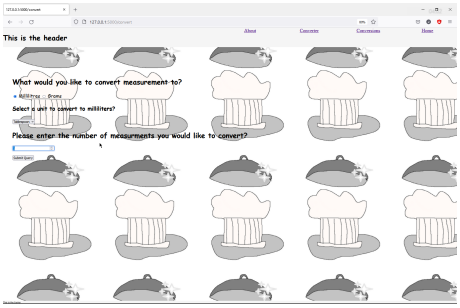
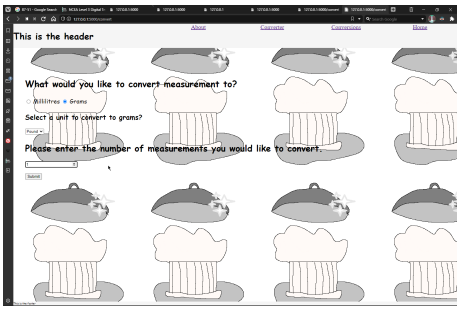
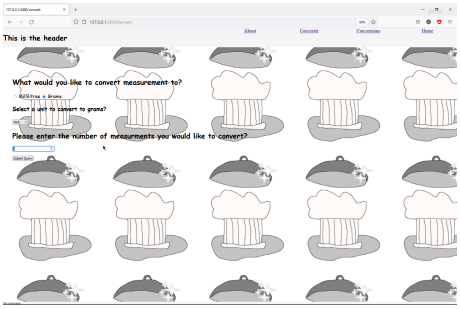
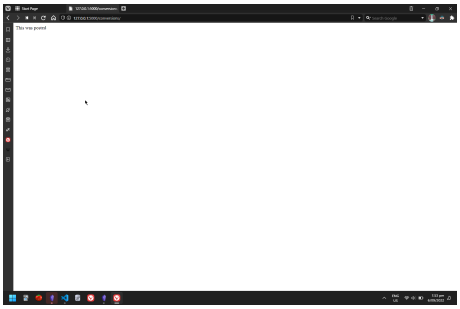
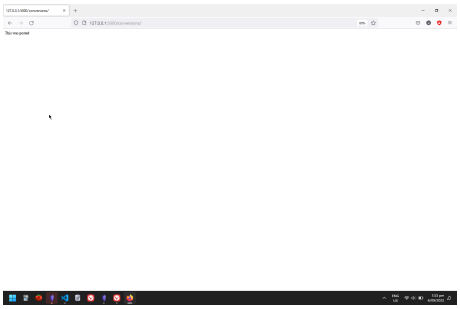
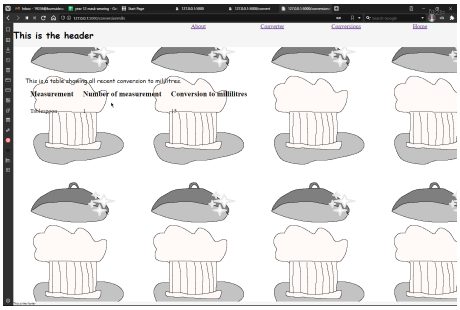
Testing

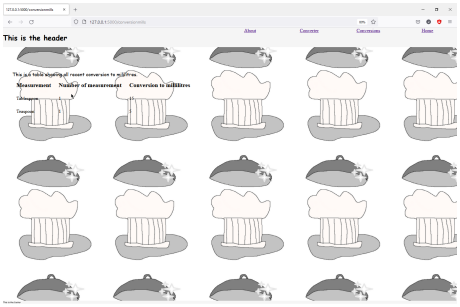
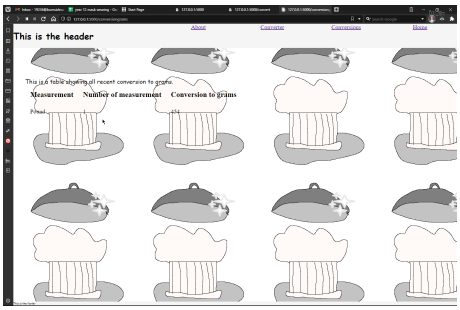
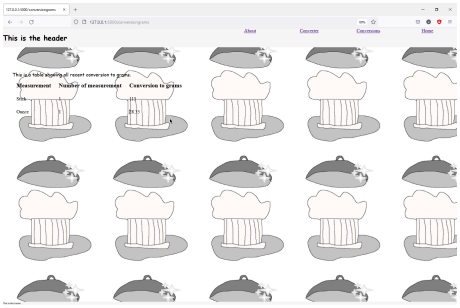
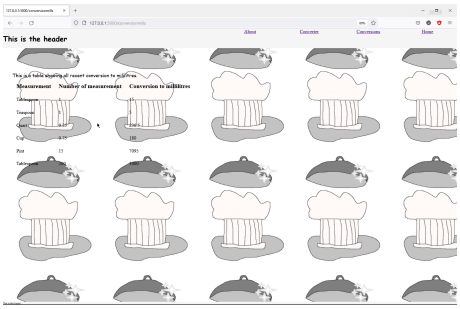
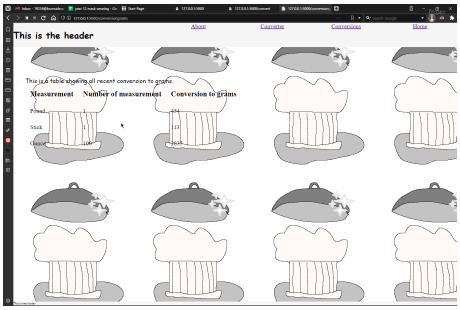
Test	Method	Result	Pass or Fail
Test improved styling on chrome	Debug program open in chrome and make sure that new styling and templates is rendered on home page.		Pass on chrome as I could see that styling was applied and new template was rendered. No gaps between the header and footer. Background image was working as well.
Test improved styling on Firefox	Debug program open in Firefox and make sure that new styling and templates is rendered on home page.		Pass on Firefox as I could see the styling had been applied to the header, footer, and background with the new image. I could say it was successful on Firefox.
Test improved styling on chrome	Debug program open in chrome and make sure that new styling and templates is rendered on about page.		Pass on chrome as I could see new styling was applied to about/help page and that the new template was rendered. Other elements from home page also working correctly.
Test improved styling on Firefox	Debug program open in Firefox and make sure that new styling and templates is rendered on about page.		Pass on Firefox as I could see all past styling was working on pages and that it functioned correctly on the about/help page in Firefox.
Test improved styling on chrome	Debug program open in chrome and make sure that new styling on converter page.		Pass on chrome as new styling was applied to converter page. While old template was still working correctly and rendering.

