System Design and Architecture Document

Campus Security Robot

CMPE 281
Cloud Technologies

Submitted to:
Dr. Jerry Gao

Prepared by - Team 17

Vinti Jain (015955446)
vinti.jain@sjsu.edu

Girish Bisane (016650348)
girish.bisane@sjsu.edu

Rajat Prashant Masurkar(016044015)
rajatprashant.masurkar@sjsu.edu

Venkata Siva Prasad Kakkera (015935101)
venkatasivaprasad.kakkera@sjsu.edu

1. Introduction **3**
1.1. An Introduction the Project: 3
1.2 Objectives: 4
1.3. Expected Outcomes: 5
2. System requirements and analysis **6**
2.1 Purpose 6
2.2 Document Conventions 6
2.3 Intended audience and reading suggestions 6
2.4 Project Scope 7
2.5 References 7
2.6 Overall description 7
2.6.1 Product Perspective 7
2.6.2 Product Features 8
2.6.3 User Class And Characteristics 8
2.6.4 Operating Environment 9
2.6.5 Design And Implementation Constraints 9
2.6.6 Assumption Dependencies 10
3. System infrastructure and architectures **10**
3.1. Cloud-based system infrastructure (using a system infrastructure diagram) 10
The overall architecture of the robot cloud proposed in this work is made up of the following parts: 11
Different user groups 12
3.2. System component-oriented function architecture design 12
3.3 System deployment infrastructure (using a system deployment diagram) 13
4. Cloud-based system design and component interaction design **15**
4.1. System database design 15
4.2. Cloud-Based System Communication Design 18
5. Project development plan and schedule **19**
5.1. Project team and roles 19
5.2. Project schedule 19
6. References **20**

Project team and roles

Team Member	Responsibilities
Rajat Prashant Masurkar	***Database management component(s)*** ***Responsible for creating the SQL schema*** ***Backend services related to those schema + backend***
Vinti Jain	**Online system dashboard** **Front-end + backend**
Venkata Siva Prasad Kakkera	***Edge-based mobile robot* (*Simulator) + backend***
Girish Bisane	**User service and billing management + backend**

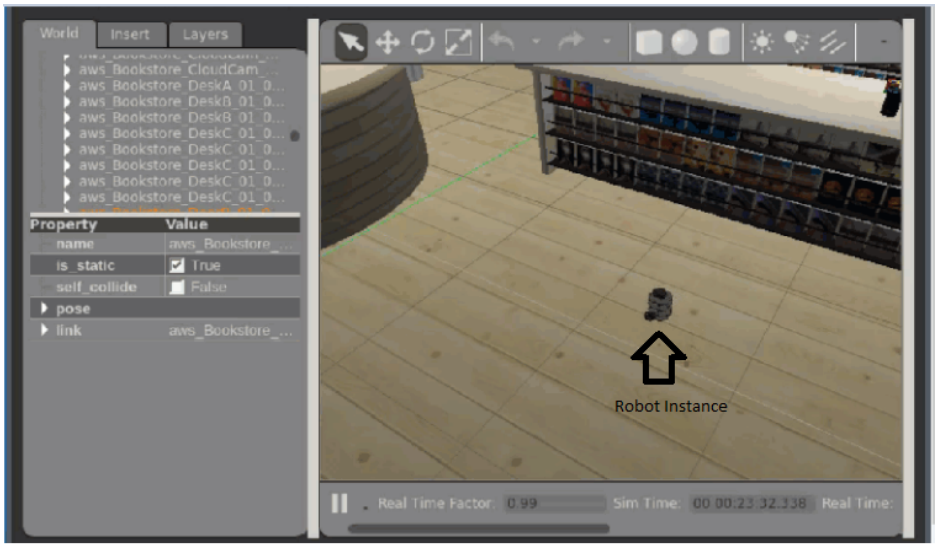
1. Introduction

1.1. An Introduction the Project:

There is great potential in the cloud robotics market as it has been one of the booming industries in recent years. This substantial growth has encouraged more and more research in this field. Still, there is a lot to explore in the field of robot cloud that supports the whole business model.

With the advancement of technology in the fields of robotics and artificial intelligence, the use of service robots is on the rise in our day-to-day lives. There are many challenges in the robotics field, including budget constraints while building robots. There is no complete, free and open-source robot cloud service available. Here, we are trying to solve the problem by building a system that is cloud-based that will support robot services. Some of the tasks are registering online from a remote location, setting up the robot, keeping track of its state, controlling it, following it, and keeping an eye on it.
In this project, we are going to focus on creating a cloud-based service/platform for mobile robots that will help in communication between the robots and a cloud server to provide all the above services for a set of robots.
cloud based
We will be creating a robot cloud service to handle the operations and business model for school campus security robots. This will be a platform which will be responsible for each and every operation required to create, configure, command, and monitor this category of robots.

