Backend Implementation Plan - FINAL

Team: Backend Developer
Duration: 2-3 hours
Dependencies: None - Start immediately
Changes: Add 4 new endpoints only (no modifications to existing code)

Overview

Add 4 missing dashboard/utility endpoints to support frontend integration while maintaining existing code structure and patterns.

Step-by-Step Implementation

Step 1: Add Patient Dashboard Endpoint

Time: 30 minutes

1.1 Open Patient Controller **File**: `backend/src/controllers/patient.controller.js`

1.2 Add Method (before `module.exports`)

```
/**
 * Get patient dashboard data (aggregated)
 * @route GET /api/patient/dashboard
 */
async getDashboard(req, res, next) {
  try {
    const patientId = req.user.patientId; // From auth middleware

    const today = new Date();
    today.setHours(0, 0, 0, 0);

    // Aggregate data from existing queries
    const [upcoming, pastCount, recordsCount] = await Promise.all([
      // Get next 5 upcoming appointments
      prisma.appointment.findMany({
        where: {
          patientId,
          appointmentDate: { gte: today },
          status: { in: ['Pending', 'Confirmed'] }
        },
        take: 5,
        orderBy: { appointmentDate: 'asc' },
        include: {
          doctor: {
```

```
            include: {
              person: {
                select: { fullName: true }
              },
              specialty: {
                select: { specialtyName: true }
              }
            }
          },
          schedule: {
            include: { room: true }
          }
        }
      }),

      // Count past completed appointments
      prisma.appointment.count({
        where: {
          patientId,
          appointmentDate: { lt: today },
          status: 'Completed'
        }
      }),

      // Count medical records
      prisma.medicalRecord.count({
        where: { patientId }
      })
    ]);

    return res.json({
      success: true,
      data: {
        upcomingAppointments: upcoming,
        stats: {
          totalPastAppointments: pastCount,
          totalMedicalRecords: recordsCount
        }
      }
    });

  } catch (error) {
    console.error('Patient dashboard error:', error);
    next(error);
  }
}
```

1.3 Register Route **File**: `backend/src/routes/patient.routes.js`

Add BEFORE `module.exports`:

```
// Dashboard
router.get('/dashboard', patientController.getDashboard);
```

1.4 Test

```
# Using your API client (Postman/Thunder Client)
GET http://localhost:3000/api/patient/dashboard
Authorization: Bearer <patient_token>

# Expected Response:
{
  "success": true,
  "data": {
    "upcomingAppointments": [...],
    "stats": {
      "totalPastAppointments": 5,
      "totalMedicalRecords": 3
    }
  }
}
```

Step 2: Add Doctor Dashboard Endpoint

Time: 30 minutes

2.1 Open Doctor Controller **File**: `backend/src/controllers/doctor.controller.js`

2.2 Add Method (before `module.exports`)

```
/**
 * Get doctor dashboard data (aggregated)
 * @route GET /api/doctor/dashboard
 */
async getDashboard(req, res, next) {
  try {
    const doctorId = req.user.doctorId;

    const today = new Date();
    today.setHours(0, 0, 0, 0);
    const tomorrow = new Date(today);
    tomorrow.setDate(tomorrow.getDate() + 1);
    const weekEnd = new Date(today);
    weekEnd.setDate(weekEnd.getDate() + 7);
```

```
// Aggregate statistics and data
const [todayCount, weekAppointments, scheduleCount] = await Promise.all([
  // Count today's confirmed appointments
  prisma.appointment.count({
    where: {
      doctorId,
      appointmentDate: { gte: today, lt: tomorrow },
      status: 'Confirmed'
    }
  }),

  // Get next week's appointments
  prisma.appointment.findMany({
    where: {
      doctorId,
      appointmentDate: { gte: today, lt: weekEnd },
      status: { in: ['Pending', 'Confirmed'] }
    },
    take: 10,
    orderBy: { appointmentDate: 'asc' },
    include: {
      patient: {
        include: {
          person: {
            select: {
              fullName: true,
              phoneNumber: true,
              gender: true
            }
          }
        }
      },
      schedule: {
        include: { room: true }
      }
    }
  }),

  // Count total schedule entries
  prisma.schedule.count({
    where: { doctorId }
  })
]);

return res.json({
  success: true,
  data: {
    stats: {
```

```
          todayAppointments: todayCount,
          totalScheduleSlots: scheduleCount
        },
        upcomingAppointments: weekAppointments
      }
    });

  } catch (error) {
    console.error('Doctor dashboard error:', error);
    next(error);
  }
}
```

2.3 Register Route File: `backend/src/routes/doctor.routes.js`

Add BEFORE `module.exports`:

```
// Dashboard
router.get('/dashboard', doctorController.getDashboard);
```

2.4 Test

```
GET http://localhost:3000/api/doctor/dashboard
Authorization: Bearer <doctor_token>

# Expected Response:
{
  "success": true,
  "data": {
    "stats": {
      "todayAppointments": 3,
      "totalScheduleSlots": 15
    },
    "upcomingAppointments": [...]
  }
}
```

