**Purpose of this document:**

The document outlines the proposed solution design for a task and driver management system. It defines the database schema, workflows for both Admin and Driver users, and the integration points for automation such as invoice handling and proof of delivery (POD).

# 1. Database Structure

## Driver Table:

✅ Stores driver details including vehicle type, availability, and load capacity.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Driver\_Id | Driver Name | Capacity (m³) | Truck Number | Truck | Status |
| 111 | Danton B | 43.91 | 445 | Pantech | Available |
| 212 | Joshua M | 44.20 | 447 | Tautliner | Busy |

## Task Table

• Stores all incoming tasks from clients.  
• Populated automatically via Excel import script.



## Driver–Task Mapping Table:

✅ Links drivers with tasks and invoices.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Mapping\_Id | Driver\_Id (FK) | Order\_No | Task\_Id (FK) | Manifest\_No | Invoice\_Id |
| 1 | 111 | 20593399 | 123 | — | — |
| 2 | 212 | 20595753 | 124 | — | — |

## Completed Task Table:

✅ Stores proof of delivery (POD) and task closure details.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Task\_Id | Order\_No | Invoice\_Id | Completed By | Completed On | POD |
| 123 | 20593399 | — | Adam | 04/09/2025 | — |
| 124 | 20595753 | — | Ben | 05/09/2025 | — |

User Table:  
✅ Stores login credentials and role details for both Admins and Drivers.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Username | Email | Password\_Hash | Role |
| 1 | Mark | Mark@gmail.com | 122\_hddksd\_dss | Admin |
| 2 | Adam | Adam@gmail.com | 1221\_dsfsn65\_ds | Driver |

**Notes:**

1. POD will be saved at one drive.
2. Invoice\_Id and Manifest\_No fields will be kept initially empty. They will be populated once the manifest/invoice sheet in generated in warehouse and shared with us. **(Clarification still pending on how it will be shared with us).**

**2. Admin Workflow**

1. **Admin Login**

A login screen shot

AI-generated content may be incorrect.

