CMPE 281-01: Cloud Technologies.

Deliverable 2: Component Design Document

Option #1: Robot Cloud (Food Delivery Robot)

Submitted to:

Dr. Jerry Gao

Date of Submission 10/23/2022

Submitted by Group 24

S.No	Name	Student ID	Email ID
1.	Chirag Arora	016726567	chirag.arora01@sjsu.edu

Contents:

Section 1: Component overview

1.1 Component purpose:

The function of the user service management component is to give users remote access to the cloud service platform's services. The user should be able to remotely configure and control robots by registering and logging into the cloud platform, and the service administrator should be able to monitor robot usage and charge the user appropriately.

1.2 Objectives:

The objectives of this cloud robot platform are as follows:

Admin:

1. Login into the system.
2. Charge the user in accordance with the billing criteria.
3. Create user-robot Platform-service analytics on the dashboard to monitor statistics relating to the number of users registered, the number of robots registered, the number of robots active, etc.

User:

4. Able to create an account and log in to the system
5. Being able to research the platform's already-available robot services
6. Configure and use the robot remotely by the user
7. Follow the billing in accordance with the billing requirements.

This component addresses all the goals outlined by Adobe, as well as their design, implementations, and the APIs needed to create the system to achieve them.

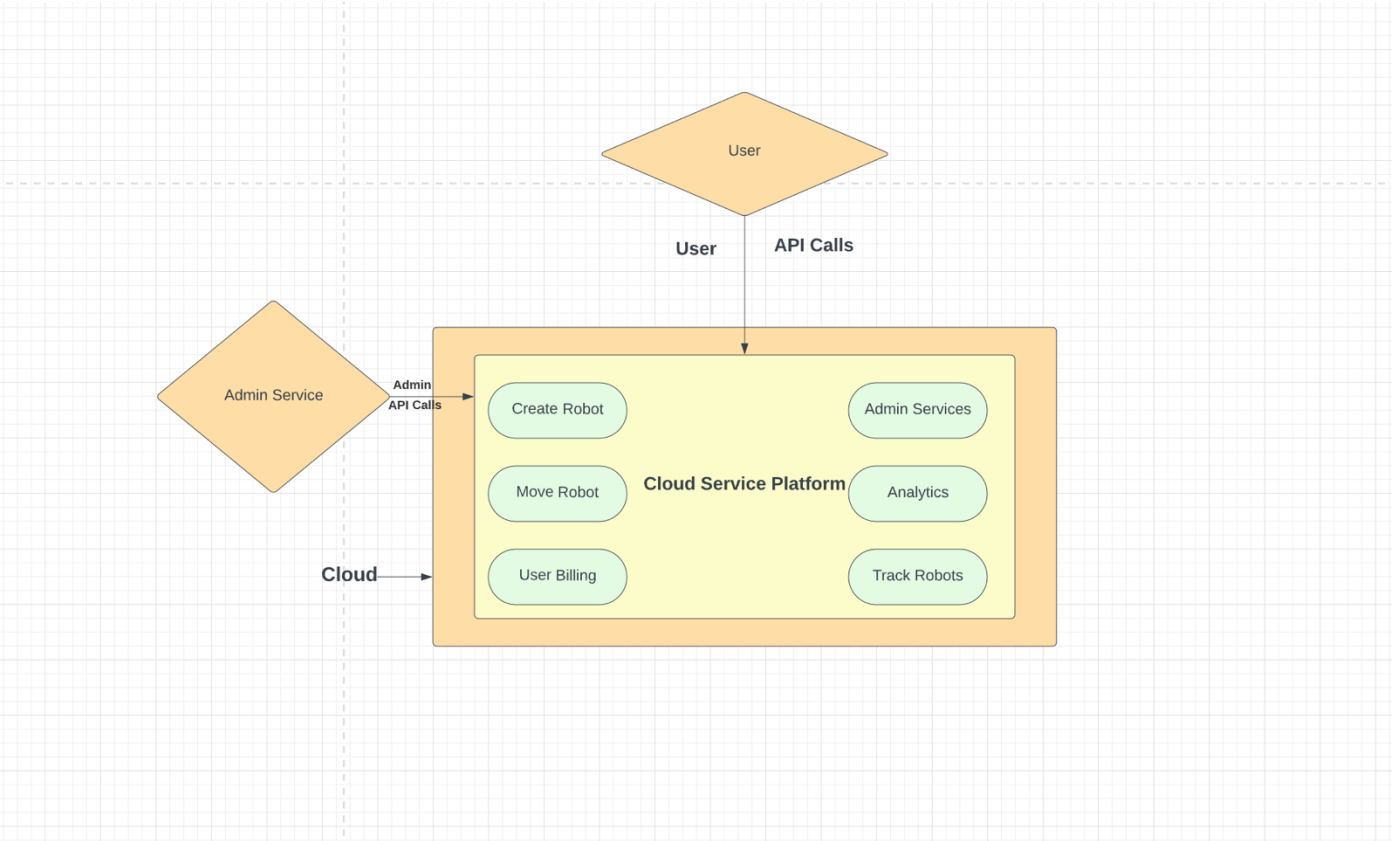
1.3 Function scope and usage:

The objective of this project's component is to provide seamless user-service administration for remotely configuring, monitoring, and using service robots. Utilize the computing power of the cloud to use these services effectively and efficiently via the cloud platform, and use the current services to communicate between the cloud and the distant robot.

Section 2: Component application interface design and analysis

2.1 Detailed component API interfaces and descriptions in terms of their parameters (RESTFUL APIs):

A higher-level user-service interface between a user and the cloud-service platform is shown in the figure below. To accomplish the component's goal, user and cloud platform communications will be conducted using the APIs listed below.



All the APIs that will be utilized in this component to manage user services on the Cloud service platform are listed below.

API's and their descriptions:

API method	URL	Parameters	Response Type
POST	login(userName, password)	userName : int Password : String	userId : int
POST	registerRobot(userName, password, userDetails)	userName : int Password : String userDetails : Object	response : String
GET	getBillingDetails(userId)	userId : int	billingInfo : Object
POST	login(adminName, adminPassword)	adminName : int adminPassword : String	adminId: int
GET	getUsers()	userId : int	response : int
GET	getAllRobots()	robotDetails : Object	response : int
GET	getActiveRobots()	activerobotDetails : Object	response : int
GET	noOfServiceOperationForRobot(robotId)	robotId : int	response : int
GET	getDistributionMetrics(robotId)	robotId : int	response : Object

Section 3: Component function design, data and DB design, and behavior analysis

3.1 Component logic design:

Class Diagram:

Generally, this component has 6 classes. One to maintain user-level information for login and registration, the other for administration. For robot operations, one class. The other is User Services, which offers every way a user could employ for tracking and monitoring robots. To examine all the analytics connected to the cloud service platform in their dashboard, admin users use the other one, Admi Services. All the classes, their corresponding variables, and the next operations are shown in the class diagram below.

