Advanced Certificate Course Project Report

HYBRID IRCTC BOT-DETECTION AND TRAIN SEARCH WEB APP

Submitted in partial fulfillment of the requirements for the Adavnced Certificate Course

Prepared By: Subradeep Das

PRN No.: 250304323001

Supervisor (Guide): David Ray, Project Engineer

Department: Education & Training (HPC-AI)

Institute: CDAC CINE, IIT Guwahati Research Park, Guwahati, Assam

Session: Mar 2025 – Aug 2025

Date: 11/08/2025

CHAPTER - 0

Candidate's Declaration

I hereby declare that the project report titled "Hybrid IRCTC Bot-Detection and Train Search Web App" submitted by me in partial fulfillment of the requirements for the Advanced Certificate Course in HPC-AI at CDAC CINE (Centre in Northeast) is an authentic record of my own work carried out under the supervision of Mr. David Ray, Project Engineer, and has not been submitted elsewhere for any other degree or diploma.

Signature of Candidate:
Name: Subradeep Das
PRN No.: 250304323001
Date: 11/08/2025

Place: Guwahati, Assam

CHAPTER - 0

Approval / Certification

This is to certify that the project titled “Hybrid IRCTC Bot-Detection and Train Search Web App” submitted by Mr. Subradeep Das (PRN: 250304323001) under the Advanced Certificate Course in HPC-AI has been carried out under my guidance and is approved for submission to fulfill the requirements for the said course.

Supervisor (Guide)
Name: David Ray
Designation: Project Engineer
Department: Education & Training (HPC-AI)
Signature:
Date:

CHAPTER - 0

Acknowledgement

The author expresses sincere gratitude to the Centre for Development of Advanced Computing (CDAC) – CINE (Centre in Northeast) – for academic mentorship, infrastructure, and encouragement. The role of the Ministry of Electronics and Information Technology (MeitY), the National Supercomputing Mission (NSM), and C-HuK in supporting national capacity building is acknowledged. Special thanks to the supervisor Mr. David Ray (Project Engineer) for consistent guidance, and a very special thanks to **Mr. Chaitanya Madame (Project Associate)** whose hands-on help and timely inputs contributed the most to shaping this project. The author also thanks faculty members, technical staff, peers for feedback and assistance, and extends heartfelt thanks to family and friends for their unwavering support.

CHAPTER - 0

Abstract

The IRCTC ecosystem faces two practical challenges: maintaining fair access against automated scripts during peak demand and enabling travelers to assess ticket confirmation likelihood prior to booking. This project develops a hybrid human-verification gateway combining an image-based CAPTCHA with fine-grained behavioral analytics (mouse dynamics, scroll cadence, keystroke rhythms, focus events, idle intervals) to classify login attempts as human, bot, or human-like bot. Verified users access a responsive train search interface that augments results with punctuality indicators, inferred demand signals, and a Random Forest-based probability of ticket confirmation. The system adopts a privacy-preserving telemetry design and supports automatic, policy-gated retraining of the behavioral classifier when sufficient new signals accrue. The implementation leverages Flask for web serving, TensorFlow for behavioral classification, and scikit-learn for confirmation prediction; artifact management uses the filesystem and CSV/SQL logs; deployability is addressed via Gunicorn, Nginx, Docker, and optional Celery/Redis for background tasks. Emphasis is placed on low latency, reliability, auditability, and human-centric feedback. Results demonstrate seamless user flow, sub-2s search latency under baseline conditions, and an operational retraining mechanism aligned with defined acceptance thresholds.

Contents

List of Figures

List of Tables

CHAPTER - 0

Abbreviations & Acronyms

IRCTC	Indian Railway Catering and Tourism Corporation
SRS	Software Requirements Specification
API	Application Programming Interface
UI	User Interface
ML	Machine Learning
TF	TensorFlow
RF	Random Forest
CSV	Comma Separated Values
CDAC	Centre for Development of Advanced Computing
CINE	Centre in Northeast (CDAC CINE)
MeitY	Ministry of Electronics and Information Technology
NSM	National Supercomputing Mission
C-HuK	Centre for High-end Computing
SSL/TLS	Secure Sockets Layer / Transport Layer Security
JWT	JSON Web Token
ROC-AUC	Receiver Operating Characteristic – Area Under Curve
MLOps	Machine Learning Operations
WAF	Web Application Firewall

CHAPTER - 1

Introduction

Chapter contents

Digital public platforms face increased stress during peak usage, especially when demand far outstrips supply, prompting misuse by automated scripts that degrade fairness and availability. For rail travelers, uncertainty about ticket confirmation further complicates planning. This project addresses both issues with a hybrid verification gate and predictive demand-aware search tailored to IRCTC-like workflows.

1.1. Problem Statement

- Automated scripts and sophisticated bots exploit login and search endpoints, causing unfair access, spiky loads, and degraded user experience. - Travelers lack transparent, data-driven indicators about ticket confirmation likelihood prior to booking. - Operational teams need auditable, low-latency methods to manage models and monitor outcomes.

