Tuijin Jishu/Journal of Propulsion Technology
ISSN: 1001-4055
Vol. 44 No. 5 (2023)

SVMVC: A Refinement to classical MVC for enhancing the Performance of web applications

Rajneesh Chaturvedi

Sr. Assistant Professor, CS & IT Dept., IIS (Deemed to be University) Jaipur (India)

Prof. Swati V. Chande

Professor and Head of Computer Science department, International school & informatics & Management, Jaipur (India)

Dr. Amita Sharma

Assistant Professor (Selection Grade), IIS (Deemed to be University) Jaipur (India)

Abstract:- Web application architectures are established design patterns employed to structure the design of web applications. Typically, web application architecture is divided into distinct layers and components, each catering to various quality enhancement requirements. These design patterns are crucial for creating web applications that are scalable, maintainable, and well-organized. Among the most renowned and widely used design patterns for web applications is the Model-View-Controller (MVC) pattern. The MVC architecture's inherent attributes, such as reusability, testability, scalability, and separation of concerns, have played a pivotal role in its widespread adoption.

As performance stands out as a prominent quality indicator for web applications, this paper introduces a novel refinement to the MVC architecture, with a primary focus on enhancing the performance of web applications.

Keywords: MVC, Web application architecture, Performance, Simulation environment, Response time, Throughput.

1. Introduction

The Internet and its applications have evolved rapidly in the last few years. It is capable of providing different kinds of services through dynamic web applications. Web applications have become more interactive and flexible to address different areas related to the needs of the society and business. Enterprise solutions have become too complex in terms of fulfilment of different kinds of customers requirements. Banking, e-commerce, communication (mails, chat), stock trading, social networking are the areas where web applications are providing services to a large number of users. In order to provide solutions for such complex problems, web applications have become too lengthy, complex, distributed and divided into different layers and components.[1]

Web application design is based on an architectural framework design so as to improve functioning of the web applications and perform better partitioning of different components or parts of application according to the architecture of that framework.[2][3] Effective architecture is necessary for structured development and increased maintainability. There are different views of architecture which can be relevant to the understanding of web application structure.[3]

- Physical view :It provides physical abstraction of client – server architecture; 2 tier , 3 tier , N tier architecture
- Logical view: It provides a high level of abstraction of different parts of application.

The Framework design pattern represents a logical view of application.

Logical views can be generated in the design and implementation phases of application development. The main objective behind this is to decompose the system into groups/layers of different components to achieve various kinds of decompositions.[3][4] This decomposition may further be used for fulfilling various web software development purposes. MVC design pattern is a majorly accepted design pattern in the industry for application development.

2. MVC –Model View Controller

MVC was conceived as a general solution to the problem of users controlling a large and complex data set. The hardest part was to hit upon good names for the different architectural components. MVC has three components Model, View and Controller for different kinds of tasks.[6]

Model: A model represents an application's data and contains the logic for accessing and manipulating that data. Any data that is part of the persistent state of the application should reside in the model objects. Model services are accessed by the controller for either querying or generating a change in the model state. The model notifies the view when change of state occurs in the model.

View: The view is responsible for rendering the state of the model. The presentation semantics are encapsulated within the view therefore, model data can be adapted for different kinds of clients. The view modifies itself when a change in the model is communicated to the view. A view forwards the user's input to the controller.

Controller: The controller is responsible for interpreting and translating the user input into actions to be performed by the model. The controller is responsible for selecting the next view based on user input and the outcome of model operations.

