**//My plans mobile app includes**

* **Scheduling all and ensuring that no conflict**
* **Streamlining Learning path**
* **Sources**
* **Scheduling**
* **Notifications**
* **Project build**
* **Job application**
* **Scholarship seeking**
* **Looking at specific websites and journals to get updated**
* **And more**
* **Integrating Ai for recommendations, best practices and advices.**

**Problem Statement**

1. **Users can get an item from where the item is available without physical barriers.**
2. **Users can trade easily.**
3. **Students/frequent travelers can get a side income while traveling for other things.**
4. **Verified/Trusted users.**

**User Story**

You're building a **P2P Postal Service** where:

1. **Senders** (people who want to send items) can find **Travelers** (people already traveling to the desired destination).
2. **Travelers** can earn money by carrying and delivering items for **Senders**.

**Suggested Names for the Two Parties:**

1. **Senders (or "Shippers")** – People who want to send an item from one location to another.
   * Example names in other platforms: "Senders" (Send Bag), "Requesters" (Roadie).
2. **Travelers (or "Carriers", "Couriers", "Bringers")** – People who are traveling and can carry/deliver items.
   * Example names in other platforms: "Travelers" (Grabr), "Carriers" (Shiply).

**Alternative Platform Names:**

* **SendAlong**
* **PeerPost**
* **TripCarry**
* **Gofer** (like "go-for" deliveries)
* **BagBuddy**
* **WaySend**

**Key Features Needed:**

* **Search & Match System** (Senders find Travelers based on date, route, weight, price).
* **Verification & Trust System** (ID checks, reviews, ratings).
* **Secure Payments & Escrow** (Hold payment until delivery is confirmed).
* **Real-Time Tracking** (Optional: GPS tracking for high-value items).

Below is a **detailed breakdown** of your **P2P Postal Service System**, including:

1. **Database Schema** (Tables & Relationships)
2. **User Flow** (How senders & travelers interact)
3. **Key Features & Considerations**

**1. Database Schema (PostgreSQL Example)**

Your system will need the following tables:

**Users Table**

Stores are all registered users (both **Senders** and **Travelers**).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Default | Data Type | Nullable | Description |
| id |  | Number | False | Primary key of the record/ unique identifier |
| Username |  | String | False | Username the user uses within the system |
| UserType |  | Enum | False | Type of user within the system |
| Email |  | String | True | Email of the user |
| Password |  | string | False | Password to log into the system |
| Phone |  | String | False | Working mobile number that is used by the user |
| Verified | False | Boolean | False | Whether the user has verified his identity or not |
| Rating | 5.00 | Decimal | False | Overall rating |
| Created At | 01.01.0001 | DateTime | false | The exact date and time when the user registered |
| Total\_deliveries | 0 | Integer | true | The total number of deliveries the user has got. |
| Successful\_deliveries | 0 | Integer | true | The successful number of deliveries the user has done. |

**User Profiles (Additional Details)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Default | Data Type | Nullable | Description |
| Profiled |  | Number | False | Primary key |
| UserId |  | FK (user) | False | Foreign key that maps to user table |
| FirstName |  | String | False | User Given name |
| MiddleName |  | String | False | User Father’s name |
| LastName |  | String | False | User grandfather’s name |
| Address |  | String | False | The place where the user permanently lives |
| IsMarried | No | Enum | True | Marriage status |
| GovernmentID |  | String | False | Government card Id number (the card may be passport(nice), AAID (good), Driving License(good), KebeleId(enough), NationalId(ok)) |
| IDPhoto |  | File(Image/pdf) | False | Id picture |
| PPhoto |  | File (Image) | False | Profile picture |
| Bio | null | String | True | Biography about the user |

**Trips Table (Posted by Travelers)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Default | Data Type | Nullable | Description |
| TripId |  | Number | False | Unique Identifier |
| Traveler\_id |  | FK(User) | False | User that travels |
| Departure\_city |  | String | False | Place where the traveler departs from |
| Arrival\_city |  | String | False | Place where the traveler trips to |
| departure\_date |  | Date | False | Date when the traveler departs |
| departure\_time |  | Time | False | Exact Time when the traveler departs |
| Max\_weight\_kg | 100.00 | Decimal(5,2) | False | Maximum weight the traveler can carry |
| Min\_weight\_kg | 0.00 | Decimal (5,2) | False | Minimum weight the traveler can carry |
| Type\_of\_price\_calc | perkg | Enum | false | Type of price calculation |
| Price\_per\_kg |  | Decimal | True | Price per kilogram if Type\_of\_price\_calc is “perkg” |
| Currency | ETB | Enum | False | Currency that is going to use as a pay currency. |
| Number\_of\_deliveries | 0 | Integer | False | Number of deliveries that a traveler is going to handle. |
| Available\_space | True | Boolean | False | Whether available or occupied. |

