Smart Crowd Management System with AI Integration for Optimised Pathfinding

Enroll.No.: 22103199, 22103204, 22103335
Name: Aishwarya Singh, Yogendra Kumar Gupta, Pratham Pandey
Project Supervisor: Dr.Vivek Kumar Singh

May-2025

Submitted in partial fulfillment of the Degree of

Bachelor of Technology

In

Computer Science Engineering

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING AND INFORMATION TECHNOLOGY

JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY, NOIDA

TABLE OF CONTENTS

Chapter No.	Topics	Page No.
Chapter-1	**Introduction**	**9**
	1.1 General Introduction	**9**
	1.2 Problem Statement	**9**
	1.3 Significance of the problem	**10**
	1.4 Empirical Study	**10**
	1.4.1 Objective of the Study	**10**
	1.4.2 Experimental Setup	**11**
	1.4.3 Methodology	**11**
	1.4.4. Challenges and Observations	**13**
	1.4.5. Future Improvements	**13**
	1.5 Description of the Solution Approach	**13**
	1.6 Comparison of Existing approaches to the problem framed	**14**
Chapter-2	**Literature Survey**	**16**
	2.1 Summary of papers studied	**16**
Chapter-3	**Requirement Analysis and Solution Approach**	**22**
	3.1 Overall description of the project	**22**
	3.2 Requirement Analysis	**22**
	3.3 Solution Approach	**23**
Chapter-4	**Modeling and Implementation Details**	**25**
	4.1 Design Diagram	**25**
	4.1.1 Use Case Diagram	**25**
	4.1.2 Control Flow Diagram	**25**
	4.1.2.1 Backend Control Flow Diagram	**25**
	4.1.2.2 Frontend Control Flow Diagram	**26**
	4.2 Implementation details and issues	**27**
	4.2.1. Data Integration and System Flow	**27**
	4.2.2. Key Implementation Components	**29**
	4.2.3. Challenges and Solutions	**32**
	4.3 Risk Analysis and Mitigation	**33**
Chapter-5	**Testing**	**35**
	5.1 Testing Plan	**35**
	5.2 Component Decomposition	**35**

	5.3 Test Result list 5.4 Error and Exception Handling 5.5 Limitations of the solution	**36** **36** **38**
Chapter-6	**Findings, Conclusion and Future Work** 6.1 Findings 6.2 Conclusion 6.3 Future Work	**39** **39** **39** **40**
References		**41**

Students' Self Declaration

We hereby declare the following usage of the open source code and prebuilt libraries in our minor project in sixth-Semester with the consent of our supervisor. We also measure the similarity percentage of pre written source code and our source code and the same is mentioned below. This measurement is true with the best of our knowledge and abilities.

1. List of pre build libraries
2. List of pre-built features in libraries or in source code.
3. Percentage of pre written source code and source written by us.

Student-id	Student-name	Signature
22103204	Yogendra Kumar Gupta	
22103199	Aishwarya Singh	
22103335	Pratham Pandey	

1. List of Pre-built Libraries Used

- In Python (multi_gate_counter.py, server.py)

- ➢ cv2 (OpenCV)
- ➢ ultralytics (YOLOv8)
- ➢ threading
- ➢ time
- ➢ requests
- ➢ flask
- ➢ flask_cors

- In JavaScript (script.js, f.html)

- ➢ Mapbox GL JS
- ➢ JavaScript (fetch, setInterval, navigator.geolocation)
- ➢ HTML5 and CSS for UI rendering

2. List of Pre-built Features from These Libraries

- OpenCV (cv2)

- ➢ cv2.VideoCapture: Open video streams (camera or file)
- ➢ ret, frame = cap.read(): Frame extraction
- ➢ cv2.imshow, cv2.rectangle, cv2.putText: Visualization on frames
- ➢ cv2.waitKey, cv2.destroyAllWindows: Control frame timing and cleanup

- Ultralytics YOLOv8

- ➢ YOLO('yolov8n.pt'): Loads the pre-trained YOLOv8 nano model
- ➢ .predict(source=...): Runs inference on images or video frames
- ➢ .boxes.xyxy: Retrieves bounding box coordinates
- ➢ .boxes.cls: Retrieves class IDs (used to detect only "person" class

- Requests

- ➢ requests.post(url, json=data): Send count data to Flask server

-Threading

- ➢ threading.Thread Allows each gate/video feed to run in parallel
- ➢ thread.start(), thread.join(): Manage concurrent execution

-Flask

- ➢ @app.route(...): Define REST API endpoints
- ➢ request.get_json(): Parse JSON from incoming POST requests

- Flask-CORS

- ➢ CORS(app): Enables cross-origin API calls from frontend (browser)

- Mapbox GL JS (JavaScript)

- ➢ new mapboxgl.Map(...): Initialize map
- ➢ new mapboxgl.Marker(...): Add gate markers to the map
- ➢ fetch(...): Call Flask API to get crowd data
- ➢ navigator.geolocation.getCurrentPosition: Get user’s location

CERTIFICATE

This is to certify that the work titled **“ Smart Crowd Management System with AI Integration for Optimised Pathfinding ”** submitted by **“Aishwarya Singh, Yogendra Kumar Gupta, Pratham Pandey”** in partial fulfillment for the award of degree of B.Tech Computer Science Engineering of Jaypee Institute of Information Technology, Noida has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Signature of Supervisor:

Name of Supervisor: Dr.Vivek Kumar Singh

Designation:Assistant Professor (Senior Grade)

Date:

ACKNOWLEDGEMENT

We would like to express my deepest gratitude to everyone who will contribute to the successful completion of this project. First and foremost, we are grateful to Dr. Vivek Kumar Singh, whose guidance, support, and expertise has been and will be invaluable throughout the course of this project. His insightful feedback and encouragement inspires us to push the boundaries of our understanding and to tackle challenges with a problem-solving mindset.

We would also like to thank Jaypee Institute of Information Technology Noida,CSE department for providing the resources and infrastructure necessary for conducting this project. The access to research materials, technology, and collaborative environments plays a crucial role in shaping the outcome of this work.

A special thank you to the group mates for their constant support and valuable discussions that enriched this project. Their collaboration and input helped refine the direction of our work.

Thank you all for making this project a success

Signature of Students:

Name of Students: Aishwarya Singh, Yogendra Kumar Gupta, Pratham Pandey

Enrollment of Students: 22103199, 22103204, 22103335

Date:

CHAPTER-1

INTRODUCTION

1.1 GENERAL INTRODUCTION

Over the past few years, there has been an urgent need for efficient crowd management systems in public and institutional places. As the population density increases and more people gather, effective crowd flow management is crucial to guarantee safety, avoid congestion, and maximize movement. Manual monitoring has been used in the past for crowd control, but this process can be slow, error-ridden, and inefficient in peak hours or during emergencies.

This project describes an intelligent Crowd Management System developed specifically for institutional settings with more than one entry and exit gate. The system combines AI-based video analysis and sensor data to track crowd density in real-time. Based on this information, it calculates the best gate for people to utilize dynamically routing to avoid overcrowding. The gathered information is processed on a backend server, which informs a user-friendly web interface displaying gate statuses and recommendations to end-users.

By automatically detecting and distributing crowd movement, this system increases safety, decreases manual labor, and provides a smooth experience for administrators and users alike. The project also aids in smart campus and smart city efforts by presenting an intelligent, scalable solution for crowd flow control.

1.2 PROBLEM STATEMENT

Public facilities and institutions frequently struggle with large groups of people, particularly during rush hours, events, or in case of emergencies. The lack of an intelligent, real-time system for tracking and directing crowd flow may result in overcrowding at specific gates, prolonged waiting times, and risks to safety in the form of stampedes or bottlenecks. Manual processes of crowd management are not only ineffective but also lack the dynamism to react to changing densities of people.

The main issue is the absence of an automated system that is capable of:

- Regularly tracking crowd volume across several gates,

- Processing the data in real-time through AI methods,
- Providing suggestions for best paths or gates to users,
- And presenting this data on an accessible, responsive platform.

This project seeks to address these problems by creating a smart crowd management system with AI-based crowd density estimation and a web-based interface for live user interaction, promoting optimal gate utilization and enhanced safety**.**

1.3 SIGNIFICANCE OF THE PROJECT

The project's significance is founded in its capability to solve realistic crowd control issues by employing an innovative technology-led approach. By implementing AI-facilitated crowd detection linked to a web-enabled guidance framework, this project provides a technologically enabled intelligent, scalable, and anticipatory solution for institutional and public setting crowd management.

Main features of the project are:

- Improved safety: Eliminates congestion and possible risk by dynamically relocating users to less crowded gates.
- Efficient operation: Lessens manual handling by automating crowd monitoring and reaction.
- User convenience: Delivers real-time gate status and best route recommendation to simplify user navigation.
- Scalability: Can be deployed to other locations such as malls, transport hubs, or sports stadiums with small modifications.
- Support for smart infrastructure: Aligns the objectives of smart campuses and smart cities through utilization of AI and data analytics.

