NDTMFI Admin Dashboard

Here’s Project Folder Structure:

```
.next
app
app/api
app/api/applications
app/api/applications/[id]
app/api/applications/[id]/route.ts
app/api/applications/route.ts
app/api/auth
app/api/branches
app/api/branches/[id]
app/api/branches/[id]/route.ts
app/api/branches/route.ts
app/api/faq
app/api/faq/[id]
app/api/faq/[id]/route.ts
app/api/faq/route.ts
app/api/health
app/api/health/route.ts
app/api/products
app/api/products/[id]
app/api/products/[id]/route.ts
app/api/products/route.ts
app/api/search
app/api/search/route.ts
app/api/stats
app/api/stats/route.ts
app/api/users
app/api/users/[id]
app/api/users/[id]/route.ts
app/api/users/route.ts
```

```
app/dashboard
app/dashboard/branches
app/dashboard/branches/[id]/edit
app/dashboard/branches/[id]/edit/page.tsx
app/dashboard/branches/create
app/dashboard/branches/create/page.tsx
app/dashboard/branches/page.tsx
app/dashboard/compliance
app/dashboard/compliance/page.tsx
app/dashboard/faq
app/dashboard/faq/[id]/edit
app/dashboard/faq/[id]/edit/page.tsx
app/dashboard/faq/create
app/dashboard/faq/page.tsx
app/dashboard/jobs
app/dashboard/jobs/[id]/edit
app/dashboard/jobs/[id]/edit/page.tsx
app/dashboard/jobs/create
app/dashboard/jobs/create/page.tsx
app/dashboard/jobs/page.tsx
app/dashboard/loan-applications
app/dashboard/loan-applications/[id]
app/dashboard/loan-applications/[id]/page.tsx
app/dashboard/loan-applications/page.tsx
app/dashboard/loan-products
app/dashboard/loan-products/[id]/edit
app/dashboard/loan-products/[id]/edit/page.tsx
app/dashboard/loan-products/create
app/dashboard/loan-products/create/page.tsx
app/dashboard/loan-products/page.tsx
app/dashboard/reports
app/dashboard/reports/applications
app/dashboard/reports/financial
app/dashboard/reports/performance
app/dashboard/reports/page.tsx
app/dashboard/settings
```

```
app/dashboard/settings/page.tsx
app/dashboard/users
app/dashboard/users/[id]/edit
app/dashboard/users/[id]/edit/page.tsx
app/dashboard/users/create
app/dashboard/users/create/page.tsx
app/dashboard/users/page.tsx
app/dashboard/layout.tsx
app/dashboard/page.tsx
app/login
app/login/page.tsx
app/favicon.ico
app/layout.tsx
app/page.tsx
components
components/common
components/dashboard
components/forms
components/layout
components/modals
components/tables
hooks
lib
lib/auth-context.tsx
lib/auth.ts
lib/db.ts
lib/mongodb.ts
lib/utils.ts
lib/validators.ts
models
models/Branch.ts
models/FAQ.ts
models/LoanApplication.ts
models/LoanProduct.ts
models/User.ts
node_modules
```

```
public
scripts
scripts/seed-users.ts
styles
styles/globals.css
types
types/admin.ts
types/application.ts
types/common.ts
types/index.ts
.env.local
.gitignore
eslint.config.mjs
middleware.ts
next-env.d.ts
next.config.js
package-lock.json
package.json
postcss.config.js
README.md
tailwind.config.ts
tsconfig.json
```

Here’s the details of each part:

```
// app/api/applications/[id]/route.ts
import { NextRequest, NextResponse } from 'next/server';
import dbConnect from '@/lib/mongodb';
import { LoanApplication } from '@/models/LoanApplication';

// GET single application
export async function GET(
  request: NextRequest,
  { params }: { params: { id: string } }
) {
```

```
  try {
    await dbConnect();
    const application = await LoanApplication.findById(params.id);

    if (!application) {
      return NextResponse.json(
        { error: 'Application not found' },
        { status: 404 }
      );
    }

    return NextResponse.json(application);
  } catch (error) {
    return NextResponse.json(
      { error: 'Failed to fetch application' },
      { status: 500 }
    );
  }
}

// UPDATE application
export async function PUT(
  request: NextRequest,
  { params }: { params: { id: string } }
) {
  try {
    await dbConnect();
    const data = await request.json();

    const application = await LoanApplication.findByIdAndUpdate(
      params.id,
      data,
      { new: true }
    );

    if (!application) {
```

```
      return NextResponse.json(
        { error: 'Application not found' },
        { status: 404 }
      );
    }

    return NextResponse.json(application);
  } catch (error: any) {
    return NextResponse.json(
      { error: error.message },
      { status: 400 }
    );
  }
}

// DELETE application
export async function DELETE(
  request: NextRequest,
  { params }: { params: { id: string } }
) {
  try {
    await dbConnect();
    const application = await LoanApplication.findByIdAndDelete(params.id);

    if (!application) {
      return NextResponse.json(
        { error: 'Application not found' },
        { status: 404 }
      );
    }

    return NextResponse.json({ message: 'Application deleted' });
  } catch (error) {
    return NextResponse.json(
      { error: 'Failed to delete application' },
      { status: 500 }
```

```
    );
  }
}
```

```
// app/api/applications/route.ts
import { NextRequest, NextResponse } from 'next/server';
import dbConnect from '@/lib/mongodb';
import { LoanApplication } from '@/models/LoanApplication';

// GET all applications
export async function GET(request: NextRequest) {
  try {
    await dbConnect();

    const url = new URL(request.url);
    const status = url.searchParams.get('status');
    const search = url.searchParams.get('search');
    const limit = url.searchParams.get('limit');

    let query: any = {};

    if (status && status !== 'All') {
      query.status = status;
    }

    if (search) {
      query.$or = [
        { fullName: { $regex: search, $options: 'i' } },
        { email: { $regex: search, $options: 'i' } },
      ];
    }

    let dbQuery = LoanApplication.find(query).sort({ createdAt: -1 });

    if (limit) {
```

```
      dbQuery = dbQuery.limit(parseInt(limit));
    } else {
      dbQuery = dbQuery.limit(100);
    }

    const applications = await dbQuery;

    return NextResponse.json(applications);
  } catch (error: any) {
    console.error('Error fetching applications:', error);
    return NextResponse.json(
      { error: 'Failed to fetch applications', details: error.message },
      { status: 500 }
    );
  }
}

// POST new application
export async function POST(request: NextRequest) {
  try {
    await dbConnect();
    const data = await request.json();

    const application = new LoanApplication(data);
    await application.save();

    return NextResponse.json(application, { status: 201 });
  } catch (error: any) {
    console.error('Error creating application:', error);
    return NextResponse.json(
      { error: error.message },
      { status: 400 }
    );
  }
}
```

```
// app/api/branches/[id]/route.ts
import { NextRequest, NextResponse } from 'next/server';
import dbConnect from '@/lib/mongodb';
import { Branch } from '@/models/Branch';

// GET single branch
export async function GET(
  request: NextRequest,
