Drivers Payments - Replit Prompt

Generated: 2025-10-27 05:54

````
You are editing a full-stack TS project:

- server/: Express + Drizzle (Postgres/Neon). Routes in server/routes.ts, storage in
server/storage.ts, schemas in shared/schema.ts.
- client/: React + TS + Vite + shadcn/ui. Router in client/src/App.tsx.
- API helper: client/src/lib/api.ts. Date helpers: client/src/lib/dates.ts.
- Timezone: Asia/Kolkata. Date ranges MUST be inclusive on both ends.

====================================
TASK: Replace "Rent Tracking" with a new page "Drivers Payments"
====================================
Goal:
- DELETE the "Rent Tracking" page/route/nav item.
- ADD a new page **Drivers Payments** (`/drivers-payments`).
- This page reuses the Weekly Summary aggregation but renders ONLY two columns:
  **Driver** | **Total**
- Date range filter (always visible):
  - **Start** default = earliest trip date in the DB (MIN(trip_date)) — inclusive
  - **End** default = today (IST) — inclusive
- Only drivers with at least one trip log within the selected range should appear.
- **Total** definition remains EXACTLY as Weekly Summary:
  - Wallet = TotalEarnings − Cash + Refund − Expenses − 100
  - Total = Collection + Wallet + Dues − Rent − Payout
  (All values come from the weekly summary upsert table and trip-log sums.)

====================================
SERVER: meta endpoint for default Start Date
====================================
1) In server/storage.ts add:

```ts
import { sql } from "drizzle-orm";
import { driverRentLogs } from "../shared/schema"; // adjust path

export async function getFirstTripDate(db): Promise<string | null> {
  const rows = await db.execute(sql`SELECT MIN(trip_date)::date AS min_date FROM driver_rent_logs`);
  const min = Array.isArray(rows) ? rows[0]?.min_date : rows.rows?.[0]?.min_date;
  return min ? String(min) : null; // 'YYYY-MM-DD' or null
}
```

2) In server/routes.ts add a meta route:

```ts
import { getFirstTripDate } from "./storage"; // adjust path

app.get("/api/meta/first-trip-date", async (req, res) => {
  const d = await getFirstTripDate(req.db);
  res.json({ firstTripDate: d }); // 'YYYY-MM-DD' or null
});
```

3) Ensure Weekly Summary GET already uses **inclusive** DATE logic:
````

Drivers Payments - Replit Prompt

Generated: 2025-10-27 05:54

```ts
// in listWeeklySummary(...) WHERE clause
.where(sql`DATE(${driverRentLogs.tripDate}) BETWEEN ${startDate}::date AND ${endDate}::date`)
```
(If not, change it now.)

```
====================================
CLIENT: API helpers
====================================
```
4) In client/src/lib/api.ts add:

```ts
export async function getFirstTripDate() {
  const r = await fetch("/api/meta/first-trip-date");
  if (!r.ok) throw new Error("Failed to load first trip date");
  const { firstTripDate } = await r.json();
  return firstTripDate as string | null; // 'YYYY-MM-DD' or null
}

// Reuse getWeeklySummary(start,end) you already created.
// It must return each row with driverId, driverName, total (computed on server).
```

```
====================================
CLIENT: New page
====================================
```
5) Create `client/src/pages/drivers-payments.tsx`:

Requirements:
- Always-visible filter with Start / End inputs.
- On first mount:
  - fetch `/api/meta/first-trip-date`
  - if null, set Start=today
  - set End=today (IST)
  - immediately fetch weekly summary with these two dates.
- Table with exactly **two** columns: **Driver** and **Total**.
- Totals formatted in INR, no decimals.
- Only drivers present in the summary are shown (server ensures this).

