

Automatic Conveyor Belt Router

Author: Mamidipelli Krishna Hasini

1. System Overview

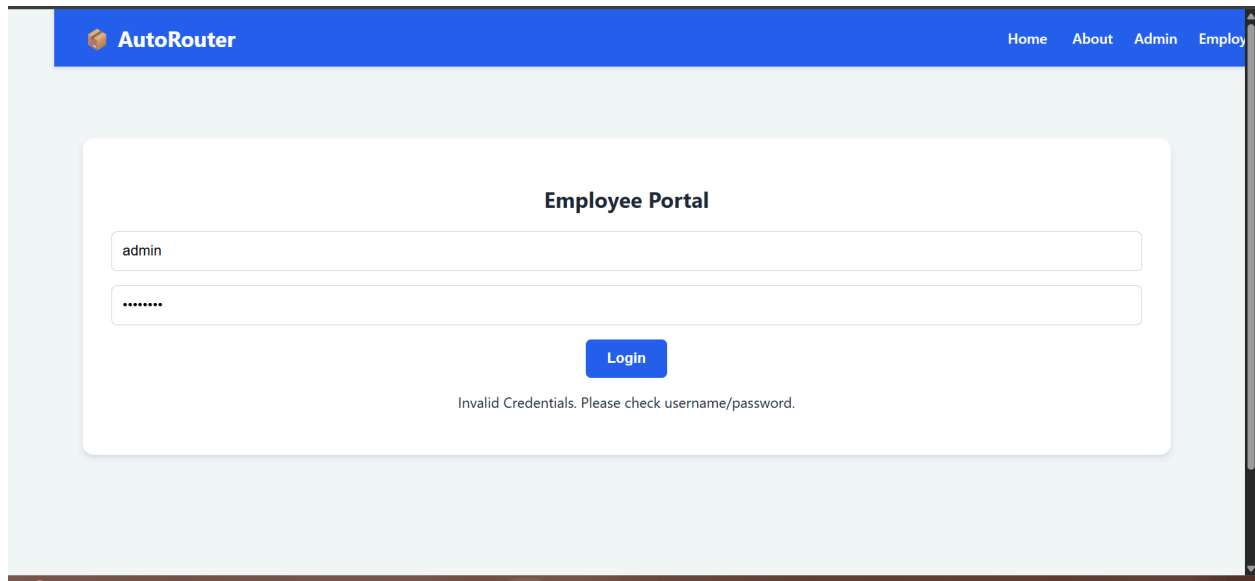
The **Automatic Conveyor Belt Router** is an intelligent logistics system designed to automate the sorting of parcels. It uses Computer Vision (OCR) to read labels from physical packages and queries a high-performance database to determine the routing direction (Straight, Left, Right) in real-time.

1.1 Architecture

The system follows a Client-Server architecture:

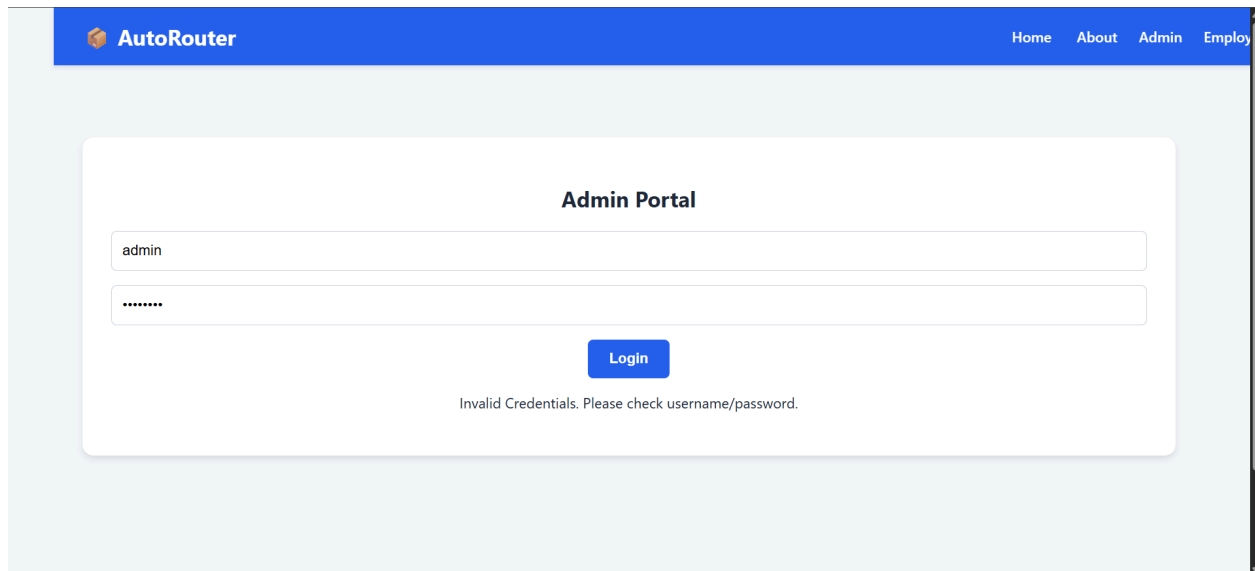
- **Frontend:** React (Vite) - Handles user interaction, camera input, and dashboard visualization.
 - **Backend:** FastAPI (Python) - Handles business logic, OCR processing, and database interactions.
 - **Database:** PostgreSQL (Supabase) - Stores user credentials, city mappings, and routing rules.
 - **AI Engine:** Tesseract OCR & OpenCV - Extracts text from images.
-