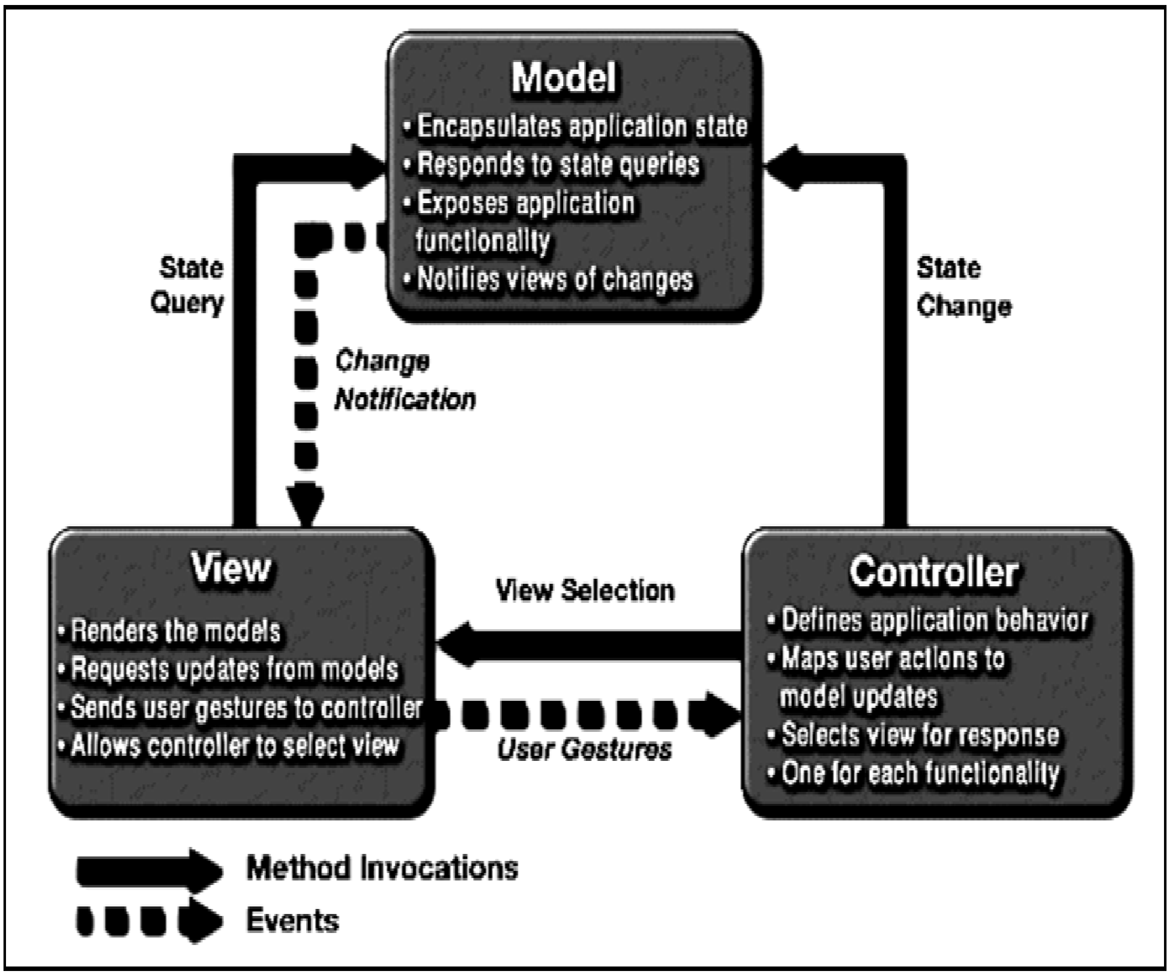
Model
• Encapsulates application state
• Responds to state queries
• Exposes application functionality
• Notifies views of changes
State Query
State Change
Change Notification
View
• Renders the models
• Requests updates from models
• Sends user gestures to controller
• Allows controller to select view
View Selection
User Gestures
Controller
• Defines application behavior
• Maps user actions to model updates
• Selects view for response
• One for each functionality
Method Invocations
Events

Figure 1: MVC Architecture framework [6]

It is concerned with making adaptations to the MVC pattern in order to develop applications in a modular way and make layers fulfil their roles independently of the other layers. The primary benefit of the MVC design pattern is a clear separation of concerns and the resulting modularity. The design isolates user interface presentation from user input handling, and isolates both of these from application state and transaction processing. This makes it possible to modify or replace one component without needing to modify or even understand the others. It also facilitates extensibility by making it possible to add a view/controller pair for a new interface medium, or to add new functionality to the model independently of the other components.

3. Performance Parameters:

1. **Response Time**: Response time is the time taken by the web server to respond to a request. This includes the time taken to process the request, generate the response, and send it back to the client.[5]

2. **Latency**: Latency is the time taken for a request to travel from the client to the server and back again. This includes network latency, server processing time, and client processing time.[5]

3. **Throughput**: Throughput is the number of requests that can be processed by the server per unit of time. It is often measured in requests per second (RPS).[5][9]

4. **Concurrent Users**: Concurrent users refer to the number of users accessing the application at the same time. This is important because the performance of the application can degrade as the number of concurrent user's increases. [9]

5. **Error Rate**: Error rate is the number of errors encountered by the application during a given period of time. This includes server errors, client errors, and network errors [11].