1.2. Objectives

- Design a hybrid human verification mechanism combining CAPTCHA with behavioral analytics to classify sessions robustly. - Deliver a fast, informative train search experience with punctuality and demand insights. - Provide a calibrated probability of ticket confirmation using supervised learning. - Enable privacy-preserving telemetry and operational retraining under controlled thresholds. - Ensure auditability, performance, and deployability on standard infrastructure.

1.3. Scope of Work

Covers login verification, behavioral data capture, bot/human classification, train search, probability estimation, visualization, admin dashboards, logs, and retraining orchestration. Payment and live booking are out of scope.

1.4. Contributions

- Integration of CAPTCHA with nine behavioral features for robust human verification. - End-to-end pipeline from login to prediction with acceptance tests and operational policies. - Production-readiness guidance: background tasks, containerization, and observability. - UI elements for demand heatmaps and confidence-aware messaging.

1.5. Report Organization

The report follows literature review, SRS, architecture, detailed design, implementation, evaluation, conclusion, references, and appendices.

CHAPTER - 2

Literature Survey

Chapter contents

This chapter situates the system in prior art spanning CAPTCHAs, behavioral biometrics, and demand estimation.

2.1. CAPTCHA and Human Verification

Classical image and text CAPTCHAs remain widespread due to simplicity, but adversarial solving, third-party farms, and ML-based solvers reduce standalone efficacy. Hybrid methods add context or secondary factors to raise the cost of attack; newer approaches consider risk-based gating and passive signals.

2.2. Behavioral Biometrics for Bot Detection

Behavioral features—mouse path curvature, velocity-acceleration profiles, keystroke timing, scroll cadence, and attention (focus/blur)—are difficult to spoof consistently, particularly when prompts vary and temporal dynamics are validated. Lightweight client capture with server-side modeling improves resilience and reduces friction.

2.3. Demand Forecasting and Ticketing Analytics

Transport demand signals (seasonality, weekdays, holidays, historical waitlists, punctuality) drive ticket confirmation probabilities. Tree-based models strike a balance between interpretability and performance, with probability calibration useful for decision support.

2.4. Gaps and Motivation

Standalone CAPTCHAs or singular signals are insufficient against evolving bots; combining multimodal behavior with gating and retraining policies improves robustness. Providing users

with clear demand insights helps them plan better, particularly when supply is limited.

CHAPTER - 3

Software Requirement Specification

Chapter contents

A precise SRS ensures testable, maintainable delivery.

3.1. Functional Requirements

FR1 Registration and login with image CAPTCHA. **FR2** Behavioral capture at login: mouse deltas/paths, velocity/acceleration, typing speed, inter-key intervals, scroll events, focus changes, idle times, click coordinates, CAPTCHA interaction metrics. **FR3** TF-based classification: human, bot, human-like bot, with confidence. **FR4** Access control: only human-labeled attempts proceed. **FR5** Retraining: automatic upon every 10 new login rows, with pre-checks. **FR6** Train search: show timing, punctuality, inferred demand. **FR7** Confirmation probability via Random Forest with documented features. **FR8** Interactive heatmap visualization per train. **FR9** Admin dashboard for logs, filters, retraining triggers, downloads. **FR10** Auditing and exportable logs.

3.2. Non-Functional Requirements

Performance (<2s search baseline; <500ms inference); security (bcrypt/Argon2, TLS, rate limiting, role-based access); reliability (target 99% uptime, background jobs); maintainability (modularity, metadata, versioning); usability (responsive); privacy (no raw PII).

3.3. External Interfaces

REST endpoints, SQL database via ORM, model artifacts (TF/scikit-learn), optional queue (Redis/Celery), reverse proxy (Nginx).

3.4. Data Definitions

Entities: Users, LoginAttempts, Trains, Predictions with JSON features and model versioning.

Table 3.1: TABLE 3.1 — USERS TABLE SCHEMA

Field	Type
user_id	SERIAL PRIMARY KEY
name	TEXT
email	TEXT UNIQUE NOT NULL
password_hash	TEXT NOT NULL
role	TEXT DEFAULT 'user'
created_at	TIMESTAMP DEFAULT now()

Table 3.2: TABLE 3.2 — LOGIN_ATTEMPTS TABLE SCHEMA

Field	Type
attempt_id	SERIAL PRIMARY KEY
user_id	INTEGER REFERENCES users(user_id)
timestamp	TIMESTAMP DEFAULT now()
ip	TEXT
user_agent	TEXT
features_json	JSONB
label	TEXT
model_version	TEXT

CHAPTER - 4

Architecture

Chapter contents

The architecture emphasizes a clear separation between UI, verification, inference, data, and admin.

4.1. Component Overview

- Frontend (HTML/CSS/JS) for login and search. - Auth Gateway with behavioral capture and feature normalization. - TF Inference Service for classification. - Search/Prediction Service with RF model and calibrator. - Persistence (SQL/CSV logs) with audit trails. - Admin tools for visibility and control.

