Frontend Implementation Plan - FINAL

Team: Frontend Developer
Duration: 10-11 hours
Dependencies: Backend dashboard endpoints (for testing Phase 4 only)
Changes: 6 service files + 9 component files + 3 new utility files

---

Overview

Complete frontend integration with backend API by: 1. Fixing critical mismatches 2. Adding architectural improvements 3. Reorganizing service layer 4. Updating components

---

Phase 1: Foundation & Configuration (2 hours)

Step 1.1: Enhanced Response Interceptor

Time: 30 minutes

File: `frontend/src/api/config.js`

Current code (around line 25):

```
// Response interceptor - handle errors
api.interceptors.response.use(
  (response) => response,
  (error) => {
    if (error.response?.status === 401) {
      localStorage.removeItem('token');
      localStorage.removeItem('user');
      window.location.href = '/login';
    }
    return Promise.reject(error);
  }
);
```

Replace with:

```
// Response interceptor - unwrap data and handle errors
api.interceptors.response.use(
  (response) => {
    // Auto-unwrap response.data for cleaner service code
    return response.data;
  },
  (error) => {
    // Handle 401 - Unauthorized
    if (error.response?.status === 401) {
      localStorage.removeItem('token');
      localStorage.removeItem('user');
```

```
      localStorage.removeItem('userRole');
      window.location.href = '/login';
    }

    // Enhance error object with backend message
    if (error.response?.data) {
      error.message = error.response.data.error || error.message;
    }

    return Promise.reject(error);
  }
);
```

Step 1.2: Error Handling Utility

Time: 20 minutes

Create file: `frontend/src/utils/errorHandler.js`

```
/**
 * Centralized error handling utility
 */

export const handleServiceError = (error, fallbackMessage = 'An error occurred') => {
  // Extract error message from various sources
  const errorMessage =
    error.response?.data?.error ||  // Backend error
    error.message ||                 // JavaScript error
    fallbackMessage;                 // Fallback

  // Log error for debugging (development only)
  if (import.meta.env.DEV) {
    console.error('Service Error:', {
      message: errorMessage,
      response: error.response,
      stack: error.stack
    });
  }

  return errorMessage;
};

// Export for use in components
export default handleServiceError;
```

Step 1.3: Environment Configuration

Time: 15 minutes

Create file: `frontend/.env.local` (or `.env`)

```
# API Configuration
VITE_API_BASE_URL=http://localhost:3000/api

# Environment
VITE_ENV=development
```

Update file: `frontend/src/api/config.js` (line 4-5)

Current:

```
const api = axios.create({
    baseURL: 'http://172.24.65.132:3000/api',
    headers: {
        'Content-Type': 'application/json'
    }
});
```

Replace with:

```
const api = axios.create({
    baseURL: import.meta.env.VITE_API_BASE_URL || 'http://localhost:3000/api',
    headers: {
        'Content-Type': 'application/json'
    }
});
```

Step 1.4: Validation Utilities

Time: 30 minutes

Create file: `frontend/src/utils/validators.js`

```
/**
 * Form validation utilities
 */

export const validateEmail = (email) => {
  const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
  return emailRegex.test(email);
};

export const validatePhone = (phone) => {
  // Egyptian phone: 11 digits starting with 01
  const phoneRegex = /^01[0-2,5]{1}[0-9]{8}$/;
  return phoneRegex.test(phone);
```

```
};

export const validatePassword = (password) => {
  // At least 6 characters
  return password && password.length >= 6;
};

export const validateOTP = (otp) => {
  // 6 digit code
  const otpRegex = /^\d{6}$/;
  return otpRegex.test(otp);
};

export const validateRequired = (value) => {
  return value && value.trim().length > 0;
};

export default {
  validateEmail,
  validatePhone,
  validatePassword,
  validateOTP,
  validateRequired
};
```

