ACCIDENT DATA ANALYSIS SYSTEM

AN INTERNSHIP PROJECT REPORT

By

THOTA DEEPIKA LAKSHMI NAGESWARI

(ROLLNO:323103211054)

Under the Esteemed

Guidance of

Dr. G.Tirupathi

Assistant Professor

IT Department

Department of information technology

GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING FOR WOMEN

[Approved by AICTE NEW DELHI, Affiliated to AU, Visakhapatnam]

Accredited by National Board of Accreditation (NBA) for B.Tech. CSE, ECE & IT – Valid from 2019-22 and 2022-25] [Accredited by National Assesment and Accreditation Council(NAAC)– Valid from 2022-27]

Kommadi , Madhurawada, Visakhapatnam–530048

2025 – 2026

GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING FOR WOMEN

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that the internship project report titled "ACCIDENT DATA ANALYSIS SYSTEM " is the bona fide work of the following B.Tech. students from the Department of information technology, Gayatri Vidya Parishad College of Engineering for Women, affiliated with Andhra University, Visakhapatnam. The project was done during II Year 2 Semester and it is being evaluated in III Year – 1 Semester of the academic year 2025 - 2026.

Thota Deepika

323103211054

Internal Guide / Mentor	**Head of the department**
Dr. G.Tirupathi	**Dr. M. Bhanu Sridhar**
Assistant Professor	**Professor**

(EXTERNAL SIGN)

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose constant guidance and encouragement crowned all efforts with success.

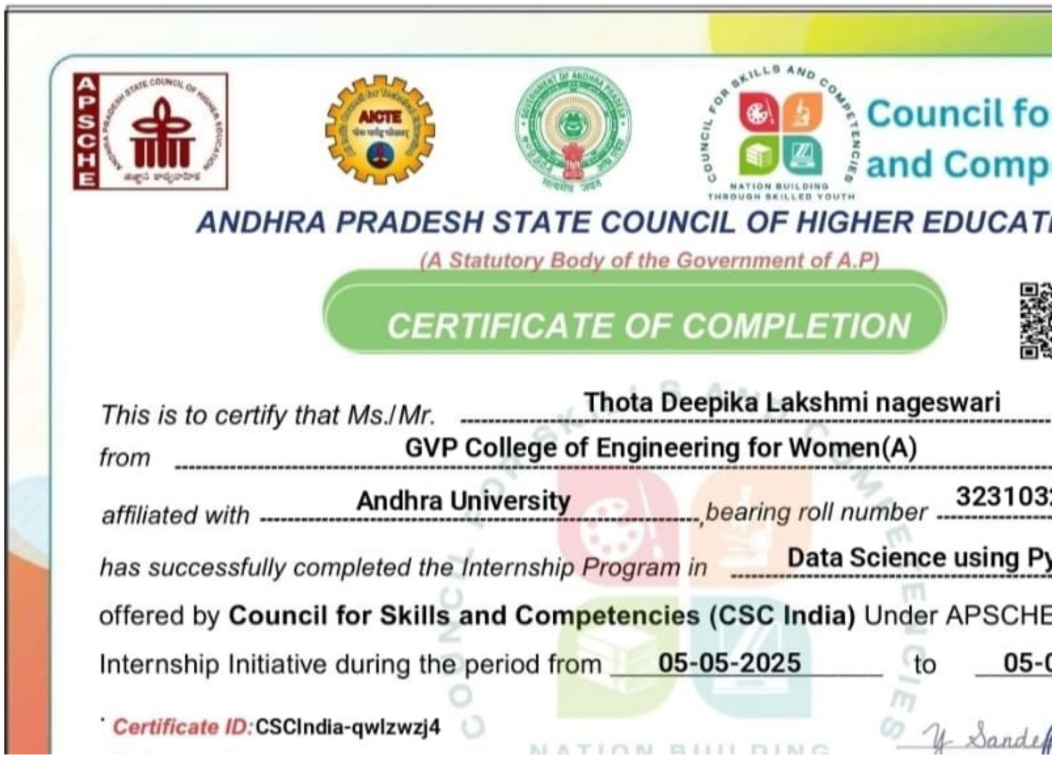
We express our deep sense of gratitude and thanks to **Dr. M. Bhanu Sridhar, Professor and Head of the Department of Information Technology and mentor**, for her guidance and valuable opinions, which greatly contributed to the development of this project. We are grateful for the lab sessions and extra hours provided to help us complete the project successfully.

We would like to take this opportunity to express our profound gratitude to **Dr. G. Sudheer, Vice Principal**, for allowing us to utilize the college resources, thereby facilitating the successful completion of our project. We also extend our sincere thanks to both the teaching and non-teaching staff of the **Department of Information Technology** for their valuable suggestions and support.

We would also like to express our heartfelt gratitude to the revered **Principal, Dr. R. K. Goswami**, for all the help, encouragement, and support extended towards the successful completion of our project.

TABLE OF CONTENTS

CERTIFICATE



Council fo
and Comp

ANDHRA PRADESH STATE COUNCIL OF HIGHER EDUCATI

(A Statutory Body of the Government of A.P)

CERTIFICATE OF COMPLETION

This is to certify that Ms./Mr. Thota Deepika Lakshmi nageswari

from GVP College of Engineering for Women(A)

affiliated with Andhra University *,bearing roll number* 323103

has successfully completed the Internship Program in Data Science using Py

offered by **Council for Skills and Competencies (CSC India)** Under APSCHE

Internship Initiative during the period from 05-05-2025 to 05-0

*Certificate ID:*CSCIndia-qwlzwzj4

ABSTRACT

The Accident Data Analysis System is an interactive data analytics application designed to analyze, visualize, and interpret road accident data to identify high-risk zones, major causes, and accident trends. The system uses Python-based data science techniques to transform raw accident records into meaningful insights through charts, tables, and statistical summaries. By analyzing parameters such as accident location, severity, time, vehicle type, and cause, the system helps users understand accident patterns and supports data-driven decision-making for road safety improvement. The platform provides interactive filtering and visualization features that allow users to explore accident data efficiently and identify critical areas requiring preventive measures.

Keywords

- Accident Data Analysis
- Python
- Data Analytics
- Road Safety
- Data Visualization
- Accident Trends
- High-Risk Zones
- CSV Dataset
- UML Diagrams

1. Introduction

Road accidents are a major public safety concern, resulting in loss of life, injuries, and economic damage every year. Large volumes of accident data are recorded by traffic departments, but raw data alone does not provide clear insights. The Accident Data Analysis System converts complex accident datasets into easy-to-understand visual representations. Using Python libraries, the system analyzes accident frequency, severity levels, and causes to help identify dangerous locations and patterns. This project demonstrates how data science techniques can be applied to improve road safety awareness and planning.

1.1 Problem Statement

Accident data is usually stored in large CSV or Excel files that are difficult to interpret manually. Identifying accident-prone areas, peak accident times, and major causes requires extensive analysis. Traditional reports lack visual clarity and do not support interactive exploration. Therefore, there is a need for a system that automatically processes accident data and presents insights in a clear and visual manner.

1.2 Objectives

The objectives of this project are:

- To analyze accident data using Python libraries
- To identify accident-prone locations and high-risk zones
- To study accident trends based on time, severity, and causes
- To visualize accident statistics using charts and graphs
- To provide an interactive and user-friendly data analysis platform

1.3 Detailed Features of the Accident Data Analysis System

1. Data Loading and Preprocessing

- The system allows loading accident data from CSV files containing records such as location, date, time, severity, and cause of accident.
- Missing values, duplicate entries, and inconsistent data formats are automatically handled during preprocessing.
- Cleaned data ensures accurate analysis and reliable visual output.

2. Accident Trend Analysis

- The system analyzes accident occurrences over different time periods such as year-wise, month-wise, and day-wise.
- It identifies peak accident times to understand when accidents are most frequent.
- Trend analysis helps in detecting increasing or decreasing accident patterns over time.

3. Severity Level Analysis

- Accidents are classified based on severity levels such as minor, serious, and fatal.
- The system calculates the percentage distribution of accidents for each severity category.
- Severity analysis helps in understanding the impact level of accidents in different regions.

4. Location-Based Risk Identification

- The system identifies accident-prone locations by counting accident frequency per area or region.
- High-risk zones are highlighted using bar charts and summary tables.
- This feature helps authorities focus on locations that require immediate safety measures.

5. Cause-Based Accident Analysis

- Accidents are analyzed based on causes such as overspeeding, drunk driving, poor road conditions, and weather.
- The system visualizes major causes contributing to accidents.
- This analysis supports preventive strategies and awareness programs.

6. Vehicle-Type Analysis

- Accident data is grouped by vehicle type such as two-wheelers, cars, buses, and trucks.
- The system identifies which vehicle categories are more frequently involved in accidents.

7. Interactive Data Visualization

- The system uses bar charts, pie charts, and line graphs to represent accident statistics.
- Visualizations make complex accident data easy to understand at a glance.
- Users can visually compare different parameters such as severity, location, and causes.

8. User-Friendly Interface

- The system provides a simple and interactive interface for exploring accident data.
- Users can easily view analysis results without technical complexity.
- Clear labeling and structured layout improve usability.

9. Scalability and Extendibility

- The system can handle large accident datasets efficiently.
- New features such as real-time accident prediction or GIS-based mapping can be added in the future.
- The modular design supports future enhancements.

2. RELATED WORK

In the field of road safety, many systems have been developed to analyze accident data and identify causes of road accidents. Understanding accident patterns helps governments and traffic authorities reduce accident rates and improve public safety. Over time, accident analysis methods have evolved from manual reporting to computerized data analysis systems.

• Traditional Accident Records

Earlier, accident data was recorded manually in registers or basic spreadsheets. Officials analyzed this data yearly, which was slow and prone to errors. This method could not efficiently handle large volumes of accident data.

• Statistical Accident Analysis

Later, statistical tools were used to identify accident trends based on time, location, and vehicle type. Though better than manual methods, these systems lacked visual clarity and user-friendly interfaces.

• Web-Based Accident Analysis Systems

Modern web-based systems allow users to upload accident datasets and analyze them using interactive dashboards. These systems reduce manual effort and provide faster insights without requiring advanced technical knowledge.

• Visual Dashboards

Recent research focuses on dashboards that use graphs and maps to show accident-prone areas and trends. Visual representation makes complex accident data easy to understand for decision-makers and the general public.

• Summary

Unlike large and costly traffic management systems, this project provides:

• Privacy: Accident data is processed locally.

• Simplicity: No need for expensive servers or expert skills.

• Clarity: Converts raw accident data into clear visual insights.

This project delivers a user-friendly “Accident Data Analysis System” for effective road safety evaluation.

S3.SYSTEM ANALYSIS

3.1 Software Requirements Specification (SRS)

Introduction –

Purpose

The purpose of the **Traffic Accident Analysis System** is to provide an interactive and visual platform for analyzing traffic accident data. The system enables users to study accident patterns, severity levels, and contributing factors such as weather conditions, driver fitness, road surfaces, lighting conditions, and peak accident hours. By transforming raw accident datasets into meaningful insights, the system supports better understanding and informed decision-making for improving road safety.

Scope

The **Traffic Accident Analysis System** analyzes accident data stored in CSV files and displays results through a web-based dashboard. It allows users to explore accident trends from different perspectives using multiple analysis modules.

The scope of the system includes:

- Analysis of overall accident statistics and severity levels
- Driver fitness and alcohol-related accident analysis
- Weather-based accident analysis
- Road surface and lighting condition analysis
- Weekly and hourly accident trend analysis
- Identification of peak accident hours

The system is developed using **Python, Streamlit, Pandas, and Plotly** and is intended for students, researchers, and traffic safety analysts.

Definitions and Abbreviations

- **SRS** – Software Requirements Specification
- **CSV** – Comma Separated Values
- **UI** – User Interface
- **Accident Severity** – Classification of accidents as Slight, Serious, or Fatal injury
- **Peak Hour** – Time period with the highest number of accidents
- **Heatmap** – A visualization showing data density using color intensity

Overall Description –

Product Perspective

The **Traffic Accident Analysis System** is a standalone web application developed using the Streamlit framework. The system directly loads accident data from CSV files without using an external database. It combines data processing, analysis, and visualization in a single platform, enabling real-time interaction with accident datasets.

Product Functions

The main functions of the system are:

- Loading and preprocessing accident data from CSV files
- Displaying overall accident statistics and key metrics
- Visualizing accident severity using pie charts
- Analyzing driver fitness and alcohol-related accidents
- Analyzing accidents based on weather conditions
- Evaluating road surface and lighting conditions
- Identifying weekly and hourly accident trends
- Detecting peak accident hours
- Providing an interactive sidebar for navigation

User Characteristics

- **Students and Researchers:** Use the system for academic analysis and project work.
- **Traffic Analysts:** Identify high-risk conditions and peak accident periods.
- **General Users:** Prefer simple visuals and easy system navigation.

Constraints

- Requires Python with Streamlit installed
- Accident data must be available in CSV format
- Internet access is needed only for library installation
- System performance depends on dataset size and available memory

Specific Requirements –

Functional Requirements

- **FR1:** The system shall load and preprocess traffic accident data from CSV files.
- **FR2:** The system shall display overall accident statistics, including total records, peak hours, and common weather conditions.
- **FR3:** The system shall visualize accident severity distribution using charts.
- **FR4:** The system shall analyze accident patterns based on driver fitness, weather conditions, road surface, and lighting conditions.

- **FR5:** The system shall generate weekly and hourly accident trend visualizations to identify risk periods.
- **FR6:** The system shall provide an interactive user interface with sidebar navigation for selecting different analysis modules.

Non-Functional Requirements

- **NFR1 (Usability):** The system shall provide a simple, interactive, and user-friendly interface that is easy to navigate.
- **NFR2 (Performance & Reliability):** The system shall efficiently process large accident datasets and handle missing or inconsistent data without failure.
- **NFR3 (Portability & Maintainability):** The system shall run on modern web browsers and be easy to maintain and enhance in the future.

3.2 System Requirements

3.2.1 Hardware Requirements –

- **Processor:** Intel i3 or higher (Intel i5/i7 recommended)
- **RAM:** Minimum 4 GB (8 GB recommended for large datasets)
- **Storage:** Minimum 1 GB free disk space
- **Display:** 1280 × 720 resolution or higher
- **Network:** Internet connection for initial setup and library installation

3.2.2 Software Requirements –

- **Operating System:** Windows 10/11, macOS, or Linux
- **Programming Language:** Python 3.9 or above
- **Framework:** Streamlit

- **Libraries:**

  - Pandas – Data loading and preprocessing
  - NumPy – Numerical operations
  - Plotly – Interactive data visualizations
  - **IDE/Editor:** VS Code, PyCharm, or Jupyter Notebook
  - **Web Browser:** Google Chrome, Microsoft Edge, or Mozilla Firefox

4. System Design

4.1 Introduction

The **Traffic Accident Analysis System** is an interactive web-based platform for analyzing traffic accident data. It follows a modular monolithic architecture where data processing and visualization are integrated within a single Streamlit application. Accident data is loaded from CSV files and processed using Python libraries to generate visual insights. The system is simple, interactive, and suitable for both technical and non-technical users.

Data Flow –

• **CSV Data Source:**
The system imports traffic accident records from CSV files, including details such as accident time, severity level, weather conditions, road surface type, lighting conditions, and driver fitness.

• **App Processing:**
The backend engine cleans and validates the accident data, extracts time-based features, and calculates key statistics such as accident frequency, severity distribution, and peak hours.

• **Analysis Views:**
o **User:** Accesses different analysis modules such as Overall Accident Analysis, Weather-Based Analysis, Driver Fitness Analysis, Road Condition Analysis, and Time-Based Analysis.

• **Output:**
Generates interactive dashboards, pie charts, bar graphs, heatmaps, sunburst charts, and trend graphs for accident analysis.

System Flow –

• **User Flow:**

1. Access the Traffic Accident Analysis dashboard.
2. Navigate to an analysis module using the sidebar menu.
3. View accident statistics and visualizations for the selected module.
4. Analyze accident trends and peak risk periods.

• **System Flow:**

1. Load accident data from CSV files.
2. Perform data cleaning and preprocessing.
3. Execute analytical computations.
4. Generate visual charts and update the dashboard.

4.2 UML Diagrams

4.2.1 Use Case Diagram

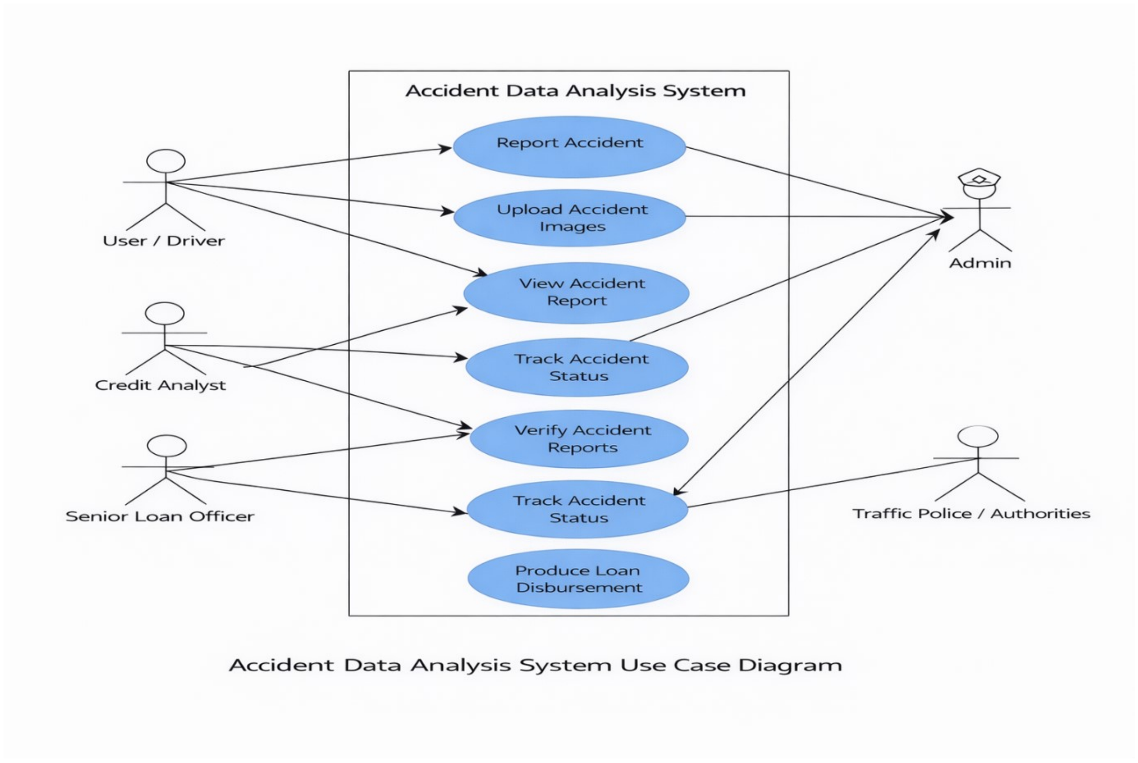
Actors

- **User / Analyst:**
  Interacts with the dashboard to analyze traffic accident data.

Use Cases

- View overall accident statistics
- Analyze accident severity
- Perform weather-based accident analysis
- Analyze driver fitness and alcohol-related accidents
- Analyze road and lighting conditions
- View weekly and hourly accident trends
- Identify peak accident hours

The Use Case Diagram represents how the user interacts with different analysis modules of the system.



4.2.2 Class Diagram

The class diagram represents the structural design of the Traffic Accident Analysis System by defining the system classes, their attributes, methods, and relationships.

Classes and their Roles –

User
Role: The analyst who interacts with the dashboard to view accident insights.
Attributes: userID, selectedModule
Methods: selectModule(), viewResults()

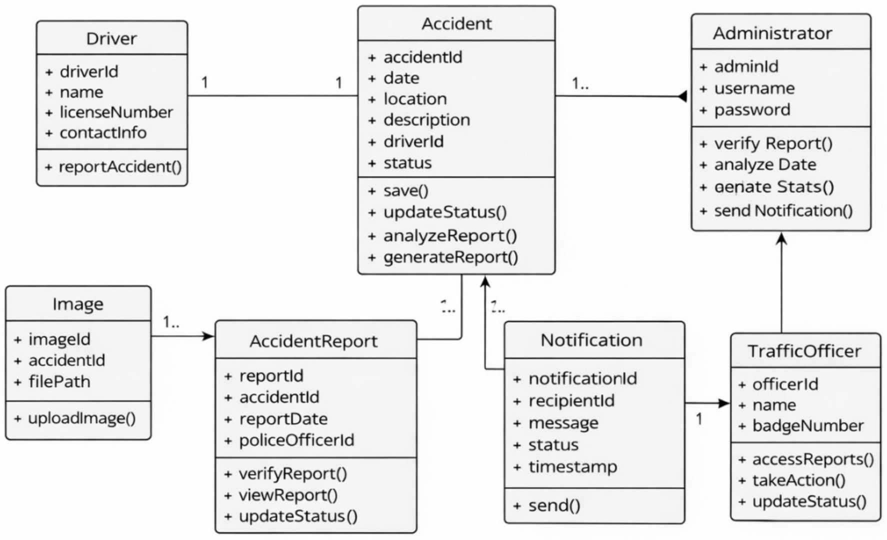
Dashboard
Role: Handles user interface, navigation, and visualization rendering.
Attributes: pageTitle, charts, filters
Methods: renderDashboard(), updateCharts()

AccidentDataset
Role: Manages the accident CSV data and preprocessing tasks.
Attributes: filePath, dataFrame
Methods: loadData(), cleanData(), filterData()

AnalysisEngine
Role: Performs analytical computations and trend analysis.
Attributes: analysisType
Methods: computeStatistics(), generateTrends()

Relationships and Multiplicities

- User to Dashboard (1 … *): A user can access multiple analysis views.
- Dashboard to AnalysisEngine (1 … 1): The dashboard uses a single analysis engine.
- AnalysisEngine to AccidentDataset (1 … 1): The analysis engine processes one dataset source.



Accident Data Analysis System Class Diagram

4.2.3 Sequence Diagram

A sequence diagram tracks the dynamic flow of interactions between system components over a timeline. It shows how the **User, Dashboard, and Accident Dataset** interact to analyze traffic accident data and generate visual insights.

User Sequence (Traffic Accident Analysis)

• **Role:**
The user analyzing traffic accident patterns, severity levels, and contributing factors.

• **Step 1:**
The User launches the application and triggers view_dashboard() to initiate the session.

• **Step 2:**
The User selects an analysis module (such as Severity Analysis, Weather Analysis, or Time-Based Analysis) using the sidebar navigation.

• **Step 3:**
The Dashboard sends the selected request to the Dataset engine, which loads and filters the accident data accordingly.

• **Step 4:**
The Dataset processes the data by cleaning records and computing required statistics.

• **Step 5:**
The Dashboard calls render_visuals() to display charts and analytical results back to the User.

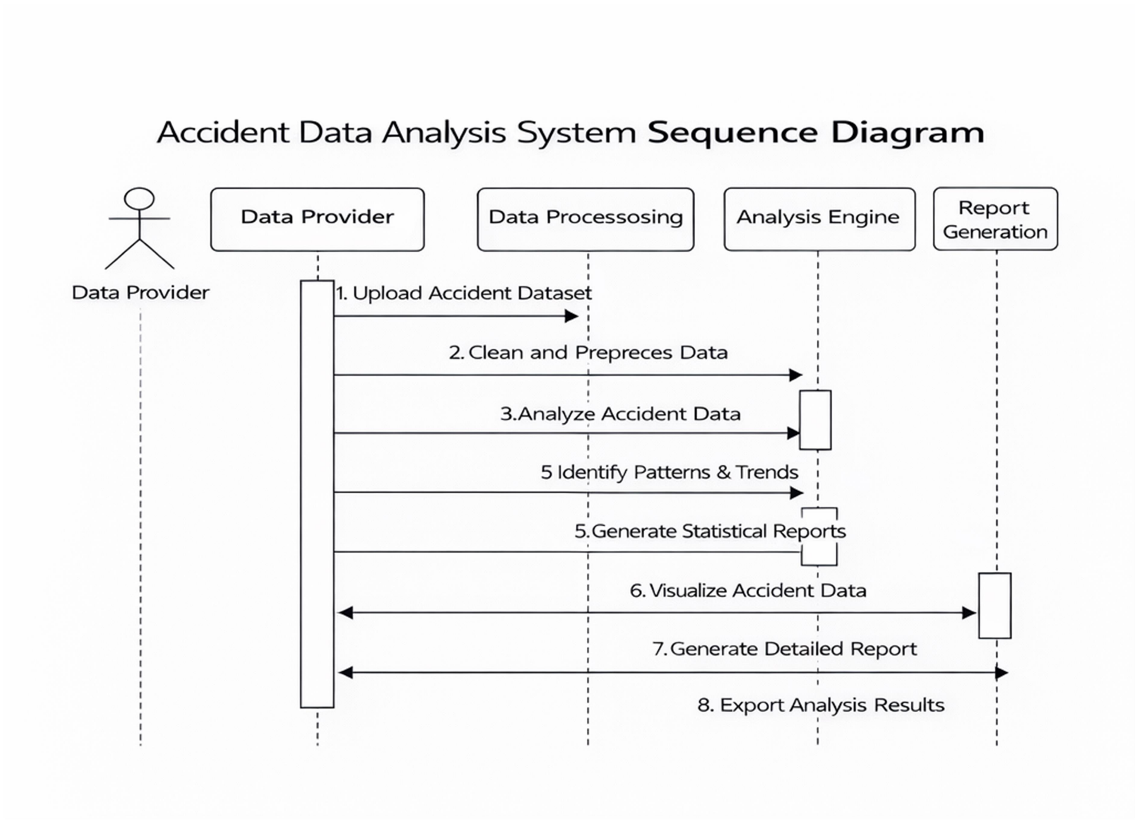
Key Components in Sequence

• **Actor:**
User / Analyst

• **App Controller:**
Dashboard (Handles navigation, user interaction, and visualization rendering)

• **Data Source:**
Accident Dataset (Loads CSV data, performs cleaning, and supplies processed data)

• **Outputs:**
Accident statistics, filtered datasets, and interactive visualizations such as pie charts, bar graphs, heatmaps, and trend charts.


Accident Data Analysis System Sequence Diagram
Data Provider
Data Provider
Data Processosing
Analysis Engine
Report Generation
1. Upload Accident Dataset
2. Clean and Prepreces Data
3.Analyze Accident Data
5 Identify Patterns & Trends
5.Generate Statistical Reports
6. Visualize Accident Data
7.Generate Detailed Report
8. Export Analysis Results

4.2.4 Activity Diagram

The activity diagram serves as a logical flowchart illustrating the sequence of actions a user performs within the **Traffic Accident Analysis System** to analyze accident data and generate insights.

Core Logic Flow

• **Start & Open App:**
The workflow begins when the user launches the Traffic Accident Analysis System.

• **View Dashboard:**
The system automatically triggers the dashboard view to present overall accident statistics and available analysis options.

• **User Choice (Decision Node):**
The user selects one of the available analysis modules using the sidebar navigation, such as:
o Overall Accident Analysis
o Driver Fitness Analysis
o Weather-Based Analysis
o Road Condition Analysis
o Weekly or Hourly Time Analysis

• **Data Processing:**
o **Load & Clean Data:** The system loads accident data from CSV files and handles missing or invalid values.
o **Filter Data:** The system filters records based on the selected analysis module.
o **Compute Statistics:** Required metrics such as severity distribution, frequency counts, and peak hours are calculated.

• **Visualization:**
The system generates interactive visualizations including pie charts, bar graphs, heatmaps, sunburst charts, and trend charts.

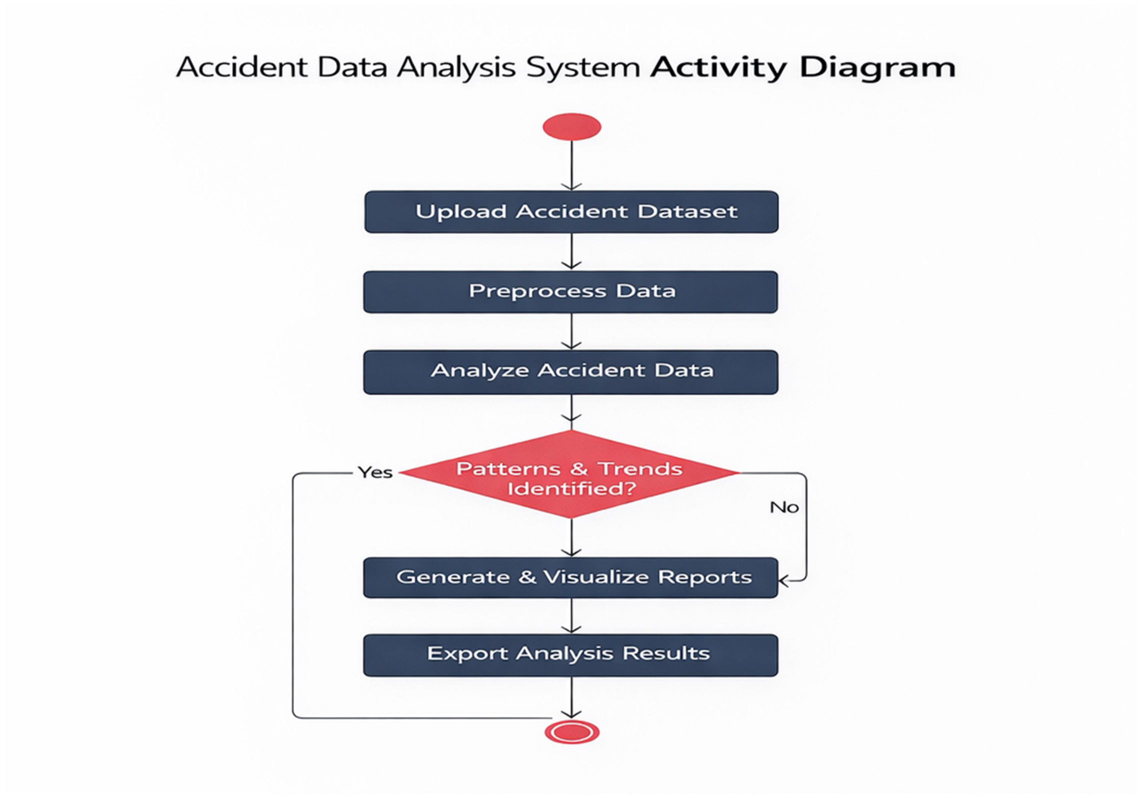
• **Output:**
The analytical results are displayed on the dashboard for user interpretation.