This system not only works as a security tool but also as a starting point for future intelligent infrastructure systems that need real-time, adaptive decision-making.

1.4 EMPIRICAL STUDY

1.4.1 Objective of the Study

The goal of this empirical work is to assess the usability, efficacy, and effectiveness of the proposed Smart Crowd Management System. It seeks to show how real-time crowd detection based on AI and gate recommendation through a web-based system can enhance safety, avoid congestion, and dynamically guide users through numerous gate systems.

1.4.2. Experimental Setup

Parameter	Description
Environment	Simulated institutional campus with 3 entry/exit gate
Cameras Used	3 HD CCTV cameras(1 per gate)
AI Model	YOLOv8(You Only Look Once - version 8)
Backend	Python
Frontend	HTML, JS, CSS, Mapbox GL JS

1.4.3. Methodology

1.4.3.1.Data Preprocessing:

1) Gate Count Data:

- Crowd counts are simulated or actual-time and come from an object detection model (e.g., YOLOv8). In simulation mode, values dynamically change every few seconds to mimic real crowd oscillation.
- Data is cleaned before display to treat non-numeric or missing values nicely.

b) Geolocation Data:

- Gate coordinates are manually entered and served via a backend API.
- User location is retrieved using browser-based geolocation and sanitized for format and accuracy before use in path routing.

1.4.3.2.Map Integration and Visualization:

1) Mapbox GL JS:

- Mapbox GL JS is utilized to display the base map and place markers on the map for every gate. The style of the base map can be tailored through Mapbox.

- The Mapbox Directions API is utilized to determine the most direct route from the user to the least populated gate. Directions are displayed on the map by Mapbox GL JS.

2) Live Count Overlay:

- A UI box provides real-time, gate-wise crowd count figures.
- The frontend updates the view every 3 seconds with count information fetched from the backend.

1.4.3.3.Decision Logic for choosing Gate:

1) Count Threshold Logic:

- The gate with the least people is dynamically calculated and chosen to be the destination.
- If the values are equal, the gate with the lowest index is chosen.

2) API Coordination:

- Backend provides the most recent gate counts via a REST API (/get_counts) and gate locations via another endpoint (/get_gate_locations).

1.4.3.4.Model Integration (Optional - Planned):

1) YOLOv8 for People Counting:

- Another instance of the YOLOv8 model can be utilized to track each gate to detect people and estimate real-time counts.
- Output can be pushed to the backend via a dedicated /update_count API per gate.

2) Model Variant:

- yolov8 is optimal for real-time inference due to its light-weight architecture.

1.4.3.5.Evaluation Metrics:

1) User Experience Metrics:

- Map and route load time.
- Accuracy of crowd gate count (when combined with live model).
- Path consistency (i.e., does the map always point to the least crowded gate).

2) System Metrics (Planned):

- Latency between real change in count and reflected UI update.

- Scalability: Ability to serve multiple users and gates at once.

1.4.4. Challenges and Observations

1) Model Performance: YOLOv8 performed well in real-time crowd detection, but its accuracy sometimes decreased under poor lighting conditions, motion blur, or occlusion (e.g., close-packed groups as one object).
2) False Positives: Poles, signs, or shadows were occasionally confused with static objects as humans, especially in detailed visual scenes.
3) Crowd Count Variability: Rapidly moving people through the frame produced varying detection counts. Temporal smoothing was needed to prevent noisy live count renderings due to frame-by-frame fluctuations.
4) Latency in Live Updates: Backend frame processing and frontend display delays sometimes led to temporary misalignments between recommended paths and real crowd data.

1.4.5. Future Improvements

1) Model Fine-tuning: Fine-tune YOLOv8 on crowd-specific and event-specific datasets to enhance detection accuracy under changing environmental conditions.
2) Temporal Smoothing: Use frame-based averaging or tracking to stabilize crowd counts and eliminate fluctuations due to transient occlusions or rapid movement.
3) Multi-Camera Integration: Support automatic stitching or coordination among multiple cameras to enhance area coverage and minimize blind spots.
4) Mobile and Edge Deployment: Optimize model size and inference speed for mobile or edge device deployment with limited resources, enabling on-site real-time crowd monitoring.
5) Smart Path Recommendation Logic: Augment the path recommendation system with predictive models that take into consideration trends in crowd motion, as opposed to existing counts.
6) User Feedback Loop: Implement a mechanism for manual input or live feedback from authorities/users in order to enable the system to learn from real-world usage and become better with the passage of time.

1.5 DESCRIPTION OF SOLUTION APPROACH

The Proposed system for crowd management employs a multi-layered platform with the combination of artificial intelligence, data science, and web technologies to responsibly direct crowd traffic through multiple entrances of an entity.

The main components of the proposed solution are:

- AI-Fueled Crowd Detection: Dynamic video feeds obtained from cameras are processed using an AI algorithm (e.g., YOLOv8) to identify individuals at each entry point.
- Density Calculation: Identified people are utilised to measure crowd density dynamically.
- Backend Processing: Crowd data is processed by a centralized backend server to identify the least crowded gate for efficient distribution.
- Web Interface: A dynamic website presents gate statuses, crowd levels, and informs users of suggested gates to visit.

This method enables real-time decision-making, minimizes the burden on any single gate, and provides user safety and convenience through intelligent redirection.

1.6 COMPARISON OF EXISTING APPROACHES TO CROWD MANAGEMENT

1)Traditional Methods (Manual or Simple Systems)

- Depend largely on human surveillance or security staff.
- Utilize CCTV monitoring but no smart analysis.
- Don't have actual real-time data processing.
- Offer static information such as fixed signage or manual re-routing.
- Response is slow and erratic.
- Hard to scale without increasing personnel.
- There is no active guidance of users during crowd events.

2)Proposed Smart Crowd Management System

- Employ AI models (e.g., YOLO) for real-time detection of people from camera streams.
- Conduct automated crowd density analysis.
- Processes data in real time through a backend server.
- Offers dynamic gate recommendations through a website interface.
- Scalable and flexible with little rise in cost.
- Interacts with users through live guidance and feedback.

Tabular Comparison:

Aspect	**Traditional Approach**	**Proposed Smart System**
Monitoring Method	Manual Observation/ basic CCTV	AI based crowd detection(e.g., YOLO)
Data Processing	Human Based, often delayed	Automated ,real time via backend
Guidance Provided	Static signs , manual redirection	Live suggestions via website
Response Time	Slow, Human Dependent	Fast and automated
Accuracy	Varies with observer	Consistent and AI-Driven
Scalability	Requires more manpower	Easily scalable with cameras and servers
User Interaction	Passive	Active- user receives suggestions
Safety and Efficiency	Reactive, risk prone	Optimises flow and safety

Figure 1.1

CHAPTER-2

LITERATURE SURVEY

2.1 SUMMARY OF PAPER STUDIED

The read literature examines the critical concern of people safety in outdoor stadium settings, with a focus on the facility management's responsibility for public safety. The research explores the issues related to large events, specifically the weaknesses inherent in stadium design, crowd management, and emergency response.

Major findings indicate that most outdoor arenas are plagued by design weaknesses and infrastructural deficiencies, which have the potential to severely undermine spectator safety. Some of these include poor entry and exit points, inadequate signage, insufficient seating, and absence of accessible emergency facilities.

Crowd management is given significant priority as a central element in ensuring safety. In the absence of proper management measures, including controlled entry, assigned emergency exits, and trained staff, the danger of crowd surges and stampedes becomes exponentially higher. Crowd behavior analysis, real-time monitoring, and well-coordinated emergency procedures are identified in this paper as critical measures for reducing these dangers.

Another important area addressed is emergency preparedness. The review emphasizes that most stadiums have inadequate emergency response plans or do not effectively communicate these to the public. Recommended are emergency drills, training of staff, and clear, easily accessible evacuation procedures to improve response times and minimize casualties in the case of a crisis.

The report also calls for increased awareness of safety among people attending the event. Informing people of the possible dangers, emergency response measures, and anticipated conduct in the event of a crisis has been shown to have a direct positive effect on overall safety performance. Mobile alerts, public address systems, and printed materials are the channels through which such information should be conveyed.

A)Purpose and Significance Mentioned[1]:

The research seeks to investigate crowd management using various important lenses:

1)Importance: It emphasizes the significance of crowd management in accident prevention, legality, and movement efficiency.

2)Analyze various crowd controlling strategies: The paper deconstructs some physical and strategic control techniques such as barriers, signs, deployment of security, and segmentation of crowds.

3)Technology's Role: It examines the revolutionary role of AI, big data, and IoT in real-time monitoring, predictive analytics, and emergency response.

4)Psychological and Behavioural aspects: Crowd psychology—social influence, emotional contagion, and spatial dynamics—is highlighted as crucial to successful interventions.

5)Case Studies: Classic events like the 2012 London Olympics (a success) and the 2021 Astroworld Festival (a failure) are reviewed for best practices and cautionary lessons.