Step 3: Add Doctor Patients-in-Clinic Endpoint

Time: 30 minutes

3.1 Open Doctor Controller File: `backend/src/controllers/doctor.controller.js`

3.2 Add Method (before `module.exports`)

```
/**
 * Get patients currently in clinic (today's confirmed appointments)
 * @route GET /api/doctor/patients-in-clinic
```

```
 */
async getPatientsInClinic(req, res, next) {
  try {
    const doctorId = req.user.doctorId;

    const today = new Date();
    today.setHours(0, 0, 0, 0);
    const tomorrow = new Date(today);
    tomorrow.setDate(tomorrow.getDate() + 1);

    const patientsInClinic = await prisma.appointment.findMany({
      where: {
        doctorId,
        appointmentDate: { gte: today, lt: tomorrow },
        status: 'Confirmed' // Checked-in patients only
      },
      orderBy: {
        schedule: {
          startTime: 'asc'
        }
      },
      include: {
        patient: {
          include: {
            person: {
              select: {
                fullName: true,
                phoneNumber: true,
                gender: true
              }
            }
          }
        },
        schedule: {
          include: {
            room: true
          }
        }
      }
    });

    return res.json({
      success: true,
      data: patientsInClinic
    });

  } catch (error) {
    console.error('Patients in clinic error:', error);
```

```
    next(error);
  }
}
```

3.3 Register Route **File**: backend/src/routes/doctor.routes.js

Add AFTER dashboard route:

```
// Patients in clinic
router.get('/patients-in-clinic', doctorController.getPatientsInClinic);
```

3.4 Test

```
GET http://localhost:3000/api/doctor/patients-in-clinic
Authorization: Bearer <doctor_token>

# Expected Response:
{
  "success": true,
  "data": [
    {
      "id": 1,
      "appointmentDate": "2024-12-05T08:00:00.000Z",
      "status": "Confirmed",
      "patient": {
        "person": {
          "fullName": "Ahmed Ali",
          "phoneNumber": "01234567890",
          "gender": "Male"
        }
      },
      "schedule": {
        "startTime": "08:00:00",
        "endTime": "09:00:00",
        "room": {
          "roomNumber": "101",
          "roomType": "Examination"
        }
      }
    }
  ]
}
```

Step 4: Add Reception Dashboard Endpoint

Time: 30 minutes

4.1 Open Reception Controller **File**: backend/src/controllers/reception.controller.js

4.2 Add Method (AFTER middleware, BEFORE other methods)

```
/**
 * Get receptionist dashboard data (aggregated)
 * @route GET /api/reception/dashboard
 */
async getDashboard(req, res, next) {
  try {
    const today = new Date();
    today.setHours(0, 0, 0, 0);
    const tomorrow = new Date(today);
    tomorrow.setDate(tomorrow.getDate() + 1);

    // Aggregate today's statistics
    const [totalToday, pending, confirmed] = await Promise.all([
      // Total appointments today
      prisma.appointment.count({
        where: {
          appointmentDate: { gte: today, lt: tomorrow }
        }
      }),

      // Pending appointments (all future)
      prisma.appointment.count({
        where: {
          appointmentDate: { gte: today },
          status: 'Pending'
        }
      }),

      // Today's confirmed appointments (checked-in)
      prisma.appointment.findMany({
        where: {
          appointmentDate: { gte: today, lt: tomorrow },
          status: 'Confirmed'
        },
        take: 15,
        orderBy: {
          schedule: {
            startTime: 'asc'
          }
        },
        include: {
          patient: {
            include: {
              person: {
                select: {
                  fullName: true,
```

```
                phoneNumber: true
              }
            }
          }
        },
        doctor: {
          include: {
            person: {
              select: { fullName: true }
            },
            specialty: {
              select: { specialtyName: true }
            }
          }
        },
        schedule: {
          include: { room: true }
        }
      }
    })
  ]);

  return res.json({
    success: true,
    data: {
      stats: {
        todayTotal: totalToday,
        pendingAppointments: pending
      },
      todayAppointments: confirmed
    }
  });

} catch (error) {
  console.error('Reception dashboard error:', error);
  next(error);
}
}
```

4.3 Register Route **File**: `backend/src/routes/reception.routes.js`

Add AFTER middleware declarations, BEFORE other routes:

```
// Dashboard (add at the top, after middleware)
router.get('/dashboard', receptionController.getDashboard);
```

4.4 Test

```
GET http://localhost:3000/api/reception/dashboard
Authorization: Bearer <receptionist_token>

# Expected Response:
{
  "success": true,
  "data": {
    "stats": {
      "todayTotal": 12,
      "pendingAppointments": 8
    },
    "todayAppointments": [...]
  }
}
```

Verification Checklist

After completing all steps:

Functional Testing

- ☐ Patient dashboard returns correct data
- ☐ Doctor dashboard shows stats and appointments
- ☐ Doctor patients-in-clinic lists today's confirmed only
- ☐ Reception dashboard aggregates correctly

Error Testing

- ☐ All endpoints return 401 without token
- ☐ All endpoints return proper error messages
- ☐ Database errors are handled gracefully

Performance Testing

- ☐ All queries execute in <500ms
- ☐ No N+1 query issues
- ☐ Proper indexing on date columns

Rollback Plan

If issues occur, simply: 1. Comment out new controller methods 2. Comment out new route registrations 3. Restart server

No database changes were made, so rollback is safe.

Completion Criteria

All 4 endpoints implemented
All routes registered
All endpoints tested
Response structure matches existing patterns
Error handling consistent
Code passes review

Next Steps

After backend completion: 1. Notify frontend team that dashboard endpoints are ready 2. Provide API documentation/examples 3. Monitor logs for any issues during integration testing

---

Notes

- All new code follows existing patterns
- No breaking changes
- No database migrations needed
- Ready for production deployment