Umbrella MTaaS

Sarthak Jain
Computer Engineering Department
San Jose State University
San Jose, USA
sarthak.jain@sjsu.edu

Amit Garg
Computer Engineering Department)
San Jose State University)
San Jose, USA
amit.garg@sjsu.edu

Alaukika Diwanji
Computer Engineering Department
San Jose State University
San Jose, USA
alaukika.diwanji@sjsu.edu

Nachiket Trivedi
Computer Engineering Department
San Jose State University
San Jose, USA
nachiket.trivedi@sjsu.edu

Jerry Gao
Computer Engineering Department
San Jose State University
San Jose, USA
jerry.gao@sjsu.edu

***Abstract*—We present a mobile testing platform as a service tool as a part of the coursework for Cloud Technologies(CMPE 281) at San Jose State University. The proposed testing as a service tool, named Umbrella MTaaS, incorporates three different entities namely Testers, Managers and Admin, in addition to primary functionalities with respect to each. The managers are the project holders who can create and post distinct mobile application projects and provide a working environment to the testers to delineate their submissions, while testers can browse through the available projects and apply to them, for intended subscriptions. Upon review by managers, the testers can either be rejected or accepted to the projects, the result being conveyed to testers in real time. Upon acceptance, the testers are added in a community forum pertaining to the project where they can engage in conversations with other peers as well answer and post queries with respect to the project. On the allocated file browser, the testers can create bug reports and upload testing scripts which they executed to find the related bug, while the managers can view and download the same. Apart from these preliminary tasks, the application also offers a unique dashboard to the users displaying stats about their work, a chat bot to help navigate the user throughout the application, a geomapping service for the managers to locate all the active testers for their projects, and an Admin who can assume the role as a superuser, thereby monitoring the entire application. The experimental results show Umbrella MTaaS, being deployed on AWS ECS along with an active load balancer, seamlessly fulfills the claimed functionalities.**

***Index Terms*—AWS, GCP, cloud, mobile testing as a service, bug tracking, community forum**

I. Introduction

The world is currently dominated by cloud computing. Be it a simple web application helping users order their favorite food, or a platform providing storage and network services to such applications, cloud is clearly the way to go. According to a latest report by Gartner [1], by 2020, the worldwide public cloud revenue is expected to grow by 17%. Considering every major player in the industry are getting their hands on the power of cloud, testing service should not be an exception as well.

With the advent of the omnipresent internet, the services offered by web applications and mobile applications have sky rocketed, along with their respective use cases. Developing these applications is another growing world in itself, and so is testing these applications. A single application development cycle involves multiple iterations of bug reporting, testing and debugging, which eventually ends up utilizing important computing resources of the organizations.

MTaaS, standing for Mobile Testing as a Service, is the task of diligent outsourcing of testing activities concerning a mobile application to a third party, mainly a cloud platform. [2] In this project, we present one such service called Umbrella MTaaS which aims to achieve the similar target. The project facilitates the user by providing them a working platform to post and review test reports, communicate with each other as well as monitor their performance, thereby providing a simulation of real-world testing environment.

In this project we give the managers a platform to create projects, their respective working directories for each testers as well as a community forum specific to each project. We used Amazon's S3 Bucket to provide storage to the managers for uploading the application specific files (for e.g. apk files), and for testers to upload their testing scripts and bug reports. The application serves as a middleman between the two communicating entities, along with providing storage and interacting platform, thereby operating a cloud simulation of the real-world testing medium.

The remainder of the paper is described as follows: Section II describes related work. Section IV presents the system design and component overview of the application. Section V describes the GUI design of the application in detail, while Section VI focuses on delineating the implementation of the application along with the major features provided. We provide experimental analysis in Section VIII, and eventually conclude our work in Section IX.

II. Related Work

In [6], the authors have mentioned various aspects of cloud testing as a service and have provided essential testing, focuses

and the required infrastructure. Following it, the authors have developed and presented a MTaaS platform to provide infrastructure for Mobile Testing as a service. In [5], the authors have focussed on testing of SaaS applications and gives the need for testing of SaaS based applications. Following this, they present an infrastructure required to meet the SaaS testing and discusses on the Cloud based platform required for SaaS testing called as CTaaS taking into consideration scalability, performance and testing requirements. In [4], the authors focuses on mobile testing rather than SaaS applications. They discuss the problems to engineers in testing the mobile applications and recommend a crowd-sourced testing platform for doing away with these limitations that engineer face in their laboratory while testing mobile applications. The paper also summarized the major players in the market providing MTaaS solutions, their products etc. Finally, in [3], the authors have sticked to the issue of discussing the problems and issues one face in SaaS testing while discussing important pertaining concepts of SaaS testing but at the same time reporting a lack of papers in SaaS Testing challenges and the need for more research in the field. These papers were a big support in getting the team started on Testing, the related concepts and the infrastructure required in providing with a Mobile Testing Platform as a service. We got major of our ideas from [3], and we proceeded with the knowledge we gained from the study of these papers. In [7], Isabel et al. discuss the techniques and use cases related to Automated mobile testing as a service and emulated devices under virtual machines and cloud infrastructure. In the paper [8], Prathibhan et al., present an automated testing framework for testing Android mobile applications in the cloud, where they incorporated an inbuilt mobile testing tool: Mobile Application Testing (MAT). In the paper [9] Murugesan et al., define a cloud powered mobile testing tool which can test projects for different mobile environments and platforms. The research work described in these publications acted as a foundation as well as a source of motivation for our project.

