

# **Technical Specification for developing**

# **E-commerce website ‘Don’t Panic’**

**Presented by:**

Margaret Egan x17168651

Gavin Hanna x17164974

Cornelia Vos x17157773

**Table of Contents**

|  |  |  |
| --- | --- | --- |
|  | | Page |
|  | Title Page | 1 |
|  | Table of Contents | 2 |
| 1. | Introduction | 3 |
| 2. | Model | 3 |
| 3. | Key User Flows | 4 |
| 4. | Access Control | 11 |
| 5. | 3rd Party Gems and external interfaces | 11 |
| 6. | Additional Features | 12 |
| 7. | References | 12 |
| 8. | Appendices | 13 |

### 

### 1. Introduction

The team developed an e-commerce website ‘Don’t Panic’ to sell products that relate to the universe portrayed in the Douglas Adams book, ‘The Hitchhiker’s Guide To The Galaxy’. The website should be fully functional by the submission date: it should showcase the abilities of the team in app design and successful implementation thereof. It must be noted that the team used the app created in tutorials as a template to structure the store. The team also relied on the book for product description.

As per industry standard, the website is fully responsive. This was accomplished by using CSS3: specifically, the use of media queries and flexbox positioning. As discussed at the presentations, this could also have been achieved in Ruby.

The design of the page has changed slightly since the original concept was mooted in the proposal. Adding social media plug-ins appeared overly complex; and as discussed in class, it appeared more appropriate to set up a dummy payment system rather than one relying on PayPal. As per the feedback provided by the lecturer, we have also removed the external linkage to the BBC website. The presentation feedback suggested that an admin section would be appropriate, so this has been included.

### 2. Model

### The webshop is dynamic in nature, so an integrated relational database is included. The database uses an Object Relational Mapping Framework called ActiveRecord, which comes with Ruby on Rails as a recommended default.

### Each user may have many orders and each order may comprise many items. The model includes the following entities:

### Categories Table

### Id: integer (PK)

### Title: string

### Description: text

### Image: string (URL)

### Created\_at: datetime

### Updated\_at: datetime

### OrderItems Table

Id: integer (PK)

### Item\_id: integer

### Title: string

### Description: text

### Quantity: integer

### Price: decimal

### Order\_id: integer (FK)

### Created\_at: datetime

### Updated\_at: datetime

### 

### Orders Table

Id: integer (PK)

Order\_date: datetime

User\_id: integer (FK)

Status: string

Created\_at: datetime

Updated\_at: datetime

### Products

Id: integer (PK)

Title: string

Description: text

Image\_url: string

Price: integer

Category: string

Created\_at: datetime

Updated\_at: datetime

### Users Table

Id: integer (PK)

Email: string

Encrypted\_password: string

Reset\_password\_token: string

Reset\_password\_sent\_at: datetime

Remember\_created\_at: datetime

Sign\_in\_count: integer

Current\_sign\_in\_at: datetime

Last\_sign\_in\_at: datetime

Current\_sign\_in\_ip: string

Last\_sign\_in\_ip: string

Admin: boolean

Created\_at: datetime

Updated\_at: datetime

### 3. Key User Flows

### The routing of the system works as follows.

### Users/Customers may:

### a) Browse the home page

### URL: https://hdwt-rails-tut-gavhanna.c9users.io

### Route: to => "site#home"

### b) Browse products

### URL: https://hdwt-rails-tut-gavhanna.c9users.io/products

### Route: to => 'products#index'

### c) Browse categories

### URL: https://hdwt-rails-tut-gavhanna.c9users.io/category

### Route: to => ‘site#home’ (partial)

### d) Sign in

### URL: https://hdwt-rails-tut-gavhanna.c9users.io/users/sign\_in

### Route: to => last page viewed

### e) Register

### URL: https://hdwt-rails-tut-gavhanna.c9users.io/users/sign\_up

### Route: to => last page viewed

### f) Interact with cart

### URL: https://hdwt-rails-tut-gavhanna.c9users.io/cart

### Route: to =>'cart#index'

### Route: to =>'cart#add'

### Route: to =>'cart#remove'

### Route: to =>'cart#clearCart'

### 

### g) Find out information about the site

### URL: https://hdwt-rails-tut-gavhanna.c9users.io/about

### Route: to => 'static\_pages#about'

### h) Go to Profile

### URL: https://hdwt-rails-tut-gavhanna.c9users.io/profile

### Route: to => 'static\_pages#profile'

### 

### i) Edit Profile

### URL: https://hdwt-rails-tut-gavhanna.c9users.io/users/edit

### Route: to => edit\_user\_registration\_path

### j) Search

### URL: https://hdwt-rails-tut-gavhanna.c9users.io/search

### Route: to => 'products#search'

### Flow diagram for customers to the websiteScreen%20Shot%202018-04-21%20at%2017.40.20.png

### The Customer path through webpages:

### Screen%20Shot%202018-04-21%20at%2017.40.20.png

* Home – Welcomes the user to the website and shows the most popular webshop products.
* Site navigation menu – Present at the side of the page, it includes a logo and links to the main pages. On smaller screened devices, a hamburger will appear, and be clickable to display the navigation. By freeing up screen real estate for pertinent material on smaller screens, and keeping the navigation menu within easy reach, it should be more user-friendly. It also includes search functionality for items entered by the user that appear in the catalogue.
* About us – Allows for users to see the story of the origins of the website
* Products– These display images, description and pricing information. Customers may add inventory to the cart by selecting ‘Add to Cart’, or view the item via selecting ‘Show’.
* Log in – Prompts the user to enter their email address and password for verification.
* Registration – Prompts user to enter details to create an account. This is used to verify the user’s permission to order items.
* Edit User- Used to amend personal details.
* Profile – Contains information on the user’s order history, current cart and sign in information.
* Cart – Can’t be accessed until log in, but shows items ordered; allows user to change items; clears cart should they change their mind; and saves items until checkout.
* Sign Out – Prompts the user to confirm that they wish to sign out, and saves cart if no purchase is made.
* Payment – Customer confirms their selection in the cart and presses check out. This leads to a thank you page confirming that they have made an order. Pressing ‘Pay now’ brings the customer to a thank you page to acknowledge a completed transaction. A banner message also appears showing ’transaction complete’. This section would need to be adapted to connect to an actual payment portal in the next iteration.
* Admin – only accessible by site administrators. The admin page contains a list of all site users, with the option to either add or remove admin privileges to and from users excluding the current user.
* Thank you – Acknowledges the purchase of items

### 

### Administrators may:

### Add; edit; or delete products

### Edit product information (pictures, text and price)

### Add or remove admin status to other users.

Site security will be ensured by:

* Providing effective session management
* Hashing password details
* Taking measures against SQL/HTML injection
* Website using HTTPS protocol

The Database contains:

* User details
* Product details
* Categories
* Orders
* Order items

The site is linked to the database so that when a query is submitted, the database will return the information and information will be displayed on screen. Once a product has been purchased, this is recorded in the database and removed from the list of available stock.