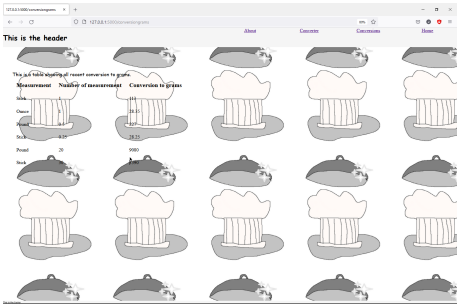
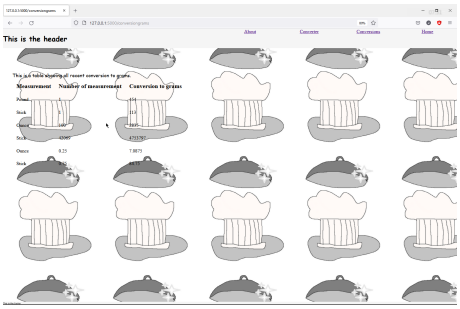
Test	Method	Result	Pass or Fail
Test improved styling on Firefox	Debug program open in Firefox and make sure that new styling on converter page.		Pass on Firefox new styling applied correctly and template was still unchanged on Firefox.
Test improved styling on chrome	Debug program open in chrome and make sure that new styling on conversions page.		Pass on Chrome new styling has been applied to conversions page. Header and footer are fixed to top and bottom of pages. No longer a gap and background image working correctly for body. Padding and font has also been applied to text correctly.
Test improved styling on Firefox	Debug program open in Firefox and make sure that new styling on conversions page.		Pass on Firefox new styled been applied to conversion page. All as expected and nothing is aesthetically bad to look at.
Test improved styling on chrome	Debug program open in chrome and make sure that new styling on conversions to mills page.		Pass on Chrome styling been applied to table data and headers. More spacing and slightly more readable than it was before. Text is slightly hard to distinguish from background. Header, footer and background all functioning properly.
Test improved styling on Firefox	Debug program open in Firefox and make sure that new styling on conversion to mills page.		Pass on Firefox styling been applied correctly to data. Distinguishable difference compared to previous unstyled table. Improvements in readability still possible but will address at a later date.
Test improved styling on chrome	Debug program open in chrome and make sure that new styling on conversion to grams page.		Pass on Chrome same as for conversions for mills. Styling all been applied well. Noticeable difference compared to conversion to grams in version 2. Still improvements to be made in how the data is presented. Works for components that has been tested.

Test	Method	Result	Pass or Fail
Test improved styling on Firefox	Debug program open in Firefox and make sure that new styling on conversion to grams page.		Pass on Firefox all parts of styling working correctly. Still improvements for how the data is present as hard to read. Needs improvement but has passed the test as styling as been applied correctly. As well header and footer like nice with the table.
See that dropdown hidden if none selected	Debug program and see that with new converter functions only radio buttons appear on converter page in Chrome.		Pass on Chrome as when no selections for the radio button is made no other elements appear. This functions as intended as a user must decide if they want to convert grams or mills. Also stopping errors of incorrectly submitting data. Passes test on chrome.
See that dropdown hidden if none selected	Debug program and see that with new converter functions only radio buttons appear on converter page in Firefox.		Pass on Firefox. No selection means no other elements from the convert template are displayed as was intended. Know that it is now working and functional.
Show dropdown millilitres Chrome	Debug program open in chrome and navigate to converter page. Select millilitres radio button and see that dropdown appear when selected.		Pass on Chrome. When millilitres radio button selected on chrome dropdown for measurement in milliliters is displayed. Means that function is working correctly and functioning as intended.
Show dropdown millilitres Firefox	Debug program open in Firefox and navigate to converter page. Select millilitres radio button and see that dropdown appear when selected.		Pass on Firefox. When millilitres is selected shows dropdown for measurements. Function is working so I can confirm that the component is also functional. Meaning that the program is more function than it was before.
Show dropdown grams Chrome	Debug program open in chrome and navigate to converter page. Select grams radio button and see that dropdown appear when selected.		Pass on Chrome. When grams radio button is selected on Chrome corresponding dropdown list for measurement in grams is displayed. Thus meaning that the function is working correctly and that the test was successful.

Test	Method	Result	Pass or Fail
Show dropdown grams Firefox	Debug program open in Firefox and navigate to converter page. Select grams radio button and see that dropdown appear when selected.		Pass on Firefox. When grams radio is selected on Firefox. The correct dropdown list for measurements is displayed to the user. Meaning it has done what I achieved to set out.
Show input millilitres Chrome	Debug program open in Chrome. Navigate to converter page. Select millilitres radio button and unit from dropdown. Then see if number input shows up.		Pass on Chrome. When measurement is selected from dropdown list for measurements to convert to millilitres. Number input is displayed and shown. Therefore working successful. Could improve by taking value and adding that to number input instead of just generic message.
Show input millilitres Firefox	Debug program open in Firefox. Navigate to converter page. Select millilitres radio button and unit from dropdown. Then see if number input shows up		Pass Firefox. When measurement from list in millilitres is selected. Number input is displayed and functions correctly. Therefore passes test as functions correctly.
Show input grams Chrome	Debug program open in Chrome. Navigate to converter page. Select grams radio button and unit from dropdown. Then see if number input shows up		Pass on Chrome. When measurement in dropdown for grams is selected number input is displayed afterwards. Working correctly meaning that it functions alright.
Show input grams Firefox	Debug program open in Firefox. Navigate to converter page. Select grams radio button and unit from dropdown. Then see if number input shows up		Pass on Firefox. When measurements is selected from dropdown input is displayed afterwards. Meaning that the function works correctly so it functions as according.
See submit is hidden until input is entered for millilitres	Debug program open in Chrome. Select radio button millilitres. Select unit from dropdown. Enter number into input and see if submit appears.		Pass on Chrome. When number inputted submit is shown afterwards. Meaning function works and that is functional so passes the test.

Test	Method	Result	Pass or Fail
See submit is hidden until input is entered for millilitres	Debug program open in Firefox. Select radio button millilitres. Select unit from dropdown. Enter number into input and see if submit appears.		Pass on Firefox. When number inputted into number of measurements JavaScript function works properly and displays the submit. As a result component works so passes the test.
See submit is hidden until input is entered for grams	Debug program open in Chrome. Select radio button grams. Select unit from dropdown. Enter number into input and see if submit appears.		Pass on Chrome. When number inputted for number of measurements to convert to grams submit is displayed afterwards. Thus meaning that it functions alright so the component passes the test.
See submit is hidden until input is entered for grams	Debug program open in Firefox. Select radio button grams. Select unit from dropdown. Enter number into input and see if submit appears.		Pass on Firefox. When number inputted shows submit button afterwards. Showing that function works. So the component is working correctly.
See that post correctly and print expected value from form in Chrome	Debug program. Open in Chrome. Go through form. Submit. See redirected to conversions page and message printed correctly.		Pass on Chrome. Once converter page is submitted user is redirected to conversions page with the test message. This showed me that it posted correctly. So the component is functioning and passes the testing.
See that post correctly and print expected value from form in Firefox	Debug program. Open in Chrome. Go through form. Submit. See redirected to conversions page and message printed correctly.		Pass on Firefox. Once submitted shows test outcome to see if posting to conversions. Working so testing is complete.
See that post correctly for millilitres in Chrome	Debug program navigate to converter. Select milters and then cups from dropdown. Submit then see that posted to database.		Pass on Chrome. Test data from the user has been correctly entered into the database and has been converted correctly. Functioning and now having more usability. It passes testing.