III. Cloud-Based Infrastructure & Components

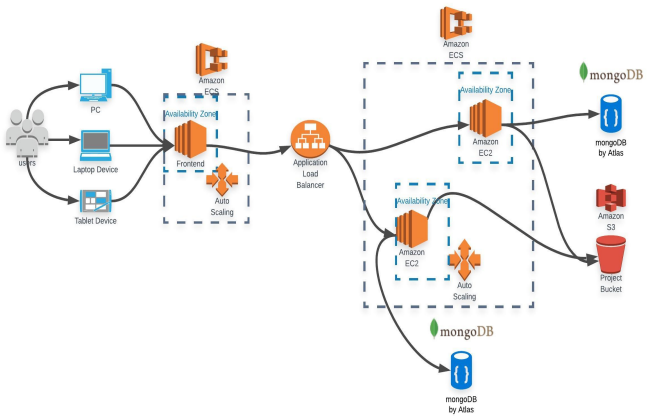

Fig. 1. Detailed past schedules information

Figure 1. gives an overview of our cloud-based infrastructure and the major components in our application. We have used Amazon Web Services (AWS) as our cloud hosting environment using its services such as Elastic Container Service (ECS), Elastic Compute Cloud (EC2), Application Load Balancer (ALB), Auto-Scaling Group (ASG), Simple Storage Service (S3). For our database, we have used MongoDb provided by Atlas which is a Database as a Service and so our Database was also cloud hosted. Along with this, we also used Geolocation api provided by Google cloud platform for getting the locations of users in a particular project. The below section gives a brief description on the above cloud components and its usage in our application.

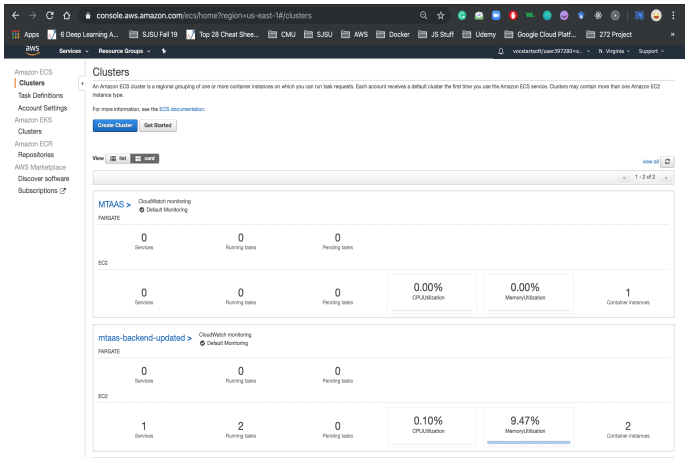

Fig. 2. Application's ECS Cluster

A. ECS & EC2

Elastic Compute Cloud (EC2) is the backbone for AWS. Whenever any user requires a computing environment, on the backend an EC2 instance is spinned up for the user giving user on-demand access to a highly scalable elastic infrastructure environment and the user being charged on a pay-as-you-go model. Elastic Container Service (ECS) is provided by Amazon for providing support to containerized applications. Our entire application has been divided into two components, backend and frontend. We have build containers for both the component and have deployed these containers on two separate ECS clusters which directly spins up two separate instances in which our containers are running. So the user talks to the frontend container, which then communicates with the backend container to process user requests. We have used t2.micro EC2 instance type for our frontend which provides 1 GB of RAM and 1 vCPU at a price of $0.0116 per hour. For our backend we have opted for t2.small which provides 2 GB of RAM and 1 vCPU at a price of $0.023 per hour.

B. Load Balancing & Auto Scaling

ECS comes with a useful feature of attaching any load balancers and auto scaling group with the instance cluster if you want while creating the ECS cluster. So we had beforehand configured our Application Load Balancer (ALB) and

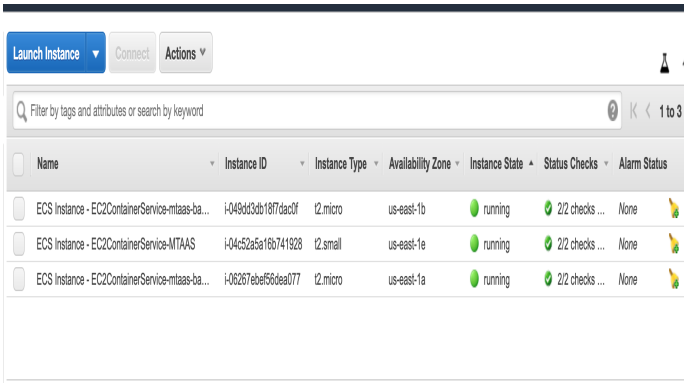

Fig. 3. Application's EC2 Instances

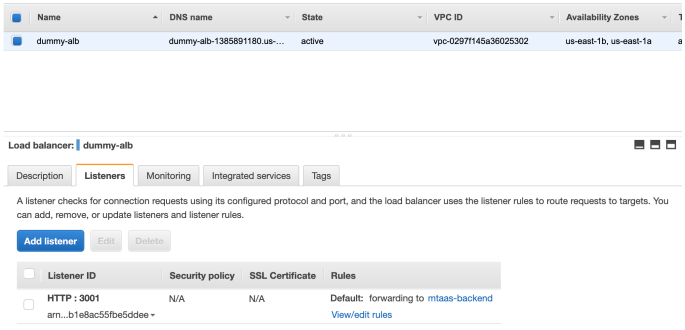

Fig. 4. Application Load Balancer for backend

Auto-Scaling Group (ASG) for attaching to this ECS cluster instance. For demo purposes we had selected maximum 2 instance for backend, minimum 1 and desired 1 for the configuration of our Auto Scaling Group. So we have at maximum 2 backend containers running on 2 different EC2 instances spinned up by Auto Scaling Group and infront of them, we have an application load balancer which is listening to requests on a specific port and directs those requests on these 2 backend instances via load balancing. The listening port and target port are configurable and is selected based on our application requirements. The ALB comes at a price of $0.027 per hour and $0.0096 per LCU-hour.