The below are some examples of the simulated robot environment from AWS Robomaker :



1.2 Objectives:

The main goal of this project is to build and use a platform for edge-based robots that is based in the cloud. We aim to provide communication between the end robots that are in the environment in an active state and the cloud server.

The cloud platform will also be responsible for user management.
The type of users we are targeting are

1. Robot clients -> Students

2. Cloud Staff -> College Staff
3. Robot System Admin -> Administrator

This cloud still helps with different edge-based robot service functions, like registering robots, keeping track of our edge-based robots, setting up robots, controlling and monitoring them, and registering robots.

We will also be a cloud platform that needs to offer user account services like managing user accounts, robot catalogs, robot services, and billing for robot services.

1.3. Expected Outcomes:

The goal of the project is to build a fully functional cloud-based platform with all the parts, like the dashboard, database management, robot management, and user management. Then comes the phase in which we deploy our virtual robot using AWS robomaker.

The college staff should be able to tell our simulated robot how to move around the school campus, and the robot should be able to talk to its customers, who are the students. The below is the example of environment that we want our simulated robot to move in

The staff will give the coordinates to the edge-based robot using our cloud service. The robot will be moving to that given location. It will receive its instructions from the robo-cloud that we are creating.

2. System requirements and analysis

2.1 Purpose

This document is a list of the requirements for building and using a robotics cloud-based service platform. This cloud service platform will be able to handle data transfer between the cloud-based robot service and those on the edge when mobile robots are used for services. This cloud will help with many remote robot service tasks, like registering, configuring, tracking, controlling, and monitoring remote robots. We will also offer the cloud platform services that robot users need, such as billing for robot services, catalog management, and account management for robots.

2.2 Document Conventions

We will be using the following conventions throughout this document

MRC	Mobile Robot Cloud
DDB	Distributed Database
CSR	Campus Security Robot

2.3 Intended audience and reading suggestions

We are creating a prototype for a school campus security robot, and this robot is supposed to work only on college premises. We are implementing this project under the guidance of college

professors. This project is useful to public MRC users, who are the client users for mobile robots, which are connected to a MRC cloud to support remote services.

2.4 Project Scope

The Robot cloud platform is meant to make life easier by running daily errands, keeping college campuses safe, and supporting a variety of remote robot service functions, such as online registration, configuration, control, tracking, and monitoring of remote robots. The system will use AWS Robomaker to simulate robots and SQL DB to keep information about user accounts and robots that have been registered. We will use a NOSQL database to get access to the robots' different service data and records. Above all, we want to give users a pleasant experience and the best prices possible.

2.5 References

- Srs_template-ieee.doc
- https://krazytech.com/projects

2.6 Overall description

2.6.1 Product Perspective

The final product will be a model of a college campus with a security robot. The robot has a few jobs to do. It can either confirm that a student is real or that they were in class or at an event. The robot can go around the classroom, checking each student's student ID number and matching it with its database. At the same time, they can check their attendance. The robot can stand at the entrance of an event and ask students to scan a QR code and fill out the registration details like confirmation number and date of attendance.

2.6.2 Product Features

The campus security robot can do

1. Move around a class, ask students to enter their student ID number, check the database if this SID is enrolled in this particular class, validate it, and send back an acknowledgement. Also mark the student's attendance for that day. The robot can be scheduled to go to that class at a particular time every week. Once it is done with all the seats in a class, it can then move to another class.

2.6.3 User Class And Characteristics

The system will support two types of user privileges -

Public MRC users: They are the client users for mobile robots (ex. students), which are connected to a MRC cloud to use remote services.

