# <u>User Story</u>

1. Authentication: User can signup/login/logout
2. Items for sale: User can view lists of items for sale and search by name, filter by price.
3. Shopping Cart: User can add items to shopping cart and the app remembers it next time you login. User can view all the items in their shopping cart. User can delete items in the shopping cart. Shopping cart uses an integer column to store "state".
4. Checkout: User can fill in form and submit billing info. After submitting billing info, items in the shopping cart will move to a different "state".

**Authentication**

Registration:

5. Create a signup page /signup
6. Add a url/controller/template /signup
7. /signup has a form, username, email, and password.
8. "Submit" button posts to /register
9. /register creates a new user

Login:

10. Create a login page /login
11. /login shows a form for username and password
12. "Submit" button posts to /login_user
13. /login_user uses the code below

Authenticate:

14. Create a new page that is only for logged in users. A members only page. Up to you what you want to show!
15. If the user is logged in, show the page.
16. If not, redirect the user to the login page

Logout:

17. Create a new url/controller for /logout
18. When /logout is called, redirect user to the home page

**Items**

19. Create a new Item Model with the following fields:

Name, Description, Price

20. Create several in the admin or shell
21. Create new routes and templates to show a listing of the items

/items ->shows all items

22. Create new route and template to show just one listing
23. Create more then 10 items

**Pagination**

24. Add pagination to the items listing page.

**Search**

25. Add search box to items listing page, search uses GET and query params to generate new page. The search query uses the name and description fields.

**Filter**

26. Allow the user to filter items by price. Use GET and query params. Filter by a range of prices (0-50, 50-100, 100+).

**Json API for Items**

- Add a format query param handler to /items where if the format equals json, then the response is in json

**Shopping cart/order**

- Create a new Model called Order (This is the shopping cart!)

An order belongs to a user, and has multiple items. A user can have many orders. An order has a status column, which is an integer field:

1 - In shopping cart

2 - Purchased

For any given user, you can only have one order with a status equal to 1.

When a user adds an item to the shopping cart, if there is no order with a status equal to 1, then create a new order for the user.

- Create a new route and view for /cart

/cart shows what items are in that users cart

- To show cart, you will need to query for the right order - match the user (request.user) and set a condition where status is equal to one.
- Allow user to delete items from the cart
- Shows the total price of all items
- Allows them to purchase items, purchasing takes the user to payment form at /payments
- Update the /item/ template to have a "purchase" button - when clicked, the item is added to the order, and the user is redirected to /cart

**Payment form**

- Create a new route and template for /payments

- Create a form that allows the user to enter billing info
- On submit, the order id status changes to purchased (2)