

# Web Development Project

**Project name: Sneakers**

**Project link:**

I want my project to be a website that gives information about shoes and the culture following it. My website will be curated for sneakerheads and people who want to get into the culture. My website has an informative purpose. The product name is 'Sneep'.

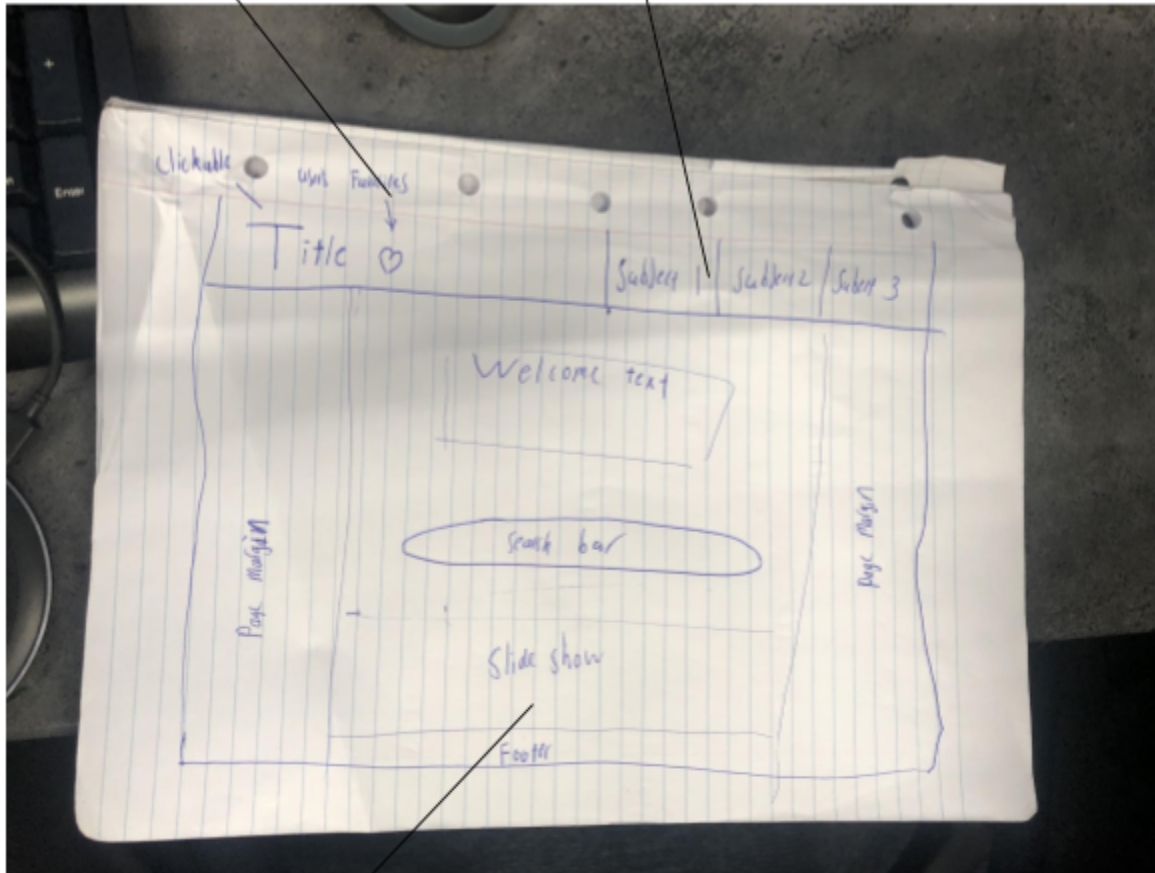
## **Planning**

**Features:**

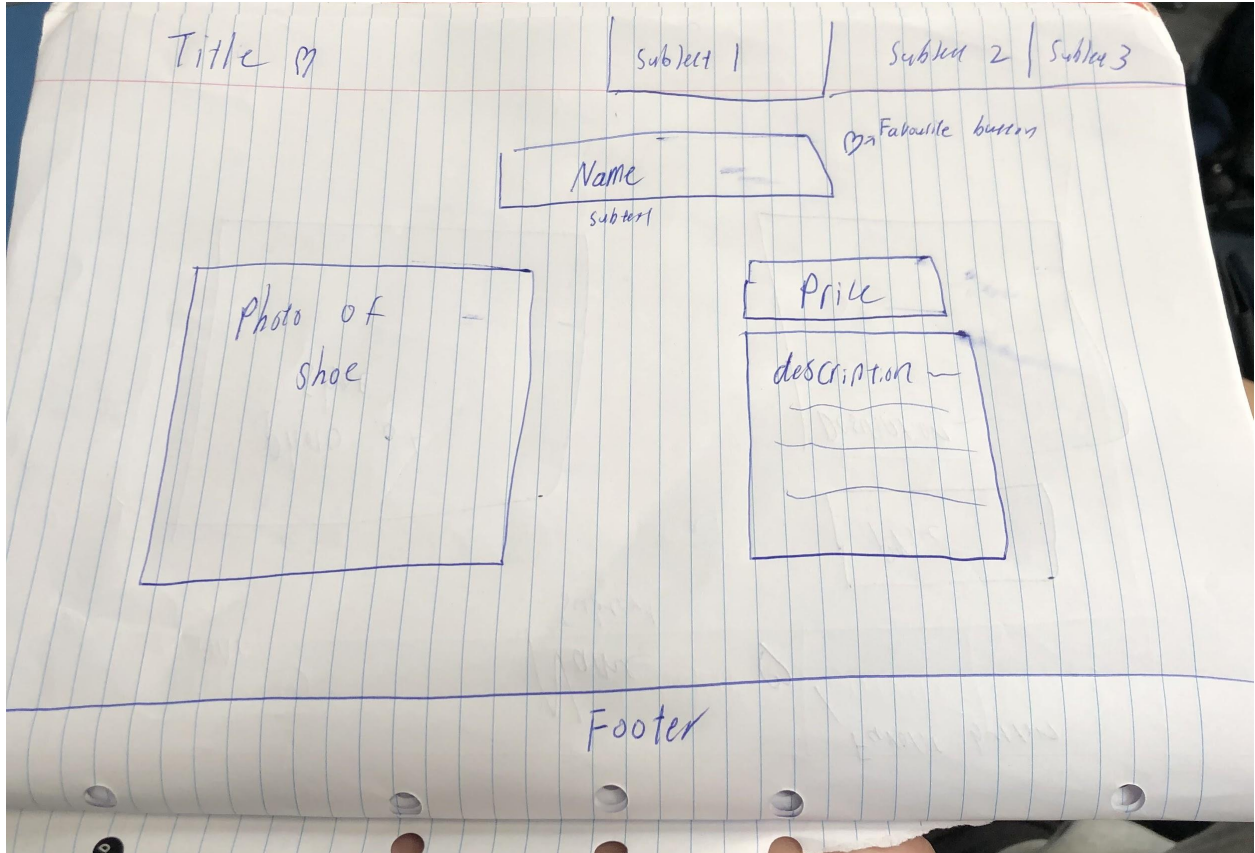
- **Price and rarity of each shoe**
- **Option for user to make an account and favorite**
- **Pictures of each shoe**
- **A Search bar**
- **Option to sort shoes**

User can favourite their most liked pairs Should only appear if logged in

These subjects will dropdown on hover allowing the user to navigate orderly



This slideshow will contain slides of popular sneakers which the user can click on to go to that sneaker's page



Color palette:

172A3A

Prussian Blue

565254

Davys Grey

FFFFFFA

Baby Powder

02111B

Rich Black FOGRA 29

000103

93C48B

Dark Sea Green

I chose these fonts because I wanted my website to have a “street” aesthetic to it. The simple minimalist and casual look of the website emphasizes this. I wanted this aesthetic as sneakers(the main topic) have a strong history with the “streets” and mid-class people.