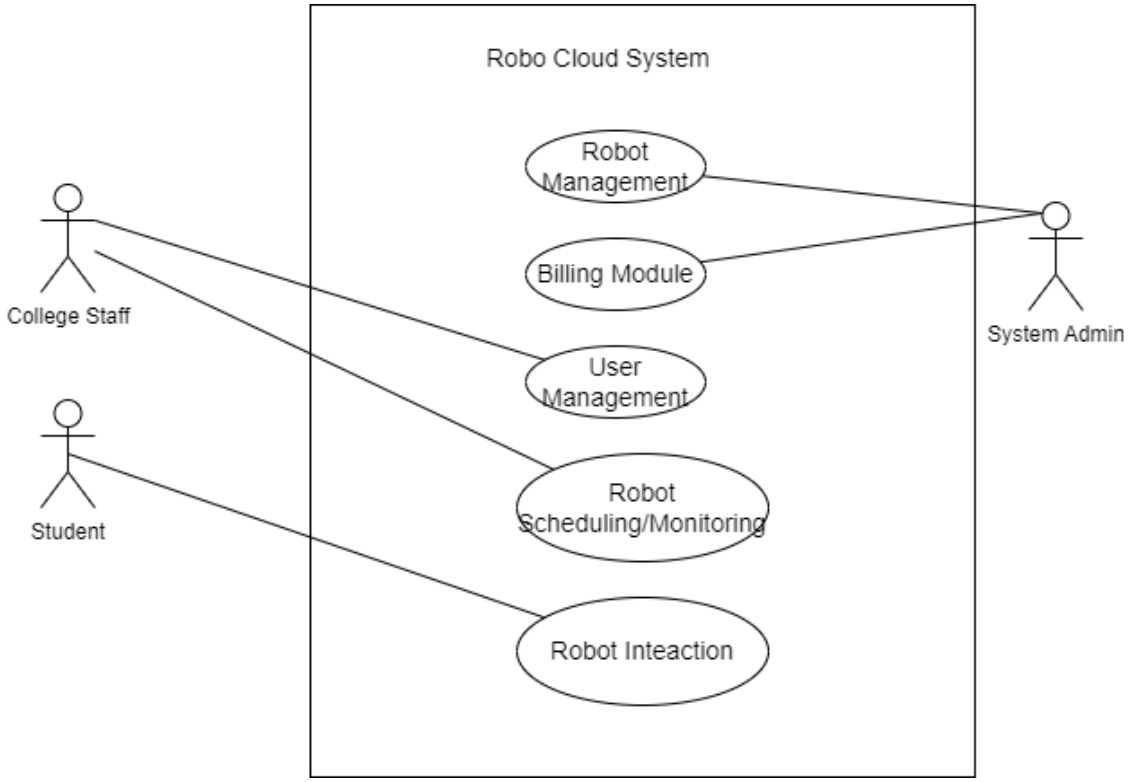
College administration and service staff – They are the system administration service staff at the back end. They conduct remote online tracking and monitoring for mobile disinfection robots.

The Public CSR User should be able to do the following functions -

1. Enter their id and mark the attendance for class via CSR.
2. Scan the QR code to show to get entry as a valid participant for any college event.

The College Admin and Service staff should be able to do the following functions -applications

1. Monitor and track the attendance of all students via the dashboard connected to MRC.
2. Upload the student database of valid recipients for events.
3. Receive an alert when an unauthorized person tries to trespass the event.



2.6.4 Operating Environment

The operating environment for the Robot cloud service platform is as listed below.

- Client/Server system
- Operating system: Windows.
- Browser : Safari, Chrome
- database: SQL/NOSQL
- platform: AWS RoboMaker

2.6.5 Design And Implementation Constraints

1. Connecting backend applications aws robomaker (At max 30 to 40$)
2. Frontend -> the input should come from the user/client. Environment (Exception handling)
3. Batch processing (daily process) -prof expectation
4. Databases - 2 dB, logs - generated aws mysql - user management,

5. Design -> billing modules, floor plans ,admin/user module , robo-booked featureWe will use a simulated robot instead of a real one, assuming that it will follow the same state-based process that supports the robot's configuration, control operation, and state tracking as the real one.
6. Trigger commands from backed -> which internally invokes and triggers moment of robos