Step 1.5: Date Utilities

Time: 25 minutes

Create file: `frontend/src/utils/dateHelpers.js`

```
/**
 * Date formatting and manipulation utilities
 */

export const formatDate = (date) => {
  if (!date) return '';
  return new Date(date).toLocaleDateString('en-GB', {
    day: '2-digit',
    month: '2-digit',
    year: 'numeric'
  });
};

export const formatTime = (time) => {
  if (!time) return '';
  return new Date(`2000-01-01T${time}`).toLocaleTimeString('en-GB', {
```

```
    hour: '2-digit',
    minute: '2-digit'
  });
};

export const formatDateTime = (datetime) => {
  if (!datetime) return '';
  return new Date(datetime).toLocaleString('en-GB', {
    day: '2-digit',
    month: '2-digit',
    year: 'numeric',
    hour: '2-digit',
    minute: '2-digit'
  });
};

export const toISODate = (date) => {
  if (!date) return '';
  return new Date(date).toISOString().split('T')[0];
};

export const isToday = (date) => {
  const today = new Date();
  const checkDate = new Date(date);
  return checkDate.toDateString() === today.toDateString();
};

export const isPast = (date) => {
  return new Date(date) < new Date();
};

export default {
  formatDate,
  formatTime,
  formatDateTime,
  toISODate,
  isToday,
  isPast
};
```

Phase 2: Critical Service Fixes (3 hours)

Step 2.1: Fix Auth Service

Time: 45 minutes

File: `frontend/src/api/authService.js`

Changes needed:

1. **DELETE `sendOTP` function** (lines 28-31):

```
//  REMOVE THIS ENTIRE FUNCTION
sendOTP: async (email) => {
    const response = await api.post('/auth/send-otp', { email });
    return response.data;
},
```

2. **FIX `verifyOTP` function** (lines 34-37):

Before:

```
verifyOTP: async (email, code) => {
    const response = await api.post('/auth/verify-otp', { email, code });
    return response.data;
},
```

After:

```
verifyOTP: async (email, otpCode) => {
    const result = await api.post('/auth/verify-otp', { email, otpCode });
    return result.data; // Interceptor already unwrapped response.data
},
```

3. **UPDATE all service methods** to use new interceptor:

Before (example):

```
register: async (userData) => {
    const response = await api.post('/auth/register', userData);
    if (response.data.token) {
        localStorage.setItem('token', response.data.token);
        // ...
    }
    return response.data;
},
```

After:

```
register: async (userData) => {
    const result = await api.post('/auth/register', userData);
    // result is already unwrapped {success, data, message}
    if (result.data?.token) {
        localStorage.setItem('token', result.data.token);
        localStorage.setItem('user', JSON.stringify(result.data.user));
        localStorage.setItem('userRole', result.data.user.role.toLowerCase());
    }
    return result.data;
},
```

Apply same pattern to: `login`, `getCurrentUser`

Final authService.js should be:

```
import api from './config';

export const authService = {
    // Register new patient
    register: async (userData) => {
        const result = await api.post('/auth/register', userData);
        if (result.data?.token) {
            localStorage.setItem('token', result.data.token);
            localStorage.setItem('user', JSON.stringify(result.data.user));
            localStorage.setItem('userRole', result.data.user.role.toLowerCase());
        }
        return result.data;
    },

    // Login
    login: async (email, password) => {
        const result = await api.post('/auth/login', { email, password });
        if (result.data?.token) {
            localStorage.setItem('token', result.data.token);
            localStorage.setItem('user', JSON.stringify(result.data.user));
            localStorage.setItem('userRole', result.data.user.role.toLowerCase());
        }
        return result.data;
    },

    // Verify OTP (FIXED parameter name)
    verifyOTP: async (email, otpCode) => {
        const result = await api.post('/auth/verify-otp', { email, otpCode });
        return result.data;
    },

    // Resend OTP
    resendOTP: async (email) => {
        const result = await api.post('/auth/resend-otp', { email });
        return result.data;
    },

    // Get current user
    getCurrentUser: async () => {
        const result = await api.get('/auth/me');
        return result.data.user;
    },

    // Logout
    logout: () => {
        localStorage.removeItem('token');
        localStorage.removeItem('user');
        localStorage.removeItem('userRole');
```

```
    },

    // Helper methods (no changes)
    getStoredUser: () => {
        const user = localStorage.getItem('user');
        return user ? JSON.parse(user) : null;
    },

    getToken: () => {
        return localStorage.getItem('token');
    },

    isAuthenticated: () => {
        return !!localStorage.getItem('token');
    }
};
```

