**Item catalog**

**Systems Design**

*Focus at the 2-3 core features and design system around that.*

User can read category and item information from a database in a web browser

User can log-in with her linkedin account, and:

Add categories and items.

Update categories and items, which belong to them.

Delete categories and items, which belong to them.

App provide an API endpoint in JSON.

User is notified by flash messages at every change of state.

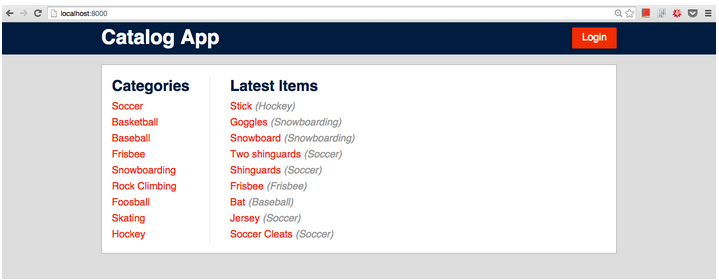
**Other:**

* Pep8 style format
* Comments
* README.me file

**Mockups**

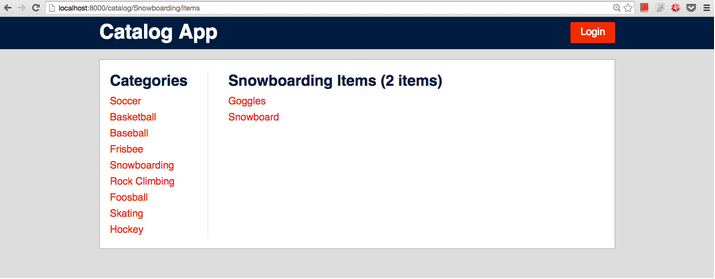
**Categories**

1. Show all categories

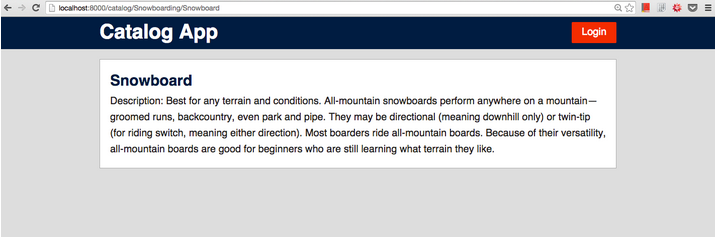


**Items**

1. Show all items in category



1. Show an item in category



1. Edit an item in category



1. Delete an item in category



**Architecture**

n Patterns of Enterprise Application Architecture, Martin Fowler outlines some common recurring themes when explaining architecture. He identifies these themes as:

“The highest-level breakdown of a system into its parts; the decisions that are hard to change; there are multiple architectures in a system; what is architecturally significant can change over a system's lifetime; and, in the end, architecture boils down to whatever the important stuff is.”

RESTful web App on Flask framework with a SQLAlchemy

1. Database
2. Flask Application
3. HTML
4. CSS

**Database**

User

user\_id (P)

name

email

profile\_pic

Category

category\_id (P)

name

user\_id (f)

Items

item\_id (P)

name

description

price  
category\_id (F)

user\_id (F)

**Application**

**Categories**

1. Show all categories
   1. ”/”
2. Create a category
   1. /new
3. Edit a category
   1. /*category\_name*/category\_id/edit
4. Delete a category
   1. /*category-name*/category\_id/delete

**Items**

1. Show all items in category
   1. /*category-name*
2. Show an item in category
   1. /category-name/item-name/
3. Create a new item
   1. /category-name/new
4. Edit a item
   1. /category-name/item-name/edit
5. Delete a item
   1. /category-name/item-name/delete

**To-dos**

* ~~Mock-ups (designs)~~
* ~~Database model~~
* ~~Routing (use flask with simple return statements)~~
* Templates & Forms
* CRUD Functionality
* API Endpoints (with JSON)
* Styling & Message flashing

**Questions for software architecture**

How will the users be using the application?

How will the application be deployed into production and managed?

What are the quality attribute requirements for the application, such as security, performance, concurrency, internationalization, and configuration?

How can the application be designed to be flexible and maintainable over time?

What are the architectural trends that might impact your application now or after it has been deployed?