Prototype II Development and Test Plan

Project Group: 10

Course: GNG 1103 - Engineering Design

Faculty: Faculty of Engineering, University of Ottawa

Date: November 8, 2024

Table of Contents

1. Introduction
2. Prototyping Objectives

    2.1 Purpose of Prototype II

    2.2 Goals of Prototype II

3. Analysis and Feedback

    3.1 Prototype II Results and Observations

    3.2 Key Insights from Feedback

4. Test Plan for Prototype III

    4.1 Objective

    4.2 Test Cases and Expected Outcomes

    4.3 Procedure

    4.4 Analysis Method

5. Project Task Plan

    5.1 Task Breakdown and Assignment

    5.2 Project Schedule

6. Transfer of Knowledge
7. Conclusion

    Appendix

1. **Introduction**

The objective of this document is to outline the development process, objectives, and testing plan for Prototype II, focusing on advancing our project's critical subsystem: **Restricted Zone Access (ZRA)**, which is the subsystem that restricts bandwidth, etc. when the device enters the wrong zone or leaves the correct zone. This prototype builds upon insights from Prototype I, with improvements informed by Shabodi, previous testing, and user feedback. The document includes a detailed analysis of Prototype II's performance, a testing plan for Prototype III, and a project task plan.

2. **Prototyping Objectives**

2.1 Purpose of Prototype II

The purpose of Prototype II is to refine and validate the Restricted Zone Access (ZRA) subsystem, a crucial component for controlled area access based on user authorization and location data. This prototype directly addresses **authorization level checks** and **real-time location validation** to ensure that only users with adequate permissions can access restricted zones.

2.2 Goals of Prototype II

1. **Accuracy in Access Control:** Ensure that access is granted only to users meeting the minimum required authority level for a given zone.

2. **Improved Location Validation:** Implement and validate accurate distance measurement to determine if the user is within the designated zone.

3. **Enhanced Notification System:** Develop a reliable notification feature to inform supervisors of access attempts (Central Control WebApp).

By creating a targeted prototype, our team aims to reduce system uncertainties and prepare a robust foundation for the final integrated system.

3. **Analysis and Feedback**

3.1 Prototype II Results and Observations

1. **Authority Validation:** Prototype II successfully implemented an authority validation function, achieving 95% accuracy in correctly granting or restricting access based on user roles.

2. **Location Validation:** The Haversine distance calculation formula was used to measure the user's proximity to zone boundaries with +/- %% accuracy. This precision level is sufficient for real-time zone access control.

3. Notification Feedback: Using a HTML coded web app run with Flask, supervisors receive notifications within 2 seconds, meeting our time response criteria. Feedback from "supervisors" (users) indicated that immediate access alerts significantly improved oversight capabilities.

3.2 Key Insights from Feedback

- **Bandwidth Restrictions**: Suggested reducing bandwidth to almost zero to implement more control on what can and can't be done with devices inside incorrect zones or as soon as they leave correct zones.

- **Authority Adjustments:** Suggested finer authority levels to better align with different zones, which will be incorporated in Prototype III.

- **Location Sensitivity:** The system performed well in open environments, but had minor issues in signal-limited areas. Enhancing location accuracy under a variety of conditions is necessary for future development.

4. **Test Plan for Prototype III**

4.1 Objective

The objective of Prototype III is to integrate feedback from Prototype II to further validate **Restricted Zone Access** through a controlled series of tests. These tests will focus on authority validation, location sensitivity, and the readability of the notification system.

4.2 Test Case and Expected Outcomes

Test Case ID	Description	Expected Outcome
TC-01	Authority validation at minimum level	Access Granted
TC-02	Authority below required level	Access Restricted
TC-03	User within zone boundary	Access Granted
TC-04	User outside zone boundary	Access Restricted
TC-05	Notification accuracy for granted access	Supervisor (User) notified within 2 seconds
TC-06	Notification accuracy for restricted access	Supervisor (User) notified within 2 seconds

4.3 Procedure

1. **Authority Level Testing:** Validate the authority level check by using a series of test users with different authority levels.

2. **Location Validation:** Conduct tests at different distances from the zone boundary to ensure accurate detection within the specified range.

3. **Notification Timing:** Measure the time taken for notifications to reach the supervisor to ensure compliance with the response time requirement.

4.4 Analysis Method

Data will be collected on access decisions, location accuracy, and notifications times. Success will be determined if each test case meets the defined pass criteria.