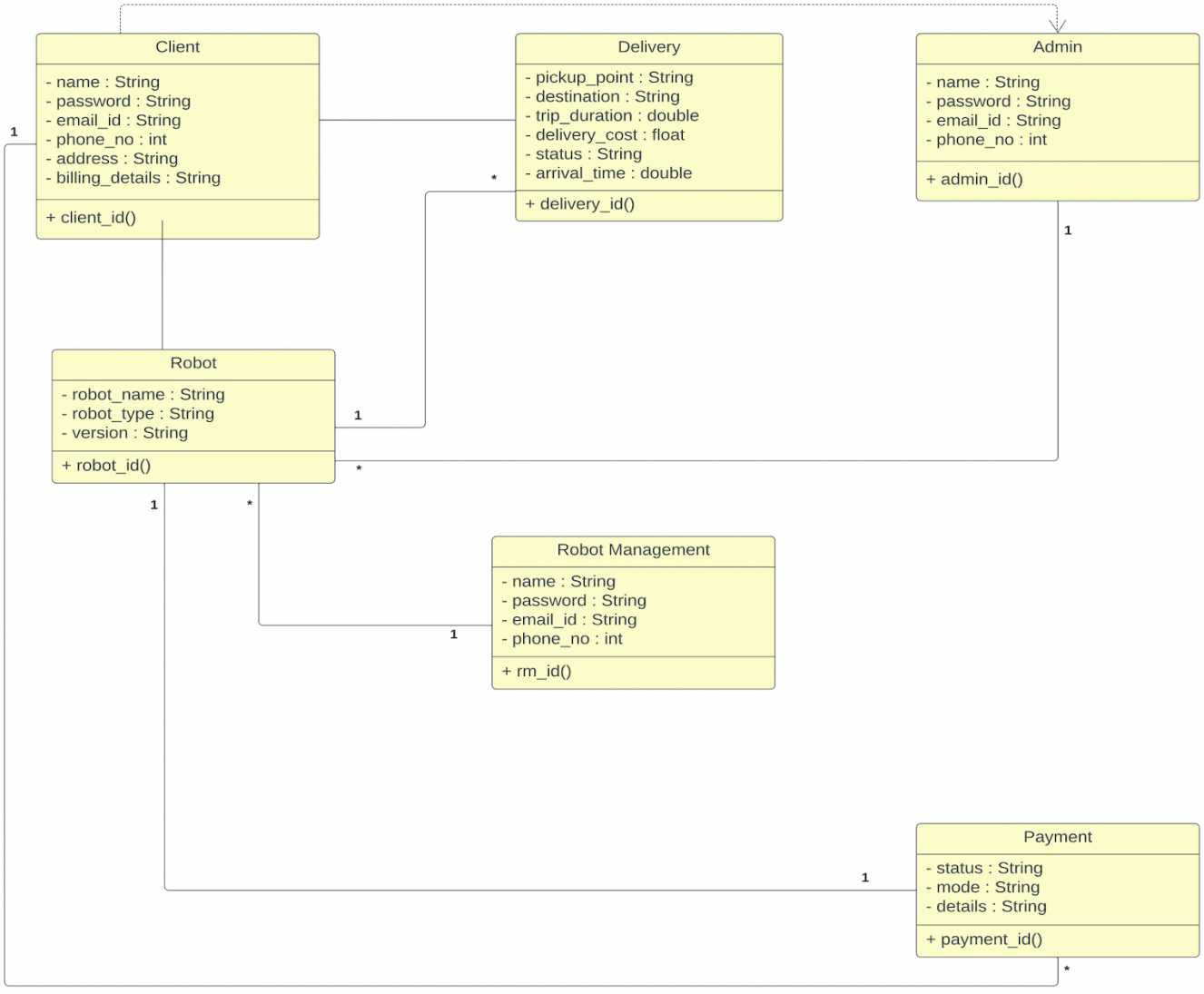


Fig: Component Class Diagram

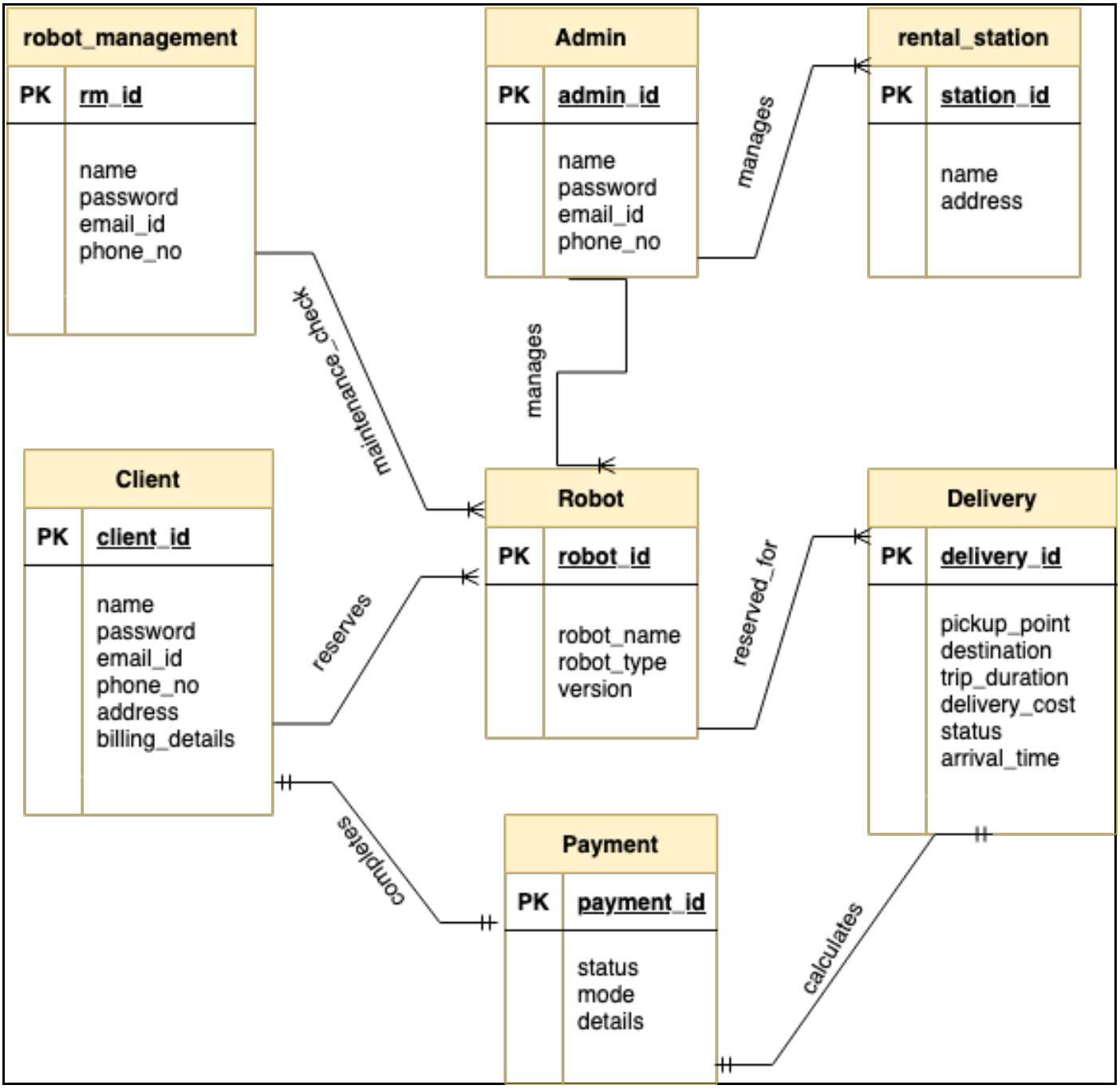


Fig: Relational DB Design – MySQL

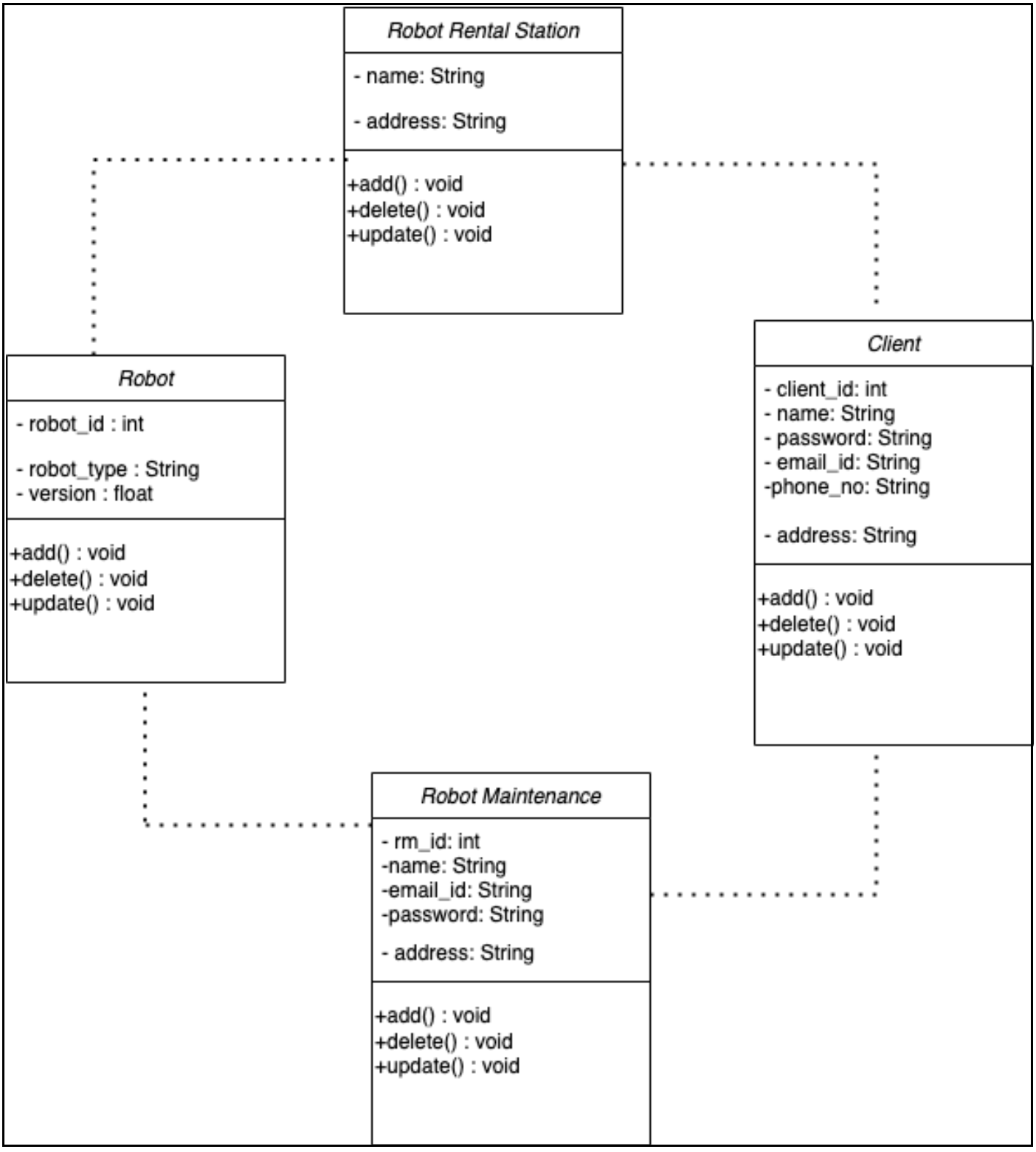


Fig: NoSQL DB Design

3.1 Program function descriptions (tables, flowcharts, program pseudo codes):

Tables:

The following details will be included in the user's schema, with user type being stored as an Integer. 1 represents the end user, and 2 represents the administrative staff.

Users

- user_id : int
- email_id: String
- password: String Encrypted
- first_name: String
- last_name:String
- user_type: int
- create_time: timeStamp
- update_time: timeStamp

The billing_details schema contains the following information with fk_user_id is a foreign key referencing to the user_id of user's schema

billing_details

- Billing_id : int
- fk_User_id : int
- Billing_amount: double
- Billing_time : timeStamp
- Create_time: timeStamp
- Update_time: timeStamp

Pseudo Code:

For Register Robot :

```
import React, {useContext} from 'react';
import { useState, useEffect } from 'react';
import axios from "axios";
import Avatar from '@mui/material/Avatar';
import Button from '@mui/material/Button';
import CssBaseline from '@mui/material/CssBaseline';
import TextField from '@mui/material/TextField';
import FormControlLabel from '@mui/material/FormControlLabel';
import Checkbox from '@mui/material/Checkbox';
import Link from '@mui/material/Link';
import Paper from '@mui/material/Paper';
import Box from '@mui/material/Box';
import Grid from '@mui/material/Grid';
import LockOutlinedIcon from '@mui/icons-material/LockOutlined';
import Typography from '@mui/material/Typography';
import { createTheme, ThemeProvider } from '@mui/material/styles';
import { Redirect, useHistory } from "react-router";
import {signup} from '../../services/authenticationService';
import { AuthContext } from '../authenticaion/ProvideAuth';


function Copyright(props) {
  return (
    <Typography variant="body2" color="text.secondary" align="center" {...props}>
      {'Copyright © '}
      <Link color="inherit" href="https://mui.com/">
        Your Website
      </Link>{' '}
      {new Date().getFullYear()}
      {'.'}
    </Typography>
  );
}

const theme = createTheme();

export default function Signup() {

  const history = useHistory();

  const [idCreated, setidCreated] = useState(0);
  const authContext = useContext(AuthContext);
  const [persona, setPersona] = useState('');

  const {setUser, setAuthState, updateLocalStorage} = authContext;

  const handleSubmit = async (event) => {
    event.preventDefault();
    const data = new FormData(event.currentTarget);
    var customer = data.get('customer');
    var carOwner = data.get('carOwner');
    var admin = data.get('admin');
    var persona;
    if (customer === 'on') persona = "customer";
    if (carOwner === 'on') persona = "owner";
    if (admin === 'on') persona = "admin";
    // eslint-disable-next-line no-console
```

```
var data1= {
      Robot Name: data.get('Robot Name'),
      Robot Type: data.get('Robot Type'),
      Manufacture Name: data.get('Manufacture Name'),
      Operating System: data.get('Operating System'),
      Version: data.get('Version),
      X location: data.get('X location),
      Y location: data.get('Y location'),
      persona:persona,
    };

    console.log(data1);
    const response = await signup(data1);
    if(response.status === 200){
      setUser(response.data);
      setAuthState(true);
      updateLocalStorage(response.data); //Need to call after setUser
      setTimeout(()=>{
        history.push('/login');
      }, 500);

    }
    else{
      setAuthState(false);
      console.log('Error', response);
    }
  };

  if(idCreated){
    return <Redirect to="/profile" />
  } else  if (idCreated==="Robot already exists"){
    return (
    <div> Robot already exists </div>
  );
  }
  return (

    <ThemeProvider theme={theme}>
      <Grid container component="main" sx={{ height: '100vh' }}>
        <CssBaseline />
        <Grid
          item
          xs={false}
          sm={4}
          md={7}
          sx={{
            backgroundImage: 'url(https://innovationatwork.ieee.org/wp-
content/uploads/2018/07/iStock-829197466-1024x683.jpg)',
            backgroundRepeat: 'no-repeat',
            backgroundColor: (t) =>
              t.palette.mode === 'light' ? t.palette.grey[50] : t.palette.grey[900],
            backgroundSize: 'cover',
            backgroundPosition: 'center',
          }}
        />
        <Grid item xs={12} sm={8} md={5} component={Paper} elevation={6} square>
          <Box
            sx={{
              my: 8,
              mx: 4,
              display: 'flex',
              flexDirection: 'column',
```

```
alignItems: 'center',
          }}
        >
          <Avatar sx={{ m: 1, bgcolor: 'secondary.main' }}>
            <LockOutlinedIcon />
          </Avatar>
          <Typography component="h1" variant="h5">
            Sign Up
          </Typography>
          <Box component="form" noValidate onSubmit={handleSubmit} sx={{ mt: 1 }}>
            <br></br>
            <Grid item xs={12}>
                  <div class="form-check form-check-inline">
                    <input
                      checked={persona === 'Customer'}
                      class="form-check-input"
                      type="radio"
                      name="customer"
                      id="customer"
                      onClick={()=>{setPersona('Customer')}}
                    />
                    <label
                      class="form-check-label" for="customer">
                      Customer
                    </label>
                  </div>
                  <div class="form-check form-check-inline">
                    <input
                      checked={persona === 'Owner'}
                      class="form-check-input"
                      type="radio"
                      name="RobotOwner"
                      id="RobotOwner"
                      onClick={()=>{setPersona('Owner')}}
                    />
                    <label class="form-check-label" for="carOwner">
                      Car Owner
                    </label>
                  </div>
                  <div class="form-check form-check-inline">
                    <input
                      checked={persona === 'Admin'}
                      class="form-check-input"
                      type="radio"
                      name="admin"
                      id="admin"
                      onClick={()=>{setPersona('Admin')}}
                    />
                    <label class="form-check-label" for="admin">
                      Admin
                    </label>
                  </div>
            </Grid>
            <br></br>
            <Grid container spacing={2}>
              <Grid item xs={12} sm={6}>
                <TextField
                  name="Robot Name"
                  required
                  fullWidth
                  id="firstName"
                  label="Robot Name"
```

```
autoFocus
                />
              </Grid>
              <Grid item xs={12} sm={6}>
                <TextField
                  required
                  fullWidth
                  id="Robot Type"
                  label="Robot Type"
                  name="Robot Type"
                  autoComplete="Robot-Type"
                />
              </Grid>
              <Grid item xs={12}>
                <TextField
                  required
                  fullWidth
                  id="Manufacture Name"
                  label="Manufacture Name"
                  name="Manufacture Name"
                  autoComplete="Manufacture Name"
                />
              </Grid>
              <Grid item xs={12}>
                <TextField
                  required
                  fullWidth
                  name="Operating System"
                  label="Operating System"
                  type="Operating System"
                  id="Operating System"
                  autoComplete="Operating System"
                />
              </Grid>
            </Grid>
              <Grid item xs={12}>
                <TextField
                  required
                  fullWidth
                  name="Version"
                  label="Version"
                  type="Version"
                  id="Version"
                  autoComplete="Version"
                />
              </Grid>
            </Grid>
              <Grid item xs={12}>
                <TextField
                  required
                  fullWidth
                  name="X location"
                  label="X location"
                  type="X location"
                  id="X location"
                  autoComplete="X location"
                />
              </Grid>
            </Grid>
              <Grid item xs={12}>
                <TextField
                  required
```

```
                fullWidth
                name="Y location"
                label="Y location"
                type="Y location"
                id="Y location"
                autoComplete="Y location"
              />
            </Grid>
          </Grid>
          <Button
            type="submit"
            fullWidth
            variant="contained"
            sx={{ mt: 3, mb: 2 }}
          >
            Sign Up
          </Button>
          <Grid container justifyContent="flex-end">
            <Grid item>
              <Link href="#" variant="body2">
                Already have an account? Sign in
              </Link>
            </Grid>
          </Grid>
            <Copyright sx={{ mt: 5 }} />
          </Box>
        </Box>
      </Grid>
    </Grid>
  </ThemeProvider>
 );
}
```