**1. Access Portal**  
 Admin navigates to the web portal login page.  
 **Technology Stack:** *ReactJS*

**2. Enter Credentials**  
 Admin enters a valid **Username/Email** and **Password**.

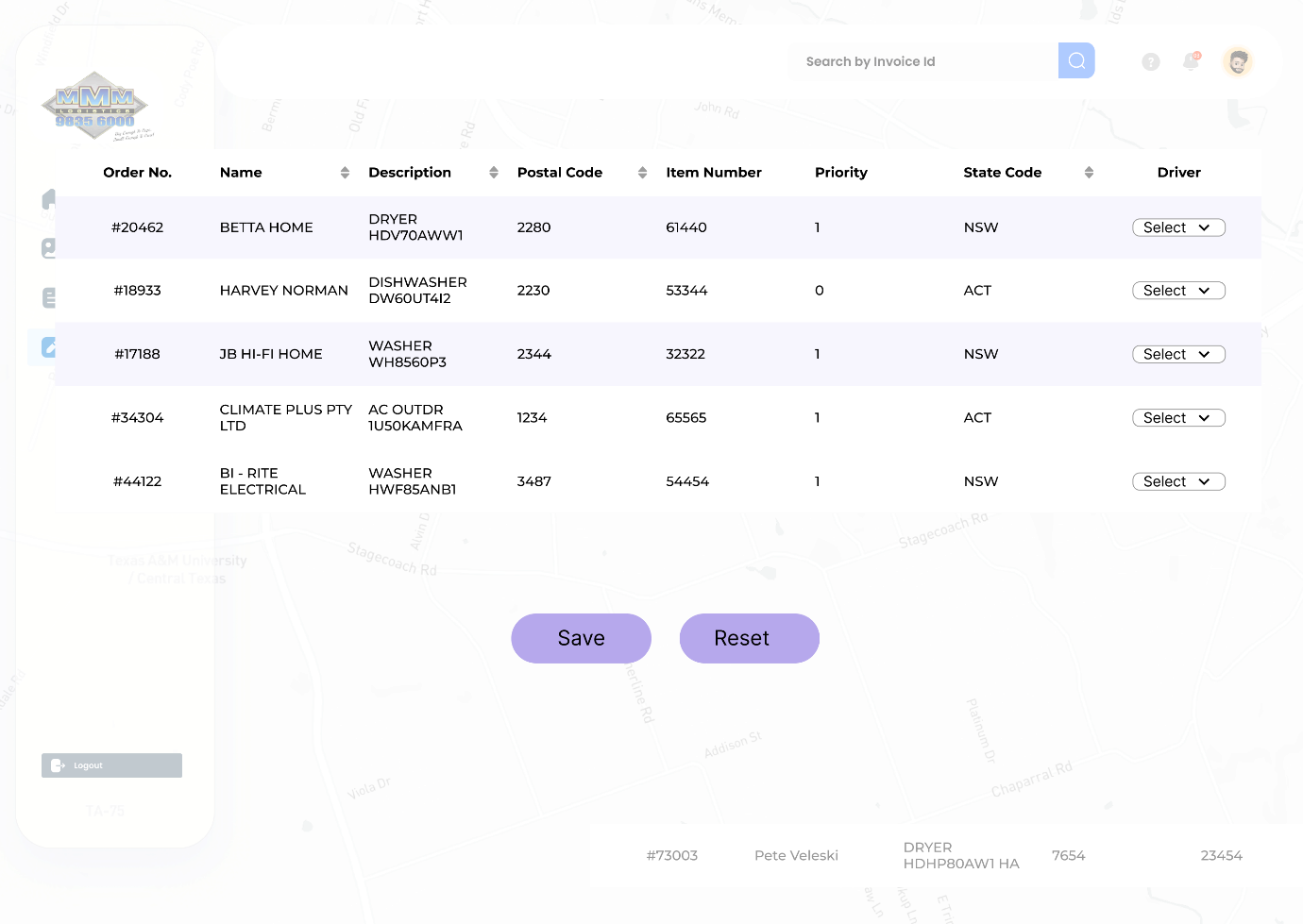
**3. Redirect to Dashboard**  
 Upon successful login, the admin is redirected to the **Dashboard Home Page**, which displays the following key panels:

* **Assign Tasks**
* **Tasks In Progress**
* **Completed Tasks**

1. **Assign Tasks**

A screenshot of a computer

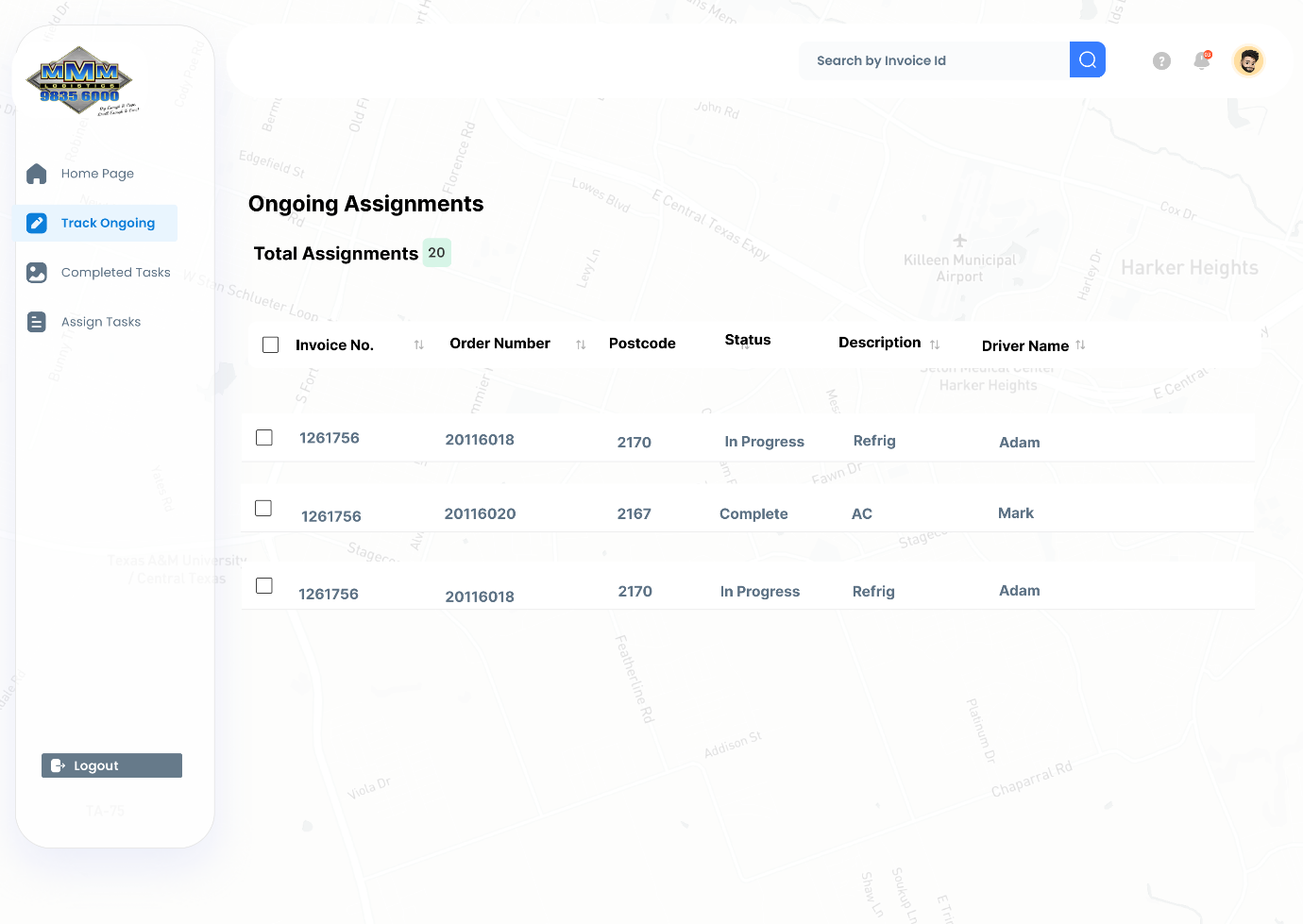
AI-generated content may be incorrect.



1. Admin clicks \*\*Assign Tasks\*\*.
2. System fetches \*\*Unassigned\*\* tasks from the Task Table and \*\*Available\*\* drivers from the Driver Table.
3. Tasks are ordered/sorted by zones. Admin can select any of the zone.
4. Clicking on any zone pops up task assessment window. Admin selects one or more tasks and chooses an available driver.
5. Admin clicks \*\*Save Button\*\*.
6. System actions on confirm:

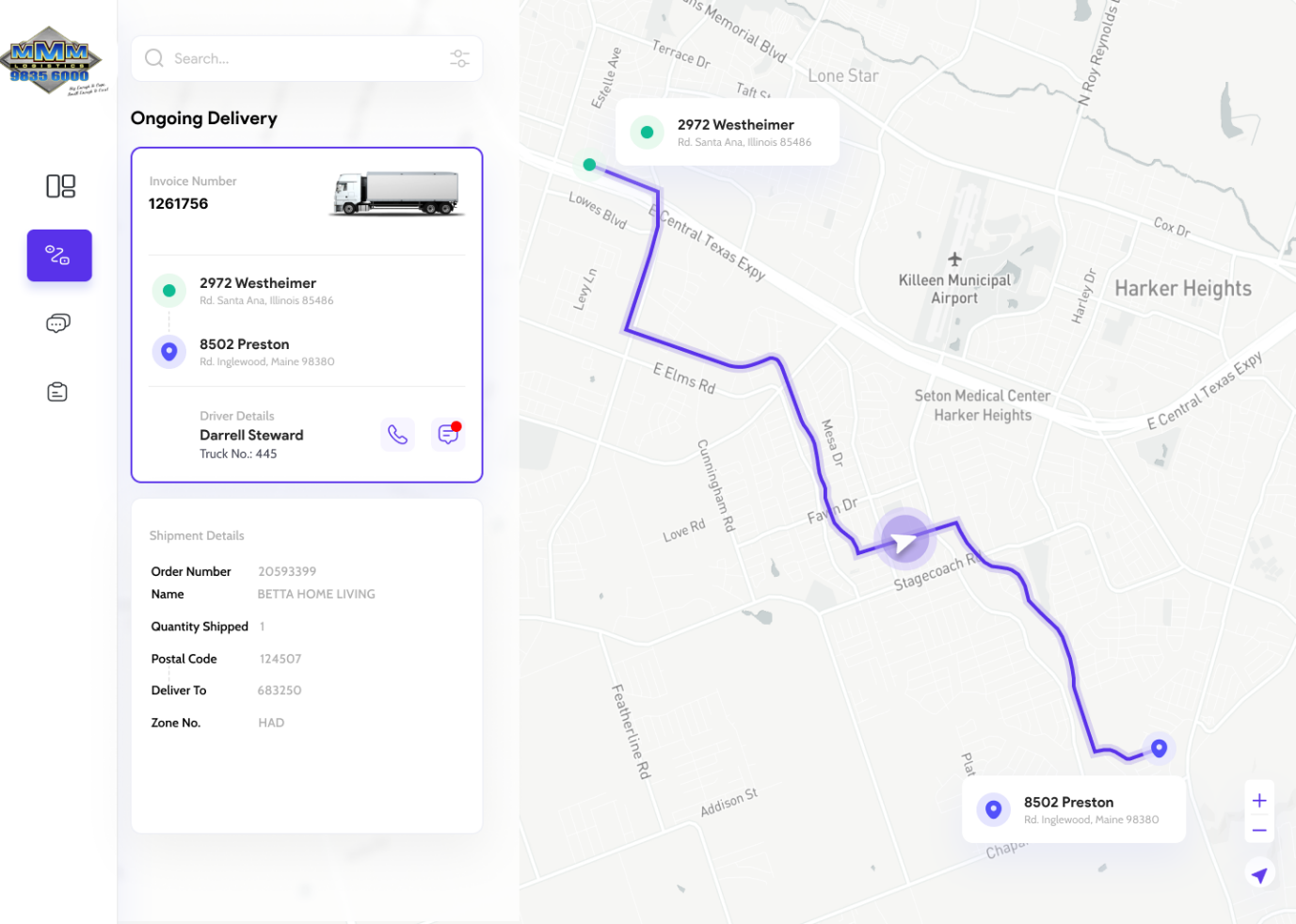
* Create/Update records in the Driver–Task Mapping Table for each selected task.
* Update the Task Table: set status to \*\*Assigned\*\*, add selected DriverId, and assignment timestamps.
* Update the Driver Table: set driver status to \*\*In-Progress/Busy\*\*.
* Redirect admin back to the Home/Dashboard. The newly assigned items now appear under \*\*Tasks in Progress\*\*.

1. **Monitor Ongoing Assignments**



1. Admin clicks \*\*Tasks in Progress\*\*.
2. System fetches ongoing assignments by joining Task Table ↔ Driver–Task Mapping ↔ Driver Table.
3. The list view shows key columns (Task ID, Client, Pickup/Drop, Driver, Status, Last Update). Filters & search help narrow results.

