# Railway Deadlock Simulation Using Banker's Algorithm

Simulating railway deadlocks to ensure safe track management.

By Yash Amte, Akshay Pise, Rohit Bharti, Prathamesh Dhomne

# Introduction to Deadlock in Railways

## Deadlock Concept

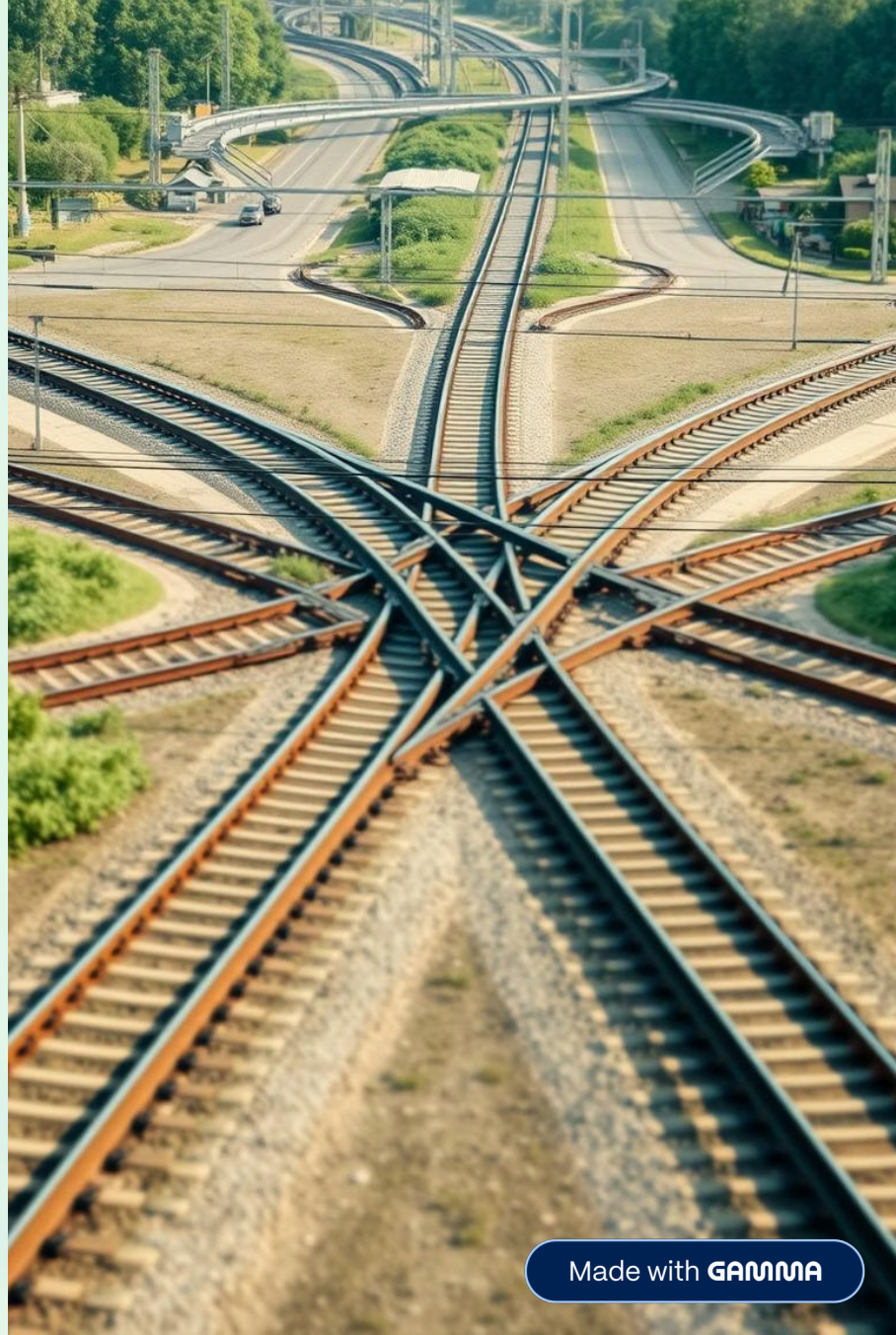
Blocking where trains wait indefinitely for tracks