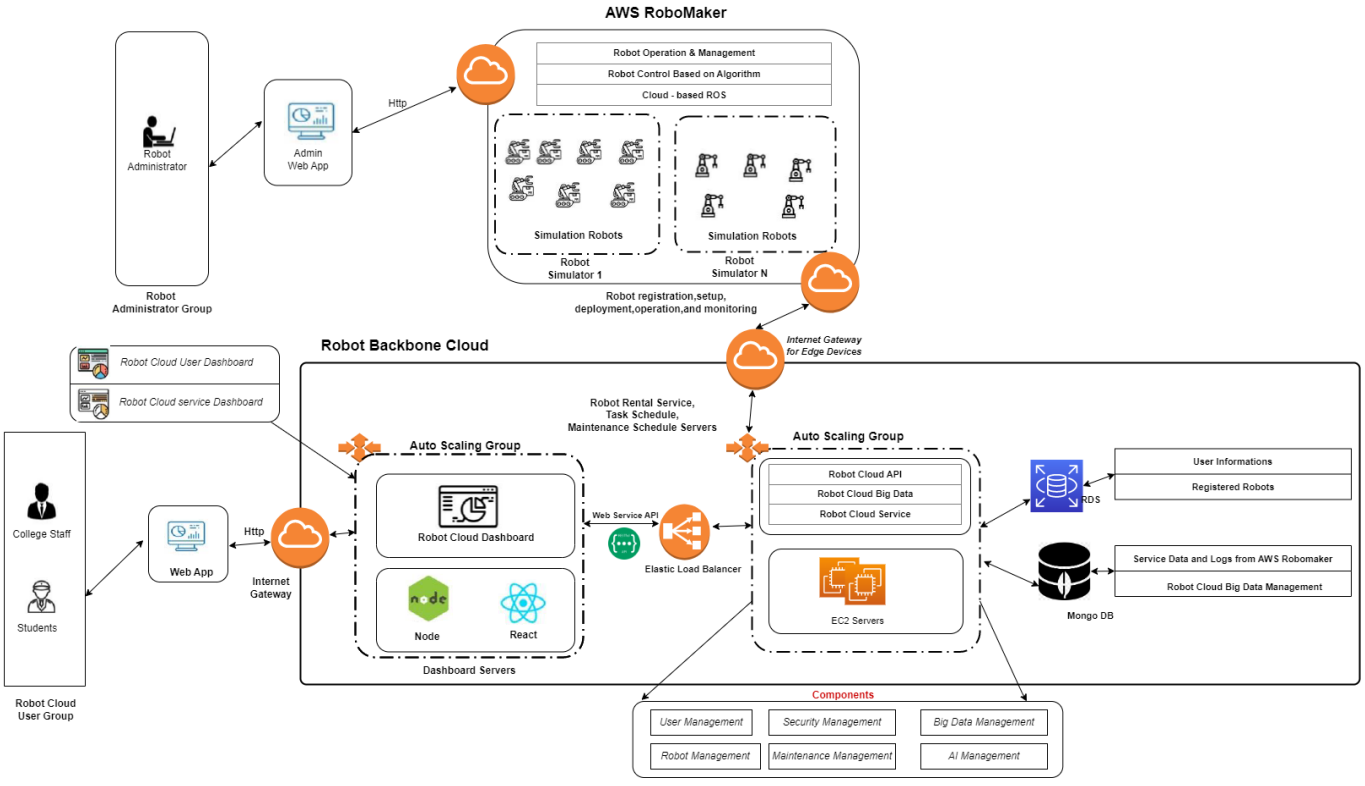
2.6.6 Assumption Dependencies

This Robot service application can be used for various things, among which we are focusing on the following applications.

It will let a client and a robot (or a simulated robot) talk to each other through an online interface or a mobile robot app.

3. System infrastructure and architectures

3.1. Cloud-based system infrastructure (using a system infrastructure diagram)



The overall architecture of the robot cloud proposed in this work is made up of the following parts:

- **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.
    - **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.
    - **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.
    - **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.
    - **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.
    - **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
        - **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.
        - **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.
        - **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.
    - **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.
    - **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.

- **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.
- **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.

Different user groups

1. Public MRC users : These are the users who will be interacting with the robot which is connected to the cloud. These users will give different tasks to robots and a robot will act accordingly.
2. System Administration Service Staff : These are the administrative users who are responsible for remote online tracing and monitoring for mobile disinfection robots.

