

## Project Overview

You will develop an application that provides a list of items within a variety of categories as well as provide a user registration and authentication system. Registered users will have the ability to post, edit and delete their own items.

## Why This Project?

Modern web applications perform a variety of functions and provide amazing features and utilities to their users; but deep down, it's really all just creating, reading, updating and deleting data. In this project, you'll combine your knowledge of building dynamic websites with persistent data storage to create a web application that provides a compelling service to your users.

## What Will I Learn?

You will learn how to develop a RESTful web application using the Python framework Flask along with implementing third-party OAuth authentication. You will then learn when to properly use the various HTTP methods available to you and how these methods relate to CRUD (create, read, update and delete) operations.

## How Does This Help My Career?

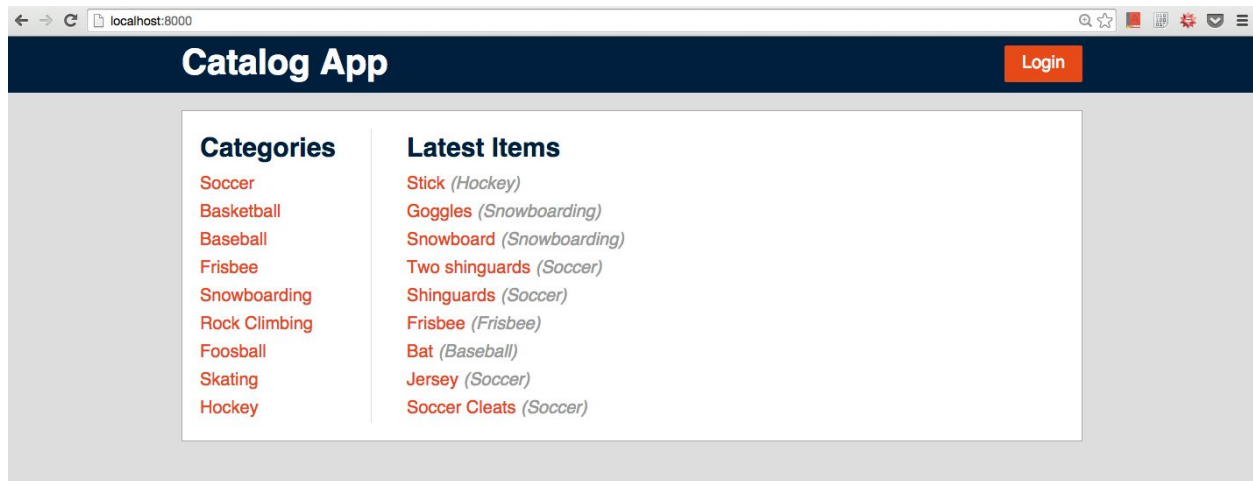
- Efficiently interacting with data is the backbone upon which performant web applications are built
- Properly implementing authentication mechanisms and appropriately mapping HTTP methods to CRUD operations are core features of a properly secured web application

## Project Display Example

**Note:** The screenshots on this page are just examples of one implementation of the minimal functionality. You are encouraged to redesign and strive for even better solutions.

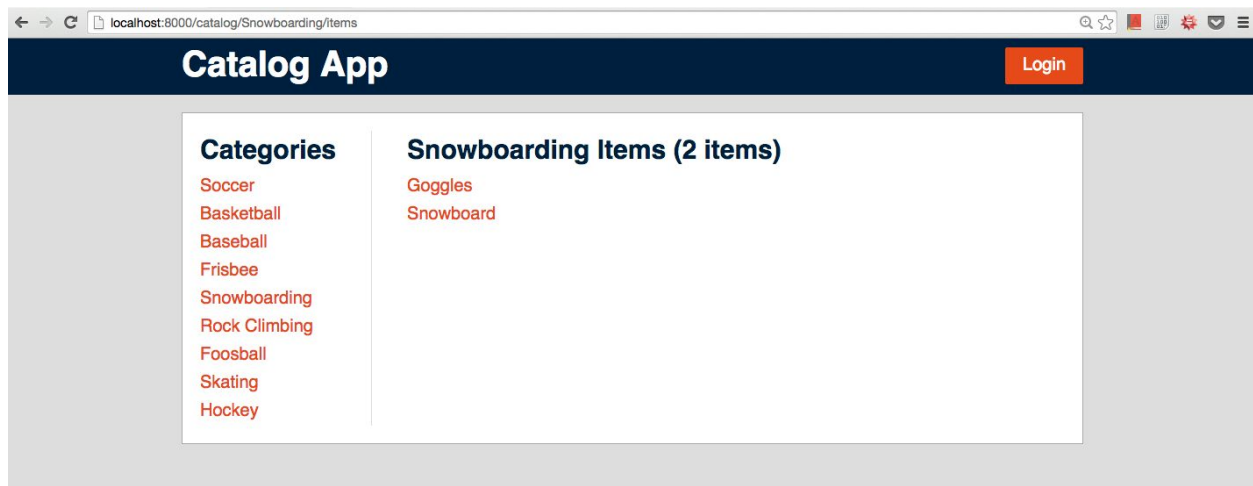
The Item Catalog project consists of developing an application that provides a list of items within a variety of categories, as well as provide a user registration and authentication system.