6)Ethical and Legal Issues: The study addresses balancing safety with privacy, accessibility, and human dignity in crowd control situations.

7)Future Forecasting: The study predicts robotic support, smart cities, and behavioral biometrics to have a central role in shaping future crowd management paradigms.

B) Objectives of the study mentioned[1] :

1)Deconstruct Effective Practices: Dissect the planning, communication, spatial design, and behavioral science that goes into crowd management.

2)Examine the Use of Technology: Consider contemporary and emerging technologies used for monitoring, data assessment, communication, and crowd profiling.

3)Identify Event-Specific Challenges: Assess how various activities and venues—like festivals, religious ceremonies, and transport facilities—need activity-specific management methods.

4)Assess the Effects of Inefficient Management: Draw upon actual incidents to evaluate the ramifications of poor planning, such as injuries, casualties, and mistrust from the public.

5)Establish Best Practices: Suggest strategic, evidence-based solutions that integrate planning, technology, and human understanding for enhanced crowd safety.

C) Important Research Gaps and Future Research Horizons[1]

The article lists five major areas where research needs to be undertaken and suggests concrete research directions for the future:

I. Technological Horizons

- Real-Time AI and Predictive Analytics: Transition from passive observation to active, adaptive decision-making frameworks.
- IoT and Smart Sensor Networks: Utilize wearable and embedded sensors for crowd density, movement, and environmental measurements.
- Augmented Reality (AR): Explore the prospects of using AR for dynamic navigation and dispersal of crowds during emergencies.
- Cybersecurity: Secure weaknesses in surveillance networks against misuse for exposure of sensitive biometric and movement information.

II. Psychological and Behavioral Dimensions[1-3]

- Panic Behavior: Analyze psychological and cultural panic initiators in crowds and formulate measures to prevent panic behavior.
- Social Media Influence: Learn about the influence of misinformation and viral content in intensifying or defusing crowd activity.
- Behavioral Predictability Models: Refine existing models to include unpredictability, social interactions, and environment-induced changes.
- Emotional Contagion: Chart how emotions are transmitted among crowds and how leadership and messaging affect collective response.

III. Policy and Legal Frameworks

- Standardized Global Guidelines: Establish harmonized rules and safety standards for crowd management globally.
- Ethical Usage of Technology: Balance surveillance with civilian freedoms by establishing obvious ethical limits and consent-driven data practice.
- Legal Accountability: Establish roles and responsibilities for crowd catastrophes, maintaining accountability for all concerned.

- Rights and Liberties: Assess the intersection of crowd control with human rights, especially where protests or sensitive religious gatherings occur.

IV. Operational and Logistical Efficiency

- Training Protocols: Create worldwide standards for crowd management staff education and readiness.
- Urban Evacuation Planning: Create real-time evacuation frameworks that work optimally in congested settings.
- Resource-Limited Settings: Establish affordable, expandable crowd control technologies for developing nations.
- Multi-Agency Collaboration: Enhance communication and shared operational practices between police, emergency services, and private security.

V. Environmental and Sustainability Considerations

- Eco-Friendly Crowd Solutions: Create environmentally friendly practices to minimize the environmental footprint of high-density events (e.g., waste, emissions).
- Climate Change: Investigate the impact of extreme weather and environmental stressors on crowd behavior and safety.
- Energy-Efficient Surveillance: Lower the carbon footprint of digital surveillance tools by means of renewable power and low-energy systems.

D) Key Findings of the Study

1)Planning and Infrastructure- Proper control of entry and exit points, clear direction markers, and routed walkways diminish overcrowding and danger. Gatherings that have well-managed crowd-control measures record reduced safety issues and more efficient working.

2)Technology's role- CCTV, computer vision-based monitoring, and smartphone alert systems enhance real-time people tracking. All these help prompt proactive actions with the ability to detect high-density areas before troubles arise.

3)Behavior depending on event type- Crowd behavior is dissimilar across the type of events:

- Sport events: High vigor, rapid pace
- Religious events: Slower pace, larger crowds

4)Emergency Preparedness- Effective communication, prepared staff, emergency signage, and first-aid stations are critical. Evacuation procedures and rapid-response teams mitigate panic and control crises effectively.

5)Personnel Training- Ongoing training of event staff and liaison with emergency services is critical for effective on-ground management.

Tabular summary:

Section	**Description**
Abstract	Public safety in outdoor stadiums; the role of facility management; issues in design, crowd control, and emergency preparedness.
Major Safety Issues	- Poor entry/exit design - Inadequate signage - Insufficient seating - Lack of emergency access facilities
Importance of Crowd Management	- Regulated entry/exit - Trained staff - Real-time monitoring - Behavior analysis[2-3] - Emergency procedure coordination
Emergency Preparedness	- Many venues lack robust emergency plans - Recommendations: drills, staff training, clear evacuation routes
Spectator Awareness	- Communicate via mobile alerts, PA systems, printed materials - Educate on hazards and appropriate crisis behavior
Purpose of Study	- Explore crowd management across lenses: strategy, psychology, legal, technological, and future directions
Significance	- Safety, legal compliance, operational efficiency
Key Lenses of Analysis	- Physical control (barriers, signs) - Tech (AI, big data, IoT) - Crowd psychology - Ethics and law - Future innovations
Objectives	- Deconstruct effective practices - Examine tech tools - Analyze context-specific needs - Evaluate poor management outcomes - Suggest best practices

Research Gaps	- Urban crowd behavior - Real-world AI/IoT deployment - Absence of global standards - Under analysis of recent incidents
Future Research Directions I. Technology	- Real-time AI & prediction - IoT sensor networks - Augmented reality for evacuation - Cybersecurity in surveillance
II. Psychology & Behavior[4]	- Panic triggers - Impact of misinformation - Better behavior prediction models - Understanding emotional contagion
III. Policy & Legal	- Global safety standards - Ethical tech usage - Legal accountability in disasters - Human rights during control
IV. Operations & Logistics	- Global training protocols - Urban evacuation systems - Low-resource solutions - Multi-agency coordination
V. Environmental Concerns	- Eco-friendly event practices - Climate-related risks - Energy-efficient surveillance
Sampling Methods Used	- Random Sampling - Systematic Sampling - Stratified Sampling - Cluster Sampling - Convenience Sampling - Time-Based Sampling
Key Findings	- Infrastructure: Entry/exit design & signage reduce risk - Technology: Enables preemptive responses - Behavior: Event type affects movement - Preparedness: Training & communication are vital

FIGURE 2.1 [1]

CHAPTER-3
REQUIREMENT ANALYSIS AND SOLUTION APPROACH

3.1 OVERALL DESCRIPTION OF THE PROJECT

This project proposes designing a Smart Crowd Management System for a multi-gate institutional setup. It wishes to facilitate secure, efficient, and well-distributed movement of individuals across varying entry and exit points by tapping into the usage of AI-facilitated video analytics along with a web-based interface.

The system works by:

- Extracting real-time video feeds of surveillance cameras located at each entrance.
- Employing an AI-based model for measuring crowd density at each site.
- Transmitting the computed data to a backend server, which computes the best gate based on real-time congestion.
- Utilizing **Mapbox integration** and **user geolocation** to provide personalized directions to the least crowded gate.
- Displaying gate statuses, congestion levels, and recommended paths dynamically on a responsive website.

This smart system not only reduces human effort but also improves safety and decision-making—especially during events, emergencies, or peak hours—by guiding users in real-time to the best entry or exit points.

3.2 REQUIREMENT ANALYSIS

1)Functional Requirements:

I. Real-time detection of crowds at every gate utilizing AI.
II. Backend processing for analyzing crowd data and making decisions on optimal gates.
III. Web interface to display live crowd status and suggestions.
IV. Alert when a gate is overcrowded.
V. Admin dashboard to track and control the system.

2)Non-Functional Requirements:

I. High system availability and quick response time.
II. Scalable to add additional gates or environments.
III. Secure handling of data and privacy for video streams.
IV. Friendly and responsive website interface.

3)Hardware Requirements:

I. Surveillance cameras (one per gate).
II. Server/computer to execute the AI model and backend logic.
III. Internet connectivity for real-time updates.
IV. Display boards (optional) for on-site gate directions.

4)Software Requirements:

I. AI model (e.g., YOLOv8 for crowd detection).
II. Backend technologies (e.g., Python, Flask).
III. Frontend technologies (e.g., HTML, CSS, JS,).
IV. Database for logging data (e.g., MySQL,).

3.3 SOLUTION APPROACH

Here is a description of each point in the solution strategy:

1. Video Capture Module:
   - This module captures continuous real-time video from surveillance cameras mounted at entry and exit gates.
   - Cameras that are placed in a strategic location to record wide views of the crowd, which will be the raw input for the system.
   - Video capture facilitates correct and timely identification of crowd size from clear video feeds.

2. AI Processing Module:
   - This module processes video frames through a trained deep learning model (e.g., YOLOv8) to identify and count people in real time.
   - AI Processing Module translates video data into quantitative crowd density data by detecting and counting individuals.
   - The importance of this module is that it generates correct real-time crowd density for every gate.

