Campus Security Robot

1st Girish Bisane
Software Engineering
San Jose´ State University
San Jose, USA
SID: 016650348

2nd Rajat Prashant Masurkar
Software Engineering
San Jose´ State University
San Jose, USA
SID: 016044015

3rd Venkata Siva Prasad Kakkera
Computer Engineering
San Jose´ State University
San Jose, USA
SID: 015935101

4th Vinti Jain
Software Engineering
San Jose´ State University
San Jose, USA
SID: 015955446

***Abstract*—Cloud-based robot service platform Campus Security Robots (CSR) allows business owners to keep one or more robots on-premises for security purposes. In an ideal world, CSR would be a software as a service (SaaS) platform that businesses could use to rent or lease robots through a user account provisioning system. Because it's a platform as a service, it also provides a collection of APIs that can simplify the development of software as a service. It makes it easier to rent out robots and set up, monitor, and run safety systems on campus. In order for users to take part in a cloud-based system, they need an application that can talk to our backend servers and handle requests. The same holds true for administrative duties, which will be impossible without remote communication. The user can order a security robot through the CSR's customer interface and have it sent anywhere they want. As part of the project's testing phase before it goes live, the Webots API and the Webots virtual environment are used to simulate real-time robots.**

***Index Terms*—Robot, Cloud Computing, AWS**

I. Introduction

Cloud robotics has been one of the fastest-growing businesses in recent years. Because this field is growing so quickly, more time and effort are being put into it. The study of robot clouds, which is important to the overall success of the business, has a lot of ground to cover. Service Robots are becoming more common as robotics and AI improve and make them more useful in our daily lives. One problem in robotics is that there aren't enough resources to make it hard to design and build robots. lack of a comprehensive, open-source, and free robot cloud service. Here, we offer a solution by saying that robot services could be delivered through a cloud-based infrastructure. In addition to remote registration, setting up the robot, and checking on its status, you also need remote control, monitoring, and tracking. The goal of this project is to create a cloud-based service or platform for mobile robots that lets the robots and a cloud server talk to each other in both directions. This way, the cloud server can provide the services listed above to the robots. In the haze We will build a robot cloud service to handle the day-to-day operations and finances of the institution. robots patrolling the campus to ensure everyone's safety. This platform will be in charge of making, configuring, controlling, and keeping an eye on this class of robots. The main goal of this project is to build a cloud-based infrastructure that can support edge-based robotics. Our goal is to make it easier for autonomous robots in the environment to connect to a server in the cloud. User administration will also be handled by the cloud service. 1. Robot clients - Students 2. Cloud Staff - College Staff 3. Robot System Admin - Administrator This cloud continues to support edge-based robot services such as registration, tracking, configuration, control, and monitoring. As a cloud platform, we will be in charge of managing user accounts, showing robot catalogs, delivering robot services, and billing customers for these things. The capabilities of the CSR infrastructure are outlined below. A comprehensive definition of CSR is provided. that which makes up architecture and what parts it consists of Information is given about the architecture of the simulation, the features of the robot simulator, and the networking options that CSR uses in their Webots simulations. Cloud data architecture and development with RDS and MongoDB are then discussed. After that, we'll talk more about cloud system architecture and the services that go along with it. Some of the CSR features that are explained in this section are account management, scalability, and load balancing. Next, each of the three portal kinds' GUI architectures and implementations are discussed. A scenario-based system application using portal graphics is shown in the next section as an example. Insights and lessons learned from CSR's history are presented in the report's last section.

Identify applicable funding agency here. If none, delete this.

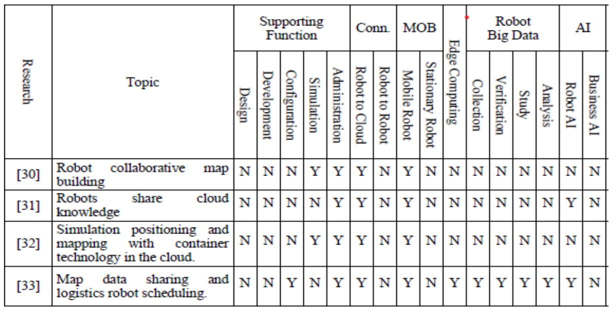
II. RELATED WORK

A. *Review of Research Work*

At present, research related to robot cloud is limited, and some people have published their research results in the last 10 years. They have used cloud technology for centralized management of robots and also for arithmetic support. These studies were relatively not that important. They have focused on robot management using different technologies . Further studies in this field have introduced the inclusion of edge computing i.e edge based robots handling, big data analytical based on cloud. Studies have also been made on cloud based AI computation . Summarizes of some research papers are shown in the given diagram.

