Mobile Testing as a Service

Pranjal Sharma
Department of Computer Engineering
pranjal.sharma@sjsu.edu

Rachit Saxena
Department of Computer Engineering
rachit.saxena@sjsu.edu

Amit Kamboj
Department of Computer Engineering
amit.kamboj@sjsu.edu

Nehal Sharma
Department of Computer Engineering
nehal.sharma@sjsu.edu

Abstract:
With the unprecedent growth in the usage of mobile phones, it becomes important to maintain the quality of mobile applications by conducting rigorous testing. This paper discusses the comprehensive design, implementation and validation of the functional components of a crowd sourced mobile testing as a service platform. Elastic provisioning of mobile device emulators, running test scripts in cloud on Android Virtual Devices using Appium server, Tester and Project Management Community, Google map view of testers locations, cost metrics based on resource consumption and bug severity and PayPal and Chatkit Pusher ntegrations are the salient features of our application, which have been elaborated upon in this paper. We also carried out stress testing using JMeter to determine the load our application can handle (in terms of serving requests).

Keywords: Mobile Testing, MTaaS, Android Virtual Device, Emulators, Cloud Testing

Introduction:

As smart phones and mobile devices have become more popular, the demand for mobile applications has increased at a rapid pace. Considering this ever-growing demand, efficient and cost-effective testing methodologies are of prime importance for quality assurance of mobile applications. However, testing of mobile and web applications for multiple platforms, browsers and devices is extremely costly and tedious because of rapid evolution of mobile devices, technologies and platforms. Traditional testing methodologies of mobile applications in laboratory incur higher costs and difficulty arises in large-scale usability testing. Therefore, online crowd sourced testing has evolved as a favorable choice amongst the members of the software development community. It can be effectively used to circulate test scripts to a group of online testers across geographical boundaries to accomplish on-demand testing of mobile applications in a cloud infrastructure. Its advantages include reduced testing costs and large-scale concurrent testing.