Employee portal to enter data



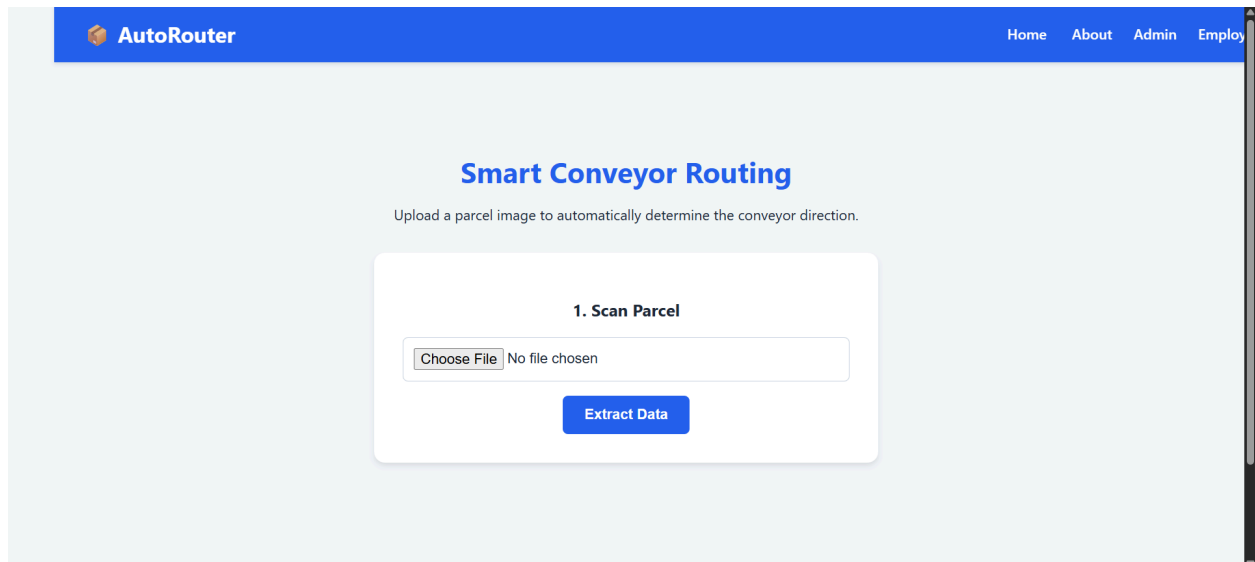
The screenshot shows the 'Employee Portal' login page. At the top is a blue header with the 'AutoRouter' logo on the left and navigation links 'Home', 'About', 'Admin', and 'Employ' on the right. The main content area is light gray and contains a white login box. Inside the box, the title 'Employee Portal' is centered. Below the title are two input fields: the first contains the text 'admin' and the second contains seven dots. A blue 'Login' button is positioned below the password field. At the bottom of the box, a message reads 'Invalid Credentials. Please check username/password.'

2. Admin portal to add data



The screenshot shows the 'Admin Portal' login page. It has the same layout as the Employee Portal page. The blue header contains the 'AutoRouter' logo and navigation links 'Home', 'About', 'Admin', and 'Employ'. The main content area is light gray and contains a white login box. Inside the box, the title 'Admin Portal' is centered. Below the title are two input fields: the first contains the text 'admin' and the second contains seven dots. A blue 'Login' button is positioned below the password field. At the bottom of the box, a message reads 'Invalid Credentials. Please check username/password.'

