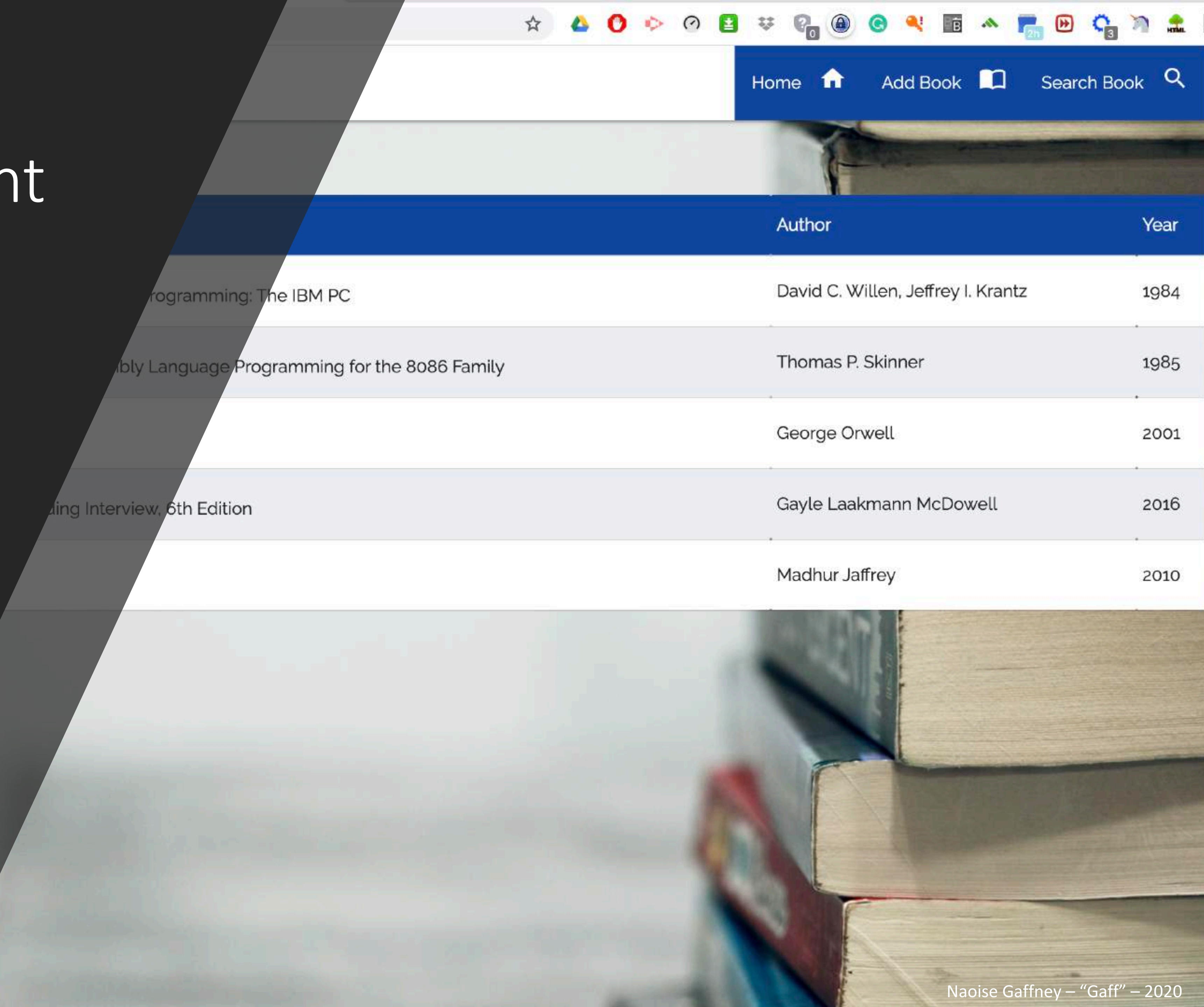
Object Document Mapper: MongoEngine

Using MongoEngine for MongoDB CRUD functions, and Flask-MongoEngine for Pagination, based on a simple Book database.