Controllers:

The **static pages controller** retrieves information required from the database. This includes login and logout session and profile information. It can create, update and delete profile information including log in details.

Views that should be rendered are:

* Sign in page
* Sign up page
* Sign out page
* Profile page with order history
* Forgot Password page

The **products controller** retrieves information required from the database. This includes stock levels and descriptions. It can create, update and delete products.

Views that should be rendered are:

* An index for products
* Product pages with detailed information on each product.
* Page allowing an admin user to add a product

The **orders controller** will retrieve information relating to the customer id; order items; order total; and order date. In the next iteration, this would include shipping details. It can create, update and delete orders.

Views that should be rendered are:

* Order information including items, cost, transaction status and date placed.

The **cart controller** retrieves information required from the database. This includes product information of item added to the cart and if the cart contains items from a previous session. It can add, update and remove products.

Views that should be rendered are:

* Shopping cart page

Flash Variables:

These include –

* “You need to sign in or sign up before continuing”
* “Signed in successfully”
* “Signed out successfully”
* “Transaction complete”
* “Admin privileges granted for: <user>”
* “Admin privileges removed for: <user>”

4. **Access Control**

**a)** **Admin**

* The website will allow multiple administrators.
* Administrators control user access privileges; add edit or destroy products; edit their own historical orders.
* Administrators also control stock details such as creating and updating levels of stock; price; item descriptions and pictures

b) **Registered users**

* Users may browse at will, but must register to buy products. If the user selects a product, they are redirected to the login screen.
* Once users are signed up, they can use the login procedures as well as requesting forgotten passwords.
* The cart is displayed at all times to the user once a product has been added.
* Once a user adds an item to a cart, they are prompted to return to shopping or view the cart.
* On the cart page, users may add or delete items before confirming details are correct and checking out.
* Users may also delete their own account.

**c) Visitors**

* Visitors may view contact details and all product information without registering.
* Visitors may not buy products without logging in.
* Visitors may subscribe to email updates.

**5. 3rd Party gems and any external interfaces**

The team employed the Devise gem for user authentication. No external interfaces used.

**6. Additional features**

* Search functionality
* Mobile responsive css
* Popular products
* Show order history
* Uses javascript to add classes that trigger CSS animations for the mobile menu

**7. References**

Adams, D., 1996. *The Hitchhiker's Guide to the Galaxy.* London: Heinemann

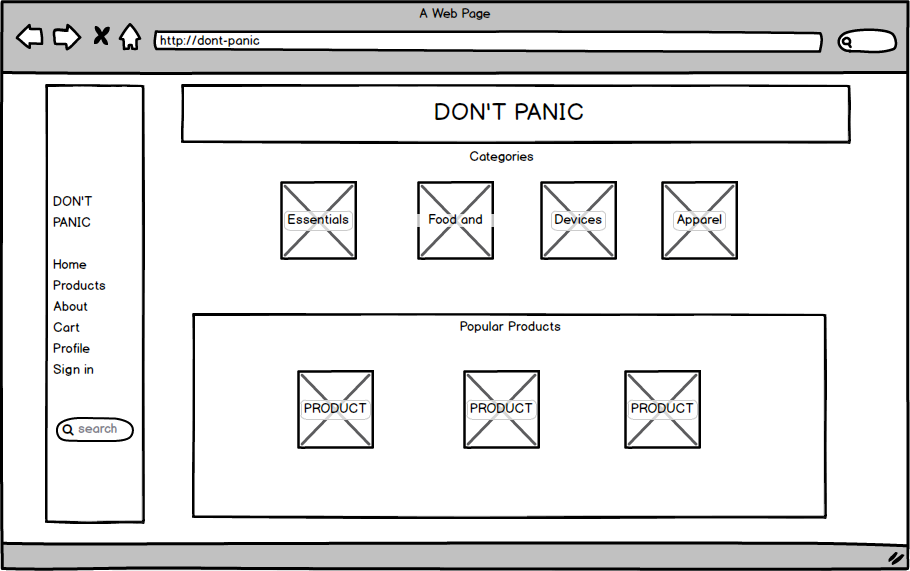
*Cheat Sheet for Devise* (2018). Available at: <https://devhints.io/devise>

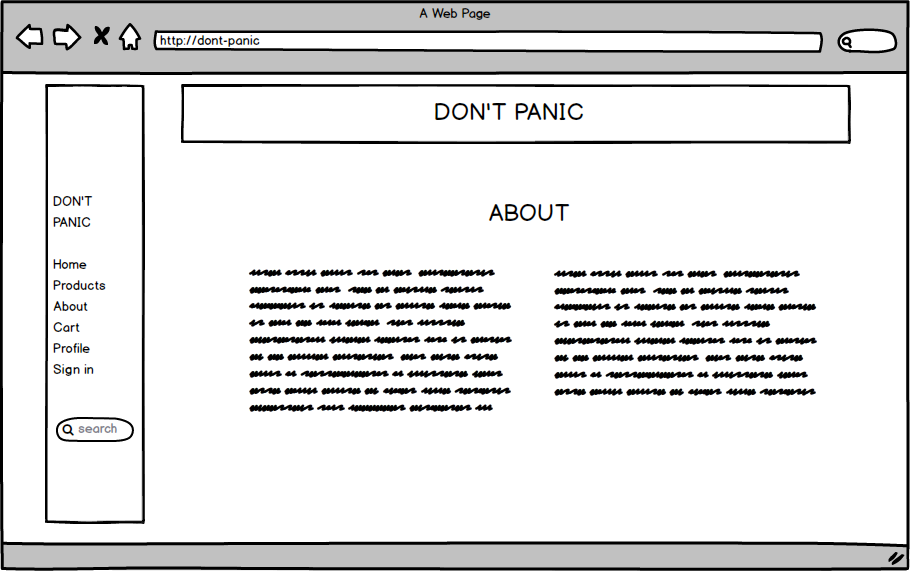
(Accessed: 19 April 2018).

