```
enum DriverStatus {
  applied
  tested
  qualified
  active
  inactive
}

enum AssignmentStatus {
  draft
  dispatched
  filled
  cancelled
}

enum ResponseType {
  pending
  accepted
  declined
  no_response
}

enum DriverAssignmentStatus {
  confirmed
  completed
  cancelled
}

// Models

// Managers (Web Portal Users)
model Manager {
  id           String   @id @default(uuid())
  email        String   @unique
  passwordHash String   @map("password_hash")
  name         String
  createdAt    DateTime @default(now()) @map("created_at")
  updatedAt    DateTime @updatedAt @map("updated_at")

  // Relations
  tests       Test[]
  assignments Assignment[]

  @@map("managers")
}

// Drivers (Mobile App Users)
```

```
model Driver {
  id            String       @id @default(uuid())
  email         String       @unique
  phone         String       @unique
  name          String
  licenseNumber String?      @map("license_number")
  status        DriverStatus @default(applied)
  createdAt     DateTime     @default(now()) @map("created_at")
  updatedAt     DateTime     @updatedAt @map("updated_at")

  // Relations
  testResults         TestResult[]
  availabilitySlots   AvailabilitySlot[]
  dispatchResponses   DispatchResponse[]
  driverAssignments   DriverAssignment[]
  selectedAssignments Assignment[]

  @@map("drivers")
}

// OTP Codes for driver authentication
model OtpCode {
  id        String   @id @default(uuid())
  email     String
  code      String   @db.VarChar(6)
  expiresAt DateTime @map("expires_at")
  verified  Boolean  @default(false)
  createdAt DateTime @default(now()) @map("created_at")

  @@index([email, code])
  @@map("otp_codes")
}

// Tests
model Test {
  id          String   @id @default(uuid())
  title       String
  description  String?
  passingScore Int      @map("passing_score")
  createdById  String   @map("created_by")
  createdAt    DateTime @default(now()) @map("created_at")
  updatedAt    DateTime @updatedAt @map("updated_at")

  // Relations
  createdBy Manager          @relation(fields: [createdById], references: [id])
  questions TestQuestion[]
  results   TestResult[]
```

```
  @@map("tests")
}

// Test Questions
model TestQuestion {
  id            String @id @default(uuid())
  testId        String @map("test_id")
  question      String
  options       Json   // Array of options ["A", "B", "C", "D"]
  correctAnswer String @map("correct_answer")
  orderIndex    Int    @default(0) @map("order_index")

  // Relations
  test Test @relation(fields: [testId], references: [id], onDelete: Cascade)

  @@map("test_questions")
}

// Test Results
model TestResult {
  id       String   @id @default(uuid())
  driverId String   @map("driver_id")
  testId   String   @map("test_id")
  answers  Json?    // Driver's answers {"q1": "A", "q2": "B"}
  score    Int
  passed   Boolean
  takenAt  DateTime @default(now()) @map("taken_at")

  // Relations
  driver Driver @relation(fields: [driverId], references: [id])
  test   Test   @relation(fields: [testId], references: [id])

  @@map("test_results")
}

// Availability Slots
model AvailabilitySlot {
  id        String   @id @default(uuid())
  driverId  String   @map("driver_id")
  startTime DateTime @map("start_time")
  endTime   DateTime @map("end_time")
  createdAt DateTime @default(now()) @map("created_at")

  // Relations
  driver Driver @relation(fields: [driverId], references: [id])
```

```
  @@map("availability_slots")
}

// Assignments
model Assignment {
  id              String          @id @default(uuid())
  managerId       String          @map("manager_id")
  title           String
  description     String?
  startTime       DateTime        @map("start_time")
  endTime         DateTime        @map("end_time")
  status          AssignmentStatus @default(draft)
  selectedDriverId String?        @map("selected_driver_id")
  createdAt       DateTime        @default(now()) @map("created_at")
  updatedAt       DateTime        @updatedAt @map("updated_at")

  // Relations
  manager         Manager         @relation(fields: [managerId], references: [id])
  selectedDriver  Driver?         @relation(fields: [selectedDriverId],
references: [id])
  dispatchResponses DispatchResponse[]
  driverAssignments DriverAssignment[]

  @@map("assignments")
}

// Dispatch Responses
model DispatchResponse {
  id          String      @id @default(uuid())
  assignmentId String      @map("assignment_id")
  driverId    String      @map("driver_id")
  response    ResponseType @default(pending)
  respondedAt  DateTime?   @map("responded_at")
  createdAt   DateTime    @default(now()) @map("created_at")

  // Relations
  assignment Assignment @relation(fields: [assignmentId], references: [id])
  driver    Driver    @relation(fields: [driverId], references: [id])

  @@unique([assignmentId, driverId])
  @@map("dispatch_responses")
}

// Driver Assignments (for overlap detection)
model DriverAssignment {
  id          String              @id @default(uuid())
```

```
  driverId     String                @map("driver_id")
  assignmentId String                @map("assignment_id")
  startTime    DateTime              @map("start_time")
  endTime      DateTime              @map("end_time")
  status       DriverAssignmentStatus  @default(confirmed)
  createdAt    DateTime              @default(now()) @map("created_at")

  // Relations
  driver     Driver     @relation(fields: [driverId], references: [id])
  assignment Assignment @relation(fields: [assignmentId], references: [id])

  @@map("driver_assignments")
}
```