Step 2.2: Fix Receptionist Service (CRITICAL)

Time: 1 hour

File: `frontend/src/api/receptionistService.js`

COMPLETE REWRITE (all paths wrong):

```
import api from './config';

export const receptionistService = {
    // Get dashboard
    getDashboard: async () => {
        const result = await api.get('/reception/dashboard');
        return result.data;
    },

    // Get upcoming appointments (filters optional)
    getUpcomingAppointments: async (filters = {}) => {
        const result = await api.get('/reception/appointments/upcoming', {
            params: filters
        });
        return result.data;
    },

    // Get doctor's appointments for specific date
    getDoctorAppointments: async (doctorId, date) => {
        const result = await api.get(`/reception/appointments/doctor/${doctorId}`, {
            params: { date }
        });
        return result.data;
```

```
    },

    // Search patients (by name, phone, or ID)
    searchPatients: async (query) => {
        const result = await api.get('/reception/patients/search', {
            params: { q: query }
        });
        return result.data;
    },

    // Add walk-in patient (no user account)
    addWalkInPatient: async (patientData) => {
        const result = await api.post('/reception/patients/walk-in', patientData);
        return result.data;
    },

    // Book appointment for any patient
    bookAppointment: async (appointmentData) => {
        const result = await api.post('/reception/appointments', appointmentData);
        return result.data;
    },

    // Check-in patient (NOTICE: 'checkin' not 'check-in')
    checkInPatient: async (appointmentId) => {
        const result = await api.patch(`/reception/appointments/${appointmentId}/checkin`);
        return result.data;
    },

    // Cancel appointment
    cancelAppointment: async (appointmentId) => {
        const result = await api.delete(`/reception/appointments/${appointmentId}`);
        return result.data;
    }
};
```

Step 2.3: Delete Appointment Service

Time: 15 minutes

Option 1: DELETE the file (Recommended)

```
rm frontend/src/api/appointmentService.js
```

Then update imports in components that use it (we'll handle this in Phase 4)

Option 2: Keep for admin-only (if admin needs it): Rename to `adminAppointmentService.js` and update to use `/admin/appointments`

Step 2.4: Delete Supporting Services

Time: 15 minutes

DELETE the file:

```
rm frontend/src/api/supportingServices.js
```

All specialty/room/schedule operations will be in `adminService` or role-specific services.

Phase 3: Service Enhancement (3-4 hours)