For Pricing :

```
import React, {useState, useContext} from 'react';
import AppBar from '@mui/material/AppBar';
import Box from '@mui/material/Box';
import Button from '@mui/material/Button';
import Card from '@mui/material/Card';
import CardActions from '@mui/material/CardActions';
import CardContent from '@mui/material/CardContent';
import CardHeader from '@mui/material/CardHeader';
import CssBaseline from '@mui/material/CssBaseline';
import Grid from '@mui/material/Grid';
import StarIcon from '@mui/icons-material/StarBorder';
import Toolbar from '@mui/material/Toolbar';
import Typography from '@mui/material/Typography';
import Link from '@mui/material/Link';
import GlobalStyles from '@mui/material/GlobalStyles';
import Container from '@mui/material/Container';
import {updateUserProfile} from '../../services/userService';
import { AuthContext } from '../authenticaion/ProvideAuth';
import { useHistory } from 'react-router';
import Snackbar from '@mui/material/Snackbar';

function Copyright(props) {
  return (
    <Typography variant="body2" color="text.secondary" align="center" {...props}>
      {'Copyright © '}
```

```
<Link color="inherit" href="https://mui.com/">
        Your Website
      </Link>{' '}
      {new Date().getFullYear()}
      {'.'}
    </Typography>
  );
}


const tiers = [
  {
    title: 'Free',
    price: '0',
    walletUpgrade: 0,
    description: [
      '100$ signup rewards',
      'Valid for 3 months',
      'Help center access',
      'Email support',
    ],
    buttonText: 'Sign up for free',
    buttonVariant: 'contained',
  },
  {
    title: 'Pro',
    price: '15',
    walletUpgrade: 15,
    description: [
      '100$ Signup rewards',
      '100$ Addtional wallet rewards',
      'Help center access',
      'Priority email support',
    ],
    buttonText: 'Get started',
    buttonVariant: 'outlined',
  },
  {
    title: 'Enterprise',
    price: '30',
    walletUpgrade: 30,
    description: [
      '100$ Signup rewards',
      '200$ Addtional wallet rewards',
      'Help center access',
      'Phone & email support',
    ],
    buttonText: 'Get Started',
    buttonVariant: 'outlined',
  },
];

const footers = [
  {
    title: 'Company',
    description: ['Team', 'History', 'Contact us', 'Locations'],
  },
  {
    title: 'Features',
    description: [
      'Cool stuff',
      'Random feature',
```

```
'Team feature',
      'Developer stuff',
      'Another one',
    ],
  },
  {
    title: 'Resources',
    description: ['Resource', 'Resource name', 'Another resource', 'Final resource'],
  },
  {
    title: 'Legal',
    description: ['Privacy policy', 'Terms of use'],
  },
];

function PricingContent() {

  const authContext = useContext(AuthContext);
  const [open, setOpen] = useState(false);
  const history = useHistory();
  const {user, setUser, token, updateLocalStorage} = authContext;

  const handleClose = () => {
    setOpen(false);
  }

  const walletUpgradeHandler = async (walletUpgrade) => {

    console.log("walletUpgrade : ", walletUpgrade);
    const obj = {
      ...user,
      walletBalance: user.walletBalance - walletUpgrade,
    }
    const response = await updateUserProfile(obj);
    console.log(response);
    if(response.status === 200){
      // window.alert("Wallet has been upgraded to $ " + walletUpgrade);
      setOpen(true);
      setUser(response.data.payload.data);
      updateLocalStorage(user, token);
      setTimeout(()=>{
        console.log("history.push('/Dashboard');");
        history.push('/Dashboard');
      }, 500);
    }
    else{
      console.log('Error Occuered');
    }
  }



  return (
    <React.Fragment>
      <Snackbar
        open={open}
        autoHideDuration={6000}
        onClose={handleClose}
        message="Payment Plan Updated"
      />
      <GlobalStyles styles={{ ul: { margin: 0, padding: 0, listStyle: 'none' } }} />
      <CssBaseline />
```

```
  color="default"
  elevation={0}
  sx={{ borderBottom: (theme) => `1px solid ${theme.palette.divider}` }}
>
</AppBar>
{/* Hero unit */}
<Container disableGutters maxWidth="sm" component="main" sx={{ pt: 8, pb: 6 }}>
  <Typography
    component="h1"
    variant="h2"
    align="center"
    color="text.primary"
    gutterBottom
  >
    Payment Plan
  </Typography>
  <Typography variant="h5" align="center" color="text.secondary" component="p">
    Choose a payment plan to initiate your wallet.
  </Typography>
</Container>
{/* End hero unit */}
<Container maxWidth="md" component="main">
  <Grid container spacing={5} alignItems="flex-end">
    {tiers.map((tier) => (
      // Enterprise card is full width at sm breakpoint
      <Grid
        item
        key={tier.title}
        xs={12}
        sm={tier.title === 'Enterprise' ? 12 : 6}
        md={4}
      >
        <Card>
          <CardHeader
            title={tier.title}
            // subheader={tier.subheader}
            titleTypographyProps={{ align: 'center' }}
            action={tier.title === 'Pro' ? <StarIcon /> : null}
            subheaderTypographyProps={{
              align: 'center',
            }}
            sx={{
              backgroundColor: (theme) =>
                theme.palette.mode === 'light'
                  ? theme.palette.grey[200]
                  : theme.palette.grey[700],
            }}
          />
          <CardContent>
            <Box
              sx={{
                display: 'flex',
                justifyContent: 'center',
                alignItems: 'baseline',
                mb: 2,
              }}
            >
              <Typography component="h2" variant="h3" color="text.primary">
                ${tier.price}
              </Typography>
              <Typography variant="h6" color="text.secondary">
                /mo
```

```
</Typography>
                    </Box>
                    <ul>
                      {tier.description.map((line) => (
                        <Typography
                          component="li"
                          variant="subtitle1"
                          align="center"
                          key={line}
                        >
                          {line}
                        </Typography>
                      ))}
                    </ul>
                  </CardContent>
                  <CardActions>
                    <Button fullWidth variant={tier.buttonVariant}
onClick={()=>walletUpgradeHandler(tier.walletUpgrade)}>
                      {tier.buttonText}
                    </Button>
                  </CardActions>
                </Card>
              </Grid>
            ))}
          </Grid>
        </Container>
        {/* Footer */}
        <Container
          maxWidth="md"
          component="footer"
          sx={{
            borderTop: (theme) => `1px solid ${theme.palette.divider}`,
            mt: 8,
            py: [3, 6],
          }}
        >
          <Grid container spacing={4} justifyContent="space-evenly">
            {footers.map((footer) => (
              <Grid item xs={6} sm={3} key={footer.title}>
                <Typography variant="h6" color="text.primary" gutterBottom>
                  {footer.title}
                </Typography>
                <ul>
                  {footer.description.map((item) => (
                    <li key={item}>
                      <Link href="#" variant="subtitle1" color="text.secondary">
                        {item}
                      </Link>
                    </li>
                  ))}
                </ul>
              </Grid>
            ))}
          </Grid>
          <Copyright sx={{ mt: 5 }} />
        </Container>
        {/* End footer */}
      </React.Fragment>
    );
}

export default function Pricing() {
```

```
return <PricingContent />;
}
```

For Wallet :

```
import  React, {useContext} from 'react';
import PropTypes from 'prop-types';
import Typography from '@mui/material/Typography';
import Grid from '@mui/material/Grid';
import Card from '@mui/material/Card';
import CardActionArea from '@mui/material/CardActionArea';
import CardContent from '@mui/material/CardContent';
import CardMedia from '@mui/material/CardMedia';
import {AuthContext} from '../authenticaion/ProvideAuth';

function Wallet(props) {


  const authContext = useContext(AuthContext);
  const { user, loading } = authContext;
    var post = {
        image: "https://payspacemagazine.com/wp-content/uploads/2018/10/dollar1-
1.jpg",
        imageLabel:"img"
    }
  return (
    <>
    {!loading && (
    <Grid item xs={12} md={15}>
      <CardActionArea component="a" href="#">
        <Card sx={{ display: 'flex' }}>
          <CardContent sx={{ flex: 1 }}>
          <Typography variant="subtitle1" paragraph>
              {/* {post.description} */}
              Wallet
            </Typography>
            <Typography variant="subtitle1" color="text.secondary">
              {/* {post.date} */}
              Current balance
            </Typography>
            <Typography component="h2" variant="h2">
              {user.walletBalance > 0 ? user.walletBalance : 0} $
            </Typography>

          </CardContent>
          <CardMedia
            component="img"
            sx={{ width: 160, display: { xs: 'none', sm: 'block' } }}
            image={post.image}
            alt={post.imageLabel}
          />
        </Card>
      </CardActionArea>
    </Grid>
    )}
    </>
  );
}

Wallet.propTypes = {
  post: PropTypes.shape({
    date: PropTypes.string.isRequired,
```

```
post: PropTypes.shape({
    date: PropTypes.string.isRequired,
    description: PropTypes.string.isRequired,
    image: PropTypes.string.isRequired,
    imageLabel: PropTypes.string.isRequired,
    title: PropTypes.string.isRequired,
  }).isRequired,
};

export default Wallet;
```

Flow Chart:

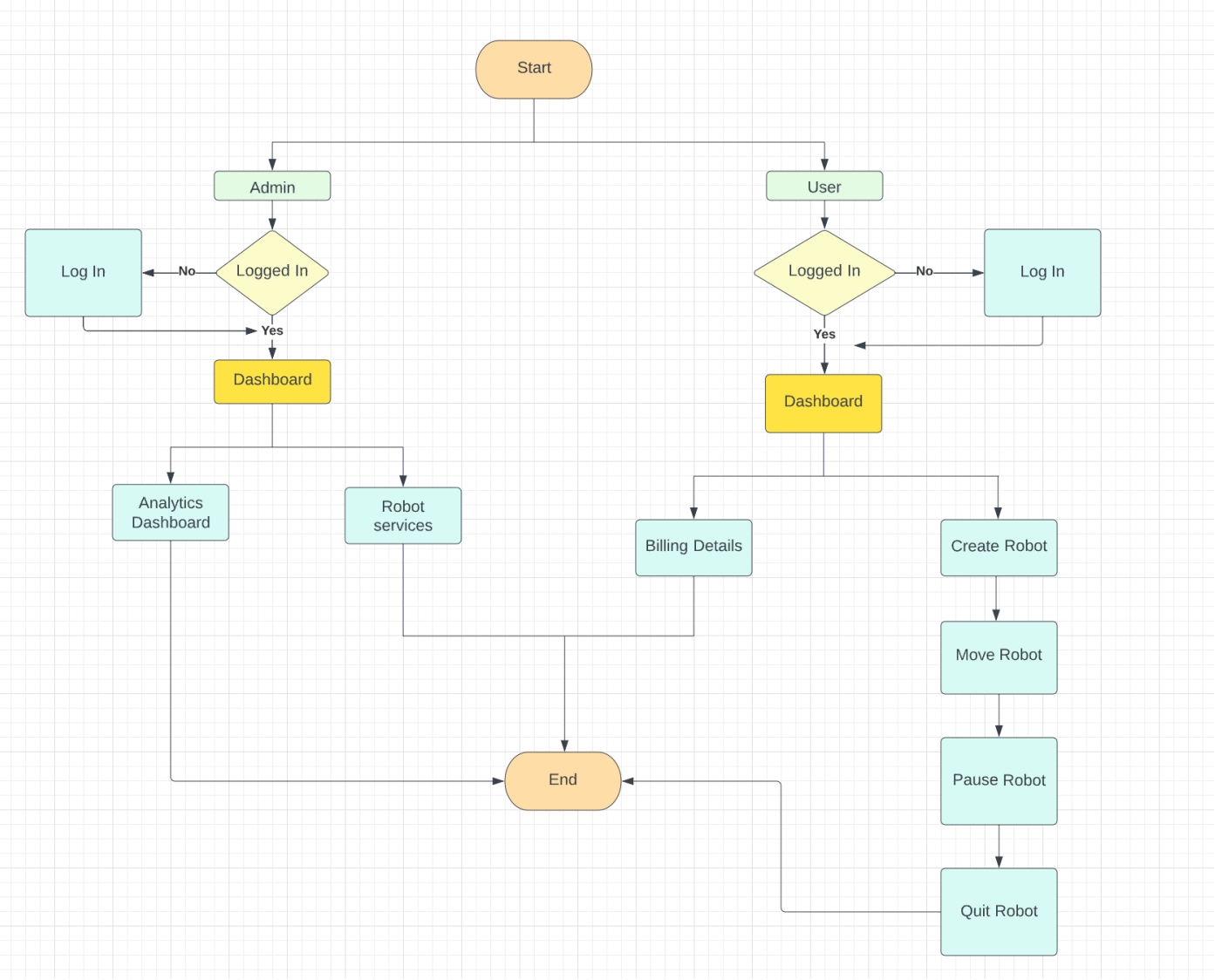


Fig: Component Class Flow Diagram

3.2 Program behaviors:

Component Class sequence diagram:

The stages between all the accessible personals in this component are shown in the below figure. This would provide a clear code from which the project would be developed.