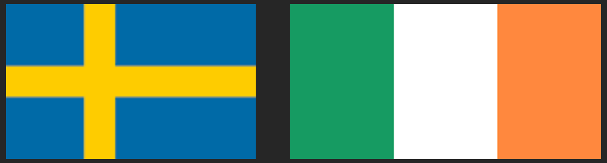


	Author	Year
rogramming: The IBM PC	David C. Willen, Jeffrey I. Krantz	1984
bly Language Programming for the 8086 Family	Thomas P. Skinner	1985
	George Orwell	2001
ing Interview, 6th Edition	Gayle Laakmann McDowell	2016
	Madhur Jaffrey	2010

Naoise Gaffney – “Gaff” – 2020

Naoise Gaffney – “Gaff”

- Code Institute Diploma in Full Stack Development Student since February 2020
- Book ODM Example is a Sub-Set of My MS3
- Worked in IT and the Business of IT since 1986; Unemployed
- Married to Fiona
- Grew up in Stockholm, Sweden, and living in Dublin, Ireland



Object Document Mapper – MongoEngine

- **Agenda**

  Overview

  Book ODM Example

  1. Initial Setup, Minimum Viable Flask, and CDD Workflow
  2. MongoDB and Flask-MongoEngine / MongoEngine
  3. Book Class
  4. Flask Templates and CRUD
  5. Flask DebugToolbar
  6. Pagination
  7. Search, Pagination, and Session

  Summary

- **Outcomes**

  ODM CRUD Functions using MongoEngine for MongoDB for Your MS3 or On-the-Job.

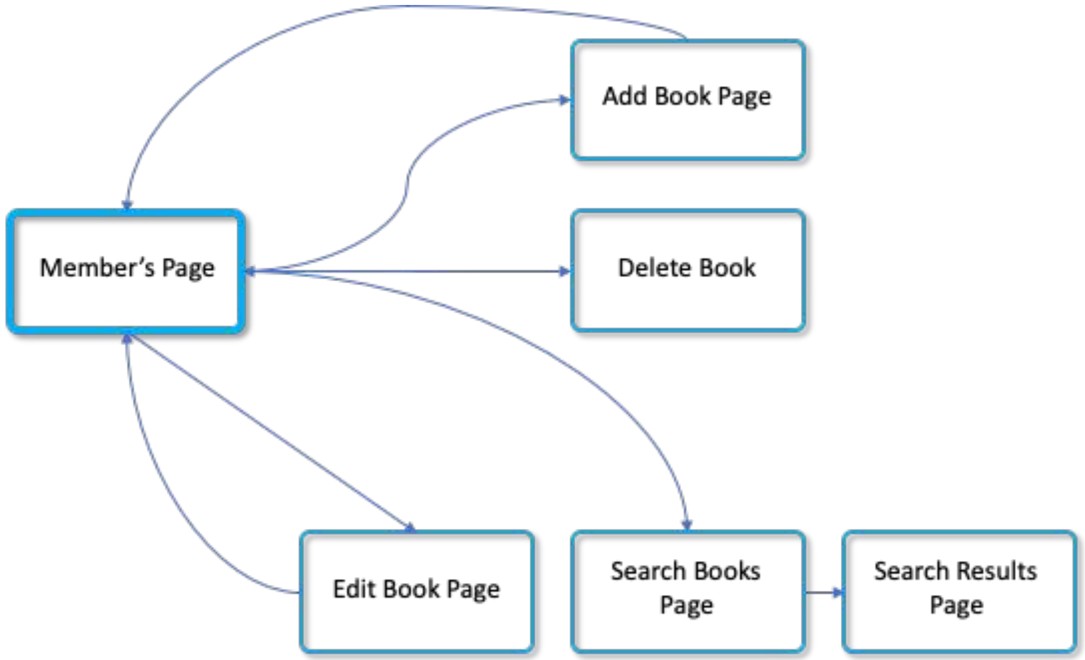
  Pagination, Flask DebugToolbar, Session Cookies.