3. Decision Engine (Backend):
   - The purpose of the decision engine is ,it serves as the system brain by taking in AI-processed data and deciding on the basis of predetermined logic.

- The Decision Engine cross-compares crowd densities at gates, finds the least congested gate, and diverts crowd flow to that gate.
- It facilitates equitable crowd distribution, avoids overcrowding, and generates alerts when needed.

4. Web Interface.
- Web Interface presents a real-time graphical interface for users and administrators.
- It presents live gate people count , displays crowd trends, and suggests gates.
- It enables guidance to the user to check the best gate.

5.Mapbox GL JS and Geolocation Integration
- The purpose is to give users real-time navigation to the least busy gate from where they are currently located. Through the user's geolocation, it harmoniously works with Mapbox GL JS to give a suggested path to the least busy gate.
- It uses Mapbox Directions API to determine the shortest path from the user's position to the least congested gate.
- The significance is that this feature is most helpful for large venues or campuses and is really useful for new visitors who are not sure where to go. Not only does it recommend the best gate but also tells you the quickest, most effective way to get to it, reducing walking time in crowded places.

CHAPTER - 4
MODELING AND IMPLEMENTATION DETAILS

4.1 DESIGN DIAGRAMS:

4.1.1 USE CASE DIAGRAM

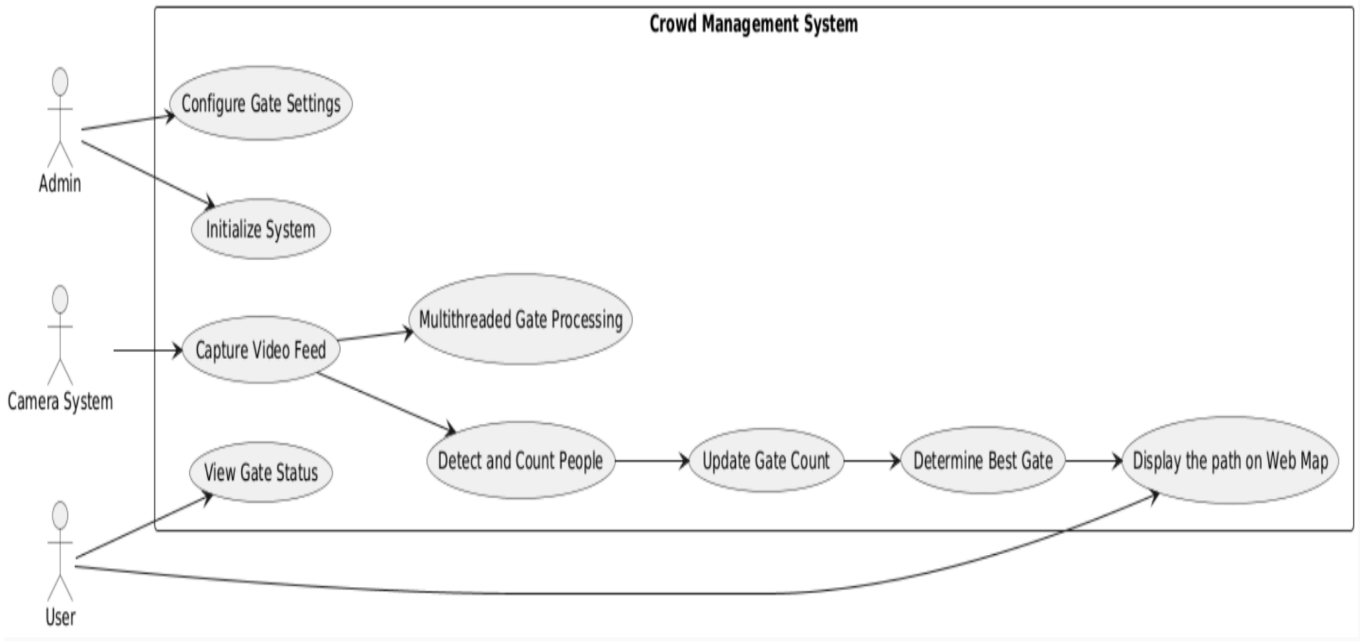


Figure 4.1[5]

4.1.2 CONTROL FLOW DIAGRAMS

4.1.2.1Backend Control Flow:

Start: Capture Video Feed from Gate N

|

YOLOv8 Crow Detection Mode
(Perform initial detection on visual data to estimate crowd density)

|

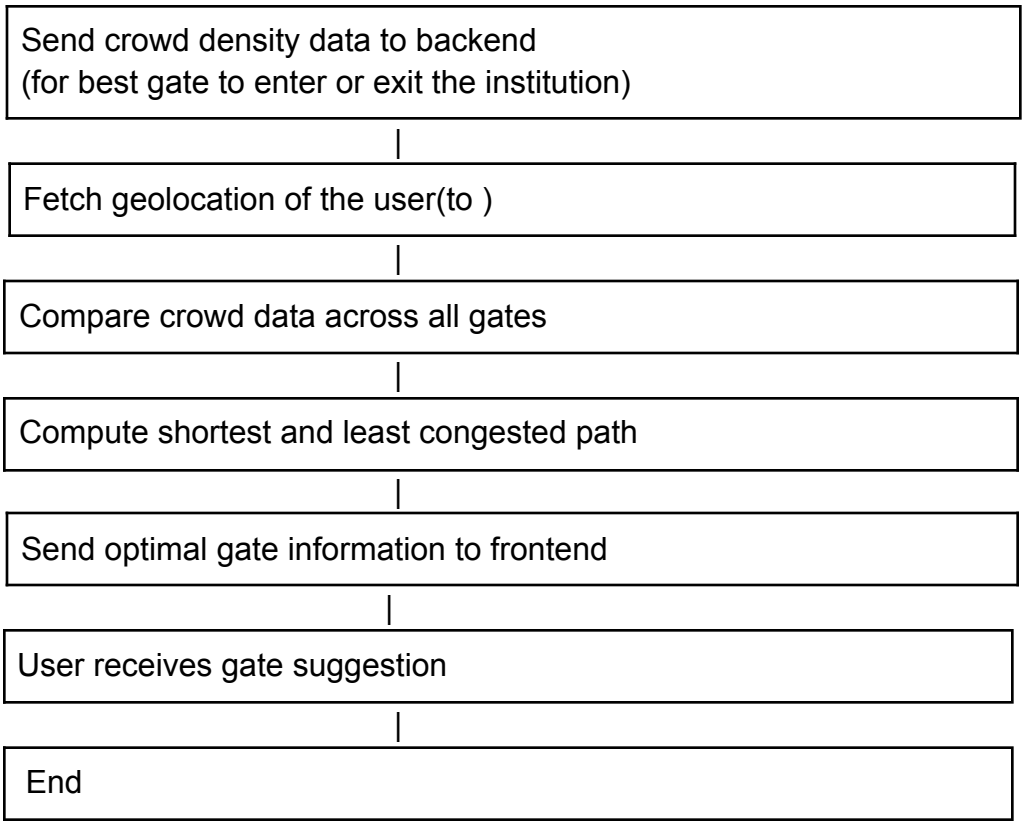
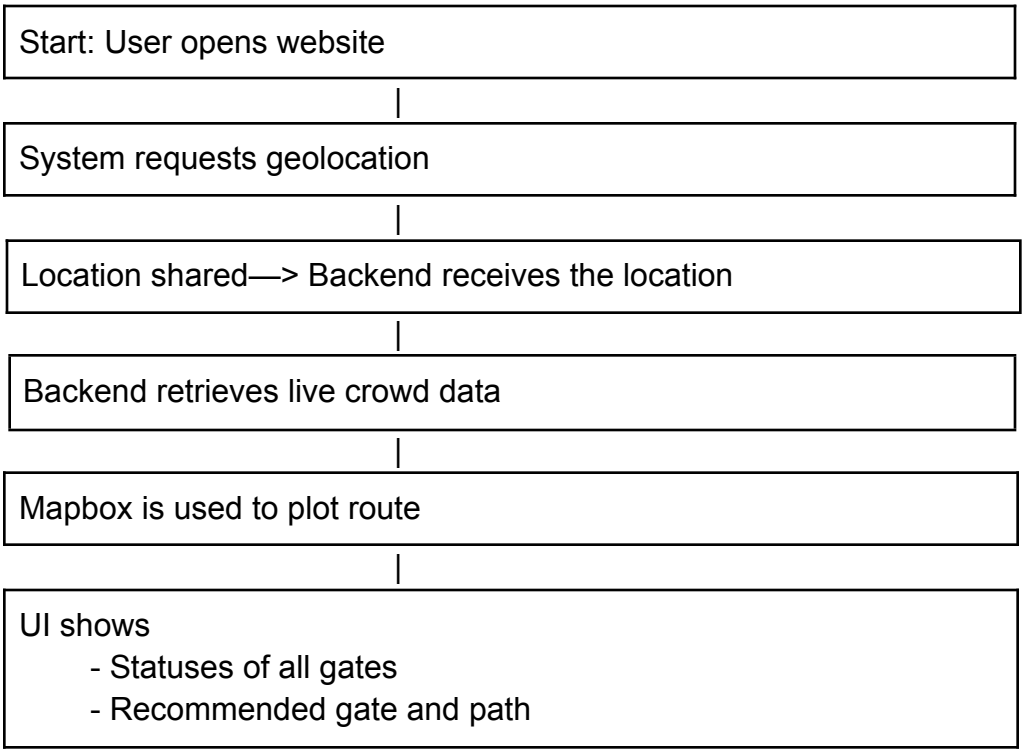