Step 3.1: Enhance Patient Service

Time: 1.5 hours

File: `frontend/src/api/patientService.js`

COMPLETE REWRITE:

```
import api from './config';

export const patientService = {
    // ===== Dashboard =====
    getDashboard: async () => {
        const result = await api.get('/patient/dashboard');
        return result.data;
    },

    // ===== Profile =====
    getProfile: async () => {
        const result = await api.get('/patient/profile');
        return result.data;
    },

    updatePhone: async (phoneNumber) => {
        const result = await api.put('/patient/profile/phone', { phoneNumber });
        return result.data;
    },

    requestEmailUpdateOTP: async (newEmail) => {
        const result = await api.post('/patient/profile/email/request-otp', { newEmail });
        return result.data;
    },

    updateEmail: async (newEmail, otpCode) => {
        const result = await api.put('/patient/profile/email', { newEmail, otpCode });
        return result.data;
    },
```

```
    updatePassword: async (currentPassword, newPassword) => {
        const result = await api.put('/patient/profile/password', {
            currentPassword,
            newPassword
        });
        return result.data;
    },

    // ===== Booking Flow =====
    getSpecialties: async () => {
        const result = await api.get('/patient/specialties');
        return result.data;
    },

    getAvailableDoctors: async (specialtyId, date) => {
        const result = await api.get('/patient/doctors', {
            params: { specialtyId, date }
        });
        return result.data;
    },

    bookAppointment: async (appointmentData) => {
        const result = await api.post('/patient/appointments', appointmentData);
        return result.data;
    },

    // ===== Appointments =====
    getUpcomingAppointments: async () => {
        const result = await api.get('/patient/appointments/upcoming');
        return result.data;
    },

    getPastAppointments: async () => {
        const result = await api.get('/patient/appointments/past');
        return result.data;
    },

    cancelAppointment: async (appointmentId) => {
        const result = await api.delete(`/patient/appointments/${appointmentId}`);
        return result.data;
    },

    // ===== Medical Records =====
    getMedicalRecords: async () => {
        const result = await api.get('/patient/medical-records');
        return result.data;
    }
};
```

Step 3.2: Enhance Doctor Service

Time: 1.5 hours

File: `frontend/src/api/doctorService.js`

COMPLETE REWRITE:

```
import api from './config';

export const doctorService = {
    // ===== Dashboard =====
    getDashboard: async () => {
        const result = await api.get('/doctor/dashboard');
        return result.data;
    },

    getPatientsInClinic: async () => {
        const result = await api.get('/doctor/patients-in-clinic');
        return result.data;
    },

    // ===== Profile =====
    getProfile: async () => {
        const result = await api.get('/doctor/profile');
        return result.data;
    },

    updatePhone: async (phoneNumber) => {
        const result = await api.put('/doctor/profile/phone', { phoneNumber });
        return result.data;
    },

    updatePassword: async (currentPassword, newPassword) => {
        const result = await api.put('/doctor/profile/password', {
            currentPassword,
            newPassword
        });
        return result.data;
    },

    // ===== Schedule Management =====
    getMySchedule: async () => {
        const result = await api.get('/doctor/schedule');
        return result.data;
    },
```

```
    addSchedule: async (scheduleData) => {
        const result = await api.post('/doctor/schedule', scheduleData);
        return result.data;
    },

    updateSchedule: async (scheduleId, scheduleData) => {
        const result = await api.put(`/doctor/schedule/${scheduleId}`, scheduleData);
        return result.data;
    },

    deleteSchedule: async (scheduleId) => {
        const result = await api.delete(`/doctor/schedule/${scheduleId}`);
        return result.data;
    },

    // ===== Appointments =====
    getTodayAppointments: async () => {
        const result = await api.get('/doctor/appointments/today');
        return result.data;
    },

    getWeekAppointments: async (startDate) => {
        const result = await api.get('/doctor/appointments/week', {
            params: { startDate }
        });
        return result.data;
    },

    getAppointmentDetails: async (appointmentId) => {
        const result = await api.get(`/doctor/appointments/${appointmentId}`);
        return result.data;
    },

    // ===== Medical Records =====
    // FIXED: Uses appointmentId, not patientId
    addMedicalRecord: async (appointmentId, recordData) => {
        const result = await api.post(
            `/doctor/appointments/${appointmentId}/medical-record`,
            recordData
        );
        return result.data;
    }
};
```

