INTERNSHIP PROJECT REPORT

on

“Full Stack Web Development”

At

Institute for Systems Studies & Analyses(ISSA),

Defence Research and Development Organisation (DRDO),

Ministry of Defence, Government of India.

SUBMITTED BY

- Raunak Sharma

(B.Tech - Computer Science and Engineering)

R.D. Engineering College, Ghaziabad (AKTU)

UNDER THE GUIDANCE OF

- Mr. Ichchha Shankar Sharma, Scientist ”F”,

Institute for Systems Studies & Analyses (ISSA), Defence Research and Development Organisation (DRDO)

CERTIFICATE

This is to certify that the Winter Internship/Training Report titled "Full Stack Web Development," conducted from February 28, 2025, to April 28, 2025, and submitted by Raunak Sharma as part of the requirements for the B.Tech degree in the Department of Computer Science Engineering at R.D. Engineering College (AKTU), is an original record of the candidate's work carried out under my supervision.

The contents of this report are entirely original and have not been submitted for the award of any other degree. I hereby declare that, to the best of my knowledge and belief, this submission represents the candidate's own work and does not contain material previously published or authored by another individual. Furthermore, it does not include any content that has been substantially accepted for any other degree or diploma from a university or other institute of higher learning, except where appropriate acknowledgments are provided in the text.

____

Mr. Ichchha Shankar Sharma, Scientist “F”,

ISSA,

DRDO

DECLARATION

I hereby declare that this submission is my own work and, to the best of my knowledge and belief, it contains no material that has been previously published or authored by another individual. Furthermore, it does not include any content that has been substantially accepted for the award of any degree or diploma from any university or other institution of higher learning, except where due acknowledgment has been appropriately made within the text.

Signature

Raunak Sharma (+91 7042563704)

ACKNOWLEDGEMENT

I would like to extend my heartfelt gratitude to Dr. Sujata Dash, For. Director of the Institute for Systems Studies & Analyses (ISSA) Laboratory, Defence Research and Development Organisation (DRDO), for granting me the invaluable opportunity to undergo training at this esteemed organization. This experience allowed me to gain exposure to the Research and Development environment and become familiar with Simulation technologies.

I am deeply grateful to Sh. Ichchha Shankar Sharma, Scientist 'F,' for his insightful guidance and support throughout the training. My sincere thanks also go to Smt. Suchitra Choudhary for her invaluable assistance in preparing the report and contributing to the lab work.

Lastly, I wish to express my appreciation to all members of the ISSA group for their constant encouragement, unwavering support, and dedication, which made my training experience truly enriching.

Raunak Sharma,	**Sh. Ichchha Shankar Sharma**
Computer Science Engineering,	(Mentor)
3rd Year	

ABOUT THE ORGANIZATION

Defence Research & Development Organisation (DRDO)

The Defence Research & Development Organisation (DRDO) operates under the Department of Defence Research and Development, part of the Ministry of Defence. DRDO is committed to enhancing self-reliance in defense systems by designing and developing world-class weapon systems and equipment. These advancements align with the expressed needs and qualitative requirements of the three services: The Army, Navy, and Air Force.

DRDO was established in 1958 through the merger of:

The Technical Development Establishment **(TDES)** of the Indian Army, The Directorate of Technical Development and Production (**DTOP),** and The Defence Science Organisation **(DSO).**

The Defense Research and Development Organisation (DRDO) is a network of more than 50 laboratories, dedicated to advancing defense technologies across various domains. These include aeronautics, armaments, electronics, combat vehicles, engineering systems, instrumentation, missiles, advanced computing and simulation, special materials, naval systems, life sciences, training, information systems, and agriculture. The organization is involved in several major projects, focusing on the development of missiles, armaments, light combat aircraft, radars, electronic warfare systems, and more. DRDO has already achieved significant milestones in many of these technologies. As an agency of the Republic of India, DRDO is responsible for the military's research and development, with its headquarters located in New Delhi. In addition to meeting the military's cutting-edge technology requirements, DRDO's innovations provide considerable spin-off benefits to society at large, contributing to national development and strengthening India's defense capabilities.

Vision

To make India prosperous by establishing a world-class science and technology base, and to provide our Defense Services with a decisive edge by equipping them with internationally competitive systems and solutions.

Mission

- To design, develop, and lead the production of state-of-the-art sensors, weapon systems, platforms, and allied equipment for our Defense Services.
- To provide technological solutions to the Services, optimizing combat effectiveness and promoting the well-being of the troops.
- To develop infrastructure, nurture committed quality manpower, and build a strong indigenous technology base.

Institute for Systems Studies & Analyses (ISSA)

ISSA is involved in System Analysis, Modeling & Simulation for various defence applications pertaining to employment/deployment, tactics & force potential evaluation, tactical/strategic & mission planning etc. We develop war games for all the three Services.

Consequent to the reorganization of the system analysis activities within DRDO, the functions of DSE were redefined and was given the present name, i.e. Institute for Systems Studies & Analyses (ISSA) in the year 1980. In the year 1972, a small group named as Aeronautical Systems Analysis Group (ASAG) was created with an objective to carry out aeronautical systems studies and analyses.

This group was functioning from National Aeronautics Laboratory, Bangalore as a detachment of the Directorate of Aeronautics. In the year 1974, ASAG was converted into a self-accounting full-fledged unit named as Centre for Aeronautical Systems Studies & Analyses (CASSA) and was shifted to the premises of Aeronautical Development Establishment, Bangalore.

From 1974 to 2003 ,CASSA contributed significantly to a series of systems analysis studies attributed to DRDO HQrs and the Indian Air Force. It contributed into several design and development activities and policy level issues with its objective analyses.

In the year 2003, CASSA was merged with ISSA with an objective to synergies systems analysis activities and wargaming development processes under integrated combat environment. With this, ISSA has emerged as a nodal systems analyses lab and in 2013 ISSA is placed under SAM Cluster with the mandate in the field of Training & Planning Wargames, Integrated Air Defence & EW, Combat Modeling, Simulation System Analysis and computerised Wargame.

Area Of Work

- Modeling, Simulation, Systems Analysis for Defence Application
- Systems Evaluation Studies
- Combat Model Development
- Computer Science and Applications for Defence Modelling & Simulation Domain
- Operations Research and Decision Support Techniques.

Vision

- Transform ISSA into centre of excellence in system analysis, modelling & simulation of defence systems to meet the challenges of the present and future requirements of the armed forces.

Mission

- Conduct system study and develop high quality integrated software for system analysis & decision support in application areas of Sensors & Weapons, Electronic Combat, Land & Naval Combat, Air-to-Air Combat and Air Defence for effective use by DRDO and Services for Design, Mission Planning, Tactics development and Training.

INDEX

Radar Visualization System

## 1. Introduction

The Radar Visualization System is a sophisticated, full-stack web application designed to provide real-time visualization and analysis of radar coverage patterns with terrain-aware capabilities. The system offers a modern platform for managing radar installations while considering terrain elevation data, enabling accurate coverage prediction and analysis for defense and research applications.

The system is divided into three major modules:

### Radar Management Module

- Users can add, edit, and manage radar installations
- Configure radar parameters including position, range, and height
- Real-time updates of radar coverage visualization

### Terrain Analysis Module

- Integration with digital elevation models
- Terrain-aware radar coverage calculations
- Line-of-sight analysis considering terrain interference

### Coverage Visualization Module

- Real-time visualization of radar coverage patterns

- Interactive map interface for coverage analysis
- WebSocket-based live updates

The application implements a modern architecture using:

- Responsive UI built with Angular
- Robust backend using Spring Boot and Java
- Terrain data processing using GeoTIFF files
- Real-time updates via WebSocket communication
- JTS (Java Topology Suite) for geometric calculations

@ Objective

The primary goal of this project is to provide accurate, real-time visualization and analysis of radar coverage patterns while considering terrain elevation. Key functionalities include:

- *Managing radar installations with configurable parameters*
- *Calculating terrain-aware radar coverage patterns*
- *Visualizing coverage areas on interactive maps*
- *Real-time updates of coverage changes*
- *Efficient processing of terrain elevation data*

@ Scope

This system is designed for:

- Defense organizations

- Research facilities

- Radar installation planning teams

- Coverage analysis specialists

The system can be extended with additional features such as:

- Advanced terrain analysis algorithms

- Multi-radar interference analysis

- Coverage optimization suggestions

- Historical coverage pattern analysis

@ Problem Statement

Prior to this system, radar coverage analysis often relied on simplified models that didn't account for terrain interference, leading to:

- Inaccurate coverage predictions

- Limited understanding of terrain impacts

- Inefficient radar placement decisions

- Lack of real-time visualization capabilities

The Radar Visualization System addresses these challenges by providing:

- Terrain-aware coverage calculations

- Real-time visualization and updates

- Interactive analysis tools

- Efficient data processing and storage

Technology Stack

## Frontend

### Angular (v15)

- Primary framework for building the interactive radar visualization interface

- Provides real-time WebSocket integration for live radar updates

- Component-based architecture for modular radar control panels and map displays

- Reactive forms for radar parameter input and configuration

### OpenLayers

- Advanced mapping library for rendering radar coverage polygons

- Supports GeoJSON and WKT format for terrain and coverage visualization

- Provides tools for distance measurement and coordinate selection

### TypeScript

- Ensures type safety in radar parameter handling and geometry calculations

- Improves code maintainability for complex radar visualization logic

- Enhanced IDE support for debugging coverage calculations

## Backend

### Spring Boot (v3.x)

- Core framework for radar coverage calculation and terrain analysis
- RESTful API endpoints for radar CRUD operations
- WebSocket support for real-time radar updates
- Integration with GeoServer for terrain data processing

### Java 17

- Powers the backend computation engine
- Utilized for complex geometric calculations and terrain analysis
- Supports modern language features for efficient radar coverage algorithms

### JTS (Java Topology Suite)

- Provides geometric operations for radar coverage calculation
- Handles spatial data types and terrain intersection computations
- Supports WKT format for geometry serialization

## Database

### **PostgreSQL (v14) with PostGIS**

- Stores radar configurations and coverage data
- PostGIS extension for spatial queries and geometric operations
- Tables include:
  - radars (configuration and metadata)

  - terrain_data (elevation information)

  - coverage_cache (computed coverage polygons)

## Geospatial Components

### GeoServer

- Serves terrain elevation data from TIFF files
- Provides WMS/WFS services for terrain visualization
- Supports on-the-fly terrain data processing

### GDAL

- Processes DEM (Digital Elevation Model) data
- Converts between different coordinate systems
- Handles terrain data formats and transformations

## Development Tools

### IDEs

- Spring Tool Suite (STS) for backend development
- Visual Studio Code with Angular extensions for frontend
- PostGIS extensions for database management

### Testing & Documentation

- JUnit for backend unit testing
- Jasmine for Angular component testing
- Postman for API testing and documentation

- Swagger/OpenAPI for API documentation

### Version Control & CI/CD

- Git for version control
- GitHub for repository hosting
- Maven for backend dependency management
- npm for frontend package management

### Monitoring & Debugging

- Spring Actuator for backend monitoring
- Chrome DevTools for frontend debugging
- PostgreSQL explain analyzer for query optimization

SYSTEM ARCHITECTURE

Overview

The Radar Visualization System follows a three-tier architecture, designed for efficient handling of radar data, terrain analysis, and real-time visualization:

1. **Presentation Layer (Frontend)**

   - Built with Angular for interactive radar visualization

   - Provides real-time map interface with radar coverage overlays

   - Handles user interactions for radar management and terrain analysis

2. **Application Layer (Backend)**

   - Powered by Spring Boot for robust radar data processing

   - Implements complex terrain-aware radar coverage calculations

   - Manages WebSocket connections for real-time updates

3. **Data Layer**

   - PostgreSQL with PostGIS extensions for spatial data management

   - Stores radar configurations, coverage areas, and terrain data

   - Optimized for geospatial queries and analysis

## 3.2 Flow of Data

[User Interface]

$\boxed{\downarrow}$ (HTTP/WebSocket)

[Angular Frontend]

$\boxed{\downarrow}$ (REST API/WebSocket)

[Spring Boot Backend]

$\boxed{\downarrow}$ (JPA/PostGIS)

[PostgreSQL Database]

## 3.3 Key Components

### Frontend Components:

- MapComponent: Interactive map visualization
- RadarComponent: Radar management interface
- TerrainAnalysisComponent: Terrain visualization
- WebSocketService: Real-time data handling

### Backend Structure:

- Controllers:
  - RadarController: Handles radar CRUD operations
  - TerrainController: Manages terrain data access

## 3.4 Security and Integration

- Secure REST endpoints with proper HTTP methods
- WebSocket security for real-time data transmission
- Terrain data integrity validation
- Separation of radar calculation logic from presentation

## 3.5 Advantages

1. **Specialized for Radar Operations**
   - Optimized for real-time radar coverage visualization
   - Efficient terrain-aware calculations
   - Scalable for multiple radar systems
2. **Enhanced Data Processing**
   - Dedicated services for radar coverage calculation
   - Efficient terrain elevation data handling
   - Real-time updates via WebSocket
3. **Maintainable Architecture**
   - Clear separation of radar-specific components
   - Independent scaling of processing modules
   - Easy integration of new radar features

Frontend Implementation

## Angular Project Structure

The Angular application follows a modular structure for efficient organization:

```
src/
├── app/
  ├── components/
  |   ├── map/
  |   |   ├── map.component.ts
  |   |   ├── map.component.html
  |   |   └── map.component.css
  |   ├── radar/
  |   |   ├── radar.component.ts
  |   |   ├── radar.component.html
  |   |   └── radar.component.css
  |   └── terrain/
  |       ├── terrain.component.ts
  |       ├── terrain.component.html
  |       └── terrain.component.css
  ├── services/
  |   ├── radar.service.ts
  |   ├── terrain.service.ts
  |   └── websocket.service.ts
  ├── models/
  |   ├── radar.model.ts
  |   └── terrain.model.ts
  ├── app.module.ts
  └── app-routing.module.ts
```

## 4.2 Major Components

### MapComponent

- Provides interactive map visualization
- Displays radar coverage overlays
- Integrates terrain elevation data

### RadarComponent

- Manages radar configuration and parameters
- Handles radar positioning and coverage settings
- Real-time radar status monitoring

### TerrainComponent

- Visualizes terrain elevation data
- Analyzes terrain interference with radar coverage
- Provides terrain profile analysis

## ***Services***

### RadarService

Handles radar-related HTTP requests:

```typescript
@Injectable({ providedIn: 'root' })
export class RadarService {
  private baseUrl = 'http://localhost:8080/api/radar';
  constructor(private http: HttpClient) {}
  getRadarConfigurations(): Observable<Radar[]> {
    return this.http.get<Radar[]>(this.baseUrl);
  }
  updateRadarConfiguration(radar: Radar): Observable<Radar> {
    return this.http.put<Radar>(`${this.baseUrl}/${radar.id}`, radar);
  }
  calculateCoverage(radarId: number): Observable<Coverage> {
    return this.http.get<Coverage>(`${this.baseUrl}/${radarId}/coverage`);
  }
}
```

### ***TerrainService***

Manages terrain data and analysis:

```typescript
@Injectable({ providedIn: 'root' })
export class TerrainService {
  private baseUrl = 'http://localhost:8080/api/terrain';
  constructor(private http: HttpClient) {}
  getElevationData(bounds: Bounds): Observable<ElevationData> {
    return this.http.post<ElevationData>(`${this.baseUrl}/elevation`, bounds);
  }
  analyzeTerrainInterference(radarId: number): Observable<Interference[]> {
    return this.http.get<Interference[]>(`${this.baseUrl}/interference/${radarId}`);
  }
}
```

### ***WebSocketService***

Handles real-time updates:

```typescript
@Injectable({ providedIn: 'root' })
export class WebSocketService {
  private socket: WebSocket;
  constructor() {
    this.socket = new WebSocket('ws://localhost:8080/ws');
  }
  subscribeToRadarUpdates(radarId: number): Observable<RadarUpdate> {
    return new Observable(observer => {
      this.socket.onmessage = (event) => {
        observer.next(JSON.parse(event.data));
      };
    });
  }
}
```

## 4.5 Features in the UI

### Real-time Visualization

- Interactive map with radar coverage overlays
- Dynamic terrain elevation visualization
- Real-time radar status updates via WebSocket

### Radar Management

- Radar configuration forms with validation
- Coverage calculation and visualization

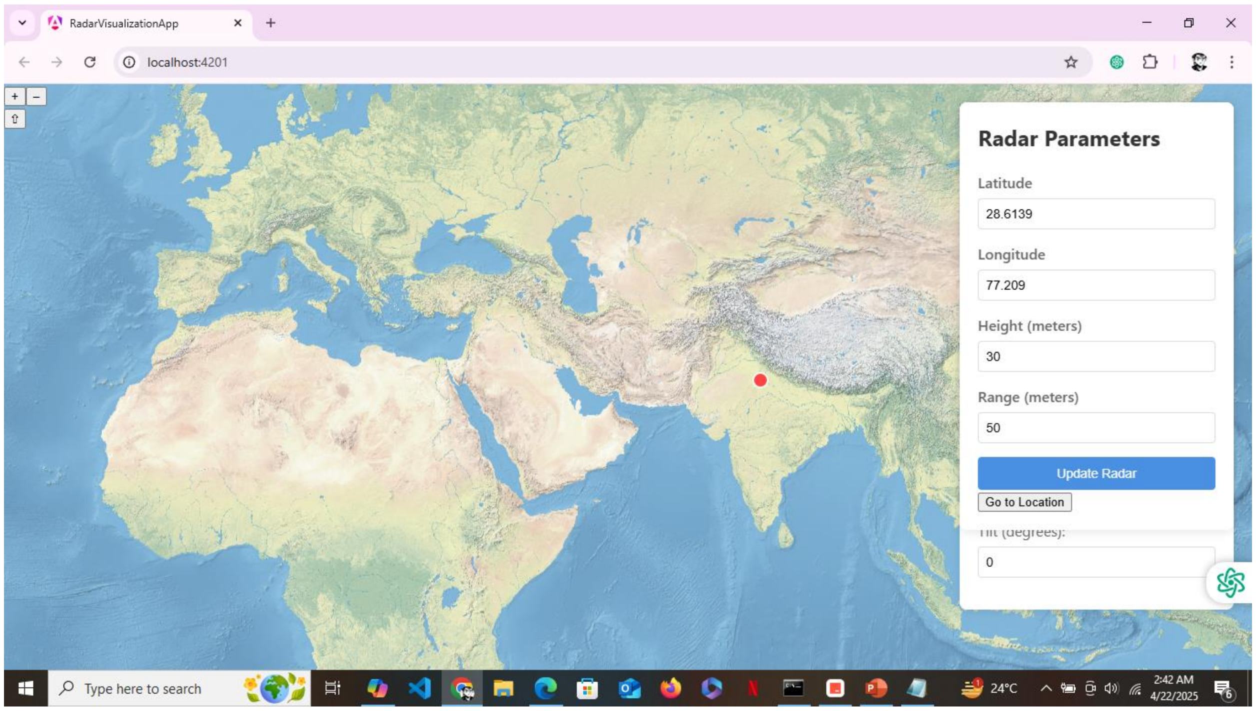
### Data Validation

- Input validation for radar parameters
- Terrain data integrity checks
- Real-time validation feedback

### Modular Design

- Separation of concerns
- Reusable UI components

Frontend UI



Radar Parameters

Latitude: 28.6139

Longitude: 77.209

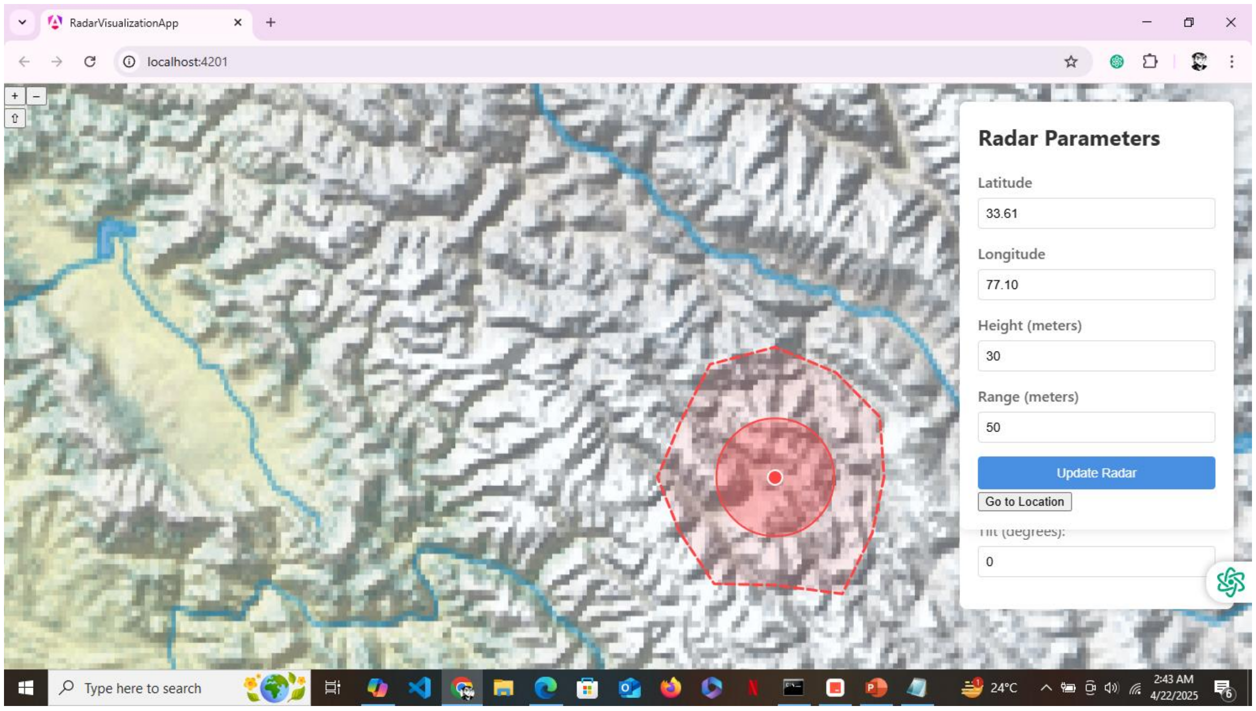
Height (meters): 30

Range (meters): 50

Update Radar

Go to Location

Tilt (degrees): 0



Radar Parameters

Latitude: 33.61

Longitude: 77.10

Height (meters): 30

Range (meters): 50

Update Radar

Go to Location

Tilt (degrees): 0

Backend Implementation

The backend system is built on Spring Boot and implements a robust architecture for radar data processing and terrain analysis.

## Core Components

### Services Layer

- **RadarCoverageService**: Calculates radar coverage considering terrain

- **TerrainService**: Processes elevation data and terrain analysis

- **WebSocketService**: Handles real-time data updates

### Data Layer

- PostgreSQL with PostGIS for spatial data storage

- Redis caching for performance optimization

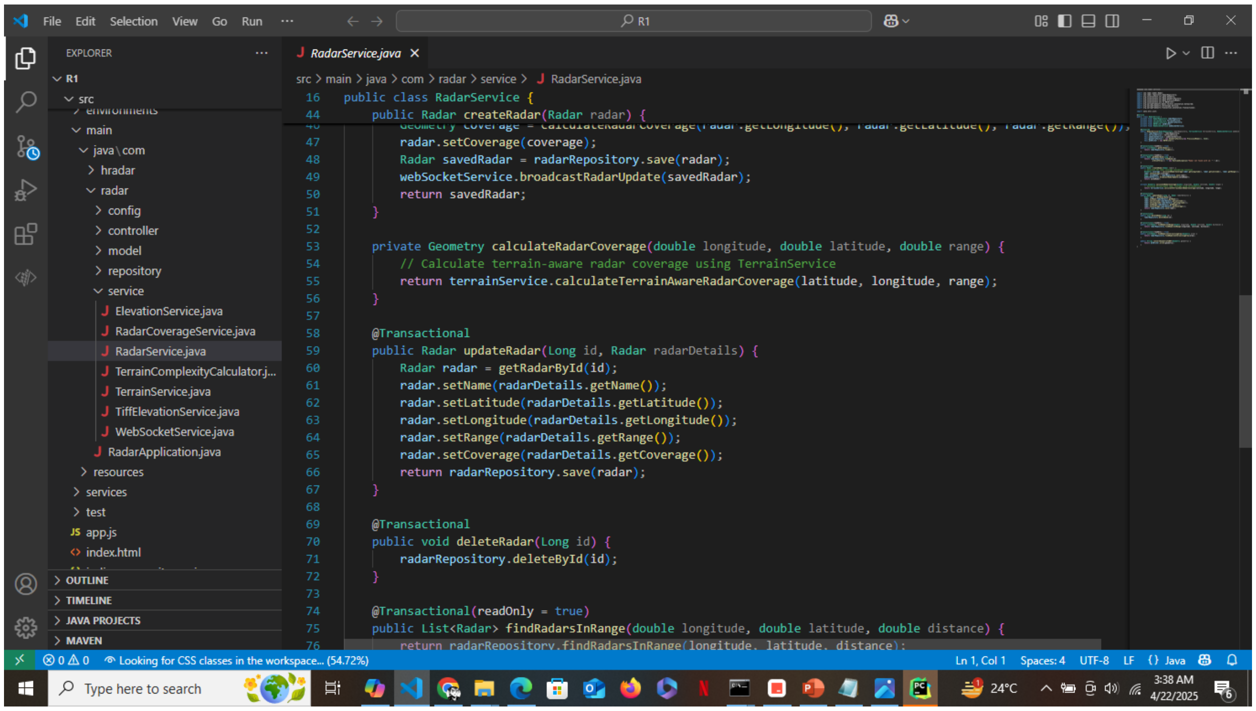
### API Layer

- RESTful endpoints for radar management

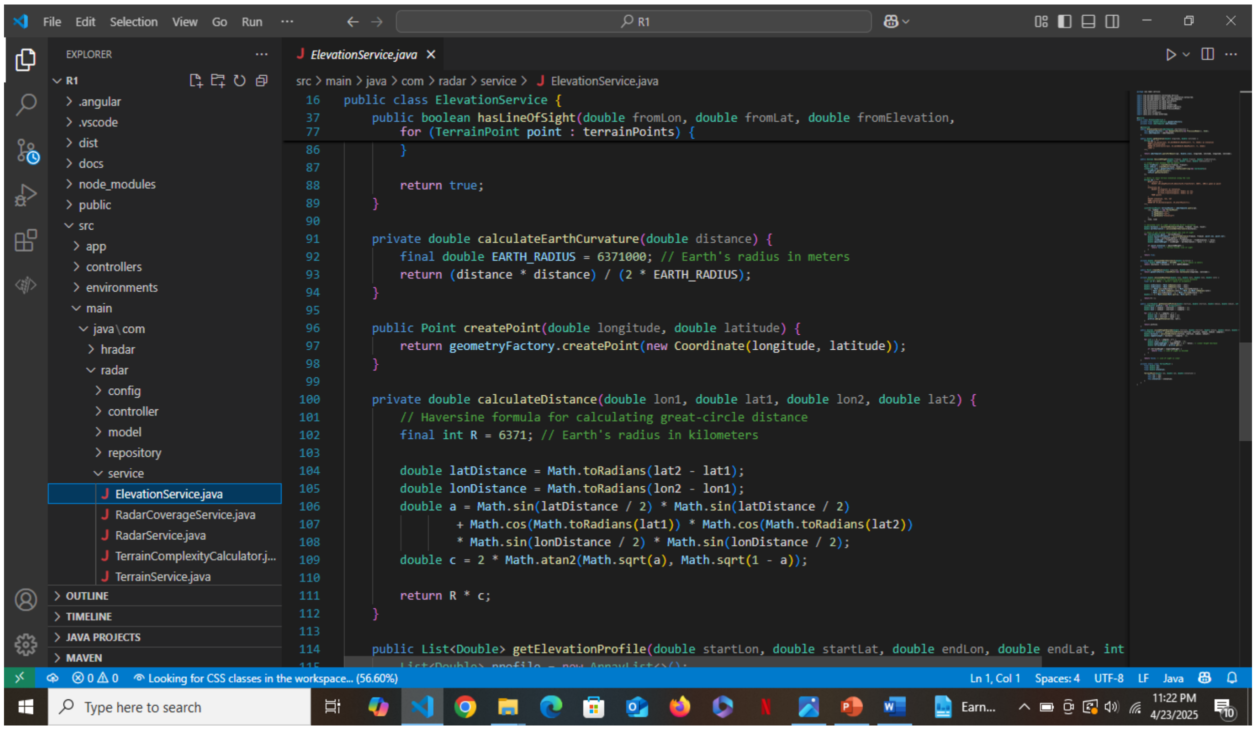
- WebSocket endpoints for real-time updates

- JWT-based authentication and authorization

Backend Code Snippet



```java
public class RadarService {
    public Radar createRadar(Radar radar) {
        Geometry coverage = calculateRadarCoverage(radar.getLongitude(), radar.getLatitude(), radar.getRange());
        radar.setCoverage(coverage);
        Radar savedRadar = radarRepository.save(radar);
        webSocketService.broadcastRadarUpdate(savedRadar);
        return savedRadar;
    }

    private Geometry calculateRadarCoverage(double longitude, double latitude, double range) {
        // Calculate terrain-aware radar coverage using TerrainService
        return terrainService.calculateTerrainAwareRadarCoverage(latitude, longitude, range);
    }

    @Transactional
    public Radar updateRadar(Long id, Radar radarDetails) {
        Radar radar = getRadarById(id);
        radar.setName(radarDetails.getName());
        radar.setLatitude(radarDetails.getLatitude());
        radar.setLongitude(radarDetails.getLongitude());
        radar.setRange(radarDetails.getRange());
        radar.setCoverage(radarDetails.getCoverage());
        return radarRepository.save(radar);
    }

    @Transactional
    public void deleteRadar(Long id) {
        radarRepository.deleteById(id);
    }

    @Transactional(readOnly = true)
    public List<Radar> findRadarsInRange(double longitude, double latitude, double distance) {
        return radarRepository.findRadarsInRange(longitude, latitude, distance);
```


ElevationService.java
src > main > java > com > radar > service > ElevationService.java
EXPLORER
R1
ElevationService.java
RadarCoverageService.java
RadarService.java
TerrainComplexityCalculator.j...
TerrainService.java
Ln 1, Col 1
Spaces: 4
UTF-8

```java
public class ElevationService {
    public boolean hasLineOfSight(double fromLon, double fromLat, double fromElevation,
        for (TerrainPoint point : terrainPoints) {
        }

        return true;
    }

    private double calculateEarthCurvature(double distance) {
        final double EARTH_RADIUS = 6371000; // Earth's radius in meters
        return (distance * distance) / (2 * EARTH_RADIUS);
    }

    public Point createPoint(double longitude, double latitude) {
        return geometryFactory.createPoint(new Coordinate(longitude, latitude));
    }

    private double calculateDistance(double lon1, double lat1, double lon2, double lat2) {
        // Haversine formula for calculating great-circle distance
        final int R = 6371; // Earth's radius in kilometers

        double latDistance = Math.toRadians(lat2 - lat1);
        double lonDistance = Math.toRadians(lon2 - lon1);
        double a = Math.sin(latDistance / 2) * Math.sin(latDistance / 2)
                + Math.cos(Math.toRadians(lat1)) * Math.cos(Math.toRadians(lat2))
                * Math.sin(lonDistance / 2) * Math.sin(lonDistance / 2);
        double c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));

        return R * c;
    }

    public List<Double> getElevationProfile(double startLon, double startLat, double endLon, double endLat, int
        List<Double> profile = new ArrayList<>();
```

Database Design

## Objective of the Database

The primary objectives of the database design are:

- Store and manage radar configuration data and coverage patterns

- Maintain terrain elevation data for coverage calculations

- Support real-time visualization and analysis

- Enable efficient spatial queries for coverage analysis

- Ensure data persistence and reliability

## Key Entities in the Database

1. **Radar** - Represents a radar installation with its configuration parameters

2. **Coverage Pattern** - Stores the calculated coverage areas for each radar

3. **Terrain Data** - Contains elevation data for terrain analysis

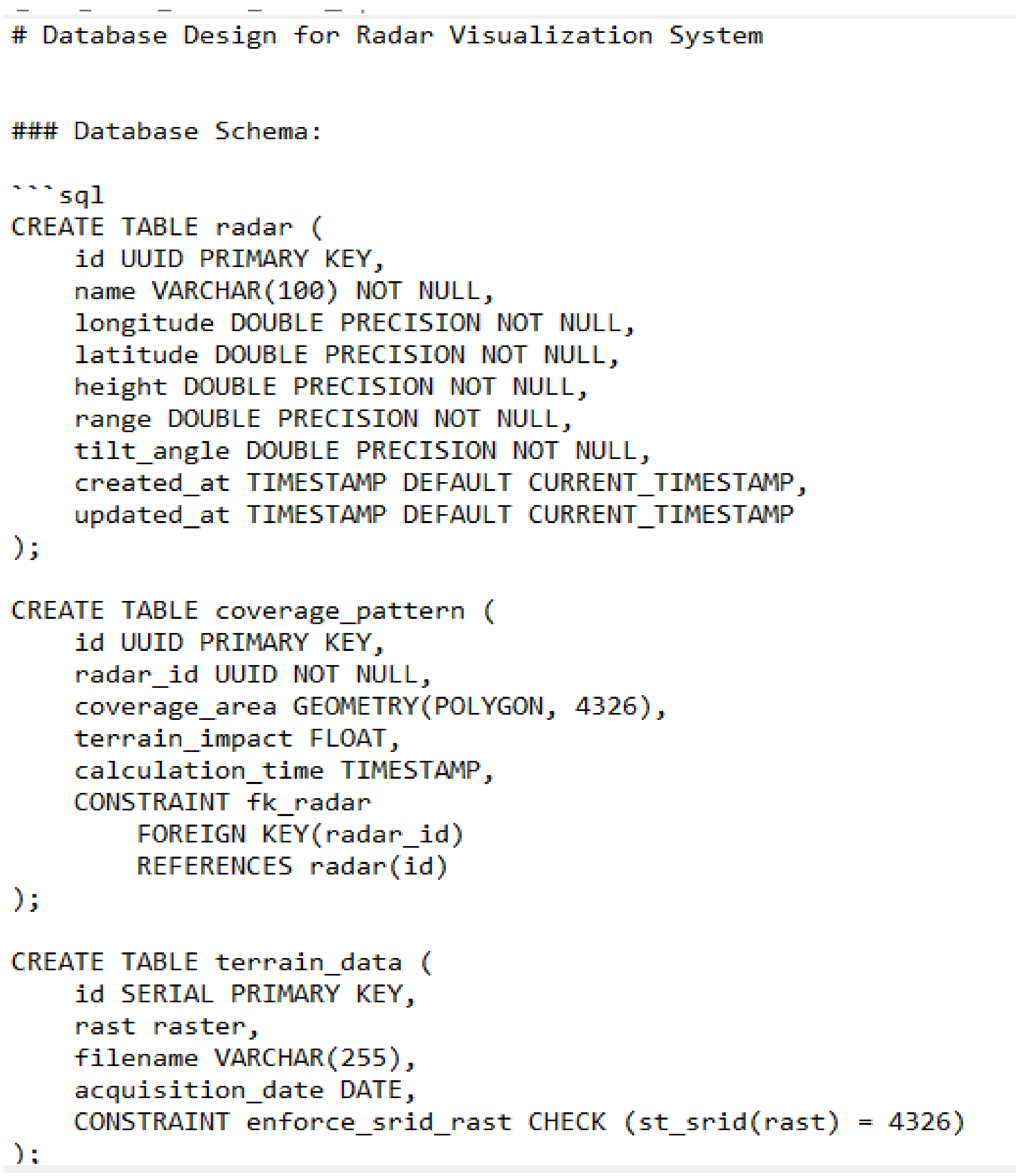
4. **Analysis Results** - Stores computed terrain complexity and visibility analysis

## Relational Design and Structure

### One-to-Many Relationships:

- One Radar can have multiple Coverage Patterns (historical data)

- One Coverage Pattern can have multiple Analysis Results



Database Design for Radar Visualization System

Database Schema:

```sql
CREATE TABLE radar (
    id UUID PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    longitude DOUBLE PRECISION NOT NULL,
    latitude DOUBLE PRECISION NOT NULL,
    height DOUBLE PRECISION NOT NULL,
    range DOUBLE PRECISION NOT NULL,
    tilt_angle DOUBLE PRECISION NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE coverage_pattern (
    id UUID PRIMARY KEY,
    radar_id UUID NOT NULL,
    coverage_area GEOMETRY(POLYGON, 4326),
    terrain_impact FLOAT,
    calculation_time TIMESTAMP,
    CONSTRAINT fk_radar
        FOREIGN KEY(radar_id)
        REFERENCES radar(id)
);

CREATE TABLE terrain_data (
    id SERIAL PRIMARY KEY,
    rast raster,
    filename VARCHAR(255),
    acquisition_date DATE,
    CONSTRAINT enforce_srid_rast CHECK (st_srid(rast) = 4326)
);
```

Why PostgreSQL?

- Native support for spatial data through PostGIS extension
- Efficient handling of raster data for terrain elevation
- Robust spatial indexing capabilities
- Support for complex geometric calculations
- High performance with large datasets
- Strong community support and documentation

6.5 How the Database Supports the Application

- When a new radar is configured through the Angular frontend, its parameters are stored in the radar table
- Coverage patterns are calculated and stored as PostGIS geometries
- Terrain data is stored as raster tiles for efficient elevation queries
- Real-time analysis results are cached for quick visualization

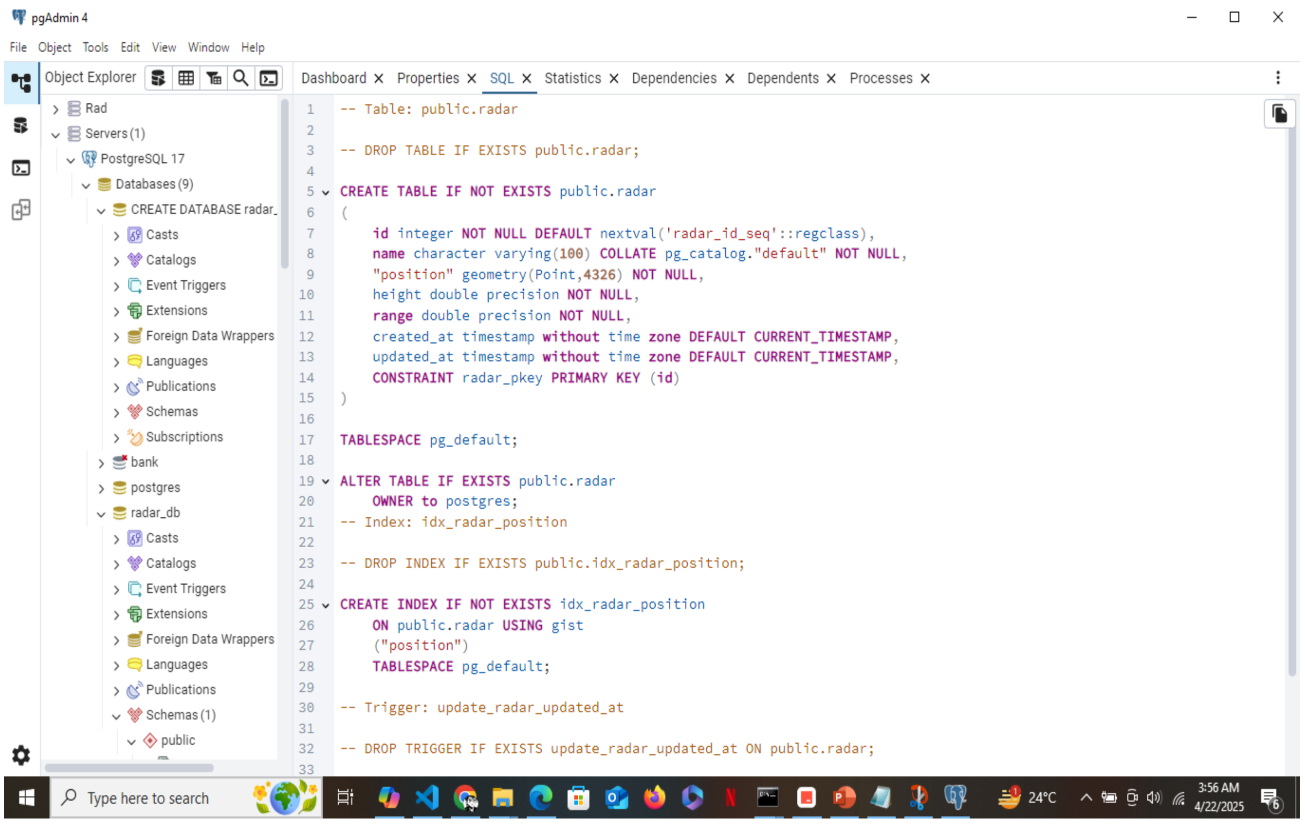
6.6 Future Scope

- Implementation of temporal analysis for coverage pattern changes
- Integration with weather data for atmospheric effects
- Support for multiple radar interference analysis
- Advanced caching mechanisms for improved performance
- Integration with external elevation data sources

6.7 Key Features

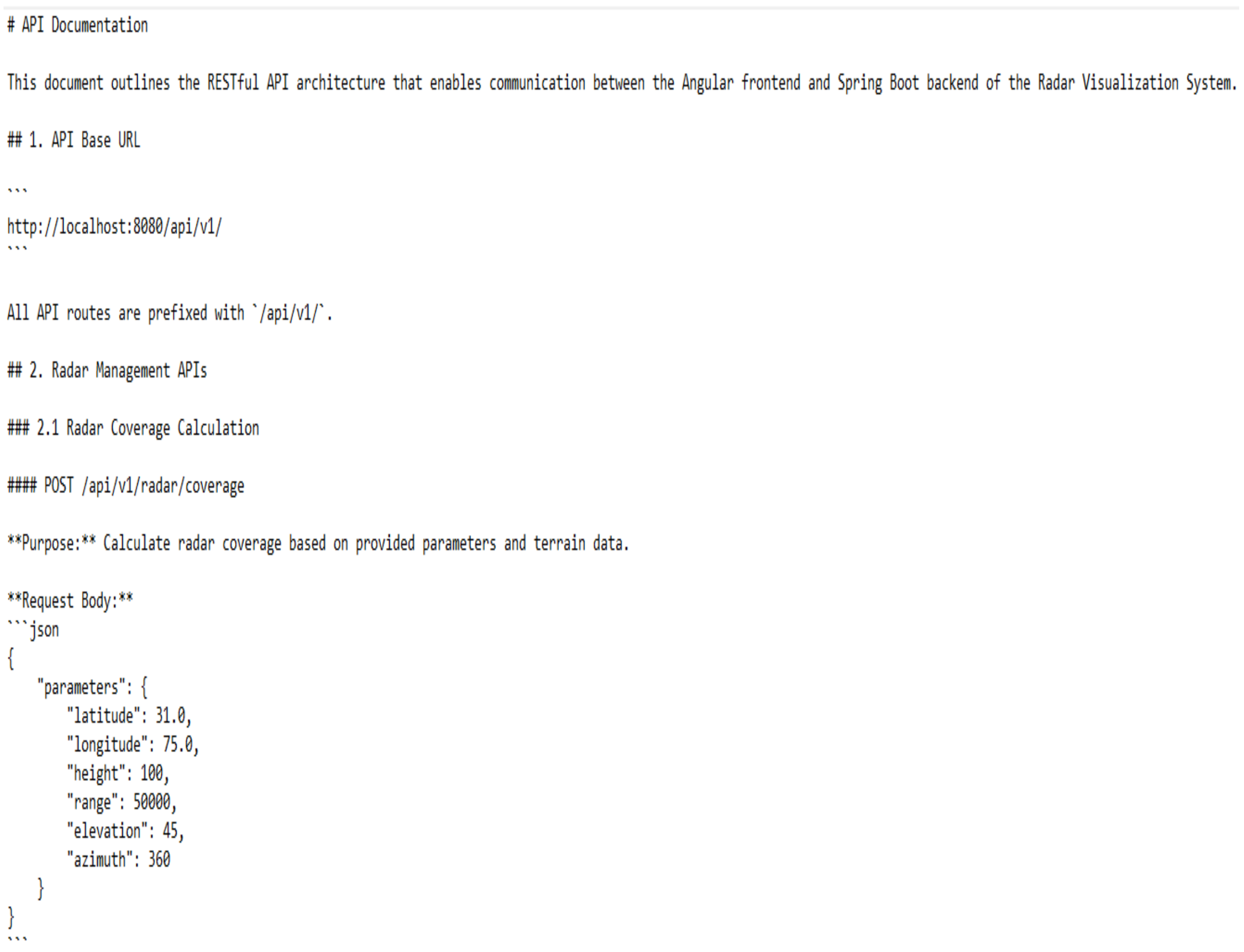
Spatial Indexing
```sql
CREATE INDEX coverage_geom_idx
    ON coverage_pattern
    USING GIST (coverage_area);

CREATE INDEX terrain_rast_idx
    ON terrain_data
    USING GIST (ST_ConvexHull(rast));
```

Database Code Snippet



```sql
-- Table: public.radar

-- DROP TABLE IF EXISTS public.radar;

CREATE TABLE IF NOT EXISTS public.radar
(
    id integer NOT NULL DEFAULT nextval('radar_id_seq'::regclass),
    name character varying(100) COLLATE pg_catalog."default" NOT NULL,
    "position" geometry(Point,4326) NOT NULL,
    height double precision NOT NULL,
    range double precision NOT NULL,
    created_at timestamp without time zone DEFAULT CURRENT_TIMESTAMP,
    updated_at timestamp without time zone DEFAULT CURRENT_TIMESTAMP,
    CONSTRAINT radar_pkey PRIMARY KEY (id)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.radar
    OWNER to postgres;
-- Index: idx_radar_position

-- DROP INDEX IF EXISTS public.idx_radar_position;

CREATE INDEX IF NOT EXISTS idx_radar_position
    ON public.radar USING gist
    ("position")
    TABLESPACE pg_default;

-- Trigger: update_radar_updated_at

-- DROP TRIGGER IF EXISTS update_radar_updated_at ON public.radar;
```

API Documentation



~~~
# API Documentation

This document outlines the RESTful API architecture that enables communication between the Angular frontend and Spring Boot backend of the Radar Visualization System.

## 1. API Base URL

```
http://localhost:8080/api/v1/
```

All API routes are prefixed with `/api/v1/`.

## 2. Radar Management APIs

### 2.1 Radar Coverage Calculation

#### POST /api/v1/radar/coverage

**Purpose:** Calculate radar coverage based on provided parameters and terrain data.

**Request Body:**
```json
{
    "parameters": {
        "latitude": 31.0,
        "longitude": 75.0,
        "height": 100,
        "range": 50000,
        "elevation": 45,
        "azimuth": 360
    }
}
```
~~~

Response:
```json
{
    "coverageData": {
        "visibilityGrid": [...],
        "coveragePercentage": 85.5,
        "affectedArea": 7850000
    },
    "timestamp": "2024-03-24T10:30:00Z"
}
```

2.2 Terrain Data Access

GET /api/v1/terrain/elevation

Purpose: Retrieve elevation data for a specific geographic area.

Query Parameters:
- `north`: Northern boundary latitude
- `south`: Southern boundary latitude
- `east`: Eastern boundary longitude
- `west`: Western boundary longitude

Response:
```json
{
    "elevationGrid": [...],
    "resolution": 30,
    "units": "meters"
}
```

2.3 WebSocket Endpoints

WS /api/v1/ws/radar-updates

Purpose: Real-time updates for radar coverage calculations.

Subscribe Message Format:
```json
{
    "action": "subscribe",
    "radarId": "radar123"
}
```

Update Message Format:
```json
{
    "type": "coverage_update",
    "data": {
        "progress": 75,
        "currentStep": "terrain_analysis"
    }
}
```

3. Response Codes

- **200 OK** - Request successful
- **201 Created** - Resource created successfully
- **400 Bad Request** - Invalid parameters or data
- **404 Not Found** - Resource not found
- **500 Server Error** - Internal server error
- **503 Service Unavailable** - Service temporarily unavailable

4. Error Response Format

```json
{
    "error": {
        "code": "INVALID_PARAMETERS",
        "message": "Invalid radar elevation angle",
        "details": "Elevation angle must be between 0 and 90 degrees"
    },
    "timestamp": "2024-03-24T10:30:00Z"
}
```

5. API Testing Tools

5.1 Development Testing
- **Postman** - Primary tool for API endpoint testing and validation
- **WebSocket Testing Client** - For testing real-time communication

5.2 Test Collections
Postman collections are available in the `/src/test/postman` directory, containing:
- Radar Coverage API Tests
- Terrain Data Access Tests
- WebSocket Communication Tests

6. Rate Limiting

- Maximum 100 requests per minute per IP
- WebSocket connections limited to 10 concurrent connections per client

7. Authentication

API authentication is handled via JWT tokens:

```
Authorization: Bearer <jwt_token>
```

Token expiration: 24 hours

8. API Versioning

API versioning is handled through the URL path:
- Current version: v1
- Legacy support: None currently
- Breaking changes will result in a new version numbe

5. System Scalability

Challenge
- Managing multiple concurrent user sessions
- Handling increasing data volumes
- Maintaining performance with growing system load

Solution
- Implemented efficient caching mechanisms
- Developed load balancing strategies
- Optimized database queries and indexing
- Created data archival and cleanup processes

Summary Table

Challenge	Solution
Real-time Visualization Performance	Implemented LOD management, WebGL acceleration, and efficient caching
Terrain Data Processing	Created multi-threaded pipeline and spatial indexing system
Frontend-Backend Integration	Developed robust WebSocket protocol and optimized API endpoints
Coverage Calculation Accuracy	Implemented adaptive algorithms and terrain interference detection
System Scalability	Added caching, load balancing, and optimized database operations

# Conclusion

# Conclusion

The Radar Visualization System represents a significant advancement in radar coverage analysis and deployment planning. Through the successful integration of cutting-edge technologies and sophisticated algorithms, the system delivers powerful capabilities that address real-world challenges in radar operations.

## Key Achievements

- Successfully merged Angular frontend with Spring Boot backend for responsive visualization

- Implemented advanced terrain-aware analysis with real-time processing capabilities

- Developed efficient data management architecture ensuring security and scalability

- Created an intuitive user interface enabling dynamic coverage analysis

## Impact and Applications

The system has demonstrated its value across multiple sectors:

- Defense organizations utilizing it for optimal radar deployment

- Research institutions conducting coverage pattern analysis

- Security agencies planning surveillance operations

- Emergency response teams assessing communication coverage

## Future Outlook

The modular architecture and robust foundation established through this project position it well for future enhancements:

- Integration with additional data sources and advanced analytics

- Implementation of machine learning capabilities

- Extension of visualization features

- Adaptation to emerging radar technologies

References

## Development Technologies

### Frontend Development

1. **Angular Framework**

  - [Angular Official Documentation](https://angular.io/docs) - Core framework documentation

  - [Angular Material Design](https://material.angular.io/) - UI component library

  - [Three.js Documentation](https://threejs.org/docs/) - 3D visualization library

### Backend Development

1. **Spring Framework**

  - [Spring Boot Documentation](https://spring.io/projects/spring-boot) - Core framework documentation

  - [Spring WebSocket Guide](https://spring.io/guides/gs/messaging-stomp-websocket/) - Real-time communication

  - [Spring Data JPA](https://spring.io/projects/spring-data-jpa) - Data persistence

### Database and Spatial Data

1. **PostgreSQL & PostGIS**

  - [PostgreSQL Documentation](https://www.postgresql.org/docs/) - Database management

  - [PostGIS Documentation](https://postgis.net/docs/) - Spatial database extensions

-
[HibernateSpatial](https://docs.jboss.org/hibernate/orm/5.4/userguide/html_single/Hibernate_User_Guide.html#spatial) - Spatial data ORM

## Development Resources

### Tutorials and Guides

1. **Web Development**

   - [MDN Web Docs](https://developer.mozilla.org/) - Web technology documentation

   - [Baeldung Spring Tutorials](https://www.baeldung.com/) - Spring framework guides

   - [JavaTpoint](https://www.javatpoint.com/) - Java and Angular tutorials

### Community Resources

1. **Developer Communities**

   - [Stack Overflow](https://stackoverflow.com/) - Programming Q&A

   - [GitHub Discussions](https://github.com/features/discussions) - Open source collaboration

   - [Dev.to](https://dev.to/) - Developer community articles

### Additional Resources

1. **Documentation and Learning**

   - [Oracle Java Documentation](https://docs.oracle.com/en/java/) - Java language reference

   - [W3Schools](https://www.w3schools.com/) - Web development tutorials

   - [FreeCodeCamp](https://www.freecodecamp.org/news/) - Programming articles and tutorials

## Research Papers and Technical Reports

1. **Radar Coverage Analysis**

   - IEEE Transactions on Aerospace and Electronic Systems

   - International Journal of Remote Sensing

   - Journal of Defense Science and Technology

2. **Terrain Visualization**

   - Computer Graphics Forum

   - IEEE Transactions on Visualization and Computer Graphics

   - International Journal of Geographical Information Science