4. Proposed Refinement in MVC:

4.1 Background

As the web application size and number of requests increases, the response time may be negatively affected. Furthermore, if a large application is reloaded onto the server, response time can increase. In order to address this issue, an improvement/refinement in MVC architecture has been proposed, which separates the validation from the model component [10].

Validations are a critical part of any web application where user data must be collected and verified in relation to the business logic [11].

While designing web applications we may use two kinds of validation techniques: client-side and server-side.

a) Client-side validation is used to initially validate data at the client-side typically done using scripting languages.

b) Server-side validation is used to validate data received from the client-side using server-side programming language. Server-side validation is a necessary component of the business logic layer of web application architecture.

In this paper we propose a refinement of the MVC architecture that enables the application to be loaded in segments or only the necessary part of the application to be loaded.

4.2 Server side Validation Model View Controller (SVMVC):

The Server-Side Model View Controller (SVMVC) is a more sophisticated version of the classical Model View Controller (MVC) that focuses on improving the performance of the "Model" layer in the MVC framework. The model layer is responsible for handling the business logic and application layer of a web application. It contains all the server-side components necessary to process all client-side requests.

Since the model layer has various components for server-side processing, it can be further refined according to their different functions. SVMVC is a refinement of the model layer that introduces server-side validation as a new component. The validation component manages all server-side validation processing before redirecting the requests to other components in the model layer. Figure 2 provides an overview of SVMVC's fundamental architecture. This refinement reduces application latency, enhancing the performance of web applications. The proposed refinement was tested for performance on a developed web application, and the results confirmed its effectiveness.

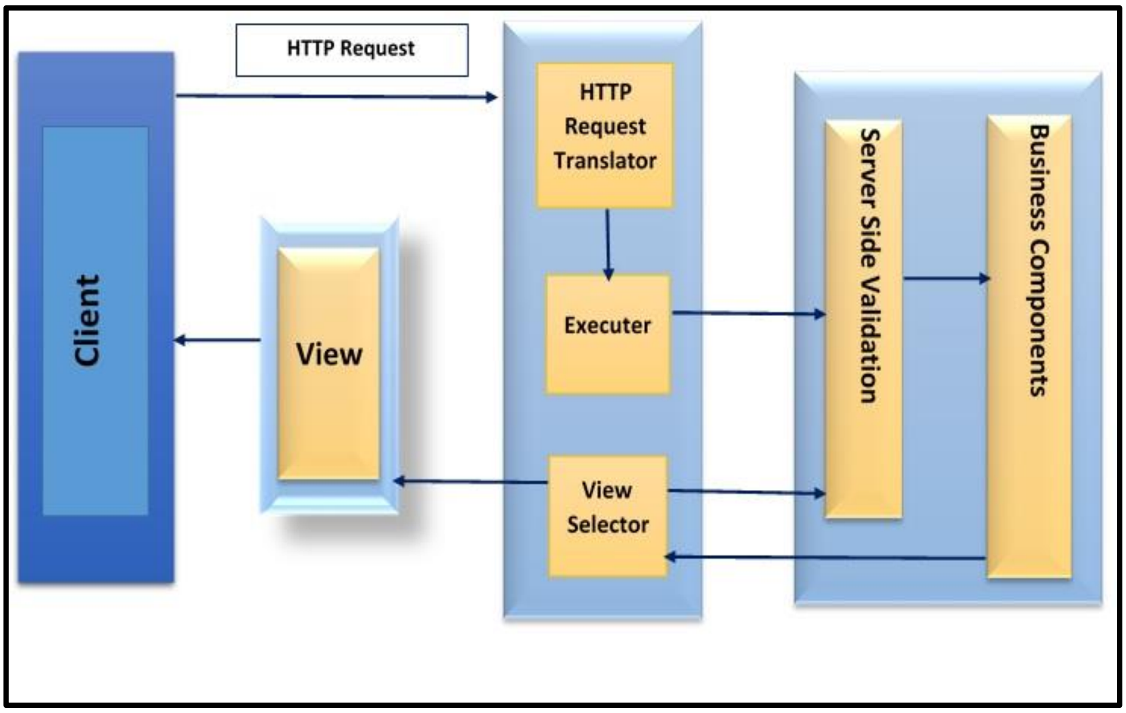

Figure 2: SVMVC Model (Refined)

5. Results Discussion and comparative analysis of MVC and SVMVC performance:

5.1 Experimental setup

The objective of this experiment was to study the behavior of classical MVC and the proposed SVMVC architecture. In order to meet this objective two web applications (web application-1 and web application-2) were designed. Following hardware and software used to develop web application and performance testing:

Hardware: Core i5, 4 GB DDR4

Software : Java Spring , Eclipse , Jmeter2.0

The first experiment was carried out in a simulated environment utilising the performance testing tool Jmeter for Web Application1, developed using classical MVC.