  { params }: { params: { id: string } }
) {
  try {
    await dbConnect();
    const branch = await Branch.findById(params.id);

    if (!branch) {
      return NextResponse.json(
        { error: 'Branch not found' },
        { status: 404 }
      );
    }

    return NextResponse.json(branch);
  } catch (error) {
    return NextResponse.json(
      { error: 'Failed to fetch branch' },
      { status: 500 }
    );
  }
}

// UPDATE branch
export async function PUT(
  request: NextRequest,
  { params }: { params: { id: string } }
) {
```

```
  try {
    await dbConnect();
    const data = await request.json();

    const branch = await Branch.findByIdAndUpdate(
      params.id,
      data,
      { new: true, runValidators: true }
    );

    if (!branch) {
      return NextResponse.json(
        { error: 'Branch not found' },
        { status: 404 }
      );
    }

    return NextResponse.json(branch);
  } catch (error: any) {
    return NextResponse.json(
      { error: error.message },
      { status: 400 }
    );
  }
}

// DELETE branch
export async function DELETE(
  request: NextRequest,
  { params }: { params: { id: string } }
) {
  try {
    await dbConnect();
    const branch = await Branch.findByIdAndDelete(params.id);

    if (!branch) {
```

```
      return NextResponse.json(
        { error: 'Branch not found' },
        { status: 404 }
      );
    }

    return NextResponse.json({ message: 'Branch deleted successfully' });
  } catch (error) {
    return NextResponse.json(
      { error: 'Failed to delete branch' },
      { status: 500 }
    );
  }
}
```

```
import { NextRequest, NextResponse } from 'next/server';
import dbConnect from '@/lib/mongodb';
import { Branch } from '@/models/Branch';

// GET all branches
export async function GET() {
  try {
    await dbConnect();
    const branches = await Branch.find();
    return NextResponse.json(branches);
  } catch (error) {
    return NextResponse.json(
      { error: 'Failed to fetch branches' },
      { status: 500 }
    );
  }
}

// POST new branch
export async function POST(request: NextRequest) {
```

```
  try {
    await dbConnect();
    const data = await request.json();

    const branch = new Branch(data);
    await branch.save();

    return NextResponse.json(branch, { status: 201 });
  } catch (error: any) {
    return NextResponse.json(
      { error: error.message },
      { status: 400 }
    );
  }
}
```

```
// app/api/faq/[id]/route.ts
import { NextRequest, NextResponse } from 'next/server';
import dbConnect from '@/lib/mongodb';
import { FAQ } from '@/models/FAQ';

// GET single FAQ
export async function GET(
  request: NextRequest,
  { params }: { params: { id: string } }
) {
  try {
    await dbConnect();
    const faq = await FAQ.findById(params.id);

    if (!faq) {
      return NextResponse.json(
        { error: 'FAQ not found' },
        { status: 404 }
      );
```

```
    }

    // Increment views
    faq.views += 1;
    await faq.save();

    return NextResponse.json(faq);
  } catch (error) {
    return NextResponse.json(
      { error: 'Failed to fetch FAQ' },
      { status: 500 }
    );
  }
}

// UPDATE FAQ
export async function PUT(
  request: NextRequest,
  { params }: { params: { id: string } }
) {
  try {
    await dbConnect();
    const data = await request.json();

    const faq = await FAQ.findByIdAndUpdate(
      params.id,
      data,
      { new: true, runValidators: true }
    );

    if (!faq) {
      return NextResponse.json(
        { error: 'FAQ not found' },
        { status: 404 }
      );
    }
```

```
    return NextResponse.json(faq);
  } catch (error: any) {
    return NextResponse.json(
      { error: error.message },
      { status: 400 }
    );
  }
}

// DELETE FAQ
export async function DELETE(
  request: NextRequest,
  { params }: { params: { id: string } }
) {
  try {
    await dbConnect();
    const faq = await FAQ.findByIdAndDelete(params.id);

    if (!faq) {
      return NextResponse.json(
        { error: 'FAQ not found' },
        { status: 404 }
      );
    }

    return NextResponse.json({ message: 'FAQ deleted successfully' });
  } catch (error) {
    return NextResponse.json(
      { error: 'Failed to delete FAQ' },
      { status: 500 }
    );
  }
}
```

```
// app/api/faq/route.ts
import { NextRequest, NextResponse } from 'next/server';
import dbConnect from '@/lib/mongodb';
import { FAQ } from '@/models/FAQ';

// GET all FAQs
export async function GET() {
  try {
    await dbConnect();
    const faqs = await FAQ.find({ status: 'Active' }).sort({ views: -1 });
    return NextResponse.json(faqs);
  } catch (error) {
    return NextResponse.json(
      { error: 'Failed to fetch FAQs' },
      { status: 500 }
    );
  }
}

// POST new FAQ
export async function POST(request: NextRequest) {
  try {
    await dbConnect();
    const data = await request.json();

    const faq = new FAQ(data);
    await faq.save();

    return NextResponse.json(faq, { status: 201 });
  } catch (error: any) {
    return NextResponse.json(
      { error: error.message },
      { status: 400 }
    );
```

```
  }
}
```

```
// app/api/products/[id]/route.ts
import { NextRequest, NextResponse } from 'next/server';
import dbConnect from '@/lib/mongodb';
import { LoanProduct } from '@/models/LoanProduct';

// GET single product
export async function GET(
  request: NextRequest,
  { params }: { params: { id: string } }
) {
  try {
    await dbConnect();
    const product = await LoanProduct.findById(params.id);

    if (!product) {
      return NextResponse.json(
        { error: 'Product not found' },
        { status: 404 }
      );
    }

    return NextResponse.json(product);
  } catch (error) {
    return NextResponse.json(
      { error: 'Failed to fetch product' },
      { status: 500 }
    );
  }
}

// UPDATE product
export async function PUT(
```

```
  request: NextRequest,
  { params }: { params: { id: string } }
) {
  try {
    await dbConnect();
    const data = await request.json();

    const product = await LoanProduct.findByIdAndUpdate(
      params.id,
      data,
      { new: true, runValidators: true }
    );

    if (!product) {
      return NextResponse.json(
        { error: 'Product not found' },
        { status: 404 }
      );
    }

    return NextResponse.json(product);
  } catch (error: any) {
    return NextResponse.json(
      { error: error.message },
      { status: 400 }
    );
  }
}

// DELETE product
export async function DELETE(
  request: NextRequest,
  { params }: { params: { id: string } }
) {
  try {
    await dbConnect();
```

```
    const product = await LoanProduct.findByIdAndDelete(params.id);

    if (!product) {
      return NextResponse.json(
        { error: 'Product not found' },
        { status: 404 }
      );
    }

    return NextResponse.json({ message: 'Product deleted successfully' });