I want my website to be simple to navigate through and user-friendly.

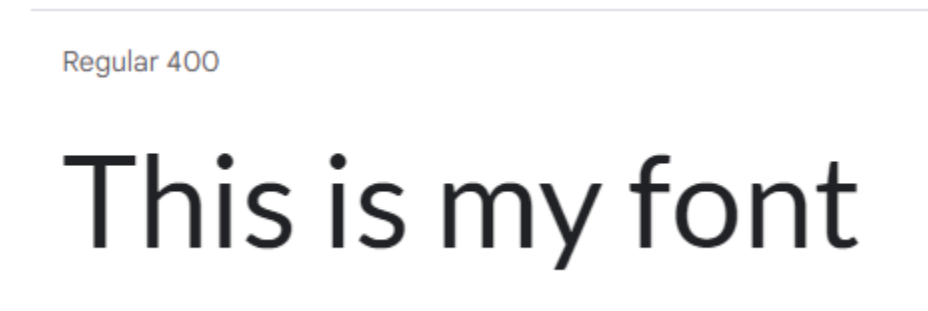
Font:

Shizuru(Title Font)



Font:

Lato(Body)



Nav buttons:

Caveat



```
<link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
<link
href="https://fonts.googleapis.com/css2?family=Caveat:wght@700&family=Lato&family=Shizuru&display=swap" rel="stylesheet">
```

```
font-family: 'Lato', sans-serif;
font-family: 'Shizuru', cursive;
font-family: 'Caveat', cursive;
```

## Function Signatures and routes

```
@app.route("/")
def home():
```

```
@app.route("/silhouette/<int:id>")
def silhouette(id):
```

```
@app.route("/shoe/<int:id>")
def shoe(id):
@app.route("/Silhouette/<int:id>")
def silhouette():
```

```
@app.route("/favourite/<int:id>", methods=["GET", "POST"])
@login_required
def favourite(id):
```

```
@app.route("/favourite/remove/<int:id>", methods=["GET", "POST"])
@login_required
def remove(id):
```

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

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

```
@app.route("/logout")
@login_required
def logout():
```

## Database Query Testing

What you are testing	How you are testing it	Expected results	Actual Results	Pass/Fail/Notes	Purpose
models.User.query.get(int(id))	Running the code	user.id	Returns expected	Pass	Query user

			results		
<code>models.Silhouette.query.filter_by(brand_id=id).order_by(models.Silhouette.id).all()</code>	Running the code	<code>silhouette.name</code>	Returns expected results	Pass	Query silhouettes
<code>results = models.Shoe.query.filter_by(silhouette_id=4).order_by(models.Shoe.name).all()</code>	Running the code	<code>Shoe.name</code>	Returns expected results	Pass	Query shoes
<code>models.User.query.filter_by(email=form.email.data).first()</code>	Running the code	<code>user.id</code> <code>user.email</code> <code>user.password_hash</code>	Returns expected results	Pass	Check if the user is registered.
<code>models.User(email=form.email.data)</code>	Running the code	Creates a new user.	Returns expected results	Pass	Registering user
<code>user.set_password(form.password.data)</code>	Running the code	Sets password for the user.	Returns expected results	Pass	Set and encrypt the password.
<code>user.check_password(form.password.data)</code>	Running the code	Check if the user inputted the correct password.	Returns expected results	Pass	Check that the password is correct.

## Data Integrity

Source

Stockx

## Jordan 1 Retro

Chicago (2015)

Condition: **New**



Size: All ▼

Place Bid

Buy for \$3,000

[Sell for \\$3000 or Ask for More →](#)

Last Sale:

**\$3,000**

▼ -\$500 (-26%)

[View Asks](#)

[View Bids](#)

[View Sales](#)

### Related Products

## SOURCE IMAGE:

› the-rumbling › static › images › Jordans › Jordan1



white1.