**Delivery Requests (Posted by Senders)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Default | Data Type | Nullable | Description |
| Request\_id |  | Number | False |  |
| Sender\_id |  | FK(User) | False |  |
| Pickup\_city |  | String | False |  |
| Delivery\_city |  | String | False |  |
| Desired\_delivery\_date |  | Date | False |  |
| Item\_description |  | String | False |  |
| Weight\_kg |  | Decimal(5,2) | False |  |
| Budget |  | Decimal(5,2) | False |  |
| Status | Pending | Enum(DeliveryStatus) | False |  |

**Delivery Status**

**Enum{**

**0:Cancelled,**

**1:Pending,**

**2:Accepted,**

**3: Delivered,**

4: Picked Up

**}**

**Match (When a Sender & Traveler Agree)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Default** | **Data Type** | **Nullable** | **Description** |
| **Match\_id** |  | **PK(Number)** | **False** |  |
| **deliveryRequest\_id** |  | **FK(request)** | **False** |  |
| **letMeDeliver\_id** |  | **FK(trip)** | **False** |  |
| **Agreed\_price** |  | **Decimal(10,2)** | **False** |  |
| **Status** |  | **Enum(DeliveryStatus)** | **False** |  |
| **Tracking\_code** |  | **String** | **False** | **Unique for updates** |
| **Pickup\_confirmed** | **False** | **Boolean** | **False** | **True, if it is handed over from sender to traveler** |
| **Delivery\_confirmed** | **False** | **Boolean** | **False** | **True, if it is handed over from traveler to receiver.** |
| **Created\_at** |  | **DateTime** | **False** |  |

**Reviews & Ratings**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | Default | Data Type | Nullable | Description | |
| Id | 0 | Num | False | Unique identifier within the table | |
| Rating | 0.00 | Float | False | Satisfaction level | |
| Comment | null | String | True | Feedback and/or comments | |
| Reviewer\_id |  | FK (user) | False | User who rates | |
| Reviewed\_id |  | FK (user) | False | User who is rated |
| MatchId |  | FK (match) | False | The service that delivers is done. | |
| RatedBy |  | enum | False | Is sender or carrier rates | |
| **Created\_At** |  | **DateTime** | **False** |  | |

**Payments & Escrow**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Default | Data Type | Nullable | Description |
| Payment\_id |  | Pk(Number) | False |  |
| Match\_id |  | FK(Match) | False |  |
| Amount |  | Decimal(10,2) | False |  |
| Payment\_status |  | Enum(Payment\_status) | False |  |
| Stripe\_payment\_id |  | String | False | If using stripe |
| Released\_At |  | DateTime | False |  |
| Created\_At |  | DateTime | False |  |

**Payment\_status**

**Enum {**

**0: Refunded,**

**1: Held,**

**2: Released**

**}**

**2. User Flow (How the System Works)**

**For Travelers (Carriers)**

1. **Sign Up & Verify Identity** (Email, Phone, ID Check)
2. **Post a Trip** (Departure, Destination, Date, Max Weight, Price)
3. **Get Matched** with Senders needing deliveries
4. **Accept a Request** & Confirm Pickup
5. **Deliver Item** & Get Paid (Escrow releases funds)

**For Senders (Shippers)**

1. **Sign Up & Verify Identity**
2. **Post a Delivery Request** (Item, Weight, Desired Date, Budget)
3. **Search for Travelers** (Filter by route, date, price)
4. **Select a Traveler** & Agree on Terms
5. **Pay via Escrow** (Held until delivery)
6. **Confirm Delivery** & Release Payment

**3. Key Features & Considerations**

**Essential Features**

✅ **Search & Matching System**

* Senders can **search by route, date, weight, price**
* Travelers can **see open delivery requests**

✅ **Trust & Safety**

* **ID Verification** (KYC)
* **Ratings & Reviews**
* **Escrow Payments** (Money held until delivery)

✅ **Notifications & Tracking**

* **Email/SMS alerts** for new matches
* **Optional GPS tracking** for high-value items

**Advanced Features (Future)**

🔹 **Bidding System** – Sender’s post requests, travelers bid.  
🔹 **Recurring Deliveries** – For frequent travelers (e.g., weekly commuters).  
🔹 **Insurance Option** – For expensive items.  
🔹 **Chat System** – Direct communication between users.

**Next Steps**

* **Build MVP** (Minimum Viable Product) with core features.
* **Implement Secure Payments** (Stripe, PayPal).
* **Add Verification** (Phone/ID checks to reduce fraud).
* **Launch in a Niche Market** (e.g., students, business travelers).