  } catch (error) {
    return NextResponse.json(
      { error: 'Failed to delete product' },
      { status: 500 }
    );
  }
}
```

```
// app/api/products/route.ts
import { NextRequest, NextResponse } from 'next/server';
import dbConnect from '@/lib/mongodb';
import { LoanProduct } from '@/models/LoanProduct';

// GET all products
export async function GET() {
  try {
    await dbConnect();
    const products = await LoanProduct.find({ status: 'Active' });
    return NextResponse.json(products);
  } catch (error) {
    return NextResponse.json(
      { error: 'Failed to fetch products' },
      { status: 500 }
    );
  }
```

```
}

// POST new product
export async function POST(request: NextRequest) {
  try {
    await dbConnect();
    const data = await request.json();

    const product = new LoanProduct(data);
    await product.save();

    return NextResponse.json(product, { status: 201 });
  } catch (error: any) {
    return NextResponse.json(
      { error: error.message },
      { status: 400 }
    );
  }
}
```

```
// app/api/search/route.ts
import { NextRequest, NextResponse } from 'next/server';
import dbConnect from '@/lib/mongodb';
import { LoanApplication } from '@/models/LoanApplication';
import { LoanProduct } from '@/models/LoanProduct';
import { Branch } from '@/models/Branch';

export async function GET(request: NextRequest) {
  try {
    await dbConnect();

    const url = new URL(request.url);
    const q = url.searchParams.get('q') || '';

    if (!q || q.length < 2) {
```

```
    return NextResponse.json({
      applications: [],
      products: [],
      branches: [],
    });
  }

  const searchRegex = { $regex: q, $options: 'i' };

  const [applications, products, branches] = await Promise.all([
    LoanApplication.find({
      $or: [
        { fullName: searchRegex },
        { email: searchRegex },
        { loanType: searchRegex },
      ],
    }).limit(5),
    LoanProduct.find({ name: searchRegex }).limit(5),
    Branch.find({
      $or: [
        { name: searchRegex },
        { city: searchRegex },
        { address: searchRegex },
      ],
    }).limit(5),
  ]);

  return NextResponse.json({
    applications,
    products,
    branches,
  });
} catch (error) {
  return NextResponse.json(
    { error: 'Failed to search' },
    { status: 500 }
```

```
    );
  }
}
```

```
// app/api/stats/route.ts
import { NextResponse } from 'next/server';
import dbConnect from '@/lib/mongodb';
import { LoanApplication } from '@/models/LoanApplication';

export async function GET() {
  try {
    await dbConnect();

    // Get statistics
    const totalApplications = await LoanApplication.countDocuments();
    const approvedApplications = await LoanApplication.countDocuments({
      status: 'Approved',
    });
    const pendingApplications = await LoanApplication.countDocuments({
      status: 'Pending',
    });
    const rejectedApplications = await LoanApplication.countDocuments({
      status: 'Rejected',
    });

    // Calculate total funds disbursed
    const disbursedData = await LoanApplication.aggregate([
      { $match: { status: 'Approved' } },
      { $group: { _id: null, total: { $sum: '$loanAmount' } } },
    ]);

    const totalDisbursed = disbursedData[0]?.total || 0;

    // Get average loan amount
    const averageData = await LoanApplication.aggregate([
```

```
      { $group: { _id: null, average: { $avg: '$loanAmount' } } },
    ]);

    const averageLoan = averageData[0]?.average || 0;

    return NextResponse.json({
      totalApplications,
      approvedApplications,
      pendingApplications,
      rejectedApplications,
      totalDisbursed,
      averageLoan: Math.round(averageLoan),
      approvalRate: totalApplications > 0
        ? Math.round((approvedApplications / totalApplications) * 100)
        : 0,
    });
  } catch (error) {
    return NextResponse.json(
      { error: 'Failed to fetch stats' },
      { status: 500 }
    );
  }
}
```

```
// app/api/users/[id]/route.ts
import { NextRequest, NextResponse } from 'next/server';
import dbConnect from '@/lib/mongodb';
import { User } from '@/models/User';

// GET single user
export async function GET(
  request: NextRequest,
  { params }: { params: { id: string } }
) {
  try {
```

```
    await dbConnect();
    const user = await User.findById(params.id).select('-password');

    if (!user) {
      return NextResponse.json(
        { error: 'User not found' },
        { status: 404 }
      );
    }

    return NextResponse.json(user);
  } catch (error) {
    return NextResponse.json(
      { error: 'Failed to fetch user' },
      { status: 500 }
    );
  }
}

// UPDATE user
export async function PUT(
  request: NextRequest,
  { params }: { params: { id: string } }
) {
  try {
    await dbConnect();
    const data = await request.json();

    // Don't allow password updates via this route
    delete data.password;

    const user = await User.findByIdAndUpdate(
      params.id,
      data,
      { new: true, runValidators: true }
    ).select('-password');
```

```
    if (!user) {
      return NextResponse.json(
        { error: 'User not found' },
        { status: 404 }
      );
    }

    return NextResponse.json(user);
  } catch (error: any) {
    return NextResponse.json(
      { error: error.message },
      { status: 400 }
    );
  }
}

// DELETE user
export async function DELETE(
  request: NextRequest,
  { params }: { params: { id: string } }
) {
  try {
    await dbConnect();
    const user = await User.findByIdAndDelete(params.id);

    if (!user) {
      return NextResponse.json(
        { error: 'User not found' },
        { status: 404 }
      );
    }

    return NextResponse.json({ message: 'User deleted successfully' });
  } catch (error) {
    return NextResponse.json(
```

```
      { error: 'Failed to delete user' },
      { status: 500 }
    );
  }
}
```

```
// app/api/users/route.ts
import { NextRequest, NextResponse } from 'next/server';
import dbConnect from '@/lib/mongodb';
import { User } from '@/models/User';

// GET all users
export async function GET() {
  try {
    await dbConnect();
    const users = await User.find().select('-password');
    return NextResponse.json(users);
  } catch (error) {
    return NextResponse.json(
      { error: 'Failed to fetch users' },
      { status: 500 }
    );
  }
}

// POST new user
export async function POST(request: NextRequest) {
  try {
    await dbConnect();
    const data = await request.json();

    // Check if user already exists
    const existingUser = await User.findOne({ email: data.email });
    if (existingUser) {
      return NextResponse.json(
```

```
        { error: 'User already exists' },
        { status: 400 }
      );
    }

    // In production, hash the password!