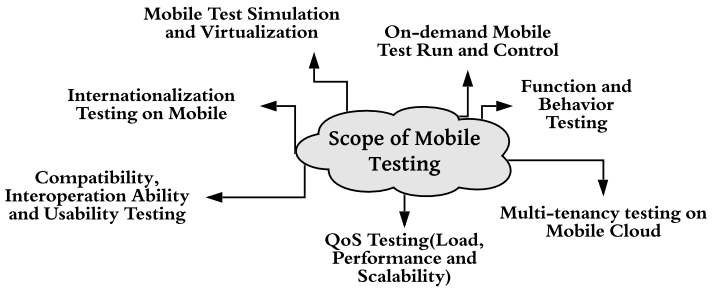
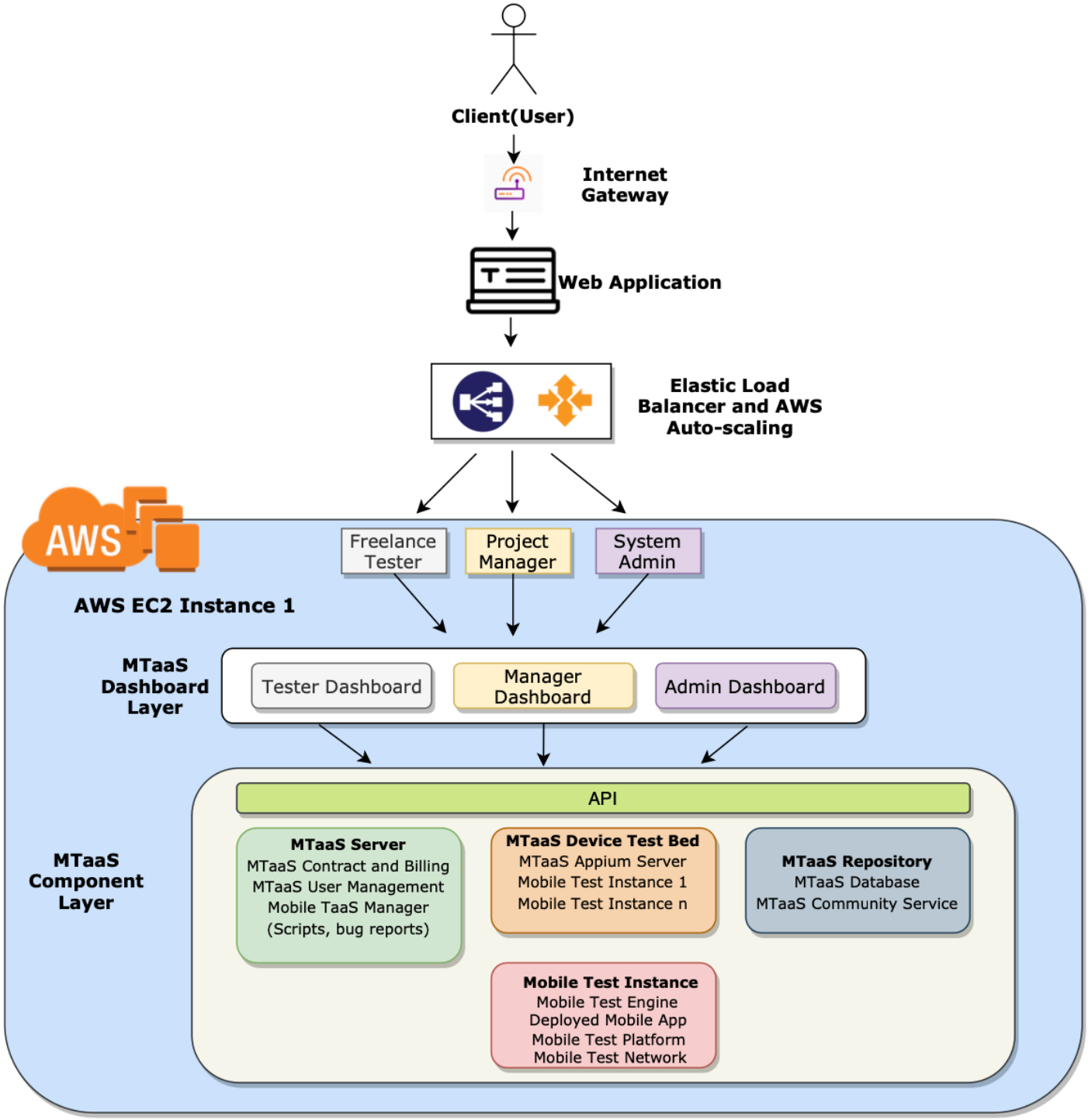


Figure 1:Scope of Mobile Testing

Our project, Mobile Testing as a Service, has been designed with the primary motive of facilitating the validation process of mobile software with included benefits of resource sharing and reduced costs for computation resources, memory and network infrastructures. In this paper, we have discussed in depth and breadth the implementation of our proposed architecture for an MTaaS, its database design, load balancing and scalability mechanisms, the graphical user interface implementation and conclude with the performance testing of our application.

Cloud based System Infrastructure and Components:



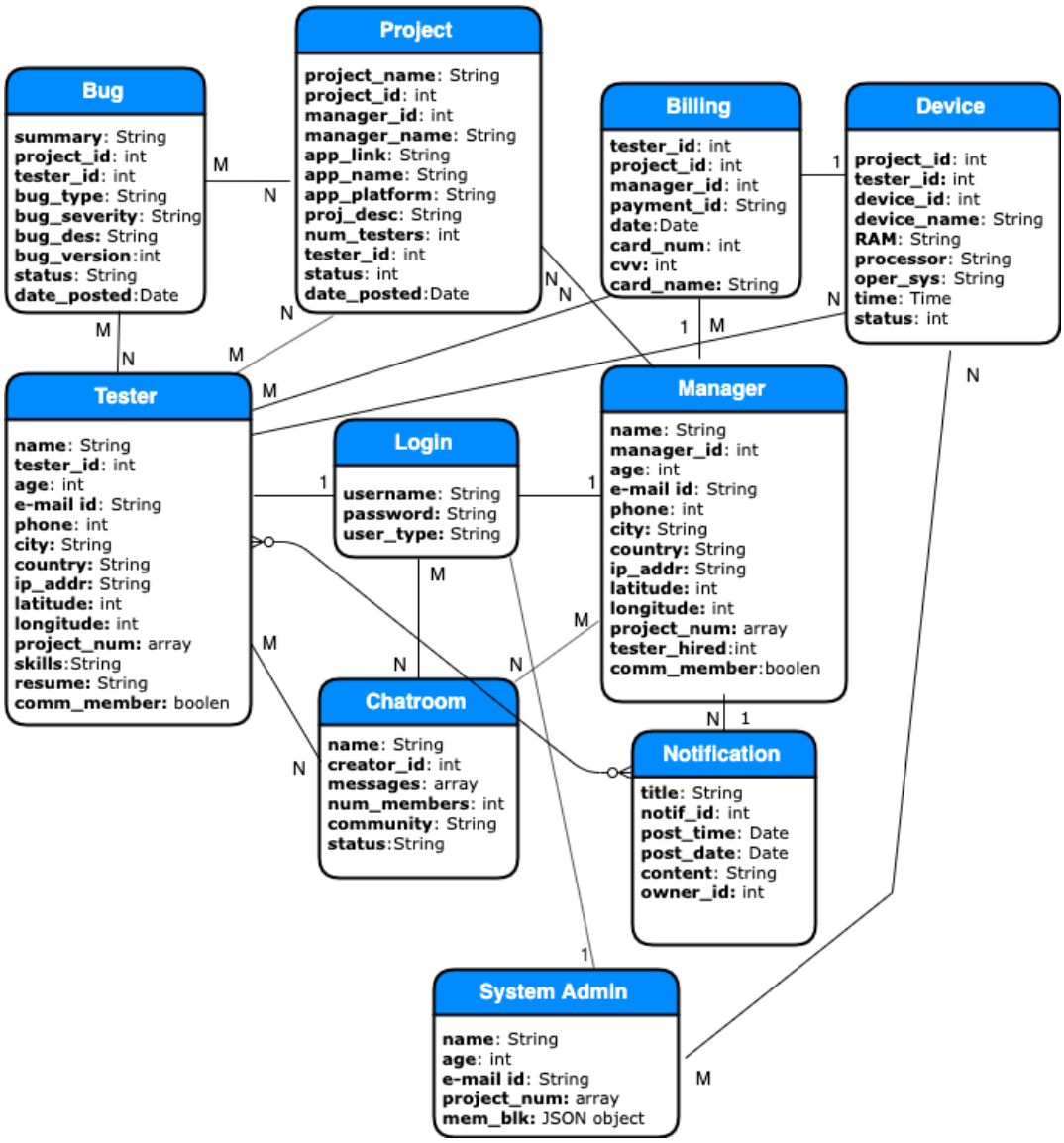
Data Design and Implementation:

For our application, we have used NoSQL database, MongoDB to store all the data. Our choice of NoSQL MongoDB over relational database MySQL was governed by the following factors:
(i)MongoDB is highly scalable and offers high performance coupled with automatic scaling and in-built sharding mechanism, which takes care of distributing data across multiple machines.

(ii)It is schema-less and can store unstructured and semi-structured data, which is appropriate for storing data such as chats, which may contain audio and image files also. Also supports geo-spatial indexes to provide support for querying geographical location based data.

(iii)MongoDB supports 'write-lock' mechanism and provides dynamic consistency to ensure the success for each write operation by blocking all other operations.

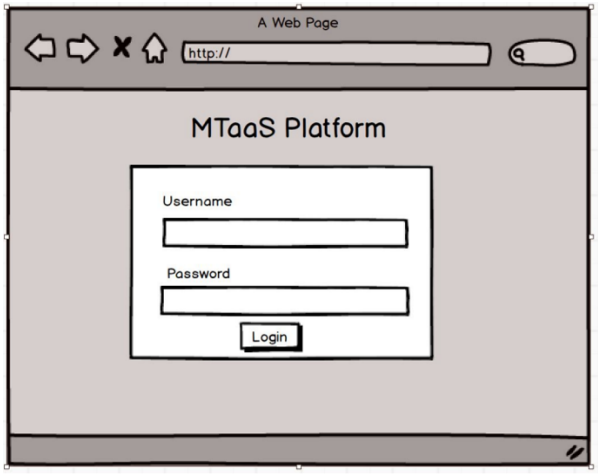
Hence, as cloud technologies are gaining prominence and popularity, we decided to go with MongoDB Atlas for managing and storing the data in our application. Given below is a schematic representation of the MongoDB collections used in our application.