Test	Method	Result	Pass or Fail
See that post correctly for millilitres in Firefox	Debug program navigate to converter. Select milters and then cups from dropdown. Submit then see that posted to database.		Pass on Firefox. Test data from converter has been inserted into the database and been queried and displayed to user in table within page. Working so I know that it is functional.
See that post correctly for grams in Chrome	Debug program navigate to converter. Select grams and then ounces dropdown. Enter one then submit form. See that data has been uploaded to database.		Pass on Chrome. Data has been successfully inserted into the database and has been queried from grams table and presented in table on conversion to grams page. Meaning that it has work so the competent has succeed and testing is complete.
See that post correctly for grams in Firefox	Debug program navigate to converter. Select grams and then ounces dropdown. Enter one then submit form. See that data has been uploaded to database.		Pass on Firefox. Data has been inserted into grams within Conversions database successfully. Has been queried and looped through in table as well so functions as intended. Thus meaning that it has passed testing.
See that post correctly for millilitres high value in Chrome	Debug program navigate to converter. Select milters and then cups from dropdown. Submit then see that posted to database.		Pass on Chrome. Entries that test whether the program is able to convert a very large conversion to millilitres have succeeded in Chrome. Thus meaning that it has been inserted into millilitres and converted properly. So it has passed testing.
See that post correctly for millilitres high value in Firefox	Debug program navigate to converter. Select milters and then cups from dropdown. Submit then see that posted to database.		Pass on Firefox. Stress test on whether program is able to convert a very large entry have worked correctly so I can be certain that it will give an accurate conversion for large conversion to millilitres. Also that it is inserted into millilitres table correctly; and queried to be present correctly as well.
See that post correctly for grams high value in Chrome	Debug program navigate to converter. Select grams and then ounces dropdown. Enter one then submit form. See that data has been uploaded to database.		Pass on Chrome. Same as testing for maximum conversion for millilitres but for grams instead. Has successfully converted measurements, inserted into the database, and being queried and presented in table so is all working correctly.

Test	Method	Result	Pass or Fail
See that post correctly for grams high value in Firefox	Debug program navigate to converter. Select grams and then ounces dropdown. Enter one then submit form. See that data has been uploaded to database.		Pass on Firefox. Again test maximum conversion this time for grams instead of millilitres. All working fine and nothing is not functioning correctly. Passed the testing as a result.
See that post correctly for millilitres low value in Chrome	Debug program navigate to converter. Select milters and then cups from dropdown. Submit then see that posted to database.		Pass on Chrome. Test for the program to see if it is able to convert very small measurements. This testing is for millilitres on chrome. Since has been successful now that it has functioned correctly.
See that post correctly for millilitres low value in Firefox	Debug program navigate to converter. Select milters and then cups from dropdown. Submit then see that posted to database.		Pass on Firefox. Program has successfully converted a very small entry for millilitres. And has inserted into millilitres table as well as querying data successfully.
See that post correctly for grams low value in Chrome	Debug program navigate to converter. Select grams and then ounces dropdown. Enter one then submit form. See that data has been uploaded to database.		Pass on Chrome. Testing to see if program is able to convert very small entry to grams. Has worked as has been converted successfully and works as according to plan.
See that post correctly for grams low value in Firefox	Debug program navigate to converter. Select grams and then ounces dropdown. Enter one then submit form. See that data has been uploaded to database.		Pass on Firefox. Testing for minimum grams entry on firefox has worked correctly so I am certain that this compontent has worked and that it is now successful.

Improvements

- Change name of anchor tags to be more appealing
- Make data displayed in table more intuitive to understand
- Add date so user can know when each conversion was made.

Sprint 4

Outline

- Change name of anchor tags to be more appealing
- Change so first entry into table is orange
- Add blurb to about page describing feature of the website
- Re-order so table prints most recent to oldest conversion.
- Change so columns of table are better looking
- Reformat text on home page to make easier to read and more appealing
- Post recent conversion from user into history page
- Rework so queries data from newest to oldest
- Fix so script shows the value that user selected from drop down in number input
- Hide submit until user has entered value into input
- Customize radio button with CSS

Process

Replace with text function

Reason for design: This built upon the pervious design Hide input based on dropdown but instead of displaying a generic message I wanted to change what would be displayed based upon what the user selected. The function would act the same as before but would take the selected value from the dropdown and use the innerHTML to replace the space with the measurement e.g. user selects cups and function will replace display message which will ask how many cups user wants to convert. Thing that is replaced is in span so it will change based on selection.

Code for template

```
<script>
function hideinput(answer) {
        document.getElementById('quantity').innerHTML=answer.value //Gets value of measurement that user has selected.//
      document.getElementById("NumberMeasurements").style.display='block'; //Display number input after user has entered
the measurement they want to convert.//
        return;
    }
</script>
```

Adding Date to table

Reason for design: I needed to add a date function to the program so that any new entries into the database would store the date which they occurred on. This would improve usability as users could see exactly which day they converted there measurements on. I achieved this through recreating the old table to include a date column as the first row. I also rearrange the order of the columns to be more initiative as well. This would mean that it would be easier for users to understand what the website converted. I also wanted to test to make sure that date time was working correctly

Code for tables

```
import datetime
print(datetime.now())
try:
    #Create table to store measurement entered in mills
    c.execute("CREATE TABLE millilitres(date TEXT,quantity INTERGER, unit TEXT,conversion INTERGER)")
    #Create table to store measurement entered in grams
    c.execute("CREATE TABLE grams(date TEXT,quantity INTERGER, unit TEXT,conversion INTERGER)")
    print("Database Setup")
    connect.close()
    connect.commit()

except:
    pass

finally:
    print("Database connected")



#Get data from tables in database

#Query data from grams table in conversions Database

try:
    c.execute("SELECT * FROM grams")
    DisplaytoTableGrams=tuple(c.fetchall())
    print("Data queried from Table grams")
except:
    pass
```

Reason for design:I also needed to incorporate this into the conversion page so once new conversion where made by the user the date would be added to the query and inserted into the database. Else there would be no way to insert the date into the database. I added this to both the milliliters and grams direct on the conversions page.

Code for conversions

```
date = datetime.today().strftime('%Y-%m-%d')
            #Print conversion to see if it is correct.
  add_to_tales(date,number_measurements,measurement,conversion)
```

Order by descending order by date.

Reason for design:I needed to edit the tuples for headings as since I changed the columns in the table I would also need to change what the program would use as headings in order to line up with the new changes. This was easy as I only needed to change the order of the tuple which made it a very simple and easy process. This would also mean in the future it would be easy to change it again if it was required.

```
heading_table_grams=('Date','Number of measurement','Measurement','Conversion to grams')

heading_table_mills=('Date','Number of measurement','Measurement','Conversion to millilitres')
```