    // For now, storing plaintext (NOT SECURE - FIX IN PRODUCTION)
    const user = new User(data);
    await user.save();

    // Don't return password
    const userWithoutPassword = user.toObject();
    delete userWithoutPassword.password;

    return NextResponse.json(userWithoutPassword, { status: 201 });
  } catch (error: any) {
    return NextResponse.json(
      { error: error.message },
      { status: 400 }
    );
  }
}
```

```
// app/dashboard/branches/[id]/edit/page.tsx
'use client';

import { useState, useEffect } from 'react';
import { useRouter, useParams } from 'next/navigation';
import { ArrowLeft, Save } from 'lucide-react';

export default function EditBranchPage() {
  const router = useRouter();
  const params = useParams();
  const [loading, setLoading] = useState(true);
  const [saving, setSaving] = useState(false);
```

```
const [formData, setFormData] = useState({
  name: '',
  city: '',
  address: '',
  phone: '',
  email: '',
  hours: '',
  status: 'Active',
});

useEffect(() => {
  fetchBranch();
}, []);

const fetchBranch = async () => {
  try {
    const token = localStorage.getItem('auth_token');
    const response = await fetch(`/api/branches/${params.id}`, {
      headers: token ? { 'Authorization': `Bearer ${token}` } : {},
    });

    if (response.ok) {
      const data = await response.json();
      setFormData({
        name: data.name || '',
        city: data.city || '',
        address: data.address || '',
        phone: data.phone || '',
        email: data.email || '',
        hours: data.hours || '',
        status: data.status || 'Active',
      });
    }
  } catch (error) {
    console.error('Error fetching branch:', error);
  } finally {
```

```
      setLoading(false);
    }
  };

  const handleChange = (e: React.ChangeEvent<HTMLInputElement | HTMLSe
lectElement>) => {
    const { name, value } = e.target;
    setFormData({ ...formData, [name]: value });
  };

  const handleSubmit = async (e: React.FormEvent) => {
    e.preventDefault();
    setSaving(true);

    try {
      const token = localStorage.getItem('auth_token');

      const response = await fetch(`/api/branches/${params.id}`, {
        method: 'PUT',
        headers: {
          'Content-Type': 'application/json',
          ...(token && { 'Authorization': `Bearer ${token}` }),
        },
        body: JSON.stringify(formData),
      });

      if (response.ok) {
        alert('Branch updated successfully!');
        router.push('/dashboard/branches');
      } else {
        const error = await response.json();
        alert(`Error: ${error.error || 'Failed to update branch'}`);
      }
    } catch (error) {
      console.error('Error updating branch:', error);
      alert('Failed to update branch');
```

```
    } finally {
      setSaving(false);
    }
  };

  if (loading) {
    return (
      <div className="flex items-center justify-center h-96">
        <div className="inline-block animate-spin rounded-full h-12 w-12 bord
er-b-2 border-blue-600"></div>
      </div>
    );
  }

  return (
    <div className="space-y-6">
      <div className="flex items-center gap-4">
        <button onClick={() ⇒ router.back()} className="p-2 hover:bg-gray-20
0 rounded-lg">
          <ArrowLeft className="w-6 h-6" />
        </button>
        <div>
          <h1 className="text-3xl font-bold text-gray-900">Edit Branch</h1>
          <p className="text-gray-600 mt-1">Update branch details</p>
        </div>
      </div>

      <form onSubmit={handleSubmit} className="bg-white rounded-lg borde
r border-gray-200 p-8">
        <div className="grid grid-cols-1 md:grid-cols-2 gap-6">
          <div>
            <label className="block text-sm font-bold text-gray-700 mb-2">Bran
ch Name *</label>
            <input
              type="text"
              name="name"
```

```
            value={formData.name}
            onChange={handleChange}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
          />
        </div>

        <div>
          <label className="block text-sm font-bold text-gray-700 mb-2">City
*</label>
          <input
            type="text"
            name="city"
            value={formData.city}
            onChange={handleChange}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
          />
        </div>

        <div className="md:col-span-2">
          <label className="block text-sm font-bold text-gray-700 mb-2">Add
ress *</label>
          <input
            type="text"
            name="address"
            value={formData.address}
            onChange={handleChange}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
          />
        </div>
```

```
        <div>
          <label className="block text-sm font-bold text-gray-700 mb-2">Pho
ne *</label>
          <input
            type="tel"
            name="phone"
            value={formData.phone}
            onChange={handleChange}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
          />
        </div>

        <div>
          <label className="block text-sm font-bold text-gray-700 mb-2">Ema
il *</label>
          <input
            type="email"
            name="email"
            value={formData.email}
            onChange={handleChange}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
          />
        </div>

        <div>
          <label className="block text-sm font-bold text-gray-700 mb-2">Ope
rating Hours *</label>
          <input
            type="text"
            name="hours"
            value={formData.hours}
            onChange={handleChange}
```

```
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
          />
        </div>

        <div>
          <label className="block text-sm font-bold text-gray-700 mb-2">Stat
us *</label>
          <select
            name="status"
            value={formData.status}
            onChange={handleChange}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
          >
            <option value="Active">Active</option>
            <option value="Inactive">Inactive</option>
          </select>
        </div>
      </div>

      <div className="flex gap-4 mt-8">
        <button
          type="button"
          onClick={() => router.back()}
          className="flex-1 px-6 py-2 border border-gray-300 text-gray-700 ro
unded-lg font-bold hover:bg-gray-50"
        >
          Cancel
        </button>
        <button
          type="submit"
          disabled={saving}
          className="flex-1 px-6 py-2 bg-blue-600 text-white rounded-lg font-
```

```
bold hover:bg-blue-700 disabled:opacity-50 flex items-center justify-center g
ap-2"
        >
          <Save className="w-5 h-5" />
          {saving ? 'Saving...' : 'Save Changes'}
        </button>
      </div>
    </form>
  </div>
 );
}
```

```