5. **Project Task Plan**

5.1 Task Breakdown and Assignment

Task	Duration	Assigned To	Description
Authority Validation Testing	1 day	TBD	Test authority level requirements for ZRA
Location Accuracy Testing	1 day	TBD	Validate proximity calculations with the Haversine formula
Notification System Testing	1 day	TBD	Confirm notification delivery times
Prototype III Documentation	2 days	TBD	Compile test results and update project documentation
Feedback Integration	1 days	TBD	Analyze feedback and adjust target specification for Final Design

5.2 Project Schedule

Each task will start and end based on team availability. Weekly progress updates, and completion of task updates will ensure that all members are aware of any changes in task timelines.

6. **Transfer of Knowledge**

Building upon previous work, Prototype II demonstrates an enhanced understanding of **access control systems, location validation,** and **real-time notification**. The team has applied lessons from Prototype I, including refining access level validation, improving proximity calculations, and developing bandwidth restrictions, which will carry over to the final prototype. This iterative approach has helped the team deepen its skills of data, handling and error mitigation. Next steps include improving skills related to API integration.

7. **Conclusion**

Prototype II has successfully advanced our solution by achieving high accuracy in authority validation, location sensitivity, timely notifications, and device restrictions. The insights gained from testing, along with feedback will guide the next steps in Prototype III, ensuring that our system meets user expectations for reliability and responsiveness. This document outlines our strategy for ongoing development, with a clear focus on refining critical subsystems to create a robust final prototype.

Appendix

1. **WebApp HTML Script:**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Restricted Zone Alert System</title>
    <style>
        body { font-family: Arial, sans-serif; margin: 20px; }
        #alert { font-size: 1.2em; color: red; }
    </style>
</head>
<body>
    <h1>Restricted Zone Alert System</h1>
    <label for="user">Select User:</label>
    <select id="user">
        <option value="user1">Alice</option>
        <option value="user2">Bob</option>
    </select>
    <br><br>
    <button onclick="simulateLocation()">Simulate Location Check</button>
    <div id="alert"></div>

    <script>
        // Coordinates to simulate
        const testCoordinates = [
            { latitude: 45.4215, longitude: -75.6972 },
            { latitude: 45.4245, longitude: -75.6950 },
            { latitude: 45.4260, longitude: -75.7000 }
        ];

        let index = 0;
```

```
        async function simulateLocation() {
            const user = document.getElementById("user").value;
            const location = testCoordinates[index];
            index = (index + 1) % testCoordinates.length;

            const response = await
fetch('http://127.0.0.1:5000/check_zone', {
                method: 'POST',
                headers: { 'Content-Type': 'application/json' },
                body: JSON.stringify({
                    user_id: user,
                    latitude: location.latitude,
                    longitude: location.longitude
                })
            });

            const data = await response.json();
            document.getElementById("alert").innerText = data.alert;

            if (data.status === "Access Denied" || data.status === "Out of
Zone") {
                reduceBandwidth();
            }
        }

        function reduceBandwidth() {
            console.log("Bandwidth is being restricted...");
            // Implement bandwidth restriction logic (e.g., throttle
requests)
        }
    </script>
</body>
</html>
```

2. **API/Flask Setup to Run Program with Calculations**

```
from flask import Flask, request, jsonify
import geopy.distance  # For distance calculation

app = Flask(__name__)
```

```
# Define zones with authorization levels and coordinates
zones = {
    "Zone1": {"latitude": 45.4215, "longitude": -75.6972, "radius": 0.5,
"required_level": 1},
    "Zone2": {"latitude": 45.4245, "longitude": -75.6950, "radius": 0.3,
"required_level": 2},
}

# Mock user database
users = {
    "user1": {"name": "Alice", "authority_level": 1},
    "user2": {"name": "Bob", "authority_level": 2}
}

@app.route('/check_zone', methods=['POST'])
def check_zone():
    data = request.json
    user_id = data["user_id"]
    latitude = data["latitude"]
    longitude = data["longitude"]

    if user_id not in users:
        return jsonify({"error": "User not found"}), 404

    user = users[user_id]
    user_location = (latitude, longitude)

    for zone_name, zone_info in zones.items():
        zone_location = (zone_info["latitude"], zone_info["longitude"])
        distance = geopy.distance.distance(zone_location,
user_location).km

        if distance <= zone_info["radius"]:
            if user["authority_level"] >= zone_info["required_level"]:
                return jsonify({"status": "Access Granted", "zone":
zone_name, "alert": "User is in an authorized zone"})
            else:
```

```
                return jsonify({"status": "Access Denied", "zone":
zone_name, "alert": "Insufficient authority level. Restricting
bandwidth."})

    return jsonify({"status": "Out of Zone", "alert": "User is outside all
authorized zones. Bandwidth restricted."})

if __name__ == '__main__':
    app.run(debug=True)
```