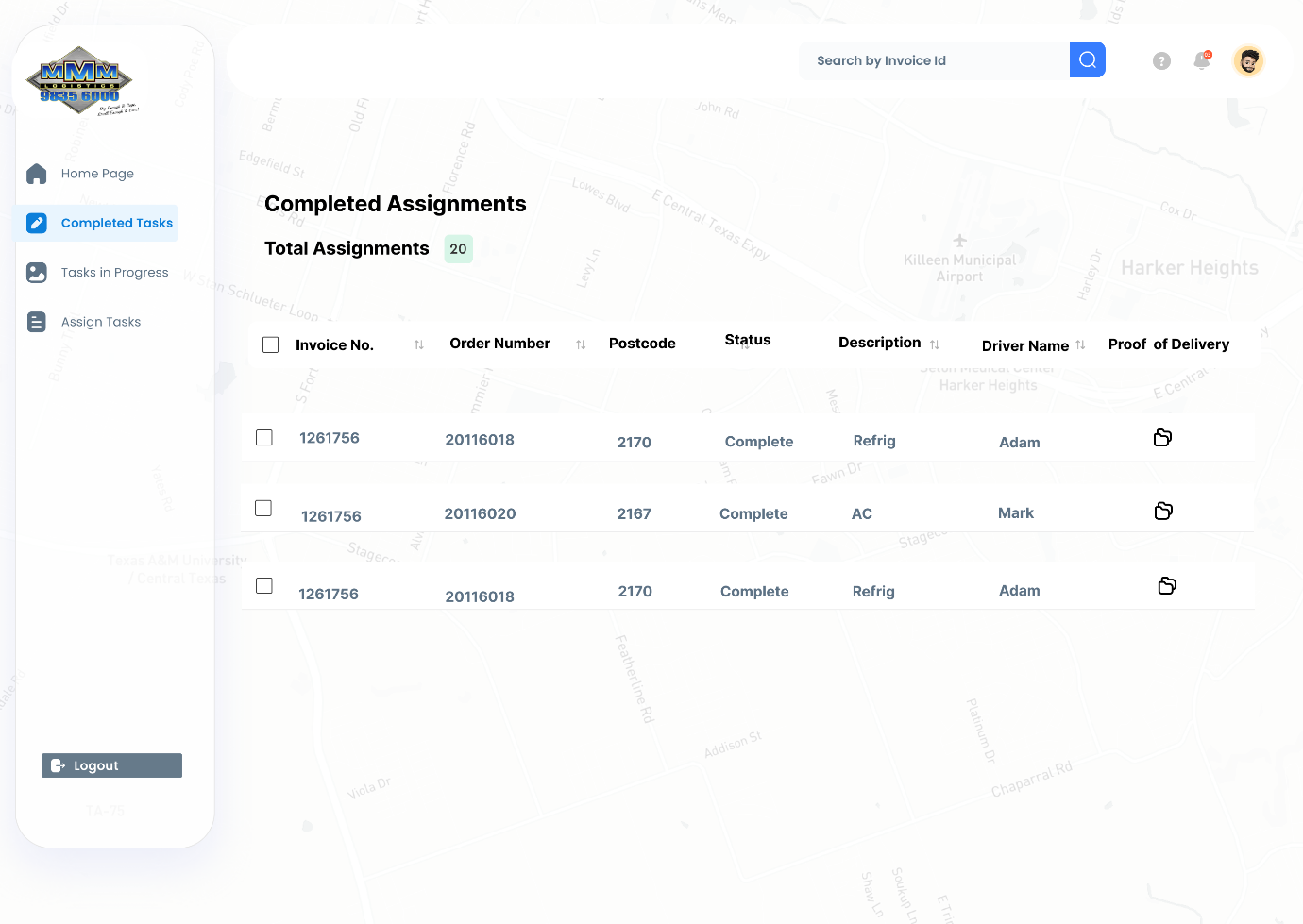
**4.View GPS Coordinates:**



1. Clicking a row or \*\*View Details\*\* opens the assignment detail view with:

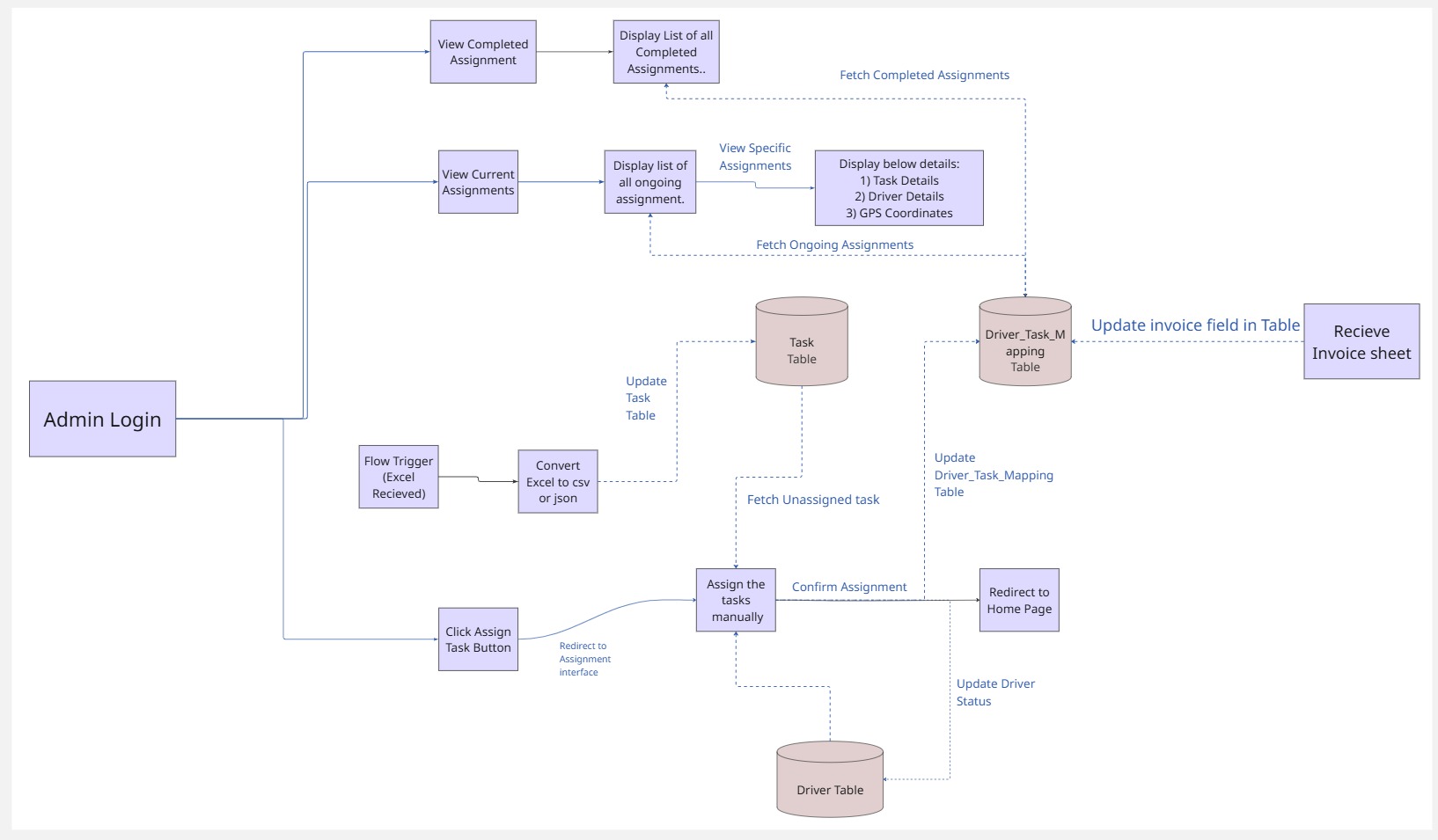
* Task details (addresses, timeslots, notes).
* Driver details (name, vehicle, contact, current status).
* Live GPS coordinates / map view (from the driver app or tracker).

**5) View Completed Assignments**



1. Admin clicks \*\*Completed Tasks\*\*.
2. System fetches records where Task status = \*\*Completed\*\* (and/or Mapping status = Completed).
3. Each row provides actions: View/Download POD (PDF compiled from delivery photos, customer signature and docket).
4. **Invoice / Manifest Handling**
   * Invoices can be uploaded manually to website once it gets generated in the warehouse.
   * Mapping DB is updated with invoice details.
   * **If the manifest/invoice sheet is generated electronically and shared directly with us, we can integrate that too.**
   * This would eliminate manual handling of invoices and make the process fully automated.