B. *Overview of our system*

The data of provided by the users during registering on our cloud paltform and our security robot data is stored on a Cloud platform in sql database and No-SQL database. The data of the different users i.e users having different role, end-user



Research	Topic	Supporting Function					Conn.		MOB		Edge Computing	Robot Big Data				AI	
		Design	Development	Configuration	Simulation	Administration	Robot to Cloud	Robot to Robot	Mobile Robot	Stationary Robot		Collection	Verification	Study	Analysis	Robot AI	Business AI
[30]	Robot collaborative map building	N	N	N	Y	Y	Y	N	Y	N	N	N	N	N	N	N	N
[31]	Robots share cloud knowledge	N	N	N	N	Y	Y	N	Y	N	N	N	N	N	N	Y	N
[32]	Simulation positioning and mapping with container technology in the cloud.	N	N	N	Y	Y	Y	N	Y	N	N	N	N	N	N	N	N
[33]	Map data sharing and logistics robot scheduling.	N	N	Y	N	Y	Y	N	Y	N	Y	Y	Y	Y	Y	Y	N

Fig. 1. Literature Survey of Robot Cloud

application's back-end infrastructure system. These college staff are in charge of tracking and monitoring the robots car by gathering data from the our edge based stimulation robots. Amazon Web Services was chosen to be our cloud provider for our project because of its wide range of software services. We will be using the services that meet our needs for our application deployment. We will be making ensure that high availability and scalability are achieved in our design.

III. CLOUD-BASED SYSTEM INFRASTRUCTURE AND COMPONENTS

In this section, we will be talking about the overall architecture and design of the Robot Cloud system. It is made up of the following parts:

1) Robot Backbone Cloud: This is the main part of the robot cloud service, which is mostly made up of things like robot cloud service, robot cloud big data, robot cloud AI, different dashboards, and basic computing resources.
    a) Robot Cloud AI: This module enables AI on the robot cloud and gives front-end robots support for AI capabilities that are more advanced. Because of this AI feature,robots will be smarter and need less computing power.
    b) Robot Cloud Big Data: This is a type of big data analysis based on the large amount of information about how robots work that the robot cloud collects.
    c) Robot CLoud Service: The core functional module of the robot cloud service,which includes user administration, robot administration, security management,and servicing among other things.
    d) Robot Cloud Dashboard: Users can get a variety of data dashboards from the robot cloud. These dashboards are split into two groups: dashboards for robot cloud users and dashboards for robot cloud services.
    e) Elastic Load Balancer: Elastic Load Balancing automatically distributes incoming network traffic across multiple locations, such as IP addresses, containers, EC2 instances, etc., in at least one of the availability zones. It keeps an eye on how well the people it has signed up are doing and sends traffic to where it needs to go.component oriented
2) Auto Scaling: Modification is able to keep up with unanticipated execution with low expense because applications are assessed. Setting up apps in different locations is simple and easy using AWS Auto Scaling.
3) AWS EC2: Amazon web services EC2 is a service from Amazon Web Services that gives you scalable computing power and virtualization software in the cloud. We can easily create new instances by using images of other systems.
4) AWS RDS: It is a controlled SQL database that Amazon Web Services assists in the operation of (AWS). RDS allows you to store and organize data using several data set motors. It also aids in duties related to social information base management such as information transfer, reinforcement, recovery, and repair.
5) Robot Simulation Cloud: Based in simulator technology, this module gives customers access to robots simulation operating services in the cloud. You can use these services to learn and train operators,test configuration solutions; and do large-scale robot deployment studies.
6) Robot Control Based on Algorithm: This is the fundamental software function that allows the robot to function, and it is utilized in the cloud to imitate the robot's inner working logic.
7) Cloud-based ROS: Robot Operating System is the software that controls how the robot hardware works. ROS is a cloud-based platform that can be used to simulate how a virtual robot works, giving it the same capabilities as the real robot.
8) Robot Simulator: This is the container in which virtual robots are employed. To teach and test, cloud platform users can set up several simulator containers or different types of virtual robots in a single container.

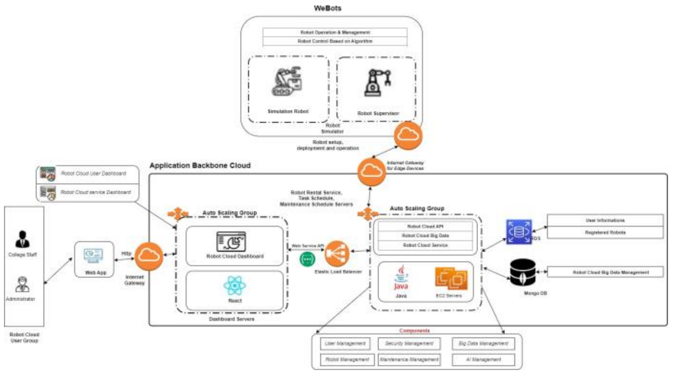


Fig. 2. Campus Security Robot System Infrastructure

IV. SIMULATOR DESIGN AND IMPLEMENTATION CSR

CRS utilizes Webots, an open-source 3D robot simulator that may be customized for a range of robot kinds and virtual environments. ”worlds” in which robots can operate. The simulator is a graphical user interface program that can be launched locally or remotely via a host server in fig -3. We constructed a virtual world that met the requirements of our work. It contained a table, classroom-style chairs, people, cars, a parking lot, a football field, and VIP game rooms. Figure 4 depicts one of the stationary Pioneer3at robots that can be found in the WeBots virtual world. An external controller connected to the CSRAPIs transmits signals to the virtual environment. In addition to manually controlling its direction based on object senser capabilities, the CSR Robot Controller APIs can initiate the execution of a script representing campus security.