// app/dashboard/branches/create/page.tsx
'use client';

import { useState } from 'react';
import { useRouter } from 'next/navigation';
import { ArrowLeft, Save } from 'lucide-react';

export default function CreateBranchPage() {
 const router = useRouter();
 const [loading, setLoading] = useState(false);
 const [formData, setFormData] = useState({
   name: '',
   city: '',
   address: '',
   phone: '',
   email: '',
   hours: 'Mon-Fri 8:30-16:30',
   status: 'Active',
 });

 const handleChange = (e: React.ChangeEvent<HTMLInputElement | HTMLSe
lectElement>) => {
   const { name, value } = e.target;
```

```
  setFormData({ ...formData, [name]: value });
};

const handleSubmit = async (e: React.FormEvent) => {
  e.preventDefault();
  setLoading(true);

  try {
    const token = localStorage.getItem('auth_token');

    const response = await fetch('/api/branches', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
        ...(token && { 'Authorization': `Bearer ${token}` }),
      },
      body: JSON.stringify(formData),
    });

    if (response.ok) {
      alert('Branch created successfully!');
      router.push('/dashboard/branches');
    } else {
      const error = await response.json();
      alert(`Error: ${error.error || 'Failed to create branch'}`);
    }
  } catch (error) {
    console.error('Error creating branch:', error);
    alert('Failed to create branch');
  } finally {
    setLoading(false);
  }
};

return (
  <div className="space-y-6">
```

```
      <div className="flex items-center gap-4">
        <button onClick={() => router.back()} className="p-2 hover:bg-gray-20
0 rounded-lg">
          <ArrowLeft className="w-6 h-6" />
        </button>
        <div>
          <h1 className="text-3xl font-bold text-gray-900">Create Branch</h1>
          <p className="text-gray-600 mt-1">Add a new branch location</p>
        </div>
      </div>

      <form onSubmit={handleSubmit} className="bg-white rounded-lg borde
r border-gray-200 p-8">
        <div className="grid grid-cols-1 md:grid-cols-2 gap-6">
          <div>
            <label className="block text-sm font-bold text-gray-700 mb-2">Bran
ch Name *</label>
            <input
              type="text"
              name="name"
              value={formData.name}
              onChange={handleChange}
              className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
              required
            />
          </div>

          <div>
            <label className="block text-sm font-bold text-gray-700 mb-2">City
*</label>
            <input
              type="text"
              name="city"
              value={formData.city}
              onChange={handleChange}
```

```
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
          />
        </div>

        <div className="md:col-span-2">
          <label className="block text-sm font-bold text-gray-700 mb-2">Add
ress *</label>
          <input
            type="text"
            name="address"
            value={formData.address}
            onChange={handleChange}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
          />
        </div>

        <div>
          <label className="block text-sm font-bold text-gray-700 mb-2">Pho
ne *</label>
          <input
            type="tel"
            name="phone"
            value={formData.phone}
            onChange={handleChange}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
          />
        </div>

        <div>
          <label className="block text-sm font-bold text-gray-700 mb-2">Ema
```

```
il *</label>
          <input
            type="email"
            name="email"
            value={formData.email}
            onChange={handleChange}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
          />
        </div>

        <div>
          <label className="block text-sm font-bold text-gray-700 mb-2">Ope
rating Hours *</label>
          <input
            type="text"
            name="hours"
            value={formData.hours}
            onChange={handleChange}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
          />
        </div>

        <div>
          <label className="block text-sm font-bold text-gray-700 mb-2">Stat
us *</label>
          <select
            name="status"
            value={formData.status}
            onChange={handleChange}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
```

```
          >
            <option value="Active">Active</option>
            <option value="Inactive">Inactive</option>
          </select>
        </div>
      </div>

      <div className="flex gap-4 mt-8">
        <button
          type="button"
          onClick={() ⇒ router.back()}
          className="flex-1 px-6 py-2 border border-gray-300 text-gray-700 ro
unded-lg font-bold hover:bg-gray-50"
        >
          Cancel
        </button>
        <button
          type="submit"
          disabled={loading}
          className="flex-1 px-6 py-2 bg-blue-600 text-white rounded-lg font-
bold hover:bg-blue-700 disabled:opacity-50 flex items-center justify-center g
ap-2"
        >
          <Save className="w-5 h-5" />
          {loading ? 'Creating...' : 'Create Branch'}
        </button>
      </div>
    </form>
  </div>
 );
}
```

```
// app/dashboard/branches/page.tsx
'use client';
```

```
import { useState, useEffect } from 'react';
import { Plus, Edit2, Trash2, MapPin } from 'lucide-react';
import Link from 'next/link';

interface Branch {
  _id: string;
  name: string;
  city: string;
  address: string;
  phone: string;
  email: string;
  status: string;
}

export default function BranchesPage() {
  const [branches, setBranches] = useState<Branch[]>([]);
  const [loading, setLoading] = useState(true);
  const [deleting, setDeleting] = useState<string | null>(null);

  useEffect(() => {
    fetchBranches();
  }, []);

  const fetchBranches = async () => {
    try {
      setLoading(true);
      const response = await fetch('/api/branches');
      const data = await response.json();
      setBranches(data);
    } catch (error) {
      console.error('Error fetching branches:', error);
    } finally {
      setLoading(false);
    }
  };
```

```
  const handleDelete = async (id: string) => {
    if (!confirm('Are you sure you want to delete this branch?')) return;

    try {
      setDeleting(id);
      const response = await fetch(`/api/branches/${id}`, {
        method: 'DELETE',
      });

      if (response.ok) {
        setBranches(branches.filter((b) => b._id !== id));
        alert('Branch deleted successfully');
      }
    } catch (error) {
      console.error('Error deleting branch:', error);
      alert('Failed to delete branch');
    } finally {
      setDeleting(null);
    }
  };

  return (
    <div className="space-y-6">
      <div className="flex items-center justify-between">