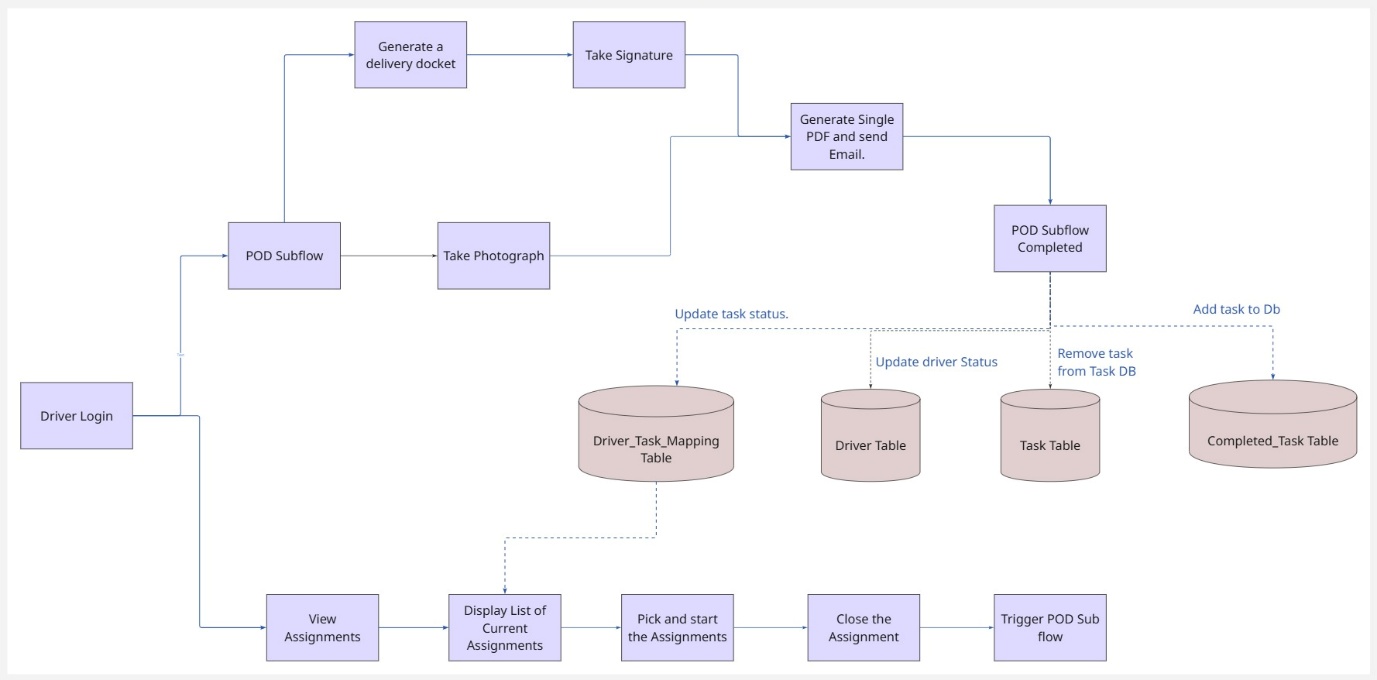
**Admin Flow Architecture**



**3. Driver Workflow**

1. **Login** → Driver logs into the system.
2. **View Assignments** → Sees all tasks assigned.
3. **Start Assignment** → Picks a task, mapping DB and driver status update.
4. **Complete Assignment** → Marks it done, triggering the **POD subflow**.
5. **POD Subflow** →
   * Generate delivery docket.
   * Capture customer signature.
   * Take delivery photos.
   * Compile into a single PDF.
   * Email automatically to the Admin.
6. **Update System** → Task is moved into the **Completed Task DB**, driver status set back to “Available.”

**Driver Flow Architecture**



So, the **only human involvement** is:

* Admin assigning tasks.
* Driver completing them.
* Everything else (task sheet upload, POD creation, invoice integration) can be automated.