Reason for design:After the date function was successfully added to the conversions page I also needed to now sort the queried date from decseding date order. This would mean more recent conversions would show up before older conversions. This was achieved from ORDER by date DESC which was a simple and easy solution which would order conversion by those which were most recent to least recent.

```
@app.route("/conversionmills")
def conversion_mills():
    connect=sqlite3.connect('Conversion') #Connect to database.
    c=connect.cursor()
    c.execute("SELECT * FROM millilitres ORDER by date DESC")
    #Gets all data crom millilitres table
    add_to_table_mills=tuple(c.fetchall())
    return render_template('conversion_mill.html',
    heading_table_mills=heading_table_mills,
    add_to_table_mills=add_to_table_mills)
        #code snippet not code.

@app.route("/conversiongrams")
def conversion_grams():
    connect=sqlite3.connect('Conversion')
    c=connect.cursor()
    c.execute("SELECT * FROM grams ORDER by date DESC")
    # Queries data from databse and sorts based on what the newest entry is.
    add_to_table_grams=tuple(c.fetchall())
    # Queired data fecthed and new list called add to table grams created.
    return
render_template('conversion_grams.html',heading_table_grams=heading_table_grams,add_to_table_grams=add_to_table_grams)
# Headings for loop and data for loop inserted into the table.

    ```

<div style="page-break-after: always;"></div>

Post to History page
Reason for design: I wanted to show the recent conversions the user has entered on the history page so that they wouldn't need to go through to the conversion to mills or grams page to see the final conversion. It would also make it easier for the user to convert another measurement as they would just have to click the anchor tag near the conversion rather than going to the nav bar again. This made it more usability as it made it quicker to user to convert more measurements if they desired.
```python
@app.route("/conversions/", methods=['POST','GET'])

def conversions():

    if request.method=='POST':

        millsorgrams=request.form['millilitres or Grams']

        if millsorgrams == 'millilitres':
            measurement_mills=request.form['unitmills']
            print(measurement_mills)
            number_measurements=float(request.form['numberofunits'])
            print(number_measurements)
            conversion_mills=
            float(conversion_to_mills_dict[measurement_mills])
            *float(number_measurements)
            #Code snippet not actual code. All one line in backend.
            print(conversion_mills)
            date = datetime.today().strftime('%Y-%m-%d')
            add_to_tale_mills=(date,number_measurements,
            measurement_mills,conversion_mills)
            #Code snippet not actual code. All one line in backend.
            print(add_to_tale_mills)
            connect=sqlite3.connect('Conversion')
            c=connect.cursor()
            c.execute("""INSERT INTO millilitres VALUES(?,?,?,?)
            """,add_to_tale_mills)
```

```
            #Code snippet not actual code. All one line in backend.
            connect.commit()
            connect.close()
            return render_template('conversions.html',x='millilitres',
            add_to_tale_mills=add_to_tale_mills)
            #Code snippet not actual code. All one line in backend.
        else:
            measurement_grams=request.form['unitgrams']
            print(measurement_grams)
            number_measurements=request.form['numberofunits']
            print(number_measurements)
            conversion_grams=
            float(conversion_to_grams_dict[measurement_grams])
            *float(number_measurements)
            #Code snippet not actual code. All one line in backend.
            print(conversion_grams)
            date = datetime.today().strftime('%Y-%m-%d')
            add_to_tale_grams=(date,number_measurements,
            measurement_grams,conversion_grams)
            #Code snippet not actual code. All one line in backend.
            print(add_to_tale_grams)
            connect=sqlite3.connect('Conversion')
            c=connect.cursor()
            c.execute("""INSERT INTO grams VALUES(?,?,?,?)""",
            add_to_tale_grams)
            connect.commit()
            connect.close()
            return render_template('conversions.html',x='grams',
            add_to_tale_grams=add_to_tale_grams)
            #Code snippet not actual code. All one line in backend.
    else:
        return render_template('conversions.html',)
```

Code for template

```
{% if x == "millilitres" %}
<p> {{add_to_tale_mills[1]}} {{add_to_tale_mills[2]}}s is {{add_to_tale_mills[3]}} millilitres.</p>
<p><a href="{{ url_for('converter')}}"> Convert another measurement</a></p>
{% elif x == "grams" %}
<p>{{add_to_tale_grams[1]}} {{add_to_tale_grams[2]}} {{add_to_tale_grams[3]}} grams.</p>
<p><a href="{{ url_for('converter')}}"> Convert another measurement</a></p>
{% else%}
<p> No units converted yet convert.</p>
{% endif%}
```

Error page

Reason for design:I created a unique error page for my website. I intially did this with a basic template to make sure it was working correctly. I had a basic HTML template which I could then change later to add more functionality.

```
@app.errorhandler(404)
def page_not_found(e):
    return render_template('404.html'), 404 # 404 error page for website.
```

This is the template for the error page

```
        <p>Oh no 404 error. </p>
```

Reason for design:I built upon the basic design on the first iteration and added more details to the page. I extended the 'base.html' as well as including a link back to home to make it easier to return home. As well I added a class to it and applied custom styling so that it would be more unique and noticeable.

Code for template

```
{% extends 'base.html'%}
{% block content%}
    <p class="error">
        Oh no 404 error. <br> <a href="{{url_for('home')}}">Take me back home</a> <!-- Error page for user in case they
make a mistake or enter a nonexisting page.
           Link back to home so they are able to go back to website
       -->
</p>
{%endblock%}
```

Code for styling

```
.error {
    text-align: center;
    background-color: white;
    width: 75%;
    height: 55%;
    margin: 125px;
    font-size: 125px;
    color: #FF8C00;
    padding: 10px;
}
```

Reformat home

Reason for design: I wanted to finally complete my home page to fit the final purpose that I wanted to have. There wouldn't be any major changes I would just rewrite the text in order to fit better with Measureverto.

```
{% extends 'base.html'%}
{%  block content%}
<body>
            <p> Welcome to Measureverto. It's a combination of measurement and converto with 'O' for some extra spice
italian action.</p>
            <p> This is a website used to convert cooking measurement into grams or mililitres</p>
            <p> I hope you enjoy :)</p>
</body>
{% endblock %}
```

Radio buttons CSS

Reason for design: I wanted to alter the styling for the radio buttons from the default and change it in order to fit with the other styling for the rest of my website. This would improve the aesthetics of the entire website as it would all have a more consistent look across all pages.

Code for styling

```
input[type='radio'] {
    transform:scale(2);
    margin: 15px;
    accent-color: #ff8000;
}

input[type='radio']::after {
    transform:scale(0.75);
}
```

Stylized dropdown list

Reason for design: I again wanted to change the dropdown list to be more in line with the rest of the website to have a more consistent aesthetics. This was harder however as I couldn't change the font of the dropdown list using normal CSS so I had to add a class to the dropdown in order to change the font to comic sans so that it could have a consistent look across the website.

Code for styling

```
option:nth-child(odd){
    font-weight: 'Comic Sans MS';
    background-color:rgba(206, 209, 209, 0.726)
}