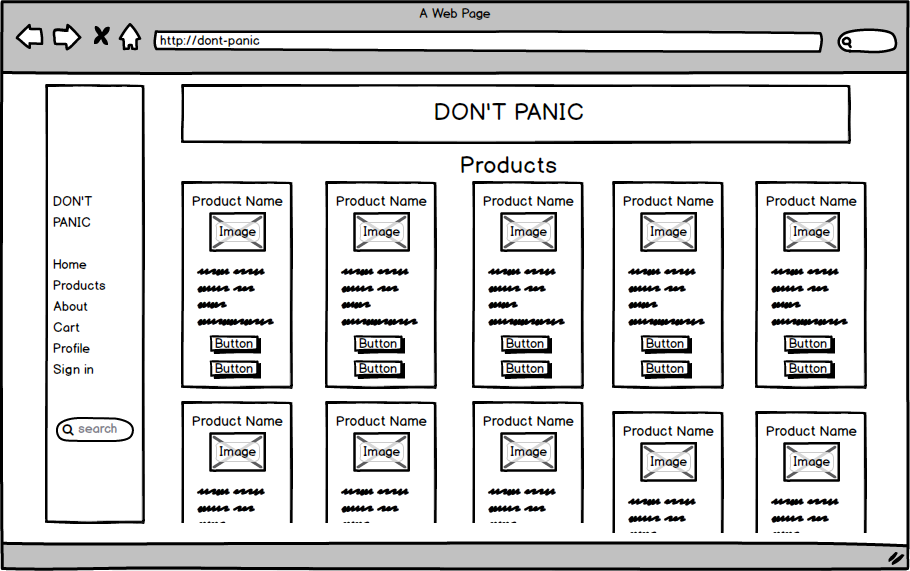
McCabe, L., 2018. *Tutorials on Ruby App development.* [oral and recorded] Semester 1 2018. Available at: < <https://moodle.ncirl.ie/mod/connecthosted/view.php?id=42389> > [Accessed throughout April 2018].