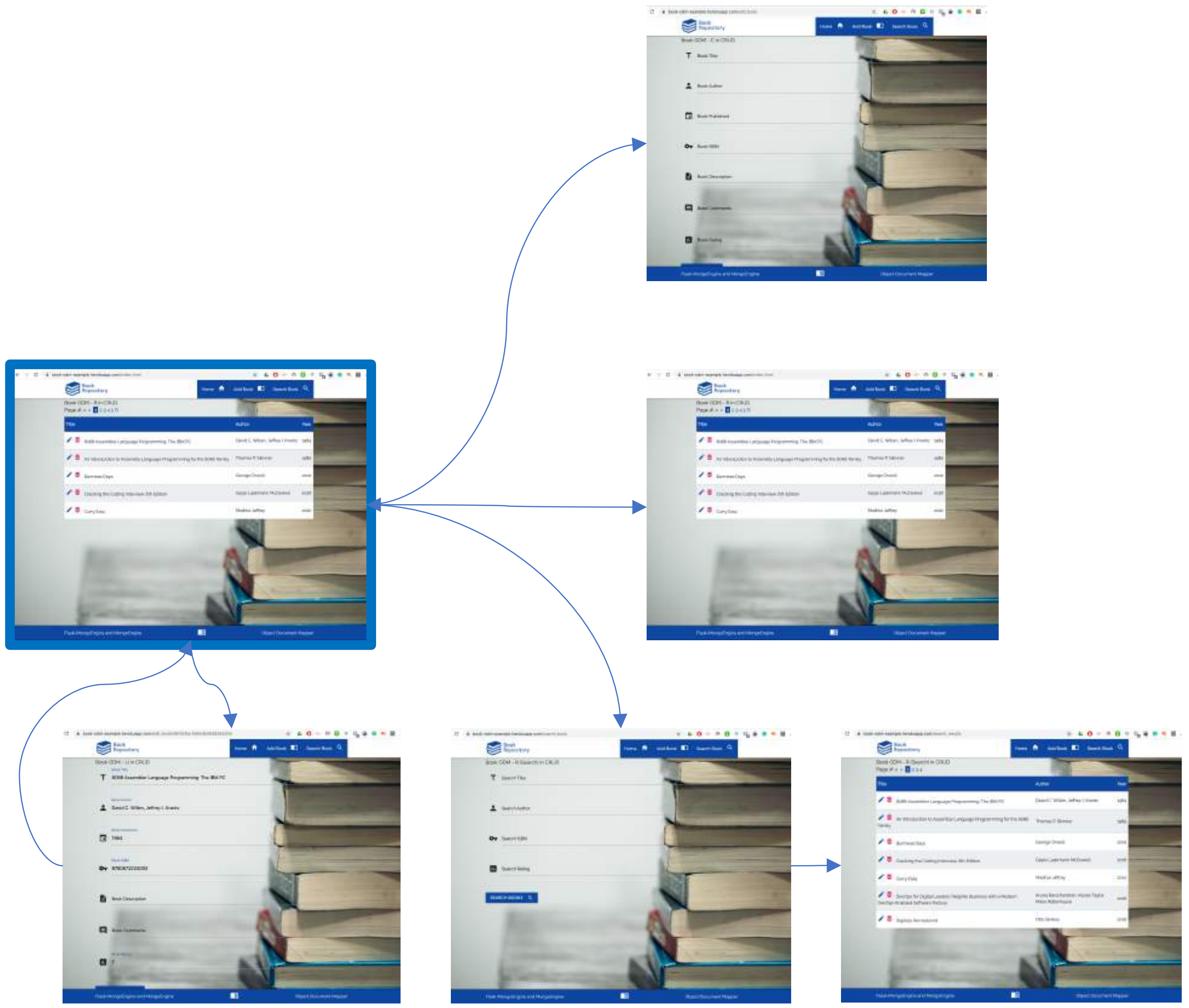
  Configure a CDD Workflow.

- **Assumptions**

  Some experience with GitHub, Visual Studio Code, and Heroku (helpful, not essential) as well as Flask and Templates.

Information Architecture and Navigation: Book ODM Example



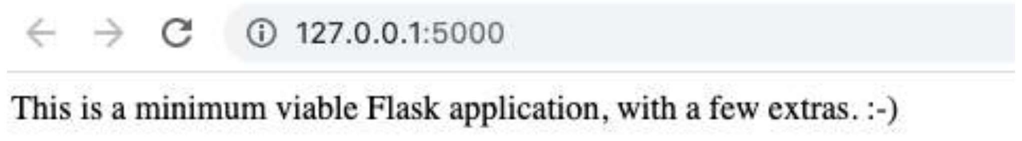


hands

on

1. Initial Setup, Minimum Viable Flask, and Continuous Development & Deployment

1. GitHub Repository using CI-full-template -> BookODMexample -> master
2. VSCode Repository: Clone Repository, create development branch
3. Create Python Virtual Environment: python3 –m venv .venv, select VSCode Python virtual environment
4. Configuration Files: .gitignore, .env (SECRET_KEY, MONGO_URI_BOOKODM, PRODUCTION, FDT, APPDEBUG)
5. Minimal Viable Flask Application: pip install pip –upgrade, pip install gunicorn, pip install Flask, pip install python-dotenv
6. Run Python File in Terminal -> SCM: Stage All Changes -> Commit All Changes -> Push
7. Heroku: Rev App, Staging, Production: pip freeze > requirements.txt, echo web: gunicorn app:app > Procfile, echo python-3.6.12 > runtime.txt, .slugignore
8. Continuous Development & Deployment: Local development -> GitHub development -> Pull Request -> Heroku Review Application -> GitHub master -> Heroku Staging -> Heroku Production



This is a minimum viable Flask application, with a few extras. :-)