The second experiment was carried out in a simulated environment utilising the performance testing tool Jmeter for Web Application2, developed using the proposed SVMVC architecture. Both the experiments concentrate on the performance analysis using several performance metrics, such as response time, throughput, average elapsed time, error percentage, and network throughput (KB/sec).

The experiments were centered on load generation for Web Application 1 and Web Application 2, taking ramp up count as 0 for concurrent users. In order to notice the unique outcomes for each group of concurrent users in the range (100-1000) with a difference of 100, loop count has been set to 1 for all trials in both experiments. In a simulation environment, the parameter Ramp up represents the overall ramp up time utilised for user thread creation.

The list of experiments are:

- Experiment1 : Performance analysis of business layer of Web Application-1 based on MVC architecture at Ramp up=0
- Experiment2: Performance analysis of business layer of Web Application-2 based on SVMVC architecture at Ramp up=0

5.2 Experiment1 Results & Inference:

Table 1 and table 2 show the results obtained from the experiment 1 performed on web application-1.

Table 1: Performance analysis of business layer of Web Application1 based on MVC architecture at Ramp up=0

Ramp up=0							
Users	**Throughput (s)**	**Execution Time(sec.)**	**Average Response Time (ms)**	**Min. Response Time (ms)**	**Max. Response Time (ms)**	**Error %**	**Network Throughput (KB/sec)**
100	102	2	391	7	965	0	327.9
200	101.2	2	842	153	1493	20	306.2
300	101.7	2	717	6	1992	27.67	300.4
400	110.5	3	1099	20	2083	68.75	284

500	103.6	2	1045	9	2771	58	276.1
600	122	2	1024	12	2098	87.67	291.8
700	127.9	3	1153	22	2698	49.14	349
800	134.5	3	1304	17	2501	87.12	322.2
900	148.6	2	9 89	6	2359	80.89	364.9
1000	128.8	2	1026	4	2263	58.30	342.9

Table 2: Average Performance Analysis in Experiment 1

Avg.Users	**Avg. Throughput (s)**	**Avg. Execution Time(ms)**	**Avg. Response Time (ms)**	**Avg. Min. Response Time** (ms)	**Avg. Max. Response Time (ms)**	**Avg. Error %**	**Avg. Network Throughput (KB/sec)**
550	118.08	2.3	959	25.6	2122.3	53.754	316.54

Inference: Due to Ramp up = 0 time, concurrent threads are created simultaneously, wherein Jmeter adds each concurrent thread in accordance with its own policy. According to the results, throughput gradually increases as the number of concurrent users rises from 100 to 1000. Network throughput (data downloaded from the server) is subsequently increased or decreased depending on the error percentage; when errors were caused by the generation of requests, network throughput decreased because fewer bytes were downloaded for valid requests. Given that the business layer components are now loading and running, the average response time turns out to be 2122.3ms for Ramp up=0.

5.3 Experiment2 Results & Inference:

Table 3 and table 4 show the results obtained from the experiment 2 performed on web application-2.

Table 3: Performance analysis of business layer of Web Application1 based on SVMVC architecture at Ramp up=0

Ramp up=0							
Users	**Throughput (s)**	**Execution Time(sec.)**	**Average Response Time (ms)**	**Min. Response Time (ms)**	**Max. Response Time (ms)**	**Error %**	**Network Throughput (KB/sec)**
100	102.9	2	213	6	564	0	547
200	112.5	3	493	23	1199	12.50	555.3
300	121.5	2	314	5	1006	1	642.3
400	117.7	2	718	4	2082	36.5	494.8
500	124.8	2	911	9	1528	85.20	340.3
600	148.3	2	687	4	2064	27.67	663.5
700	152.3	2	906	6	2261	64.29	512.1

800	154.9	2	471	7	1706	23.25	714.3
900	148.1	2	1134	4	2695	71.56	465.1
1000	178.7	2	810	3	1587	75	542.5

