Project Final Report

Prajesh Praveen Anchalia
UNI: pp2546
Email: pp2546@columbia.edu

Comparison of the Python based Flask and JavaScript based MEAN Framework for Web Application Development

At the end of this project, I have delivered two versions of the same application on Python Flask and MEAN Stack with the intention to compare the performance of the two popular web frameworks. Both the systems have used Bootstrap as the front-end technology and monogDB as the database.

The two Web Frameworks use the following tech stack.

- Python Flask
  -- Werkzeug
  -- Jinja2
  -- MongoDB
- Mean Stack based on Javascript
  -- NodeJS
  -- ExpressJS
  -- AngularJS
  -- MongoDB

For performance management I used AppDynamics Trial Version :

https://www.appdynamics.com/

Application Outline:

- The application that I intend to build is an inventory management for a Vehicle Showroom.

- There a three access layers to the application i.e Admin, Employee and Customer.
- Admin: This account is the master of the application and has control over the inventory and employees. He can add/delete and view the inventory and also create accounts for Employees. He also has the option to manage his own account details on a personalized page.
- Employee: This account can view the inventory list and has two sets of prices visible to him Maximum retail price (MSRP) and least offer price (LOP) based on which the employee can showcase and offer priced to the customer taking the MSRP and LOP as guidelines. The employee can also manage his own account details on a personalized page.
- Customer: This account can view the list of inventory available at the showroom and has only access to the MSRP of the vehicle and not on the LOP. He also has access to manage his own account details on a personalized page
- The application will have a Mongo database for data storage
- A Bootstrap frontend
- Backend with Flask and Mean.
- This gives a good opportunity to evaluate both the models based on the above-mentioned criteria.
- In addition to those, the application can also be compared in terms of addressing security issues like injections, URL hacks and session managements.
-

Deliverables:

I have uploaded the code to a public GitHub repository:

https://github.com/prajeshanchalia/coms6156project.git

Flask App:

Flask and the required packages are based on Python and it can be installed using:

- sudo pip install Flask
- sudo pip install Flask-WTF
- sudo pip install flask-bootstrap

The instruction to run the apps is mentioned in the Readme file in the Repository.

Once flask is setup, run the flask app, by executing the app.py file and the application runs on port 5000

Screenshots:

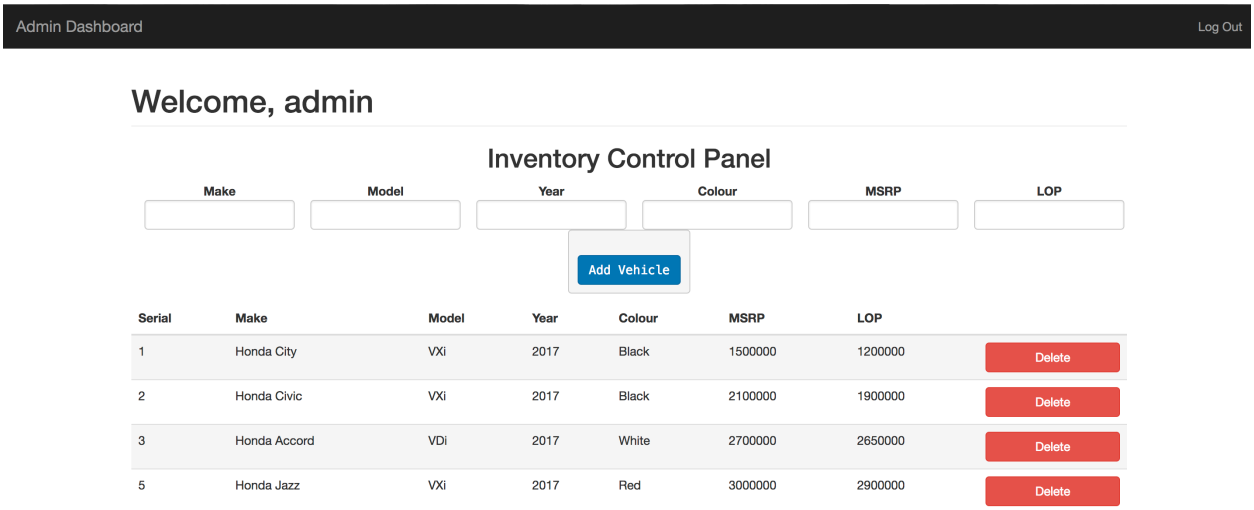

Figure: Admin Inventory Control Panel

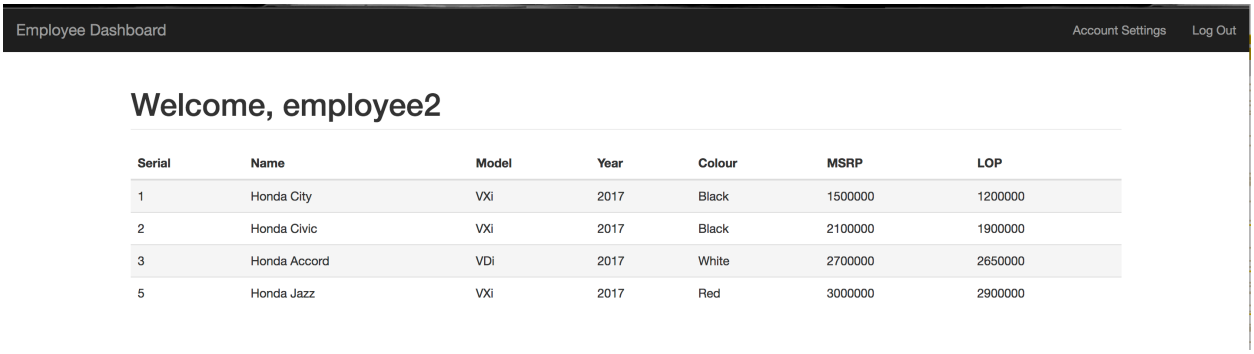

Figure: Employee Dashboard

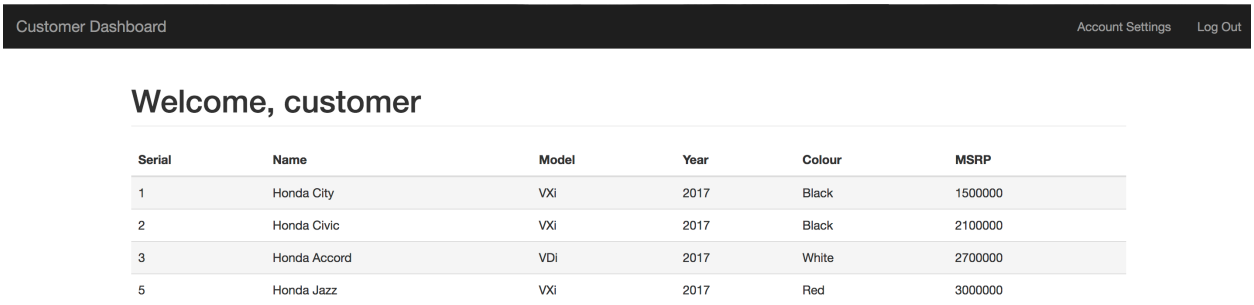

Figure: Customer Dashboard

Mean App:

To install MEAN execute the following commands:

- npm install -g bower
- npm install -g grunt-cli
- npm install -g yo
- npm install -g generator-meanjs@0.1.12
- sudo npm install -g mean-cli

Once the stack is set up you can run the application using node server and the app runs on port 3000.

Screenshots

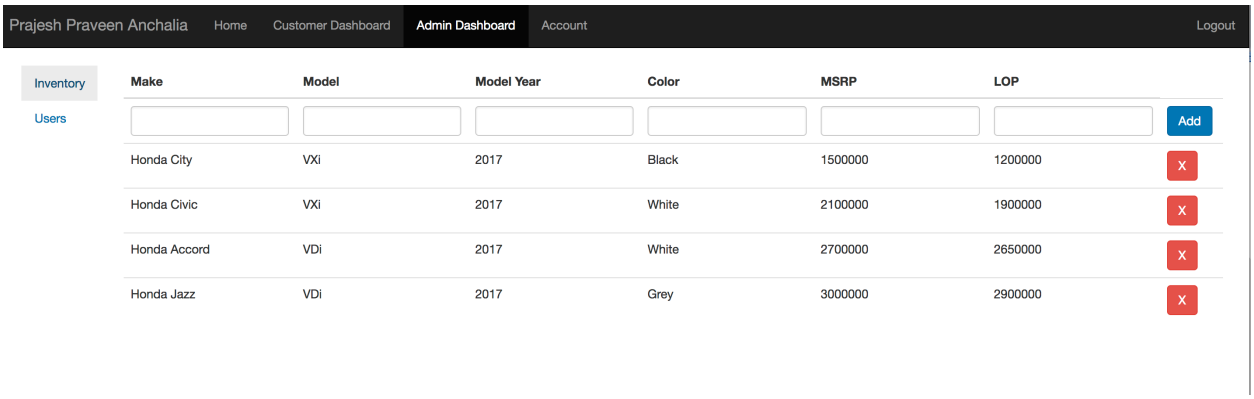

Figure: Admin Inventory Control Panel

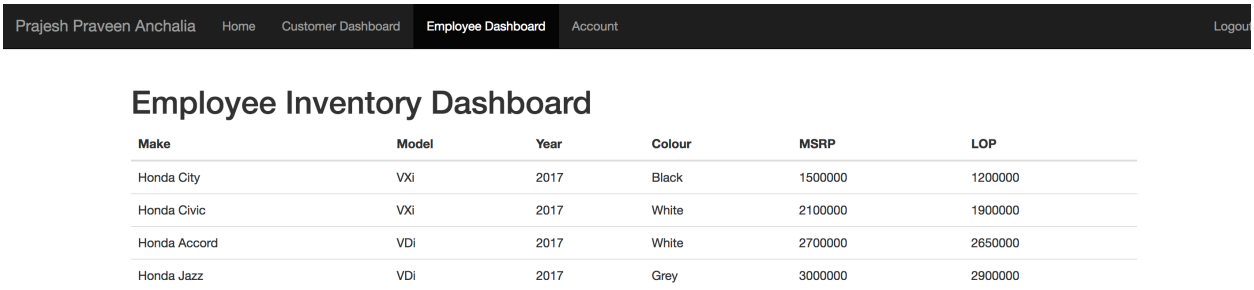

Figure: Employee Dashboard

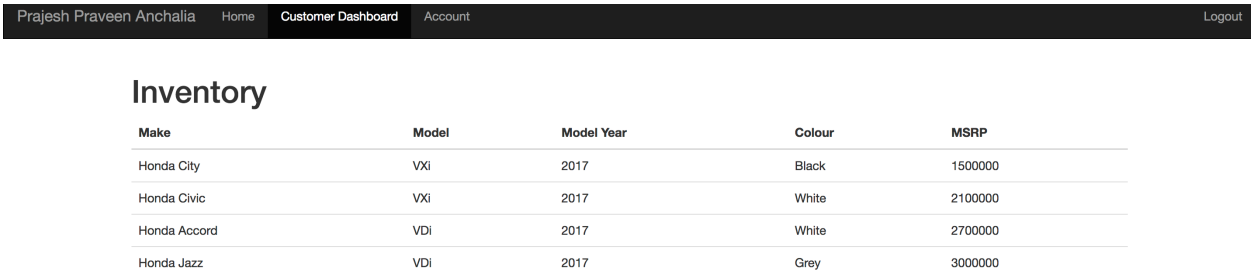

Figure: Customer Dashboard

Measuring performance of the frameworks:

- I have used AppDynamics and New-Relics to measure the performance of the application frameworks.
- Each of the applications when put under test has the same sequence of events taking place.
- The sequence of events was, the admin logging into the app and creating a new entry for inventory and employee and logging out. Next, the newly created employee will log in, change the password and view the inventory list. Lastly, the customer would sign up and view the inventory list and log out.
- The AppDynamics daemon would capture the framework activity.
- The top command would monitor the system load.
- The tcpdump and netstat would give you the activity at ports.
- All this information would be recorded for each app and compared against one and the other.

Mean Performance Chart:

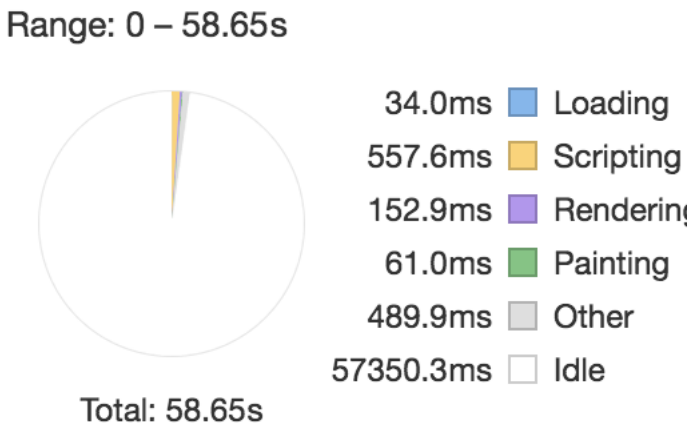

Figure: Mean AppDynamics Metrics

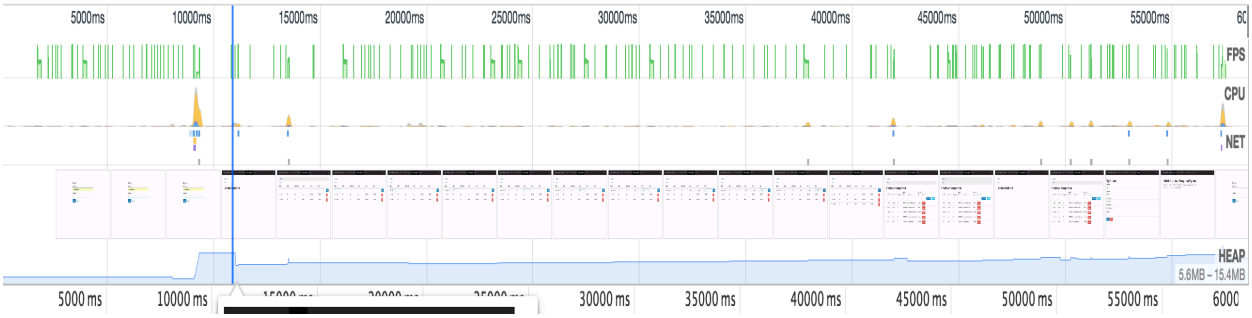

Figure: Mean AppDynamics Timeline

Flask Performance Chart:

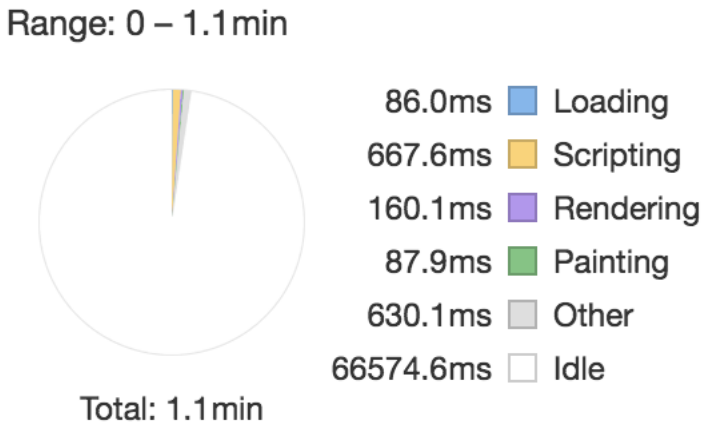

Figure: Flask AppDynamics Metrics

Figure: Flask AppDynamics Timeline

Results:

Flask out performs The Mean Framework performs better on responsiveness and speed but is slightly behind on the system resource consumption and ease of development. MEAN uses multiple platforms, though all based on JavaScript have their own designs, hence making it a lot more complex. Also, Flask is a lot richer in packages, easier to develop, active user and development base and more popular. From experience in this project, it uses a simple MVC architecture and there is no vast distinction in the designs of these components compared to MEAN, closely knit and less complex. Bottom-line performance=MEAN, efficiency and ease of development=Flask

Live Demo was pushed to 27th April due to lack of time on the 25th :

30 Minute detailed Demo: https://youtu.be/LKK1gJB9xO4

15 Minute Demo: https://youtu.be/n34hr2Bvl2c

Learning from the Project:

I developed the applications individually working as a full stack developer and had no prior development experience in Flask before this project and had a small exposure to MEAN. From this project I learnt to develop apps using Flask and MEAN. Performance measurement and analysis was completely new for me. Profiling and Timing are two important aspects of performance measurements and I used AppDynamics for the first time to gauge the performance of the frameworks. It was a fascinating journey from development to performance monitoring and always a great learning experience. There were a few problems that I faced while building the apps, the whole debug journey and happiness to get things working was beyond all. I would also like to offer my applications as a starting point for other students to build on in their 4516 and 6156 courses. All in all a great learning experience and fascinating journey from start to finish.