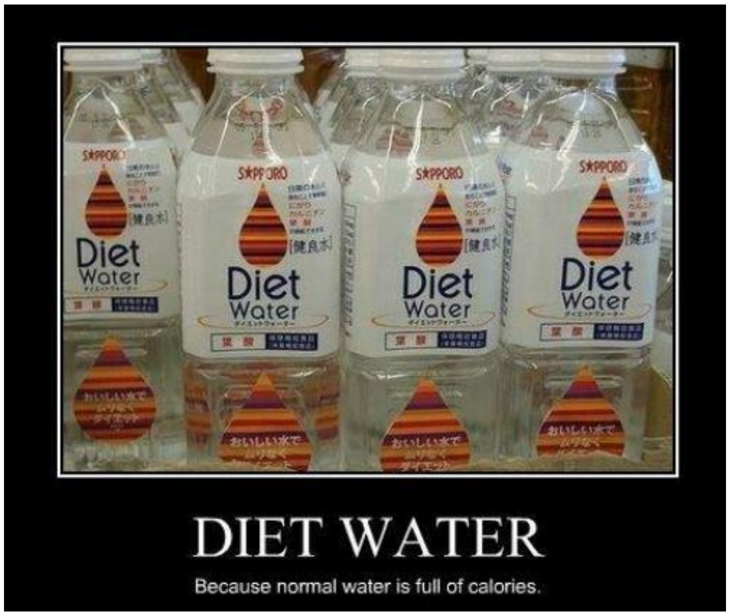
LC101: Class 2.12

Unit 2 Final Review

Congratulations! You've been selected to be the lead developer at Premier Water Solutions, Inc. This is a start-up company specializing in imported and overpriced bottled water. Apparently, fancy water is all the rage these days, and the CEO has asked you to implement an online store to increase profitability and make it easier for others to experience the wonders of modern, twenty-first century hydration.

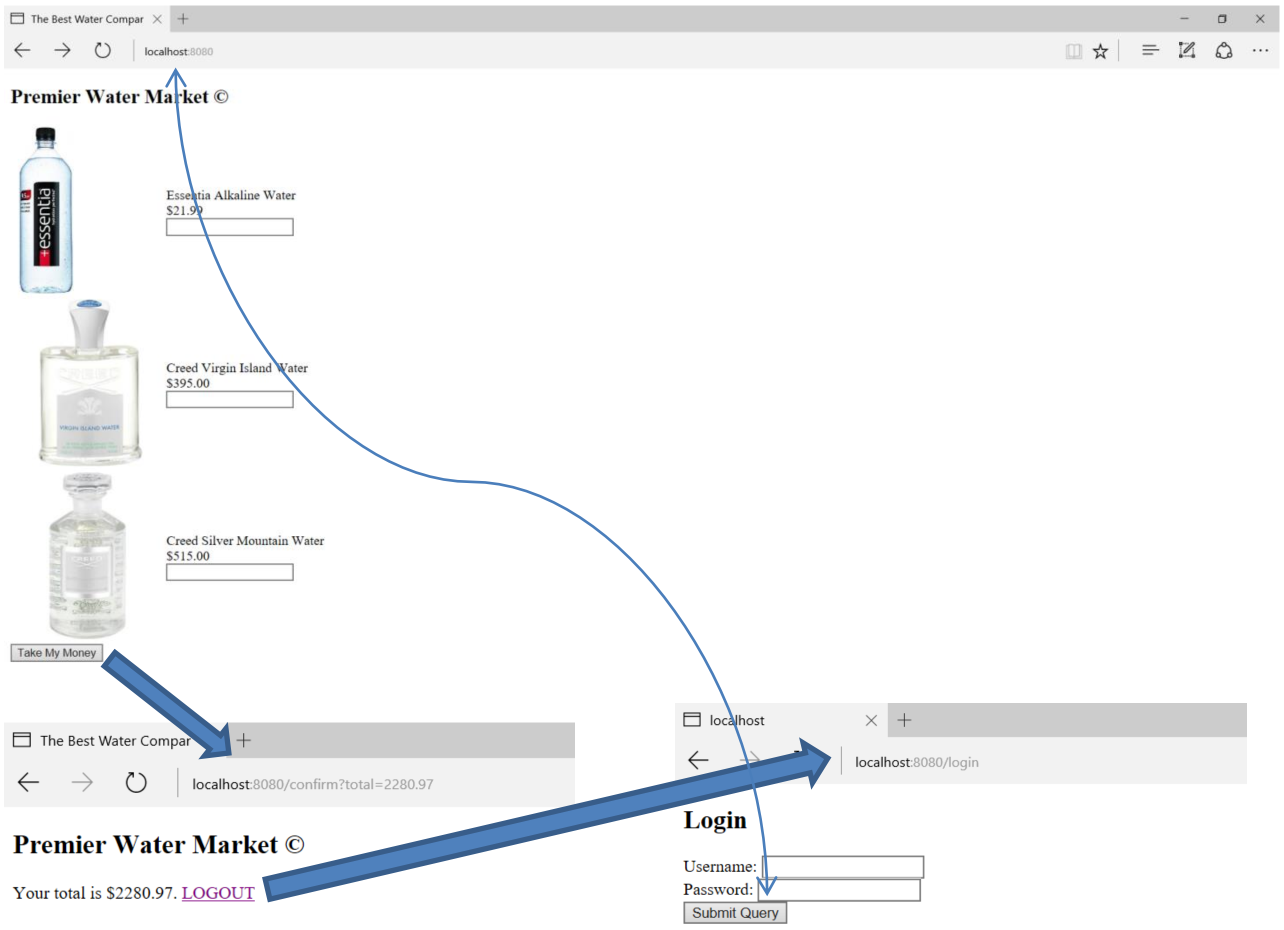
While the marketing team is obviously working round-the-clock . . .