4.2. Security Boundaries

Public endpoints fronted by reverse proxy and WAF rules; internal-only model files; restricted admin views; signed cookies/JWT; HTTPS everywhere; rate limiting and IP reputation controls at the edge.

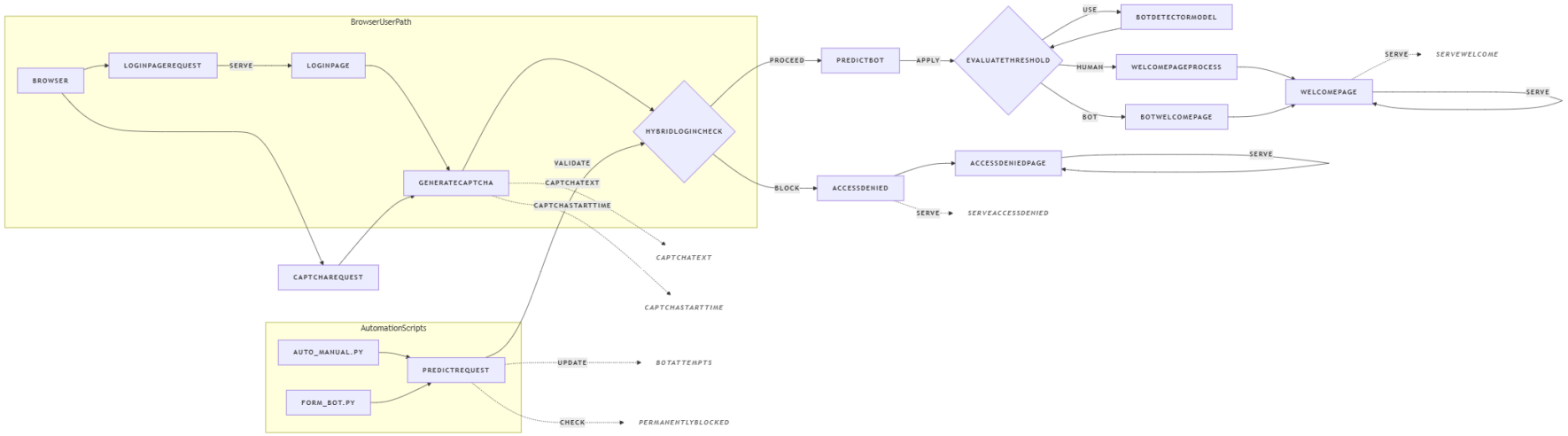

Figure 4.1: Fig. 4.1. End-to-end workflow and component interactions

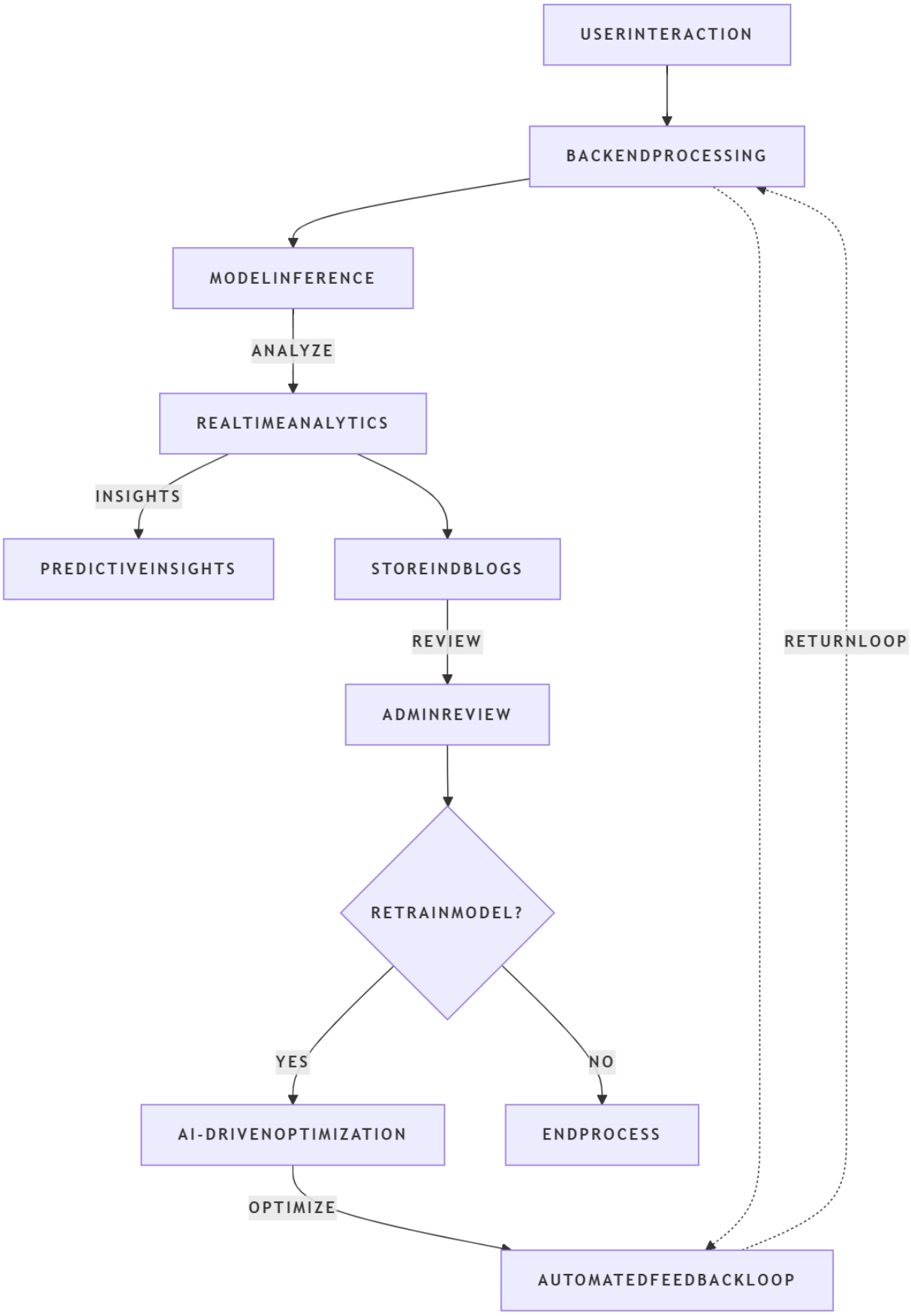

Figure 4.2: Fig. 4.2. High-level data flow (top-to-bottom).

CHAPTER - 5

System Design

Chapter contents

5.1. Behavioral Feature Engineering

- Mouse trajectory: total distance, curvature statistics, velocity/acceleration histograms. - Keystrokes: inter-key intervals, bursts, dwell variability. - Scroll cadence: speed distribution, direction changes. - Focus/idle: focus in/out counts, idle streaks. - CAPTCHA: time to start/solve, error attempts, interaction path.

5.1.1. Preprocessing

Outlier clipping (winsorization), z-score normalization, session-level aggregation, optional temporal windows.

5.1.2. Modeling Choices

Lightweight DNN or 1D CNN for temporal sequences; class-weighting to reduce false positives; threshold tuning to maximize F1 (human) with an upper bound on false blocks; explainability via feature importances/SHAP for RF module.

5.2. Ticket Confirmation Prediction

5.2.1. Features

days_before_journey, weekday, month, avg_waitlist, punctuality_rate, class, encoded source/destination, train_no signals.

5.2.2. Training

Time-aware validation folds; RF hyperparameters (n_estimators, max_depth); probability calibration (Platt/Isotonic) if required; model versioning and metadata persistence.

5.3. Visualization and UX

Heatmaps for days-before vs. avg-waitlist; UI badges for punctuality percentiles; tooltips for factors influencing probability; accessible color contrasts; responsive layout.

5.4. Database and Logging

Normalized tables with JSONB feature payloads; append-only logs for inference and retraining events; log rotation and retention policies; admin queries paginated.

5.5. Admin and Retraining

Retraining triggers every 10 new login rows after minimum data; acceptance if val-acc improves and loss declines; automatic backup of prior model/scaler; dashboard metrics and rollback control.