.dropdown {
   font-family: 'Comic Sans MS';
    font-size: 25px; ;
}
```

Add highlight to text

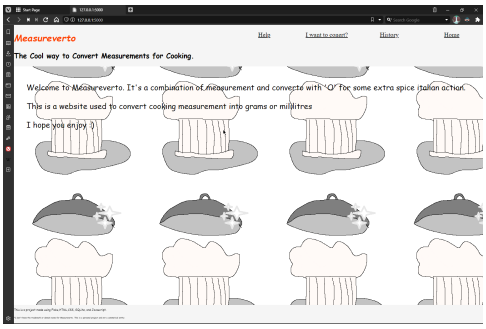
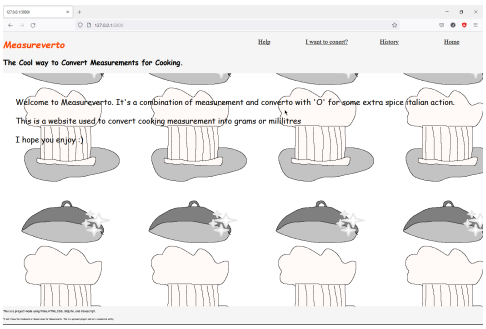
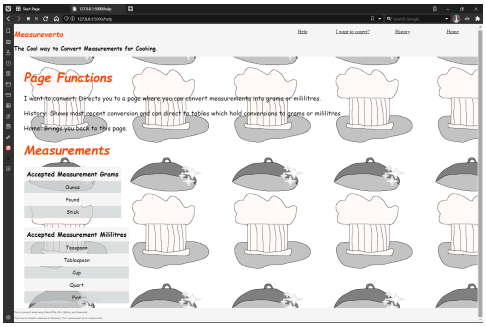
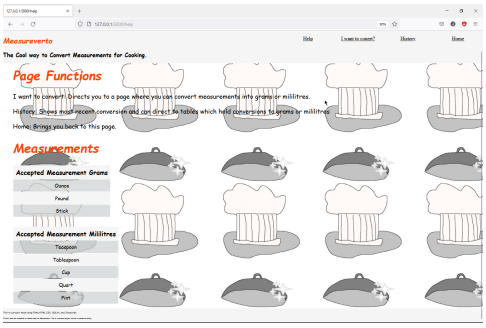
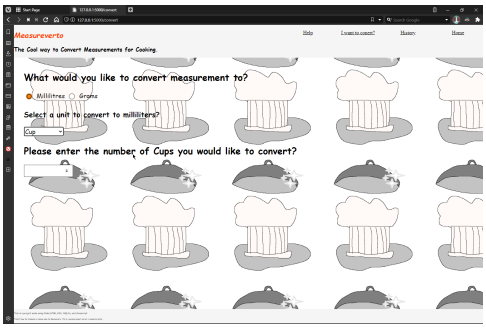
Reason for design: I wanted to add a highlight to the text should that it would be more readable against the background image. I did this to improve usability of the website by making the text on the home and about pages easier to read.

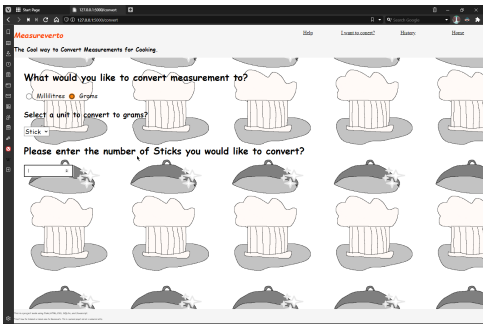
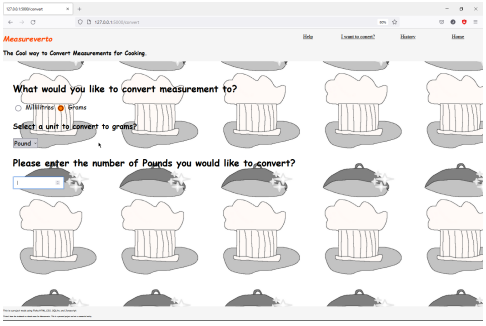
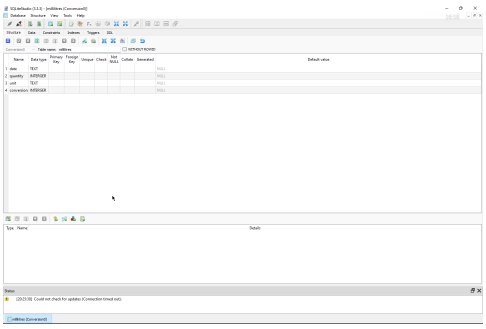
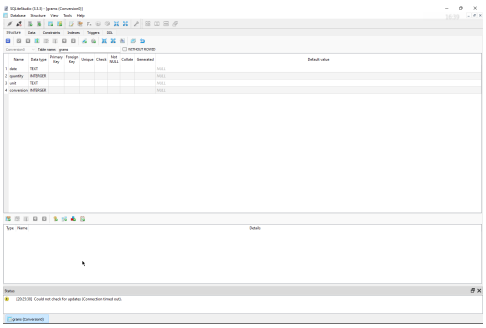
Code for template

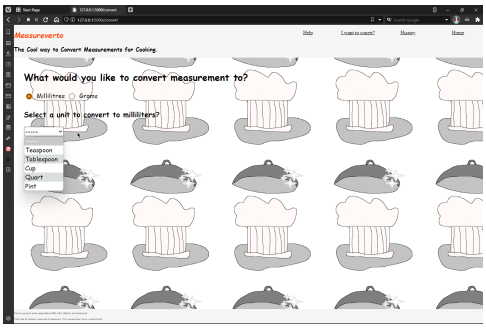
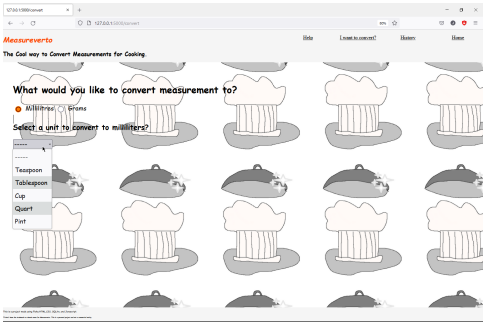
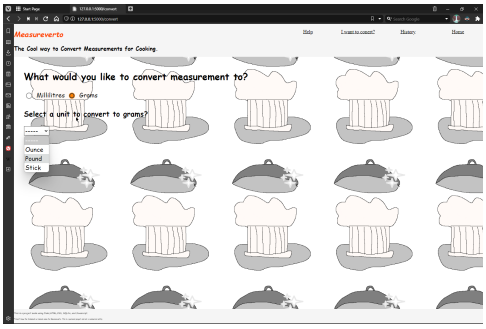
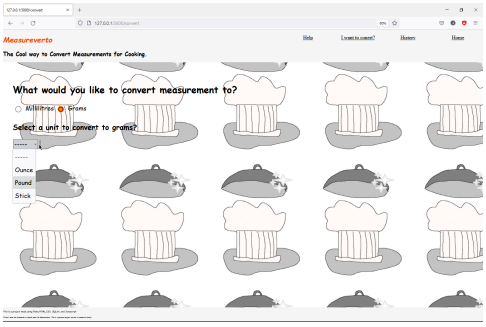
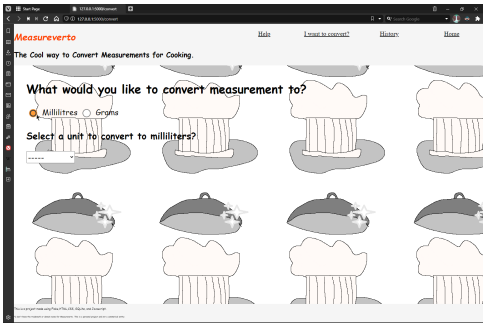
```
<p><span style="background-color: #e5e3e1ee">Welcome to Measureverto. It's a combination of measurement and converto with 'O' for some extra spice italian action.</span> </p>
<p><span style="background-color: #e5e3e1ee">This is a website used to convert cooking measurement into grams or mililitres </p></span>
<p><span style="background-color: #e5e3e1ee">I hope you enjoy :) </span></p>
```

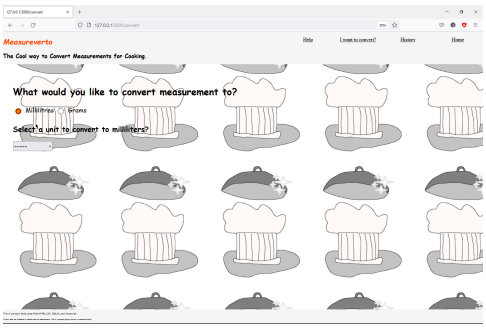
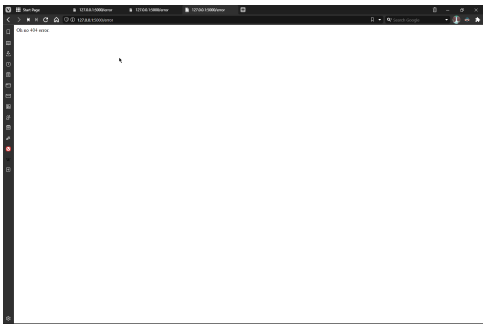
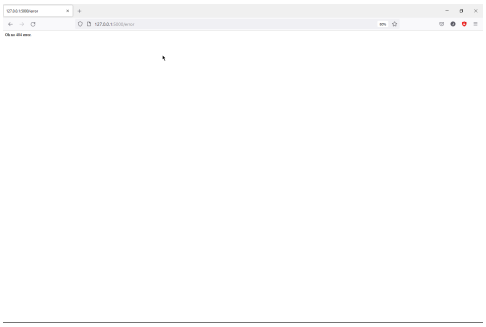
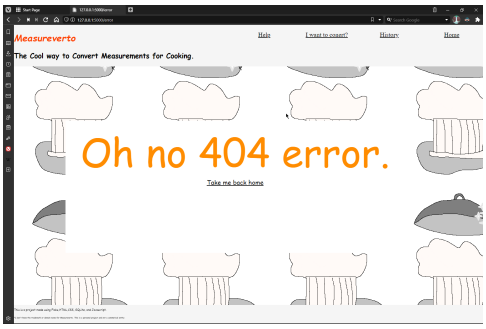
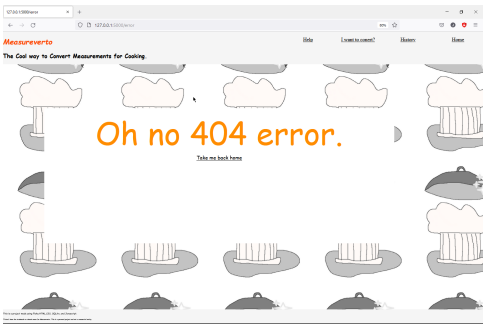
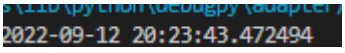
```
<p><span style="background-color: #e5e3e1ee">I want to convert: Directs you to a page where you can convert measurements into grams or mililitres.</span> </p>
<p><span style="background-color: #e5e3e1ee">History: Shows most recent conversion and can direct to tables which hold conversions to grams or mililitres</span></p>
<p><span style="background-color: #e5e3e1ee">Home: Brings you back to this page.</span></p>
```

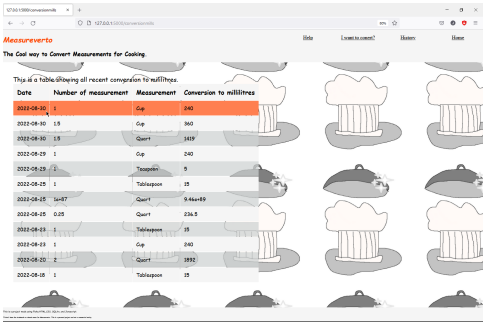
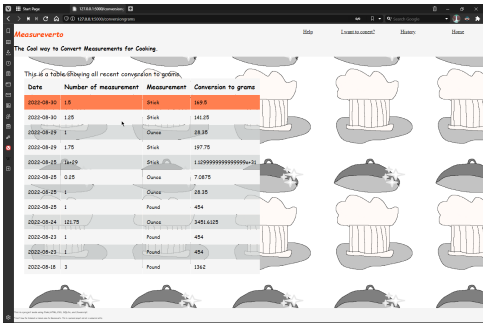
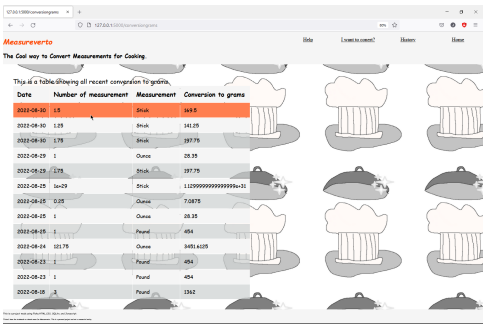
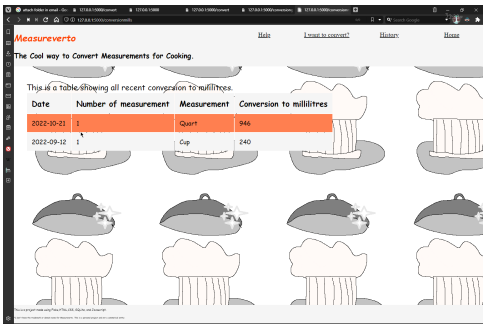
Testing Sprint 4

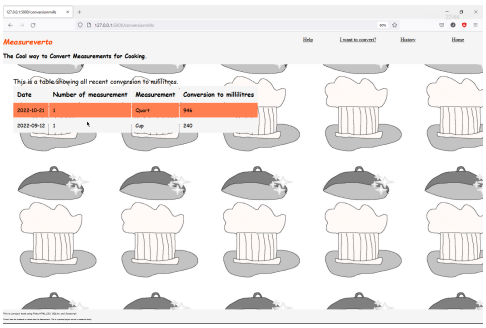
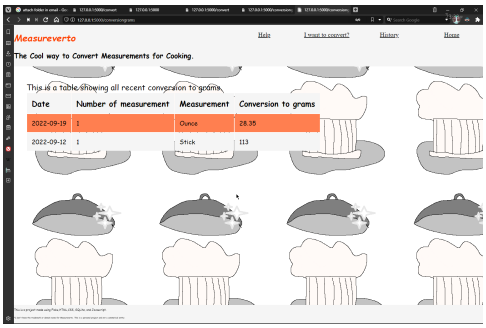
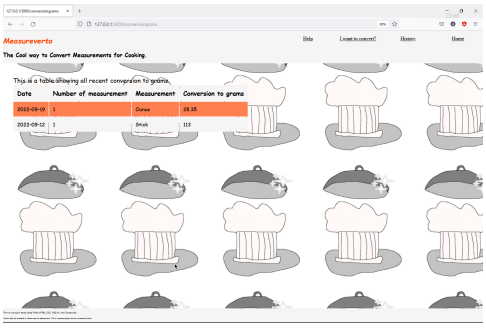
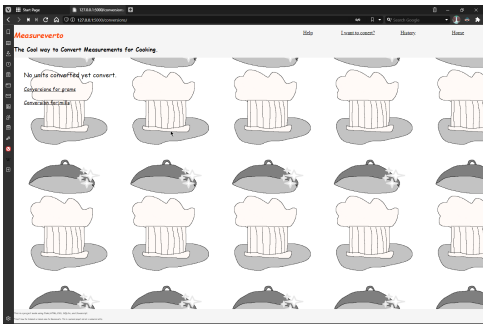
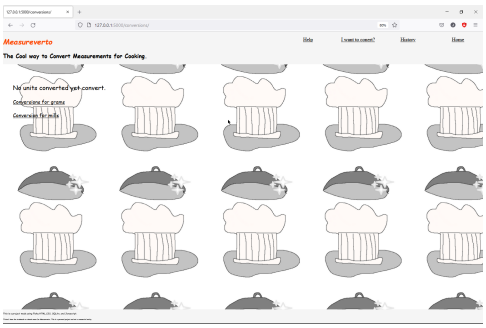
Test	Method	Result	Pass or Fail
See that home working	Debug program see that the home page has rendered correct template and styling has been correctly applied to the page on Chrome.		Pass on Chrome as New home page was rendered correctly again and final iteration is finally completed.
See that home working	Debug program see that the home page has rendered correct template and styling has been correctly applied to the page on Firefox.		Pass on Firefox. New Home page template was successfully rendered and styling been applied as expected.
See that about working	Debug program see that the home page has rendered correct template and styling has been correctly applied to the page on Chrome.		Pass on Chrome. New template for help rendered properly and functional. Therefore passes test on Chrome.
See that about working	Debug program see that the home page has rendered correct template and styling has been correctly applied to the page on Firefox.		Pass on Firefox. New template rendered properly again so better aesthetics. Also increase usability by providing clearer directions for how to use website. Therefore it passes testing.