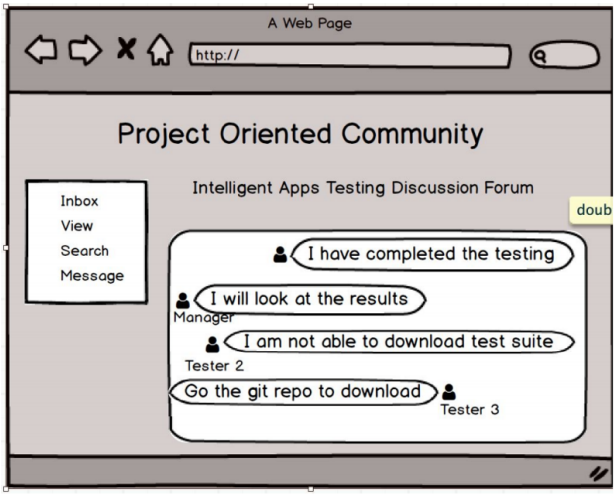


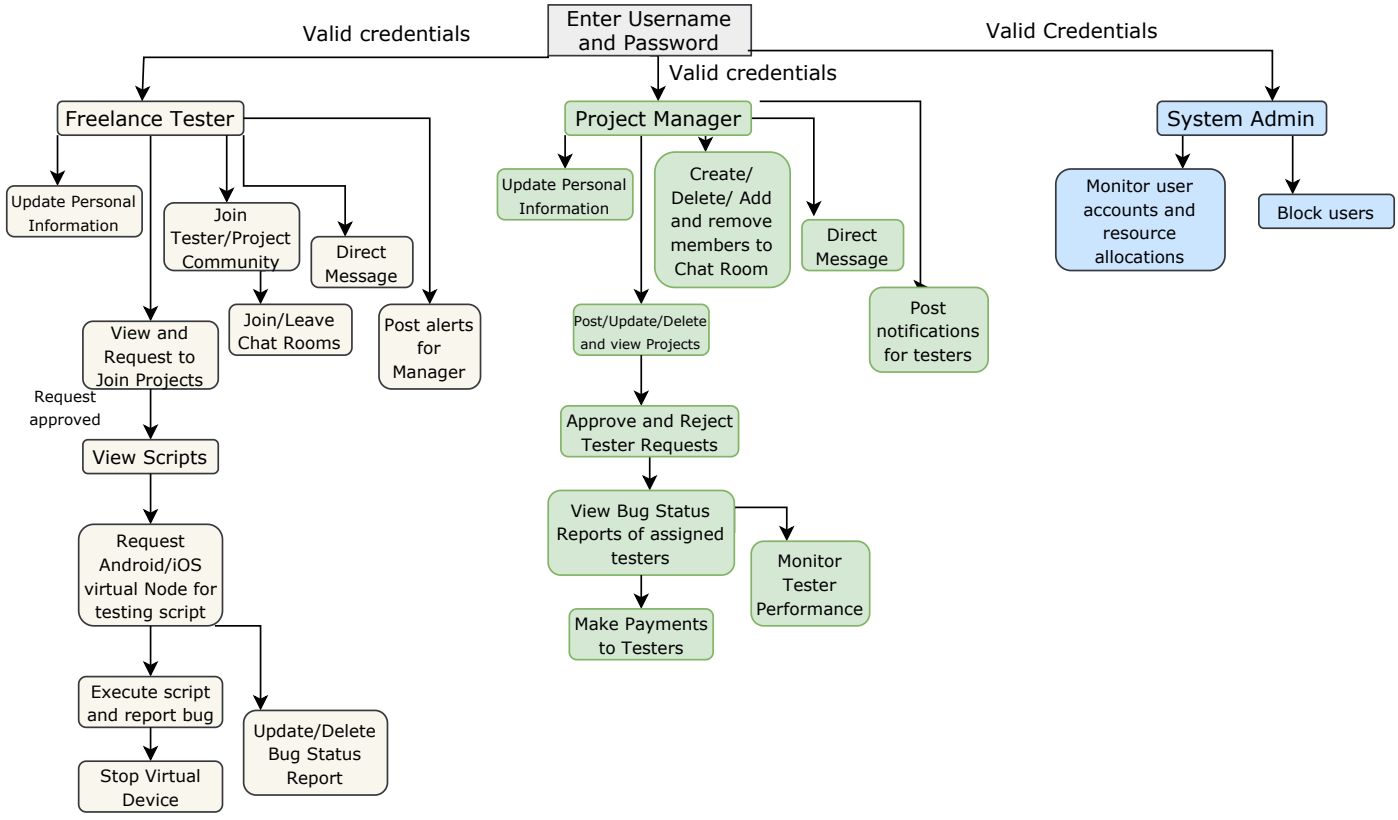
System GUI Design and Implementation:

For designing the Graphical User Interface of our system, we started out by understanding the requirements for the three categories of our system users i.e., freelance testers, project managers and system administrators. Since the roles are different for each user, each of them has different privileges and therefore, the GUI had to be designed accordingly so that each user has a view which is relevant to him/her.

For designing the user interface, initially we developed a low-fidelity prototype using Balsamiq(few screenshots shown below). After 2 repetitive iterations for prototyping, we actually started out with implementing the high-fidelity design of the application using ReactJS.







System Performance Evaluation and Experiments:

Throughout the development cycle, we carried out rigorous testing mechanisms to ensure correct functioning and optimal performance of our application. We carried out validation testing during the development phase and at the end to evaluate whether our system satisfies the client's needs and business requirements. Following were the main components of validation testing cycle that we carried out: Unit Testing, Integration testing, System Testing and Acceptance Testing. For our project, we also carried out functional testing by designing test cases to cover all functionalities to effectively seal all loopholes. For testing the performance of our application, we used Apache JMeter to record the average response time for different number of users and requests sent. The results played a pivotal role in helping us determine the maximum number of requests our system could handle without registering a degrade in performance. We tried to generate client behavior for different APIs in our system and produced load by making several concurrent requests to the application to estimate how the performance changes as the load progresses from low to normal to high(stress test cases).Stress tests are beneficial in evaluating the bottlenecks and hence planning for allocating more resources when the traffic starts affecting the application performance.We carried out JMeter testing for 200, 500, 1000, 5000 and 100000 requests and noted the time taken by the server to serve the requests. The graph for the results(throughput) are shown below (green lines represent throughput):