• **End:**
The workflow ends when the user completes the analysis or closes the application.

Role-Based Benefits of This Flow

• **For the User:**
Provides a clear and efficient path from raw accident data to meaningful visual insights and trend identification.

• **For the System:**
Ensures smooth data processing, accurate computation, and consistent visualization across all analysis modules.


Accident Data Analysis System Activity Diagram
Upload Accident Dataset
Preprocess Data
Analyze Accident Data
Patterns & Trends Identified?
Yes
No
Generate & Visualize Reports
Export Analysis Results

5. Methodology

Algorithms and Techniques Used

5.1. Introduction

The **Traffic Accident Analysis System** uses specific data processing algorithms and visualization techniques to analyze large accident datasets and identify important safety patterns. These methods enable the system to handle thousands of accident records, compute statistical insights, and present the results in an easy-to-understand visual format for users. The approach focuses on simplicity, accuracy, and real-time interaction through a web-based dashboard.

5.2. Algorithms

5.2.1 Accident Severity Distribution Calculation

• **Input:**
Accident severity data (Slight, Serious, Fatal) from the CSV dataset.

• **Process:**
Counts the number of accidents in each severity category and calculates their percentage distribution.

• **Output:**
A visual representation of accident severity using pie or donut charts.

5.2.2 Time-Based Trend Analysis

• **Input:**
Accident time and day information from the dataset.

• **Process:**
Extracts hour and day values from the accident time and computes accident frequency for each time period.

• **Output:**
Weekly and hourly trend charts, including heatmaps and area graphs, highlighting peak accident hours.

5.2.3 Weather-Based Accident Analysis

• **Input:**
Weather condition data associated with each accident record.

• **Process:**
Groups accident records by weather conditions and analyzes their frequency and severity impact.

• **Output:**
Bar charts showing accident distribution under different weather conditions.

5.2.4 Road and Lighting Condition Analysis

• **Input:**
Road surface type and lighting condition details from the dataset.

• **Process:**
Analyzes the relationship between road conditions, lighting conditions, and accident occurrence.

• **Output:**
Sunburst charts representing the combined distribution of road surface and lighting conditions.

5.2.5 Driver Fitness Analysis

• **Input:**
Driver fitness and alcohol-related data from accident records.

• **Process:**
Classifies accidents based on driver fitness conditions and evaluates their impact on accident severity.

• **Output:**
Grouped bar charts showing the influence of driver fitness on accident outcomes.

5.3. Techniques

5.3.1 Data Handling

• **CSV Parsing:**
Python Pandas is used to efficiently read and process accident data from CSV files.

• **Data Cleaning:**
Missing or inconsistent values are handled to ensure accurate analysis and visualization.

• **Filtering and Aggregation:**
Accident data is filtered and grouped based on severity, time, weather, and road conditions to generate summary statistics.

5.3.2 Visualization

• **Pie and Donut Charts:**
Used to represent accident severity distribution clearly.

• **Bar Charts:**
Used for comparing accident frequency across weather conditions and driver fitness categories.

• **Heatmaps:**
Used to visualize accident density across days of the week and hours of the day.

• **Sunburst Charts:**
Used to display hierarchical relationships between road surface and lighting conditions.

• **Trend Charts:**
Used to identify accident peak hours and time-based patterns.

5.3.3 Software Design

• **Single-User Analytical View:**
Provides a unified dashboard for analyzing accident data without role-based complexity.

• **Interactive User Interface:**
Developed using Streamlit to allow instant updates of charts based on user selections.

• **Diagram Modeling:**
The system design is planned using Use Case, Class, Sequence, and Activity diagrams before implementation.

6.Implementation of Code with Comments

app.py –

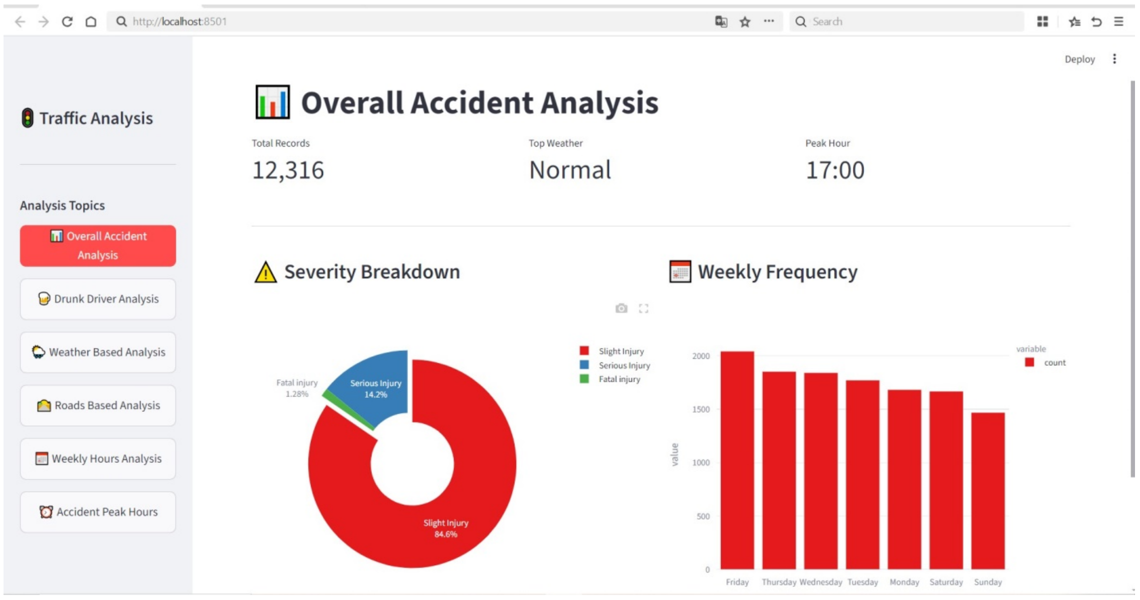
```
import streamlit as st
import pandas as pd
import plotly.express as px
# 1. Page Configuration
st.set_page_config(layout="wide", page_title="Traffic Analysis System")
# 2. FIXED: Corrected CSS Error (unsafe_allow_html)
st.markdown("""
    <style>
    .stButton>button { border-radius: 8px; height: 3.5em; font-weight: bold; font-size: 16px; }
    </style>
    """, unsafe_allow_html=True)
# 3. Sidebar Navigation
with st.sidebar:
    st.title("🚦 Traffic Analysis")
    st.markdown("---")
    if 'active_page' not in st.session_state:
        st.session_state.active_page = "Overall Accident Analysis"
    topics = {
        "Overall Accident Analysis": "📊",
        "Drunk Driver Analysis": "🥤",
        "Weather Based Analysis": "🌨□",
        "Roads Based Analysis": "🛣□",
        "Weekly Hours Analysis": "📅",
        "Accident Peak Hours": "□"
    }
    st.write("### Analysis Topics")
    for t, icon in topics.items():
        if st.button(
            f"{icon} {t}",
            use_container_width=True,
            type="primary" if st.session_state.active_page == t else "secondary"
        ):
            st.session_state.active_page = t
            st.rerun()
# 4. Data Loading & Cleaning (Fixes the "Faded" & "None" errors)
@st.cache_data
def load_data():
    df = pd.read_csv("traffic_accidents.csv")
    df['Time'] = pd.to_datetime(df['Time'])
    df['Hour'] = df['Time'].dt.hour
    # Fill missing values so the Sunburst chart isn't empty/faded
    cols = ['Road_surface_type', 'Light_conditions', 'Weather_conditions',
'Fitness_of_casuality']
    for col in cols:
        df[col] = df[col].fillna('Not Specified').replace('', 'Not Specified')
    return df
topic = st.session_state.active_page
try:
    df = load_data()
    # 5. Main Screen Logic
    st.title(f"{topics[topic]} {topic}")
    if topic == "Overall Accident Analysis":
        m1, m2, m3 = st.columns(3)
        m1.metric("Total Records", f"{len(df):,}")
        m2.metric("Top Weather", df['Weather_conditions'].mode()[0])
```