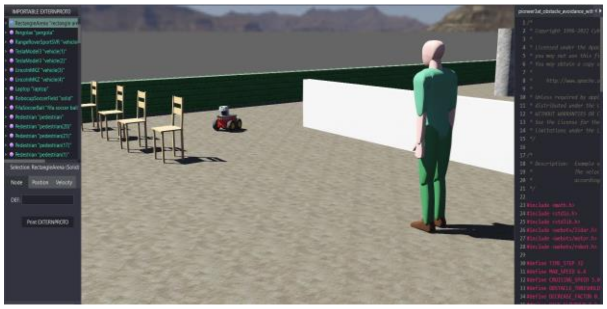


Fig. 3. Webots Virtual World for CSR

A. Simulator Design and Implementation

The campus security controller makes sure that robot orders from the customer portal are sent to the right business.

Fig. 4. Pioneer3at robot

This chooses a robot that is available and working at the location given. As a simulation, the request sends out a virtual robot into the Webbot’s virtual world that is currently being used. So that it can find people, the robot moves through a set of preprogrammed positions. Figure 6 shows how the Pioneer3at robot’s separate motors can be used to control the direction of objects and pitch, the forearm’s pitch and twist, and the gripper’s movement, twist, and opening and closing. To give users full control, the Robot Controller makes each of these movements available as its own API.

Figure 5 shows the CSR APIs that can be used to manage robots, such as renting, tracking, and listing active robots, as

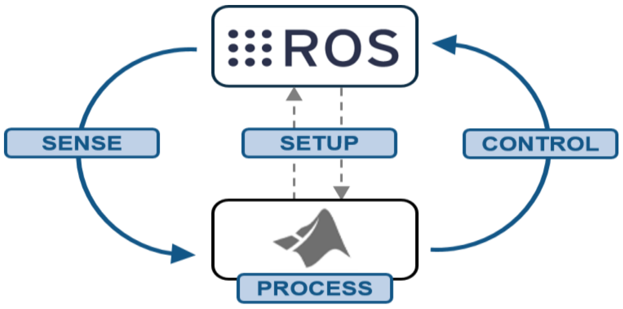


Fig. 5. Virtual robot movement support

well as manually controlling a virtual robot with a single motor. So that we could show a sample job and not a complicated one, we turned making coffee into a series of simple steps. In a more realistic setting, the sequence of movements would be more complicated, and the Webots’ virtual environment would include more things that looked like a coffee-making machine. However, since adding scripts to the coffee-making process can make it more complex, we place connectivity and the ability to control robots through a cloud-based system at the top of our list.

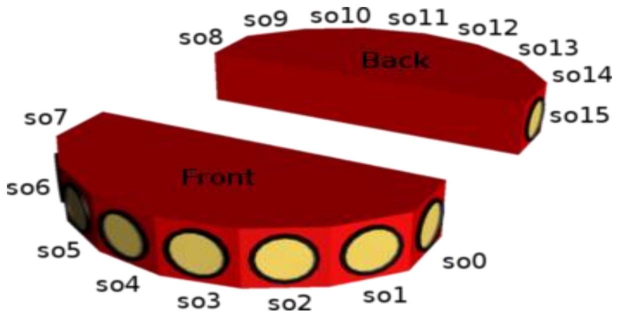


Fig. 6. Pioneer3at robot Sonars

shows that the robot has a few different states that it can be in. When the robot is made, it goes to a state called ”available.” When it is turned off, it goes to a state called ”terminated.” If a robot that is available is rented, the robot is turned on. Once the robot is turned on, it can be set up. When the robot starts running, it goes into a state called ”running,” which can be stopped and started again.

B. Simulation Connectivity Design

CSR was developed and implemented primarily to facilitate the control of simulated robots. In other words, CSR apps don’t engage in any sort of real-world communication with physical robots. Instead, they engage with computer-generated worlds. In order to accomplish this, the CSR APIs in the backend were designed to communicate with a particular WeBots virtual environment. Due to limitations with the WeBots simulator, our team had to devise our own method of sending out many robots because it could only run one virtual environment at a time and could only talk to a controller that was attached locally. Webots were used to get around these obstacles. Running code in a Docker container with its controller through a dedicated port This allowed the container to represent a unified WeBots universe. The use of many containers within a very potent EC2 instance in the background allowed for the final robot execution to be sent to any one of the running worlds. Figure 7 displays these interrelationships.

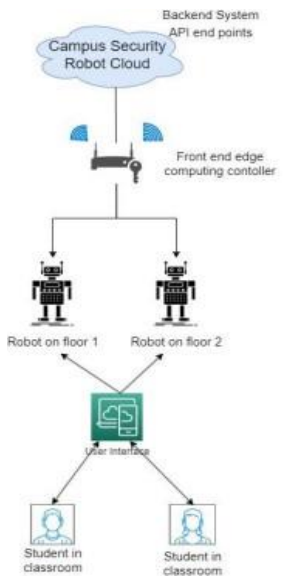


Fig. 7. Communication between robot simulation and users

Similarly, a less complex arrangement than the one described was implemented as well. and made sure that the goals for creating and testing a unified cloud environment for CSR had been met. Figure 8 shows that the developer's local machine and the AWS-deployed system can talk to each other in both directions. Remotely operated robots are known as ”websites.” when user participation with the virtual robot is required for the application's functionality. The central CSR APIs make internet calls to the local machine, which then provides the necessary responses. To set up this kind of link, port forwarding was set up on the local machine, and a public IP address was used to connect it to the right AWS service. The local machine stands in for the powerful G4 EC2 instance we discussed earlier.