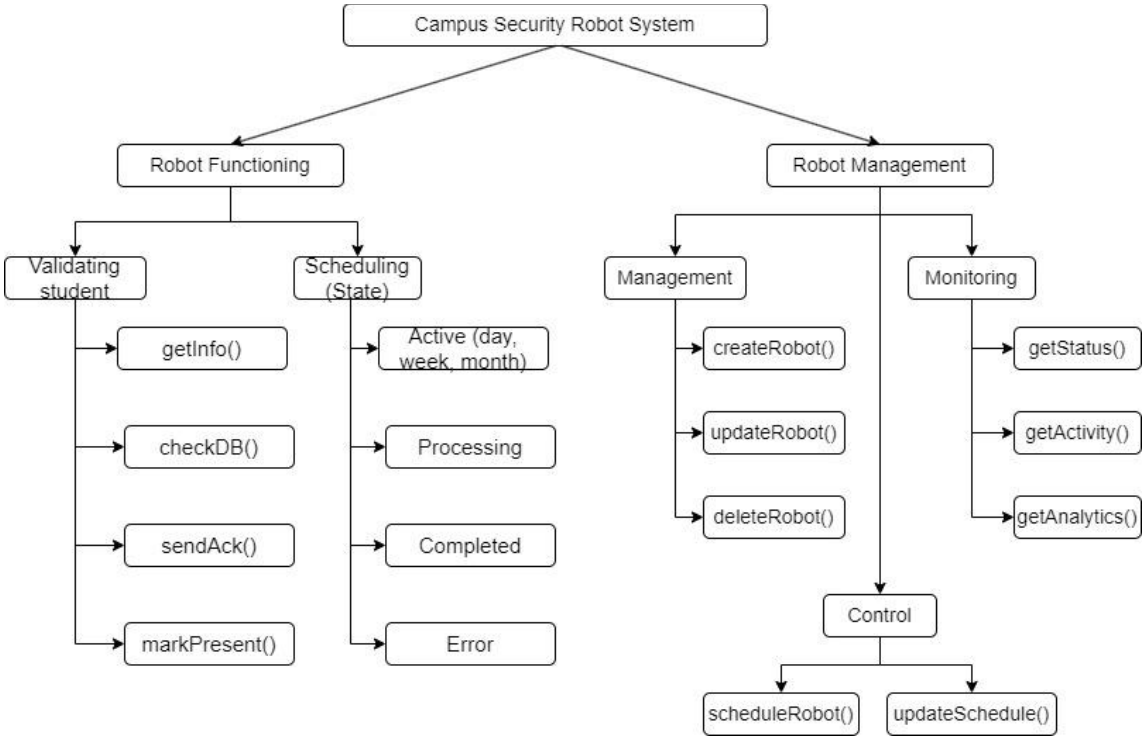
3.2. System component-oriented function architecture design

- The function design depicts what functions will every component perform in order to meet project requirements. Here you can see 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.

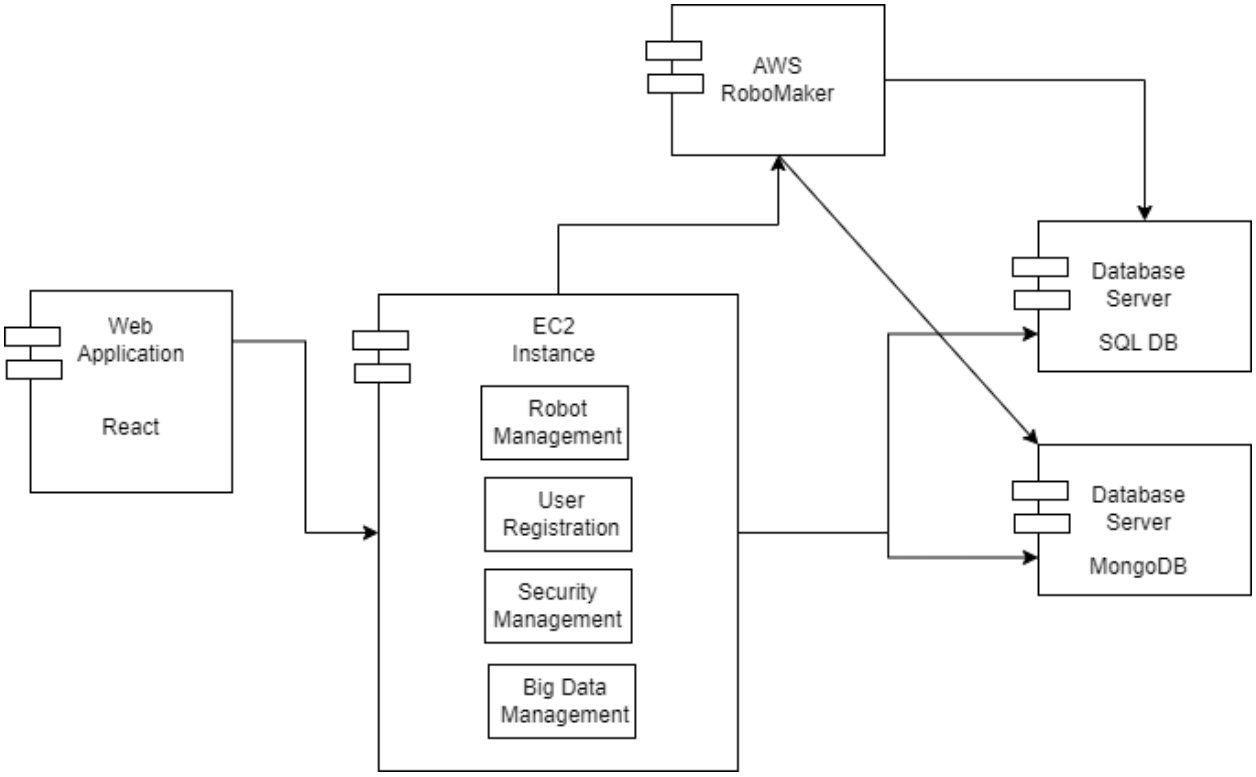
2. Management and monitoring of the robot:

   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.