Table 4: Average Performance analysis of Experiment 2

Avg.Users	**Avg. Throughput (s)**	**Avg. Execution Time(ms)**	**Avg. Response Time (ms)**	**Avg. Min. Response Time** (ms)	**Avg. Max. Response Time (ms)**	**Avg. Error %**	**Avg. Network Throughput (KB/sec)**
550	136.17	2.1	665.7	7.1	1669.2	39.697	547.72

Inference: Due to Ramp up = 0 time, concurrent threads are created simultaneously, wherein Jmeter adds each concurrent thread in accordance with its own policy. According to the results, throughput gradually increases as the number of concurrent users rises from 100 to 1000. Network throughput (data downloaded from the server) is subsequently increased or decreased depending on the error percentage; when errors were caused by the generation of requests, network throughput decreased because fewer bytes were downloaded for valid requests.

5.6 Comparative Analysis of MVC and proposed SVMVC

Since the response time and throughput act as the base indicators for evaluating the performance of the web application, hence we evaluated our web application1 (based on MVC) and web application2 (based on SVMVC) using the Jmeter tool. The ramp up was set as 0 for 100,200,300,400,500,600,700,800,900 and 1000 concurrent users. As depicted in the following figures the response time and throughput are improving with increase in the number of concurrent users in case of SVMVC.

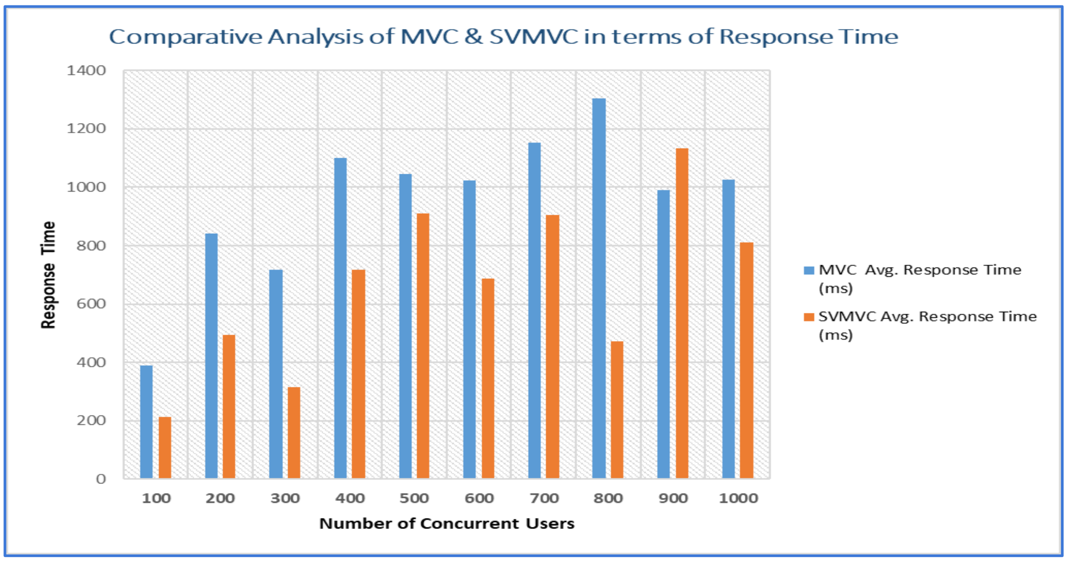

Figure 3: Comparative Analysis of MVC & SVMVC in terms of Response Time

The chart depicting response time analysis above clearly illustrates that in nearly every scenario, the refined SVMVC model outperforms the traditional MVC model when applied to web applications. In simpler terms, we can conclude that web applications based on SVMVC exhibit faster response times in comparison to websites built using the MVC model.