        <h1 className="text-3xl font-bold text-gray-900">Branch Locations</h1
>
        <Link
          href="/dashboard/branches/create"
          className="bg-blue-600 hover:bg-blue-700 px-4 py-2 rounded-lg flex
items-center gap-2 font-medium"
          style={{ color: '#ffffff' }}
        >
          <Plus className="w-5 h-5" style={{ color: '#ffffff' }} />
          <span style={{ color: '#ffffff' }}>Add Branch</span>
        </Link>
      </div>
```

```
    <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-6">
      {loading ? (
        <div className="col-span-full text-center py-8 text-gray-600">Loadin
g...</div>
      ) : branches.length === 0 ? (
        <div className="col-span-full text-center py-8 text-gray-600">No bra
nches found</div>
      ) : (
        branches.map((branch) => (
          <div key={branch._id} className="bg-white rounded-lg border borde
r-gray-200 p-6">
            <div className="flex items-start justify-between mb-4">
              <div className="flex items-center gap-3">
                <div className="w-10 h-10 bg-blue-100 rounded-lg flex items-ce
nter justify-center">
                  <MapPin className="w-6 h-6 text-blue-600" />
                </div>
                <div>
                  <h3 className="font-bold text-gray-900">{branch.name}</h3>
                  <p className="text-sm text-gray-600">{branch.city}</p>
                </div>
              </div>
              <span className="bg-green-100 text-green-700 px-2 py-1 rounde
d text-xs font-bold">
                {branch.status}
              </span>
            </div>
            <div className="space-y-2 text-sm text-gray-700 mb-4">
              <p><strong>Address:</strong> {branch.address}</p>
              <p><strong>Phone:</strong> {branch.phone}</p>
              <p><strong>Email:</strong> {branch.email}</p>
            </div>
            <div className="flex gap-2">
              <Link
                href={`/dashboard/branches/${branch._id}/edit`}
```

```
              className="flex-1 p-2 bg-green-50 text-green-600 rounded hove
r:bg-green-100 flex items-center justify-center gap-1"
            >
              <Edit2 className="w-4 h-4" /> Edit
            </Link>
            <button
              onClick={() => handleDelete(branch._id)}
              disabled={deleting === branch._id}
              className="flex-1 p-2 bg-red-50 text-red-600 rounded hover:bg-
red-100 flex items-center justify-center gap-1 disabled:opacity-50"
            >
              <Trash2 className="w-4 h-4" /> Delete
            </button>
          </div>
        </div>
      ))
    )}
  </div>
 </div>
);
}
```

```
// app/dashboard/compliance/page.tsx
'use client';

import { useState, useEffect } from 'react';
import { Shield, FileText, CheckCircle, AlertCircle, Download } from 'lucide-rea
ct';

const complianceItems = [
 {
  title: 'License & Registration',
  status: 'Compliant',
  description: 'Bank of Lao PDR License NDTMFI-2023-001',
  icon: Shield,
```

```
  },
  {
    title: 'Anti-Money Laundering',
    status: 'Compliant',
    description: 'AML/CFT Standards Compliant',
    icon: FileText,
  },
  {
    title: 'Data Protection',
    status: 'Compliant',
    description: 'Customer data protection policies in place',
    icon: Shield,
  },
  {
    title: 'Financial Reporting',
    status: 'Compliant',
    description: 'Quarterly audited reports submitted',
    icon: FileText,
  },
];

const documents = [
  { name: 'License Certificate', date: '2024-01-15', size: '2.4 MB' },
  { name: 'Annual Report 2024', date: '2024-11-20', size: '5.8 MB' },
  { name: 'Transparency Report Q4', date: '2024-11-01', size: '1.2 MB' },
  { name: 'Risk Disclosure', date: '2024-10-15', size: '0.8 MB' },
];

export default function CompliancePage() {
  const [applications, setApplications] = useState([]);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    fetchApplicationsStats();
  }, []);
```

```
  const fetchApplicationsStats = async () => {
    try {
      const response = await fetch('/api/applications');
      const data = await response.json();
      setApplications(data);
    } catch (error) {
      console.error('Error fetching applications:', error);
    } finally {
      setLoading(false);
    }
  };

  return (
    <div className="space-y-6">
      <h1 className="text-3xl font-bold text-gray-900">Compliance Managem
ent</h1>

      {/* Compliance Status */}
      <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-4">
        {complianceItems.map((item, idx) => {
          const Icon = item.icon;
          return (
            <div key={idx} className="bg-white p-6 rounded-lg border border-gr
ay-200">
              <div className="flex items-start justify-between mb-2">
                <Icon className="w-6 h-6 text-blue-600" />
                <CheckCircle className="w-5 h-5 text-green-600" />
              </div>
              <h3 className="font-bold text-gray-900 mb-1">{item.title}</h3>
              <p className="text-xs text-gray-600 mb-3">{item.description}</p>
              <span className="bg-green-100 text-green-700 px-2 py-1 rounded
text-xs font-bold">
                {item.status}
              </span>
            </div>
          );
```

```
      ))}
    </div>

    {/* Compliance Documents */}
    <div className="bg-white p-6 rounded-lg border border-gray-200">
      <h2 className="text-lg font-bold text-gray-900 mb-4">Compliance Do
cuments</h2>
      <div className="space-y-3">
        {documents.map((doc, idx) => (
          <div
            key={idx}
            className="flex items-center justify-between p-4 bg-gray-50 round
ed-lg border border-gray-200 hover:bg-gray-100 transition"
          >
            <div className="flex items-center gap-3">
              <FileText className="w-5 h-5 text-blue-600" />
              <div>
                <p className="font-bold text-gray-900">{doc.name}</p>
                <p className="text-xs text-gray-600">{doc.date} • {doc.size}</p
>
              </div>
            </div>
            <button className="p-2 text-blue-600 hover:bg-white rounded tran
sition">
              <Download className="w-5 h-5" />
            </button>
          </div>
        ))}
      </div>
    </div>

    {/* Application Statistics */}
    <div className="bg-blue-50 border-l-4 border-blue-400 p-6 rounded-l
g">
      <div className="flex items-start gap-3">
        <AlertCircle className="w-6 h-6 text-blue-600 flex-shrink-0 mt-0.5" /
```

```
>
        <div>
          <p className="font-bold text-blue-900">Compliance Status</p>
          <p className="text-sm text-blue-700 mt-1">
            All compliance requirements are being met. Total applications in syste
m: {applications.length}. Next audit scheduled for Q1 2025.