3. Scanner portal to input the courier image



AutoRouter Home About Admin Employ

Smart Conveyor Routing

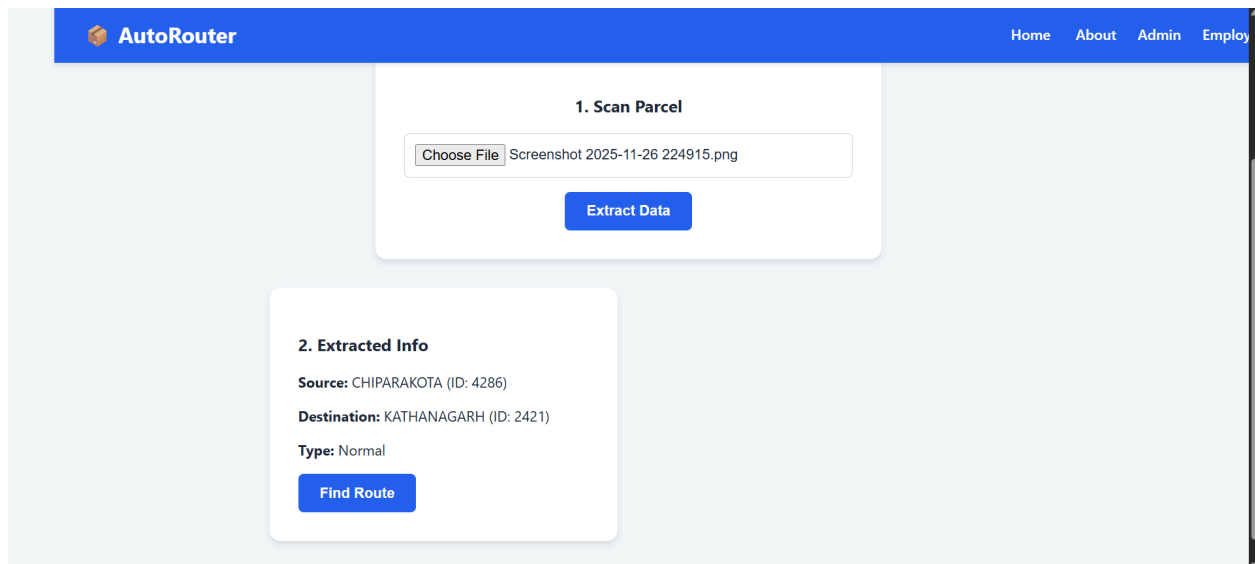
Upload a parcel image to automatically determine the conveyor direction.

1. Scan Parcel

Choose File No file chosen

Extract Data

4. Extracted information shown under this



AutoRouter Home About Admin Employ

1. Scan Parcel

Choose File Screenshot 2025-11-26 224915.png

Extract Data

2. Extracted Info

Source: CHIPARAKOTA (ID: 4286)

Destination: KATHANAGARH (ID: 2421)

Type: Normal

Find Route

5 find route will give the route decision by our system

The screenshot displays the 'AutoRouter' web application interface. At the top, a blue navigation bar contains the 'AutoRouter' logo and links for 'Home', 'About', 'Admin', and 'Employ'. The main content area is divided into three sections:

- 1. Scan Parcel:** A white box containing a 'Choose File' button and a text input field with the filename 'Screenshot 2025-11-26 224915.png'. Below this is a blue 'Extract Data' button.
- 2. Extracted Info:** A white box showing the results of the scan: 'Source: CHIPARAKOTA (ID: 4286)', 'Destination: KATHANAGARH (ID: 2421)', and 'Type: Normal'. A blue 'Find Route' button is at the bottom.
- 3. Routing Decision:** A light green box displaying a blue left-pointing arrow icon followed by the word 'LEFT' in large green letters. Below this, it says 'Route Code: 1'.

6. Employee Portal has option to add new cities to the database and new routes to the system

The screenshot shows the 'AutoRouter' web application in the 'Employee Portal' section. The top navigation bar includes 'Home', 'About', 'Dashboard', and a 'Logout' button. On the left, a sidebar menu has options: 'Data Points' (highlighted in blue), 'Cities Database', 'My Profile', and 'Refresh System'.