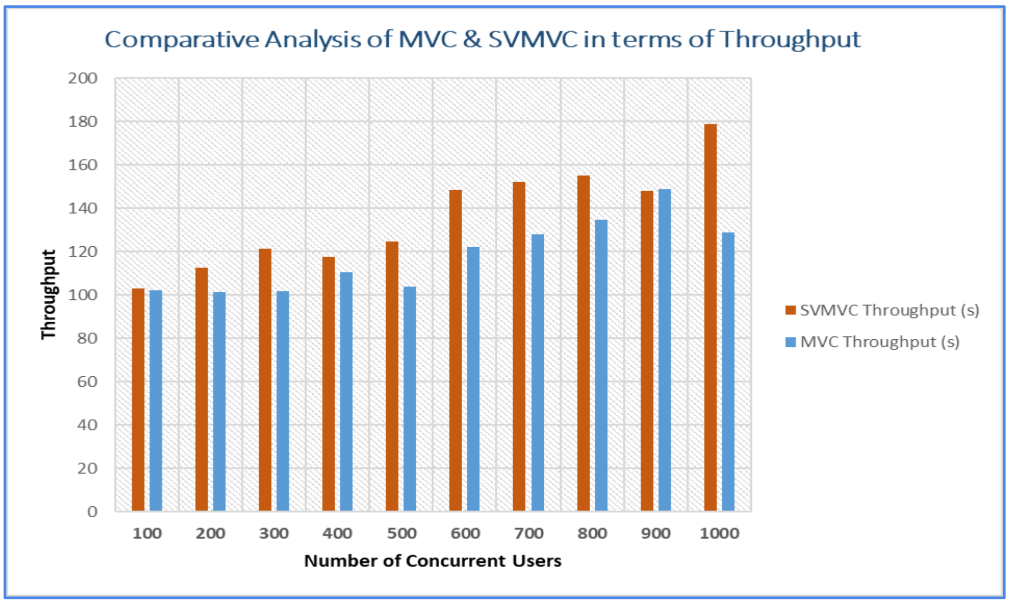

Figure 4: Comparative Analysis of MVC & SVMVC in terms of Throughput

The chart above demonstrates that SVMVC-based web applications have significantly higher throughput compared to MVC-based web applications. In other words, SVMVC-based web applications are more adept at accommodating a larger number of client requests when compared to MVC-based applications.

6. Conclusion and future work:

The experiments indicate that the proposed modifications to the MVC framework result in a substantial enhancement in response time and throughput. Consequently, it is reasonable to assert that the suggested SVMVC framework can be effectively employed to augment the performance and functionality of web applications. Future research endeavors can concentrate on further refining the application layer and other components within the MVC framework.

7. References

1. Subraya BM, Subrahmanya SV. Object driven performance testing of Web applications. InProceedings First Asia-Pacific Conference on Quality Software 2000 Oct 30 (pp. 17-26). IEEE.
2. Zhu K, Fu J, Li Y. Research the performance testing and performance improvement strategy in web application. In2010 2nd international Conference on Education Technology and Computer 2010 Jun 22 (Vol. 2, pp. V2-328). IEEE.
3. Leff A, Rayfield JT. Web-application development using the model/view/controller design pattern. InProceedings fifth ieee international enterprise distributed object computing conference 2001 Sep 4 (pp. 118-127). IEEE.
4. Schwabe D, Rossi G. The object-oriented hypermedia design model. Communications of the ACM. 1995 Aug 1;38(8):45-6.
5. Huang SQ, Zhang HM. Research on improved MVC design pattern based on struts and XSL. In2008 International Symposium on Information Science and Engineering 2008 Dec 20 (Vol. 1, pp. 451-455). IEEE.
6. Thung PL, Ng CJ, Thung SJ, Sulaiman S. Improving a web application using design patterns: A case study. In2010 International Symposium on Information Technology 2010 Jun 15 (Vol. 1, pp. 1-6). IEEE.
7. Aljamea M, Alkandari M. MMVMi: A validation model for MVC and MVVM design patterns in iOS applications. IAENG Int. J. Comput. Sci. 2018 Aug 1;45(3):377-89.
8. Kumar V, Chopra V, Makkar RS, Panesar JS. DESIGN & IMPLEMEN-TATION OF JMETER FRAMEWORK FOR PERFORMANCE COMPARISON IN PHP & PYTHON WEB APPLICATIONS. InInternational Interdisciplinary Conference on Science Technology Engineering Management Pharmacy and Humanities 2017 Apr 22.
9. Meier J, Farre C, Bansode P, Barber S, Rea D. Performance testing guidance for web applications: patterns & practices. Microsoft press; 2007 Nov 21.
10. Meier J, Farre C, Bansode P, Barber S, Rea D. Performance testing guidance for web applications: patterns & practices. Microsoft press; 2007 Nov 21.
11. Madupu, S. V. N., & Kumar, Kiran. (2020). An Examination of Applications Using the MVC Architecture. International Journal of Innovative Research in Computer and Communication Engineering, 8, 558-566. https://doi.org/10.15680/IJIRCCE.2020.0803065.
12. Agnihotri J, Phalnikar R. Development of performance testing suite using apache jmeter. InIntelligent Computing and Information and Communication: Proceedings of 2nd International Conference, ICICC 2017 2018 (pp. 317-326). Springer Singapore.