. . . you've been given your own set of tasks and a timeline of ~~1 week~~ 1 hour to complete the project (turns out the remaining developer just quit). The requirements are as follows:

1.) Try to replicate the wireframes and mockups shown on the next page. The UI/UX team (i.e. the CEO) has great artistic vision, but will allow for *some* alterations since we're on a tight schedule.
2.) Miraculously, the previous team left you bits of functioning code. The user authentication and login features are working, so no need to look at hashutils.py! Clearly, nothing can go wrong.
3.) If a user doesn't enter a quantity, assume that it's zero. Once the user submits the form, calculate and display the total that has been charged.
4.) The confirmation page has one button that allows the user to logout and redirect to the login page. Since there's very limited stock, bad logic tells us the lack of a reorder button will stop users from buying too much. Of course, there are plenty of loopholes.
5.) We could just conduct a proper inventory and limit quantities, but since everyone's lazy, the higher-ups have instructed you to simply pretend we have an infinite supply of water. In fact, no additional data validation is necessary since we'll just assume our customer base is composed entirely of competent, omniscient, and non-malicious people. Problem solved!
6.) The database is pre-populated. Don't display products under $10, and order by **ascending** price.

The diagram:



The Database Schema:

Autogenerated ID	URL address to thumbnail photo	Product name	Price in USD

Since management neglected to pay the utilities last month, there’s neither access to electricity nor the internet. Fortunately, you may supply your own candles, quill pens, and typewriters (non-reimbursable expenses!) and code by hand. Don’t worry; we’ve saved you the pain of having to translate the code to machine language (individual 1’s and 0’s) by hiring a group of monkeys to perform the billions of operations necessary. You’ve also been supplied with a few reference materials, courtesy of GAE.

main.py

```
import webapp2
import os, jinja2
import hashutils
from google.appengine.ext import db
from models import Inventory, User

template_dir = os.path.join(os.path.dirname(__file__), 'templates')
jinja_env = jinja2.Environment(loader = jinja2.FileSystemLoader(template_dir), autoescape = True)

def user_required(handler):
    def check_login(self, *a, **kw):
        cookie_hash = self.request.cookies.get('user')
        if cookie_hash and hashutils.check_secure_val(cookie_hash):
            return handler(self, *a, **kw)
        else:
            self.redirect('/login')
    return check_login

class Handler(webapp2.RequestHandler):
    def write(self, *a, **kw):
        self.response.write(*a, **kw)
    def render_str(self, template, **params):
        t = jinja_env.get_template(template)
        return t.render(params)
    def render(self, template, **kw):
        self.write(self.render_str(template, **kw))

class LoginHandler(Handler):
    def get(self):
        self.render("login.html", error ="")
    def post(self):
        username = self.request.get("username")
        password = self.request.get("password")
        user = db.GqlQuery(" SELECT * FROM User WHERE username = '%s' " % username).get()
        if user and hashutils.is_valid_pwd(username, password, user.pwd_hash):
            cookie_hash = hashutils.make_secure_val(user.key().id())
            self.response.headers.add_header('Set-Cookie', 'user=%s; path=/' % cookie_hash)
            self.redirect('/')
        else:
            self.render("login.html", error = "Invalid credentials.")
```

```
class MainHandler(Handler):
    @user_required
    def get(self):





    def post(self):










class LogoutHandler(Handler):






class ConfimationHandler(Handler):






app = webapp2.WSGIApplication([
    ('/', MainHandler),






], debug=True)
```

models.py

```
from google.appengine.ext import db
class Inventory(db.Model):







class User(db.Model):
    username = db.StringProperty(required = True)
    pwd_hash = db.TextProperty(required = True)
```

login.html

```
<!DOCTYPE html>
<h2> Login </h2>
<form method="post">
    <label>
        Username:
        <input name="username">
    </label><br>
    <label>
        Password:
        <input name="password" type="password">
    </label><br>
    <div>{{error}}</div>
    <input type="submit">
</form>
```

base.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>The Best Water Company</title>
    </head>
    <body>
        <h2>Premier Water Market &copy;</h2>
        {% block content %}{% endblock %}
    </body>
</html>
```

confirm.html

main.html