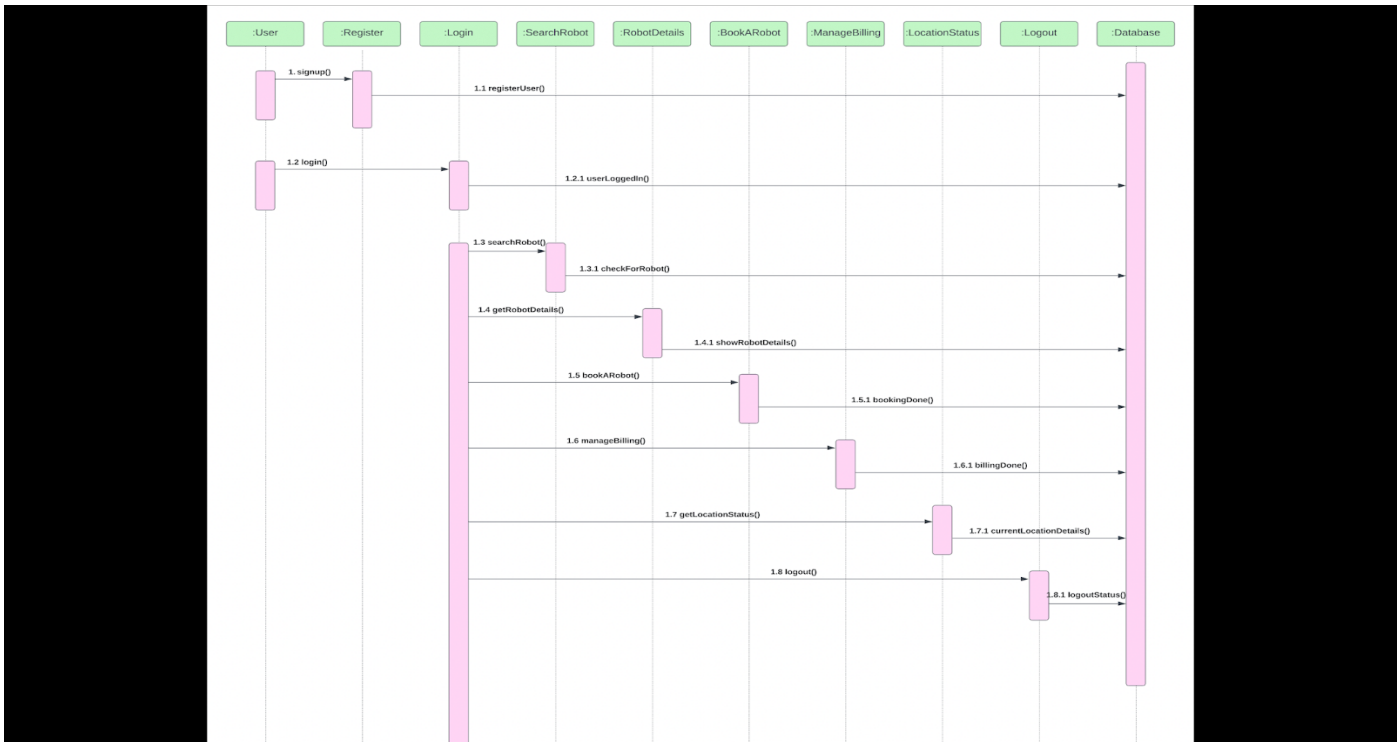


Fig: User Side Sequence Diagram

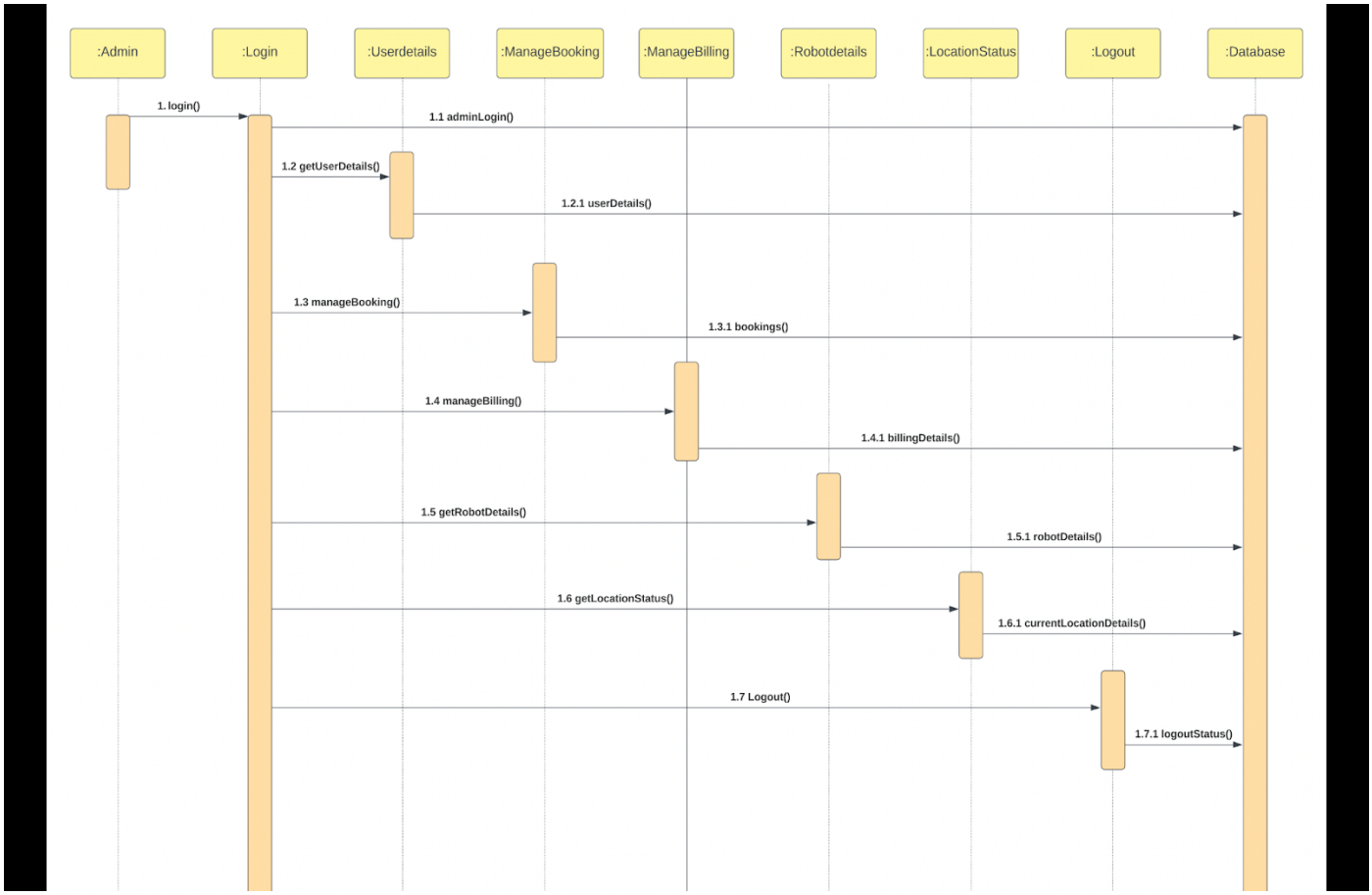


Fig: Admin Side Sequence Diagram

Component Class State Diagram:

User:

The user can be viewed in all the states shown in the state diagram below, along with the component's flow from one state to the next.

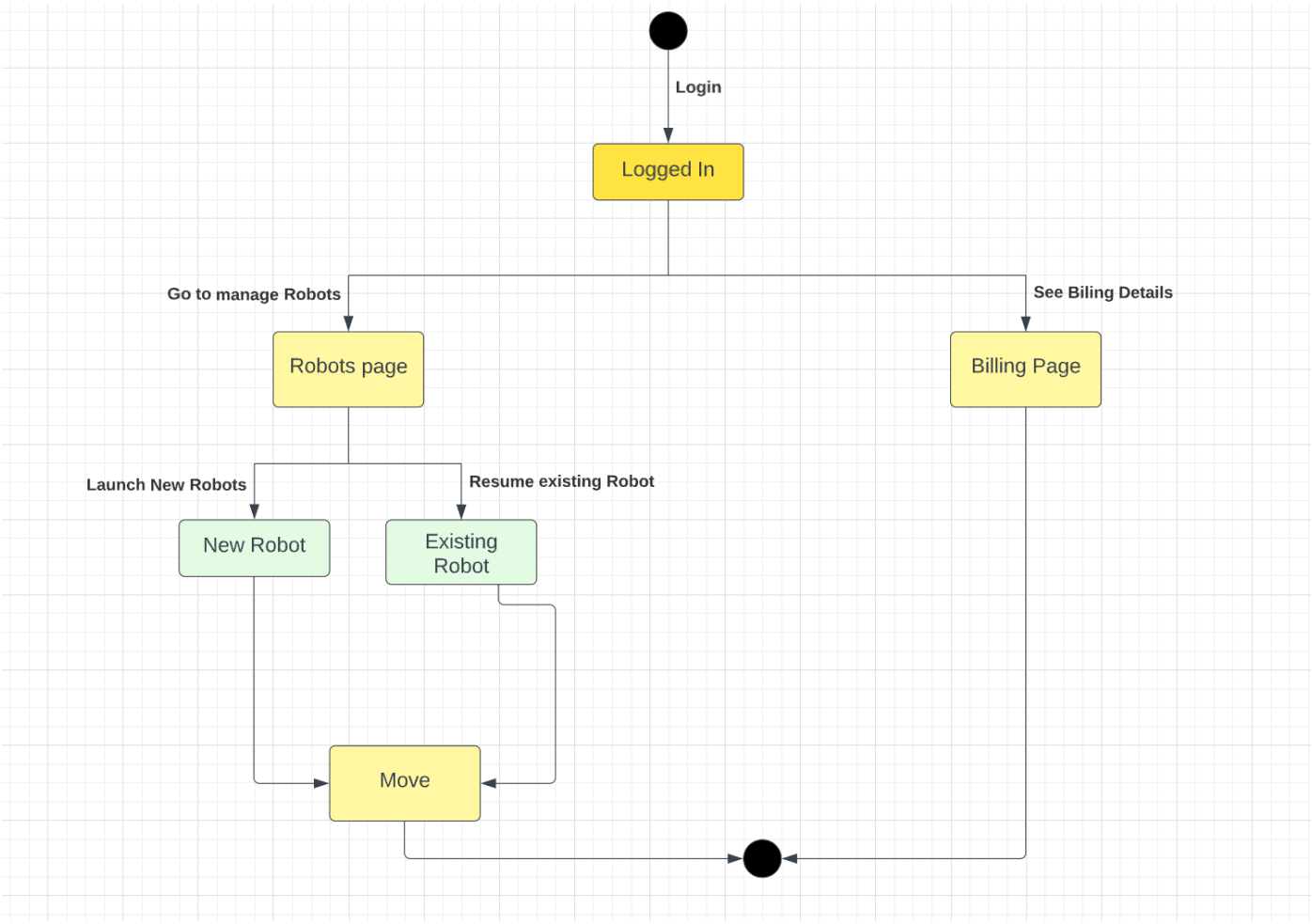


Fig: User related State diagram

Admin:

The component's transition from one state to the next is shown in the state diagram below, along with all the stages in which an admin can be observed.