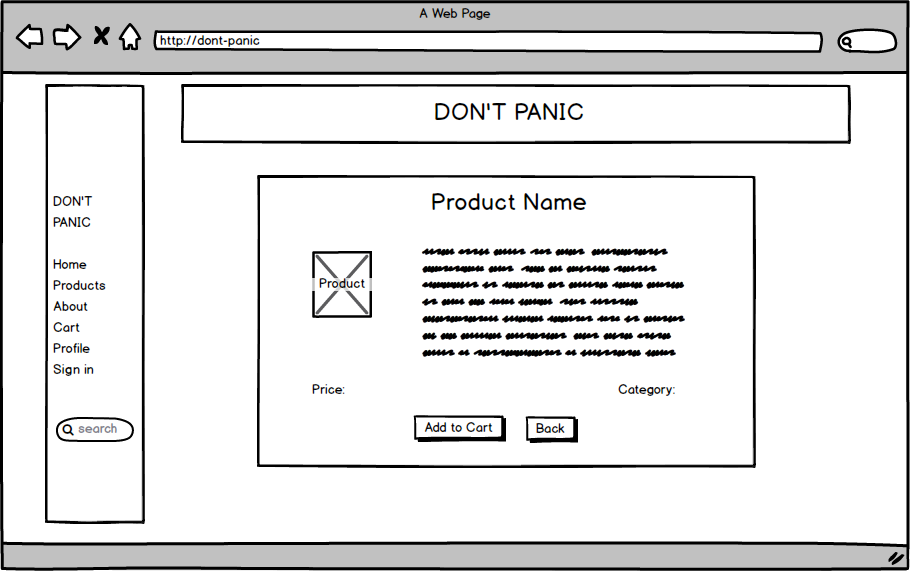
**8. Appendices**

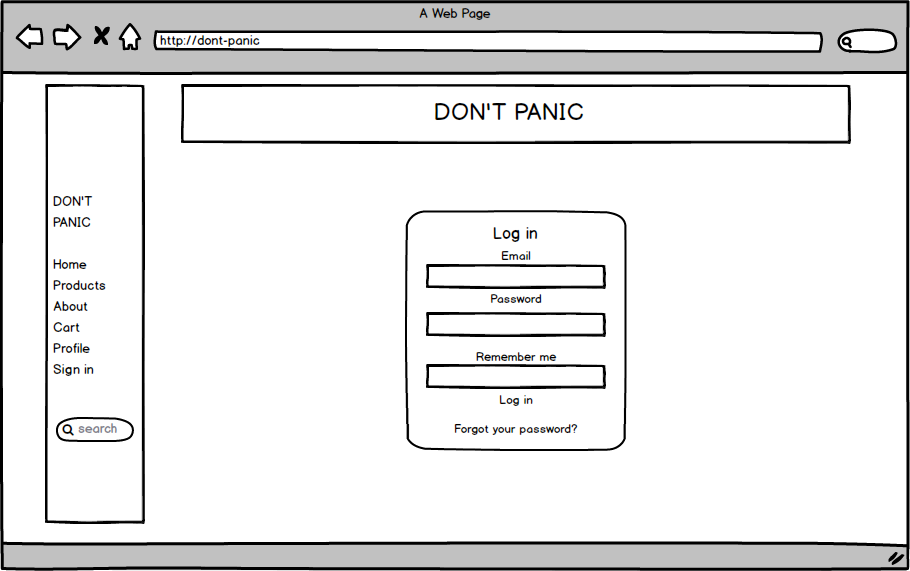
Appendix (i) Wireframes

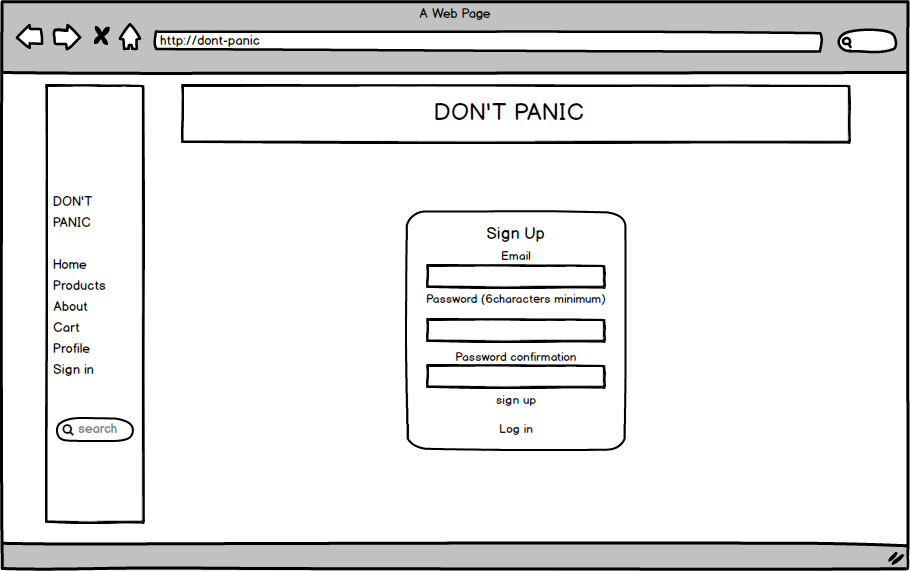