See if text function works with conversion millilitres	Debug program navigate to "/convert" select radio button millilitres. Then select a unit from dropdown see that it is correctly displayed on Chrome		Pass on Chrome. When unit from dropdown is selected function runs and takes the value to display in the input. This means that it is has greater usability and is now more functional than it was before. As a result it passes testing.

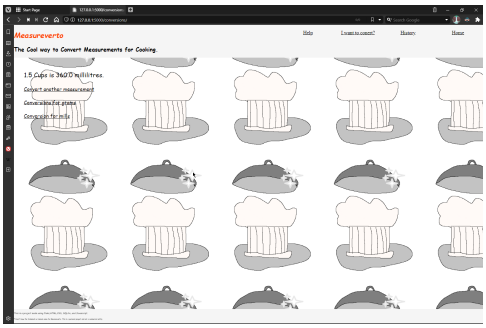
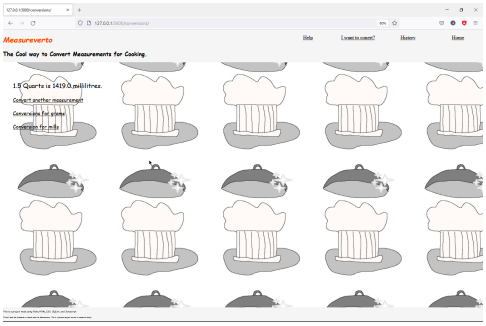
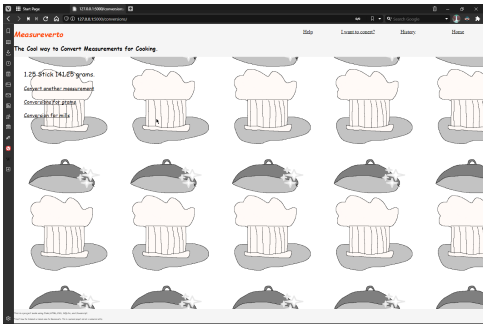
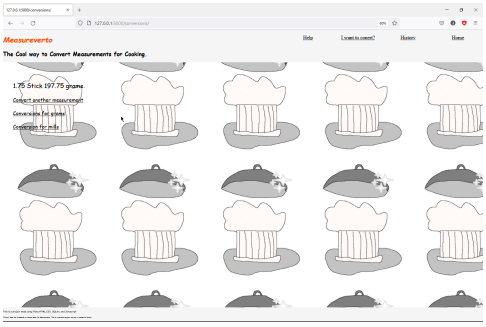
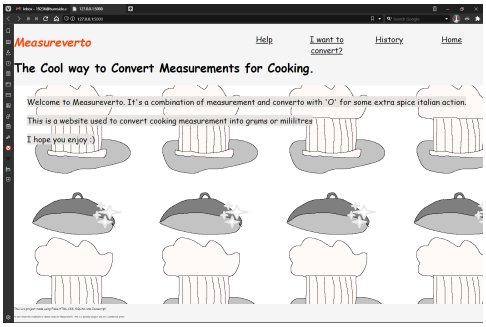
Test	Method	Result	Pass or Fail
See if text function works with conversion millilitres	Debug program navigate to "/convert" select radio button millilitres. Then select a unit from dropdown see that it is correctly displayed on Firefox		Pass on Firefox. When measurement selected from dropdown list it's value is displayed on the number input afterwards. This passes testing as a result.
See if text function works with conversions to grams	Debug program navigate to "/convert" select radio button grams. Then select a unit from dropdown see that it is correctly displayed on Chrome		Pass on Chrome. When measurement for grams selected it is displayed for the input afterwards. Meaning that the testing is successful.
See if text function works with conversions to grams	Debug program navigate to "/convert" select radio button grams. Then select a unit from dropdown see that it is correctly displayed on Firefox		Pass on Firefox. When unit selected from dropdown for measurements to convert to grams it's value is taken using a JavaScript function and then works accordingly. This means that testing is successful on Firefox.
See that correct table is created	Delete old database and tables. Run program and then use SQLITE studios to see the columns in the table		Pass. Successfully created a new millilitres table which has date column in addition to change order of columns.
See that correct table is created	Delete old database and tables. Run program and then use SQLITE studios to see the columns in the table		Pass. Successfully created a new grams table which has date column in addition to change order of columns.

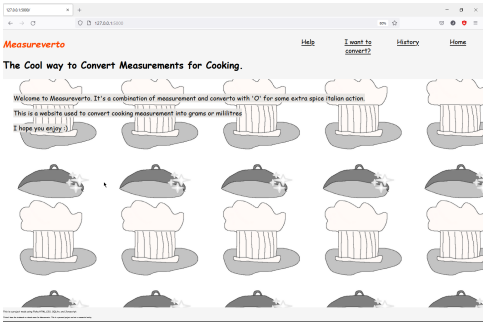
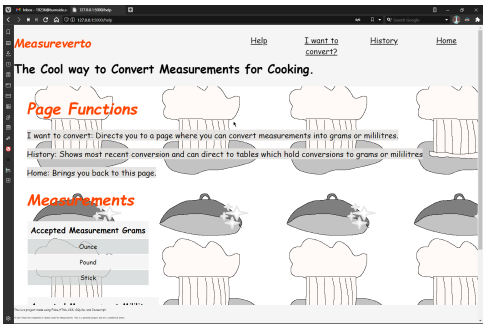
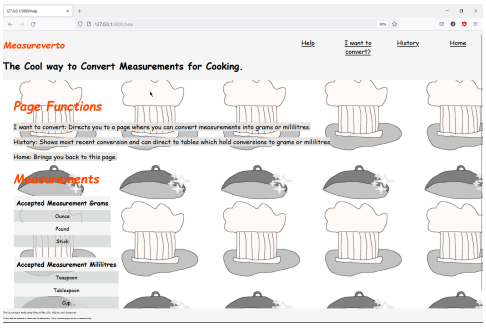
Test	Method	Result	Pass or Fail
See that pattern applied to dropdown list	Debug program navigate to converter page. Then use program and interact with dropdown to see that dropdown list has been applied on Chrome.		Pass on Chrome. Styling has been applied to dropdown list which now has a zebra stripe pattern instead of the default white.
See that pattern applied to dropdown list	Debug program navigate to converter page. Then use program and interact with dropdown to see that dropdown list has been applied on Firefox.		Pass on Firefox. Styling been applied and now has zebra stripe pattern with different shades of grey. Therefore passes testing.
See that pattern applied to dropdown list	Debug program navigate to converter page. Then use program and interact with dropdown to see that dropdown list has been applied on Chrome.		Pass on Chrome. Pattern has been applied to dropdown for measurements for grams in chrome. Therefore I know it is working so it passes the test.
See that pattern applied to dropdown list	Debug program navigate to converter page. Then use program and interact with dropdown to see that dropdown list has been applied on Firefox.		Pass on Firefox. Zebra stripe pattern has been applied to dropdown for measurement to grams in Firefox. Therefore I know it is functional so it passes testing.