Figure 4.2

4.1.2.1 Frontend Control Flow:



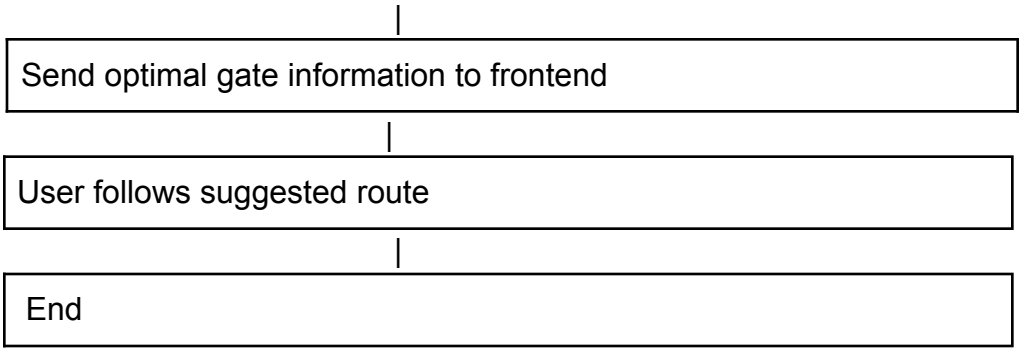


Figure 4.3

4.2 IMPLEMENTATION DETAILS AND ISSUES

4.2.1. Data Integration and System Flow

1)Project Overview: This project consolidates real-time crowd count data from three gates (Gate 1, Gate 2, Gate 3) with video processing based on YOLOv8 and merges it with live geolocation routing based on Mapbox Directions API. The aim is to direct users dynamically to the least congested gate and present live headcounts for all gates superimposed on the map interface.

2)Frontend-Backend Architecture:

I)People Detection and Counting – multi_gate_counter.py

```
(constant) VIDEO_SOURCES: dict[int, str]
VIDEO_SOURCES = {
    1: "C:\\Users\\rekha\\OneDrive\\Desktop\\pc.mp4",
    2: "C:\\Users\\rekha\\OneDrive\\Desktop\\pc2.mp4",
    3: "C:\\Users\\rekha\\OneDrive\\Desktop\\pc3.mp4"
}

RECT_AREAS = {
    1: ((100, 100), (500, 400)),
    2: ((100, 100), (500, 400)),
    3: ((100, 100), (500, 400))
}

SERVER_URL = "http://127.0.0.1:5000/update_count"

# Load model once
model = YOLO('yolov8n.pt')
```

- Uses YOLOv8 to process video feeds per gate.

- Defines a Region of Interest (ROI) for each gate.
- In each frame:
    - Run detection.
    - Counts people whose centroid falls inside the ROI.

II)Backend (Flask):

```
app = Flask(__name__)
CORS(app)


gate_counts = {1: 0, 2: 0, 3: 0}


gate_locations = {
    1: {'lat': 28.637837093024938, 'lng': 77.37822128540587},
    2: {'lat': 28.63890283190635, 'lng': 77.37585341448568},
    3: {'lat': 28.63891615357386, 'lng': 77.37221053614691},
}
```

- Hosts APIs to provide live gate counts and fixed geo locations of every gate.
- Updates in real-time from a YOLOv8-based pipeline through the /update_count endpoint.
- Replaces legacy simulation-based updates with real object detection output.

III)Frontend (HTML + JavaScript + Mapbox):

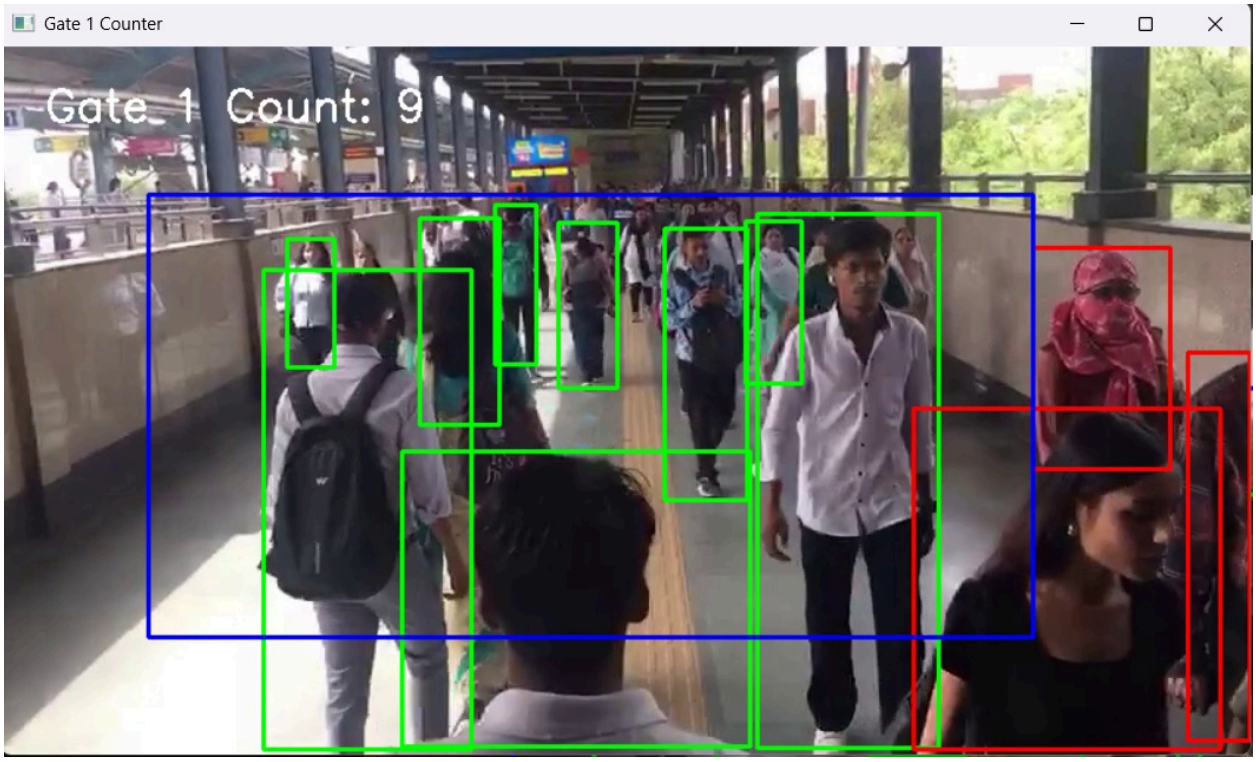
```
navigator.geolocation.getCurrentPosition(successLocation, errorLocation, {
  enableHighAccuracy: true,
});
```

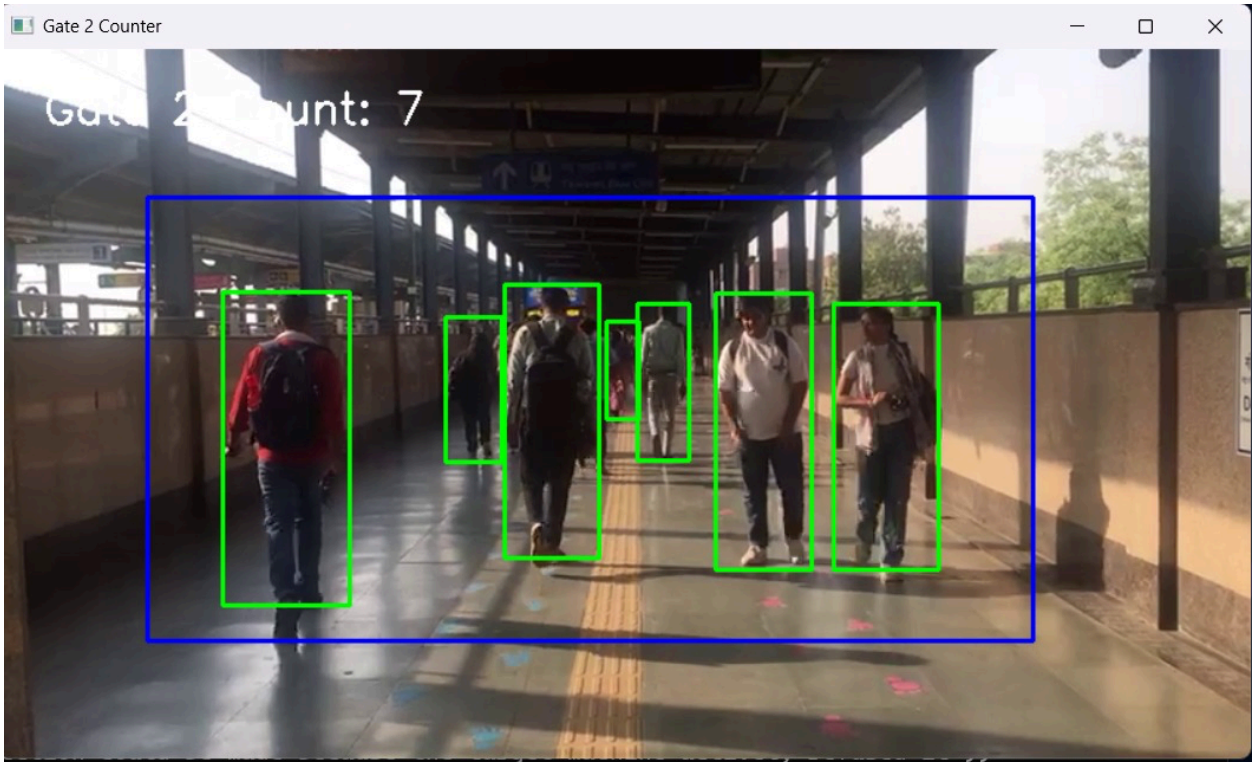
```
  <div id='map'></div>
  <div id="gate-counts" style="position: absolute; top: 10px; left
  <h3>Live Gate Counts</h3>
  <p>Gate 1: <span id="gate1-count">Loading...</span></p>
  <p>Gate 2: <span id="gate2-count">Loading...</span></p>
  <p>Gate 3: <span id="gate3-count">Loading...</span></p>
</div>
```

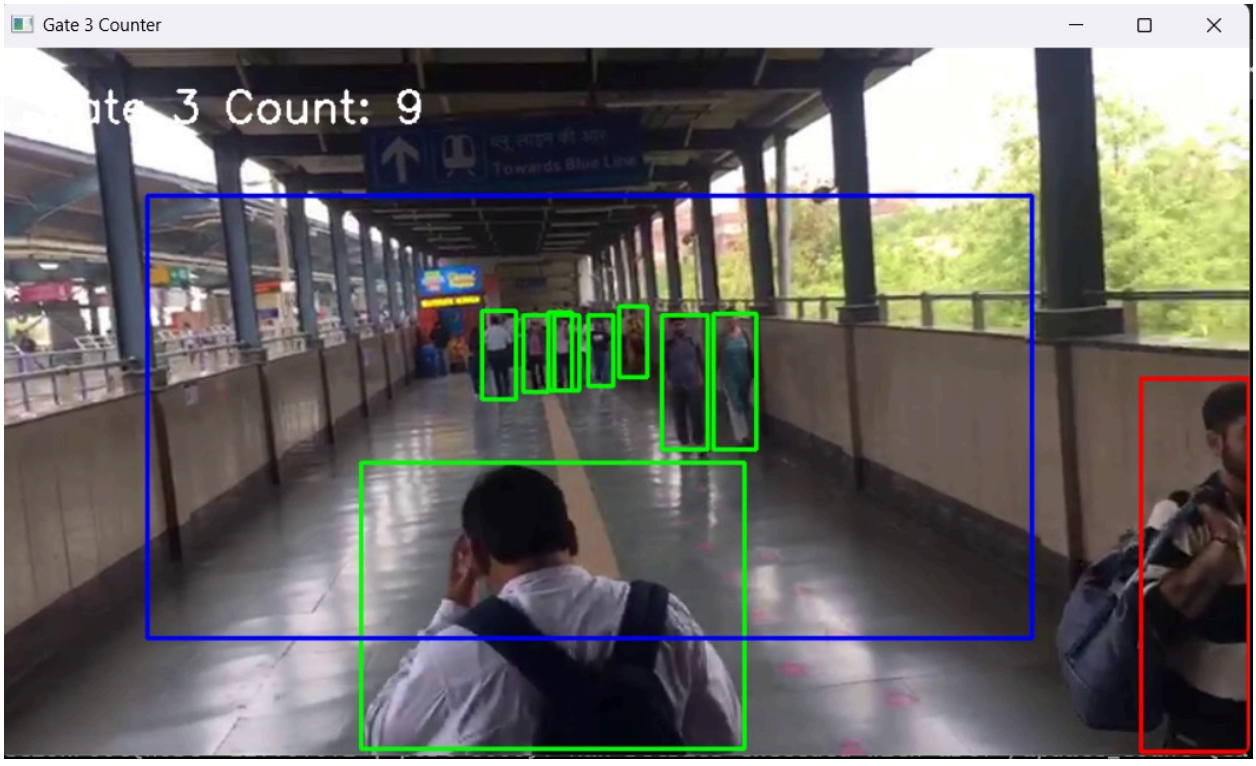
- Utilizes the user's live geolocation to figure out the closest gate with the least crowd.
- Retrieves gate information from Flask APIs.
- Plots a path from the user's location to the chosen gate.
- Shows a permanent overlay box on the map displaying live counts for every gate.

4.2.2. Key Implementation Components

1) Live Count Updates via YOLOv8 Integration:



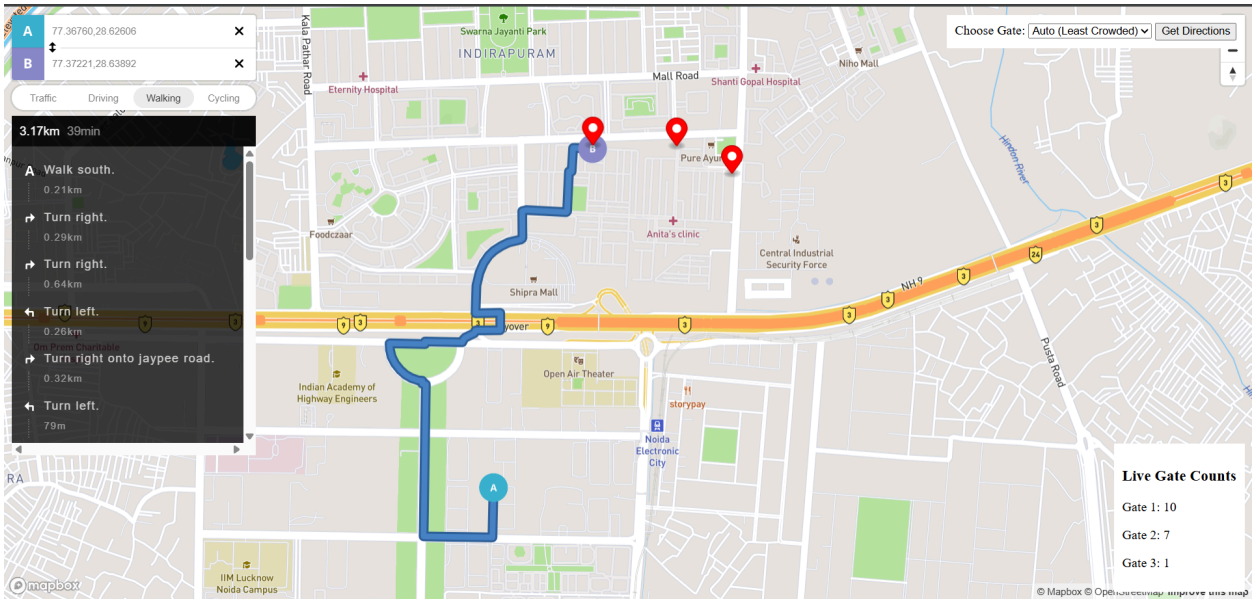


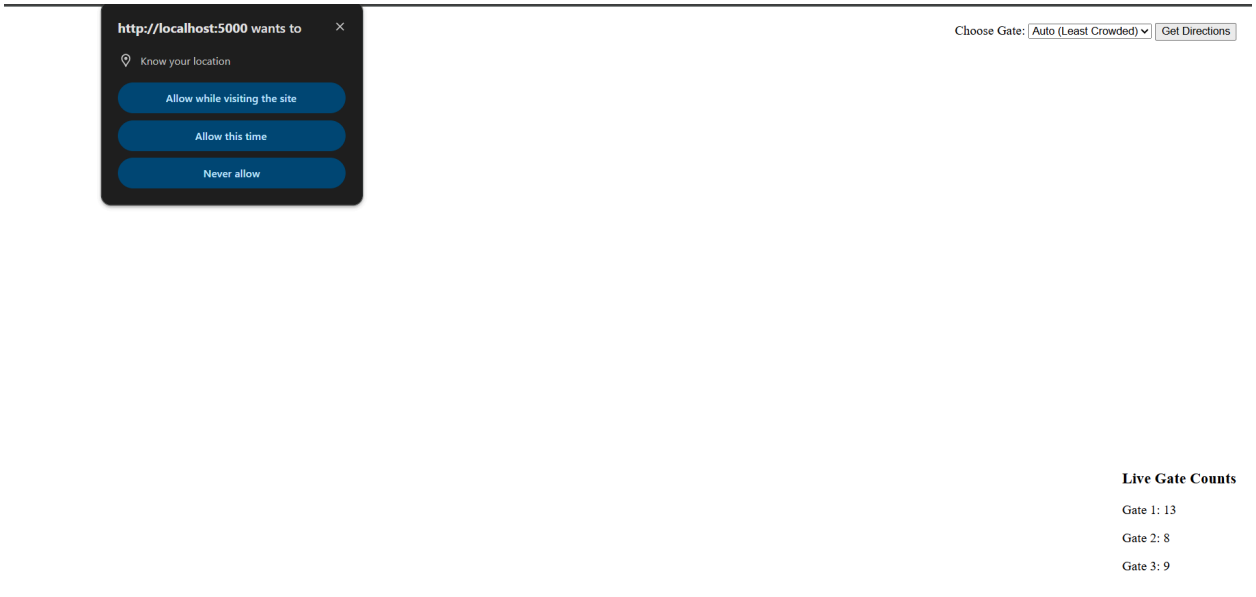


- A YOLOv8 script handles video streams and sends real-time person counts to the Flask backend via HTTP POST (/update_count) labeled by gate ID.

- The frontend invokes /get_counts every 3 seconds to show the latest counts.

2) Map Routing with Mapbox Directions API:

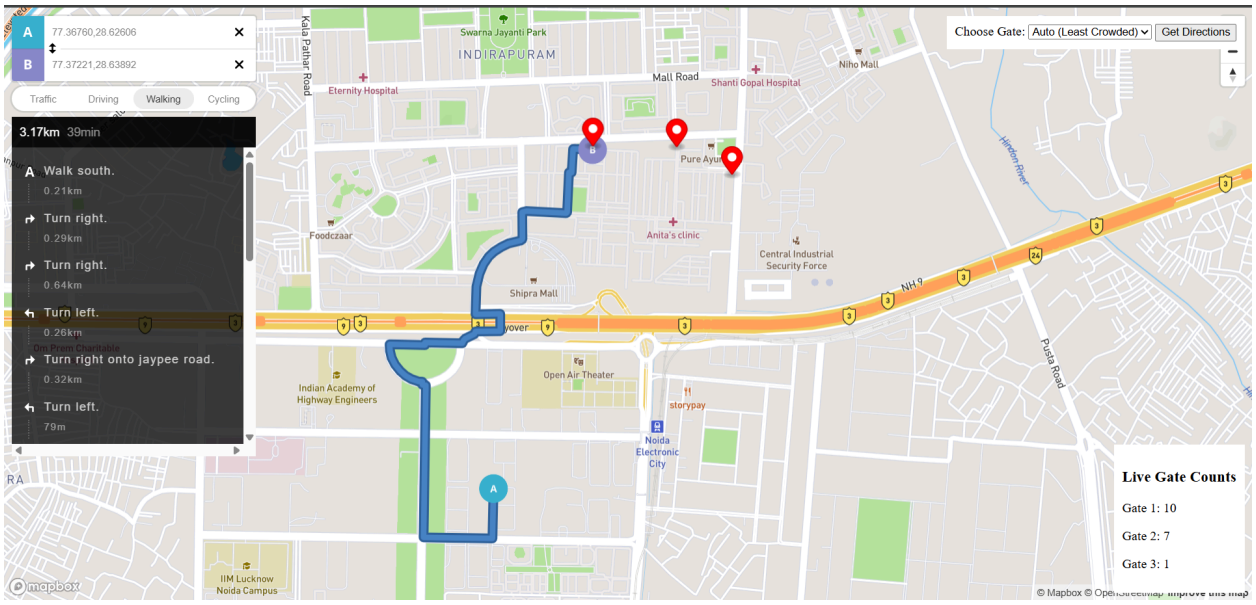




- On loading, the geolocation of the user is acquired through the browser's Geolocation API.
- The backend /get_gate_locations is called, returning the coordinates of all the gates.

- The lowest live gate is determined in JavaScript, and a walking route is plotted using Mapbox Directions API.

3) Dynamic UI Display:



- The fixed-position control shows the gate counts under the map interface.
- The values within the box are refreshed every 3 seconds with new report data coming from /get_counts.

4.2.3. Challenges and Solutions

1) **UI Overlay Not Showing Properly Over Map:**
   - **Issue:** The live gate count box was initially not appearing.
   - **Solution:** Ensured correct z-index and that the element was placed after the container in the HTML. Also confirmed no CSS conflicts.
2) **Live Counts Not Updating:**
   - **Issue:** The counts were stuck at "Loading...".
   - **Solution:** Fixed a mismatch in key names (gate1, gate2, gate3 vs numeric keys like 1, 2, 3 in the Flask gate_counts dictionary). Standardized to use string keys or mapped them on the frontend accordingly.
3) **Dynamic Routing Performance:**
   - **Issue:** On slower connections or if geolocation failed, the route wasn't shown.

- **Solution:** Added fallback error handling in navigator.geolocation and map setup logic. Also added console logging for debugging.

4) Path and Gate Count Misalignment:

- **Issue:** The path was visible on the map, but not the live count box.
- **Solution:** This was due to the asynchronous nature of script execution. Ensured the JS file was loaded with the defer attribute, and gate-counts were loaded after Mapbox map rendering to prevent visual conflicts.

5) Hardcoding of Gate Updates:

- **Limitation:** For now, gate updates are simulated and not connected to live camera feeds or YOLO-based counters.
- **Future Scope:** Integrate this with YOLOv8-based people counting pipelines via /update_count API.

4.3 RISK ANALYSIS AND MITIGATION

1)Risk: Failure or corruption of video feed
- Mitigation: OpenCV verifies ret on every frame; system bypasses processing if frame read fails.

2)Risk: Server unavailability or downtime
- Mitigation: try-except block in multi_gate_counter.py around requests.post() catches server errors and logs them without making the system crash.

3)Risk: Misdetection by model resulting in inaccurate people count
- Mitigation: Utilizing YOLOv8, a recent state-of-the-art object detection model, and specifying ROI (region of interest) to cull detections.

4)Risk: Gate IDs not consistent or missing data in requests
- Mitigation: Flask server (server.py) checks gate IDs and only updates defined gates.

5)Risk: Frontend not receiving data or map display errors
- Mitigation: The map updates gate data every 3 seconds with error logging in JavaScript (script.js) to alert the console if backend requests fail.

6)Risk: User location access denied
- Mitigation: JavaScript in script.js provides fallback and alert when geolocation is unavailable or permission is not granted.

7)Risk: Issues with multi-threading for video processing

- Mitigation: Each gate video is processed in a different thread with correct joining upon completion (multi_gate_counter.py), preventing blocking or race conditions.

8)Risk: Hardcoded paths or addresses that lead to deployment problems

- Mitigation: It can be better improved by relocating video paths and URLs into a config file or environment variables in production.

9)Risk: UI not being displayed or stale data

- Mitigation: HTML (f.html) employs JavaScript polling and responsive design to manage loading states and dynamic updates.

CHAPTER-5
TESTING

5.1 Testing Plan

The Crowd Management System was tested on three levels:
1)Unit Testing:

- Confirmed that video capture and YOLO-based detection produced anticipated outputs.
- Methods such as is_inside_area(cx, cy, rect) were tested using some sample coordinates.

2)Integration Testing:

- Confirmed that YOLO detection accurately sends count data over HTTP to the Flask backend.
- Confirmed that the frontend map correctly retrieves and displays data from the backend.

3) System Testing

- Activated the system using three test video feeds.
- Verified live counting from all gates, accurate data transmission, and mapping display.
- Verified robustness with server restarts, slow responses, and network failure.

5.2 Component Decomposition

1)multi_gate_counter.py

- Utilizes YOLOv8 to identify and count individuals in a given ROI per gate. Every gate is processed through a different thread for parallel processing. Pushes real-time information to the backend server.

2)server.py

- Flask backend server with /update_count and /get_counts endpoints. Stores and serves crowd data per gate. Can test-update when run standalone.

3)script.js

- Front-end JavaScript with Mapbox to pull live crowd information and dynamically relocate gate markers. Discovers and routes users through the least crowded gate while displaying route optimizations.

4)f.html

- The system's UI structure, combining the map and styled gate information boxes. Invokes script.js for dynamic functionality.