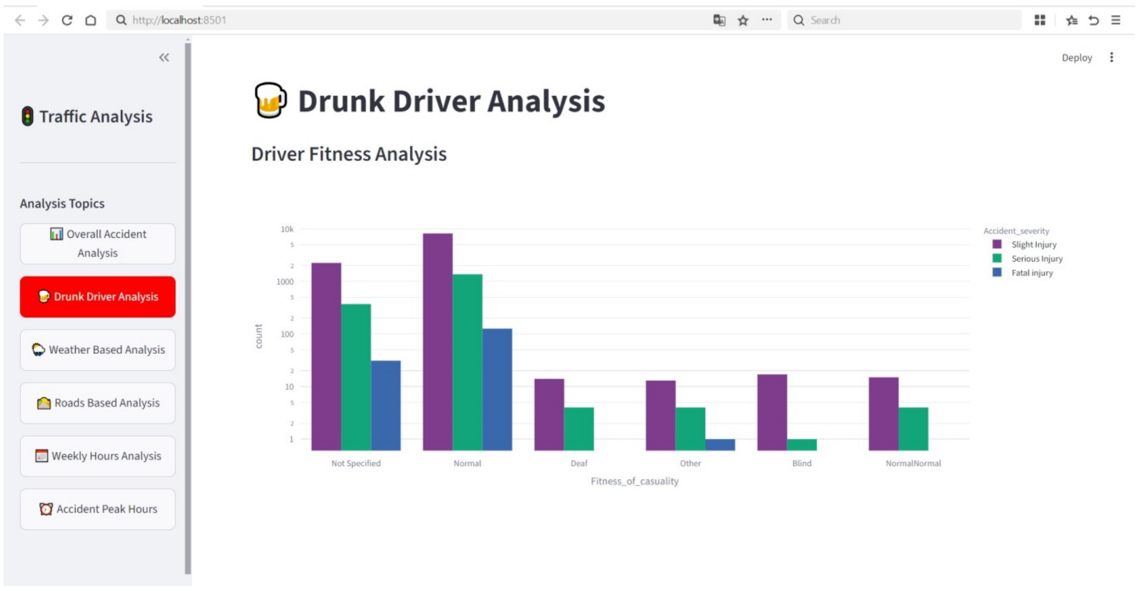
```
     m3.metric("Peak Hour", f"{df['Hour'].mode()[0]}:00")
      st.markdown("---")
      col1, col2 = st.columns(2)
      with col1:
         st.subheader("□ Severity Breakdown")
         # Using high-contrast colors
         fig_pie = px.pie(df, names='Accident_severity', hole=0.4,
                     color_discrete_sequence=px.colors.qualitative.Set1)
         fig_pie.update_traces(textinfo='percent+label', pull=[0.1, 0, 0])
         st.plotly_chart(fig_pie, use_container_width=True)
      with col2:
         st.subheader("📅 Weekly Frequency")
         fig_day=px.bar(df['Day_of_week'].value_counts(),
color_discrete_sequence=['#E41A1C'])
         st.plotly_chart(fig_day, use_container_width=True)
   elif topic == "Drunk Driver Analysis":
      st.subheader("Driver Fitness Analysis")
      # Log Scale makes small categories visible
      fig = px.histogram(df, x='Fitness_of_casuality', color='Accident_severity',
                  barmode='group', log_y=True,
                  color_discrete_sequence=px.colors.qualitative.Bold)
      st.plotly_chart(fig, use_container_width=True)
   elif topic == "Weather Based Analysis":
      st.subheader("Accidents by Weather")
      # Horizontal bars prevent names from overlapping
      fig = px.histogram(df, y='Weather_conditions', color='Accident_severity',
                  orientation='h', log_x=True,
                  color_discrete_sequence=px.colors.qualitative.Vivid)
      st.plotly_chart(fig, use_container_width=True)
```

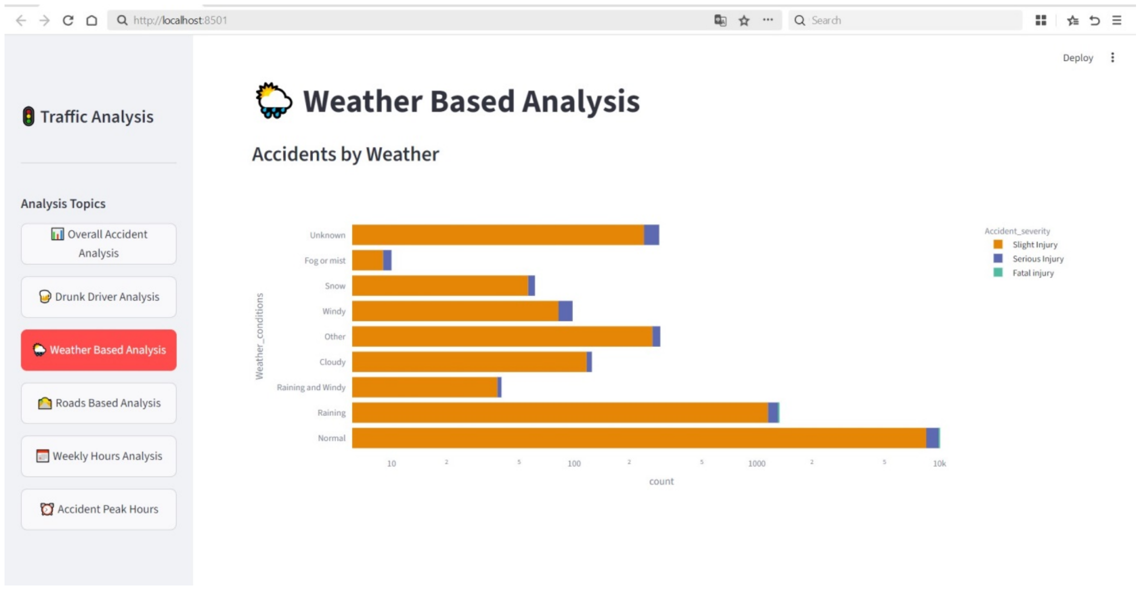
```
    elif topic == "Roads Based Analysis":
        st.subheader("Road & Light Condition Distribution")
        # FIXED: Bold color scale so it doesn't look faded
        fig = px.sunburst(df, path=['Road_surface_type', 'Light_conditions'],
                          color='Road_surface_type',
                          color_discrete_sequence=px.colors.qualitative.Dark24)
        fig.update_traces(textinfo='label+percent entry')
        st.plotly_chart(fig, use_container_width=True)
    elif topic == "Weekly Hours Analysis":
        st.subheader("Accident Intensity Heatmap")
        fig = px.density_heatmap(df, x="Hour", y="Day_of_week", text_auto=True,
                                 color_continuous_scale='YlOrRd')
        st.plotly_chart(fig, use_container_width=True)
    elif topic == "Accident Peak Hours":
        st.subheader("Hourly Accident Trends")
        hour_counts = df['Hour'].value_counts().sort_index().reset_index()
        hour_counts.columns = ['Hour', 'Accidents']
        fig = px.area(hour_counts, x='Hour', y='Accidents', markers=True,
                      color_discrete_sequence=['#377EB8'])
        st.plotly_chart(fig, use_container_width=True)
except Exception as e:
    st.error(f"Something went wrong: {e}")
```