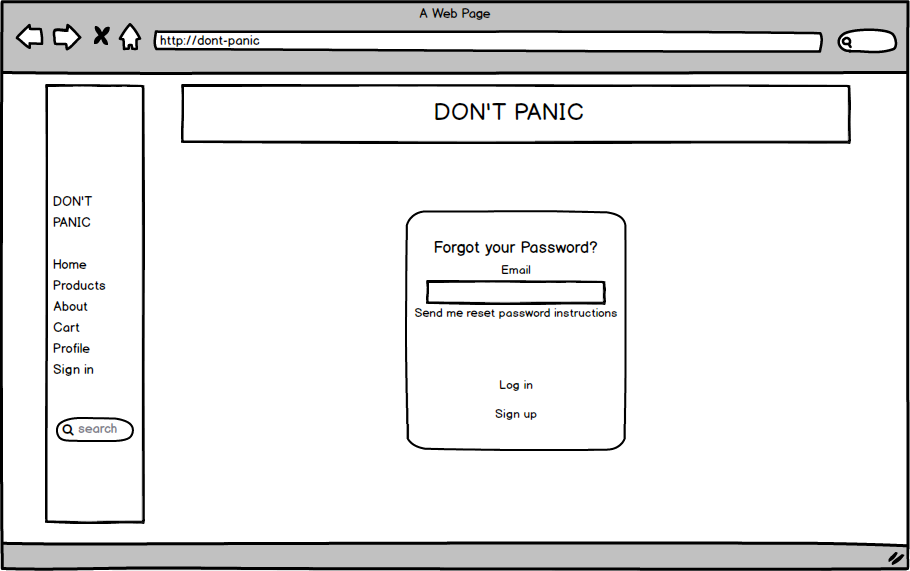


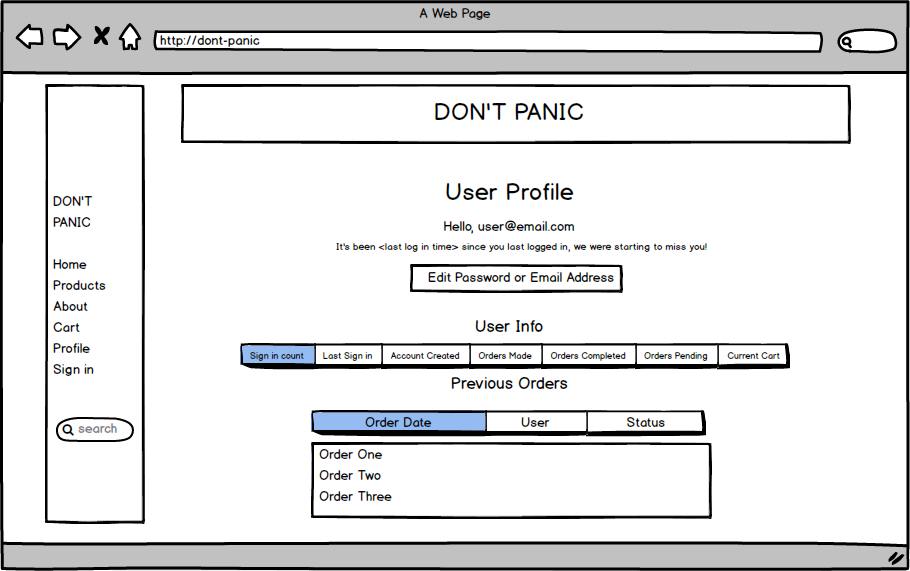




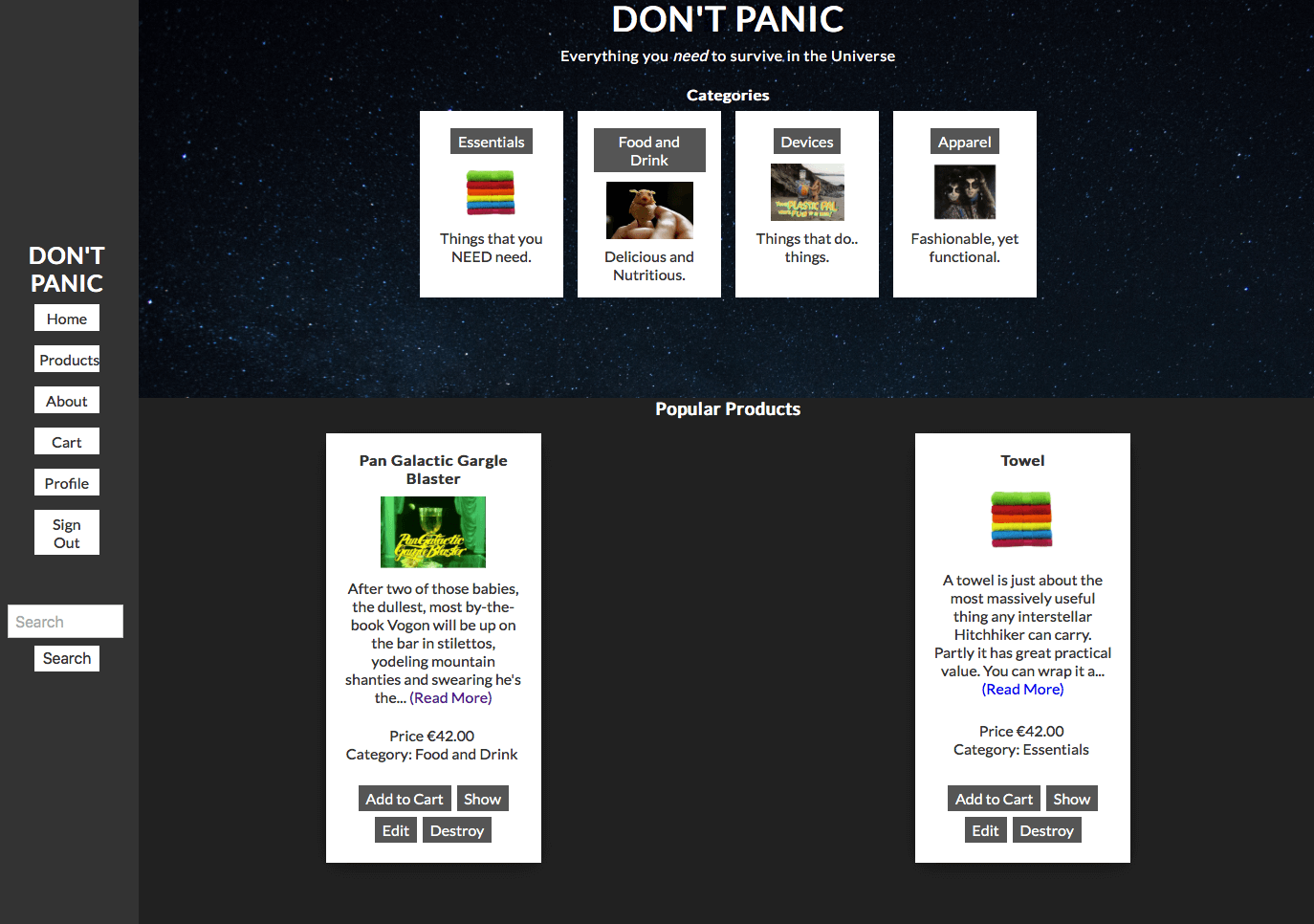




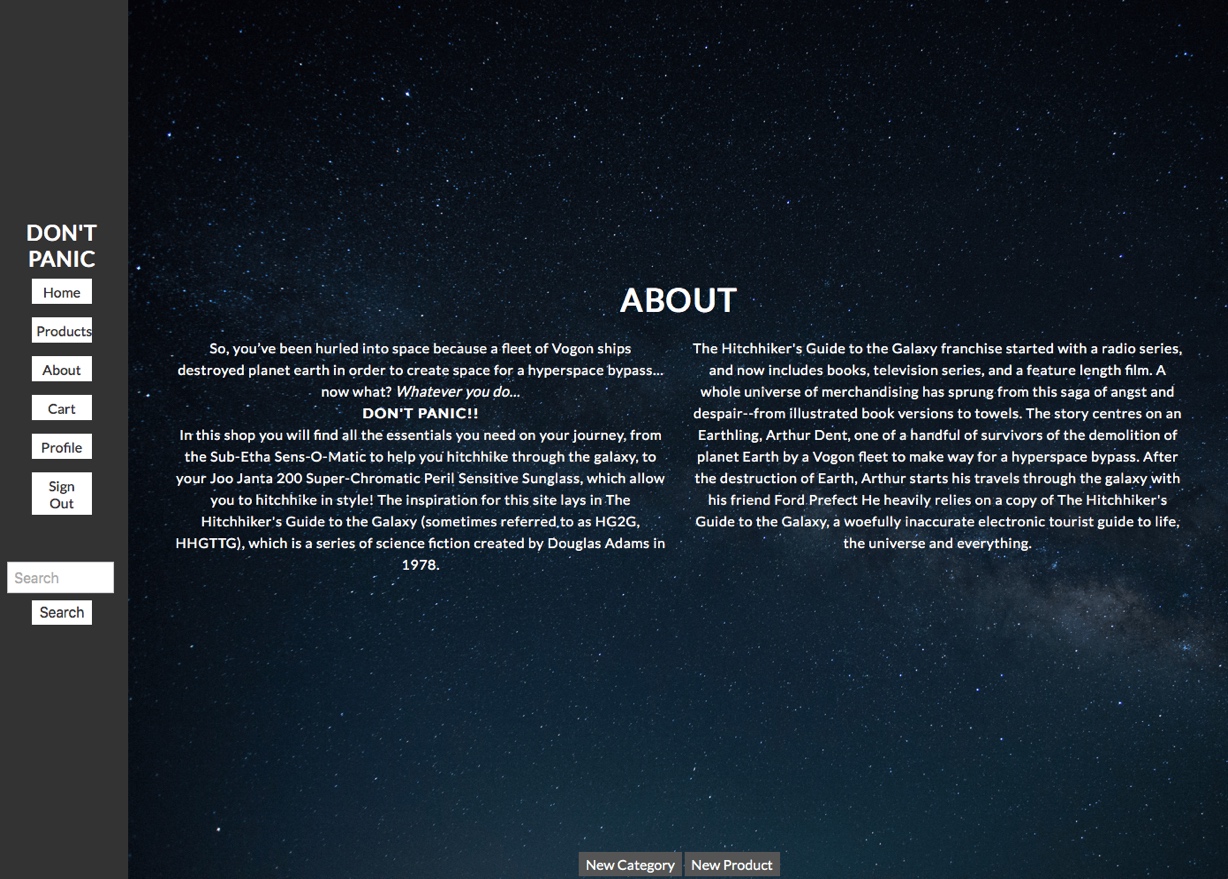