3.3 System deployment infrastructure (using a system deployment diagram)

This shows the physical view of the system in terms of hardware boxes, how they are connected, and the users who use them.



Our Robot cloud service consists of four major components:

1. 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
2. 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
3. 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.
4. 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.

4. Cloud-based system design and component interaction design

4.1. System database 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.

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 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

The database design shows how information is planned, stored, and managed. To make sure the data is correct, create a database with only useful information. A well-designed database keeps information consistent, gets rid of unnecessary data, and makes the database work better.

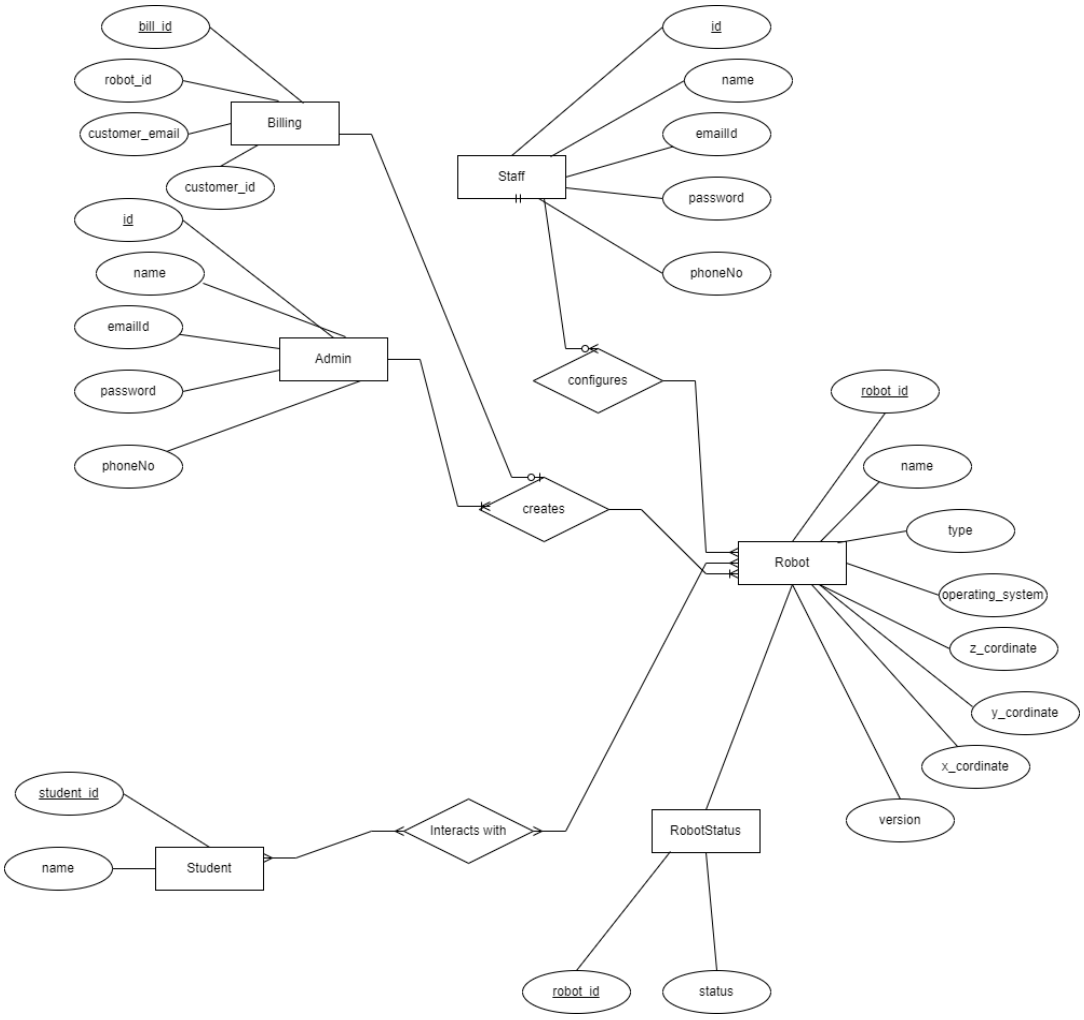
A database design method based on fields of study saves time. The structure of a table determines how reliable the data is, while primary and unique keys make sure everything is the same. Make a table of likely values with a key to avoid having to re-enter the same information. The main table is changed when a value changes. Performance is affected by design. A good database design is easy to understand and quick to execute.