In this sample project, the homepage displays all current categories along with the latest added items.



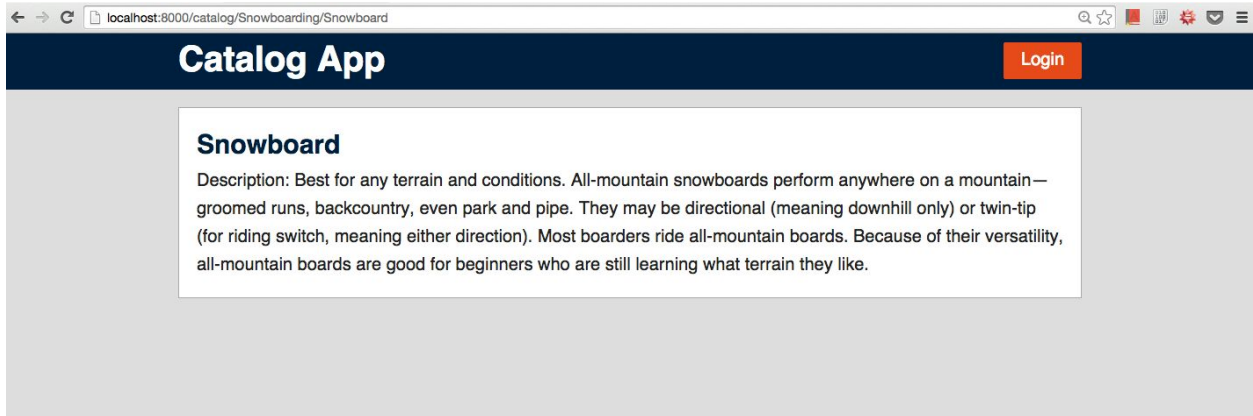
<http://localhost:8000/>

Selecting a specific category shows you all the items available for that category.



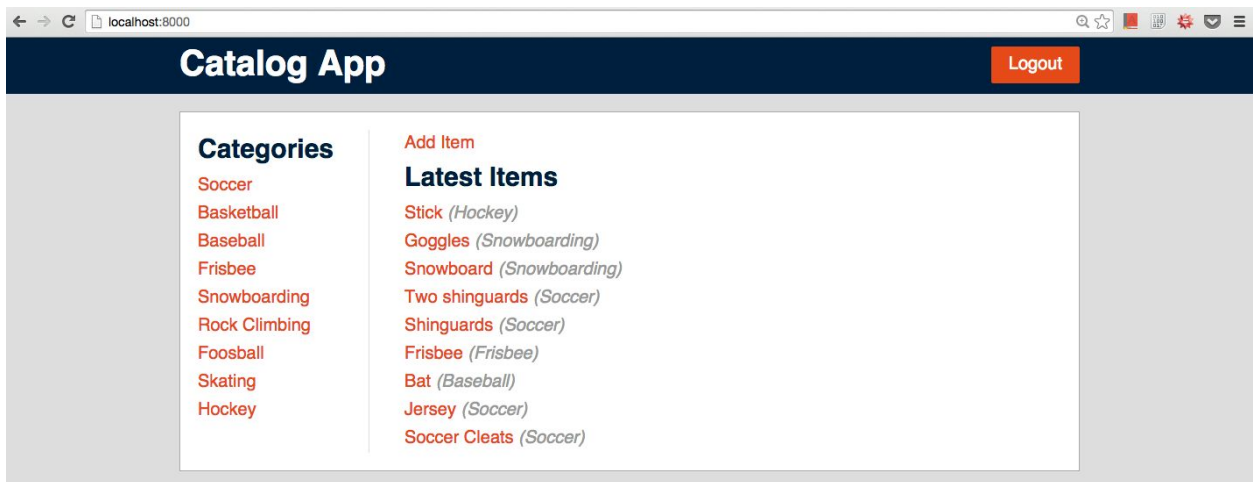
<http://localhost:8000/catalog/Snowboarding/items>

Selecting a specific item shows you specific information of that item.