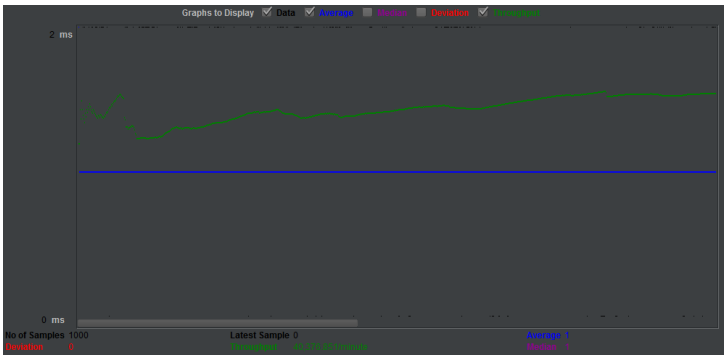


Figure: JMeter Graph for throughput for 1000 calls

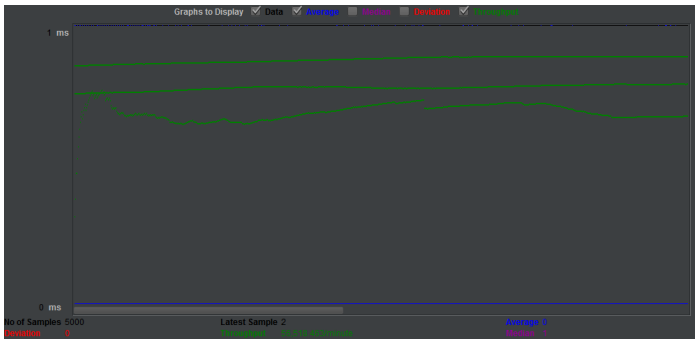


Figure: JMeter Graph for throughput for 5000 calls

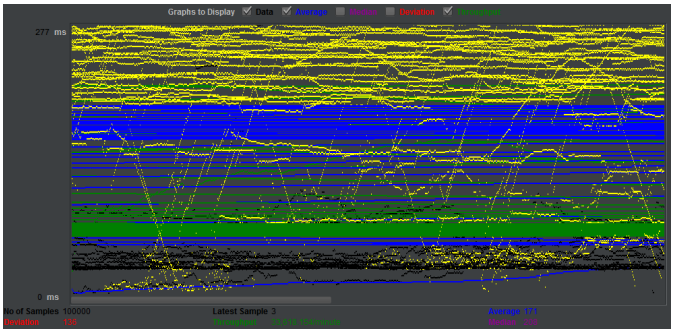
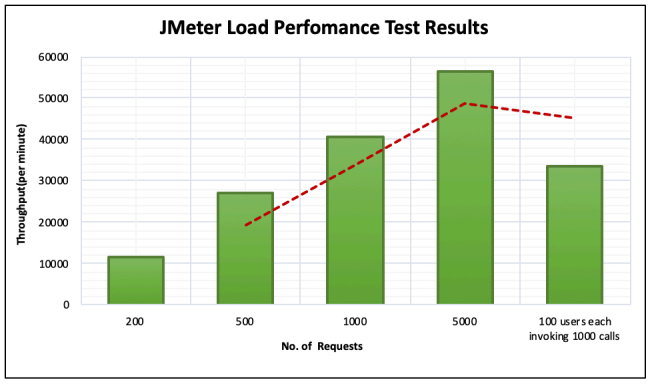


Figure: JMeter Graph for throughput for 100000 calls



From the above graph, we can see that the throughput consistently increases as the number of request increase. However, the values register a slight decrease when the number of requests reach 100000.Hence, we can conclude that this is the point where our application performance starts to degrade

Conclusion and Future Enhancements:

Through this paper, our team has successfully tried to explain our design and justify our system architecture choices for an MTaaS. It is a platform for crowd sourced testing of mobile applications which provides the facility to run test scripts on android virtual devices by allowing the tester to request virtual mobile nodes to simulate a testing environment. We have discussed the cloud-based system infrastructure, functional components, database design, load balancing and scaling mechanisms and the services offered by our system in great detail. To validate our system, we also conducted load performance testing and its results indicate that the server can handle user requests effectively under high load scenarios. Our current application can successfully execute android test scripts by running Android Virtual Device through Appium Server. As the future pathway for our application, we intend to integrate automated testing support for iOS scripts as well. The cloud computing domain has evolved rapidly and our team is grateful to Dr. Gao for giving us an opportunity to work on a project which aims to deliver mobile testing as a service through crowd sourced testing at a low cost.

References:

1. Guilherme Antonio Borges, Romulo Reis de Oliveira, Tiago Coelho Ferreto, Claudio Fernando Resin Geyer, "A Novel Model to Computational Offloading on Autonomic Managers: a Mobile Test Bed", *High Performance Computing & Simulation (HPCS) 2018 International Conference on*, pp. 157-164, 2018
2. Amro Al-Said Ahmad, Pearl Brereton, Peter Andras, "A Systematic Mapping Study of Empirical Studies on Software Cloud Testing Methods", *Software Quality Reliability and Security Companion (QRS-C) 2017 IEEE International Conference on*, pp. 555-562, 2017.
3. Jerry Gao et. El, “Mobile Testing-As-A-Service (MTaaS) –Infrastructures, Issues, Solutions and Needs”, IEEE 15th International Symposium on High-Assurance Systems Engineering 2014, pp. 158-168
4. Zachary Parker, Scott Poe and Susan V. Vrbsky, “Comparing NoSQL MongoDB to MySQL DB”, ACMSE ’13, April 4-6, 2013, Savannah, GA, USA, pp.36-42
5. *Xamarin test cloud: Mobile app testing made easy*, 2016, [online] Available: https://xamarin.com/test-cloud/.
6. Jia Yao, Babak Maleki Shoja, Nasseh Tabrizi, *Cloud Computing – CLOUD 2019*, vol. 11513, pp. 303, 2019.