7.RESULTS

7.1:OUTPUT SCREENS

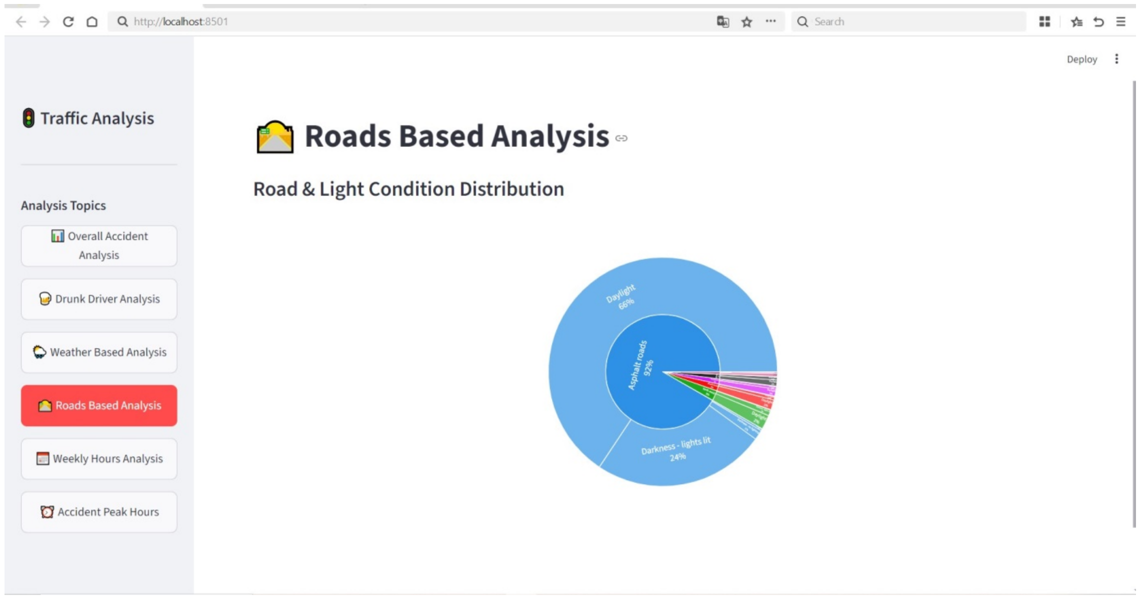






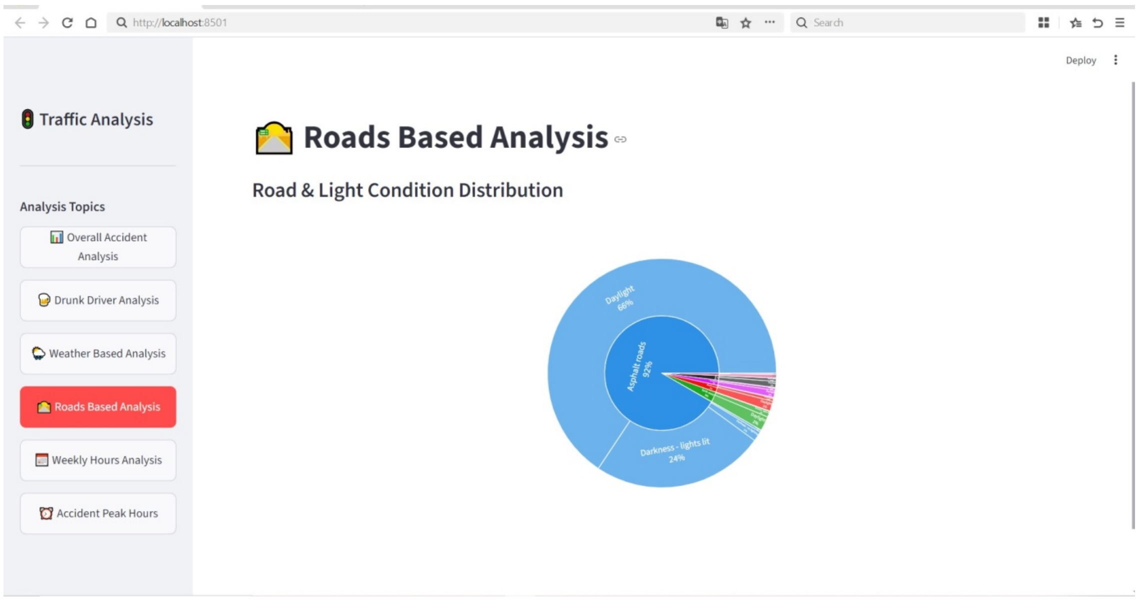
Weather Based Analysis

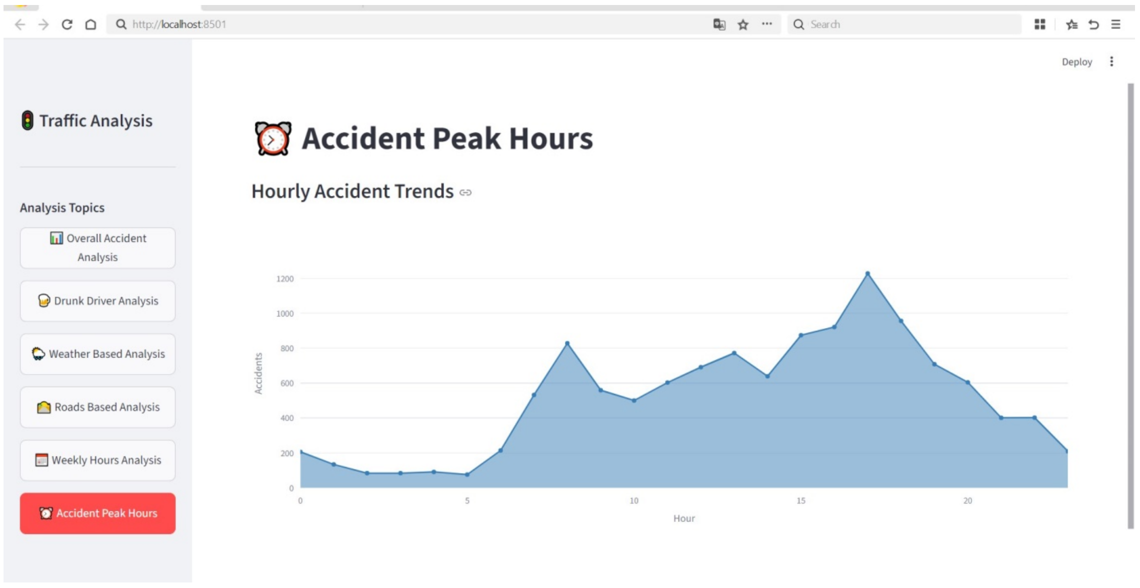
Accidents by Weather



Roads Based Analysis

Road & Light Condition Distribution


http://localhost:8501
Search
Deploy
Traffic Analysis
Analysis Topics
Overall Accident Analysis
Drunk Driver Analysis
Weather Based Analysis
Roads Based Analysis
Weekly Hours Analysis
Accident Peak Hours
Roads Based Analysis
Road & Light Condition Distribution
Daylight 66%
Asphalt roads 92%
Darkness - lights lit 24%


http://localhost:8501
Search
Deploy
Traffic Analysis
Analysis Topics
Overall Accident Analysis
Drunk Driver Analysis
Weather Based Analysis
Roads Based Analysis
Weekly Hours Analysis
Accident Peak Hours
Accident Peak Hours
Hourly Accident Trends
Accidents
Hour
0
200
400
600
800
1000
1200
0
5
10
15
20

8. Testing

Quality Assurance and Validation

8.1. Introduction

Testing ensures that the **Traffic Accident Analysis System** functions correctly, processes accident datasets accurately, and generates reliable visual insights. This phase verifies that all analysis modules, data handling operations, and visualizations perform as expected and meet the project requirements.

