**Bazer Book store**

**System Component:-**

* **Servers:-**
* search server1 (<http://bara111.pythonanywhere.com/>)
* search server2 ([http://bara1111.pythonanywhere.com/](http://bara111.pythonanywhere.com/))
* buy server1 ([http://ambara059.pythonanywhere.com/](http://bara111.pythonanywhere.com/))
* buy server2 ([http://ambara056.pythonanywhere.com/](http://bara111.pythonanywhere.com/))

5. front-end node

* **FrameWorks:-**

1.Flask (python)

* **Database:-**

file-system

**System Description:-**

the user starts using the system with front end node the user uses search field to get the book by id or by category the first time user runs a search it caches the response of the request in the local storage of browser with TTL of 10 minutes the browser removes the cache from the system.

the system handles the Round-Robin technique by saving the server id the local storage the turn request to the next server for the search and order servers

the system handle consistency, the system must wait for the server to send to all other replicas to update the database to remain consistent in all replicants for each buy request the system needs to send n-1 of requests to need for n replicants and to check all the server are updated to push invalidate to the client to remove the cache from the local storage of the browser.

**How to run:-**

1.before running the client must install this extension to google chrome



2.run the HTML page in the path

( BookStore-master\webapp\homePage\index.html)

Search by category gets you all the books in category

3.buy a book with id 100 of DOS type

then try to search for it again

**System improvement:-**

System improvement to improve the overall system performance we must buy the upgraded version of the server get the server as close as possible to the users, move the database to SQL to reduce the complexity of the code when adding more category in the future, make a server for each category this way to reduce the overhead of too much request from other types using AWS will improve the system EC2+S3 storage

.1

|  |  |  |  |
| --- | --- | --- | --- |
| avg | search | buy |  |
| 426ms | 378ms | 474ms | without caching |
| 265ms | 56ms | 474ms | with caching |
|  |  |  |  |



2. for the overhead of caching consistency the system must wait for the server to send to all other replicas to update the database to remain consistent in all replicants for each buy request the system needs to send n-1 of requests to need for n replicants and to check all the server are updated to push invalidate to the client to remove the cache from the local storage of the browser.