          </p>
        </div>
      </div>
    </div>
  </div>
 );
}
```

```
// app/dashboard/faq/[id]/edit/page.tsx
'use client';
import { useState, useEffect } from 'react';
import { useRouter, useParams } from 'next/navigation';
import { ArrowLeft, Save, AlertCircle } from 'lucide-react';

export default function EditFAQPage() {
  const router = useRouter();
  const params = useParams();
  const [loading, setLoading] = useState(true);
  const [saving, setSaving] = useState(false);
  const [error, setError] = useState('');
  const [formData, setFormData] = useState({
    question: '',
    answer: '',
    category: 'General',
    status: 'Active',
  });

  useEffect(() => {
    fetchFAQ();
```

```
  }, []);

  const fetchFAQ = async () => {
    try {
      setLoading(true);
      setError('');

      const token = localStorage.getItem('auth_token');
      const response = await fetch(`/api/faq/${params.id}`, {
        headers: token ? { 'Authorization': `Bearer ${token}` } : {},
      });

      if (response.ok) {
        const data = await response.json();
        setFormData({
          question: data.question || '',
          answer: data.answer || '',
          category: data.category || 'General',
          status: data.status || 'Active',
        });
      } else {
        const errorData = await response.json().catch(() => ({ error: 'Failed to loa
d FAQ' }));
        setError(errorData.error || 'Failed to load FAQ data');
      }
    } catch (error) {
      console.error('Error fetching FAQ:', error);
      setError('Network error: Could not connect to the server');
    } finally {
      setLoading(false);
    }
  };

  const handleChange = (e: React.ChangeEvent<HTMLInputElement | HTMLTe
xtAreaElement | HTMLSelectElement>) => {
    const { name, value } = e.target;
```

```
  setFormData({ ...formData, [name]: value });
};

const handleSubmit = async (e: React.FormEvent) => {
  e.preventDefault();
  setSaving(true);

  try {
    const token = localStorage.getItem('auth_token');
    const response = await fetch(`/api/faq/${params.id}`, {
      method: 'PUT',
      headers: {
        'Content-Type': 'application/json',
        ...(token && { 'Authorization': `Bearer ${token}` }),
      },
      body: JSON.stringify(formData),
    });

    if (response.ok) {
      alert('FAQ updated successfully!');
      router.push('/dashboard/faq');
    } else {
      const error = await response.json();
      alert(`Error: ${error.error || 'Failed to update FAQ'}`);
    }
  } catch (error) {
    console.error('Error updating FAQ:', error);
    alert('Network error: Failed to update FAQ');
  } finally {
    setSaving(false);
  }
};

// Show loading spinner while data is being fetched
if (loading) {
  return (
```

```
    <div className="flex items-center justify-center h-96">
      <div className="text-center">
        <div className="inline-block animate-spin rounded-full h-12 w-12 bor
der-b-2 border-blue-600"></div>
        <p className="mt-4 text-gray-600">Loading FAQ data...</p>
      </div>
    </div>
  );
 }

 // Show error state with retry option
 if (error) {
  return (
    <div className="space-y-6">
      <div className="flex items-center gap-4">
        <button
          onClick={() => router.back()}
          className="p-2 hover:bg-gray-200 rounded-lg transition"
        >
          <ArrowLeft className="w-6 h-6" />
        </button>
        <div>
          <h1 className="text-3xl font-bold text-gray-900">Edit FAQ</h1>
        </div>
      </div>

      <div className="bg-white rounded-lg border border-red-200 p-8">
        <div className="flex items-start gap-4">
          <AlertCircle className="w-8 h-8 text-red-600 flex-shrink-0 mt-1" />
          <div className="flex-1">
            <h3 className="text-lg font-bold text-red-900 mb-2">Failed to Loa
d FAQ</h3>
            <p className="text-red-700 mb-4">{error}</p>
            <div className="flex gap-4">
              <button
                onClick={() => fetchFAQ()}
```

```
              className="px-6 py-2 bg-blue-600 text-white rounded-lg font-bol
d hover:bg-blue-700 transition"
            >
              Retry
            </button>
            <button
              onClick={() ⇒ router.push('/dashboard/faq')}
              className="px-6 py-2 border border-gray-300 text-gray-700 roun
ded-lg font-bold hover:bg-gray-50 transition"
            >
              Back to FAQ List
            </button>
          </div>
        </div>
      </div>
    </div>
  );
 }

 // Only render form after data is loaded successfully
 return (
  <div className="space-y-6">
   {/* Header */}
   <div className="flex items-center gap-4">
    <button
     onClick={() ⇒ router.back()}
     className="p-2 hover:bg-gray-200 rounded-lg transition"
    >
     <ArrowLeft className="w-6 h-6" />
    </button>
    <div>
     <h1 className="text-3xl font-bold text-gray-900">Edit FAQ</h1>
     <p className="text-gray-600 mt-1">Update FAQ details</p>
    </div>
   </div>
```

```
      {/* Form */}
      <form onSubmit={handleSubmit} className="bg-white rounded-lg borde
r border-gray-200 p-8">
        <div className="space-y-6">
          {/* Question */}
          <div>
            <label className="block text-sm font-bold text-gray-700 mb-2">
              Question *
            </label>
            <input
              type="text"
              name="question"
              value={formData.question}
              onChange={handleChange}
              className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
              required
            />
          </div>

          {/* Answer */}
          <div>
            <label className="block text-sm font-bold text-gray-700 mb-2">
              Answer *
            </label>
            <textarea
              name="answer"
              value={formData.answer}
              onChange={handleChange}
              rows={5}
              className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
              required
            />
          </div>
```

```
        {/* Category */}
        <div>
          <label className="block text-sm font-bold text-gray-700 mb-2">
            Category *
          </label>
          <select
            name="category"
            value={formData.category}
            onChange={handleChange}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
          >
            <option value="General">General</option>
            <option value="Loan Products">Loan Products</option>
            <option value="Application Process">Application Process</option>
            <option value="Eligibility">Eligibility</option>
            <option value="Repayment">Repayment</option>
            <option value="Documentation">Documentation</option>
          </select>
        </div>

        {/* Status */}
        <div>
          <label className="block text-sm font-bold text-gray-700 mb-2">
            Status *
          </label>
          <select
            name="status"
            value={formData.status}
            onChange={handleChange}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
          >
```

```
            <option value="Active">Active</option>
            <option value="Inactive">Inactive</option>
          </select>
        </div>
      </div>

      {/* Buttons */}
      <div className="flex gap-4 mt-8">
        <button
          type="button"
          onClick={() ⇒ router.back()}
          className="flex-1 px-6 py-2 border border-gray-300 text-gray-700 ro
unded-lg font-bold hover:bg-gray-50 transition"
        >
          Cancel
        </button>
        <button
          type="submit"
          disabled={saving}
          className="flex-1 px-6 py-2 bg-blue-600 text-white rounded-lg font-
bold hover:bg-blue-700 transition disabled:opacity-50 flex items-center justif
y-center gap-2"
        >
          <Save className="w-5 h-5" />
          {saving ? 'Saving...' : 'Save Changes'}
        </button>
      </div>
    </form>
  </div>
 );
}
```

```
// app/dashboard/faq/create/page.tsx
'use client';
import { useState } from 'react';
```

```
import { useRouter } from 'next/navigation';
import { ArrowLeft, Save } from 'lucide-react';

export default function CreateFAQPage() {
  const router = useRouter();
  const [loading, setLoading] = useState(false);
  const [formData, setFormData] = useState({
    question: '',
    answer: '',
    category: 'General',
    status: 'Active',
  });

  const handleChange = (e: React.ChangeEvent<HTMLInputElement | HTMLTe
xtAreaElement | HTMLSelectElement>) => {
    const { name, value } = e.target;
    setFormData({ ...formData, [name]: value });
  };

  const handleSubmit = async (e: React.FormEvent) => {
    e.preventDefault();
    setLoading(true);

    try {
      const token = localStorage.getItem('auth_token');
      const response = await fetch('/api/faq', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
          ...(token && { 'Authorization': `Bearer ${token}` }),
        },
        body: JSON.stringify(formData),
      });

      if (response.ok) {
        alert('FAQ created successfully!');
```

```
        router.push('/dashboard/faq');
      } else {
        const error = await response.json();
        alert(`Error: ${error.error || 'Failed to create FAQ'}`);
      }
    } catch (error) {
      console.error('Error creating FAQ:', error);
      alert('Failed to create FAQ');
    } finally {
      setLoading(false);
    }
  };

  return (
    <div className="space-y-6">
      {/* Header */}
      <div className="flex items-center gap-4">
        <button
          onClick={() => router.back()}
          className="p-2 hover:bg-gray-200 rounded-lg transition"
        >
          <ArrowLeft className="w-6 h-6" />
        </button>
        <div>
          <h1 className="text-3xl font-bold text-gray-900">Create FAQ</h1>
          <p className="text-gray-600 mt-1">Add a new frequently asked quest
ion</p>
        </div>
      </div>

      {/* Form */}
      <form onSubmit={handleSubmit} className="bg-white rounded-lg borde
r border-gray-200 p-8">
        <div className="space-y-6">
          {/* Question */}
          <div>
```

```
          <label className="block text-sm font-bold text-gray-700 mb-2">
            Question *
          </label>
          <input
            type="text"
            name="question"
            value={formData.question}
            onChange={handleChange}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
            placeholder="What is the minimum loan amount?"