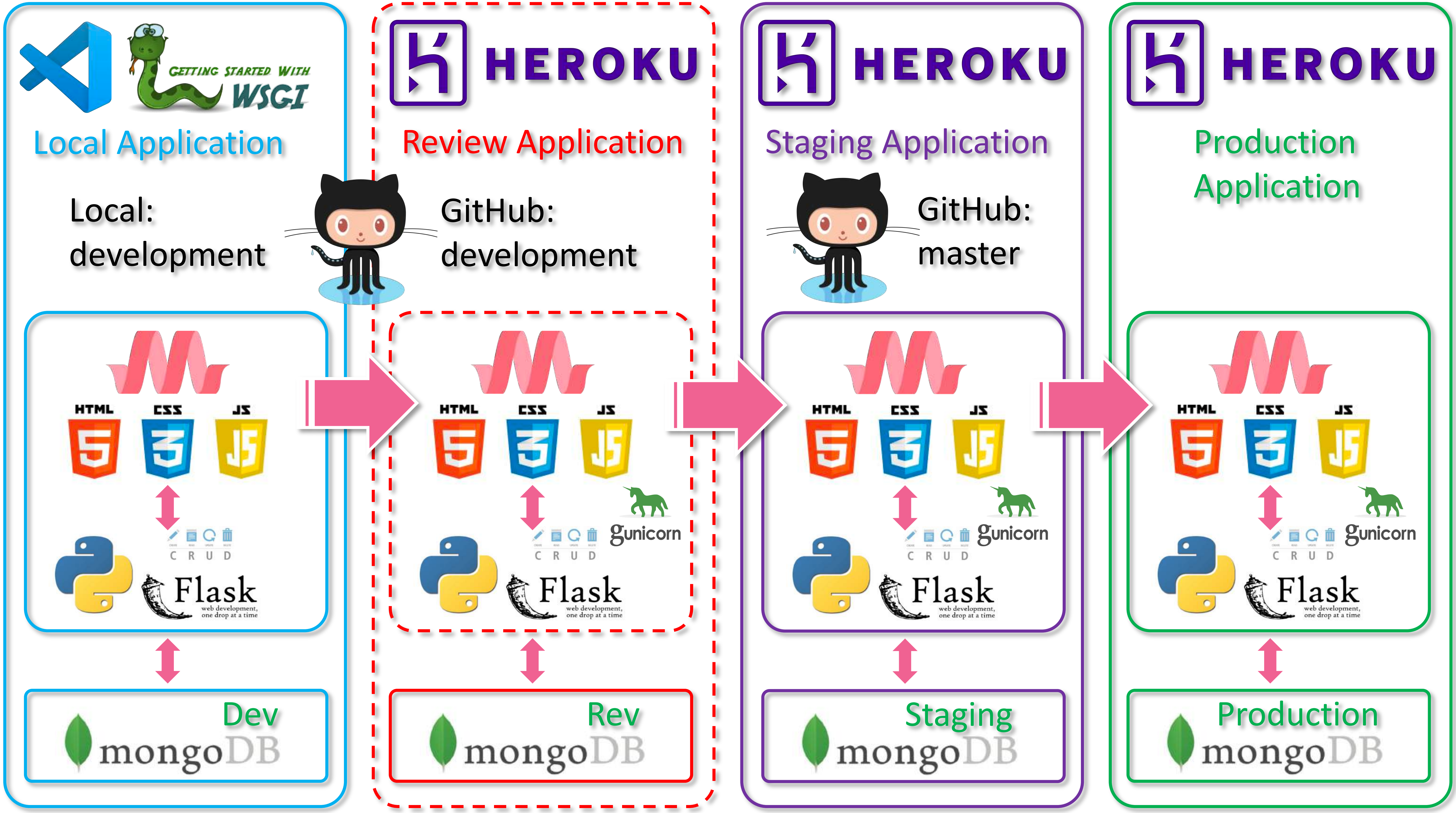
1. Use <gitpod-full-template>
2. New Repo: NaoiseGaffney / BookRepository (public)
3. Branch: master + development

4. Clone Repo: BookRepository
5. Create Python Virtual Environment (. venv/bin/activate)
6. Install Frameworks: pip install pip –upgrade → pip install gunicorn → pip install Flask → pip install python-dotenv
7. Requirements: pip freeze > requirements.txt
8. Configure Heroku Dyno: echo web: gunicorn app:app > Procfile
9. Create Minimum Flask App
10. Development: git add. → git commit –m “Initial commit” → git push (VS Code: Stage, Commit, Sync)
11. Master: git checkout master → git merge development → git push (VS Code: Pull Request)

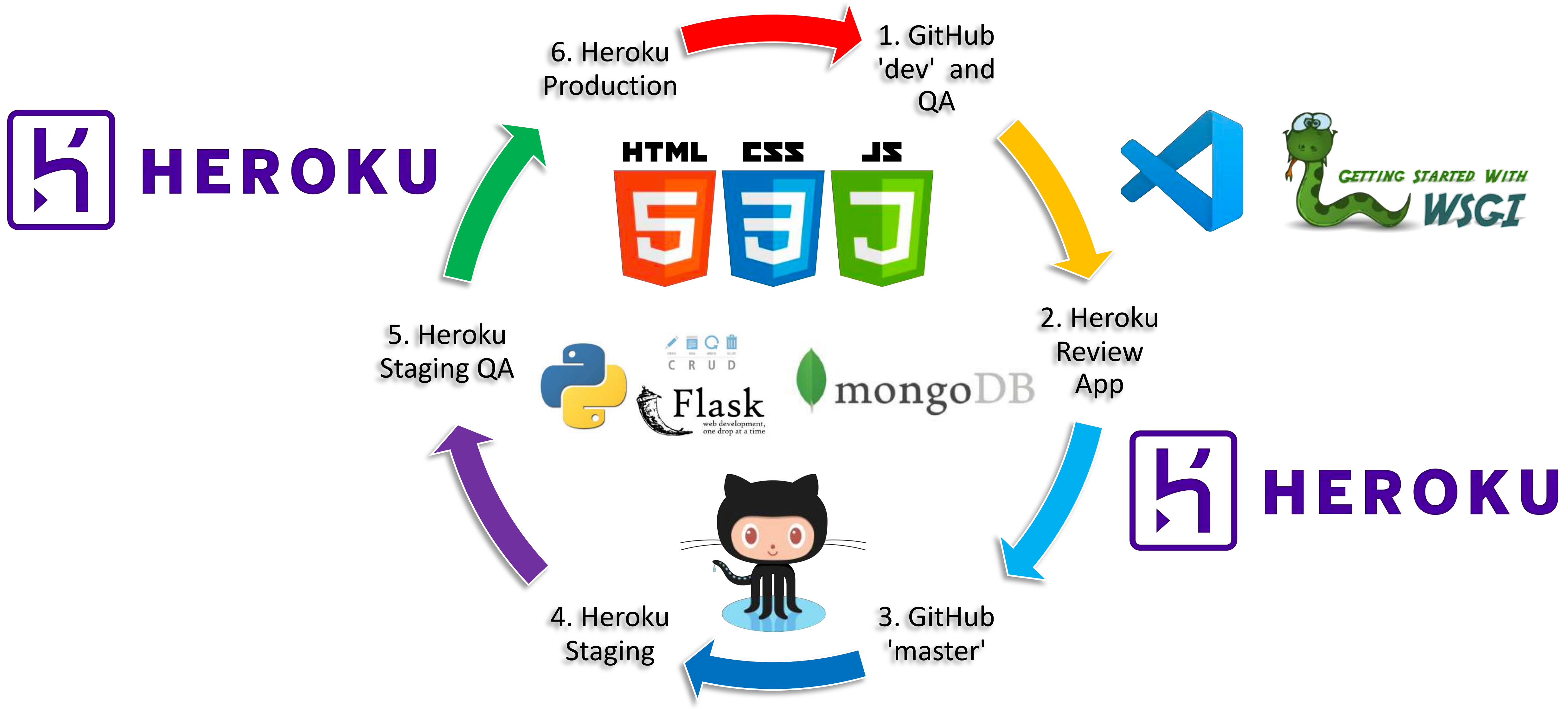
HEROKU

12. Create Pipeline: BookODMPipeline
13. Create Staging App and Link it to GitHub master, Configure Vars (staging, debug=on), AutoDeploy
14. Create Production App to Allow Promotion from Staging, Configure Vars (production, debug=off)
15. Review Apps linked to GitHub development branch
16. Environment Variables and MongoDB Databases (MONGO_URI_BOOKODM):

- VS Code: BookODMDev
- Heroku Review App: BookODMRev
- Heroku Staging App: BookODMStaging
- Heroku Production App: BookODM


Getting Started With WSGI
Local Application
Local:
development
HEROKU
Review Application
GitHub:
development
HEROKU
Staging Application
GitHub:
master
HEROKU
Production
Application
HTML
CSS
JS
C R U D
gunicorn
Flask
web development, one drop at a time
Dev
Rev
Staging
Production
mongoDB

Development & Test Cycle



Branch: development

1. VS Code: development
2. Source Control: [“Message…”] + Stage All Changes + Commit All + Push
3. Pull Request: Create Pull Request -> master + [“Message…”] -> Heroku Review App.

HEROKU

4. GitHub development → Heroku Review App (Unit Test!)

Branch: development

5. VS Code: development
6. Pull Request: Merge Commit (from development to master)

9. VS Code: master → Source Control: Pull

HEROKU

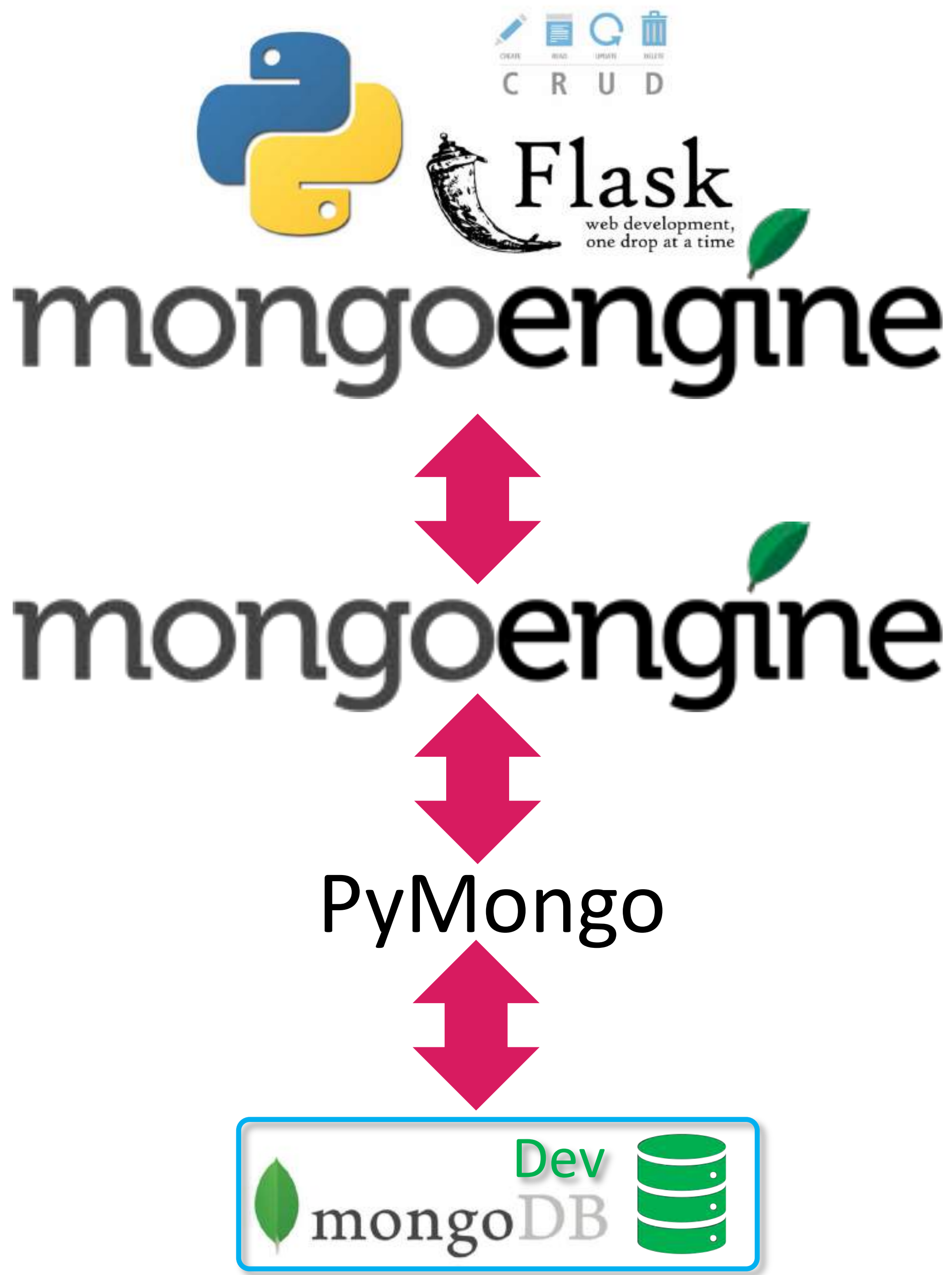
7. GitHub master → Heroku Staging App Deployed Automatically (Integration / System Test!)
8. Heroku Staging App Manually Promoted to Production

hands
on

2. MongoDB with Flask-MongoEngine and MongoEngine

- “Hands-Off”!
  - Define URI
  - Database Creation
  - Connect, Disconnect to Database
  - Create Collections and Documents
- pip install flask-mongoengine
- pip install dnspython
- MongoDB URI in Python, ’.env’, and Heroku

```
# ‘.env’ and Heroku Variables
MONGO_URI_BOOKODM =
"mongodb+srv://mdb_c_root:**********
@mdbcluster-
vhvci.mongodb.net/BookODM?retryWrite
s=true&w=majority"

# Flask-MongoEngine settings
MONGO_URI_BOOKODM =
os.environ.get("MONGO_URI_BOOKODM")
app.config["MONGODB_SETTINGS"] = {
'host': MONGO_URI_BOOKODM
}
db = MongoEngine(app)
```



Connection Management
Custom Queryset
Paginate
Session

Object Document Mapper
C: Collection.save()
R: Collection.objects(), Collection.objects.get()
U: Collection.update(**dict)
D: Collection.delete()

PyMongo is a Native
MongoDB Driver

noSQL Database

hands
on

3. Define the Book Class (Collection)

- Fields
  - Title, Author, Year, ISBN, Description, Comments, Rating, Genre, Private
- Field Parameters
- Meta
  - auto_create_index
  - index_background
  - indexes
  - ordering

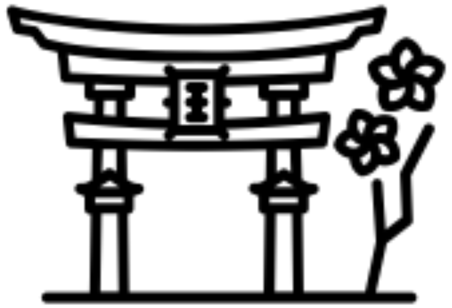
```
class Book(db.Document):
      title = db.StringField(default="", maxlength=250)
      author = db.StringField(default="", maxlength=250)
      year = db.IntField(maxlength=4)
      ISBN = db.IntField(maxlength=13)
      short_description = db.StringField(default="", maxlength=2000)
      user = db.StringField(required=True, default="BookODM")
      creation_date = db.DateTimeField(default=datetime.datetime.now)
      comments = db.StringField(default="", maxlength=3500)
      rating = db.IntField(choices=[1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
      genre = db.StringField(default="")
      private_view = db.StringField(default="off")
      book_thumbnail = db.StringField(default="")

      meta = {
            "auto_create_index": True,
            "index_background": True,
            "indexes": ["title"],
            "ordering": ["title"]
}
```

hands
on

4. Flask Templates and CRUD

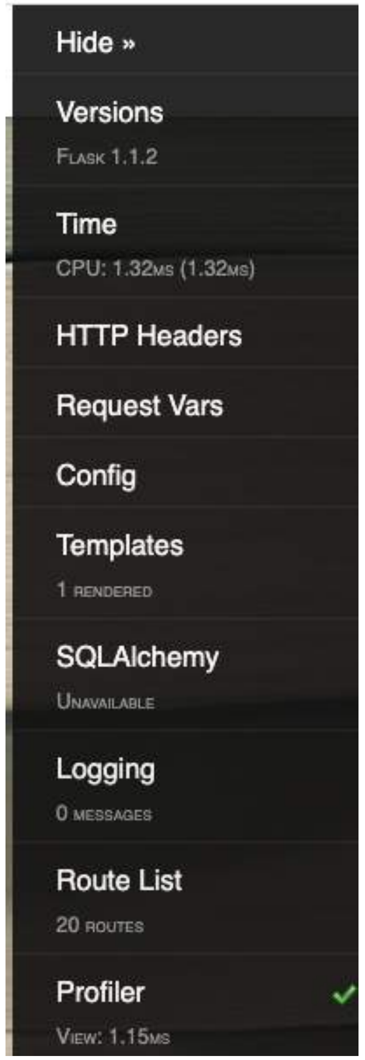
- Base.html
  - Navigation Bar
  - Flash Messages
  - Fixed Footer
- Index.html
  - Home_page(page=1)
  - Edit_book(book_id)
  - Delete_book(book_id)
- Add_book.html
  - Add_book()
  - Save_book()
- Edit_book.html
  - Edit_book(book_id)
  - Update_book(book_id)