V. Cloud Data Design and Implementation

A. Database Design

A database design is a set of steps that help a business create, implement, and keep up its data management system. When you design a database, the main goal is to create physical and logical models of how the proposed database system will work. A good process for designing a database follows two rules. Redundancy is the first rule of making a database. It wastes space and makes it more likely that mistakes and wrong matches will happen in the database. The second rule is that you should care about how accurate and

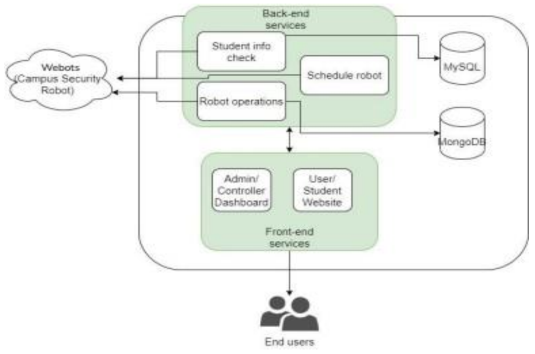


Fig. 8. Communication between Components

complete the information is. If there is wrong information in a database, it will lead to bad analysis and reporting. So, it can lead people making decisions astray and hurt a company's performance. Because of this, it's important to follow some rules when making the database for your organization. So, there are a few important steps to make sure that your database design is good:

- It divides your data into tables based on where each subject is located to get rid of duplicate data.
- To connect the data in the tables, you need information about the database.
- guarantee and back up the accuracy and dependability of data.
- functions with the database operations in a loop

B. Cloud DB Design and Implementation (SQL DB Design)

A database is a collection of bits of information that are organized in a way that makes it easier to find and look into relevant information. A database that was made well has accurate, up-to-date information that can be used for analysis and reports. The database design is very important for making sure that queries work well and that information is consistent.CCRS database design is based on the entity relationship diagram which is shown in the below. We have 3 major types of users : Student Users, Admin Users, and College staff. Each user type is connected to different entities based on the needs The main tables in the above RDS SQL database are :

- Admin : He is responsible for creating and initializing the robots. He will be configuring the type of robot that needs to be created. Also will be responsible for generating the billings for all the robots that are placed in the network.
- Staff : It will be the college staff in our case who will be using the robot services. He will be responsible for commanding the robot, what tasks need to be done and the scheduling of the robot.
- Student : Students will be acting as clients in our system. They will be interacting with the edge-based robots. All the information related to students will be stored in this table.

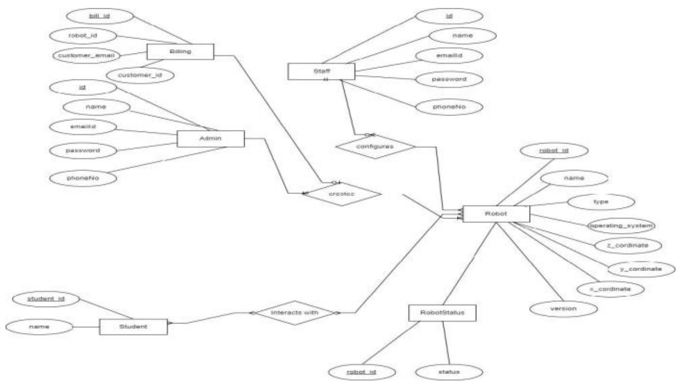


Fig. 9. Db Diagram

- Robot : This is the major component of our system which will represent the edge based robot which will be stimulated in AWS Robomaker. This table will contain all the information related to the robot and will make sure that the state of the robot is stored in a proper way.

C. Cloud DB Design and Implementation (MongoDB Schema)

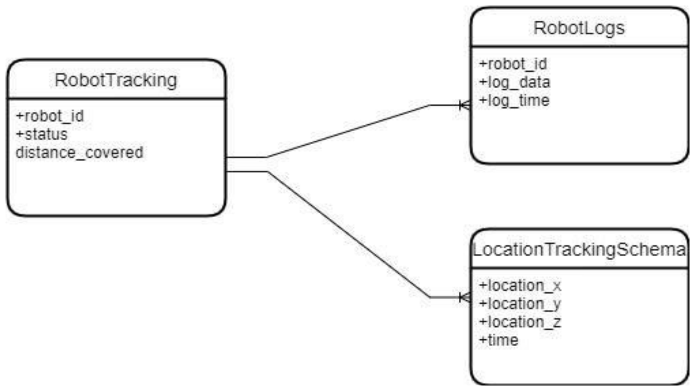


Fig. 10. Db Diagram

The above diagram shows the data that is going to be stored in our NoSQL database, which is mongoDB. The major schemas that will be present in our database are :

- Robot Tracking : we will be storing all the information related to our robot trips and tasks that our edge based robot has performed.
- RobotLogs : This information will be related to each entry corresponding to robotTracking. We will be getting information from AWS RoboMaker.
- LocationTrackingSchema : This will be responsible for getting the current location of any particular robot at a particular time. We can know where a robot was present at that time. This information can be used for bill generation purposes

VI. CLOUD SYSTEM DESIGN AND SERVICES

A. Cloud Based System Communication Design

In a cloud setup, the system communication design diagram shows how the system parts, which are mostly front-end and back-end parts, talk to each other. In Fig. 7 and Fig. 8, you can see two views of a communication system between these two components in our project.
The backend and frontend services are hosted on AWS EC2. AWS RDS is used to store relational data like user info, etc. The MongoDB Cloud is used to store robot-related data. End users interact with the frontend UI, which interacts with the backend server that has APIs for interacting with the deployed robot. It also connects to both databases and can give the user information in the format they want if they ask for it.

B. System component-oriented function architecture design

The function design depicts what functions will every component perform in order to meet project requirements. You can see in Fig 11, the campus security robot has two major components:

1) Functioning of the robot - what will the robot do? The main task of the robot is to validate students in a classroom. For that, it needs to collect information (like student ID number) from the student. Next, it needs to check the student ID with the data in the database (whether the student is enrolled in this class or not). Once the student is verified, it sends an alert saying the student is validated and marks his/her attendance.
   The robot will be set to go into certain classrooms at certain times throughout the day. The robot's states will be active, processing, completed, and error.
2) Management and monitoring of the robot - For managing and keeping an eye on the robot, the administrator can create, update, and delete the robot. The administrator also has control privileges like scheduling and updating the schedule of the robot. For monitoring, the admin can see the current status, the activity performed by the robot, and see a dashboard with some insights about the robot.

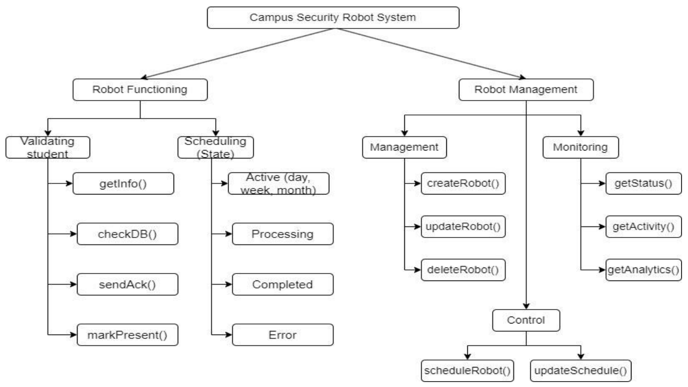


Fig. 11. Component-oriented Function Diagram

To see a granulated view of the components and their connectivity, we can look at the Deployment Diagram given in Fig 12.
Our Robot cloud service consists of four major components:

- The web application which is supposed to used by the staff and the administrator to configure robot and give the robots particular tasks that needs to be done
- The application is supposed to communicate with the aws cloud server. This server is the major component of our project. This server is responsible for all the backend task and communication between the database. Here, we have services that are responsible for robot Management, User Registration, Security management, and big data management
- We have a component called AWS Robomaker. We are treating it as a black box and will be using it to stimulate our virtual robots and virtual environments. Our backend services will be connected to AWS RoboMaker.
- Next we have the database servers. Here we will be using SQL databases for storing all the user information and the state of robots and the information related to billing and other modules. We also have a MongoDB database which will be storing the logs and statistics from AWS RoboMaker. All the data in Mongo will come from AWS RoboMaker via our backend server. This data is used for logging purposes.

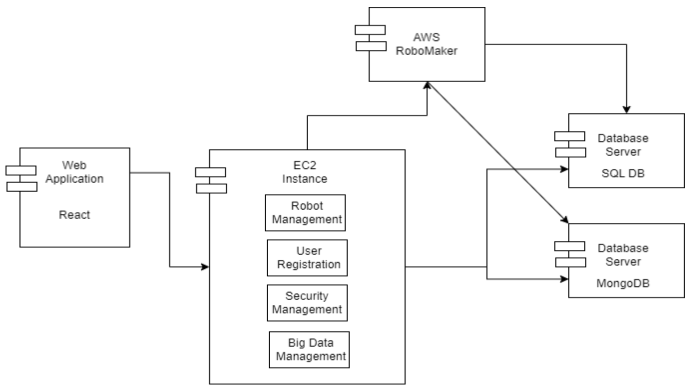


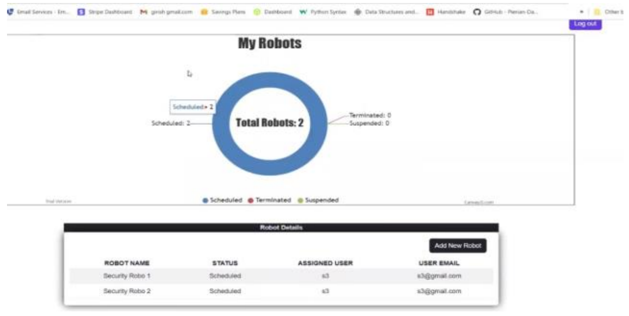
Fig. 12. Deployment Diagram

VII. SYSTEM GUI DESIGN AND IMPLEMENTATION

A. *Customer(Staff) Portal*

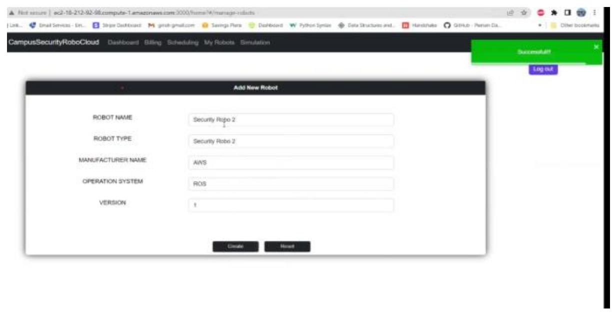
The Customer Portal's main UI components are as shown in the figure. Here the user will be able to login and will be able to perform various task such as create a robot, schedule a robot, view its billing etc. The screenshots show the complete flow of what is the functanality provided for the users in our application. Here we are allowing our customer to create a type of robot and see all the robots that are available for particular user. The user is then able to create a schedule for the robot using our portal. The given diagram tells us how a schedule is been created for the user.

Here the successful entry for schedule has been created. After this the user can check his bills in our bill section. Also we have provided the option of paying the bill. For this we have integrated stripe API in our application. The diagram shows how our payment API looks like in realtime.



ROBOT NAME	STATUS	ASSIGNED USER	USER EMAIL
Security Robo 1	Scheduled	s3	s3@gmail.com
Security Robo 2	Scheduled	s3	s3@gmail.com

Fig. 13. Staff Robo Dashboard



ROBOT NAME: Security Robo 2
ROBOT TYPE: Security Robo 2
MANUFACTURER NAME: AWS
OPERATION SYSTEM: ROS
VERSION: 1

Fig. 14. Staff Robo Creation Page

B. *Admin Portal*

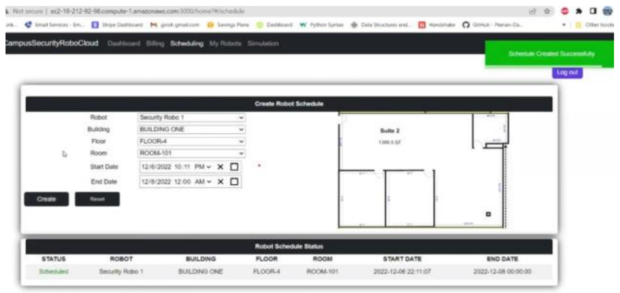
Admin has the task of configuring the users and it can be done by access management tab in our application. The figure shows the access management tab for Admin users. Once the access is been given the new user will be able to login to our system. After this the admin also has the capabilities to generate bills for all of the users. The diagram (Bill generation by ADMIN) shows this usecase also in detail.

VIII. EDGE STATION DESIGN AND SIMULATION

A robot simulator can be very helpful when developing security protocols. However, there are not many planning or simulation systems for security robots. To do so, we need to set up systems at both the high and low levels to control the actions and movements of the robots. The Security Robot Simulator (SRS) was built on Amazon Web Services (AWS). Robomaker is equipped with a graphics engine and a stablephysis engine for robotics modeling. Redundancies improve the robot's trajectory in an unstable network, as shown by simulation and real-world results. To take over, non-open redundancies can use an approximation of the state they were in before. We use an adaptable and extensible simulator to program the robots to perform realistic tasks.

A. *Edge-Base Mobile Application Interface*

Edge computing demonstrates applications and services such as location services and radio network services. Future conversations will center on both options. Depending on a robot's connectivity to a web application, the Edge Computing System will plan and regulate its motions. On its control panel dashboard, the robot features a very perceptive security app.


CampusSecurityRoboCloud Dashboard Billing Scheduling My Robots Simulation

Schedule Created Successfully

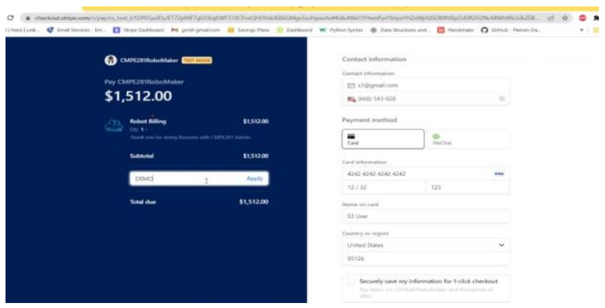
Create Robot Schedule

Robot	Security Robo 1
Building	BUILDING ONE
Floor	FLOOR-4
Room	ROOM-101
Start Date	12/6/2022 10:11 PM
End Date	12/8/2022 12:00 AM

Robot Schedule Status

STATUS	ROBOT	BUILDING	FLOOR	ROOM	START DATE	END DATE
Scheduled	Security Robo 1	BUILDING ONE	FLOOR-4	ROOM-101	2022-12-06 22:11:07	2022-12-08 00:00:00

Fig. 15. Schedule Creation by Staff


CMPE281RoboMaker TEST MODE

Pay CMPE281RoboMaker

$1,512.00

Robot Billing	$1,512.00
Subtotal	$1,512.00
Total due	$1,512.00

Contact information: s3@gmail.com, (668) 543-828

Payment method: Card

Card information: 4242 4242 4242 4242, 12 / 32, 123

Name on card: S3 User

Country or region: United States, 95126

Securely save my information for 1-click checkout

Fig. 16. Payment page for Staff

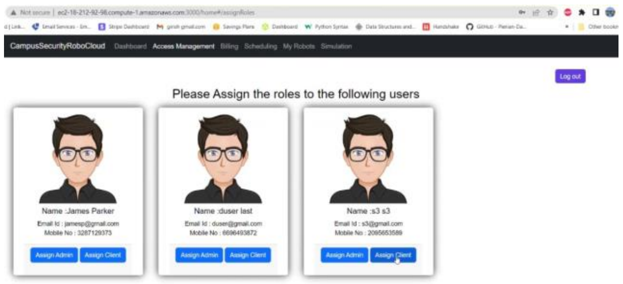
Selecting "Start Simulation" will connect you to the online control panel of the robot. Once the robot and web application have been paired, the user can instruct the robot to push the "X/Y-axis" button to retrieve the most recent values from the angle direction sensors. Changing an axis' value enables the user to alter the direction. Once the "Go to Coordinate" option has been selected, the sensor readings are presented in the Set Point column. The alert value and threshold for when a user receives an alert can be modified by the system administrator for each sensor. The presence of anonymous students in the classroom, at school events, or anywhere else on campus will not activate any alarm

B. *Edge-Based Mobile Business Logic*

Here is an order of the most important topics: Unauthorized users are unable to access the portal without authentication, putting your data and system's integrity at risk fig-19.