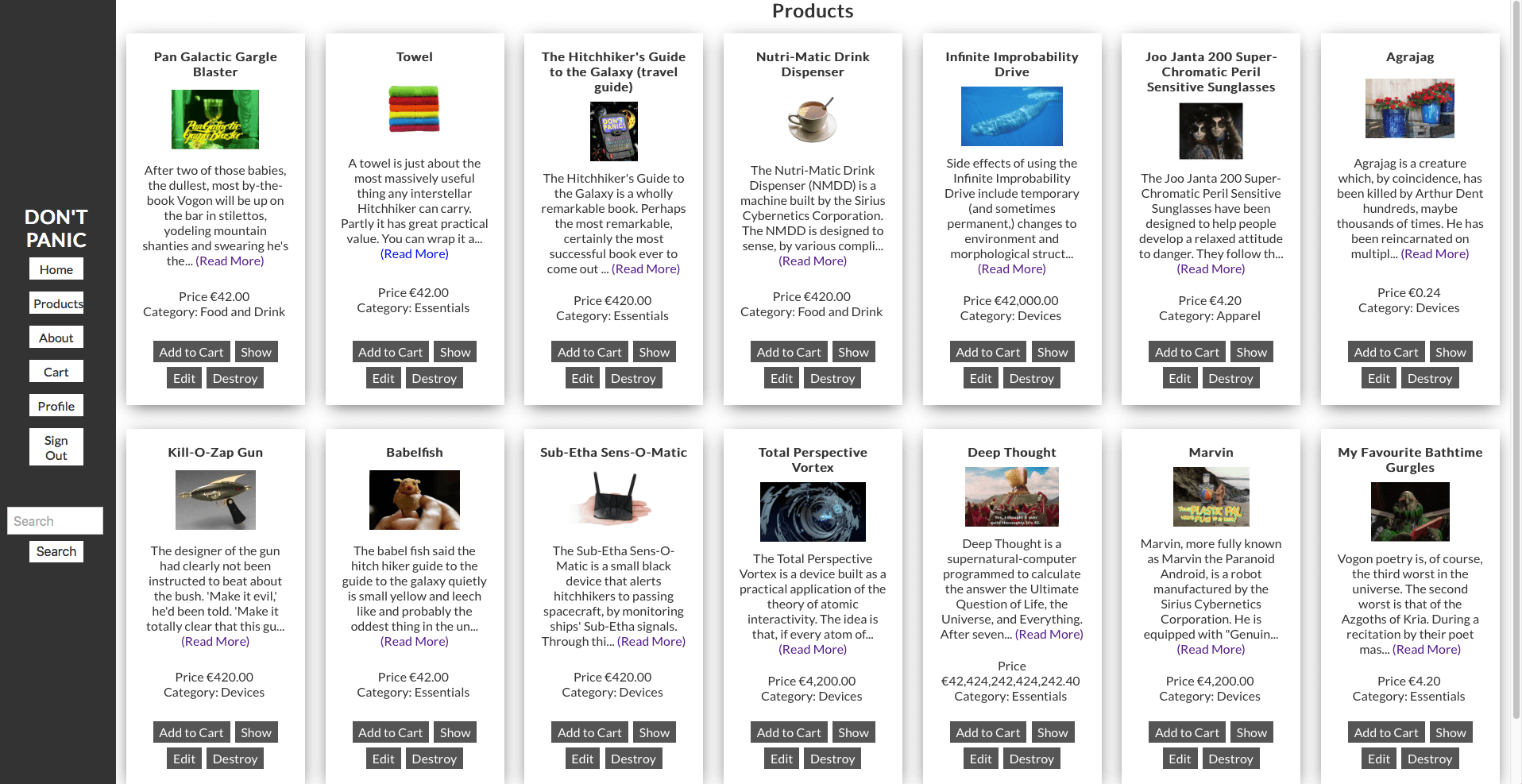
Appendix (ii) Screenshots

Home:

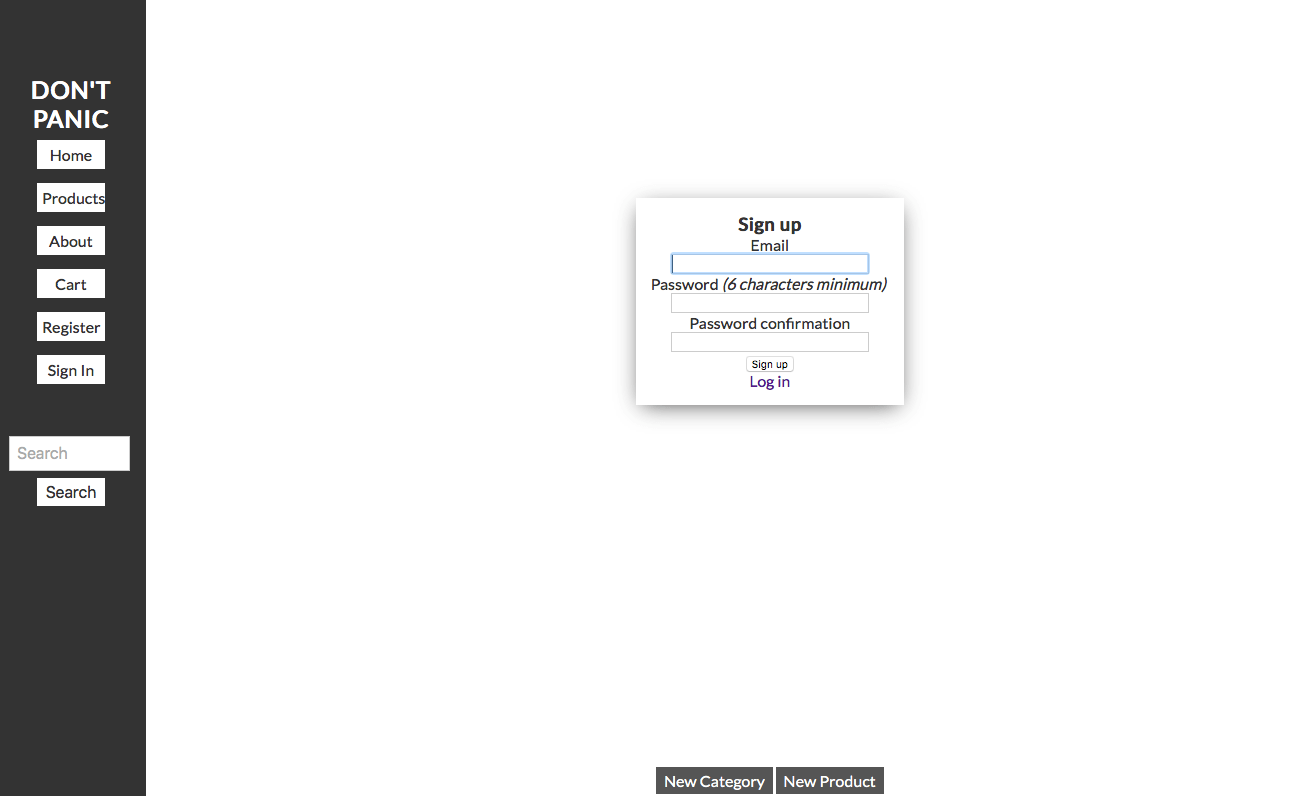
About:



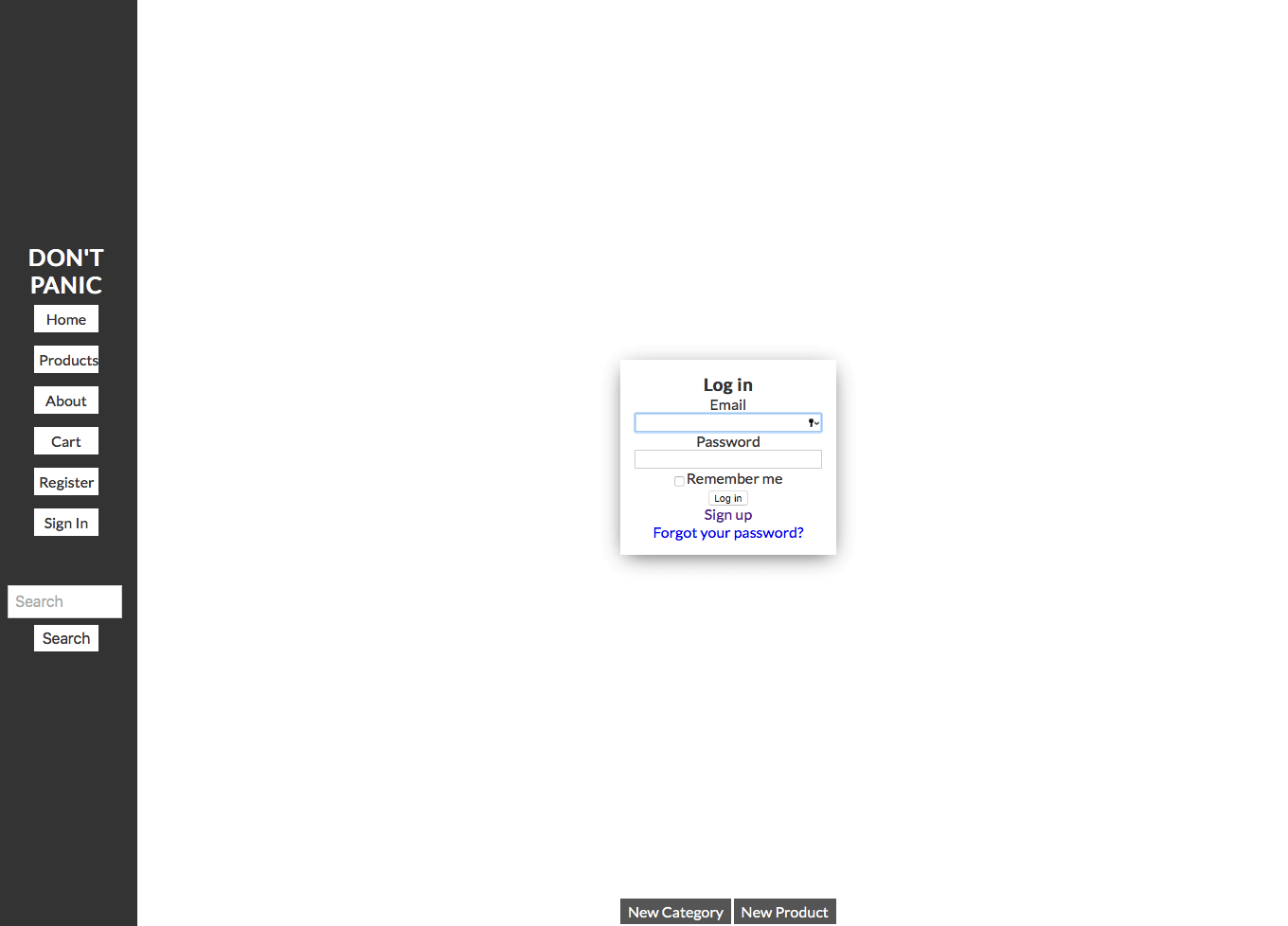
Products:



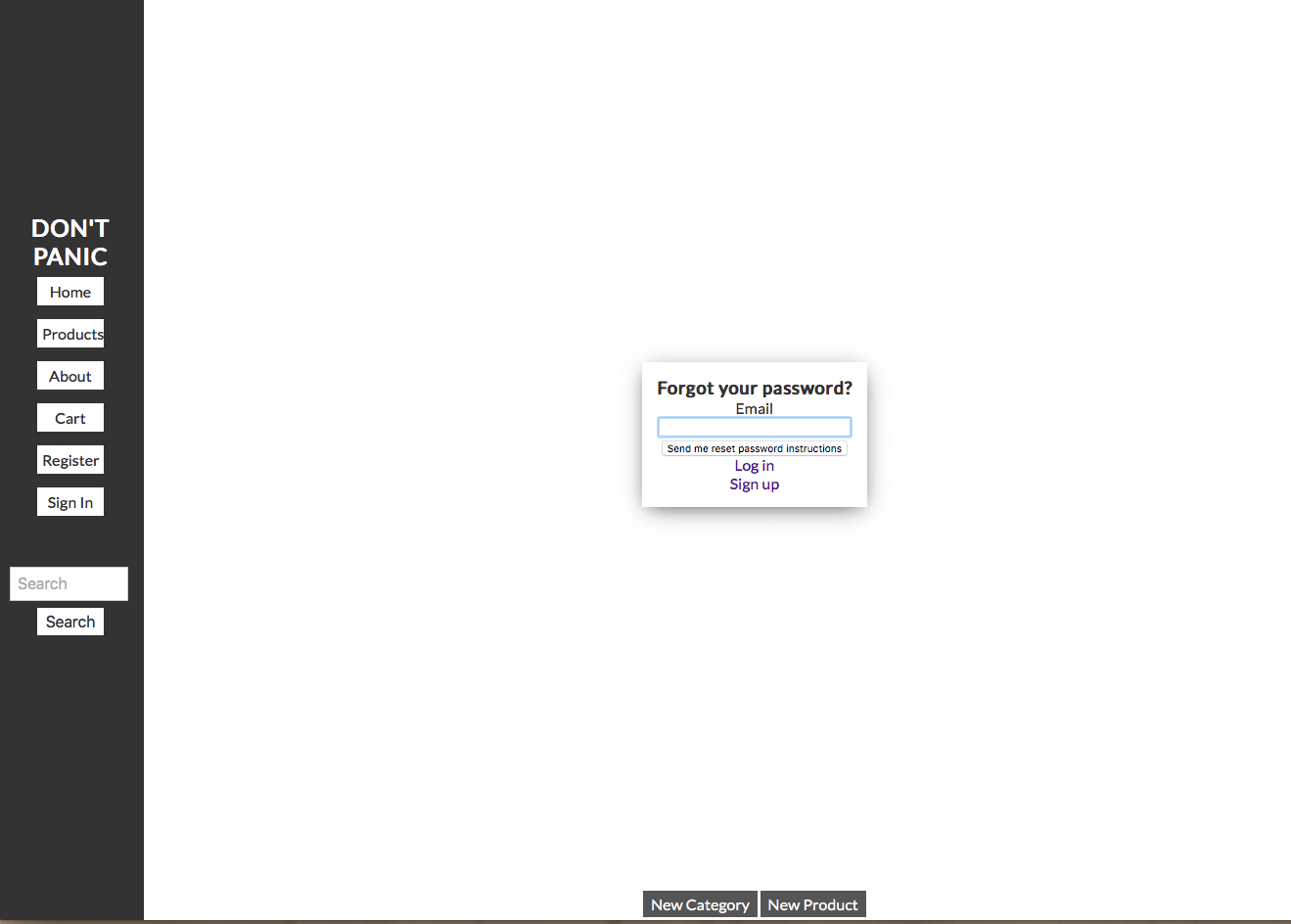
Register:



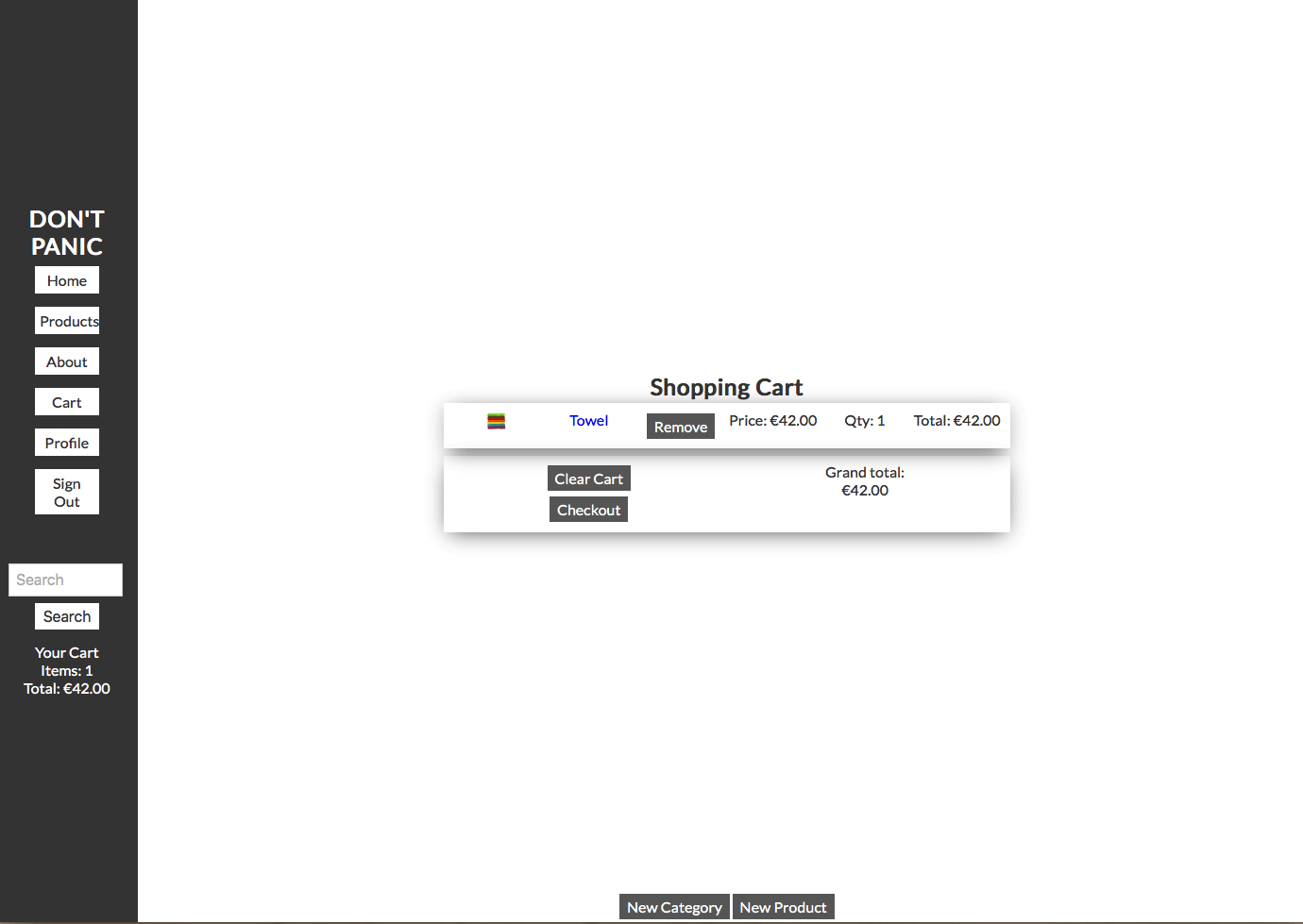
Sign in:



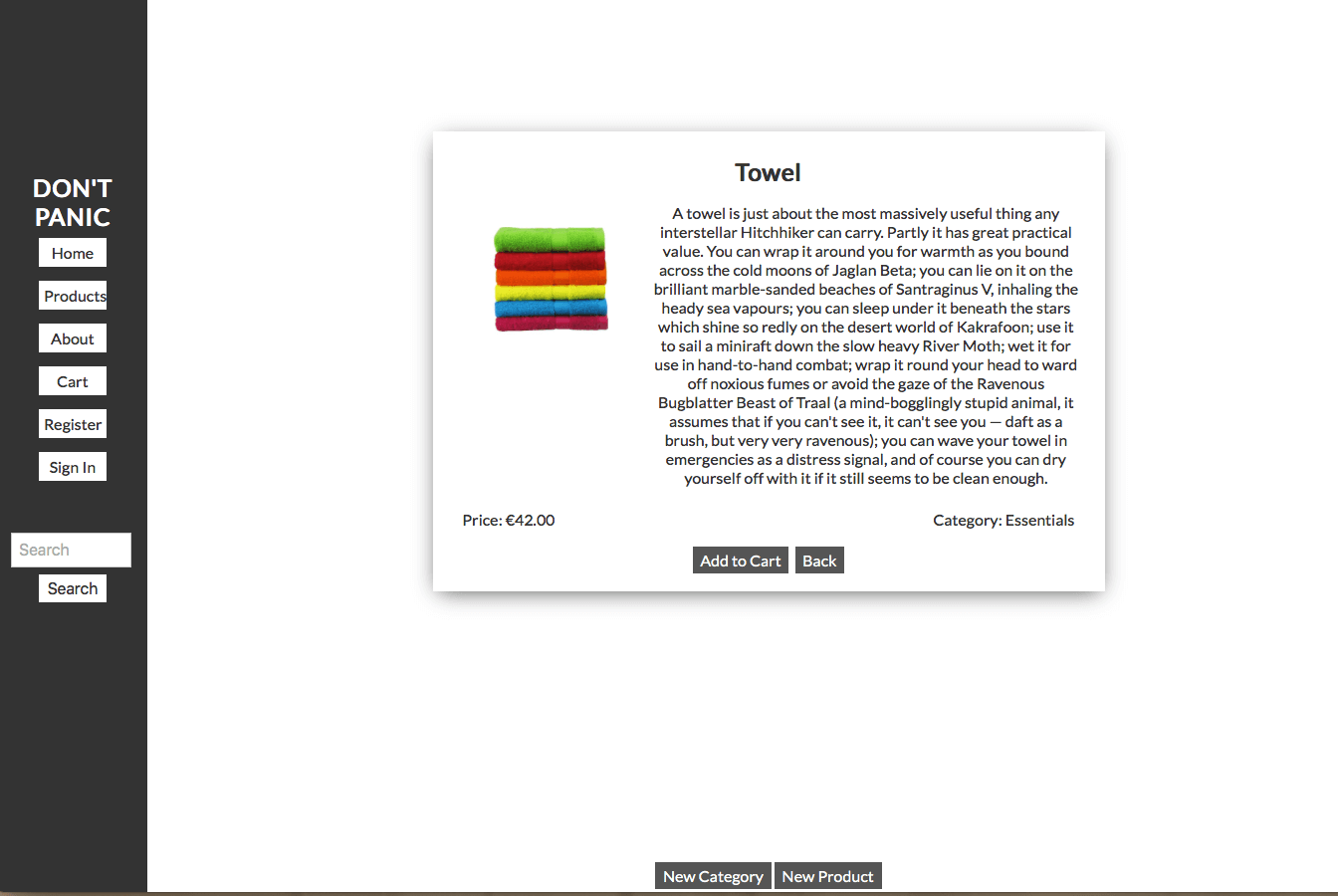
Forgot your Password:



Cart:



Individual Product:



User Profile:

