SMART BIKE RFID ENABLED ENGIENE SELF START & LOCATION TRACKING

MINI PROJECT REPORT

Submitted by

NITHILAN M	**2116230701216**
JAYASUDHAN V	**2116230701131**

In partial fulfillment for the award of the degree

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE

ANNA UNIVERSITY: CHENNAI 600 025

MAY 2025

BONAFIDE CERTIFICATE

Certified that this project **"SMART BIKE RFID ENABLED ENGIENE SELF START & LOCATION TRACKING**" is the bonafide work of "**NITHILAN M (2116230701216) and JAYASUDHAN V (2116230701131)"** who carried out the project work under my supervision.

SIGNATURE
Dr.N.Duraimurugan, M.Tech., Ph.D.

Associate Professor,

Computer Science & Engineering

Rajalakshmi Engineering College
(Autonomous)

Thandalam, Chennai -602105.

Submitted for the **ANNA UNIVERSITY** practical examination Mini-Project work viva voice held on_____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

Initially we thank the Almighty for being with us through every walk of our life and showering his blessings through the endeavor to put forth this report. Our sincere thanks to our Chairman **Mr. S.MEGANATHAN, B.E, F.I.E.,** our Vice Chairman **Mr. ABHAY SHANKAR MEGANATHAN, B.E., M.S.,** and our respected Chairperson **Dr. (Mrs.) THANGAM MEGANATHAN, Ph.D.,** for providing us with the requisite infrastructure and sincere endeavoring in educating us in their premier institution.

Our sincere thanks to **Dr. S.N. MURUGESAN, M.E., Ph.D.,** our beloved Principal for his kind support and facilities provided to complete our work in time. We express our sincere thanks to **Dr. P. KUMAR, M.E., Ph.D.,** Professor and Head of the Department of Computer Science and Engineering for his guidance and encouragement throughout the project work.

We also extend our sincere and hearty thanks to our Internal Guide **Dr.N.Duraimurugan, M.Tech., Ph.D.** Associate Professor, Department of Computer Science and Engineering for his valuable guidance and motivation during the completion of this project. Our sincere thanks to our family members, friends and other staff members of information technology.

NITHILAN M **2116230701216**
JAYASUDHAN V **2116230701131**

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATION

ABBREVIATION	ACRONYM
IOT	Internet of Things
HTTP	HyperText Transfer Protocol
GPS	Global Positioning System
RFID	Radio Frequency Identification
SQL	Structured Query Language

ABSTRACT

In today's world, vehicle security and user convenience are major concerns. This project proposes a Smart Bike system that uses RFID technology to enable engine self-start, allowing only authorized users to access and operate the bike. Additionally, the system integrates GPS-based location tracking, enabling real-time monitoring of the bike's position through a mobile application. The use of IoT ensures secure communication between the bike and the user's device, offering enhanced safety and ease of use. This smart solution not only minimizes the risk of theft but also modernizes bike operations, creating a more intelligent and connected riding experience. Moreover, the system provides instant notifications to the user in case of suspicious activities, enhancing real-time security. The mobile application allows users to view historical location data, providing insights into routes taken over time. The architecture is designed to be scalable, enabling future upgrades like speed monitoring, accident detection, and geo-fencing alerts. By integrating affordable hardware components with robust IoT services, this project delivers a cost-effective and reliable smart bike solution suitable for urban and rural environments alike. Overall, the system promotes safer, smarter, and more efficient two-wheeler usage in the growing era of smart transportation.

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

In recent years, the rise in vehicle thefts and the demand for smarter transportation solutions have emphasized the need for advanced bike security systems. Traditional lock-and-key methods are increasingly vulnerable, pushing towards the adoption of modern technologies that enhance both safety and user convenience. This project introduces a Smart Bike system that leverages RFID technology for engine self-start, ensuring that only authorized users can operate the bike. It also incorporates GPS-based location tracking integrated with a mobile application, allowing owners to monitor their vehicle's real-time location.

1.2 SCOPE OF THE WORK

The project aims to develop a smart bike system that enhances vehicle security and user convenience by using RFID technology for engine self-start, allowing only authorized users to access and operate the bike. It also focuses on implementing a GPS-based location tracking system to enable real-time monitoring of the bike's movement through a mobile application. The work includes designing a seamless IoT communication framework between the bike's embedded system and the user's smartphone to ensure reliable data transfer and control. Additionally, the project targets building a user-friendly mobile app for tracking, and receiving alerts of unauthorized access attempts.

1.3 PROBLEM STATEMENT

In today’s world, the increasing cases of bike theft and unauthorized vehicle access highlight the need for more secure and smart solutions. Traditional key-based ignition systems are vulnerable to theft, and stolen vehicles often become difficult to trace due to the lack of real-time tracking mechanisms. Existing systems fail to provide users with immediate control or updates regarding their vehicle's status and location. Therefore, there is a pressing need to design a smart bike system that ensures secure engine access through RFID authentication and enables continuous location tracking via IoT-based solutions, empowering users with enhanced control, safety, and monitoring capabilities over their vehicles.

1.4 AIM AND OBJECTIVES OF THE PROJECT

The aim of this project is to design and develop a smart bike system that utilizes RFID technology for secure engine self-start and integrates IoT-based GPS tracking to provide real-time monitoring of the bike's location. The project aims to design and implement an RFID-based authentication system to prevent unauthorized bike access and integrate a GPS module for real-time location tracking. A mobile application will be developed to enable users to monitor the bike’s location and receive status notifications. The system will ensure secure IoT communication between the bike and the user's device, enhancing bike security and providing a connected user experience. The solution will also be scalable, allowing future upgrades like geo-fencing and remote engine shutdown.

CHAPTER 2

SYSTEM SPECIFICATIONS

2.1 IOT DEVICES

1. ARDUINO UNO
2. RFID Module
3. ESP32 with Wi-Fi Module
4. GPS NEO M8N Module
5. RELAY Module

2.2 SYSTEM HARDWARE SPECIFICATIONS

PROCESSOR	Intel i5 11th Gen
MEMORY SIZE	8 GB (Minimum)
HDD	40 GB (Minimum)

2.3 SOFTWARE SPECIFICATIONS

Operating System	Windows 11
Front – End	React JS & Tailwind CSS
Back – End	Supabase
Browser	Google Chrome
IDE	Visual Studio Code

CHAPTER 3
SYSTEM DESIGN

3.1 ARCHITECTURE DIAGRAM

An architecture diagram is a graphical representation of a set of concepts, that are part of an architecture, including their principles, elements and components.

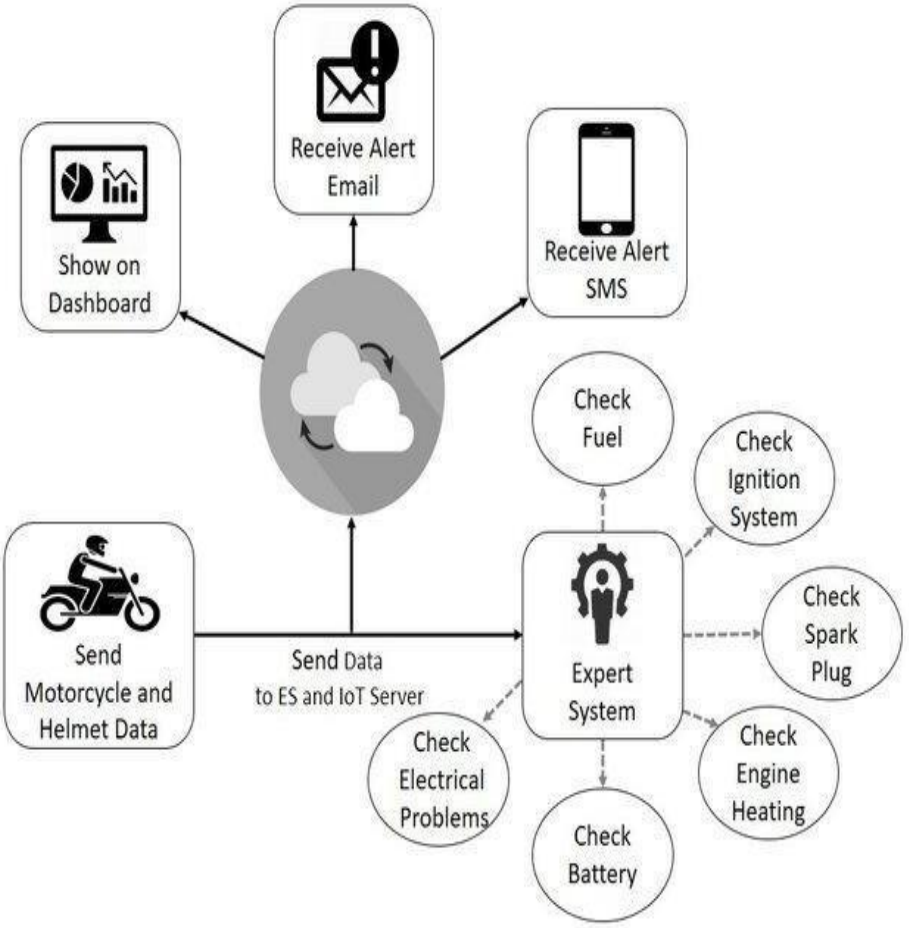

Figure 3.1 Architecture Diagram

From the above Figure 3.1, the architecture of the system is well understood.

3.2 USE CASE DIAGRAM

A use case is a list of actions or event steps typically defining the interactions between a role (known in the Unified Modelling Language as an actor) and a system to achieve a goal. The actor can be a human or other external system.

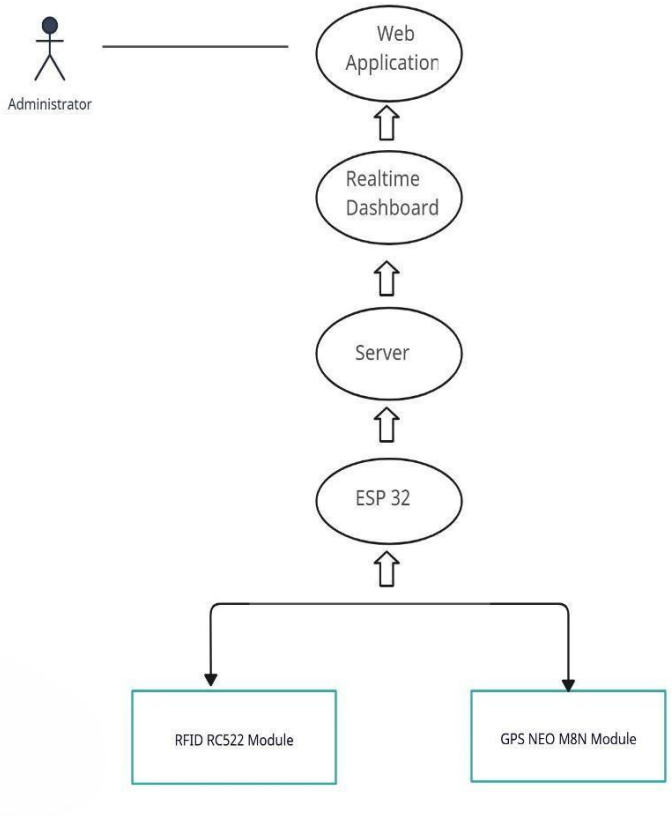

Figure 3.2 Use Case Diagram

From the above figure 3.2, the use cases are shown

3.3 ACTIVITY DIAGRAM

An activity in Unified Modelling Language (UML) is a major task that must take place in order to fulfill an operation contract. Activities can be represented inactivity diagrams. An activity can represent: The invocation of an operation. A step in a business process.

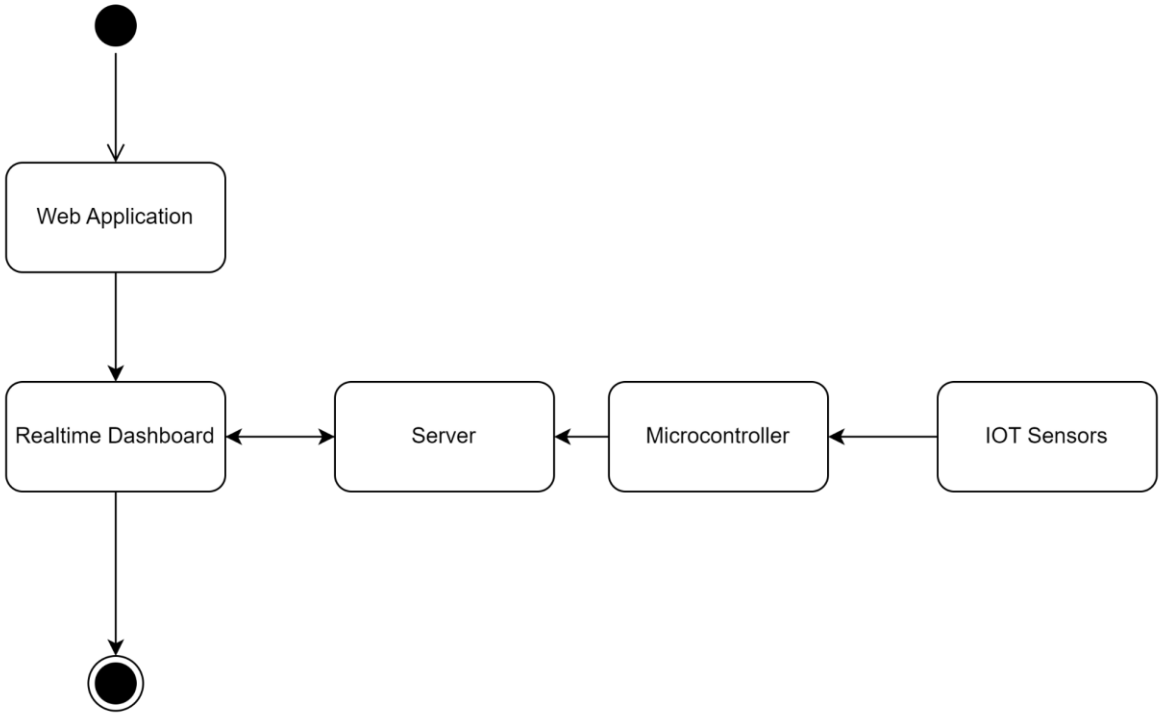

Figure 3.3 Activity Diagram

From the above figure 3.3, the activities of the system are shown

3.4 CLASS DIAGRAM

A class diagram is an illustration of the relationships and source code dependencies among classes in the Unified Modelling Language (UML). In this context, a class defines the methods and variables in an object, which is a specific entity in a program or the unit of code representing that entity.

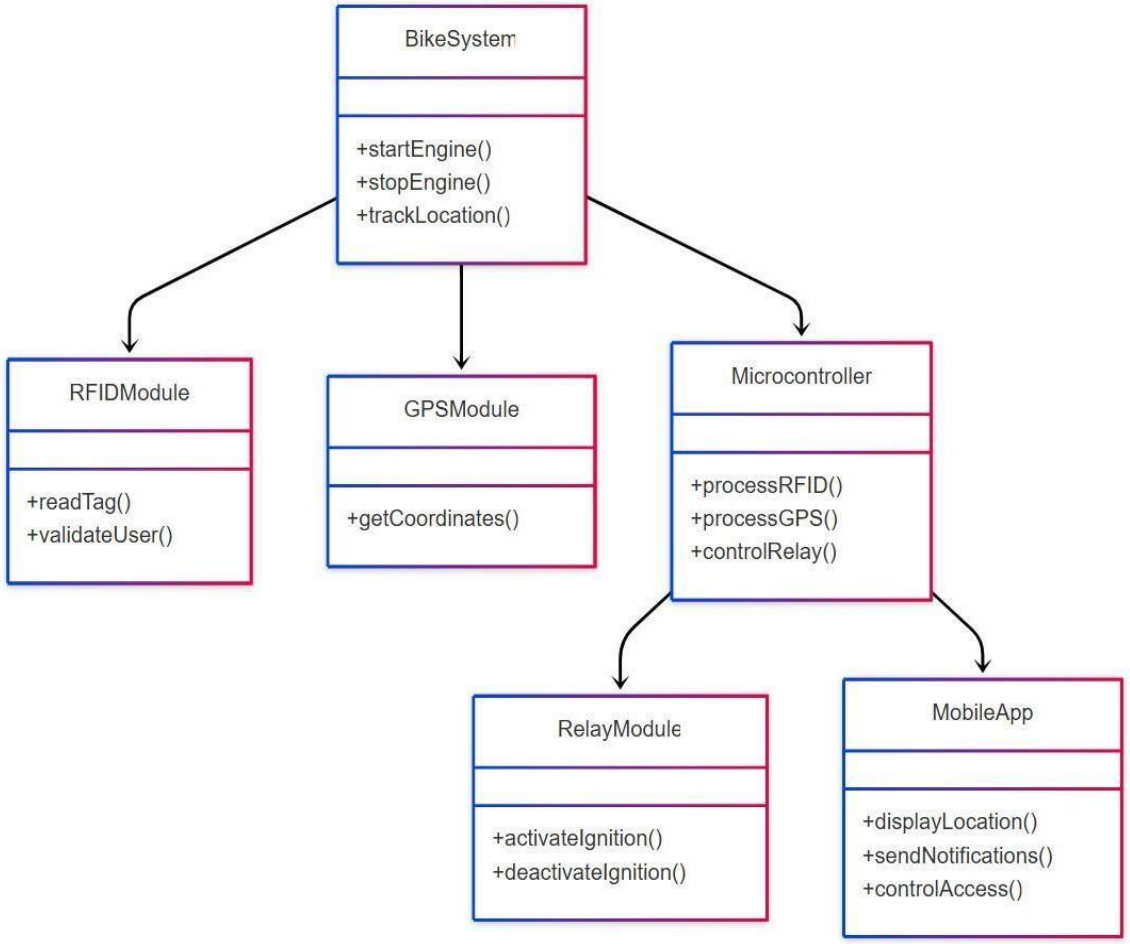

Figure 3.4 Class Diagram

The above Figure 3.4 is the class diagram for the system.

CHAPTER 4

MODULE DESCRIPTION

4.1 HARDWARE MODULE:

The system will include an RFID module for secure user authentication, ensuring only authorized users can start the bike's engine. A GPS module will track the bike’s location in real-time and provide position data. The microcontroller (e.g., Arduino) will process inputs from the RFID and GPS modules, controlling the bike's ignition via a relay module. A power supply unit will ensure proper voltage regulation for all components.

4.2 DATA COLLECTION MODULE:

The RFID module will collect user authentication data by scanning RFID tags for access control to the bike’s engine. The GPS module will continuously collect real-time location data, including latitude and longitude. The microcontroller will gather and process data from both RFID and GPS modules. This data will be transmitted to the mobile app via IoT, allowing users to track and monitor the bike remotely.

4.3 RELAY MODULE:

The relay module is used to control the bike’s ignition system by switching the engine's power on or off based on RFID authentication. When the RFID tag is authenticated, the relay allows current to flow, enabling the engine to start. The relay is controlled by the microcontroller, which processes input from the RFID module. This ensures that only authorized users can operate the bike, enhancing security.

4.4 WEB APPLICATION MODULE:

Develops a user-friendly web interface with a real-time dashboard for visualizing sensor data. The application will communicate securely with the bike through IoT, offering remote alerts and notifications. Users will also be able to control certain features, such as geo-fencing or emergency alerts, via the app.

4.5 INTEGRATION MODULE:

The integration module connects all hardware components, including the RFID, GPS, relay, and microcontroller, to ensure seamless data flow. It enables secure communication between the bike's system and the mobile/web application via IoT protocols. The module processes and integrates data from RFID authentication and GPS tracking, triggering the engine start or stopping based on user authentication. It also manages real-time alerts and updates for the user through the application.

CHAPTER 5

TABLE

5.1 GPS_DATA TABLE

ID	LATITUDE	LONGITUDE	MAPS_URL
0124a8ff-8e9d-4e51-9922-0f4e620bbc2b	13.007143	80.003665	https://maps.google.com/?q=13.007143,80.003665
07b6ec88-6470-4427-812f-65f45c354bc2	13.007181	80.00365	https://maps.google.com/?q=13.007181,80.003650
07e58ca5-56cd-4564-a377-a92529336582	13.007151	80.003662	https://maps.google.com/?q=13.007151,80.003662
0c90ecd9-898d-41c0-9543-767445194907	13.007160	80.003660	https://maps.google.com/?q=13.007160,80.003660
1d0e9498-c2d2-461d-b65a-d57ad6e829f2	13.007143	80.003665	https://maps.google.com/?q=13.007143,80.003665

CHAPTER 6

SAMPLE CODING

Arduino RFID Program

```
#include <SPI.h>
#include <MFRC522.h>
#define RST_PIN 9
#define SS_PIN 10
#define RELAY_PIN 8
MFRC522 rfid(SS_PIN, RST_PIN);
bool motorState = false;
String authorizedUIDs[] = {"71CA48C", "33D365C8"};
void setup() {
  Serial.begin(9600);
  SPI.begin();
  rfid.PCD_Init();
  pinMode(RELAY_PIN, OUTPUT);
  digitalWrite(RELAY_PIN, LOW);
  Serial.print("Place your RFID tag near the reader.");
}
void loop() {
  if (!rfid.PICC_IsNewCardPresent() || !rfid.PICC_ReadCardSerial()) {
    return;
  }
  String uid = getUID();
  Serial.println("Scanned UID: " + uid);
  if (isAuthorizedTag(uid)) {
  toggleMotor();
  } else {
    Serial.println("Unauthorized tag!");
  }
  rfid.PICC_HaltA();
```

```
}

void toggleMotor() {
  motorState = !motorState;
  digitalWrite(RELAY_PIN, motorState ? HIGH : LOW);
  if (motorState) {
    Serial.println("Motor turned OFF");
  } else {
    Serial.println("Motor turned ON");
  }
}
String getUID() {
  String uid = "";
  for (byte i = 0; i < rfid.uid.size; i++) {
    uid += String(rfid.uid.uidByte[i], HEX);
  }
  uid.toUpperCase();
  return uid;
}
bool isAuthorizedTag(String uid) {
  for (String authorizedUID : authorizedUIDs) {
    if (uid == authorizedUID) {
      return true;
    }
  }
  return false;
}
```

ESP32 GPS Program

```
#include <WiFi.h>
#include <HTTPClient.h>
#include <TinyGPSPlus.h>
#include <HardwareSerial.h>

const char* ssid = "OnePlus 11R 5G";
const char* password = "i8r23pg3";
const char* supabase_url =
"https://oajubsmkcazrrflpzauf.supabase.co/rest/v1/gps_data";


const char* supabase_api_key =
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIs
InJlZiI6Im9hanVic21rY2F6cnJmbHB6YXVmIiwicm9sZSI6ImFub24iLCJp
YXQiOjE3NDQ0NzU3NjAsImV4cCI6MjA2MDA1MTc2MH0.G3LMSrw
KEB1rO7hGVv60iAVvt_ptcjAv26ecN-LwXk0";

HardwareSerial gpsSerial(1);
TinyGPSPlus gps;

void setup() {
  Serial.begin(9600);
  gpsSerial.begin(9600, SERIAL_8N1, 16, 17);
  Serial.print("Connecting to WiFi");
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
```

```
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nWiFi connected!");
}
void loop() {
  while (gpsSerial.available() > 0) {
    gps.encode(gpsSerial.read());
  }
  if (gps.location.isUpdated()) {
    double latitude = gps.location.lat();
    double longitude = gps.location.lng();
    String maps_url = "https://maps.google.com/?q=" + String(latitude, 6) +
"," + String(longitude, 6);
    Serial.println("Sending GPS data to Supabase...");
    Serial.println("Lat: " + String(latitude, 6));
    Serial.println("Lng: " + String(longitude, 6));
    Serial.println("URL: " + maps_url);
    if (WiFi.status() == WL_CONNECTED) {
      HTTPClient http;
      http.begin(supabase_url);
      http.addHeader("Content-Type", "application/json");
      http.addHeader("apikey", supabase_api_key);
      http.addHeader("Authorization", "Bearer " + String(supabase_api_key));
      String payload = "{";
      payload += "\"latitude\": " + String(latitude, 6) + ",";
      payload += "\"longitude\": " + String(longitude, 6) + ",";
      payload += "\"maps_url\": \"" + maps_url + "\"";
      payload += "}";
```

```
      int httpResponseCode = http.POST(payload);
      Serial.print("HTTP Response code: ");
      Serial.println(httpResponseCode);
      http.end();
    } else {
      Serial.println("WiFi not connected");
    }
    delay(10000);
  }
}
```

Web Application for GPS Tracking Dashboard

1. LiveTracking.tsx

```
import React, { useEffect, useState } from 'react';
import { LocationData } from '@/lib/types';
import { Button } from '@/components/ui/button';
import { Card, CardContent, CardFooter, CardHeader, CardTitle } from
'@/components/ui/card';
import { MapPin, Clock } from 'lucide-react';
import { supabase } from '@/integrations/supabase/client';
import { useToast } from '@/hooks/use-toast';
import { Separator } from '@/components/ui/separator';
import Map from './Map';

interface LiveTrackingProps {
  initialLocation?: LocationData | null;
  onClose: () => void;
}

const LiveTracking = ({ initialLocation, onClose }: LiveTrackingProps) => {
const [liveLocation, setLiveLocation] = useState<LocationData |
null>(initialLocation || null);
  const [isConnected, setIsConnected] = useState(true);
  const [lastUpdate, setLastUpdate] = useState<Date | null>(
    initialLocation ? new Date(initialLocation.inserted_at) : null
  );
  const { toast } = useToast();
```

```
useEffect(() => {
  console.log('Setting up realtime subscription');
  const channel = supabase
    .channel('gps_updates')
    .on(
      'postgres_changes',
      { event: 'INSERT', schema: 'public', table: 'gps_data' },
      (payload) => {
        console.log('New GPS data received:', payload);
        const newLocation = {
          id: payload.new.id as string,
          latitude: payload.new.latitude as number,
          longitude: payload.new.longitude as number,
          maps_url: payload.new.maps_url as string | undefined,
          inserted_at: new Date(payload.new.inserted_at).toISOString()
        } as LocationData;

        setLiveLocation(newLocation);
        setLastUpdate(new Date(newLocation.inserted_at));
        setIsConnected(true);

        toast({
          title: "Live Update",
          description: "Received new GPS position",
          duration: 3000,
        });
      }
    )
```

```
    .subscribe((status) => {
      console.log('Subscription status:', status);
    });

  return () => {
    console.log('Cleaning up subscription');
    supabase.removeChannel(channel);
  };
}, [toast]);

useEffect(() => {
  if (!lastUpdate) return;

  const interval = setInterval(() => {
    const now = new Date();
    const diffSeconds = (now.getTime() - lastUpdate.getTime()) / 1000;

    if (diffSeconds > 30) {
      setIsConnected(false);
    }
  }, 5000);

  return () => clearInterval(interval);
}, [lastUpdate]);

const getGoogleMapsLink = (lat: number, lng: number) => {
  return `https://www.google.com/maps?q=${lat},${lng}`;
};
```

```
  return (
    <div className="fixed inset-0 bg-background z-50 flex flex-col">
      <header className="bg-white shadow-sm border-b p-4 flex justify-between
items-center">
        <div className="flex items-center">
          <MapPin className="text-tracking-primary mr-2 h-5 w-5" />
          <h2 className="text-xl font-semibold">Live GPS Tracking</h2>
          <div className={`ml-3 h-2.5 w-2.5 rounded-full ${isConnected ?
'bg-green-500' : 'bg-red-500'}`}></div>
          <span className="ml-1 text-sm text-gray-500">
            {isConnected ? 'Connected' : 'Waiting for updates...'}
          </span>
        </div>
        <Button variant="ghost" size="sm" onClick={onClose}>
          Close
        </Button>
      </header>

      <div className="flex-1 flex flex-col md:flex-row">
        <div className="flex-1 relative">
          <Map
            locations={liveLocation ? [liveLocation] : []}
            selectedLocation={liveLocation}
            onSelectLocation={() => {}}
            isLoading={false}
          />
        </div>

        <div className="w-full md:w-80 bg-white border-l overflow-auto p-4">
```

```
        <h3 className="font-semibold mb-2">Latest Position</h3>

        {liveLocation ? (
          <Card className="bg-gray-50">
            <CardHeader className="pb-2">
              <CardTitle className="text-base flex items-center">
                <MapPin className="h-4 w-4 mr-1 text-tracking-primary" />
                GPS Location
              </CardTitle>
            </CardHeader>
            <CardContent className="space-y-4">
              <div className="grid grid-cols-2 gap-2 text-sm">
                <div>
                  <span className="text-gray-500">Latitude:</span>
{liveLocation.latitude.toFixed(6)}
                </div>
                <div>
                  <span className="text-gray-500">Longitude:</span>
{liveLocation.longitude.toFixed(6)}
                </div>
              </div>

              <Separator />

              <div className="flex justify-between text-xs">
                <div className="flex items-center gap-1">
                  <Clock className="h-3 w-3" />
                  <span>{new
Date(liveLocation.inserted_at).toLocaleTimeString()}</span>
```

```
              </div>
              <div>
                <span>{new
Date(liveLocation.inserted_at).toLocaleDateString()}</span>
              </div>
            </div>
          </CardContent>
          <CardFooter>
            <Button
              variant="outline"
              size="sm"
              className="w-full text-xs"
              onClick={() => {
                const mapUrl = liveLocation.maps_url ||
                  getGoogleMapsLink(liveLocation.latitude,
liveLocation.longitude);
                window.open(mapUrl, '_blank');
              }}
            >
              Open in Google Maps
            </Button>
          </CardFooter>
        </Card>
      ) : (
        <div className="text-center p-8 text-gray-500">
          <p>Waiting for GPS updates...</p>
          <p className="text-xs mt-2">No live location data received yet</p>
        </div>
      )}
```

```
        <div className="mt-4">
          <h4 className="font-medium mb-2">About Live Tracking</h4>
          <p className="text-sm text-gray-600">
            This view automatically updates when new GPS data is received.
            Each new location will be displayed on the map immediately.
          </p>
        </div>
      </div>
    </div>
  </div>
 );
};

export default LiveTracking;
```

2. LocationSidebar.tsx

```
import React from 'react';
import { LocationData } from '@/lib/types';
import { ScrollArea } from '@/components/ui/scroll-area';
import { Button } from '@/components/ui/button';
import { MapPin, Clock, Calendar, ExternalLink, AlertCircle } from
'lucide-react';
import { Alert, AlertDescription, AlertTitle } from '@/components/ui/alert';

interface LocationSidebarProps {
  locations: LocationData[];
  selectedLocation: LocationData | null;
```

```
  onSelectLocation: (location: LocationData) => void;
  isLoading: boolean;
}

const LocationSidebar = ({
  locations,
  selectedLocation,
  onSelectLocation,
  isLoading
}: LocationSidebarProps) => {

  if (isLoading) {
    return (
      <div className="w-full h-full flex items-center justify-center">
        <div className="animate-pulse flex flex-col items-center">
          <div className="h-4 bg-slate-200 rounded w-3/4 mb-2.5"></div>
          <div className="h-4 bg-slate-200 rounded w-1/2"></div>
        </div>
      </div>
    );
  }

  if (locations.length === 0) {
    return (
      <div className="w-full h-full flex flex-col items-center justify-center
p-4">
        <Alert variant="destructive" className="mb-4">
          <AlertCircle className="h-4 w-4" />
          <AlertTitle>No Data Found</AlertTitle>
```

```
          <AlertDescription>
            No GPS location data was found in the database.
          </AlertDescription>
        </Alert>
        <p className="text-sm text-gray-500 text-center mt-2">
          Make sure your device is sending data to the Supabase database.
        </p>
      </div>
    );
  }

  const sortedLocations = [...locations].sort((a, b) =>
    new Date(b.inserted_at).getTime() - new Date(a.inserted_at).getTime()
  );

  return (
    <ScrollArea className="h-full">
      <div className="p-4">
        <h3 className="text-lg font-semibold mb-4">GPS History
({sortedLocations.length})</h3>

        <div className="space-y-4">
          {sortedLocations.map((location) => {
            const isSelected = selectedLocation?.id === location.id;
            const time = new Date(location.inserted_at).toLocaleTimeString();
            const date = new Date(location.inserted_at).toLocaleDateString();
            const mapUrl = location.maps_url ||

`https://www.google.com/maps?q=${location.latitude},${location.longitude}`;
```

```
        return (
          <div
            key={location.id}
            className={`p-3 rounded-md transition-colors relative ${
              isSelected
                ? 'bg-tracking-light border-l-4 border-tracking-primary'
                : 'bg-gray-50 hover:bg-gray-100'
            }`}
            onClick={() => onSelectLocation(location)}
          >
            <div className="flex justify-between items-start">
              <div className="flex-1">
                <p className="font-medium flex items-center gap-1">
                  <MapPin className="h-4 w-4" />
                  GPS Location
                </p>
                <div className="flex items-center gap-2 text-sm text-gray-500">
                  <Clock className="h-3 w-3" /> {time}
                  <Calendar className="h-3 w-3" /> {date}
                </div>

                <div className="grid grid-cols-2 gap-2 text-xs mt-2">
                  <div>
                    <span className="text-gray-500">Latitude:</span>
{location.latitude.toFixed(6)}
                  </div>
                  <div>
                    <span className="text-gray-500">Longitude:</span>
```

```
{location.longitude.toFixed(6)}
                  </div>
                </div>

                <Button
                  variant="outline"
                  size="sm"
                  className="text-xs flex items-center gap-1 mt-2 w-full"
                  onClick={(e) => {
                    e.stopPropagation();
                    window.open(mapUrl, '_blank');
                  }}
                >
                  <ExternalLink className="h-3 w-3" /> Open in Google Maps
                </Button>
              </div>
            </div>
          </div>
        );
      })}
    </div>
  </div>
 </ScrollArea>
);
};

export default LocationSidebar;
```

CHAPTER 7

SCREEN SHOTS

1. Dashboard Page

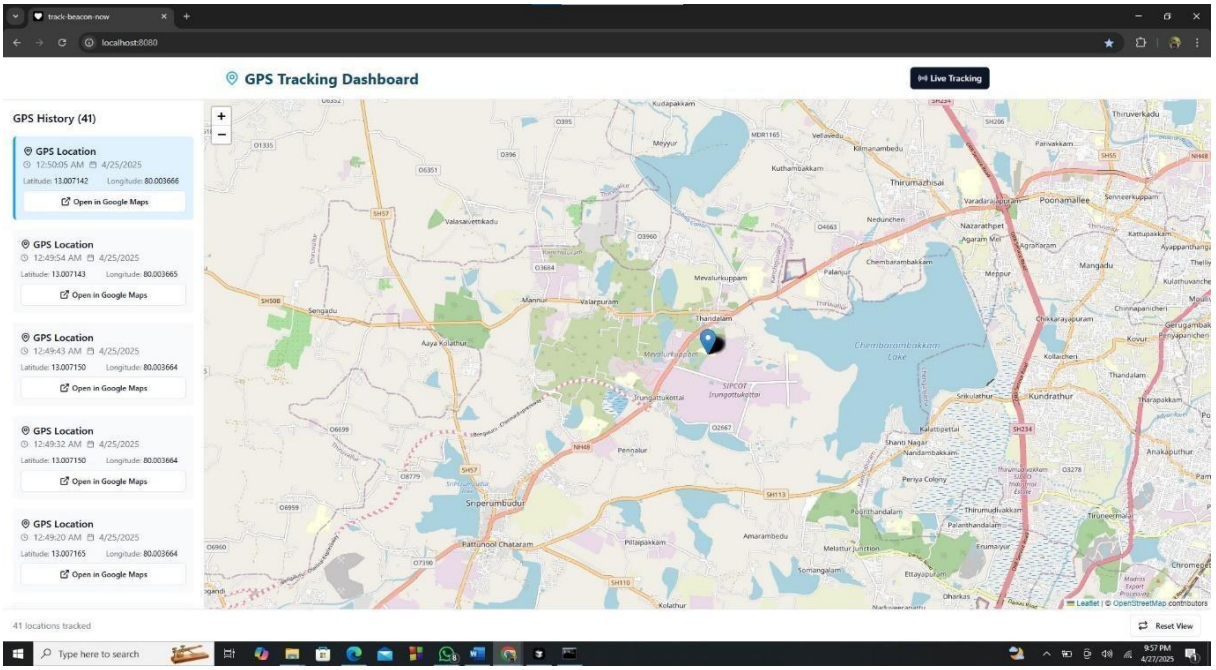

Figure 7.1 GPS History Dashboard

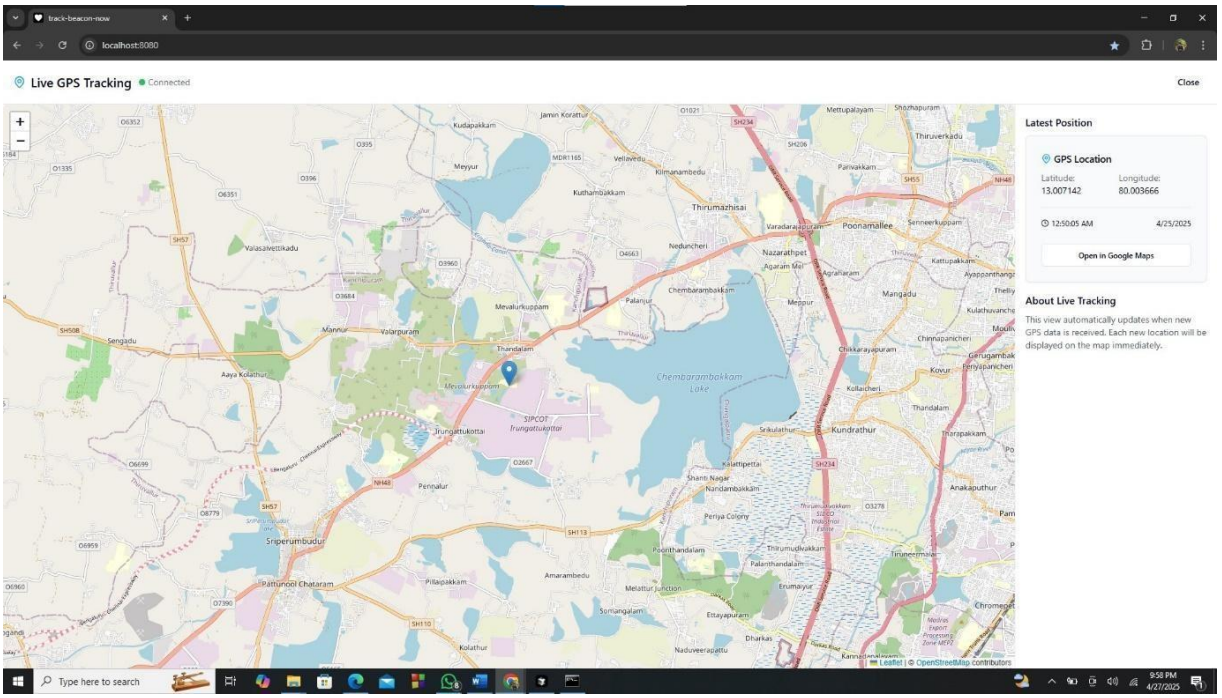

Figure 7.2 Live GPS Tracking Dashboard

2. Data Sent from ESP32 (GPS NEO M8N) to Server

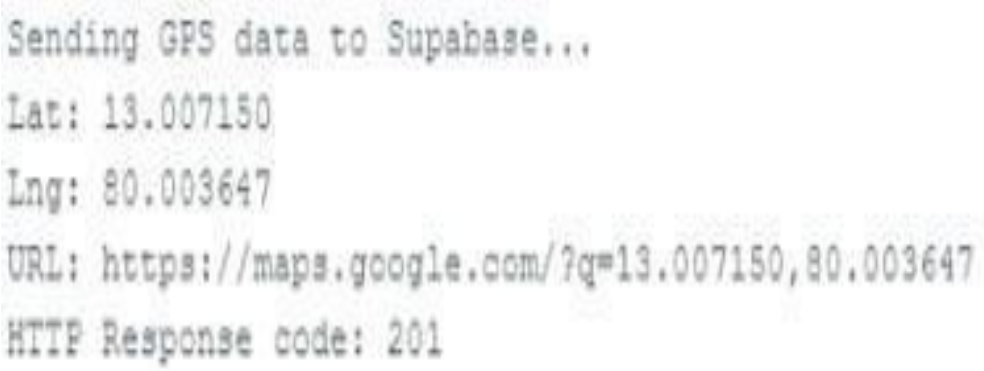

Figure 7.2 Data Received by Server from ESP32

CHAPTER 8

CONCLUSION AND FUTURE ENHANCEMENT

The smart bike system developed in this project successfully integrates RFID technology for secure engine self-start functionality and GPS-based location tracking, offering enhanced security and convenience to bike users. By utilizing IoT communication, the system enables real-time monitoring and control through a mobile application, providing users with a seamless experience. This solution minimizes the risk of theft and modernizes bike operation, aligning with the evolving demand for smarter, more connected transportation systems.

While the current system addresses the primary objectives of security and tracking, future enhancements can further improve its functionality. These enhancements include implementing features like **geo-fencing**, where users are alerted if the bike leaves a predefined area, and **remote engine shutdown** for added safety. Additionally, **speed monitoring** and **accident detection** can be integrated to provide a comprehensive safety solution. The system can also be scaled to accommodate multiple bikes and additional user profiles, offering broader applications for fleet management or smart city transportation initiatives.

REFERENCES

[1] V. R. Udugade, V. S. Pande, V. R. Naik, and S. R. Wankhade, “Bike security system using RFID and GSM technology,” International Research Journal of Engineering and Technology (IRJET), vol. 7, no. 9, pp. 563–566, Sept. 2020.

[2] P. Singh, S. N. Patel, and H. D. Padhya, “Smart bike using RFID technology,” International Journal of Scientific Research in Engineering and Management (IJSREM), vol. 6, no. 3, pp. 1–5, Mar. 2022.

[3] N. Pandey, V. Dubey, and S. Gupta, “RFID based smart key for theft protection and vehicle security,” 2015 International Conference on Computational Intelligence and Communication Networks (CICN), Jabalpur, India, 2015, pp. 911–915, doi: 10.1109/CICN.2015.187.

[4] P. R. Patil, M. B. Alase, B. A. Madival, H. S. Mahamuni, and P. K. Akulwar, “Smart bike security system,” International Journal of Creative Research Thoughts (IJCRT), vol. 6, no. 1, pp. 1256–1258, Mar. 2018.

[5] S. Sowjanya, M. V. Ramana, and B. Sruthi, “Smart bike system using RFID,” International Journal of Engineering and Techniques, vol. 4, no. 1, pp. 136–138, Jan. 2018.