Updates and maintenance are simple. Stored events, views, and utilities can be hurt by bad database architecture. There are steps to making a database. There is no need for order.

Requirement analysis, database design, and implementation make up the database development life cycle.

- First is planning for database development and an analysis of the information system's strategy. Second, it says what the scope of the proposed database is.
- This design shows the whole structure on paper, without any DBMS needs. The logical model is put into action in the physical modal model.
- Implementation involves importing old data into a new database. Tests. This stage finds problems with the system and makes sure the database needs are met.

The below diagram illustrates the database schema that is responsible for storing the information related to our custom roboCloud

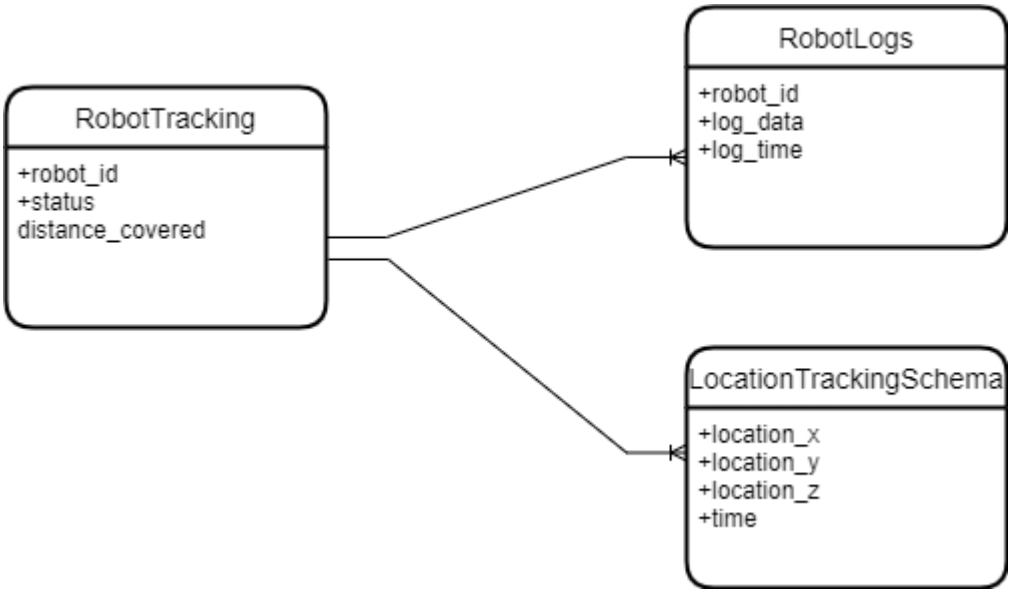


The main tables in the above RDS SQL database are :

1. 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.

2. 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.
3. 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.
4. 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.