## Causes

Multiple trains requiring shared tracks simultaneously

## Impact

Delays, safety hazards, inefficiencies



# Banker's Algorithm Overview

## Purpose

Detect and avoid deadlocks by checking safe states

## Core Idea

Ensure each resource allocation keeps system safe

## Application

Allocating tracks without causing deadlocks

# System Design & Implementation

## C Language

Core logic of Banker's algorithm for track allocation

## Python HTTP Server

Backend CGI server handles user input requests

## HTML & CSS

Frontend interface for train and track input

# User Interaction and Input

## Input Parameters

- Number of trains
- Number of tracks
- Track allocation requests

## Process

User submits data to server for simulation

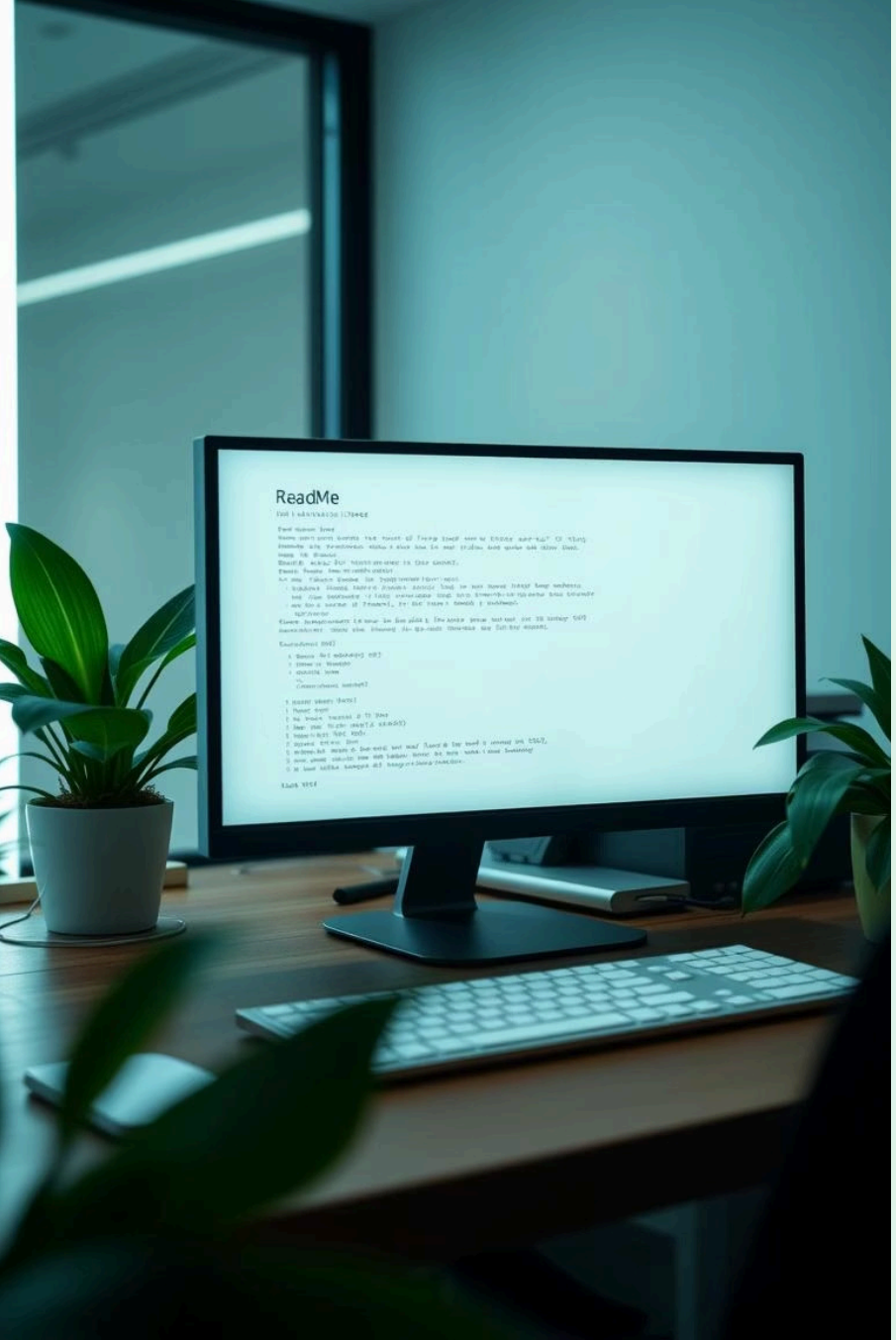
# Simulation Output & Results

### Safe State

Tracks allocated without causing deadlock

### Unsafe State

Potential deadlock detected, requires action




# Additional Functionalities

### README File

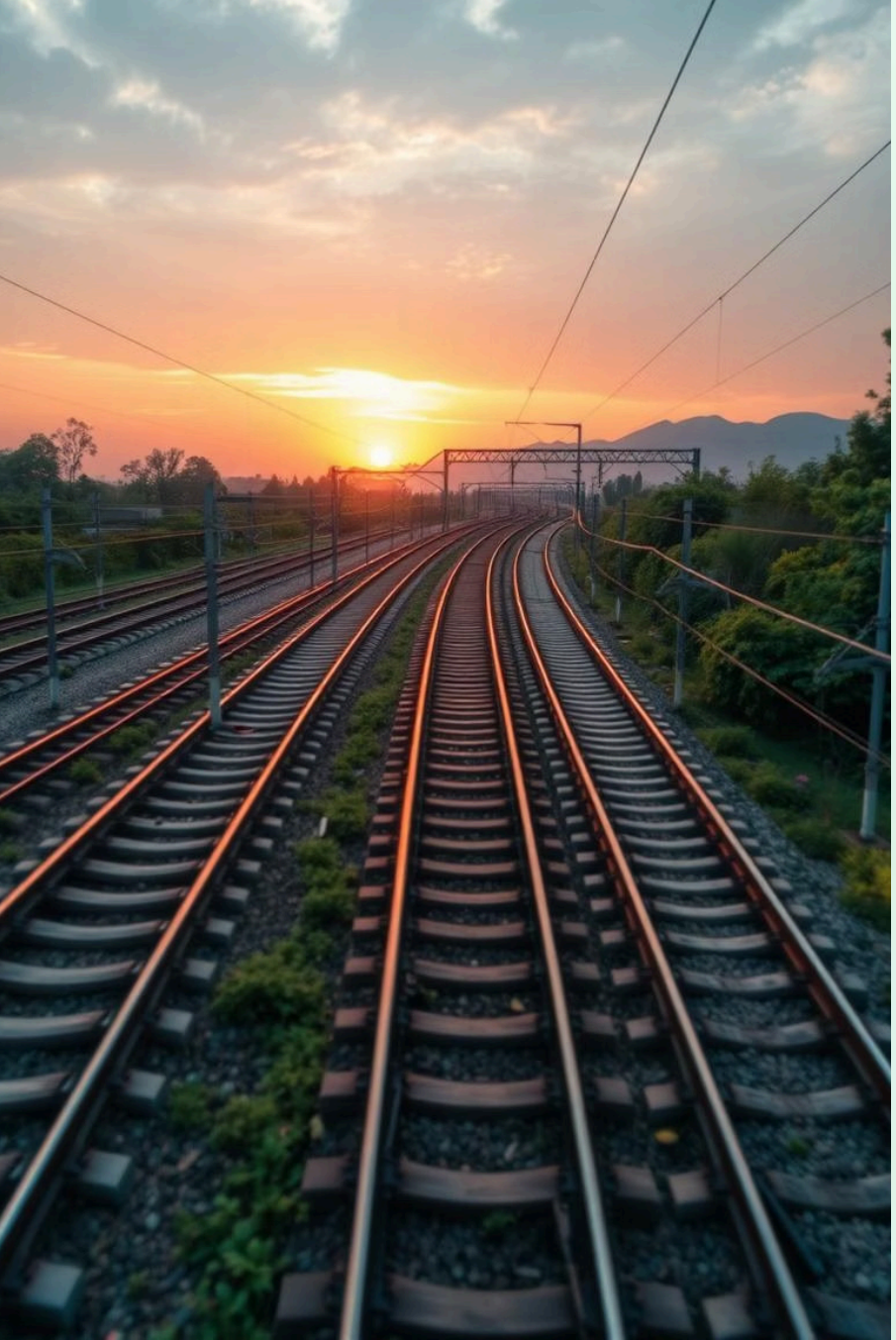
Detailed setup and usage instructions

### MIT License

Open-source permissions and legal terms

### Extra Features

Extensions for better usability and documentation



# Conclusion & Future Work

## Validate Safe Operations

Ensure no deadlocks in real-time railway control

## Enhance Interface

Improve frontend usability and visualization

## Extend Algorithm

Adapt for dynamic track and train scenarios

Improving railway deadlock prevention for safe, efficient transport systems.