5.3 Test Result List:-

Test Case	Description	Expected Result	Actual Result	Status
TC1	Run YOLO on test video feed	Detects people in ROI	Correct displayed feed	Pass
TC2	Server receives count data	Count appears in Flask log	Log and memory updated correctly	Pass
TC3	Map displays live gate data	Count info update every 3 sec	Data Reflected accurately	Pass
TC4	Gate with least crowd highlighted	Smallest count display as “Best Gate”	Correct gate highlighted	Pass
TC5	Multiple gate threads running	Counts posted concurrently	All threads work in parallel	Pass

5.4 Error and Exception Handling

The project includes several key exception-handling mechanisms:

1)Network Errors in multi_gate_counter.py:Logs the issue but continues running, ensuring system resilience.

```
try:
    requests.post(SERVER_URL, json={'gate_id': gate_id, 'count': people_count})
except Exception as e:
    print(f"[Gate {gate_id}] Server Error:", e)

cv2.imshow(f"Gate {gate_id} Counter", frame)
if cv2.waitKey(1) & 0xFF == ord('q'):
    break
```

2)Invalid Video Feeds:

- Handled by checking ret in OpenCV before processing each frame.

- If the video feed is missing or corrupt, it safely exits the loop instead of throwing an exception.

```
while True:
    ret, frame = cap.read()
    if not ret:
        break
```

3)Flask Server:

- Includes basic validation for gate IDs and count fields in requests.
- If the data is incomplete, it responds with a bad request and an error message.

```
@app.route('/update
def update_count():
    data = request.
    gate_id = data.get('gate_id')
    count = data.get('count')
    if gate_id in gate_counts:
        gate_counts[gate_id] = count
    return jsonify(success=True)
```

(function) get: Any

See Real World Examples From GitHub

4) Missing or Corrupted Geolocation:

- Handles location retrieval errors with a fallback error handler **(errorLocation** function).

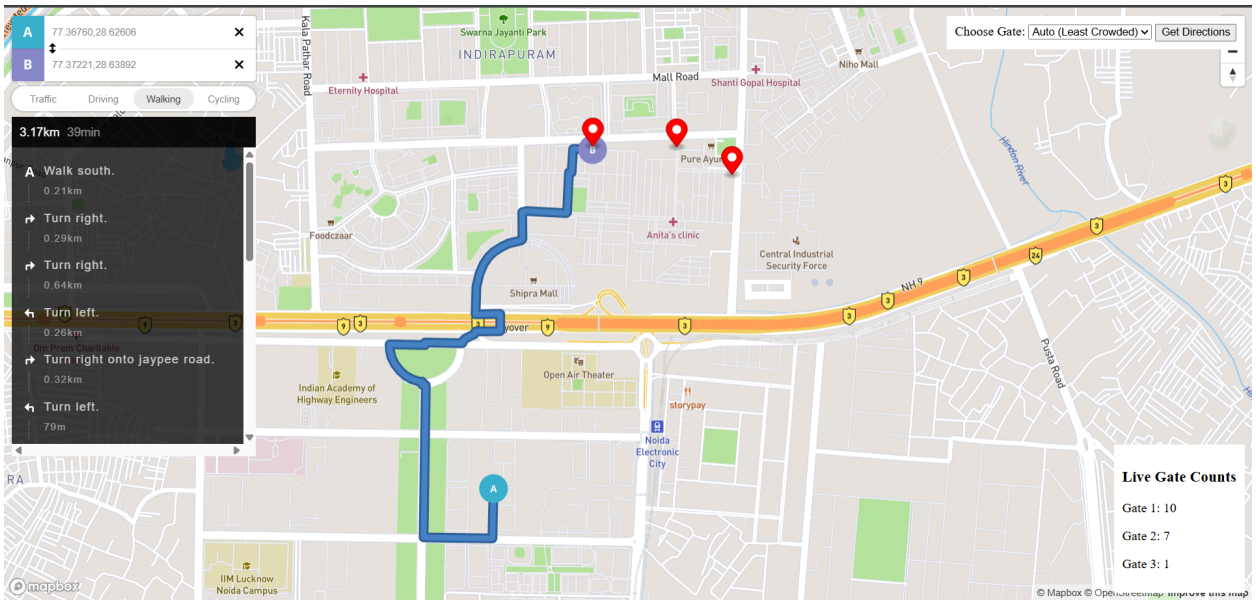
```
function successLocation(position) {
  const userCoords = [position.coords.longitude, position.coords.latitude];
  fetchBestGateAndShowRoute(userCoords);
}

function errorLocation() {
  console.error("Geolocation failed.");
}
```

5.5 Limitations of the Solution

1)No Persistent Storage: Count data is stored in-memory on the server. A restart loses all data.
2)Hardcoded Video and ROI Settings: Paths to video files and ROI regions are fixed per gate.
3)No Authentication or Security: API endpoints are open and unprotected.
4)Model Limited to Class 0: YOLO is only detecting humans (class 0); misclassifications may still occur.
5)Not Scalable: Built for three gates with hard-coded logic. Would need to be redesigned for 10+ gates.
6)No Historical Data or Analytics: Live crowd counts only are displayed—no logging or trends are kept.

Final output:



CHAPTER- 6

6.1 FINDINGS

1. Accuracy in Real-Time Detection
Utilization of AI models such as YOLOv8 accurately detected people with high precision, allowing dynamic crowd density estimation per gate in real-time.

2.Effective Load Distribution
The decision engine of the system accurately computed data through every gate and suggested the least crowded ones, significantly minimizing peak-hour crowding.

3. Improvement in User Guidance
Integration with Mapbox and geolocation facilitated personalized, real-time directions to recommended gates, greatly improving user experience—particularly for first-timers or in vast areas.

4. Automation Minimized Manual Intervention
The project demonstrated that conventional manual monitoring and intervention could be dramatically reduced without reducing safety or efficiency.

5. Responsive and Scalable Interface
The web interface was responsive under concurrent user requests and scalable to different institutions beyond the original use case.

6. Security and Privacy Compliance
Through effective anonymization and secure data transfer, the system maintained user privacy, even when dealing with live video streams and location information.

6.2 CONCLUSION

The Smart Crowd Management System developed here presents an intelligent, scalable, and user-friendly answer to an increasingly critical need in institutional crowd management. Through the application of artificial intelligence, real-time video analysis, and geolocation-based route suggestions, the system guarantees:

- Improved Safety through preventing overcrowding and congestion.
- Greater Efficiency via detection and decision-making automation.
- Enhanced User Experience by way of data-driven, real-time navigation aids.

Combining several technologies in the form of YOLOv8 as a detection framework, a backend for centralized hosting, and web-based frontend infrastructure makes this solution not just technology-wise solid, but also extensible across the range of high-traffic environments like arenas, shopping centers, or stations.

This solution makes a valuable contribution to smart campus and smart city projects, providing a template for data-driven, real-time public safety and convenience infrastructure systems.

6.3 FUTURE WORK

For further development and enhancement of this system, the future work could include the following aspects:

1. Behavior Prediction Models
Implement predictive algorithms based on historical data to predict peak hours and crowd flow patterns, enabling even earlier intervention.

2. Mobile App Development
Expand the system to mobile devices for easier access and real-time push alerts.

3. Multi-Story and Indoor Navigation
Add indoor location systems (e.g., Bluetooth beacons) for multi-story building navigation.

4. Integration with Disaster Services
Provide automatic alerts and emergency rerouting in the event of fire, stampede danger, or other threats.

5. Edge Computing for Speedy Processing
Shift portions of AI computing to the camera (edge devices) to minimize latency and reliance on central servers.

6. Crowd Emotion and Movement Analysis
Investigate AI models that can identify abnormal behavior, panic, or aggression for predictive security interventions.

7. Multilingual Interface and Accessibility Features
Use multilingual capabilities and voice control to support users from all backgrounds and with diverse abilities.

REFERENCES

RESEARCH PAPER:

[1]H. H. Pothireddy, R. GV, E. Alladi, N. Chaudhary, and A. A., “A Study Into The Aspect Of Crowd Management,” International Journal of Research Publication and Reviews, vol. 6, no. 3, pp. 8707–8715, Mar. 2025, doi: https://doi.org/10.55248/gengpi.6.0325.12181.

[2] Yi, S., H. Li, and X. Wang, Pedestrian behavior modeling from stationary crowds with applications to intelligent surveillance. IEEE transactions on image processing, 2016. 25(9): p. 4354-4368.

[3]Chaker, R., Z. Al Aghbari, and I.N. Junejo, Social network model for crowd anomaly detection and localization. Pattern Recognition, 2017. 61: p. 266-281.

[4] Aveni, A.F., The not-so-lonely crowd: Friendship groups in collective behavior. Sociometry, 1977: p. 96-99.

[5]“PlantUML Web Server,” PlantUML.com, 2025. http://www.plantuml.com/plantuml/png/ (accessed May 06, 2025).

[6]Lu, L., et al., A study of pedestrian group behaviors in crowd evacuation based on an extended floor field cellular automaton model. Transportation research part C: emerging technologies, 2017. 81: p. 317-329.

[7] Liu, Z., et al., Decision-Making Framework for GI Layout Considering Site Suitability and Weighted Multi-Function Effectiveness: A Case Study in Beijing Sub-Center. Water, 2022. 14(11): p. 1765.

[8] Singh, U., et al., Crowd monitoring: State-of-the-art and future directions. IETE Technical Review, 2021. 38(6): p. 578-594.]