8.2 Types of Testing Used

Unit Testing:
Validates individual functionalities such as data loading, time extraction (hour and day), and severity classification.

Integration Testing:
Ensures that processed accident data flows correctly from the dataset engine to the visualization modules.

System Testing:
Verifies the complete workflow from loading the CSV file to displaying analytical dashboards.

Usability Testing:
Checks the ease of navigation, readability of charts, and overall user experience of the dashboard.

8.3 Test Cases

Test Case 1: Data Loading and Preprocessing

- **Description:** Verify that the system correctly loads accident data from CSV files.
- **Precondition:** CSV dataset is available in the specified directory.
- **Test Steps:**
    1. Launch the application.
    2. Observe if the dashboard loads without errors.
- **Expected Result:** Accident data is loaded and displayed successfully.
- **Actual Result:** Data loaded correctly (Pass).

Test Case 2: Accident Severity Visualization

- **Description:** Verify that accident severity distribution is displayed correctly.
- **Precondition:** Dataset contains valid severity values.
- **Test Steps:**
    1. Navigate to Overall Accident Analysis.
    2. Observe the severity pie chart.
- **Expected Result:** Pie chart displays correct severity proportions.
- **Actual Result:** Severity distribution displayed accurately (Pass).

Test Case 3: Weather-Based Analysis

- **Description:** Verify that accidents are grouped correctly based on weather conditions.
- **Precondition:** Weather condition data is present in the dataset.
- **Test Steps:**
    1. Select Weather-Based Analysis from the sidebar.
    2. Observe the bar chart.
- **Expected Result:** Chart reflects accident frequency by weather condition.
- **Actual Result:** Weather analysis displayed correctly (Pass).

Test Case 4: Time-Based Trend Analysis

- **Description:** Verify weekly and hourly accident trends.
- **Precondition:** Time data is available in the dataset.
- **Test Steps:**
    1. Navigate to Weekly Hours Analysis.
    2. Observe the heatmap visualization.
- **Expected Result:** Heatmap shows correct accident density by day and hour.
- **Actual Result:** Trends visualized correctly (Pass).

Test Case 5: Road and Lighting Condition Analysis

- **Description:** Verify the sunburst visualization for road and lighting conditions.
- **Precondition:** Road surface and lighting condition data exist.
- **Test Steps:**
    1. Select Roads Based Analysis.
    2. Observe the sunburst chart.
- **Expected Result:** Chart displays accurate hierarchical distribution.
- **Actual Result:** Visualization generated correctly (Pass).

Test Case 6: Sidebar Navigation

- **Description:** Verify that the sidebar navigation switches between analysis modules correctly.
- **Precondition:** Application is running.
- **Test Steps:**
    1. Click different analysis options in the sidebar.
    2. Observe page transitions.
- **Expected Result:** Selected analysis module loads without delay.
- **Actual Result:** Navigation works smoothly (Pass).

9. Conclusion

The **Traffic Accident Analysis System** project successfully demonstrates the design and implementation of an interactive accident analytics platform. By bridging the gap between raw traffic accident datasets and meaningful safety insights, the system provides a unified solution for analyzing accident patterns, severity levels, and contributing factors affecting road safety.

Key Achievements

• **Interactive Analytical Dashboard:**
Implementation of a web-based dashboard that allows users to explore accident data through multiple analytical modules such as severity analysis, weather-based analysis, driver fitness analysis, road condition analysis, and time-based trends.

• **Data-Driven Accident Analysis:**
Development of statistical logic to analyze accident severity distribution, peak accident hours, and high-risk conditions using historical accident records.

• **Advanced Visualization:**
Integration of interactive Plotly visualizations including pie charts, bar graphs, heatmaps, sunburst charts, and trend graphs for clear and intuitive accident analysis.

• **Data Integrity & Performance:**
Use of Pandas for efficient data cleaning and preprocessing, ensuring accurate results while handling large accident datasets smoothly.

• **Time-Based Risk Identification:**
Identification of accident-prone time periods such as peak hours and high-risk weekdays, supporting targeted road safety planning.

Final Summary

Through structured methodologies—including UML modeling, systematic data processing, and thorough testing—the project delivers a reliable and scalable solution for traffic accident analysis. By transforming large volumes of accident data into meaningful visual insights, the system enables users to identify patterns that contribute to accidents and understand critical safety risks.

In conclusion, the **Traffic Accident Analysis System** meets its functional objectives and provides significant value in improving accident awareness and road safety analysis. The project stands as a practical, well-documented analytical platform that can be extended in the future for predictive accident modeling, real-time data integration, and advanced traffic safety management.

10. FUTURE SCOPE

While the current **Traffic Accident Analysis System** provides an effective and interactive platform for analyzing historical traffic accident data, there are several opportunities to enhance and expand its functionality in future versions. The following enhancements can significantly improve system accuracy, usability, and real-world applicability:

• **Real-Time Accident Data Integration:**
Future versions can integrate live traffic and accident data from traffic authorities or IoT sensors to enable real-time accident monitoring and analysis.

• **Predictive Accident Analysis:**
Machine learning algorithms can be implemented to predict accident-prone locations, time periods, and risk levels based on historical trends.

• **Geographical Mapping and GIS Integration:**
Incorporating map-based visualization using tools such as Google Maps or GIS platforms can help identify accident hotspots and visualize location-based risk patterns.

• **Advanced Driver Behavior Analysis:**
Additional data such as vehicle speed, braking patterns, and driver behavior can be analyzed to gain deeper insights into accident causes.

• **Automated Report Generation:**
Future enhancements can include automatic generation of downloadable accident analysis reports for authorities and researchers.

• **Multi-User Role Support:**
Introducing role-based access such as traffic analysts and administrators can enable better data management and controlled access.

• **Mobile Application Development:**
Developing mobile applications can allow users to access accident analytics on smartphones and tablets for better accessibility.

• **Integration with Emergency Response Systems:**
The system can be extended to support emergency services by providing real-time alerts and risk notifications.

These enhancements will significantly broaden the system’s capabilities and improve its usefulness for traffic authorities, researchers, and road safety planners. With continued development, the **Traffic Accident Analysis System** can evolve into a comprehensive intelligent traffic safety and decision-support platform.

11. References

The following resources were used for the programming logic, data processing, and dashboard development of the **Traffic Accident Analysis System**:

• **Python Official Site:**
The core programming language used for data processing, analysis logic, and application development.
https://www.python.org/

• **Streamlit Documentation:**
The primary framework used to build the interactive web-based dashboard and sidebar navigation.
https://docs.streamlit.io/

• **Pandas Library:**
Used for reading traffic accident CSV files, cleaning missing or inconsistent data, and performing data aggregation.
https://pandas.pydata.org/

• **Plotly Graphing Library:**
Used to create interactive visualizations such as pie charts, bar graphs, heatmaps, sunburst charts, and trend plots.
https://plotly.com/python/

• **W3Schools Python Tutorials:**
Referenced for understanding Python syntax, data structures, and modular programming concepts.
https://www.w3schools.com/python/