“Our Team” Section Documentation

Overview

The “Our Team” section appears on the Home Details page and displays staff members for a specific care home.

Current Implementation

Location

- **Page:** Home Details (`/homes/[id]`)
- **Component:** `src/app/homes/[id]/page.tsx`
- **Section:** Lines 1088-1112

Data Source

Current State: Mock/hardcoded data in `MOCK_HOME` object

```
staff: [
  {
    id: "s1",
    name: "Dr. Robert Chen",
    title: "Medical Director",
    photo: "https://placehold.co/300x300/e9ecef/495057?text=Dr.+Chen",
    bio: "Dr. Chen has over 20 years of experience in geriatric medicine..."
  },
  {
    id: "s2",
    name: "Sarah Johnson",
    title: "Administrator",
    photo: "https://placehold.co/300x300/e9ecef/495057?text=Sarah+J",
    bio: "Sarah has been with Sunshine for 8 years..."
  },
  // ... more staff members
]
```

UI Display

- **Layout:** Grid (4 columns on large screens, 2 on tablets, 1 on mobile)
- **Each Card Shows:**
- Staff photo (aspect ratio 1:1)
- Name
- Title (in primary color)
- Bio/Description
- **Styling:** White card with border, hover shadow effect

How to Add Team Members (Production)

Current Limitation

⚠️ **The “Our Team” section currently uses MOCK DATA ONLY.**

To enable real team member management in production, you need to implement:

1. Database Schema

Add a new model to `prisma/schema.prisma`:

```
model HomeStaff {
  id          String   @id @default(cuid())
  homeId      String
  name        String
  title       String
  bio         String?
  photoUrl    String?
  email       String?
  phone       String?
  order       Int      @default(0)
  active      Boolean  @default(true)
  createdAt   DateTime @default(now())
  updatedAt   DateTime @updatedAt

  home        Home     @relation(fields: [homeId], references: [id], onDelete:
Cascade)

  @@index([homeId])
  @@index([homeId, order])
}
```

Then run:

```
npx prisma migrate dev --name add_home_staff
npx prisma generate
```

2. API Endpoints

Create the following API routes:

GET `/api/homes/[homeId]/staff`

```
// Fetch all staff for a home
export async function GET(req, { params }) {
  const staff = await prisma.homeStaff.findMany({
    where: { homeId: params.homeId, active: true },
    orderBy: { order: 'asc' }
  });
  return NextResponse.json({ data: staff });
}
```

POST `/api/homes/[homeId]/staff`

```
// Add new staff member
export async function POST(req, { params }) {
  const body = await req.json();
  const staff = await prisma.homeStaff.create({
    data: {
      homeId: params.homeId,
      ...body
    }
  });
  return NextResponse.json({ data: staff });
}
```

PATCH `/api/homes/[homeId]/staff/[staffId]`

```
// Update staff member
export async function PATCH(req, { params }) {
  const body = await req.json();
  const staff = await prisma.homeStaff.update({
    where: { id: params.staffId },
    data: body
  });
  return NextResponse.json({ data: staff });
}
```

DELETE `/api/homes/[homeId]/staff/[staffId]`

```
// Remove staff member (soft delete)
export async function DELETE(req, { params }) {
  await prisma.homeStaff.update({
    where: { id: params.staffId },
    data: { active: false }
  });
  return NextResponse.json({ success: true });
}
```

3. Admin Management Interface

Create an operator page: `src/app/operator/homes/[id]/team/page.tsx`

Features:
- List of current team members with edit/delete buttons
- "Add Team Member" button
- Form with fields: Name, Title, Bio, Photo upload
- Drag-and-drop reordering (updates `order` field)
- Photo upload to Cloudinary

4. Update Home Details Page

Replace mock data with API call:

```
// src/app/homes/[id]/page.tsx
const [staff, setStaff] = useState([]);

useEffect(() => {
  fetch(`/api/homes/${id}/staff`)
    .then(res => res.json())
    .then(data => setStaff(data.data || []));
}, [id]);
```

5. Image Upload

Integrate with existing Cloudinary setup:

```
// Upload staff photo
const formData = new FormData();
formData.append('file', photoFile);
formData.append('upload_preset', process.env.NEXT_PUBLIC_CLOUDINARY_UPLOAD_PRESET);

const response = await fetch(
  `https://i.ytimg.com/vi/bAc-w5jEwMg/hq720.jpg?sqp=-oaymwEhCK4FEIIDSFryq4qpAxMIARU-
AAAAAGAElAADIQj0AgKJD&rs=AOn4CLC16_2rOhmwAZhRD36YP5h6ldsfow`,
  { method: 'POST', body: formData }
);
const { secure_url } = await response.json();
```

Sample Mock Data

For reference, here's the structure of the mock data:

```
{
  id: string,         // Unique identifier
  name: string,       // Full name (e.g., "Dr. Robert Chen")
  title: string,      // Job title (e.g., "Medical Director")
  photo: string,      // Image URL
  bio: string         // Short biography/description
}
```

UI Behavior

Responsive Design

- **Mobile (< 640px):** 1 column
- **Tablet (640-1024px):** 2 columns
- **Desktop (> 1024px):** 4 columns

Interactions

- Hover effect: Card shadow increases
- No click action on cards (view-only)
- Images maintain 1:1 aspect ratio
- Text truncation for long bios

Related Files

```
src/app/homes/[id]/page.tsx          # Main home details page
src/components/homes/PhotoGallery.tsx # Photo display component (reference)
prisma/schema.prisma                  # Database schema (needs HomeStaff model)
```

Next Steps

Phase 1: Database Setup

1. Add `HomeStaff` model to Prisma schema
2. Create and run migration
3. Seed initial data for existing homes

Phase 2: API Development

1. Create CRUD endpoints
2. Add authentication/authorization
3. Implement image upload

Phase 3: Admin Interface

1. Build team management page
2. Create add/edit forms
3. Add photo upload UI
4. Implement reordering

Phase 4: Integration

1. Update home details page to fetch real data
2. Add loading states
3. Handle empty states
4. Test thoroughly

Estimated Effort

- **Database Schema:** 1 hour
- **API Endpoints:** 4 hours
- **Admin Interface:** 8 hours
- **Integration & Testing:** 4 hours
- **Total:** ~17 hours (2-3 days)

Priority

Recommendation: Medium priority

This is a **nice-to-have feature** that enhances home profiles but is not critical for core functionality. Consider implementing after:
1. Critical bugs are fixed
2. Core search/booking flow is stable
3. Resident/caregiver management is complete

Last Updated: December 15, 2025
Status: Mock data only - full implementation pending
Owner: Operator/Admin team