Bredtoe1.png



Chicago1.png



Deepblue1.png

## Database

Colourway matches, price matches and silhouette is 1 which is for jordan1 so they all match. Shoe image in the static folder has the correct shoe image and the path is the same as in my database.

id	silhouette_id	name	image	price
1	1	Chicago	static/images/Jordans/Jordan1/Chicago1.png	3000

## Web



Everything queried from the database matches what is shown on my website, so data integrity is good.

## Relevant Implications

### Database Relevant Implications

**Intellectual property** is a property that is intangible and has commercial value, for example, creations, copyright, trademarks, patents. People cannot legally use this property without explicit permission from the owner. I got images of the shoes from google images using the available for reuse tag, so it all falls under fair use.

**Sustainability and future proofing** is whether the database was designed so that it adapts to changes that may occur in the future. My database design has different tables for different silhouettes of shoes and tables for shoes. So that the shoe information can be queried without the irrelevant information from the silhouettes, by reducing the amount of irrelevant information we query we increase the performance of the query and therefore reduce the waste.

### Web relevant implications

**Aesthetics** is how visually appealing the website looks. This includes things like colour palette, fonts, etc. Having good aesthetics on a website creates good impressions on the user and they are more likely to use the website rather than leave. My website addresses this by having a simple colour palette in order to not distract from the crucial information and a very readable font. It also uses a very common layout with a header and navigation bar at the top and the information on the bottom. As this is a common layout it will be easier for the user to use and they are more likely to stay on the website.

**Usability** is how easy a website's interface is to understand, these can be separated into heuristics that help the user have a good user experience. One such method employed by my



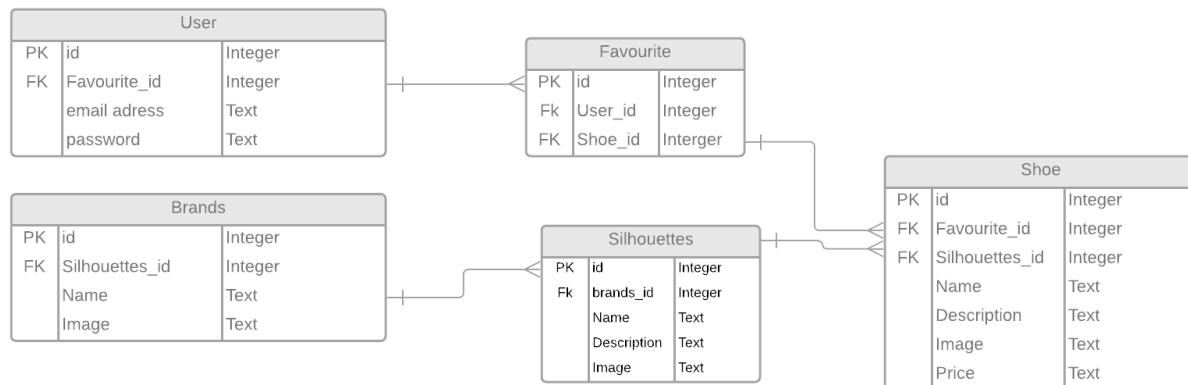
website is preventing errors and error handling. E.g. when users enter a url that does not exist it sends them to a 404 page. And ensuring that all functions work and don't produce any errors.

## Relevant conventions

**Match between system and the real world** is ensuring that it speaks the same language as the user, and uses phrases that the user is familiar with. My website should be familiar to those who have an interest in sneakers as my website is similar to sneaker marketplace sites like stockx.

**User control and freedom** is ensuring the user doesn't feel trapped in your website. My website has navigation buttons to prevent users from being stuck on one page.

## DataBase



## Tools used

IDE - Visual Studio Code

Database - SQLiteStudio

Languages - CSS, HTML5, Python Flask Library, and SQLiteStudio - Query stuff

General -

Google Chrome - used for both viewing site and searching online for documentation to aid in problems I was experiencing

Git - version control service.