1) : Session should terminate after a specified period of inactivity. The user should be prompted to log out after a preset amount of inactivity.

2) : Another pressing business matter. If this is not done, unauthorized users can damage the data. Multiple HTTP requests from a single public IP address constitute a denial of service attack. With the aid of the decision support system, separating SRS services is simplified. One thing: they make decisions through conversation. The actions of others have no effect on how services operate. This indicates that robot services can be utilized multiple times. The robot explorer task service delivers a message to the hardware driver service in order to control a real robot. It is also difficult to transfer communications from one provider to another. Two bumper sensors choose which of two messages given simultaneously to examine first.


CampusSecurityRoboCloud Dashboard Access Management Billing Scheduling My Robots Simulation

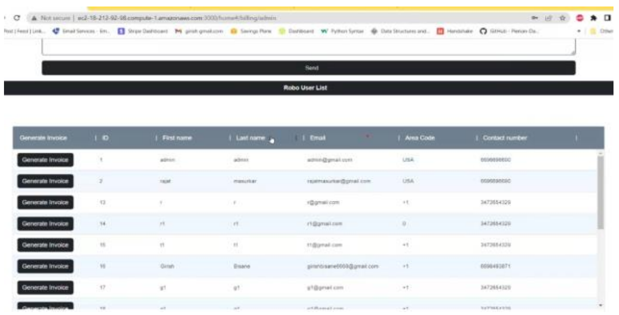
Please Assign the roles to the following users

Name :James Parker
Email Id : jamesp@gmail.com
Mobile No : 3287129373

Name :duser last
Email Id : duser@gmail.com
Mobile No : 6696493872

Name :s3 s3
Email Id : s3@gmail.com
Mobile No : 2095653589

Fig. 17. Access management for ADMIN


Robo User List

Generate Invoice	ID	First name	Last name	Email	Area Code	Contact number
Generate Invoice	1	admin	admin	admin@gmail.com	USA	6696696600
Generate Invoice	2	rajat	masurkar	rajatmasurkar@gmail.com	USA	6696696600
Generate Invoice	13	r	r	r@gmail.com	+1	3472654329
Generate Invoice	14	r1	r1	r1@gmail.com	0	3472654329
Generate Invoice	15	t1	t1	t1@gmail.com	+1	3472654329
Generate Invoice	16	Girish	Bisane	girishbisane6669@gmail.com	+1	6696493871
Generate Invoice	17	g1	g1	g1@gmail.com	+1	3472654329

Fig. 18. Bill generation by ADMIN

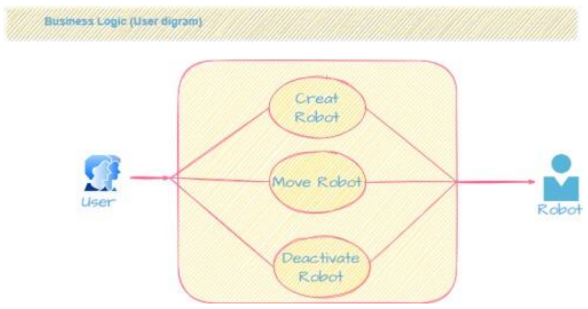
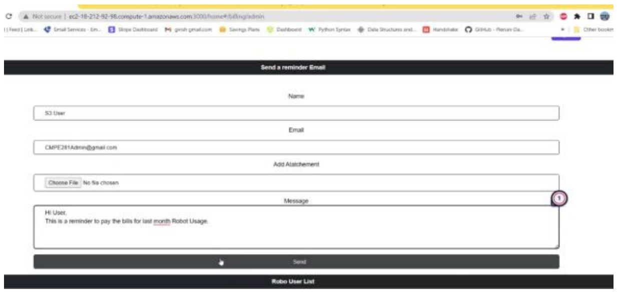


Fig. 19. Business Logic (User Diagram)

IX. System Application Examples

The following are system application examples which can come across our application and their illustrations. The below is the admin page for n number of generated robots. The admin will have to manage all the robot stimulation's. He will have to make sure that only a particular number of robots are assigned to the same user. If not he will have to generate the alerts for the user and send the reminder emails to customers so as to pay their bills for all the assigned robots. We have provided the functionality to send reminders to the users. The diagram for email reminders depicts our implementation of the mentioned functionality


Send a reminder Email

Name

S3 User

Email

CMPE281Admin@gmail.com

Add Attachement

Choose File No file chosen

Message

Hi User,
This is a reminder to pay the bills for last month Robot Usage.

Not secure | ec2-18-212-92-98.compute-1.amazonaws.com:3000/home#/billing/admin
Send
Robo User List

Fig. 20. Email Reminder

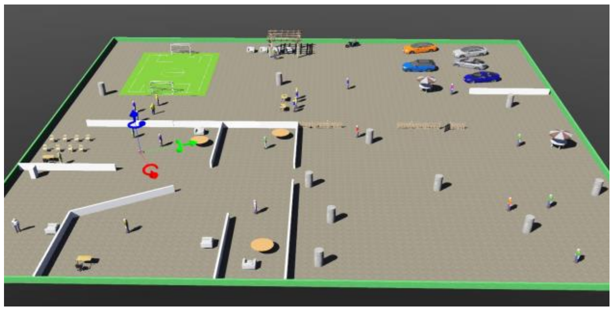


Fig. 23. CSR Simulation

X. SYSTEM PERFORMANCE EVALUATION AND EXPERIMENTS (EXTRA POINTS)

The application was deployed on Cloud and tested against few factors. We can see in Fig. 21 and Fig. 22 the application was tested against a connection pool of 100 users and without connection pooling with 500 users using JMeter. It was found out that the application is robust and can handle huge capacity of concurrent and non-concurrent load.

The application gives satisfactory outputs with minimal resources and can perform even better with better load balancing scheme and larger instances.

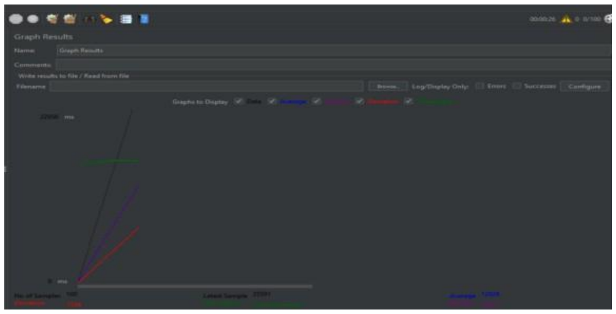


Fig. 21. Testing without connection pooling - 100 users

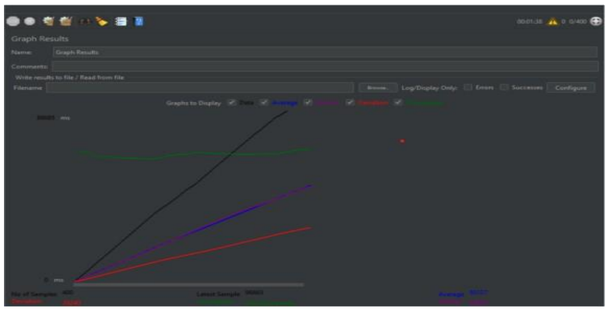


Fig. 22. Testing without connection pooling - 500 users

XI. CONCLUSION

The Campus Security Robot is a robot simulation application that can help school, University and other official campuses secure by helping deploy robots in the campus that can detect people, objects around and when programmed correctly, the robots can verify identities of people around. The robot can send notifications to the admin/security about unauthorized people inside campus and also keep an update of their status through constantly updating databases.

The Robot simulation system is deployed on AWS Cloud stack which includes EC2, S3, RDS, ECR, Robomaker and Mongo Cloud. It is scalable and error tolerant as the deployment has load-balancers and auto-scaling features. The application consists of frontend made in ReactJS, deployed on an EC2 instance and a backend in Java deployed on multiple EC2 instances. The backend interacts with the robot controller to control robot actions and the billing system helps keep track of the bill of users.

This simulation of campus security robot can create and maintain the trust and safety of students, staff and workers inside a college campus. We highly recommend using this system to build robots that can create a safe campus environment. As we progressed in building campus security robot application, we gained knowledge of system design, IAM in Cloud, multi-tenancy, full-stack development, REST framework and we have a broader understanding of ReactJS, Java and their connectivity with each other and databases.

The deployment of the application on EC2 instance and the ability to scale, load-balance, and monitor the cloud resources was a great way to learn Cloud Technologies and its insights. The robot simulation also helped us understand the wide ranging things that we can do with the power of Cloud. Moving forward, we will research and develop solutions that can be scalable and accessible to all through cloud.

REFERENCES

[1] https://www.researchgate.net/publication/361727002_Robot_Cloud_Computing_ A_Tutor ial_-Infrastructure_service_needs_and_challenges.
[2] https://docs.aws.amazon.com/robomaker/?icmpid=docs_homepage_robotics.
[3] https://aws.amazon.com/solutions/case-studies/robotcaresystems/
[4] https://aws.amazon.com/solutions/case-studies/irobot-case-study/
[5] https://aws.amazon.com/blogs/iot/utilizing-aws-services-to-quickly-build-solutio ns-for-ro botics-use-cases/
[6] https://aws.amazon.com/robomaker/customers/
[7] https://aws.amazon.com/blogs/robotics/tag/aws-robotics-aws-robomaker-simulati on/
[8] https://aws.amazon.com/blogs/robotics/run-any-high-fidelity-simulation-in-aws-r obomaker-with-gpu-and-container-support/
[9] https://docs.aws.amazon.com/whitepapers/latest/aws-overview/robotics.html?sc_i channel =ha&sc_icampaign=acq_awsblogsb&sc_icontent=robotics-resources
[10] https://aws.amazon.com/blogs/robotics/aws-robomaker-now-supports-ros2-foxy-f itzroy-f eaturing-navigation2/