

FRONTEND IMPLEMENTATION PLAN - 5 TEAM MEMBERS

PROJECT GOAL

Build complete frontend UI for all user roles (Patient, Doctor, Admin) using the **existing** backend APIs - NO backend changes needed.

PHASE 1: FOUNDATION & INFRASTRUCTURE

* MEMBER 1: Core Infrastructure & Authentication

Objective: Set up shared services and authentication flow

Tasks Breakdown:

A. Create Shared API Service (2 hours)

- Create `Services/ApiService.cs` - HTTP client wrapper
- Methods: `GetAsync<T>`, `PostAsync<T,R>`, `PutAsync<T>`, `DeleteAsync(int)`
- Session management helpers: `GetCurrentUser()`, `SetCurrentUser()`, `ClearSession()`

B. Configure Application (1 hour)

- Update `Program.cs`:
 - Add session configuration
 - Register `HttpClient` with API base URL
 - Register `ApiService` as scoped
- Add `appsettings.json` entry for API URL

C. Create View Models (1 hour)

- `Models/LoginViewModel.cs`
- `Models/RegisterViewModel.cs`
- `Models/UserSession.cs`

D. Update Authentication Pages (3 hours)

- Update `AccountController`:
 - `Login([FromBody] LoginDto)` action (GET/POST) - call `/api/auth/login`
 - `Register([FromBody] RegisterDto)` action (GET/POST) - call `/api/auth/register`
 - `Logout` action - clear session
- Update `Login.cshtml`:

- Bind to ViewModel
- Add validation messages
- Add error display
- Update Register.cshtml:
 - Bind to ViewModel
 - Add validation messages
 - Add error display

E. Create Shared Layout Components (2 hours)

- Update _Layout.cshtml:
 - Add user menu (show when logged in)
 - Add logout button
 - Add role-based navigation
- Create Views/Shared/_UserNav.cshtml partial
- Create Views/Shared/_Alerts.cshtml partial for messages

F. Create Base Controller (1 hour)

- Create Controllers/BaseController.cs
- Add helper methods: `GetCurrentUserId()`, `GetCurrentUserRole()`, `IsAuthenticated()`
- All other controllers will inherit from this

Deliverables:

- ✓ Working login/register/logout
- ✓ Session management
- ✓ ApiService ready for team
- ✓ Base infrastructure for all pages

PHASE 2: ROLE-BASED DASHBOARDS

* MEMBER 2: Patient Dashboard & Booking

Objective: Build complete patient experience

Tasks Breakdown:

A. Patient Dashboard (3 hours)

- Create `Controllers/PatientController.cs`
- Create `Views/Patient/Dashboard.cshtml`
- API calls needed:
 - GET `/api/patients/profile/{userId}` - patient info
 - GET `/appointment/my?patientId={id}` - upcoming appointments
- Display:
 - Welcome card with patient name
 - Upcoming appointments (cards)
 - Quick stats (total appointments, next appointment)
 - Navigation buttons

B. Appointment Booking Page (4 hours)

- Update `AppointmentController.cs`
- Update `Book.cshtml`
- API calls needed:
 - GET `/api/doctors` - get all doctors
 - GET `/appointment/check` - check availability
 - POST `/appointment/add` - create booking
- Features:
 - Specialty dropdown (hardcoded or from API)
 - Doctor dropdown (filtered by specialty - JavaScript)
 - Date/time picker
 - Availability check (AJAX call)
 - Form validation

- Success message/redirect

C. My Appointments Page (2 hours)

- Create Views/Patient/Appointments.cshtml
- API call: GET /api/patients/{id}/appointments
- Display:
 - Appointment cards (doctor, date, time, status)
 - Filter by status (upcoming, past, cancelled)
 - Cancel button (calls POST /appointment/cancel/{id})
 - Status badges (color-coded)

D. Patient Profile Page (2 hours)

- Create Views/Patient/Profile.cshtml
- API calls:
 - GET /api/patients/profile/{userId} - load data
 - PUT /api/patients/{id} - update data
- Features:
 - Editable form (name, phone, DOB, gender)
 - View-only email
 - Save button
 - Success/error messages

Deliverables:

- ✓ Patient dashboard
- ✓ Working appointment booking
- ✓ Appointment management
- ✓ Profile editing

* MEMBER 3: Doctor Dashboard & Schedule

Objective: Build complete doctor experience

Tasks Breakdown:

A. Doctor Dashboard (3 hours)

- Create `Controllers/DoctorController.cs`
- Create `Views/Doctor/Dashboard.cshtml`
- API calls needed:
 - `GET /api/doctors/{id}` - doctor info
 - `GET /appointment/today?doctorId={id}` - today's appointments
- Display:
 - Welcome card
 - Today's appointments list
 - Quick stats (today's total, pending, confirmed)
 - Navigation to schedule