## Changes

- On february 11th I layed out my initial framework and did my first commit.

- February 25th, I committed for the second time, adding my layout that would appear on each route/page.


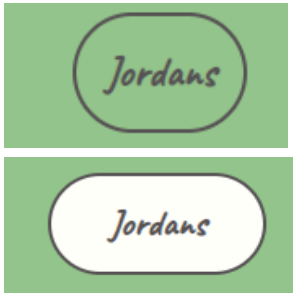


#### Website changes due to testing





- removed pages for individual shoes as it is redundant, no new information is being presented.
- Removed search feature because it was too difficult to implement.
- Removed footer because it was showing redundant information

#### Database changes due to testing

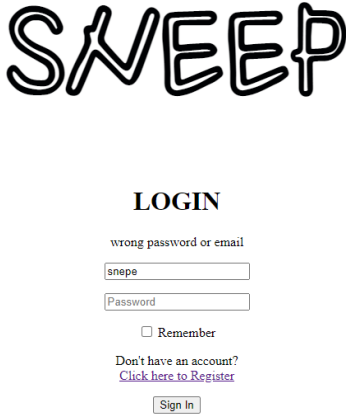

- Removed description for shoe because each individual shoe has the say design as the silhouette just different colours, which you can just look at the picture to see.

#### Testing

Features to test	Input	Expectations	Results
Sticky Navbar		Navbar should be in a constant position	Navbar successfully sticks to the top of the page
Functional buttons		Button should expand and give users the ability to click on it	Worked as expected
Proper Navbar layout		The nav buttons are properly center	Buttons are not lopsided
Overloaded login page		Reject massive values	Works as normal thus needs to implement a value limit

<b>Testing route buttons work</b>  Testing jordans		Jordans button will take me to jordans page	Worked as expected.
Testing adidas		Take me to adidas page	Worked as expected.
Testing nike		Take me to nike page	Worked as expected.
Testing other		Take me to other page	Worked as expected.


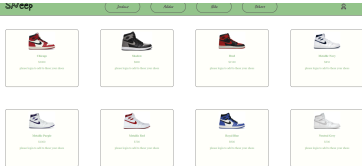

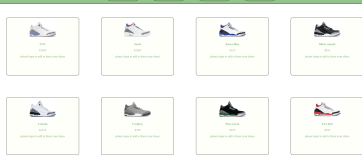
### Testing Login and Register

Login with incorrect details		Should tell me that my details are wrong.	Shows wrong password or email so works as expected.
Login with correct details		Should log me in and show a message.	Does as expected.


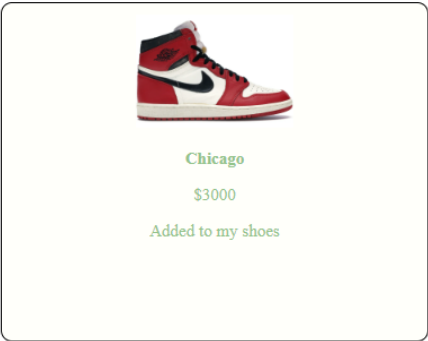
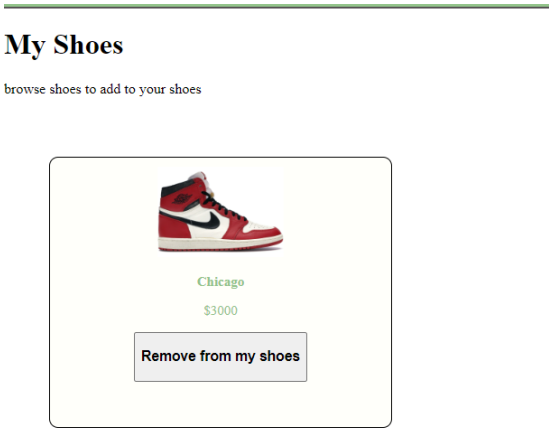
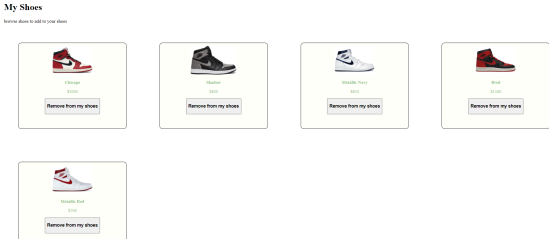
Register with existing email address	<p><b>Register</b></p> <p>email is already being used</p> <p>18031@burnside.school.nz</p> <p>.....</p> <p>.....</p> <p>Register</p>	Should tell the user that the email is already being used.	Does as expected.
Registering with invalid email	<p><b>Register</b></p> <p>18031@b</p> <p>Invalid email address.</p> <p>Password</p> <p>Confirm Password</p> <p>Register</p>	Tell the user email is invalid	pass
Registering with password too big	<p><b>Register</b></p> <p>18031@burnside.school.nz</p> <p>Password</p> <p>Field cannot be longer than 36 characters.</p> <p>Confirm Password</p> <p>Field must be equal to password. Field cannot be longer than 36 characters.</p> <p>Register</p>	Tell the user password is too long	pass

<p>Registering with correct details</p>	<h2 data-bbox="605 281 805 331">LOGIN</h2> <p data-bbox="526 386 888 415">You are now a registered user.</p> <div data-bbox="542 447 875 485"> <input type="text"/> </div> <div data-bbox="542 516 875 554"> <input type="password"/> </div> <div data-bbox="623 590 797 621"> <input type="checkbox"/> Remember         </div> <p data-bbox="566 657 847 686">Don't have an account?</p> <p data-bbox="574 690 839 720"><a href="#">Click here to Register</a></p> <div data-bbox="652 751 761 789"> <input type="button" value="Sign In"/> </div>	<p>Tell the user that they are registered and redirect to login page</p>	<p>pass</p>
<p>Registering when confirm password does not match password</p>	<h2 data-bbox="558 945 753 995">Register</h2> <div data-bbox="506 1037 807 1073"> <input type="text"/> </div> <div data-bbox="506 1100 807 1136"> <input type="password"/> </div> <div data-bbox="506 1163 807 1199"> <input type="password"/> </div> <p data-bbox="477 1228 834 1257">Field must be equal to password.</p> <div data-bbox="600 1285 709 1323"> <input type="button" value="Register"/> </div>	<p>Tell the user that the confirm password field must be the same as password.</p>	<p>pass</p>

## Shoes

Clicking on shoes image or text jordan1	 <p><b>Jordan 1</b></p> <p>The original Air Jordan sneakers were produced exclusively for Michael Jordan in late 1984, and released to the public on April 1, 1985. The shoes were designed for Nike by Peter Moore, Tinker Hatfield, and Bruce Kilgore.</p>	Take me to all shoes that relate to jordan 1	 <p>Take me to the correct page so works.</p>
Clicking on shoes image or text jordan3	 <p><b>Jordan 3</b></p> <p>The Air Jordan III released in 1988 and retailed for \$100. Michael Jordan wore them during the 1987-88 NBA season. Unlike today's standard barrage of colorways for every signature shoe, the Air Jordan III released in only four original colors, including White/Cement Grey (that Michael won the '88 Slam Dunk contest in), White/Fire Red, "True Blue", and the most famous colorway of Black/Cement Grey.</p>	Take me to all shoes that relate to jordan 3	 <p>Takes me to correct page so is working.</p>

## My many to many functions

Opening my shoes page when no shoes are added		Show some text that this is the shoes page and tell them how to add shoes	Does what is expected.
Adding a shoe		Replace the add button with text that shows that it's been added.	Works as expected
Opening my shoe page when a shoe has been added		Display the shoe that has been added.	Works as expected
Adding multiple shoes to my shoes		Add all the shoes to database and show all the shoes that I added	Works as expected