See that radio button is different	Debug program navigate to converter page. Then see if radio button have had styling applied changing radius and colour once selected.		Pass on Chrome. Radio button accent colour has been changed to orange. As well radius and other details are now bigger so it passes testing. All styling correctly applied on Chrome.

Test	Method	Result	Pass or Fail
See that radio button is different	Debug program navigate to converter page. Then see if radio button have had styling applied changing radius and colour once selected.		Pass on Firefox. Radio button accent colour has been changed to orange. As well radius and other details are now bigger so it passes testing. All styling correctly applied on Chrome.
See that basic page without styling functions	Debug program enter invalid URL into URL bar e.g. "/error" Since this is in a invalid address will return error page.		Pass on Chrome. When a non-existent page entered into URL bar correctly redirected to error page on Chrome. Passes testing as is functional.
See that basic page without styling functions	Debug program enter invalid URL into URL bar e.g. "/error" Since this is in a invalid address will return error page.		Pass on Firefox. When non-existent page entered into URL bar correctly redirected to error page on Firefox. Passes testing as is functional.
See that error page working with styling of class	Debug program enter invalid URL into URL bar e.g. "/error" Since this is in a invalid address will return error page.		Pass on Chrome. Error page is correctly redirected as well as styling been applied as expected to template on Chrome.
See that error page working with styling of class	Debug program enter invalid URL into URL bar e.g. "/error" Since this is in a invalid address will return error page.		Pass on Firefox. Error page is correctly redirected as well as styling been applied as expected to template on Firefox.
Import datetime and see working	import datetime then print current date and time in format for program		Pass as printed current date at time of testing when program was debugged.

Test	Method	Result	Pass or Fail
See that mills ordered by descending date order	Debug program run Navigate to conversion to millilitres page table		Pass on Chrome. Data queried correctly from millilitres table. Date is displayed in correct column and is functional.
See that mills ordered by descending date order	Debug program run Navigate to conversion to millilitres' page table		Pass on Firefox. Data queried correctly from millilitres table. Date displayed in Firefox.
See that grams ordered by descending date order	Debug program run Navigate to conversion to grams page table		Pass on Chrome. Data queried correctly from grams table. Date displayed in Firefox.
See that grams ordered by descending date order	Debug program run Navigate to conversion to grams page table		Pass on Firefox. Data queried correctly from grams table. Date displayed in Firefox.
Edit data and see if sorts newer date first in table to millilitres	Open SQLite3 database change date for entry for quart to newer entry than cup within millilitres table. See that Quart displayed before Cup in table.		Pass on Chrome. When date is changed to newer date in millilitres table, it's displayed before older date in conversion to millilitres page. Therefore it is working correctly on Chrome.

Test	Method	Result	Pass or Fail