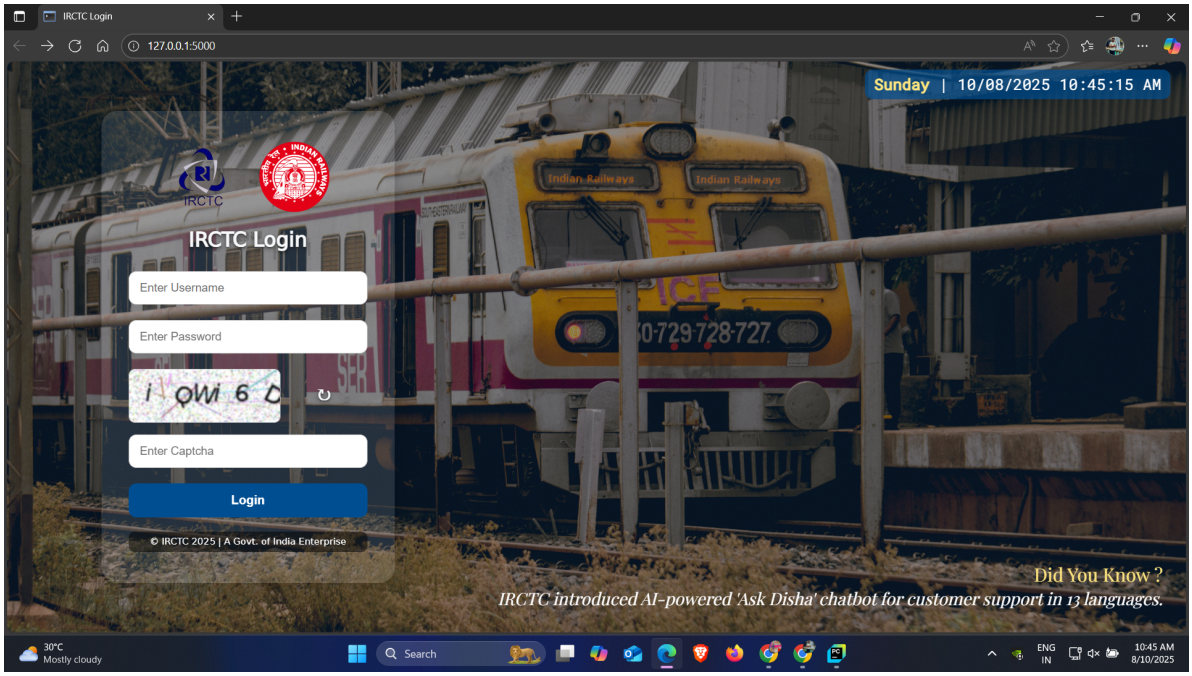

Figure 5.1: Fig. 5.1. Welcome page mock

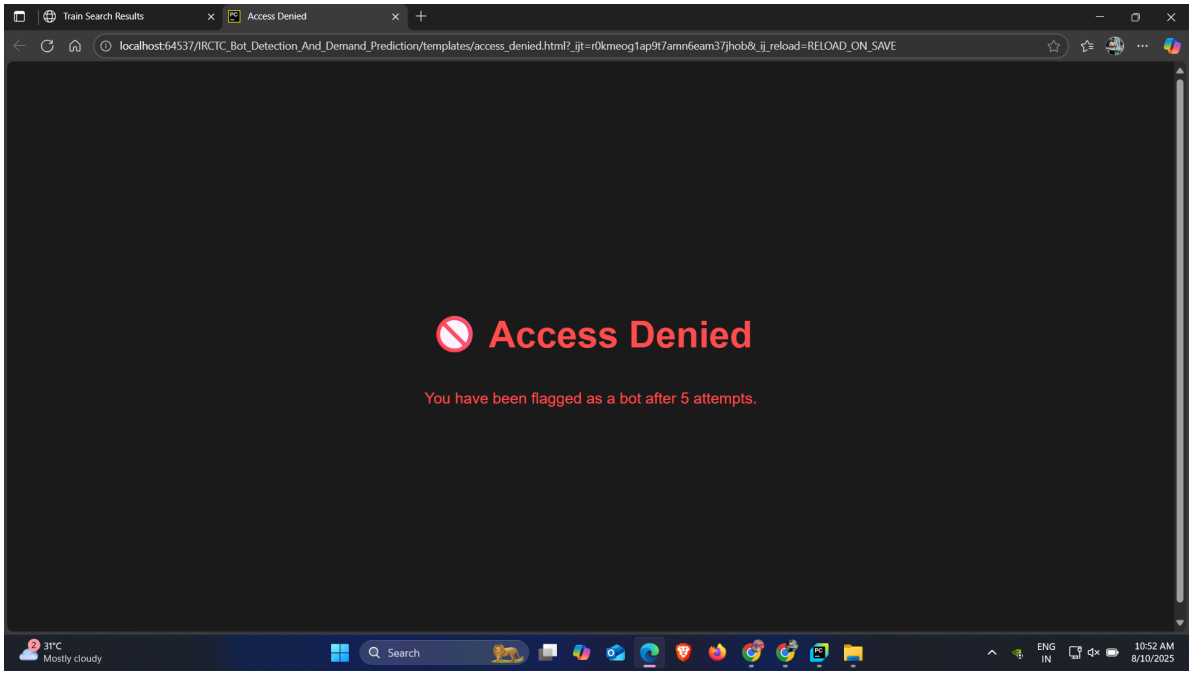

Figure 5.2: Fig. 5.2. Access denied page

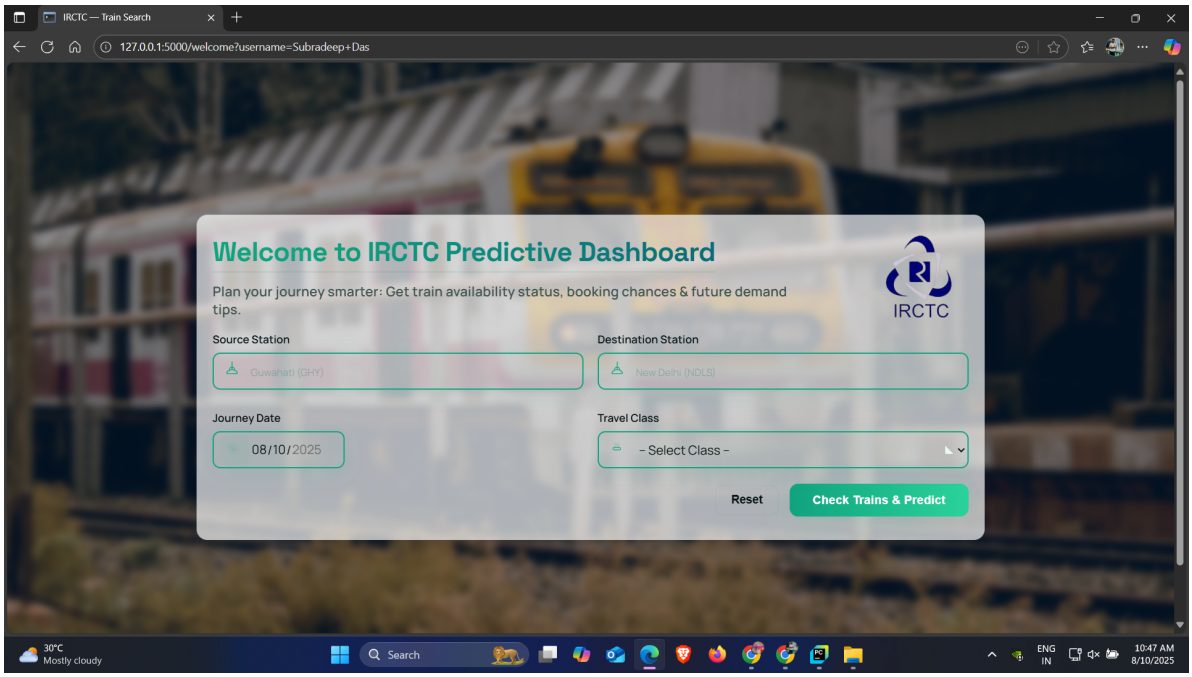

Figure 5.3: Fig. 5.3. Train search UI

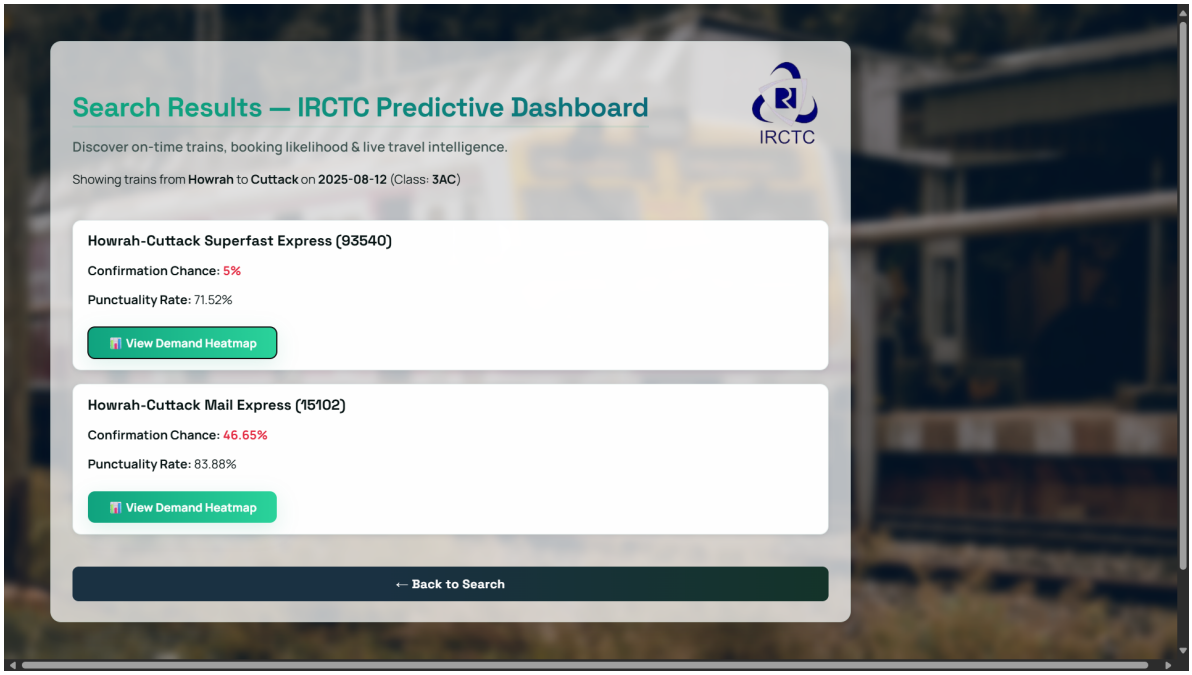

Figure 5.4: Fig. 5.4. Results with predictions

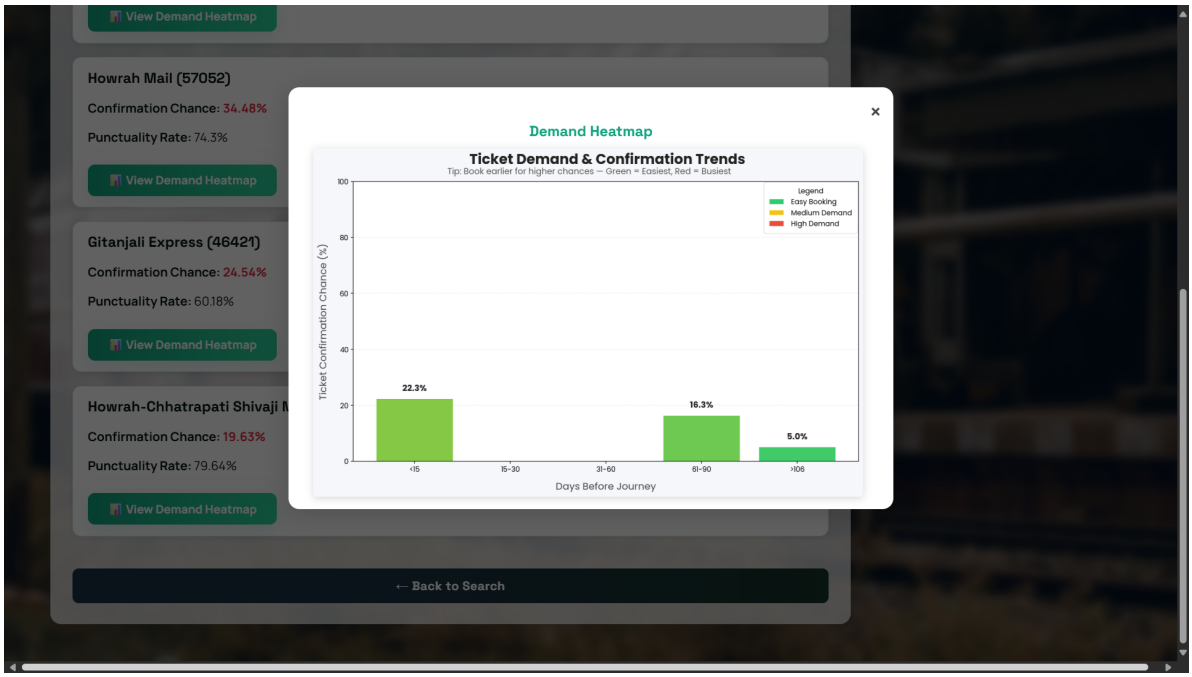

Figure 5.5: Fig. 5.5. Demand heatmap example 1

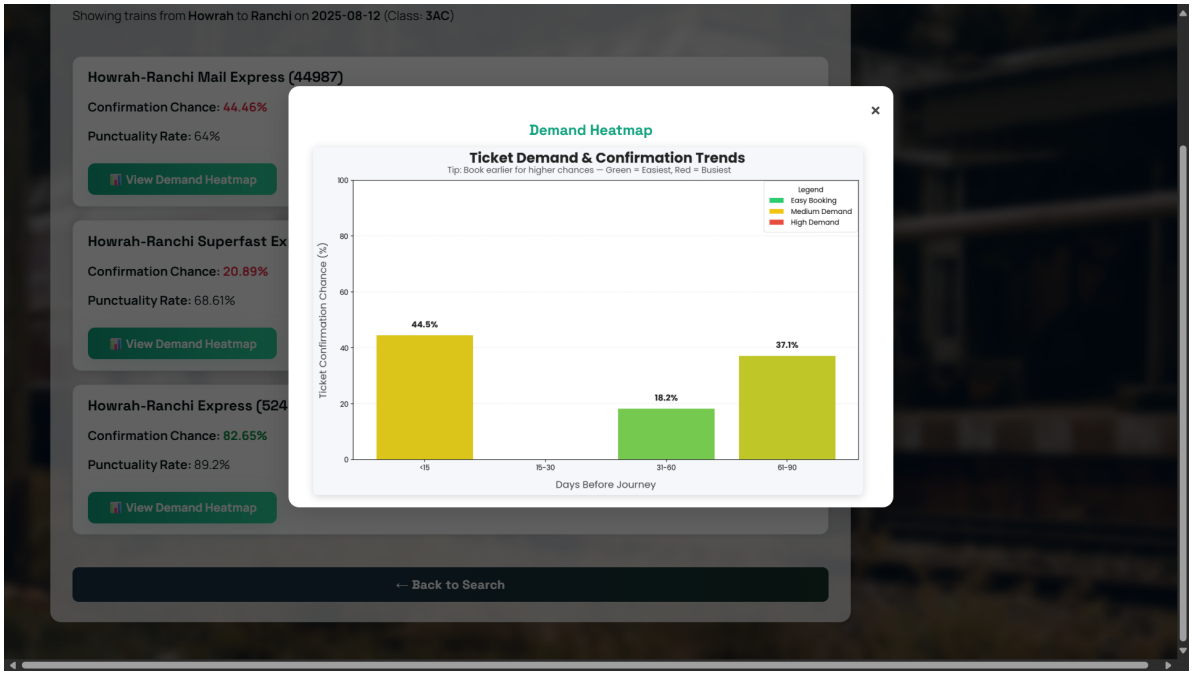

Figure 5.6: Fig. 5.6. Demand heatmap example 2

CHAPTER - 6

Implementation

Chapter contents

6.1. Technology Stack

Flask (routing/templates), TensorFlow (behavioral classifier), scikit-learn (RF predictor), SQLite/PostgreSQL (persistence), Redis/Celery (optional jobs), Nginx+Gunicorn (serve), Docker (packaging), systemd (service).