Implementation outline:

```tsx
import { useEffect, useState } from "react";
import { useQuery } from "@tanstack/react-query";
import { getFirstTripDate, getWeeklySummary } from "../lib/api";
import { todayIST, toYYYYMMDDIST } from "../lib/dates";
import { Input } from "@/components/ui/input"; // shadcn
import { Card } from "@/components/ui/card";
import { Table, TableHead, TableHeader, TableRow, TableBody, TableCell } from
"@/components/ui/table";

const inr = (n:number)=> new Intl.NumberFormat("en-
IN",{style:"currency",currency:"INR",maximumFractionDigits:0}).format(n||0);
```

Drivers Payments - Replit Prompt

Generated: 2025-10-27 05:54

```
export default function DriversPaymentsPage() {
  const [start, setStart] = useState<string>("");
  const [end, setEnd] = useState<string>("");

  // bootstrap defaults
  useEffect(() => {
    (async () => {
      try {
        const first = await getFirstTripDate();
        const today = toYYYYMMDDIST(todayIST());
        setStart(first ?? today);
        setEnd(today);
      } catch {
        const today = toYYYYMMDDIST(todayIST());
        setStart(today);
        setEnd(today);
      }
    })();
  }, []);

  const { data, isLoading, error } = useQuery({
    queryKey: ["drivers-payments", start, end],
    queryFn: () => getWeeklySummary(start, end),
    enabled: !!start && !!end,
  });

  const items = data?.items ?? [];

  return (
    <div className="space-y-4">
      <h1 className="text-2xl font-bold">Drivers Payments</h1>

      {/* Filter bar */}
      <Card className="p-4 flex gap-4 items-end">
        <div>
          <label className="block text-sm font-medium mb-1">Start</label>
          <Input type="date" value={start} onChange={(e)=>setStart(e.target.value)} />
        </div>
        <div>
          <label className="block text-sm font-medium mb-1">End</label>
          <Input type="date" value={end} onChange={(e)=>setEnd(e.target.value)} />
        </div>
      </Card>

      <Card className="p-0 overflow-hidden">
        <Table>
          <TableHead>
            <TableRow>
              <TableHeader>Driver</TableHeader>
              <TableHeader className="text-right">Total</TableHeader>
            </TableRow>
          </TableHead>
          <TableBody>
            {isLoading && (
```

Drivers Payments - Replit Prompt

Generated: 2025-10-27 05:54

```
              <TableRow><TableCell colSpan={2}>Loading…</TableCell></TableRow>
            )}
            {error && (
              <TableRow><TableCell colSpan={2}>Failed to load.</TableCell></TableRow>
            )}
            {!isLoading && items.length === 0 && (
              <TableRow><TableCell colSpan={2}>No drivers in this range.</TableCell></TableRow>
            )}
            {items.map(row => (
              <TableRow key={row.driverId}>
                <TableCell>{row.driverName}</TableCell>
                <TableCell className="text-right">{inr(row.total)}</TableCell>
              </TableRow>
            ))}
          </TableBody>
        </Table>
      </Card>
    </div>
  );
}
```

```
=====================================
CLIENT: Router & Nav
=====================================
6) In `client/src/App.tsx`:
- REMOVE the route and nav link for the old **Rent Tracking** page.
- ADD:
```tsx
import DriversPaymentsPage from "./pages/drivers-payments";
// ...
<Route path="/drivers-payments" element={<DriversPaymentsPage />} />
```
- Add a sidebar/top-nav link “Drivers Payments” → `/drivers-payments`.

=====================================
BEHAVIOR & VALIDATION
=====================================
- Start/End range is **inclusive** (server uses `DATE(...) BETWEEN start::date AND end::date`).
- Default Start = earliest trip date from DB; End = today (IST).
- Only drivers with ≥ 1 trip in range appear.
- Page shows ONLY two headers: **Driver** and **Total**. No editable fields here.
- INR formatting (no decimals).

=====================================
ACCEPTANCE TESTS
=====================================
- When opening Drivers Payments, Start auto-loads as the earliest trip date; End = today (IST).
- If there’s a trip on the Start day, it is included in totals (no off-by-one).
- Changing either date refetches results immediately.
- The table lists only drivers who have logs in the range.
- Columns shown are exactly “Driver” and “Total”, and totals match Weekly Summary’s computation.
```