Edit data and see if sorts newer date first in table to millilitres	Open SQLite3 database change date for entry for quart to newer entry than cup. See that Quart displayed before Cup in table.		Pass on Firefox. When date is changed to newer date in millilitres table, it's displayed before older date in conversion to millilitres page. Therefore it is working correctly on Firefox.
Edit data and see if sorts newer date first in table to grams	Open SQLite3 database change date for entry grams for stick to newer entry than ounce. See that stick is displayed before ounce in table within conversion to grams page.		Pass on Chrome. When date is changed to newer date in grams table, it's displayed before older date in conversion to grams page. Therefore it is working correctly on Firefox.
Edit data and see if sorts newer date first in table to grams	Open SQLite3 database change date for entry grams for stick to newer entry than ounce. See that stick is displayed before ounce in table within conversion to grams page.		Pass on Firefox. When date is changed to newer date in grams table, it's displayed before older date in conversion to grams page. Therefore it is working correctly on Firefox.
See that Default message display posting to conversion	Debug program navigate to conversion page on chrome. Go to conversions page and see that default message is displayed.		Pass on Chrome. When Conversions page accessed normally default message appear as since no conversions have been made. Works on Chrome so passes testing.
See that Default message display posting to conversion	Debug program navigate to conversion page on Firefox. Go to conversions page and see that default message is displayed.		Pass on Firefox. When Conversions page accessed normally default message appear as since no conversions have been made. Works on Firefox so passes testing.

Test	Method	Result	Pass or Fail
See that post data correctly millilitres	Debug program navigate to converter page. Enter sample data into form submit.		Pass on Chrome. Data entered in converted for millilitres test has been successfully displayed on the conversions page once it has been submitted on Chrome.
See that post data correctly millilitres	Debug program navigate to converter page. Enter sample data into form submit.		Pass on Firefox. Data entered in converted for millilitres test has been successfully displayed on the conversions page once it has been submitted on
See that post data correctly grams	Debug program navigate to converter page. Enter sample data into form submit.		Pass on Chrome. Data entered in converted for grams test has been successfully displayed on the conversions page once it has been submitted on
See that post data correctly grams	Debug program navigate to converter page. Enter sample data into form submit.		Pass on Firefox. Data entered in converted for grams test has been successfully displayed on the conversions page once it has been submitted on
See that highlighted text working	Debug program open to home page and see that text has high light applied on.		Pass on Chrome. Highlight has been applied to text on home page for Chrome.

Test	Method	Result	Pass or Fail
See that highlighted text working	Debug program open to home page and see that text has high light applied on.		Pass on Firefox. Highlight has been applied to text on home page for Firefox.
See that highlighted text working	Debug program open to about page and see that text has high light applied on.		Pass on Chrome. Highlight has been applied to text on about page for Chrome.
See that highlighted text working	Debug program open to about page and see that text has high light applied on.		Pass on Firefox. Highlight has been applied to text on about page for Firefox.