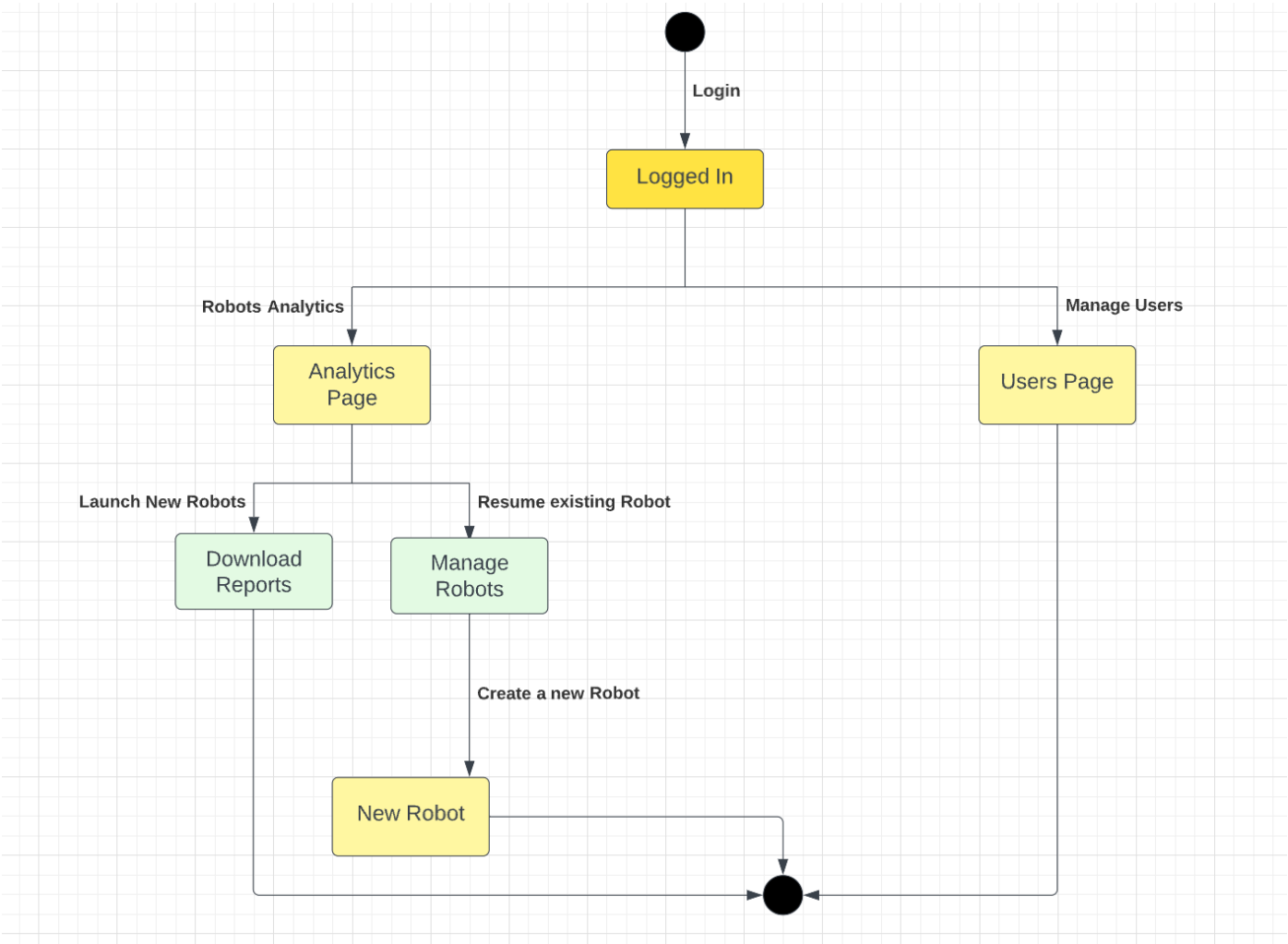


Fig: Admin related State diagram

Section 4: Component business logic

4.1 Business logics (decision tables):

The main goal of this component is to create a user-service management system that empowers users to utilize the services deployed by administrators. It also provides administrators with a dashboard from which they can manage and view all analytics related to cloud service management, including the number of users registered, registered robots, active robots, and the distribution of states for all the existing robots.

There are services that make it possible for new users to sign up for and log in to the Cloud service management platform. Every user has a distinct userId defined on the user table in the database, and every robot created by that specific user is tagged with that userId. We can ultimately obtain all the registered robots for a specific user.

Once a user has successfully logged in, the dashboard offers several features, including the ability to build, move, track, pause, and delete robots. The session times of each robot associated with each user are stored in the existing MySQL and mongo DB databases, which are used to keep track of this. The tables also offer the option of storing each robot's current state.

Admin registration is only possible from the backend for security reasons. A platform administrator can log in and view all the analytics being run on the cloud service platform once the process is complete. To track the number of users and robots in the system, there are numerous APIs available. The admin can also choose to view the billing data for every robot and statistics.

As a result, the business logic of this component mostly consists of giving the admin access to the cloud platform's analytics and users' APIs so they can log in and see data about billing and automated tracking. The other parts handle tracking, keeping track of the robot's status, database tables, edge-based mobile robot simulation, interactions, and supplying a user interface.

Section 5: Component graphic user interface design

5.1 Component GUI style, storyboard:

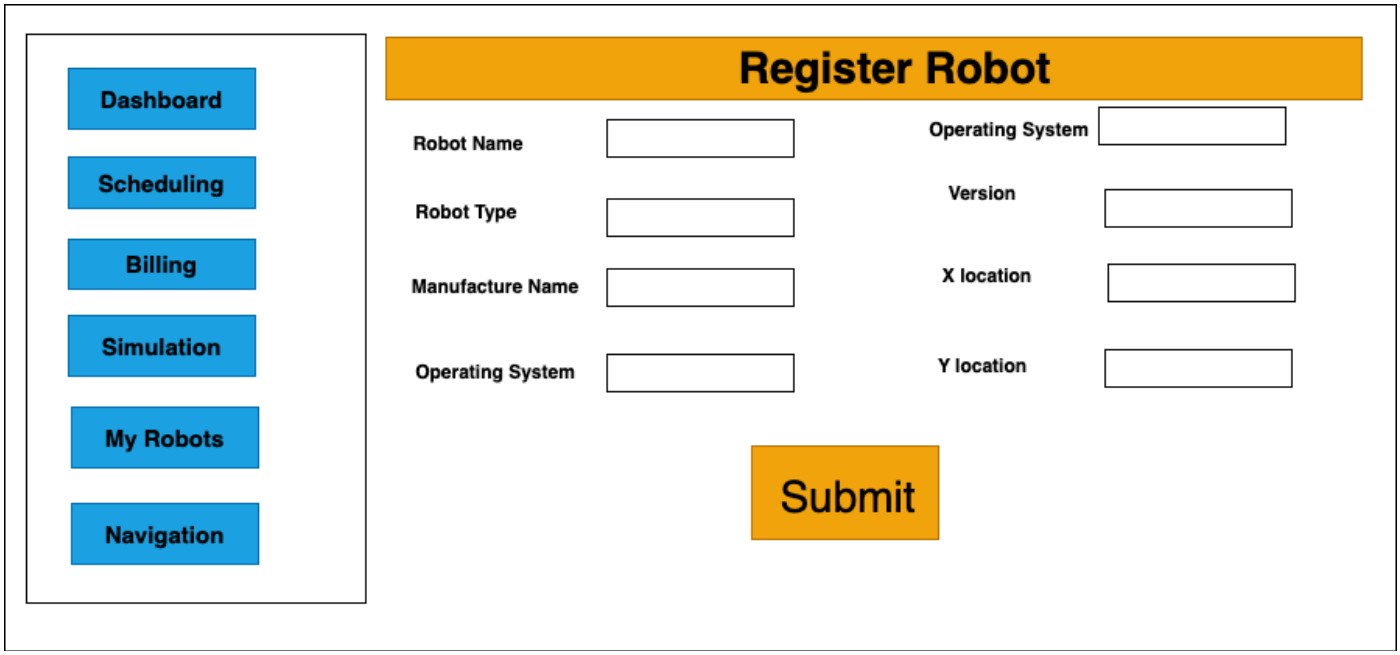
The GUI style is a straightforward web page, as seen in section 5.2 below, with a navigation bar to take users from anywhere in the application to the home page. There will always be a side bar that displays all the options that are available to a certain user, such as billing and logging out.

The stories for this component will revolve around developing user and admin services as well as effective user and admin communication using the cloud service platform.

5.2 UI markup diagram, and GUI templates:

Here are a few of the user interface (UI) wireframes for user accounts and billing details. Related to the analytics on the admin dashboard:

User/Admin Login(Register):



User Billing Information: