CONTAINMENT ZONE ALERTING APPLICATION

(TEAM ID : PNT2022TMID34803)

PROJECT REPORT

Submitted by

ABIKESH S. (962819104002)

ADHIL AHAMED H. (962819104006)

DHINAKAR S.(962819104030)

GOGILA NAVEEN K. (962819104041)

inpartial fulfillment for the award of

degreeof

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

UNIVERSITY COLLEGEOF ENGINEERING, NAGERCOIL

ANNA UNIVERSITY: CHENNAI 600 025

NOVEMBER 2022

INDEX

CHAPTER 1
INTRODUCTION

Due to the COVID-19 outbreak , many people are affected by the corona virus. About 66.1 Lakh people have lost their lives. It can easily spread through close contact. Everytime when the world controls the virus, another variant of the virus is formed and it continues the problem. Inorder to avoid corona transmission, people are advised to wear masks and to avoid close contact. People are even advised to stay in their homes during high transmission rate of the virus. But people can't stay in their homes for a long time, they need to come outside for their daily needs. Even after wearing the masks, people get infected by the virus. It is because they don't know whether the person they contact is infected or not. People also don't know about the areas with high infection. People should get to know about the areas with high infection so that they can avoid going there. People who gain knowledge about the containment zones will avoid entering the containment zones.

1.1 PROJECT OVERVIEW

In this project , we create a containment zone alerting application. It consists of an admin app (portal) and a user app (mobile app). The admin login to the app and update the containment zone locations in the database. The user will have a user registration and login the mobile app. The user will enter the location where he or she is planning to go. The admin receives the request location and checks whether if it matches with any of the location in the database. If the location matches with the database an alert notification is sent to the user. On receiving the alert message the user will avoid going to the containment zone. Hence by using this application the user can avoid the corona transmission by avoiding the containment zone areas.

1.2 PURPOSE

The purpose of this project is that the user should avoid contact with the infected person and to provide information about the containment zones.

CHAPTER 2

LITERATURE SURVEY

1. The World Health Organization has declared the outbreak of the novel coronavirus, Covid-19 as pandemic across the world. With its alarming surge of affected cases throughout the world, lockdown, and awareness (social distancing, use of masks etc.) among people are found to be the only means for restricting the community transmission. In a densely populated country like India, it is very difficult to prevent the community transmission even during lockdown without social awareness and precautionary measures taken by the people. Recently, several containment zones had been identified throughout the country and divided into red, orange and green zones, respectively. The red zones indicate the infection hotspots, orange zones denote some infection and green zones indicate an area with no infection. This paper mainly focuses on development of an Android application which can inform people of the Covid-19 containment zones and prevent trespassing into these zones. This Android application updates the locations of the areas in a Google map which are identified to be the containment zones. The application also notifies the users if they have entered a containment zone and uploads the user's IMEI number to the online database. To achieve all these functionalities, many tools, and APIs from Google like Firebase and Geofencing API are used in this application. Therefore, this application can be used as a tool for creating further social awareness about the arising need of precautionary measures to be taken by the people of India.

2. In this paper presents a study on GPS Based Location Monitoring System with Geo-fencing Capabilities. This system provided a highsecurity system that prevents vehicles from being stolen. It also issued an alert to the user based on the boundary of the location by using the Internet of Things (IoT). In this study, the system could easily monitor and track the location of the vehicle and was able to issue an alert when the vehicle exited the geofence area. This system was separated into two parts which were the hardware and software. The hardware parts were the ESP8266 Node MCU and GPS module while Google Maps and IoT platform were the software parts.

The admin could monitor the vehicle via the computer, and the notification alert was sent to the registered email of the admin when the vehicle exited or entered the geofence area. The prototype system was tested by moving the vehicle around the geofence area. The results showed the correct location of the vehicle and email notification alert when it exited or entered the boundaries. The location accuracy of about 95% compared to the real-map on the mobile phone.

3. This paper proposes a disaster information system using the geofencing technology to detect the movement of users and provide information of the risk for them. The system is composed of client-server architecture; the server collects risk information from various information sources and the client watches the user to notify the information as the need arises. To detect the user's movement, the client creates a virtual fence called geofence at the dangerous area based on the risk information stored in the server, and monitors the user's entry and exit of the fence. Thus the system can deliver warnings and advices timely to specific users in danger. We implemented a prototype system and evaluated the accuracy of the system. The location of the user was detected with high accuracy when entering the fence, but the accuracy was low when exiting the fence.the movement of users and provide information of the risk for them. The system is composed of client-verser architecture; the server collects risk information from various information sources and the client watches the user to notify the information as the need arises. To detect the user's movement, the client creates a virtual fence called geofence at the dangerous area based on the risk information stored in the server, and monitors the user's entry and exit of the fence. Thus the system can deliver warnings and advices timely to specific users in danger. We implemented a prototype system and evaluated the accuracy of the system. The location of the user was detected with high accuracy when entering the fence, but the accuracy was low when exiting the fence.

4. Geo-fencing has been predicted to be a multi-billion dollar market in areas such as retail, ambient intelligence, entertainment, healthcare, etc. Businesses have been adopting geo-fencing technology, and now there are several platform providers such as Google, Qualcomm, Esri, Urban Airship, and others. These tools are continuing to

attract application developers; however, best practices for choosing the specific performance options within this technology is still ambiguous. For example, Esri provides a geo-trigger service that allows developers to send targeted messages to users when they enter, exit, or dwell in a geo-fenced area. This service also provides the ability to choose higher levels of accuracy or battery saving by offering different location tracking profiles. This paper investigated two geo-trigger tracking profiles (Fine and Adaptive) to assess their performance in small, outdoor, geo-fenced areas; these two profiles are the most accurate but vary in their batteryuse. The results show the Adaptive tracking profile to provide 100% reliability and average accuracy of 68.53 meters in geo-fences between 20-70 meter radii. In addition, the Adaptive tracking profile saved 15.20% battery-life while the user is stationery and 9.23% while the user is moving.

5. The success of disaster handling often depends on the efficient flow of information. The social media and networks receive a growing attention as potential source of valuable data in disaster scenarios. The social media and networks receive a growing attention as potential source of valuable data in disaster scenarios. The social network based information flow is real-time, direct, two directional and often geo-tagged. Unfortunately, besides these obvious advantages, social network data suffers from drawbacks: it is unstructured, dispersed and lacks reliability. This paper proposes an approach based on combining a geo-fencing technology with social network platform to combat this problem and deliver a novel service for disaster management. The service groups users ad-hoc based on their location. Social network features allow users to exchange real-time information, coordinate rescue efforts, issue and report tasks. The geofences are visualized to provide a good overview of the disaster zone. The service was evaluated by disaster management experts, with an encouraging feedback.

Existing problem

When the person needs to travel, he doesn't know whether the zone is affected by covid or not. If he enter the containment zone he may get affected by the virus.

2.1 References

1. Ranajoy Mallik, Amlan Protim Hazarika, Dilip Singh and Bandyopadhyay. “Development of An Android Application for Viewing Covid-19 Containment Zones and Monitoring Violators Who are Trespassing into It Using Firebase and Geofencing”,2020.

2. A.H.Abbas, Mohammed I. Habelalmateen, Syukran Jurdi, L. Audah, N. Alduais. “GPS based location monitoring system with geo-fencing capabilities”,2019.

3. Akira Suyama, Ushio Inoue. “Using geofencing for a disaster information system”, 2016.

4. Mohammed Alsaqer, Brian Hilton, Tom Horan and Omar Aboulola. “Performance Assessment of Geo-triggering in Small Geo-fences: Accuracy, Reliability, and Battery Drain in Different Tracking Profiles and Trigger Directions”,2015.

5. Piotr Szczytowski. “Geo-fencing Based Disaster Management Service”, 2014.

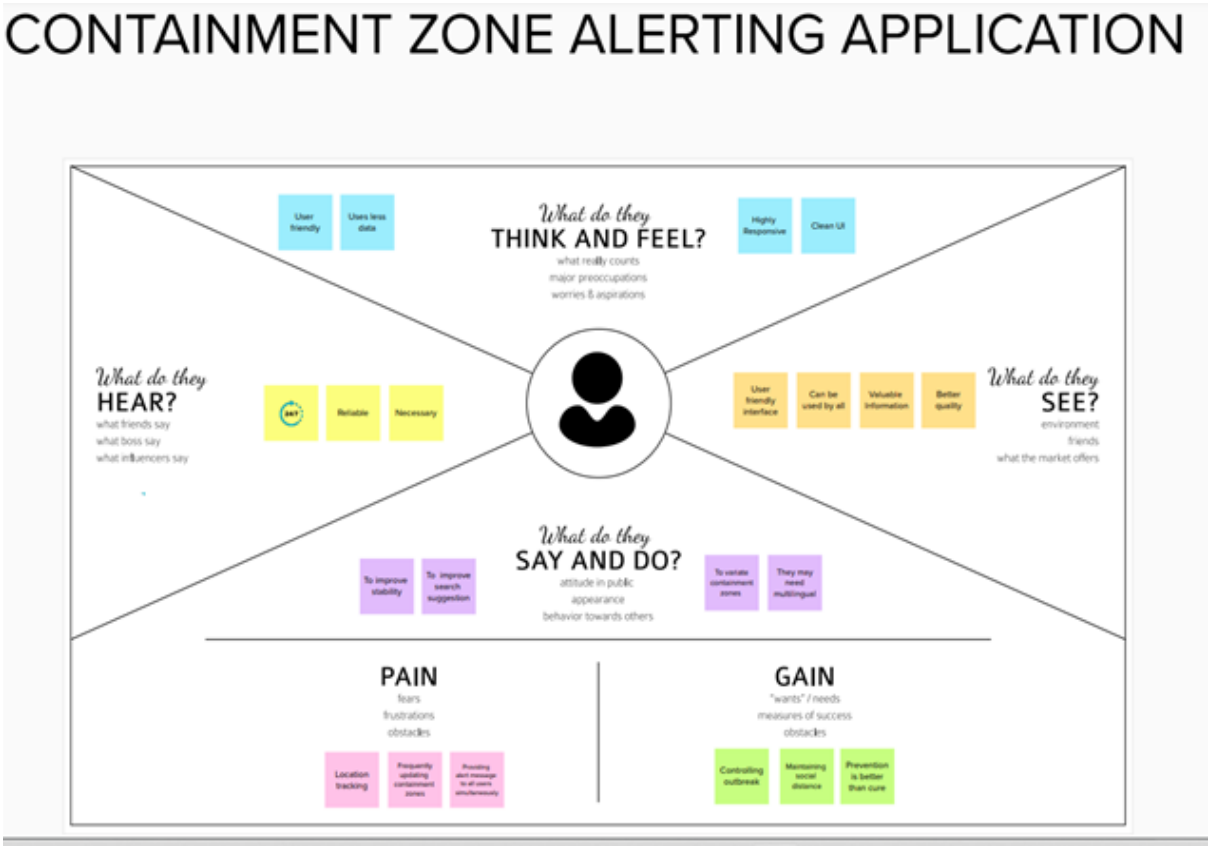
2.2 Problem Statement Definition

Problem Statement (PS)	I am (Customer)	I’m trying to	But	Because	Which makesme feel
PS-1	A traveller	Avoid the continment zones.	I accidentally stepped on the containme nt zones.	I don’t have any medium to get the alert notification.	I’m vulnerable
PS-2	A traveller	Get the updation of containme nt zoAnes.	It leads some knowledge about no of patient	Knowledge about society.	Updated .

CHAPTER 3

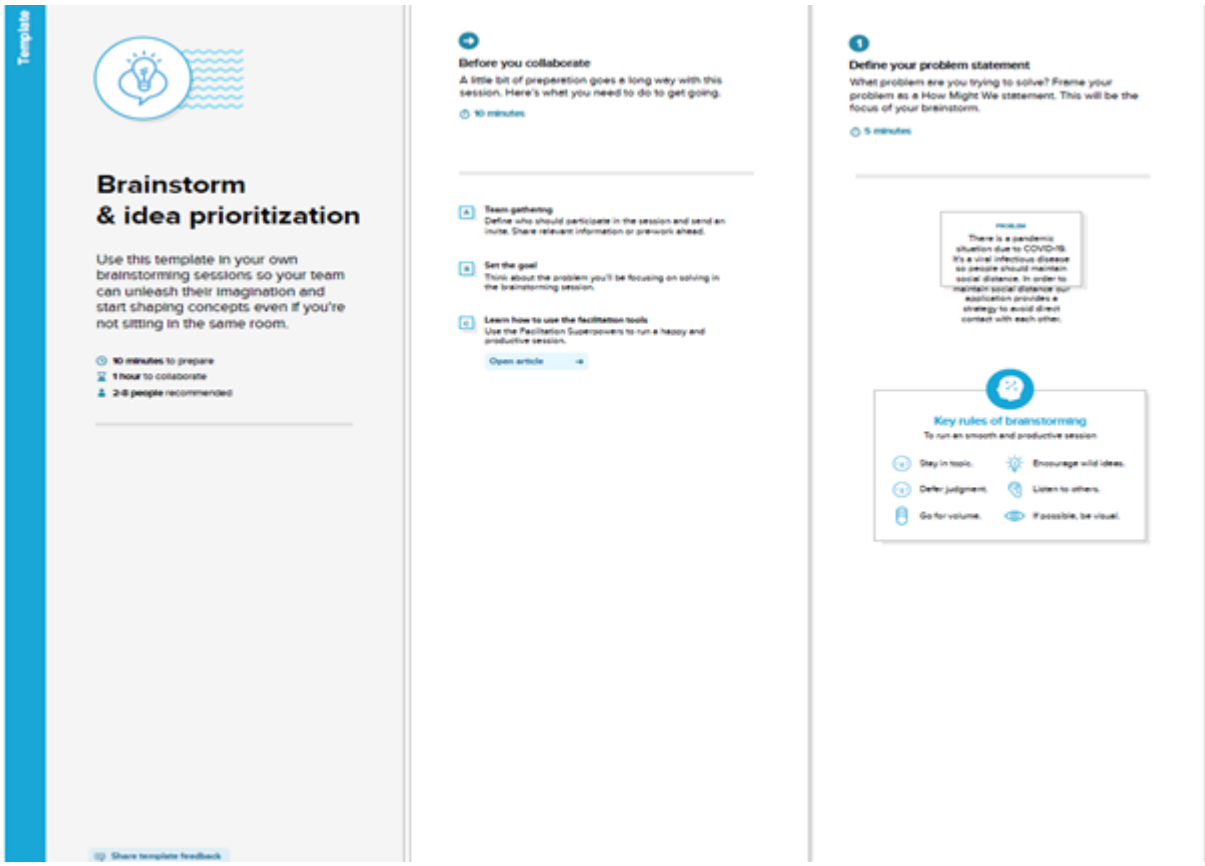
IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



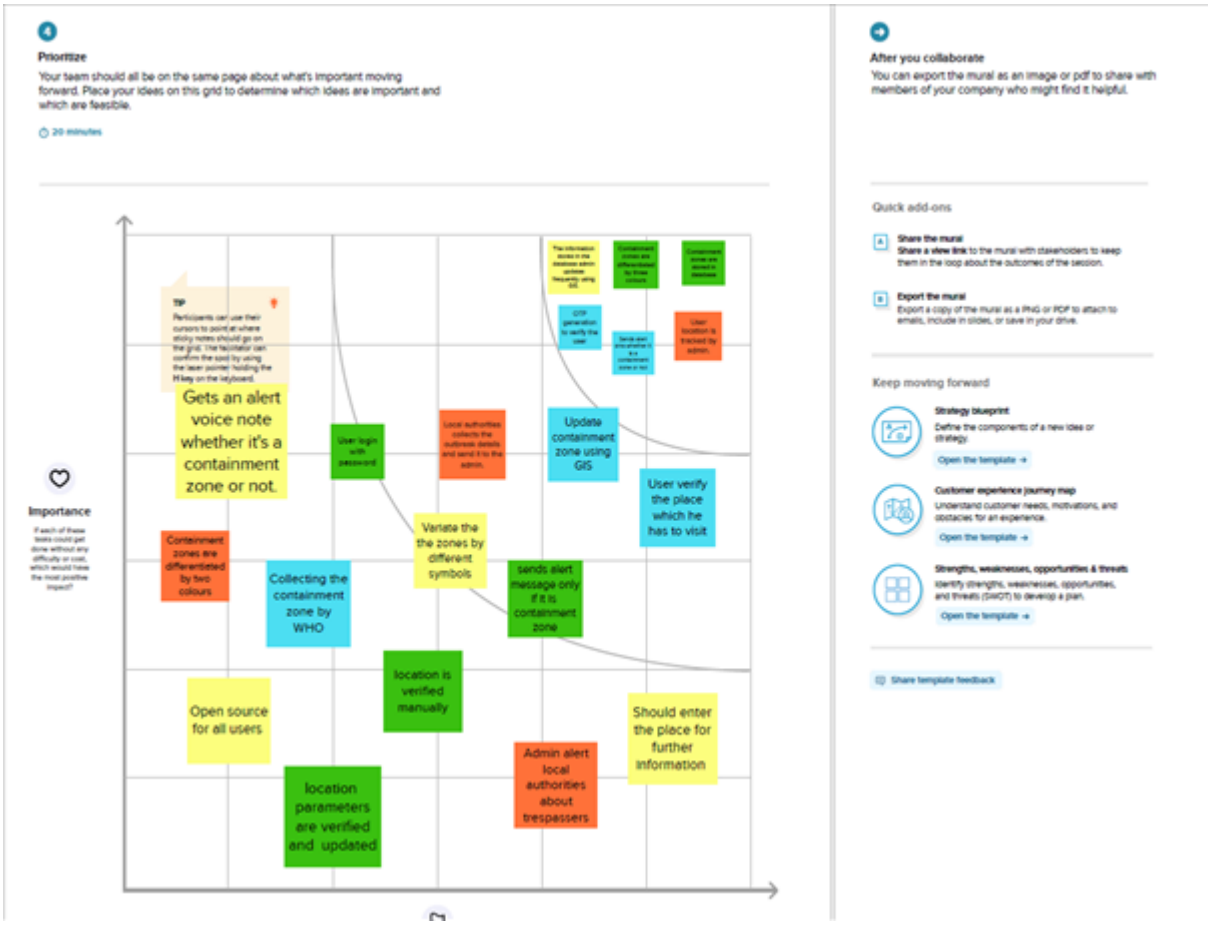
3.2 IDEATION AND BRAINSTORMING

Step-1: Team Gathering, Collaboration and Select the Problem Statement



Step-2: Brainstorm, Idea Listing and Grouping

Step-3: Idea Prioritization



3.3 Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to besolved)	There is a pandemic situation due to COVID-19. It's a viral infectious disease so people should maintain social distance. Inorder to maintain social distance our application provides strategy to avoid direct contact with each other.
2.	Idea / Solution description	This application is intended to provide information about containment zones in a particular region by alerting people through continuous monitoring of an individual's location.
3.	Novelty / Uniqueness	This application shows the locations of the containment zones to the users. It also notifies the user when he or she trespasses the boundary of a containment zones or stays in the containment zones.
4.	Social Impact/ CustomerSatisfaction	With the alarming increase of COVID-19 affected cases throughout the world, this application can be employed as a tool for creating further social awareness among the people.
5.	Business Model (RevenueModel)	A free application which highlights the need for taking for further precautionary measures for combating COVID-19.
6.	Scalability of the Solution	Multiple users can access the application in a mean time. By that, they can aware of entering the isolated area.

3.1 Problem Solution fit

Project Title: Containment zone alerting Application. **Project Design Phase-I - Solution Fit Template** **Team ID:** PNT2022TMID34803

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

Public who needs to travel / Police who needs the information about containment zones.

6. CUSTOMER CONSTRAINTS CC

Internet access / language difficulties.

5. AVAILABLE SOLUTIONS AS

Public doesn't have the knowledge about containment zones, In this application a map which shows the clear information about the containment zones.

Explore AS, differentiate

Focus on J&P, tap into BE, understand RC

2. JOBS-TO-BE-DONE/PROBLEMS J&P

Containment zones identification for further updation.

9. PROBLEM ROOT CAUSE RC

Increasing or spreading of disease makes people to stay at home inorder to travel you should need the information about the non-containment zones.

7. BEHAVIOUR BE

Directly related : Easy to use, accurate location about containment zones.

Indirectly associated : Require high internetspeed inorder to generate otp for user login.

Focus on J&P, tap into BE, understand RC

Identify strong TR & EM

3. TRIGGERS TR

If the operation is done and successfully completed by using this project by a person, that triggers government to make it official.

4. EMOTIONS: BEFORE / AFTER EM

Before: Contagious disease.

After: Faster response, avoid contagious disease.

10. YOUR SOLUTION SL

1) Create a login webpage using Id and password for both the user and admin.
2) Admin who updates the containment zones using geo fencing algorithm.
3) If user has the access he/she can have knowledge about the containment zones.
4) Alert messages will be provided.

8. CHANNELS of BEHAVIOUR CH

Online: To upload or update the containment zones frequently. To get the user Id to access the application.

Offline: Stores the locations of containment zones.

Extract online & offline CH of BE

CHAPTER 4

REQUIREMENT ANALYSIS

4.1 Functional Requirements

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration throughEmail ID or phone number.
FR-2	User Confirmation	Confirmation via Email or via otp.
FR-3	Tracking Containment Zones	Using Geo-fence sketching.
FR-4	Notification alert system	Using Sendgrid and Notification services.

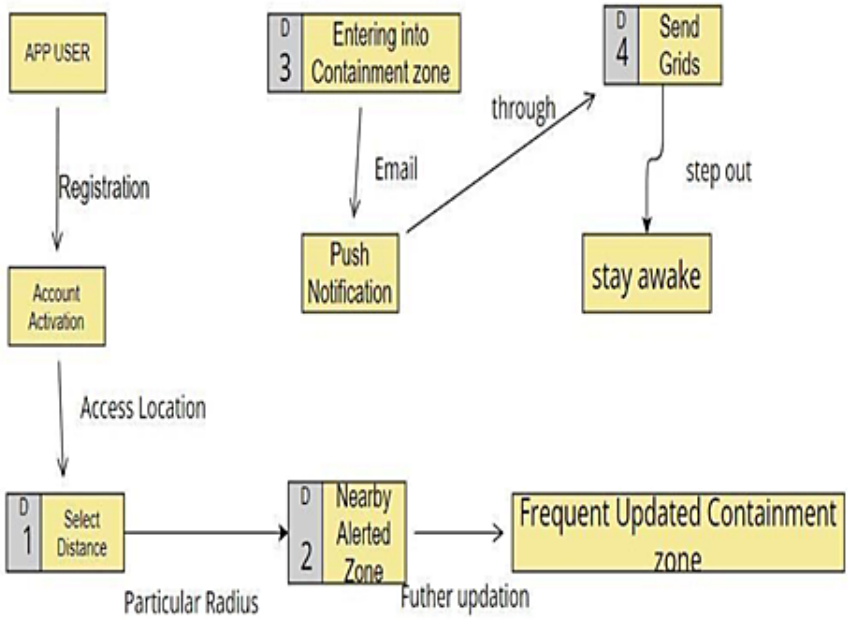
4.2 Non-functional Requirements

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	User interface is clean and easy to use.
NFR-2	Security	User data should be securely stored on the server.
NFR-3	Reliability	Most appropriate results can be obtained by using Geofencing and GPS.
NFR-4	Performance	User interface is responsive and real time location sharing will be accurate.
NFR-5	Availability	Reruired data should be provide without any interruption while on a good internet connection.
NFR-6	Scalability	User friendly and can be used in any environment.

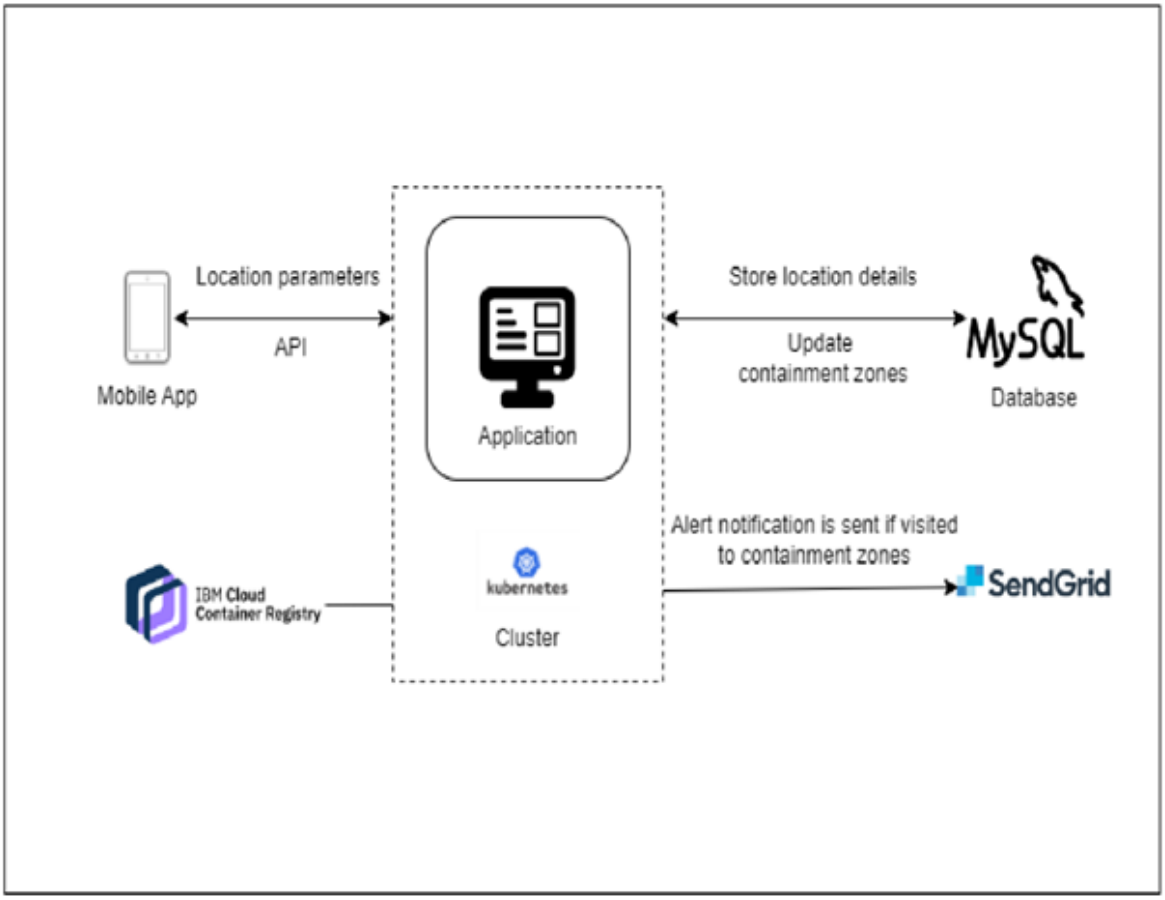
CHAPTER 5

PROJECT DESIGN

5.1 DATAFLOW DIAGRAM



5.2 Solution Architecture



Technical Architecture

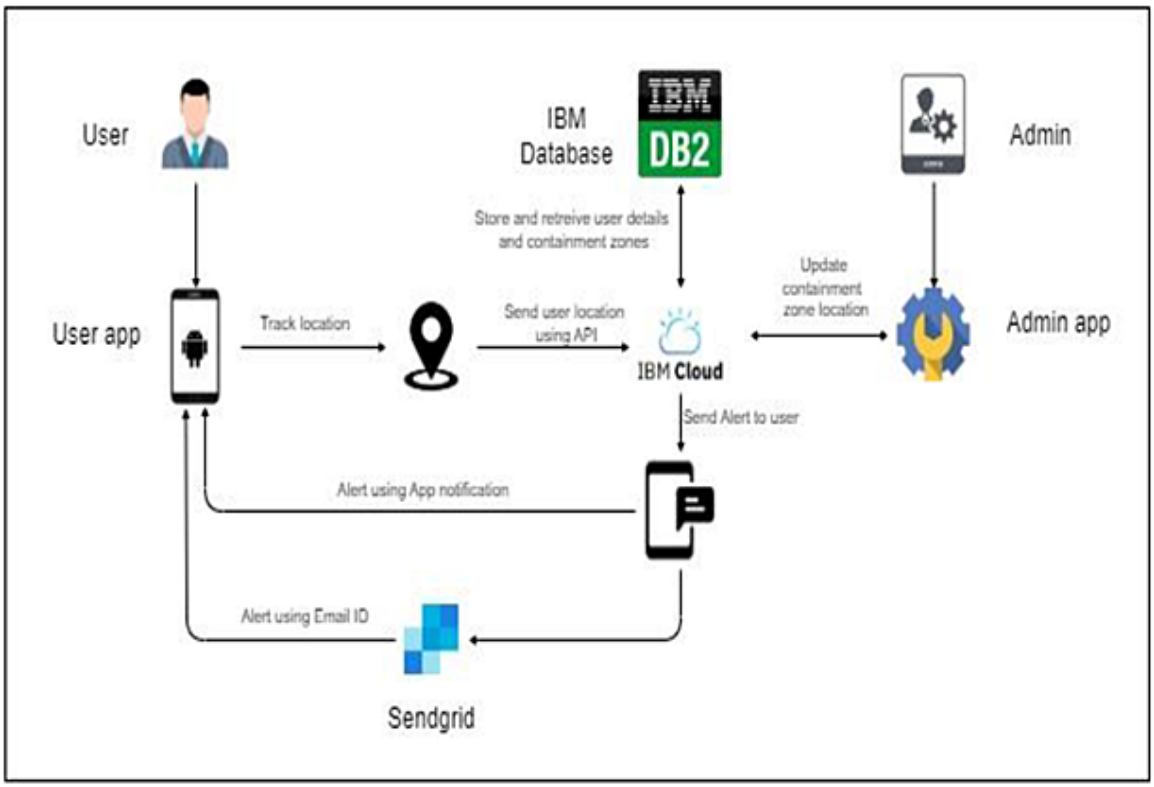


Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	User interacts with Web UI	HTML, CSS, JavaScript
2.	Application Logic-1	The application is developed using Flask	Java / Python-Flask
3.	Application Logic-2	The application is directly deployed in cloud.	IBM Watson STT Service.
4.	Database	The user credentials are stored	MySQL, NoSQL, etc.
5.	Cloud Database	Location of containment zones are stored	IBM DB2, IBMCloudant etc.
6.	File Storage	File storagerequirements	IBM Block Storage
7.	External API-1	Sends user location	IBM WeatherAPI, etc.
8.	Infrastructure (Server/ Cloud)	Application Deployment on Local System / CloudSystem	Local, Cloud Foundry, Kubernetes, etc.

Table-2 : Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Python is used for backend purpose and flask isimported for frontend purpose	Python, Flask
2.	Security Implementations	Data inside the system will be protected against unauthorized access	AES-256, HTTPS
3.	Scalable Architecture	It can be used in any environment and userfriendly	Kubernetes Cluster
4.	Availability	The application will be available24/7	IBM Cloudcontainer registry
5.	Performance	Fast and responsive	IBM Cloud

5.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail		Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password		High	Sprint-1
	Dashboard	USN - 6	As a User, I can manually plot the alerted zone for my convenience only.	It can be viewed in the user dashboard	Low	Sprint - 2
Customer (Web user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	User account activities can be viewed in dashboard.	High	Sprint - 2
			Confirmation code has been sent through the registered mail id or phone number			
	Location Access	USN - 2	As a User, I can view the zones by using location services	Location can be enabled through device settings	High	Sprint - 2
	Containment Zones	USN - 3	Alert messages will pop out immediately If I entered into the zone and the messages are properly received through email.	Alerted messages are sent using sendgrid through the registered mail id	High	Sprint - 3
Administrator	Frequent Updates	USN - 4	Admin should update the containment zones frequently through their portals and those can be viewed by the user,	It can be accessed by Geo fencing.	Medium	Sprint - 4

CHAPTER 6

PROJECT PLANNING AND SCHEDULING

6.1 Sprint Planning and Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	User Registration	USN-1	As a user, I can register for the application by entering my username,mobile number, and email address.	10	High	Abikesh S, Adhil Ahamed H.
Sprint-1	User login	USN-2	As a user, I can log into the application by entering usernameand mobile number.	5	Medium	Gogila Naveen K
Sprint-1	Admin login	USN-3	As an admin, I would login the app and fetch the containment zone areas.	5	Medium	Dhinakar S
Sprint-2	User's location	USN-4	As a user, I must give my location access to the application.	10	High	Abikesh S, Adhil Ahamed H, Gogila Naveen K, Dhinakar S.
Sprint-2	Admin's access to user's location	USN-5	As an admin, I should track the user's location.	10	High	Gogila Naveen K, Dhinakar S.
Sprint-3	Updation of containment zone	USN-6	As an admin, I should update the location of the containment zone.	10	High	Abikesh S, Adhil Ahamed H, Gogila Naveen K
Sprint-3	Creation of Geofence	USN-7	A Geofence is created within a 100m radius around the containment zone.	5	Medium	Dhinakar S
Sprint-4	Monitoring user's location	USN-8	As an admin, I should monitor the user's location and alert him with a notification when he or she enters the containment zone.	10	High	Dhinakar S, Abikesh S.
Sprint-4	Notification alert	USN-9	When the user enters the containment zone, he or she will receive an alert notification.	5	Medium	Adhil Ahamed H, Gogila Naveen K.

6.2 Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	15	6 Days	7 Nov 2022	12 Nov 2022	15	12 Nov 2022
Sprint-4	15	6 Days	14 Nov 2022	19 Nov 2022	15	14 Nov 2022

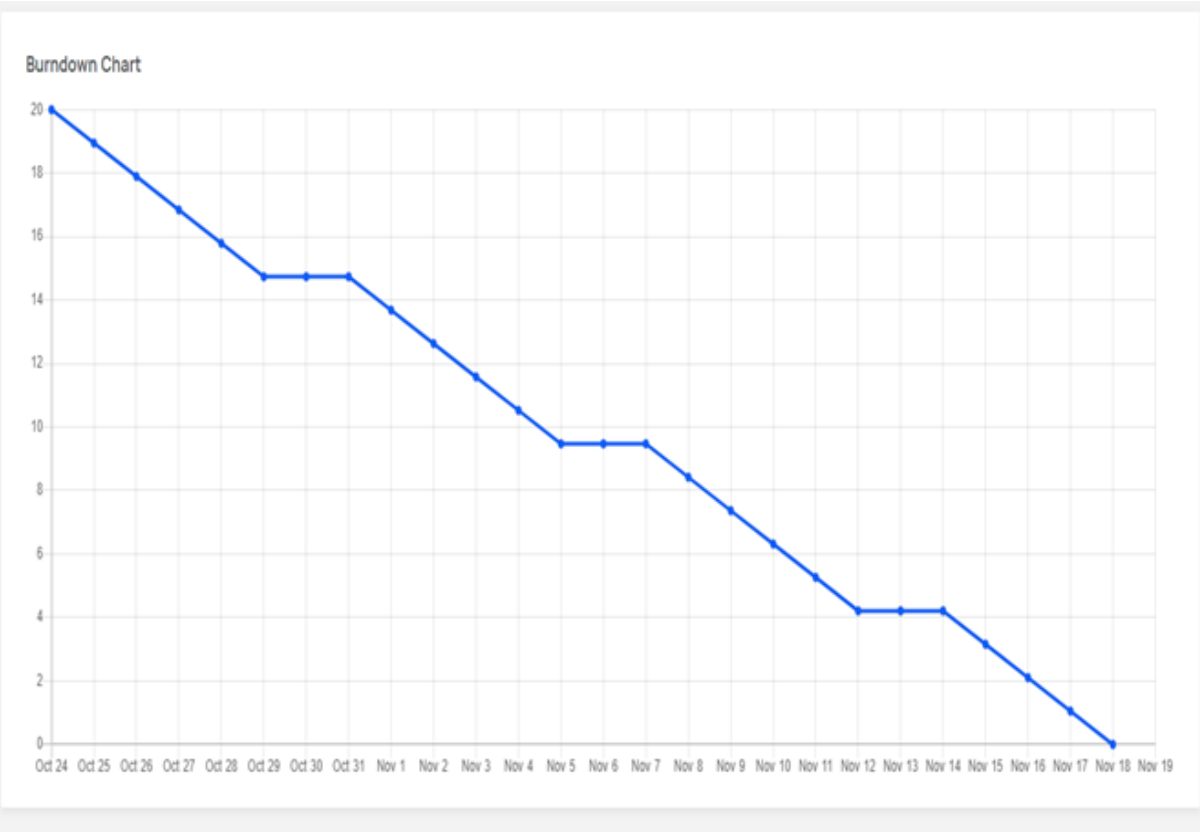
Velocity:

Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

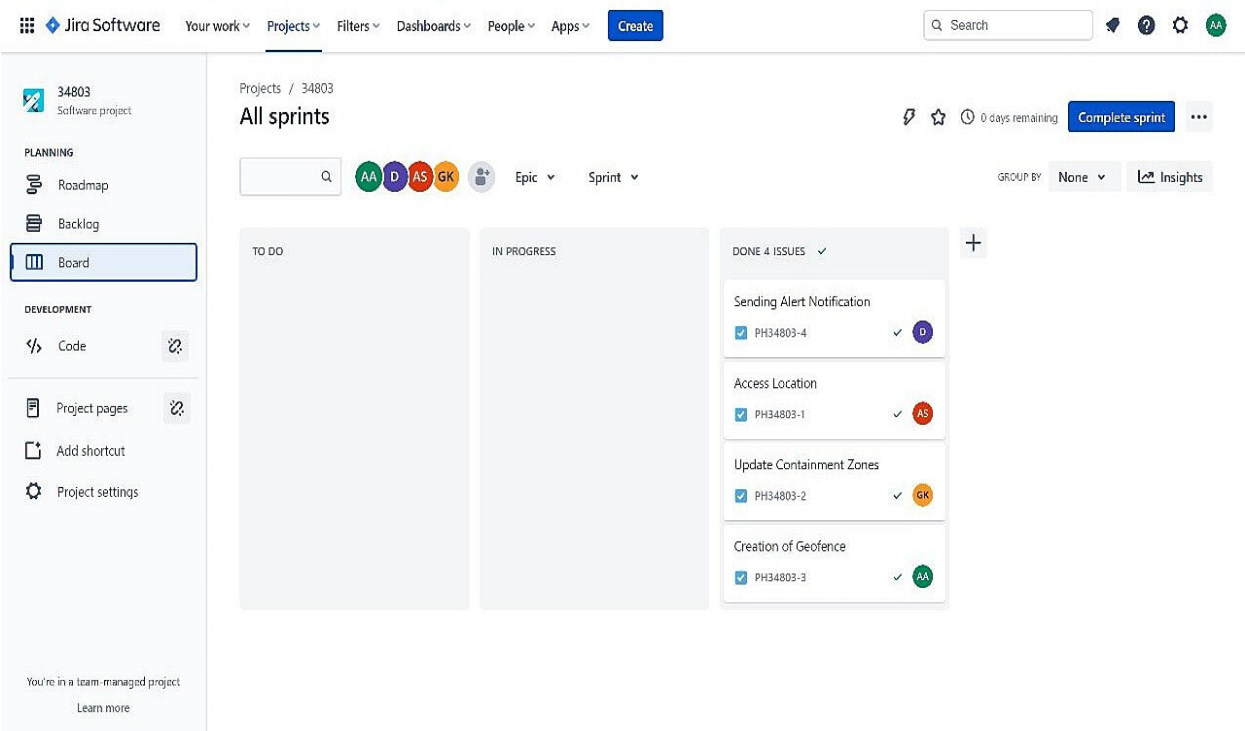
AV= Sprint duration velocity = 6/20 = 0.3

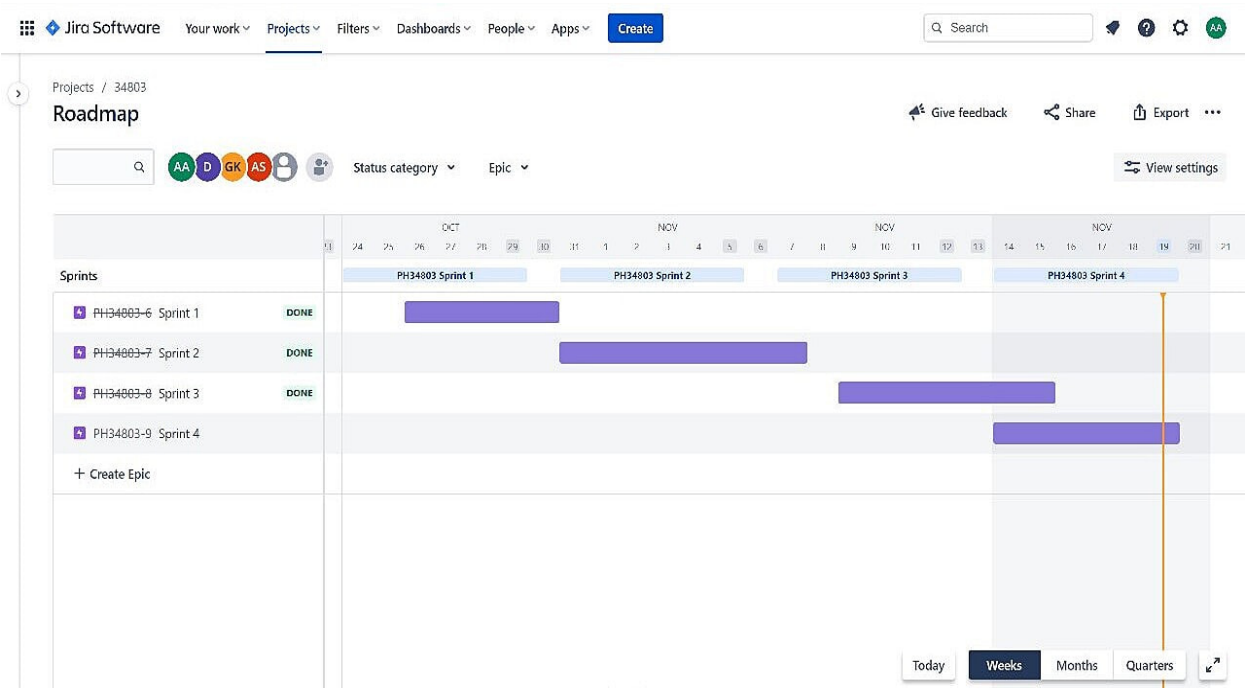
6.3 Burn down Chart:

A burn down chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies suchas Scrum. However, burn down chartscan be applied to any project containing measurable progress over time.



6.3 Reports From JIRA





CHAPTER 7

CODING AND SOLUTIONING

7.1 Login.html

```
<!DOCTYPE html>
<html lang="en">

<head>
   <!-- Required meta tags -->
   <meta charset="utf-8" />
   <meta name="viewport" content="width=device-width, initial-scale=1,
shrink-to-fit=no" />

   <!-- Bootstrap CSS -->
   <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstra
p.min.css"
      integrity="sha384-
Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf
23Q9Ifjh" crossorigin="anonymous" />
   <link rel="stylesheet" href="style.css" />
   <title>Log In</title>
   <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
</head>

<body class="text-center">
   {% if error == 1 %}
   <script>
      alert("Incorrect Password");
   </script>
```

```
{% elif error == 2%}
<script>
    alert("Create An Account");
</script>
{% else %}
{% endif %}
<form class="form-login" method="POST" action="/">
    <h1 class="h3 mb-3 font-weight-normal">Log In to add the location
of the containment zone</h1>
    <label for="email" class="sr-only">Email address</label>
    <input type="email" name="email" class="form-control"
placeholder="Email address" required autofocus />
    <label for="password" class="sr-only">Password</label>
    <input type="password" class="form-control"
placeholder="Password" name="password" required />
    <button type="submit" class="btn btn-lg btn-primary btn-block mt-3">
        Login
    </button>
    <a href={{url_for("signup")}}>Don't have an account ... Create
One</a>
</form>

<!-- Optional JavaScript -->
<!-- jQuery first, then Popper.js, then Bootstrap JS -->
<script src="https://code.jquery.com/jquery-3.4.1.slim.min.js"
    integrity="sha384-
J6qa4849blE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1yYfoR
SJoZ+n"
    crossorigin="anonymous"></script>
<script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.mi
n.js"
    integrity="sha384-
Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxM
```

```
fooAo"
        crossorigin="anonymous"></script>
    <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.
min.js"
        integrity="sha384-
wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3Og
8ifwB6"
        crossorigin="anonymous"></script>
</body>
</html>
```

7.2 signup.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <!-- Required meta tags -->
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1,
shrink-to-fit=no" />

    <!-- Bootstrap CSS -->
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstra
p.min.css"
        integrity="sha384-
Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf
23Q9Ifjh" crossorigin="anonymous" />
    <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
    <title>Sign Up</title>
</head>
```

```
<body class="text-center">
   {% if error %}
   <script>
      alert("Email id already exists in the database");
   </script>
   {% endif %}
   <form class="form-login" method="POST" action="/signup">
      <h1 class="h3 mb-3 font-weight-normal">Sign Up to create an
account with us</h1>
      <label for="name" class="sr-only">Email address</label>
      <input type="text" name="name" class="form-control"
placeholder="Name" required autofocus />
      <label for="email" class="sr-only">Email address</label>
      <input type="email" name="email" class="form-control"
placeholder="Email address" required />
      <label for="password" class="sr-only">Password</label>
      <input type="password" class="form-control"
placeholder="Password" name="password" required />
      <button type="submit" class="btn btn-lg btn-primary btn-block mt-3">
         Signup
      </button>
      <a href={{url_for("login")}}>Already have an account ... Login</a>
   </form>

   <!-- Optional JavaScript -->
   <!-- jQuery first, then Popper.js, then Bootstrap JS -->
   <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js"
      integrity="sha384-
J6qa4849blE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1yYfoR
SJoZ+n"
      crossorigin="anonymous"></script>
   <script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.mi
n.js"
```

```
        integrity="sha384-
Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxM
fooAo"
        crossorigin="anonymous"></script>
    <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.
min.js"
        integrity="sha384-
wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3Og
8ifwB6"
        crossorigin="anonymous"></script>
</body>

</html>
```

7.3 home.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Document</title>
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstra
p.min.css"
        integrity="sha384-
Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf
23Q9Ifjh" crossorigin="anonymous" />
    <style>
        body {
```

```
            padding-top: 30px;
            padding-bottom: 30px;
            background-color: #699cc5;
        }

        a {
            color: black;
        }
    </style>
</head>

<body>
    {% if success == True %}
    <script>
        alert("Location Uploaded Successfully");
    </script>
    {% elif success == 0 %}
    <script>
        alert("Enter Proper Location data");
    </script>
    {% endif %}
    <div class="m-3 float-right">
        <button type="button" class="btn btn-primary"><a
href={{url_for("logout")}}>Log Out</a></button>
    </div>
    <div class="container m-3">
        <h1><u>Declare Containment Zone</u></h1>
    </div>
    <div class="container m-3">
        <h3>welcome: {{name}}</h3>
    </div>
    <form method="POST" action="/home">
        <div class="container">
            <div class="form-group row">
```

```
            <div class="col-sm-6">
                <label class="control-label">Lat.:</label>
                <input type="text" class="form-control" id="lat" name="lat" />
            </div>
            <div class="col-sm-6">
                <label>Long.:</label>
                <input type="text" class="form-control" id="lon" name="lon" />
            </div>
            <div class="col-sm-6">
                <label>Get current Location:</label>
                <button type="button" class="btn btn-warning"
onclick="getLocation()">Current Location</button>
                <label>(Click this first)</label>
            </div>
        </div>

        <!-- map -->
        <div id="map_disp" style="height: 400px;width: 500px;"></div>
        <div class="m-3 float-right">
            <button type="submit" class="btn btn-danger">Declare
Containment Zone</button>
        </div>
        <div class="m-3">
            <button onclick="toggleTips()" type="button" class="btn btn-
secondary">Tutorial</button>
            <div id="tips" class="m-3">
                <ol>
                    <li>Select The Location By Clicking the Current Location
Button</li>
                    <li>Drag the Pin to change the location</li>
                    <li>Click on Declare Containment Zone to save the location
to the database </li>
                </ol>
            </div>
```

```
        </div>
        <div class="m-3 float-right">
            <button type="button" class="btn btn-warning"><a
href="{{url_for('data')}}">Click Here To View The
                Containment Zones and Number of
                people visited</a></button>
        </div>
    </div>

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/js/bootstrap.min
.js"
        integrity="sha384-
+YQ4JLhjyBLPDQt//I+STsc9iw4uQqACwlvpslubQzn4u2UU2UFM80nGisd
026JF"
        crossorigin="anonymous"></script>
    <script src="https://code.jquery.com/jquery-2.2.4.min.js"></script>
    <script
src="https://maps.google.com/maps/api/js?sensor=false&amp;libraries
=places"></script>
    <script
        src="https://rawgit.com/Logicify/jquery-locationpicker-
plugin/master/dist/locationpicker.jquery.js"></script>

    <script>
        function getLocation() {
            if (navigator.geolocation) {
                navigator.geolocation.getCurrentPosition(showPosition);
            } else {
                alert("No location");
            }
        }
        function showPosition(position) {
            $('#map_disp').locationpicker({
```

```
                location: {
                    latitude: position.coords.latitude,
                    longitude: position.coords.longitude
                },
                radius: 0,
                inputBinding: {
                    latitudeInput: $('#lat'),
                    longitudeInput: $('#lon'),
                },
                enableAutocomplete: true,
                onchanged: function (currentLocation, radius,
isMarkerDropped) {
                    // Uncomment line below to show alert on each Location
Changed event
                    // alert("Location changed. New location (" +
currentLocation.latitude + ", " + currentLocation.longitude + ")");
                }
            });
        }
        function toggleTips() {
            var x = document.getElementById("tips");
            if (x.style.display === "none") {
                x.style.display = "block";
            } else {
                x.style.display = "none";
            }
        }
    </script>
</body>

</html>
```

7.4 data.html

```
<!DOCTYPE html>
<html lang="en">

<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-
scale=1.0">
   <title>Zones</title>
   <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstra
p.min.css"
      integrity="sha384-
Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf
23Q9Ifjh" crossorigin="anonymous" />
   <style>
      body {
         padding-top: 30px;
         padding-bottom: 30px;
         background-color: #699cc5;
      }

      a {
         color: black;
      }
   </style>
</head>

<body>
   <div class="m-4 container">
      <h1><u>Location data and Visited People</u></h1>
```

```
    </div>
    <div class="m-4 container">
        <table class="table">
            <thead>
                <tr>
                    <th scope="col">S.No</th>
                    <th scope="col">Latitude</th>
                    <th scope="col">Longitude</th>
                    <th scope="col">No_Visited</th>
                </tr>
            </thead>
            <tbody>

                {%- for row in responses %}
                <tr>
                    <th scope="row">{{loop.index}}</th>
                    <td>{{row[1]}}</td>
                    <td>{{row[2]}}</td>
                    <td>{{row[3]}}</td>
                </tr>
                {%- endfor %}
            </tbody>
        </table>
    </div>
    <div class="m-3 float-right">
        <button type="button" class="btn btn-danger"><a
href={{url_for("home")}}>Go to location update Page</a></button>
    </div>

</body>

</html>
```

7.5 Signup.java

```
package com.example.client_containment;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.EditText;



import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.Response;
import com.android.volley.VolleyError;

import com.android.volley.toolbox.JsonObjectRequest;

import com.android.volley.toolbox.Volley;

import org.json.JSONException;
import org.json.JSONObject;



public class SignUp extends AppCompatActivity {
    private EditText name;
    private EditText email;
    private EditText password;
```

```
    SharedPreferences sharedpreferences;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_sign_up);
        name = findViewById(R.id.name);
        email = findViewById(R.id.email);
        password = findViewById(R.id.password);

        sharedpreferences =
getApplicationContext().getSharedPreferences("user_data", 0);
        SharedPreferences.Editor editor = sharedpreferences.edit();
        editor.clear();
        editor.commit();
        if(sharedpreferences.getAll().size() >= 3){
            Intent intent = new Intent(this,MainActivity.class);
            startActivity(intent);
        }

    }

    public void signUp(View view) {
        if(!name.getText().equals("") || !email.getText().equals("") ||
!password.getText().equals("")){

postDataUsingVolley(name.getText().toString(),email.getText().toString()
,password.getText().toString());
        }
    }
    private void postDataUsingVolley(String name, String email,String
password) {
        final RequestQueue queue = Volley.newRequestQueue(this);
```

```
String url = "http://192.168.161.115:5000/android_sign_up";

JSONObject postparams = new JSONObject();
try {
    postparams.put("name", name);
    postparams.put("email", email);
    postparams.put("password", password);
} catch (JSONException e) {
    e.printStackTrace();
}

JsonObjectRequest jsonObjReq = new
JsonObjectRequest(Request.Method.POST, url, postparams,
        new Response.Listener<JSONObject>() {
            @Override
            public void onResponse(JSONObject response) {
                Log.d("response",response.toString());

                try {
                    int userId = response.getInt("id");

                    SharedPreferences.Editor editor =
sharedpreferences.edit();

                    editor.putString("name", name);
                    editor.putString("email", email);
                    editor.putInt("id", userId );
                    editor.commit();
                    Intent intent = new Intent(SignUp.this,MainActivity.class);
                    startActivity(intent);

                } catch (JSONException e) {
                    e.printStackTrace();
                }
```

```
            }
        },
        new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
                Log.d("error",error.toString());
            }
        });

    queue.add(jsonObjReq);
  }
}
```

7.6 MainActivity.java

```
package com.example.client_containment;

import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;

import android.Manifest;
import android.annotation.SuppressLint;
import android.app.PendingIntent;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.util.Log;
import android.widget.TextView;
import android.widget.Toast;

import com.example.client_containment.Service.MyLocationService;
import com.google.android.gms.location.FusedLocationProviderClient;
```

```
import com.google.android.gms.location.LocationRequest;
import com.google.android.gms.location.LocationServices;
import com.karumi.dexter.Dexter;
import com.karumi.dexter.PermissionToken;
import com.karumi.dexter.listener.PermissionDeniedResponse;
import com.karumi.dexter.listener.PermissionGrantedResponse;
import com.karumi.dexter.listener.PermissionRequest;
import com.karumi.dexter.listener.single.PermissionListener;


public class MainActivity extends AppCompatActivity {
    @SuppressLint("StaticFieldLeak")
    static MainActivity instance;
    LocationRequest locationRequest;
    TextView loc;
    FusedLocationProviderClient fusedLocationProviderClient;
    SharedPreferences sharedPreferences;

    public static MainActivity getInstance() {
        return instance;
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        instance = this;
        loc = findViewById(R.id.location_text);
        sharedPreferences =
getApplicationContext().getSharedPreferences("user_data", 0);
        Log.d("shared_pref",sharedPreferences.getString("name","0"));
        Dexter.withActivity(this)

.withPermission(Manifest.permission.ACCESS_FINE_LOCATION)
```

```
            .withListener(new PermissionListener() {
                @Override
                public void
onPermissionGranted(PermissionGrantedResponse response) {
                    updateLocation();
                }

                @Override
                public void onPermissionDenied(PermissionDeniedResponse
response) {
                    Toast.makeText(MainActivity.this, "No location",
Toast.LENGTH_LONG).show();
                }

                @Override
                public void
onPermissionRationaleShouldBeShown(PermissionRequest permission,
PermissionToken token) {

                }
            }).check();

    }
    private void updateLocation() {
        buildLocationRequest();
        fusedLocationProviderClient =
LocationServices.getFusedLocationProviderClient(this);
        if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
            // TODO: Consider calling
```

```
            //    ActivityCompat#requestPermissions
            // here to request the missing permissions, and then overriding
            //   public void onRequestPermissionsResult(int requestCode,
String[] permissions,
            //                                          int[] grantResults)
            // to handle the case where the user grants the permission. See
the documentation
            // for ActivityCompat#requestPermissions for more details.
            return;
        }

fusedLocationProviderClient.requestLocationUpdates(locationRequest,
getPendingIntent());
    }
    private PendingIntent getPendingIntent() {
        Intent intent = new Intent(this, MyLocationService.class);
        intent.setAction(MyLocationService.ACTION_PROCESS_UPDATE);
        return getPendingIntent(intent);
    }

    private PendingIntent getPendingIntent(Intent intent)
     {
        return PendingIntent.getBroadcast(this, 0, intent,
PendingIntent.FLAG_UPDATE_CURRENT);
     }

    private void buildLocationRequest(){
        locationRequest = new LocationRequest();
        locationRequest
.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
        locationRequest.setInterval(3000);
        locationRequest.setFastestInterval(1000);
        locationRequest.setSmallestDisplacement(10f);
    }
```

```
    public void updateTextView(String location){
        MainActivity.this.runOnUiThread(new Runnable()
        {
            @Override
            public void run() {
                loc.setText(location);
            }
        });
    }
}
```

7.7 App.py

```
# import statements
from logging import error
from flask import *
from jinja2.utils import select_autoescape
import bcrypt
from flask_mysqldb import MySQL
import json
from sendgrid import SendGridAPIClient
from sendgrid.helpers.mail import Mail


# initialization
app = Flask(__name__)

# config
app.secret_key =
"\x19Ts\xbe\xe7\x8c_\r\x12Q\x14\x13>q\xb7'WTH0\x9f\xe4\xec\xb1"
app.config['MYSQL_HOST'] = 'localhost'
app.config['MYSQL_USER'] = 'root'
app.config['MYSQL_PASSWORD'] = ''
app.config['MYSQL_DB'] = 'zone2'
```

```
mysql = MySQL(app)

# functions


def send_mail(email):
    print(email)
    message = Mail(from_email='varundutia.h@gmail.com',
                   to_emails=email,
                   subject='caution',
                   plain_text_content='Please Stay Safe',
                   html_content='<h2>You are entering into a containment
Zone</h2>')

    try:
        sg = SendGridAPIClient(

'SG.7BJDtQDlS8unH0r5_TufVQ.Ykpcz19QcqgcNwYZC3a0mNRPhGksG1
17YURqOTa2HL')
        response = sg.send(message)
        print(response.status.code)
        print(response.body)
        print(response.headers)
    except Exception as e:
        print(e)


def create_bcrypt_hash(password):
    # convert the string to bytes
    password_bytes = password.encode()
    # generate a salt
    salt = bcrypt.gensalt(14)
    # calculate a hash as bytes
```

```
    password_hash_bytes = bcrypt.hashpw(password_bytes, salt)
    # decode bytes to a string
    password_hash_str = password_hash_bytes.decode()
    return password_hash_str


def verify_password(password, hash_from_database):
    password_bytes = password.encode()
    hash_bytes = hash_from_database.encode()

    # this will automatically retrieve the salt from the hash,
    # then combine it with the password (parameter 1)
    # and then hash that, and compare it to the user's hash
    does_match = bcrypt.checkpw(password_bytes, hash_bytes)

    return does_match

# Api's


@app.route("/", methods=["GET", "POST"])
def login():
    if(request.method == "POST"):

        # get the data from the form
        password = request.form['password']
        email = request.form['email']

        # initialize the cursor
        signup_cursor = mysql.connection.cursor()

        # check whether user already exists
        user_result = signup_cursor.execute(
            "SELECT * FROM USERS WHERE user_email=%s", [email]
```

```
        )

        if(user_result > 0):
            data = signup_cursor.fetchone()
            data_password = data[3]
            if(verify_password(password, data_password)):
                signup_cursor.close()
                session['id'] = data[0]
                session['name'] = data[1]
                session['email'] = data[2]
                return redirect(url_for("home"))
            else:
                return render_template('login.html', error=1)
        else:
            return render_template('login.html', error=2)
    return render_template('login.html', error=3)


@app.route("/signup", methods=["POST", "GET"])
def signup():
    if(request.method == "POST"):

        # get the data from the form
        name = request.form['name']
        email = request.form['email']
        password = request.form['password']

        # hash the password
        pw_hash = create_bcrypt_hash(password)

        # initialize the cursor
        signup_cursor = mysql.connection.cursor()

        # check whether user already exists
```

```
        user_result = signup_cursor.execute(
            "SELECT * FROM USERS WHERE user_email=%s", [email]
        )
        if(user_result > 0):
            signup_cursor.close()
            return render_template('signup.html', error=True)
        else:
            # execute the query
            signup_cursor.execute(
                'INSERT INTO
USERS(user_name,user_email,user_password,user_type)
VALUES(%s,%s,%s,%s)', (
                    name, email, str(pw_hash), "2"
                )
            )

            mysql.connection.commit()
            signup_cursor.close()
            return redirect(url_for('login'))

    return render_template('signup.html', error=False)


@app.route("/home", methods=["POST", "GET"])
def home():
    if(session['id'] == None):
        return redirect(url_for('login'))

    if(request.method == "POST"):
        # get data
        lat = request.form["lat"]
        lon = request.form["lon"]
        vis = 0
        if(lat == "" or lon == ""):
```

```
        return render_template('home.html', name=session['name'],
email=session['email'], id=session['id'], success=0)

    # create a location cursor
    location_cursor = mysql.connection.cursor()

    # Execute the query
    location_cursor.execute(
        'INSERT INTO
LOCATION(location_lat,location_long,location_visited)
VALUES(%s,%s,%s)', (
            lat, lon, vis
        )
    )
    mysql.connection.commit()
    location_cursor.close()
    return render_template('home.html', name=session['name'],
email=session['email'], id=session['id'], success=True)
  return render_template('home.html', name=session['name'],
email=session['email'], id=session['id'])


@app.route("/logout")
def logout():
  # remove the username from the session if it is there
  session['id'] = None
  session['name'] = None
  session['email'] = None
  return redirect(url_for('login'))


@app.route("/data")
def data():
  if(session['id'] == None):
```

```
        return redirect(url_for('login'))

    location_cursor = mysql.connection.cursor()

    # check whether user already exists
    user_result = location_cursor.execute(
        "SELECT * FROM LOCATION"
    )
    if(user_result == 0):
        return render_template("data.html", responses=0)
    else:
        res = location_cursor.fetchall()
        print(res)
        return render_template("data.html", responses=res)


@app.route("/android_sign_up", methods=["POST"])
def upload():
    if(request.method == "POST"):

        # get the data from the form
        name = request.json['name']
        email = request.json['email']
        password = request.json['password']

        # hash the password
        pw_hash = create_bcrypt_hash(password)

        # initialize the cursor
        signup_cursor = mysql.connection.cursor()

        # check whether user already exists
        user_result = signup_cursor.execute(
            "SELECT * FROM USERS WHERE user_email=%s", [email]
```

```
        )
        if(user_result > 0):
            signup_cursor.close()
            return {'status': 'failure'}
        else:
            # execute the query
            signup_cursor.execute(
                'INSERT INTO
USERS(user_name,user_email,user_password,user_type)
VALUES(%s,%s,%s,%s)', (
                    name, email, str(pw_hash), "1"
                )
            )

            mysql.connection.commit()
            id_result = signup_cursor.execute(
                'SELECT user_id FROM USERS WHERE user_email = %s', [email]
            )
            if(id_result > 0):
                id = signup_cursor.fetchone()
                return {"id": id[0]}
            signup_cursor.close()

    return {"status": "failure"}


@app.route("/get_all_users")
def getusers():
    signup_cursor = mysql.connection.cursor()

    # check whether user already exists
    user_result = signup_cursor.execute(
        "SELECT * FROM USERS"
    )
```

```
    if(user_result > 0):
        rv = signup_cursor.fetchall()
        row_headers = [x[0] for x in signup_cursor.description]
        json_data = []
        for result in rv:
            json_data.append(dict(zip(row_headers, result)))
        return json.dumps(json_data)


@app.route("/post_user_location_data", methods=["POST"])
def post_user_location():
    if(request.method == "POST"):

        # get the data from the form
        lat = request.json['lat']
        lon = request.json['long']
        id = request.json['id']
        ts = request.json['timestamp']

        # initialize the cursor
        user_location_cursor = mysql.connection.cursor()

        # execute the query
        user_location_cursor.execute(
            'INSERT INTO
USER_LOCATION(location_lat,location_long,user_id,timestamp)
VALUES(%s,%s,%s,%s)', (
                lat, lon, id, ts
            )
        )

        mysql.connection.commit()

        return {"response": "success"}
```

```
@app.route("/location_data")
def location_data():
    location_cursor = mysql.connection.cursor()

    # check whether user already exists
    user_result = location_cursor.execute(
        "SELECT * FROM LOCATION"
    )
    if(user_result != 0):
        res = location_cursor.fetchall()
        print(res)
        row_headers = [x[0] for x in location_cursor.description]
        json_data = []
        for result in res:
            json_data.append(dict(zip(row_headers, result)))
        return json.dumps(json_data)
    else:
        return {"response": "failure"}


@app.route("/send_trigger", methods=["POST"])
def send_trigger():
    if(request.method == "POST"):
        # get the data from the form
        email = request.json['email']
        location_id = request.json['id']
        location_cursor = mysql.connection.cursor()

        # check whether user already exists
        user_result = location_cursor.execute(
            "SELECT location_visited FROM LOCATION WHERE
location_id=%s", [
```

```
            location_id]
        )
        if(user_result == 0):
            return {"response": "failure"}
        else:
            res = location_cursor.fetchone()
            print(res[0])
            visited = res[0]
            visited = visited+1
            location_cursor.execute(
                "UPDATE LOCATION SET location_visited = %s WHERE
location_id=%s",
                (visited, location_id)
            )
            mysql.connection.commit()

        send_mail(email)
        return {"response": "success"}


# main
if __name__ == "__main__":
    app.run(host='0.0.0.0', port=5000)
```

CHAPTER 8

TESTING

8.1 Test Cases

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status
LoginPage_TC_OO2	UI	Home Page	Verify the UI elements in Login/Signup popup	Username & Password	1. Open the website 2.Enter details and press login 3.Verify that users are notified of login process		Users should be notified of login process	Working as expected	Pass
LoginPage_TC_OO3	Functional	Home page	Verify user is able to log into application with valid credentials		1. Open the website 2.Enter details and press login 3.Verify that users are logged into website properly	Username:User_admin password: admin	User should be logged into website properly	Working as expected	Pass
HomePage_TC_001	Functional	Home Page	Verify that user provide the current location properly		1. Open the website 2.User can able to mark his current location		user should provide the location as an input	Working as expected	Pass
HomePage_TC_OO 2	Functional	Home page	Verify that no.of visit displayed in the table		1. Open the website 2.Enter details and press login 3.No. Of visits to that location will be tabled		location data and visited people will be shown releated to that location	Working as expected	Pass
HomePage_TC_OO 3	Functional	Home page	Verify that when user will receive the mail properly		1. Open the website 2.Enter details and press login 3.Mail will be sent to the respective user		When the registered user enters into the containment zone, an alert message will be send	Working as expected	Pass

8.2 User Acceptance Testing

8.2.1 Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issuesof the [ProductName] project at the time of the release to User Acceptance Testing (UAT).

8.2.2 Defect Analysis

This report showsthe number of resolved or closed bugs at each severity level, and how they were resolved.

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	8	3	5	7	23
Duplicate	1	0	3	0	4
External	2	2	0	1	5
Fixed	7	4	7	14	32
Not Reproduced	0	0	1	1	2
Skipped	0	2	0	1	3
Won't Fix					

	0	1	1	3	5
Totals	18	12	17	26	73

8.2.3 Test Case Analysis

This report shows the number of test cases that have passed, failed,and untested.

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	5	0	1	4
Client Application	28	0	3	25
Security	2	0	0	2
Outsource Shipping	5	1	0	4
Exception Reporting	8	0	1	7
Final Report Output	4	1	0	3
Version Control	2	0	0	2

CHAPTER 9

RESULTS

In this project,we found that we can provide the notification if the user enter into the containment zone. Using geofencing technology we can update the containment zones.

Final findings(Output) of the project along with screenshots as follows.

9.1 Performance Metrics
Model Performance Testing

			NFT - Risk Assessment					
S.No	Project Name	Scope/feature	Functional Changes	Hardware Changes	Software Changes	Impact of Downtime	Load/Volume Changes	Risk Score
1	Containment Zone Alerting Application	New	Low	No Changes	Moderate	Yes, 2hrs	>10 to 30%	GREEN

NFT - Detailed Test Plan				
S.No	Project Overview	NFT Test approach	Assumptions/Dependencies/Risks	Approvals/SignOff
1	Login Page	1) Open the Containment Zone Alerting Application 2) Login with user Credentials	No Risks	N/A
2	Signup Page	1) Open the Containment Zone Alerting Application 2) Enter the Details and Create a new User	No Risks	N/A
3	Dashboard	1) Log in to Containment Zone Alerting Application 2) User provides the current locatio as an input	No Risks	N/A
4	Records	1) Log in to Containment Zone Alerting Application 2) Location data and visited people will be shown	No Risks	N/A
5	Report	1) Log in to Containment Zone Alerting Application 2)Admin declares the containment zone as a report.	No Risks	N/A
5	Email Acknowledgement	1) Mails are Sent to the Registered user if the user enters into the containment zone	No Risks	N/A

End Of Test Report								
S.No	Project Overview	NFT Test approach	NFR - Met	Test Outcome	GO/NO-GO decision	Recommendations	Identified Defects (Detected/Closed/Open)	Approvals/SignOff
1	Containment Zone Alerting	1) Log in to Containment Zone Alerting Application 2) Test for all Testcases 3) Log out to Containment Zone Alerting Application	YES	Test Passed	GO/NO-GO decision	N/A	None	N/A

CHAPTER 10

ADVANTAGES & DISADVANTAGES

Advantages

By using this application , every user can avoid entering the containment zone. Since the application is free of cost , all users are able to download and utilize the application. The app seeks to simplify the contact-tracing procedure. Contact tracing is an essential tool for diminishing the spread of infectious disease. The application is quite simple and easy to use as it works upon database analysis. The user just have to enter the location they are willing to go. In other containment zone alerting applications, the users have to give their location access and the admin in the portal application has to track and monitor the user's location. When the user enters the containment zone , the admin sends an alert message. In those kind of applications , the users are not traceable ,when they turn off their location or run out of battery. During these conditions these apps fail to achieve the goals. But in our containment zone alerting system, the user's request is processed at once. There is no requirement of accessing the location of users. So there are no privacy issues for the users. The application is compatible in all android applications.

Disadvantages

Eventhough there is no need for location access , the admins are able to know the location where the users are planning to go. So the privacy of the users are not always guaranteed. There are risks of security issues , because other third party applications can hack the application. The transmission of the message could be delayed when the network is down. People who don't have an android smart phone will not be able to utilize the application. People who don't have English knowledge will also not be able to utilize the application effectively.

CHAPTER 11

CONCLUSION

The containment zone alerting system provides an efficient way of showing the identified Covid-19 containment zones to the users. With an alarming increase of Covid-19 affected cases throughout the world , this developed application can be employed as a tool for creating further social awareness among the people. This application always updates the current containment zones in the database and further checks the user's location whether he or she is about to enter the containment zones. It sends a separate notification alert to the user before entering the containment zones. The application has been tested in various locations and has been found to yield accurate results.

CHAPTER 12

FUTURE SCOPE

The application can be further used for many purposes like maritime and forest safety to prevent users from entering restricted areas , because there are no effective applications for warning the users in such cases. It can also be used to identify people who enter other states, without proper E-pass during lockdown. The application can be further enhanced with advanced features like tracking the users who violate the security warnings and report them to the police and other respective departments. By increasing the punishments , people will be afraid to violate the security polices. Overall by handing this application to the government , the government will get a good economic growth.

CHAPTER 13

APPENDIX

SOURCE CODE LINK

https://drive.google.com/drive/folders/1gDziS1IDbmICgT2c0tABq3XmlOZXIcIQ

GitHub link

https://github.com/IBM-EPBL/IBM-Project-31617-1660203597/tree/main/Final%20Deliverables/Final%20Code

PROJECT DEMO VIDEO LINK

YouTube Link

https://youtu.be/g6z97ShCqYc