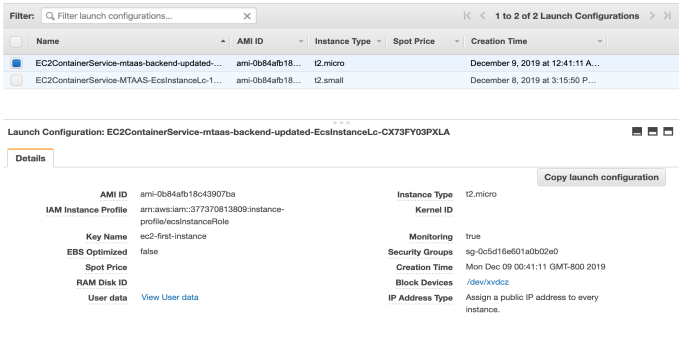

Fig. 5. Auto Scaling Group for frontend and backend cluster

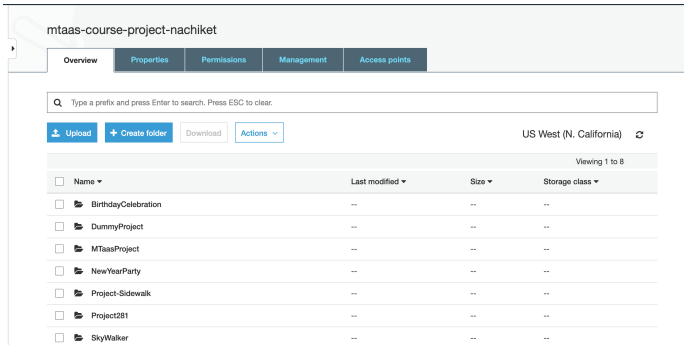

Fig. 6. S3 Bucket and contents for our application

C. Simple Storage Service

AWS's Simple Storage Service (S3) is used to store project and user files, if any. This service is primarily used while storing application files during project posting, storing script files during bug report creation, storing tester daily report files and act as a backbone for the custom file directory and bug tracking embedded into the application. S3 charges users based on a number of factors such as size of objects stored in S3 bucket, number of objects stored, number of requests and data retrievals, amount of data transferred and for data monitoring and replication. Their pricing can be easily found in S3 website. We have taken inspiration from this pricing to provide for a billing scheme of our own for the users of our application which have been mentioned later in the report.

D. MongoDB by Atlas

We have opted for MongoDB by Atlas for our database support. Our backend instances communicate with this cloud hosted database. While creating a cluster, we had the options to deploy this fully managed MongoDB either at AWS, Azure or Google Cloud. We opted for the deployment in AWS. So in a sense our Database cluster is hosted in AWS region but is fully managed by Atlas. MongoDB Atlas comes with its own pricing range with different pricing plans starting at $49 per month to hourly charges as well. For demonstration purposes we remained in the free tier cluster which is shared RAM and 512MB of storage capabilities.

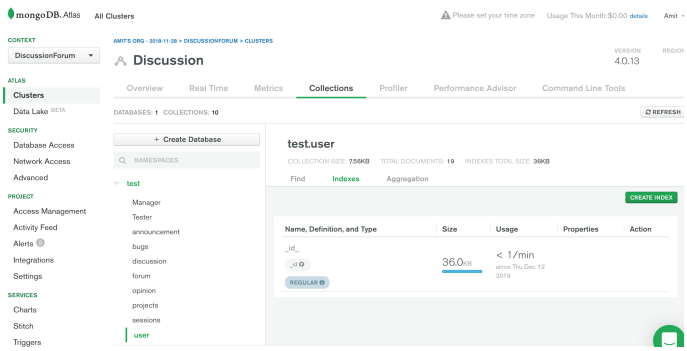

Fig. 7. Snapshot of MongoDb by Atlas and Schemas

E. Geolocation API by Google Cloud

For providing functionality such as getting all user's location in a particular project we have made use of Geolocation API provided by Google Cloud Platform. The api assigns it's users a key using which the user's can get a person's latitude and longitude giving a zipcode as an input to the api. We have collected tester's and manager's zipcode during sign-up and are using those zipcodes to fetch user's latitude and longitude at run time and are pinning those coordinates on the google map.

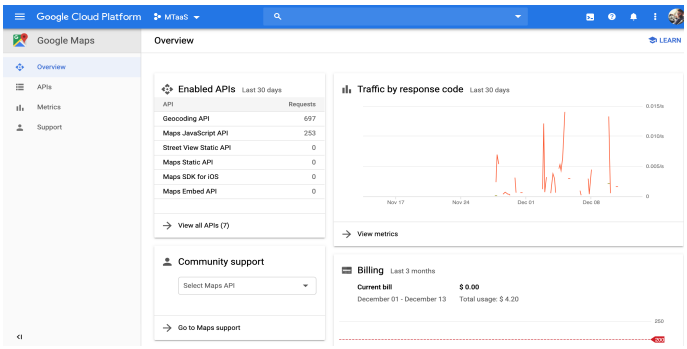

Fig. 8. GeoLocation API by Google Cloud

IV. System Design and Component Overview

Umbrella MTaaS is basically a web application, with it's frontend developed in ReactJS while the backend in NodeJS and ExpressJS. The data for the application is handled by MongoDB Atlas in a MongoDB cluster, which is accessed by ExpressJS for database querying. The database design of the application comprise of 6 collections namely:

- **user:** A common collection for testers' and managers' information like their technical and personal profile, and the projects(a list of project keys) they're directly involved with. In case of testers, another field for the projects they've applied to is also included.
- **projects:** A collection which will contain documents describing each individual projects and their details.
- **announcement:** Collection for announcement messages for each projects.
- **bugs:** Collection for each bug and it's details
- **discussion:** Platform for Tester and Manager to interact and discuss topics related to the project
- **opinion:** Tester/ Manager can give his views on the specific discussion.

The application features three major entities:

A. Admin

- **Pertaining to Project**
  The Admin will be a superuser who'll have access to all the database entities. They can log in and view their landing page where they'll have the option of viewing all the projects the application supports. This includes all the applications which the project manager has hosted, their project details, testers subscribed to that project, tester's submissions for the project and the chat details pertaining to that project. The Admin can remove an already hosted project from the database if the case arises because they already have access to the project collection. Inside this sub-module, the admin can also view the chat of the project.
- **Pertaining to Tester**
  As specified, the admin also has access to all the testers' details. They can view every tester's submitted information (apart from their passwords) including but not limited to their portfolio, skillsets, profile. They can even view the project subscriptions of the testers and the projects the testers have applied to. They can view the tester's submission details as well as their conversations with the manager. If the Admin feels a particular tester is behaving erratically, they can remove that tester at her will. This will cause the removal of the tester's account as well as de-subscription from their respective projects.