Step 3.3: Enhance Admin Service

Time: 45 minutes

File: `frontend/src/api/adminService.js`

ADD missing methods (after existing ones):

```
// ... existing admin, doctor, patient, receptionist methods ...

// ===== Specialties =====
getAllSpecialties: async () => {
    const result = await api.get('/admin/specialties');
    return result.data;
},

getSpecialtyById: async (id) => {
    const result = await api.get(`/admin/specialties/${id}`);
    return result.data;
},

createSpecialty: async (data) => {
    const result = await api.post('/admin/specialties', data);
    return result.data;
},

updateSpecialty: async (id, data) => {
    const result = await api.put(`/admin/specialties/${id}`, data);
    return result.data;
},

deleteSpecialty: async (id) => {
    const result = await api.delete(`/admin/specialties/${id}`);
    return result.data;
},

// ===== Rooms =====
getAllRooms: async () => {
    const result = await api.get('/admin/rooms');
    return result.data;
},

getRoomById: async (id) => {
    const result = await api.get(`/admin/rooms/${id}`);
    return result.data;
},

createRoom: async (data) => {
    const result = await api.post('/admin/rooms', data);
```

```
    return result.data;
},

updateRoom: async (id, data) => {
    const result = await api.put(`/admin/rooms/${id}`, data);
    return result.data;
},

deleteRoom: async (id) => {
    const result = await api.delete(`/admin/rooms/${id}`);
    return result.data;
},

// ===== Schedules =====
getAllSchedules: async (filters = {}) => {
    const result = await api.get('/admin/schedules', { params: filters });
    return result.data;
},

getScheduleById: async (id) => {
    const result = await api.get(`/admin/schedules/${id}`);
    return result.data;
},

createSchedule: async (data) => {
    const result = await api.post('/admin/schedules', data);
    return result.data;
},

updateSchedule: async (id, data) => {
    const result = await api.put(`/admin/schedules/${id}`, data);
    return result.data;
},

deleteSchedule: async (id) => {
    const result = await api.delete(`/admin/schedules/${id}`);
    return result.data;
},

// ===== Appointments (Admin CRUD) =====
getAllAppointments: async (filters = {}) => {
    const result = await api.get('/admin/appointments', { params: filters });
    return result.data;
},

getAppointmentById: async (id) => {
    const result = await api.get(`/admin/appointments/${id}`);
    return result.data;
```

```
},

createAppointment: async (data) => {
    const result = await api.post('/admin/appointments', data);
    return result.data;
},

updateAppointment: async (id, data) => {
    const result = await api.put(`/admin/appointments/${id}`, data);
    return result.data;
},

deleteAppointment: async (id) => {
    const result = await api.delete(`/admin/appointments/${id}`);
    return result.data;
},

// ===== Medical Records (Admin CRUD) =====
getAllMedicalRecords: async (filters = {}) => {
    const result = await api.get('/admin/medical-records', { params: filters });
    return result.data;
},

getMedicalRecordById: async (id) => {
    const result = await api.get(`/admin/medical-records/${id}`);
    return result.data;
},

createMedicalRecord: async (data) => {
    const result = await api.post('/admin/medical-records', data);
    return result.data;
},

updateMedicalRecord: async (id, data) => {
    const result = await api.put(`/admin/medical-records/${id}`, data);
    return result.data;
},

deleteMedicalRecord: async (id) => {
    const result = await api.delete(`/admin/medical-records/${id}`);
    return result.data;
}
```

Phase 4: Component Updates (2 hours)

Step 4.1: Update RegisterPage

Time: 20 minutes

File: `frontend/src/pages/RegisterPage.jsx`

Changes:

1. **Remove** `authService.sendOTP()` calls (lines 39, 82)
2. **Update** `verifyOTP()` parameter

Find (around line 39):

```
await authService.sendOTP(formData.email);
```

DELETE this line - OTP is sent automatically with register

Find (around line 56):

```
await authService.verifyOTP(formData.email, otp);
```

Change to:

```
await authService.verifyOTP(formData.email, otp); // 'otp' variable name is fine
```