B. Doctor Schedule/Appointments (3 hours)

- Create `Views/Doctor/Appointments.cshtml`
- API call: Get all doctor appointments (filter by date)
- Display:
 - Appointments table/cards
 - Filters (date range, status)
 - Patient name, time, status
 - Action buttons (confirm, cancel, complete)
- Actions:
 - Confirm: `POST /appointment/confirm/{id}`
 - Cancel: `POST /appointment/cancel/{id}`

C. Appointment Details Modal (2 hours)

- Create partial view `_AppointmentDetails.cshtml`

- Display patient info
- Add notes field
- Update status options

D. Doctor Profile Page (2 hours)

- Create `Views/Doctor/Profile.cshtml`
- API calls:
 - `GET /api/doctors/{id}` - load data
 - `PUT /api/doctors/{id}` - update data
- Features:
 - Edit professional info (specialty, bio, qualifications)
 - View appointments count
 - Save button

Deliverables:

- ✓ Doctor dashboard
- ✓ Schedule management
- ✓ Appointment confirmation/cancellation
- ✓ Profile editing

* MEMBER 4: Admin Dashboard & Management

Objective: Build complete admin panel

Tasks Breakdown:

A. Admin Dashboard (3 hours)

- Create `AdminController.cs` (MVC side)
- Create `Views/Admin/Dashboard.cshtml`
- API calls:
 - `GET /admin/dashboard` - stats
 - `GET /admin/appointments` - recent appointments

- Display:
 - Stats cards (users, doctors, patients, appointments)
 - Recent appointments table
 - Pending approvals count
 - Quick navigation

B. User Management (3 hours)

- Create `Views/Admin/Users.cshtml`
- API calls:
 - `GET /admin/users` - list all users
 - `PUT /admin/users/{id}` - update user
 - `DELETE /admin/users/{id}` - delete user
- Features:
 - Users table (name, email, role, status)
 - Search/filter functionality
 - Edit button (inline or modal)
 - Delete button (with confirmation)
 - Toggle active/inactive status

C. Doctor Management (3 hours)

- Create `Views/Admin/Doctors.cshtml`
- API calls:
 - `GET /admin/doctors` - list all doctors
 - `POST /admin/doctors/approve/{id}` - approve doctor
 - `POST /api/doctors/{id}/toggle` - toggle status
- Features:
 - Doctors table (name, specialty, status)

- Approve/reject buttons
- View details
- Toggle active status

D. Appointment Management (2 hours)

- Create `Views/Admin/Appointments.cshtml`
- API call: `GET /admin/appointments?status=X&doctorId=Y&patientId=Z`
- Features:
 - Appointments table
 - Filters (status, doctor, patient, date)
 - Force cancel: `POST /admin/appointments/{id}/cancel`
 - View details

Deliverables:

- ✓ Admin dashboard with stats
- ✓ User management (CRUD)
- ✓ Doctor approval system
- ✓ Appointment oversight

PHASE 3: PUBLIC PAGES & ENHANCEMENTS

* MEMBER 5: Public Pages, Search & UX

Objective: Build public-facing pages and polish UI/UX

Tasks Breakdown:

A. Dynamic Doctor Listing (3 hours)

- Update `HomeController.cs` - add `Doctors` action
- Update `Doctors.cshtml`
- API call: `GET /api/doctors`
- Features:
 - Display all doctors as cards

- Search by name (JavaScript filter)
- Filter by specialty (dropdown)
- "View Profile" link
- "Book Appointment" button

B. Doctor Detail Page (2 hours)

- Create Views/Doctors/Details.cshtml
- API call: `GET /api/doctors/{id}`
- Display:
 - Doctor photo, name, specialty
 - Bio/qualifications
 - Work hours
 - "Book Appointment" button (redirect with pre-selected doctor)

C. Departments/Specialties Page (2 hours)

- Update Departments.cshtml
- Make it dynamic (if specialties API exists) or keep static
- Add "Find Doctors" button for each specialty
- Links to doctors filtered by specialty

D. Contact Form (1 hour)

- Update Contact.cshtml
- Add form submission (save to database or send email)
- Add validation
- Success message

E. Notification System (3 hours)

- Create notification bell icon in _Layout.cshtml
- API calls:
 - `GET /api/notifications/user/{userId}/unread` - get count

- `GET /api/notifications/user/{userId}` - get all
- `PATCH /api/notifications/{id}/read` - mark as read
- Features:
 - Bell icon with badge (unread count)
 - Dropdown with recent notifications
 - "Mark as read" functionality
 - Link to full notifications page

F. UI/UX Polish (3 hours)

- Add loading spinners (create `_Spinner.cshtml` partial)
- Create toast notification system (JavaScript)
- Add client-side validation to all forms
- Responsive design testing/fixes
- Add smooth transitions (CSS)
- Error page styling

Deliverables:

- ✓ Dynamic doctor listing with search
- ✓ Doctor detail pages
- ✓ Working contact form
- ✓ Notification system
- ✓ Polished, responsive UI