- **Pertaining to Manager**
  In addition to the aforementioned data, the Admin also has the power to 'manage' the managers as well. They can view the manager's profile, their past experiences, their total available projects as well as the testers they're in contact with. The Admin can suspend a project manager from the service and delete his account permanently if need be.

B. Tester

If the user is a tester, they'll first view the announcements provided by their subscribed managers on their home page. They can navigate throughout the application using a navigation bar provided to them on the side. They can view all the available projects and it's brief description. Here they can also apply for the existing projects. The next tab is the subscription, where the tester can view all the projects which they are subscribed to. For the subscribed projects, they also have access to a a file browser and a bug reporting tool where they can upload and download their test scripts and submit their work to the managers for review. The next component on the tester side is the community forum where they can chat with other fellow testers as well as the respective managers for each specific project. The data will be stored in a MongoDB collection. The next section is of profile, which is further sub-divided into 2 sections: Personal Profile and Technical Profile. The personal profile will display the tester their name, email, address, etc, and will also provide them an option to update their information. In the technical profile section, the testers can view as well as set their professional information like LinkedIn url, GitHub url, and technical skills. They'll enter their technical skills as comma-separated items and the application will take care of the rest. The next component is

the dashboard where the testers can view their statistics and logs.

C. Manager

If the user is a manager, they'll first view the announcements that they have already broadcasted to their subscribed testers on their home page. They can also set new announcements and thereby notifying their testers accordingly. They can navigate throughout the application using a navigation bar provided to them on the side. The managers are provided a section showcasing all their projects. Here they can even add new ones. If some testers are working on any projects, it'll reflect here as well. The next part on the manager side is the community forum where she can chat with other testers for each specific project. The data will be stored in a MongoDB collection. The managers can view and download bug reports created by the testers and check their submissions in file browser as well. The managers can even user the application's geolocation feature to get the locations of each testers working for them across all their projects. Furthermore in their dashboard, the managers can view the statistics based on their active projects, number of testers subscribed etc.

V. System GUI Design

A. Frontend and GUI of the application

The frontend of the application is developed using ReactJS, and the styling is designed using CSS3 and react-bootstrap. The first page of the application is a landing screen, from where the user will 'Start' his work. If the user is logged out of the session or he's not registered, then he'll be redirected to the signin screen where the user can either login to an existing account or create a new one. Be it Tester, Manager, or the Admin, the user will be provided a red coloured sidebar for navigating each functionalities available them by the application. This navbar will their primary medium of navigation and 'jumping' across components. The remaining UI components are individually delineated in the following sections.

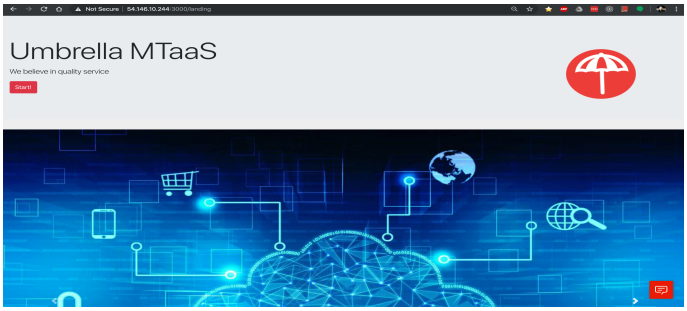

Fig. 9. Landing Page

VI. Functionalities

A. Project enrollment cycle

The project essentially revolves around testers working on their subscribed projects. This starts when a Manager

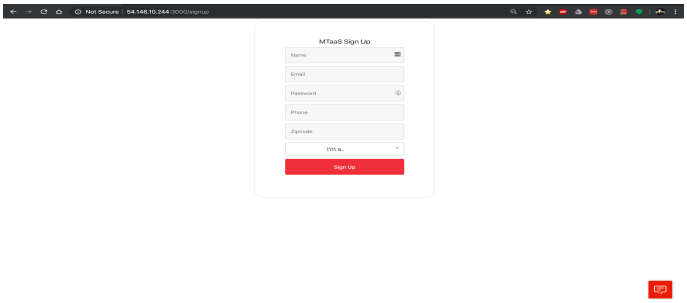

Fig. 10. Signup Page

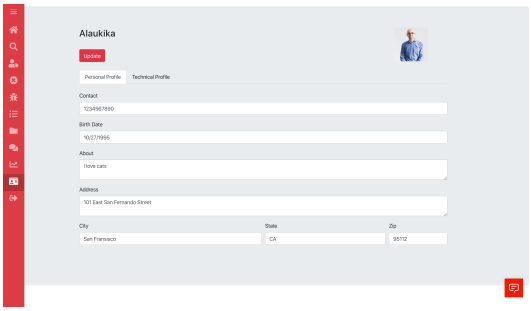

Fig. 11. Profile

Fig. 12. Side Navigation Bar for Tester

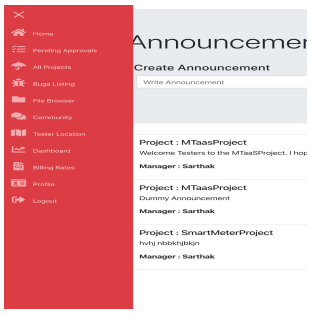

Fig. 13. Side Navigation Bar for Manager

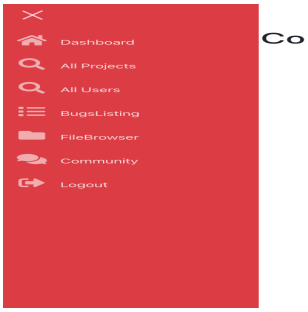

Fig. 14. Side Navigation Bar for Admin

creates a new project in the first place. They'll include project description, details, necessary files(like .apk), technologies involved etc and hence create a new project. This project will be posted in the public pool. Every tester registered to the