Find (around line 82):

```
await authService.sendOTP(formData.email);
```

REPLACE with:

```
await authService.resendOTP(formData.email);
```

Step 4.2: Update Settings Pages (4 files)

Time: 40 minutes (10 min each)

Files: - `DoctorSettings.jsx` - `PatientSettings.jsx` - `ReceptionistSettings.jsx` - `AdminSettings.jsx`

Same changes for all:

1. **Remove** `service.updateSettings()` calls
2. **Use** individual profile methods instead
3. **Fix** OTP calls

Example for PatientSettings.jsx:

Find (around line 56):

```
await authService.sendOTP(email);
```

REPLACE:

```
await authService.resendOTP(email);
```

Find (around line 93):

```
await patientService.updateSettings(updateData);
```

REPLACE:

```
// Update phone
if (updateData.phoneNumber) {
    await patientService.updatePhone(updateData.phoneNumber);
}

// Update password
if (updateData.newPassword) {
    await patientService.updatePassword(updateData.currentPassword, updateData.newPassword);
}
```

Apply same pattern to other settings pages using their respective services.

Step 4.3: Update ReceptionistDashboard

Time: 30 minutes

File: `frontend/src/pages/ReceptionistDashboard.jsx`

Changes:

1. **Import change** (line 4):

```
// Remove appointmentService import
// import { appointmentService } from '../api/appointmentService';
```

2. **Find** (line 44, 199):

```
receptionistService.getTodayAppointmentsByDoctor()
```

REPLACE:

```
receptionistService.getUpcomingAppointments()
```

3. **Find** (line 89):

```
await appointmentService.checkIn(appointmentId);
```

REPLACE:

```
await receptionistService.checkInPatient(appointmentId);
```

4. **Find** (line 189):

```
await appointmentService.book({...});
```

REPLACE:

```
await receptionistService.bookAppointment({...});
```

Step 4.4: Update DoctorMedicalRecordEditor

Time: 15 minutes

File: `frontend/src/pages/DoctorMedicalRecordEditor.jsx`

CRITICAL: This component needs appointmentId instead of patientId

Changes needed depend on how component receives data

Option 1: If appointment is passed as prop:

```
const { appointment } = props; // Get full appointment object
const appointmentId = appointment.id;

// Then use:
await doctorService.addMedicalRecord(appointmentId, recordData);
```

Option 2: If only patient is passed: Component needs refactoring to receive appointmentId. This may require parent component changes.

Step 4.5: Update PatientDashboard

Time: 15 minutes

File: `frontend/src/pages/PatientDashboard.jsx`

Find (line 57):

```
await appointmentService.cancel(id);
```

REPLACE:

```
await patientService.cancelAppointment(id);
```

Remove appointmentService import (line 3)

Testing Phase

Test Each Phase

After Phase 1: - [ ] App still runs - [ ] Config loads correctly - [ ] Utilities work

After Phase 2: - [ ] Login works - [ ] Registration + OTP verification works - [ ] No console errors

After Phase 3: - [ ] All services import correctly - [ ] No missing method errors

After Phase 4 (requires backend completion): - [ ] All dashboards load data - [ ] All CRUD operations work - [ ] No integration errors

---

Final Checklist

Code Quality

- ☐ No hardcoded URLs
- ☐ Consistent error handling
- ☐ All promises handled
- ☐ No console.log in production code

Functionality

- ☐ Auth flow works end-to-end
- ☐ All roles can access their features
- ☐ Dashboard data displays correctly
- ☐ Form validations work

Performance

- ☐ No unnecessary re-renders
- ☐ Loading states implemented
- ☐ Error states handled

Completion Criteria

All 6 service files updated
All 9 component files fixed
3 utility files created
Environment config added
All tests passing
No console errors
Integration with backend successful

Notes

- Can start Phases 1-3 immediately (don't need backend)
- Phase 4 component testing requires backend dashboard endpoints
- Keep backups before making changes
- Test incrementally after each phase