The main content area features a form titled 'Add New Routing Rule' with the following fields:

- Source City:** A text input field.
- Destination City:** A text input field.
- Parcel Type:** A dropdown menu currently showing 'Normal (0)'.
- Route:** A dropdown menu currently showing 'Straight'.

A blue 'Add Entry' button is positioned below the form fields.

Below the form, there is a section titled 'Recent Data Entries' which begins with a table header:

Source	Dest	Type	Route
--------	------	------	-------

2. Database Schema

The system uses a relational PostgreSQL database. Below are the table definitions.

2.1 admin-logins & employee-logins

Stores authentication credentials for different roles.

2.2 cities

Master list of valid source/destination locations.

2.3 datapoints (Routing Rules)

The core logic table. Defines how a parcel should move based on source, destination, and type.

3. API Reference

3.1 Authentication

Login

- **Endpoint:** POST /login/{role}
- **Params:** role (string) -> "admin" or "employee"

Body:

JSON

```
{  
  "username": "admin",  
  "password": "admin123"  
}
```

- **Response:** Returns User ID and Role.

3.2 Intelligent Routing (The Core Logic)

Extract Text from Image

- **Endpoint:** POST /extract-from-image
- **Body:** multipart/form-data (File upload)
- **Process:**
 1. Image is received and pre-processed using OpenCV (Grayscale, Thresholding).
 2. Tesseract OCR extracts raw text.
 3. Regex patterns identify "Source", "Destination", and "Parcel Type".

Response:

JSON

```
{
  "source_city": "MUMBAI",
  "source_id": 101,
  "dest_city": "DELHI",
  "dest_id": 204,
  "type": 0
}
```

-

Find Route

- **Endpoint:** POST /find-route

Body:

JSON

```
{
  "source_city_id": 101,
  "destination_city_id": 204,
  "parcel_type": 0
}
```

-

- **Logic:** Checks the in-memory ROUTE_CACHE (O(1) lookup) for a match.

Response:

JSON

```
{
  "found": true,
  "route_code": 2,
  "direction": "Right"
}
```

-

3.3 Management Endpoints (Admin Only)

Method	Endpoint	Description
GET	/users/{role}	Get list of admins or employees.
POST	/add-user/{role}	Create a new user account.
DELETE	/remove-user/{role}/{id}	Delete a user account.
GET	/cities	List all registered cities.
POST	/add-city	Register a new city.
POST	/add-datapoint	Add a new routing rule (e.g., Mumbai->Delhi = Left).
POST	/refresh-data	Forces backend to reload DB rules into RAM cache.

Export to Sheets

4. Algorithms & Logic

4.1 Image Pre-processing Pipeline

To ensure high accuracy in reading labels, images undergo the following OpenCV transformations before OCR:

1. **Upscaling:** `cv2.resize(fx=3, fy=3)` - Increases image resolution.
2. **Grayscale:** `cv2.cvtColor(BGR2GRAY)` - Removes color noise.
3. **Otsu's Thresholding:** `cv2.threshold` - Converts to strict Black & White to separate text from background.

4.2 Caching Mechanism (Performance Optimization)

Database queries are slow. To ensure the conveyor belt runs smoothly:

1. **On Startup:** The server fetches **all** rows from the datapoints table.
2. **Memory Map:** It builds a Python Dictionary (Hash Map):
 - **Key:** (source_id, dest_id, type)
 - **Value:** route_direction
3. **Lookup:** When scanning a parcel, the system checks this Hash Map.
 - **Time Complexity:** $O(1)$ (Instant).
 - **Benefit:** No database lag during operation.

5. Deployment Configuration

5.1 Environment Variables (.env)

DATABASE_URL

`postgresql://postgres.clgjswrlcwzmdxhhegtk:adhithya365@aws-1-ap-south-1.pooler.supabase.com:6543/postgres`

`,DEBUG`

5.2 Docker Container

Because the backend requires **Tesseract OCR** (a C++ binary), it cannot run on standard Python hosting. It runs in a Docker container:

- **Base Image:** python:3.10-slim
- **System Dependencies:** tesseract-ocr, libgl1 (for OpenCV).
- **Port:** 8000

5. Deployment Under Process

Server: <https://automaticconveyorbeltroutersystemdesign-l3l8.onrender.com/docs>

Client :<https://automaticconveyorbeltroutersystemdesign.onrender.com>