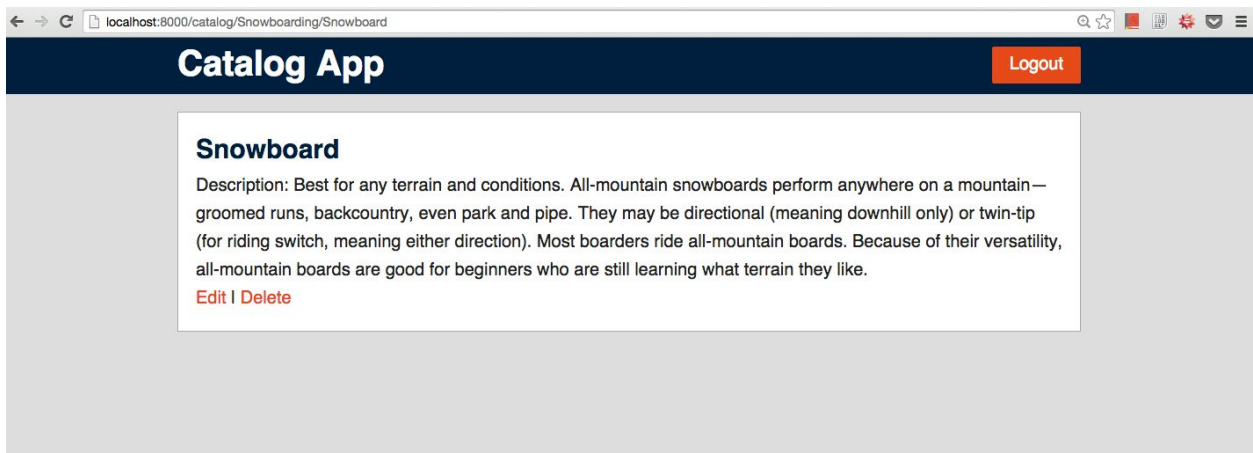


`http://localhost:8000/catalog/Snowboarding/Snowboard`

After logging in, a user has the ability to add, update, or delete item info.



`http://localhost:8000/ (logged in)`



`http://localhost:8000/catalog/Snowboarding/Snowboard (logged in)`

localhost:8000/catalog/Snowboard/edit

## Catalog App

Logout

### Edit Item

Title  
Snowboard

Description  
Best for any terrain anc

Category  
Soccer

Submit

*http://localhost:8000/catalog/Snowboard/edit (logged in)*

localhost:8000/catalog/Snowboard/delete

## Catalog App

Logout

### Delete Item

Are your sure you want to delete ?

Submit

*http://localhost:8000/catalog/Snowboard/delete (logged in)*

The application provides a JSON endpoint, at the very least.

A screenshot of a web browser window displaying a JSON file named 'catalog.json' at the URL 'localhost:8000/catalog.json'. The JSON data is a list of categories, each with an ID and name. Some categories have associated items with their own IDs, descriptions, and titles. The categories include Soccer, Basketball, Baseball, Frisbee, Snowboarding, and Rock Climbing. The browser's address bar and standard navigation icons are visible at the top.

```
{
  "Category": [
    {
      "id": 1,
      "name": "Soccer",
      "Item": [
        {
          "cat_id": 1,
          "description": "The shoes",
          "id": 1,
          "title": "Soccer Cleats"
        },
        {
          "cat_id": 1,
          "description": "The shirt",
          "id": 2,
          "title": "Jersey"
        }
      ]
    },
    {
      "id": 2,
      "name": "Basketball"
    },
    {
      "id": 3,
      "name": "Baseball",
      "Item": [
        {
          "cat_id": 3,
          "description": "The bat",
          "id": 3,
          "title": "Bat"
        }
      ]
    },
    {
      "id": 4,
      "name": "Frisbee"
    },
    {
      "id": 5,
      "name": "Snowboarding",
      "Item": [
        {
          "cat_id": 5,
          "description": "Best for any terrain and conditions. All-mountain snowboards perform anywhere on a mountain\u2014groomed runs, backcountry, even park and pipe. They may be directional (meaning downhill only) or twin-tip (for riding switch, meaning either direction). Most boarders ride all-mountain boards. Because of their versatility, all-mountain boards are good for beginners who are still learning what terrain they like.",
          "id": 7,
          "title": "Snowboard"
        }
      ]
    },
    {
      "id": 8,
      "name": "Rock Climbing "
    }
  ]
}
```

<http://localhost:8000/catalog.json>

## How will I complete this project?

This project is connected to the [Full Stack Foundations](#) and [Authentication and Authorization](#) courses, but depending on your background knowledge you may not need the entirety of both courses to complete this project. Here's what you should do:

1. Install Vagrant and VirtualBox
2. Clone the fullstack-nanodegree-vm
3. Launch the Vagrant VM (vagrant up)
4. Write your Flask application locally in the vagrant/catalog directory (which will automatically be synced to /vagrant/catalog within the VM).
5. Run your application within the VM (python /vagrant/catalog/application.py)
6. Access and test your application by visiting <http://localhost:8000> locally

Get started with this helpful [guide](#).