- Create: Book.save()
- Read: Book.objects(), Book.objects.get(id=id)
- Update: Book.update(**dictionary)
- Delete: Book.delete()



hands
on

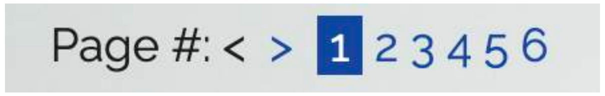
5. Flask DebugToolbar


Hide »
Versions
Flask 1.1.2
Time
CPU: 1.32ms (1.32ms)
HTTP Headers
Request Vars
Config
Templates
1 rendered
SQLAlchemy
Unavailable
Logging
0 messages
Route List
20 routes
Profiler ✓
View: 1.15ms

```python
# FDT Extension Load IF FDT == ON.
if os.environ.get("FDT") == "ON":
from flask_debugtoolbar import
DebugToolbarExtension

# FDT Extension app.debug = True IF
FDT == ON.
if os.environ.get("FDT") == "ON":
app.debug = True

# FDT Extension App IF FDT == ON.
if os.environ.get("FDT") == "ON":
toolbar = DebugToolbarExtension(app)
```

hands
on

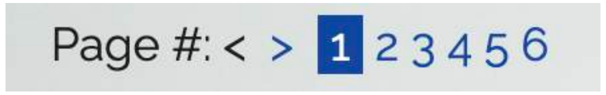
6. Pagination


Page #: < > 1 2 3 4 5 6

```
@app.route("/index.html/<int:page>")
def home_page(page=1):

books_pagination =
Book.objects.paginate(page=page, per_page=5)

return render_template("index.html",
books_pagination=books_pagination,
page_prev=(page - 1), page_next=(page + 1))
```

hands
on

7. Search, Pagination, and Session


Page #: < > 1 2 3 4 5 6

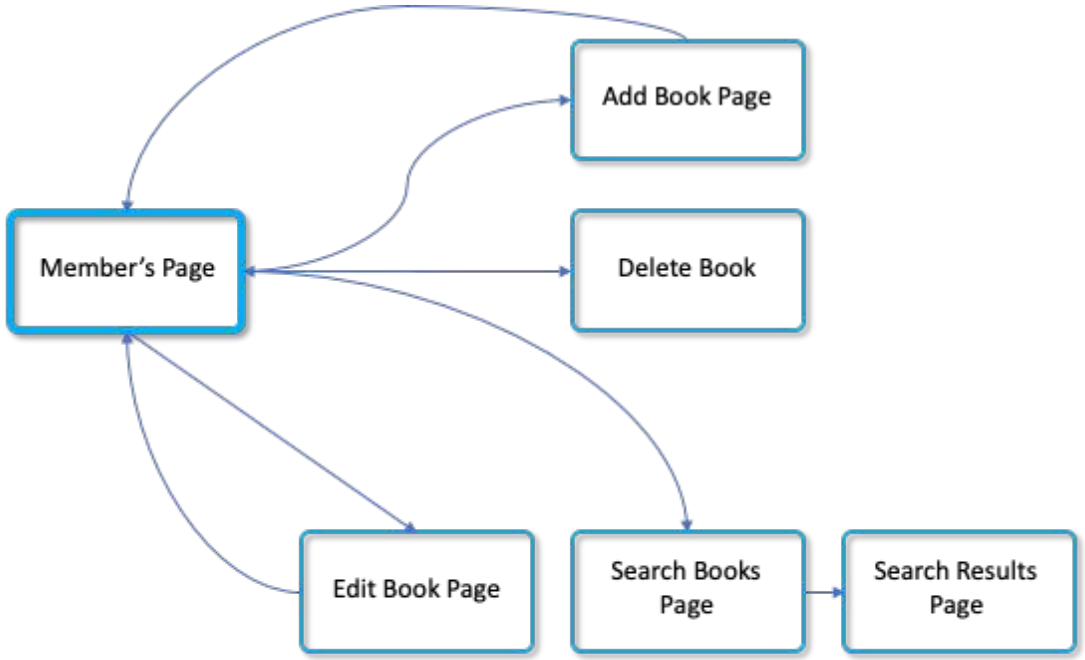
```
fields = {
    "title": request.form.get("title"),
    "author": request.form.get("author"),
    "year": request.form.get("year"),
    "ISBN": request.form.get("isbn"),
    "short_description":
    request.form.get("short_description"),
    "comments":
    request.form.get("comments"),
    "rating": request.form.get("rating"),
    "genre": request.form.get("genre"),
    "private_view":
    request.form.get("private_view")
}
session["fields"] = fields
return
redirect(url_for("search_results"))
```

```
book_query_results =
Book.objects.filter(title__icontains=form
_title, author__icontains=form_author,
rating__gte=form_rating,
private_view="off").order_by("+title",
"+author", "-rating")

book_query_results =
book_query_results.paginate(page=page,
per_page=7)

return
render_template("search_results.html",
book_query_results=book_query_results,
page_prev=(page - 1), page_next=(page +
1))
```

hands
on

Summary

Information Architecture and Navigation: Book ODM Example


Add Book Page
Member's Page
Delete Book
Edit Book Page
Search Books Page
Search Results Page