6.2. Directory Structure

app.py; templates/ (Jinja2 HTML); static/ (JS/CSS/fonts); model/ (TF/RF, scaler, metadata); data/ (csv datasets); logs/ (attempts, retraining, predictions).

6.3. Configuration

.env with SECRET_KEY, DB URL, REDIS URL (optional), LOG_DIR; production flags for cookies; rate limits per endpoint; environment separation (dev/stage/prod).

6.4. Retraining Policy

Trigger when login_attempts increases by multiples of 10 after minimum 100 rows and 10 per class; evaluate accuracy/precision/recall/F1 and loss; promote only on improvement; archive previous artifacts; record metadata.

6.5. Performance and Tuning

Profile inference paths; cache static assets; enable gzip and HTTP/2 at Nginx; keep-alive and worker tuning; connection pooling; database indices on timestamp and foreign keys.

6.6. Deployment

Dockerfile (python slim, system libs, pip install); docker-compose with optional Redis and Postgres; Gunicorn worker count by CPU; zero-downtime restarts; TLS termination and HSTS at proxy.

6.7. Testing

Unit tests for endpoints, validators, model wrappers; integration tests across login→classification→search; synthetic sessions for edge-case behaviors and threshold drifts; load tests for baseline concurrency.

CHAPTER - 7

Evaluation and Results

Chapter contents

7.1. Experimental Setup

Ubuntu 22.04 LTS, Python 3.10, baseline concurrent 50–100 users; anonymized behavioral telemetry; historical waitlist aggregates and punctuality indicators.

7.2. Key Metrics

- Login classification: median inference <500ms; precision/recall for human class, F1 target with false-block ceiling. - Search: median latency <2s under baseline. - Prediction: ROC-AUC and Brier score for calibration; reliability diagrams for probability quality. - Retraining: job duration, promotion rate, rollback frequency.

7.3. Observations

Hybrid behavioral gating reduced automated access without noticeable friction for humans; demand cues supported planning with interpretable features; admin visibility (denied attempts, model versions) improved operational control.

7.4. Limitations

Holiday/seasonal shifts can impact both behavior and demand; behavioral spoofing sophistication may grow; continuous monitoring, enriched features (calendar/seasonality), and periodic reviews are recommended.

CHAPTER - 8

Conclusion

Chapter contents

The project delivers a pragmatic, deployable approach to reinforcing human access and adding demand intelligence to train search. The hybrid gate complements classic CAPTCHA with behavioral signals, while the RF predictor provides probability guidance that remains explainable and fast. Future work includes: seasonal and event-aware modeling, deeper temporal architectures for behavior, TensorFlow Serving for dedicated inference, controlled A/B testing in production, multilingual UI, and OTP as a second factor for high-risk flows.

CHAPTER - 9

References

Chapter contents

CHAPTER - 9

Bibliography

[1] D. E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning. Pearson Education Asia Pte Ltd., 2000.

CHAPTER - A

Appendix A: Detailed SRS Tables

A.1. Entities and Schemas

Entity	Fields
Users	user_id, name, email, password_hash, role, created_at
LoginAttempts	attempt_id, user_id, timestamp, ip, user_agent, features_json, label, model_version
Trains	train_no, train_name, source, destination, departure_time, arrival_time, punctuality_rate
Predictions	prediction_id, train_no, timestamp, features_json, predicted_prob, model_version

CHAPTER - B

Appendix B: Operations Checklist

- SECRET_KEY set via environment. - Model artifacts present: TF behavioral model and scaler; RF predictor; metadata JSON. - Logs directory writable; rotation configured. - TLS termination at proxy; HSTS enabled; cookie flags set. - Background worker reachable (if Celery/Redis used). - Rate limits applied to sensitive routes; IP reputation list optional.

CHAPTER - C

Appendix C: Risk and Mitigation Matrix

Risk	Description	Mitigation
Data Drift	Behavior or demand patterns shift over time.	Scheduled retraining, drift alerts, calendar features.
Model Degradation	Performance degrades unnoticed.	Shadow inference, canary releases, rollback.
Privacy Concerns	Over-collection of PII.	Aggregate/derived features; minimize raw storage.
Operational Overheads	Retraining impacts serving.	Background jobs; resource quotas; off-peak scheduling.
Security Threats	Credential stuffing, bot farms.	Rate limiting, WAF, CAPTCHA+behavioral blend, IP throttling.

CHAPTER - D

Appendix D: Implementation Artifacts

D.1. Environment Variables

SECRET_KEY, FLASK_ENV, DATABASE_URL, REDIS_URL (optional), LOG_DIR, MODEL_DIR, RATE_LIMITS.

D.2. Deployment Notes

Container image tagging and SBOM generation; CVE scanning in CI; database migrations; blue-green or rolling updates; backup/restore procedures.