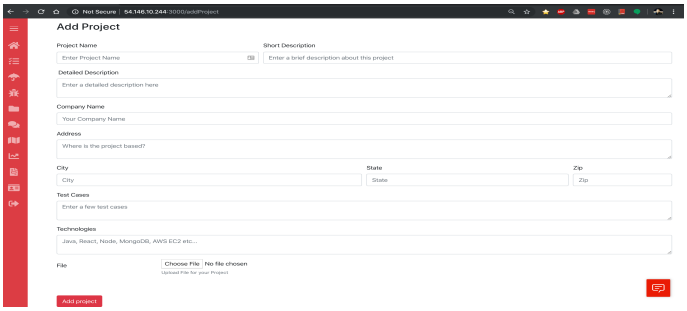

Fig. 15. Add Projects

application can view the project. Now on the testers' side, they can browse through all the available projects and apply for the ones they're interested in. This will prompt the application to share the Tester's personal and technical profile with the manager. Now, on the manager's side, they'll get a newly

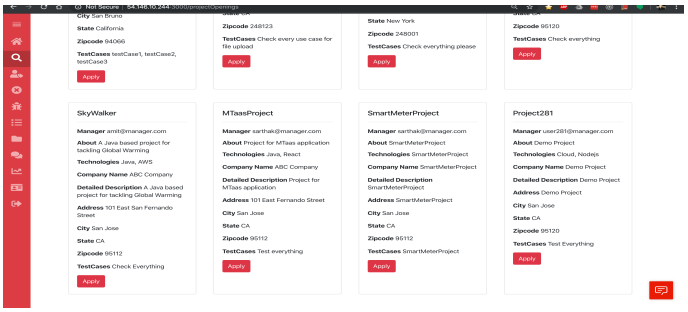

Fig. 16. Browse Projects

created request for approval in the Pending Approvals section. Here the manager can view all the Testers' request who have applied for all the projects they're managing. The manager can view the tester's personal and technical profile and decide for themselves whether to approve the tester or reject them. If the tester is approved, they're added to the project working testers' list. The tester is even assigned a file directory for his work and submissions and an enrollment in the community forum pertaining to the particular project, more on this in the following sections. If the tester is rejected by the manager, they'll not be added to the working testers' list, and the tester will view their rejections in the unapproved applications list.

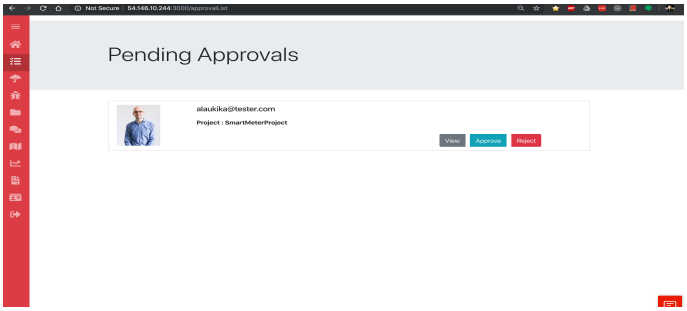

Fig. 17. Pending Approvals

The manager can even create announcements specific to each projects which all the testers enrolled in the project can view. The testers can even dismiss the announcements once read.

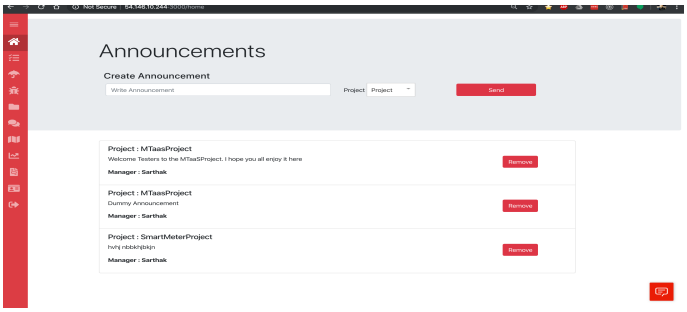

Fig. 18. Announcements

B. Bug Reporting & Tracking

Testing an application would require testers running scripts on an application possibly to find issues where the application breaks a.k.a. bugs. We have built and integrated our own custom bug reporting and tracking tool. Tester's would be able to create bug reports for a particular project, select the operating system & version, along with providing summary for the found bug & finally provide the testing script they ran due to which they were able to find the bug report. This testing script is stored in the tester's space inside Project's space in S3 bucket. This testing script is available for download to tester, manager & admin. Tester's have ability to edit the bug if they found a mistake, delete the bug.

Tester's can see all the bugs they have created in a particular project by the Bug Tracking tool given to them. Manager of the projects would be able to see all the bugs that have been created by all testers in a particular project. They have the option to download the testing script to recreate the bug found by the tester and possibly report it to their application developers to fix the bug found. This tool provides basis for the manager to later pay the tester on the basis of number of scripts testers ran and the number of bugs they found.

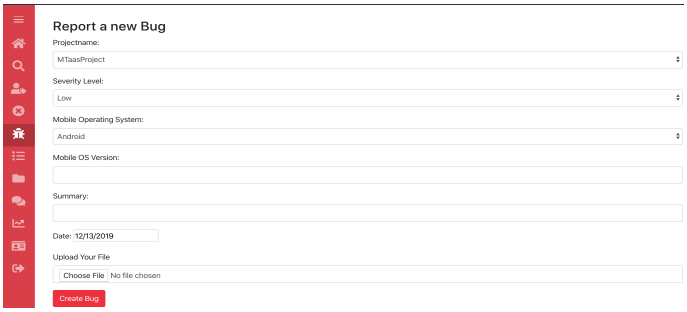

Fig. 19. Custom Bug Reporting Tool

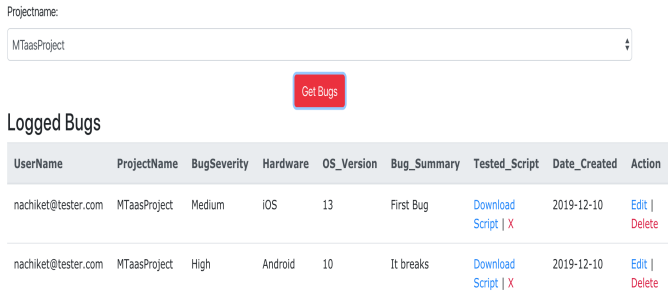

Fig. 20. Bug Tracker for Tester

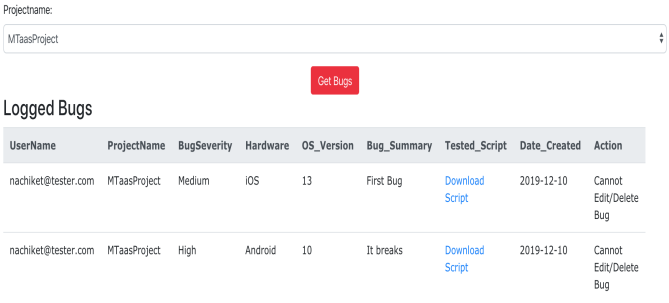

Fig. 21. Bug Tracker for Manager

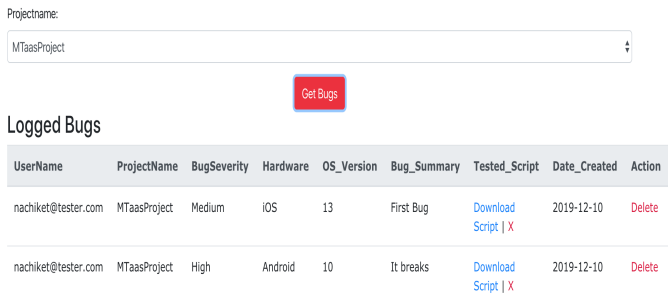

Fig. 22. Bug Tracker for Admin

C. File Browser

Every project is backed by the File Directory embedded in the application to see the project related files. Every Tester would be able to fetch files that are common for every person in the project such as application .apk files and other files uploaded by Project Manager and Admin that are present in the Common folder inside the Project folder. Additionally, every tester would have their own folder inside the Project where all their testing scripts and other files upload would be present. Similarly Project Manager would be able to see entire Project folder content of their projects and Admin would be able to see entire content of every project. Project Manager and Admin can access tester's folder in the project, download their uploaded files, upload files to common folder and even delete files if necessary. Below snapshots provide a view of the File Browser from Tester, Manager & Admin logins.

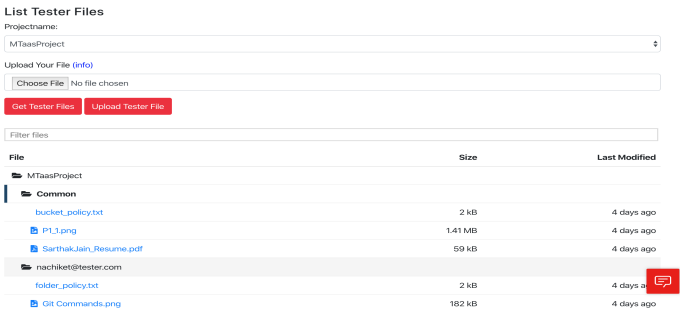

Fig. 23. File Browser for Tester

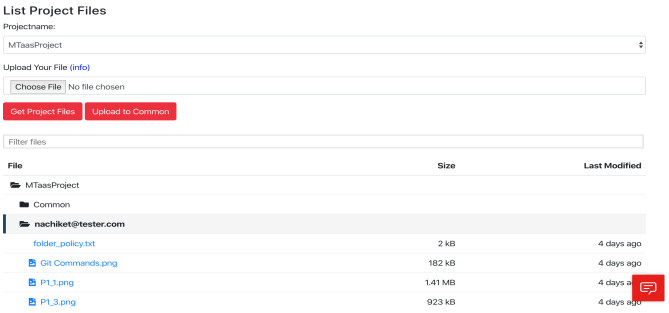

Fig. 24. Project Manager & Admin File Browser

D. Community Forum

Platform for Testers/Managers to interact with each other and discuss topics related to particular project. Testers will be able to see all the enrolled project in the listed project above and can switch b/w projects Whereas Managers can see all the projects, posted by him/her. Both Testers/Managers have privilege to post discussion/opinion via Discussion Forum. Admin has some special rights like he/she can view all the posted projects/ Discussion/ Opinion. Admin has privilege to delete/create any discussion/opinion posted by testers/managers.

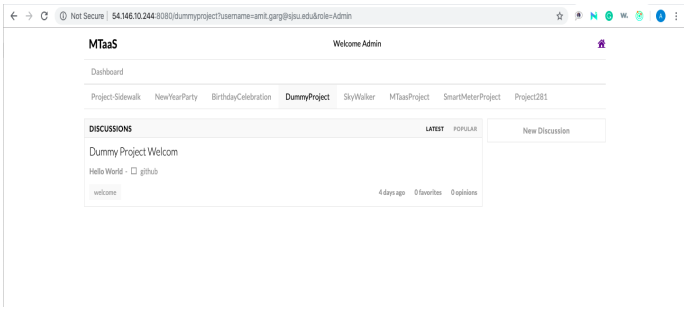

Fig. 25. Discussion Forum HomePage

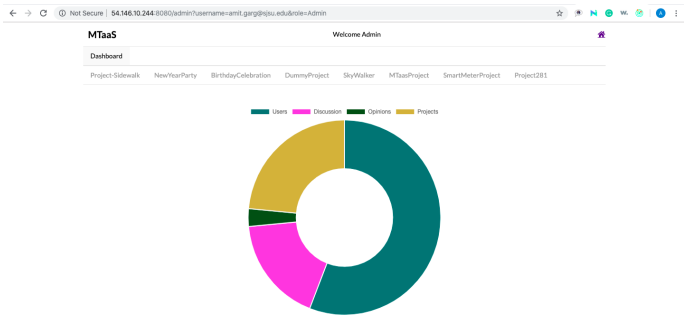

Fig. 26. Discussion Forum Admin Dashboard

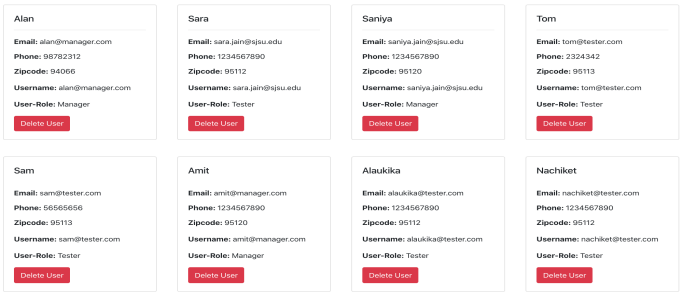

Fig. 27. Admin User Delete Component

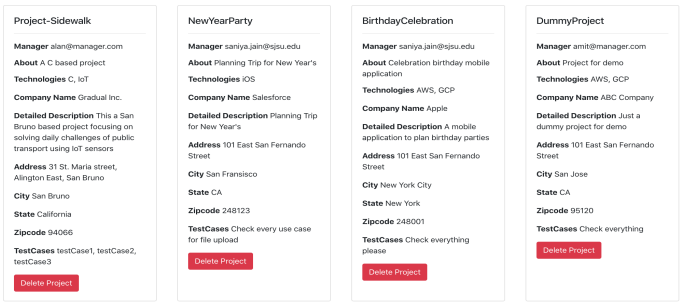

Fig. 28. Admin Project Delete Component

E. Admin functionalities

The platform owners are by default the Admin of the application. They have access to all the projects, all the users, all the S3 space, all the reported bugs. Admin of the platform can delete a project which would remove all the project files from S3, remove project from database along with removing project from tester's project subscriptions. Admin can delete a user as well. If a tester is deleted, the tester will be removed from every project, all the tester files is deleted from S3, all the reported bugs will be deleted. If a manager is deleted, then all the projects posted by manager will be deleted, all the project space will be deleted and those projects will be removed from tester's project subscription too. These user and project delete privileges are only given to Admin. Admin have access to File Browser just like Project Manager but additionally, Admin has privileges to see all Projects file browser. Similar concept is for Bug Tracking, Admin has privileges to see reported bugs for all projects.

F. Dashboard

We have created a Dashboard for Tester, Manager & Admin where they will be able to see different reports concerning to them. For tester, they will have reports like number of bugs they have reported per project and number of files they have uploaded in S3 per project. For Project Manager, they have reports like number of testers in each project, number of reported bugs from all testers in each project, cost per project and number of files in S3 per project. For Admin, reports like different number of users, tester & manager, number of projects with different count of testers, different project size in S3 are provided for better transparency into project, user and storage space.

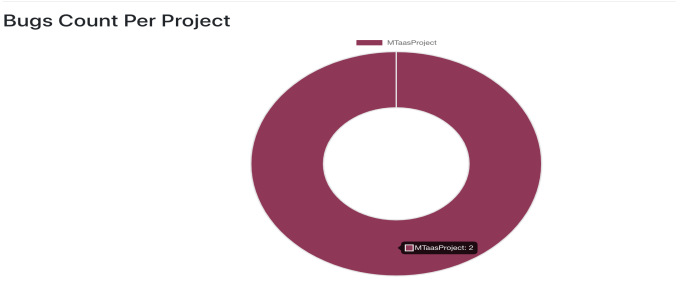

Fig. 29. Number of Bugs Created Per Project from Tester Dashboard

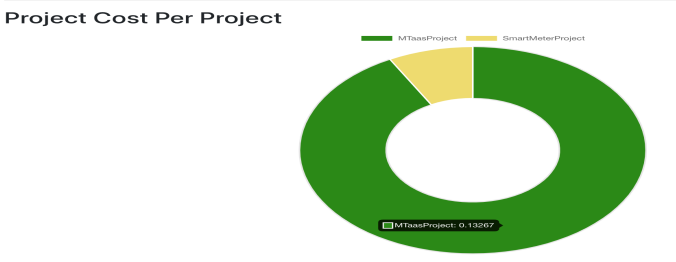

Fig. 30. Project Costs per project from Manager Dashboard

G. Miscellaneous features

1) Chatbot: In this application, we've incorporated a chatbot powered by IBM Watson for easy navigation of the

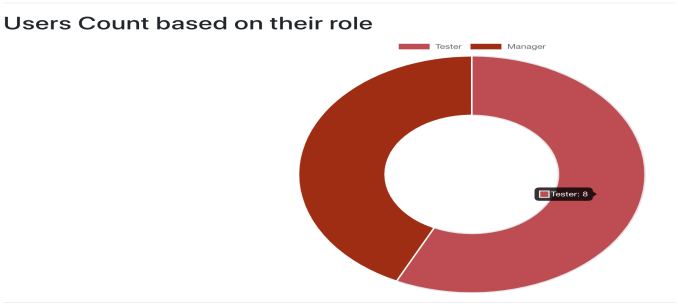

Fig. 31. User count based on roles from Admin Dashboard

application. The chatbot will remain omnipresent throughout the application, on all the webpages and will remain active as well. The user, be it tester or manager, can ask the bot questions 'Tell me about the application?', 'How to start with the application?' etc.

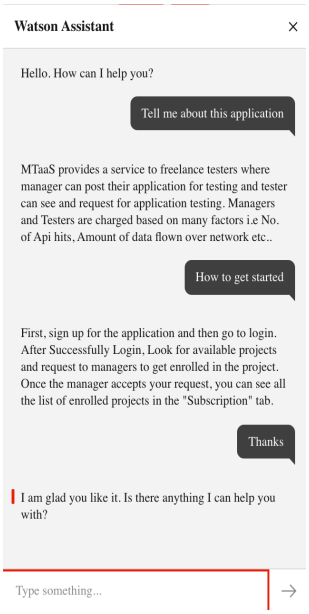

Fig. 32. Chatbot

2) Billing: On the manager's side, the application provides a pay-as-you-go model based on the parameter of storage. We've set $0.030 per MB, and $0.20 per object as the billing rates. The manager will utilize the application's resources, allocate those to the testers and projects, and hence shall pay for the increased consumption. We've integrated PayPal payment service for payment of bills where the manager will be redirected to PayPal's portal and they can proceed to pay likewise.