MongoDB Schema

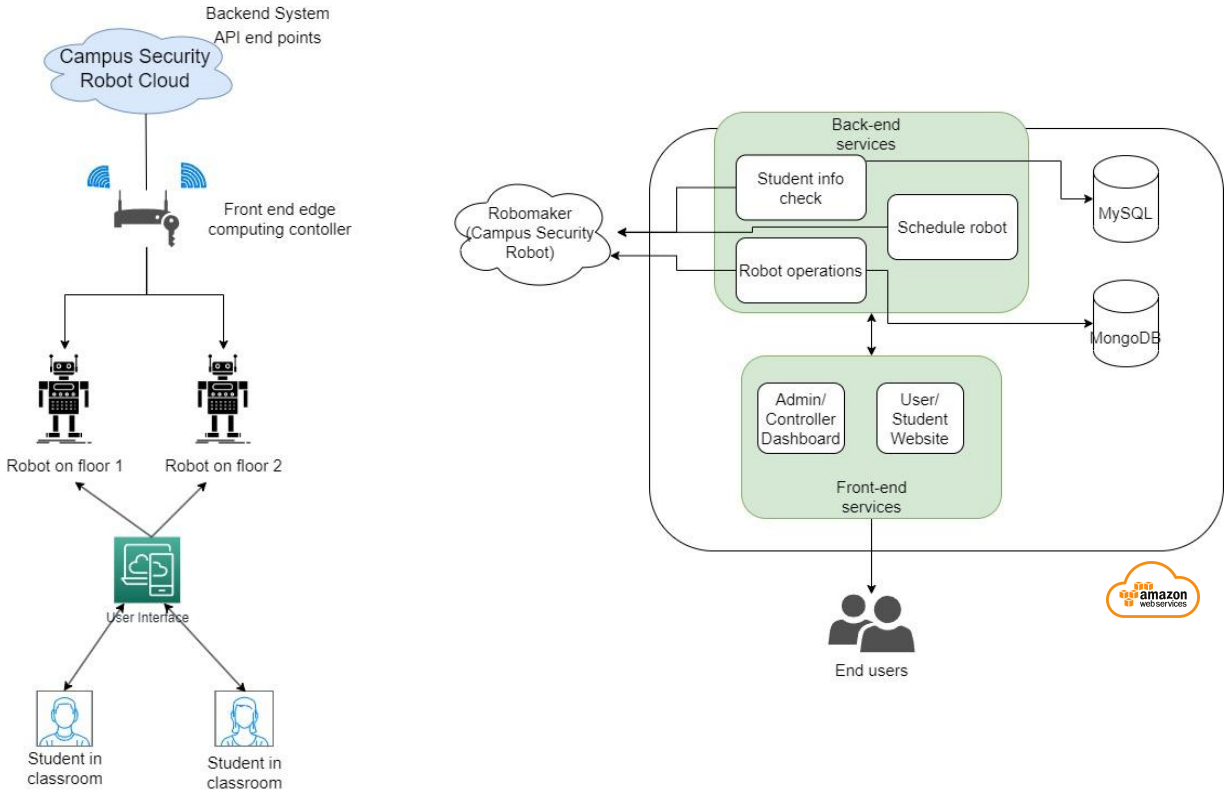


The above diagram shows the data that is going to be stored in our NoSQL database, which is mongoDB.
The major schemas that will be present in our database are :

1. Robot Tracking : we will be storing all the information related to our robot trips and tasks that our edge based robot has performed.
2. RobotLogs : This information will be related to each entry corresponding to robotTracking. We will be getting information from AWS RoboMaker.
3. 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

4.2. 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. Below, 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.



5. Project development plan and schedule

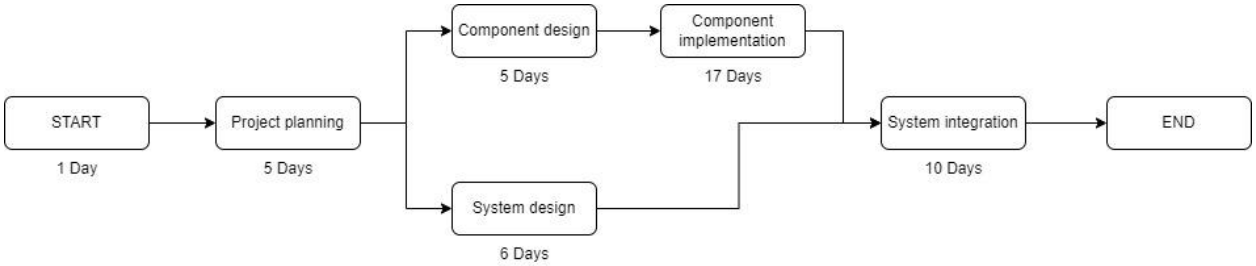
5.1. Project team and roles

Team Member	Responsibilities
Rajat Prashant Masurkar	***Database management component(s)*** ***Responsible for creating the SQL schema*** ***Backend services related to those schema + backend***
Vinti Jain	**Online system dashboard** **Front-end + backend**
Venkata Siva Prasad Kakkera	***Edge-based mobile robot* (*Simulator) + backend***
Girish Bisane	**User service and billing management + backend**

5.2. Project schedule

Task	Development	Date
Deliverable -1	Project planning, analysis and design document	Oct 17th
Deliverable -2	Component analysis and design UML diagrams	Oct 31st
Deliverable -3	Project independent component implementation	Nov 15th
Deliverable -4	Project system component integration and demo	Dec 6th
Deliverable -5	Final project video demo, Program, project final report & PPT	Dec 15th

Pert Chart



6. References

1. https://www.researchgate.net/publication/361727002_Robot_Cloud_Computing_A_Tutorial_-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-solutions-for-robotics-use-cases/
6. https://aws.amazon.com/robomaker/customers/
7. https://aws.amazon.com/blogs/robotics/tag/aws-robotics-aws-robomaker-simulation/
8. https://aws.amazon.com/blogs/robotics/run-any-high-fidelity-simulation-in-aws-robomaker-with-gpu-and-container-support/
9. https://docs.aws.amazon.com/whitepapers/latest/aws-overview/robotics.html?sc_ichannel=ha&sc_icampaign=acq_awsblogsb&sc_icontent=robotics-resources
10. https://aws.amazon.com/blogs/robotics/aws-robomaker-now-supports-ros2-foxy-fitzroy-featuring-navigation2/