VII. System Application Examples

The system acts as a Platform connecting Project Managers who wants to get their application tested to crowd-sourced testers. This also acts as an opportunity for freelance testers

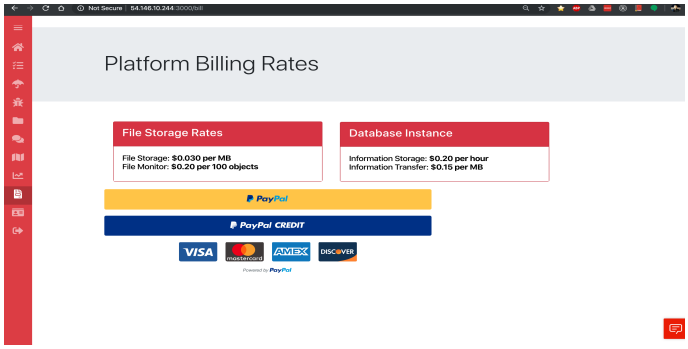

Fig. 33. Billing

who wants to put their skills to use and possibly earn too. Thus the application is beneficial to both the developers and testers and can be applicable to different applications in present day world. Below are some of the use cases:

- Companies can use the platform to get their applications tested from different testers exceeding companies outreach and thus would have a better transparency into the vulnerabilities of their application before actually releasing their product to the world.
- The platform can be used in educational institutes among student who are working on developing products for their startup ideas and they possibly cannot hire testers at the moment. The platform can be used to connect them to professional free lancing testers who can test their applications on demand through our platform. Students will have access to reports shared by testers.
- Embedded file browser can act as a storage directory among organizations where they want to limit access to files among users based on their roles. Thus our application could fit right into their demands giving them anywhere access to their files.

VIII. System Performance Evaluation and Experiments

A. Performance Evaluation

We've deployed the backend of our application on AWS ECS, the instances being handled by a load balancer. For performance evaluation, we tested two API endpoints using JMeter and captured the results, which are described as follows: For 10000 concurrent users and a ramp up period of 5 seconds, the login api route of Umbrella MTaaS gave a throughput of 30069.159 requests per minute. For 10000 concurrent users and a ramp up period of 5 seconds, the viewAllProjects api route of Umbrella MTaaS gave a throughput of 58731.402 requests per minute.

IX. Conclusion

We would like to thank our professor Dr. Jerry Zeyu Gao for giving us an opportunity to work on this application. Without his guidance the application would not have been possible. This application has provided all of us in the team experience

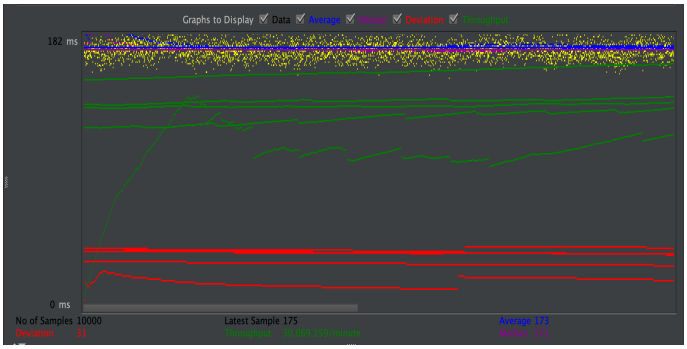

Fig. 34. Performance of Login

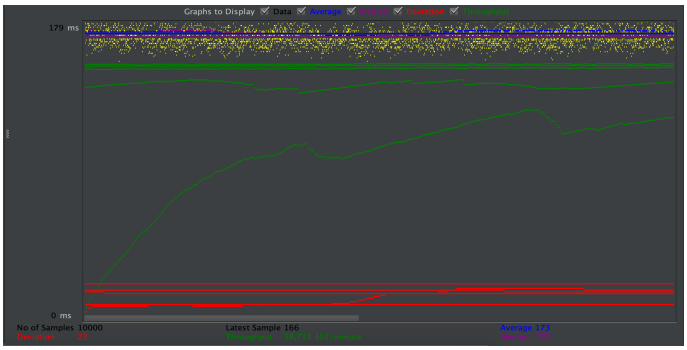

Fig. 35. Performance of View All Projects

into Cloud Stack and how cloud technologies have made it easier for developers to develop and deploy and reach the market seamlessly. The application has indeed provided us much needed skills in the cloud domain by giving us an opportunity to explore Amazon web services. Additionally, the project helped us in thinking and making decisions regarding multi-tenancy, considering applications scalability and thus making use of load balancers and auto scaling groups. It helped us to think for highly scalable and available platform which is connecting Project Managers to crowd-sourced testers on demand.

REFERENCES

[1] https://www.techrepublic.com/article/global-public-cloud-services-market-expected-to-grow-17-in-2020/, Lance Whitney, November 13, 2019
[2] https://www.softwaretestingclass.com/software-testing-as-a-service-taas/, September 27, 2018
[3] https://www.researchgate.net/publication/261155381_SaaS_Testing_on_Clouds_-_Issues_Challenges_and_Needs
[4] https://par.nsf.gov/servlets/purl/10092502
[5] https://ieeexplore.ieee.org/document/6427555
[6] https://www.researchgate.net/publication/309885645_On_Building_a_Cloud-Based_Mobile_Testing_Infrastructure_Service_System
[7] Isabel Karina Villanes ; Erick Alexandre Bezerra Costa ; Arilo Claudio Dias-Neto, IEEE, 2015. https://ieeexplore.ieee.org/abstract/document/7196507
[8] C.Mano Prathibhan ; A. Malini ; N. Venkatesh ; K. Sundarakantham, IEEE, 2014, https://ieeexplore.ieee.org/abstract/document/7019292
[9] Logeshwaran Murugesan ; Prakash Balasubramanian, IEEE, 2014, https://ieeexplore.ieee.org/abstract/document/6912148