          />
        </div>

        {/* Answer */}
        <div>
          <label className="block text-sm font-bold text-gray-700 mb-2">
            Answer *
          </label>
          <textarea
            name="answer"
            value={formData.answer}
            onChange={handleChange}
            rows={5}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
            placeholder="Provide a detailed answer..."
          />
        </div>

        {/* Category */}
        <div>
          <label className="block text-sm font-bold text-gray-700 mb-2">
            Category *
```

```
          </label>
          <select
            name="category"
            value={formData.category}
            onChange={handleChange}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
          >
            <option value="General">General</option>
            <option value="Loan Products">Loan Products</option>
            <option value="Application Process">Application Process</option>
            <option value="Eligibility">Eligibility</option>
            <option value="Repayment">Repayment</option>
            <option value="Documentation">Documentation</option>
          </select>
        </div>

        {/* Status */}
        <div>
          <label className="block text-sm font-bold text-gray-700 mb-2">
            Status *
          </label>
          <select
            name="status"
            value={formData.status}
            onChange={handleChange}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
          >
            <option value="Active">Active</option>
            <option value="Inactive">Inactive</option>
          </select>
        </div>
      </div>
```

```
        {/* Buttons */}
        <div className="flex gap-4 mt-8">
          <button
            type="button"
            onClick={() => router.back()}
            className="flex-1 px-6 py-2 border border-gray-300 text-gray-700 ro
unded-lg font-bold hover:bg-gray-50 transition"
          >
            Cancel
          </button>
          <button
            type="submit"
            disabled={loading}
            className="flex-1 px-6 py-2 bg-blue-600 text-white rounded-lg font-
bold hover:bg-blue-700 transition disabled:opacity-50 flex items-center justif
y-center gap-2"
          >
            <Save className="w-5 h-5" />
            {loading ? 'Creating...' : 'Create FAQ'}
          </button>
        </div>
      </form>
    </div>
  );
}
```

```
// app/dashboard/faq/page.tsx
'use client';

import { useState, useEffect } from 'react';
import { Plus, Edit2, Trash2, HelpCircle } from 'lucide-react';
import Link from 'next/link';

interface FAQItem {
```

```
  _id: string;
  question: string;
  answer: string;
  category: string;
  views: number;
  status: string;
}

export default function FAQPage() {
  const [faqs, setFaqs] = useState<FAQItem[]>([]);
  const [loading, setLoading] = useState(true);
  const [deleting, setDeleting] = useState<string | null>(null);

  useEffect(() => {
    fetchFAQs();
  }, []);

  const fetchFAQs = async () => {
    try {
      setLoading(true);
      const response = await fetch('/api/faq');
      const data = await response.json();
      setFaqs(data);
    } catch (error) {
      console.error('Error fetching FAQs:', error);
    } finally {
      setLoading(false);
    }
  };

  const handleDelete = async (id: string) => {
    if (!confirm('Are you sure you want to delete this FAQ?')) return;

    try {
      setDeleting(id);
      const response = await fetch(`/api/faq/${id}`, {
```

```
        method: 'DELETE',
      });

      if (response.ok) {
        setFaqs(faqs.filter((f) => f._id !== id));
        alert('FAQ deleted successfully');
      }
    } catch (error) {
      console.error('Error deleting FAQ:', error);
      alert('Failed to delete FAQ');
    } finally {
      setDeleting(null);
    }
  };

  return (
    <div className="space-y-6">
      <div className="flex items-center justify-between">
        <h1 className="text-3xl font-bold text-gray-900">FAQ Management</h
1>
        <Link
          href="/dashboard/faq/create"
          className="bg-blue-600 hover:bg-blue-700 px-4 py-2 rounded-lg fon
t-medium flex items-center gap-2"
          style={{ color: '#ffffff' }}
        >
          <Plus className="w-5 h-5" style={{ color: '#ffffff' }} />
          <span style={{ color: '#ffffff' }}>Add FAQ</span>
        </Link>
      </div>

      <div className="space-y-4">
        {loading ? (
          <div className="text-center py-8 text-gray-600">Loading...</div>
        ) : faqs.length === 0 ? (
          <div className="text-center py-8 text-gray-600">No FAQs found</div
```

```
>
      ) : (
        faqs.map((faq) => (
          <div key={faq._id} className="bg-white p-6 rounded-lg border borde
r-gray-200">
            <div className="flex items-start justify-between">
              <div className="flex-1">
                <div className="flex items-center gap-3 mb-2">
                  <HelpCircle className="w-5 h-5 text-blue-600" />
                  <h3 className="font-bold text-gray-900">{faq.question}</h3>
                </div>
                <p className="text-gray-700 ml-8 mb-2">{faq.answer}</p>
                <div className="flex items-center gap-4 ml-8 text-sm text-gray-5
00">
                  <span>Category: {faq.category}</span>
                  <span>Views: {faq.views}</span>
                  <span className="bg-green-100 text-green-700 px-2 py-1 round
ed text-xs font-bold">
                    {faq.status}
                  </span>
                </div>
              </div>
              <div className="flex gap-2">
                <Link
                  href={`/dashboard/faq/${faq._id}/edit`}
                  className="p-2 text-green-600 hover:bg-green-50 rounded"
                >
                  <Edit2 className="w-5 h-5" />
                </Link>
                <button
                  onClick={() => handleDelete(faq._id)}
                  disabled={deleting === faq._id}
                  className="p-2 text-red-600 hover:bg-red-50 rounded disabled:
opacity-50"
                >
                  <Trash2 className="w-5 h-5" />
```

```
              </button>
            </div>
          </div>
        </div>
      ))
    )}
  </div>
 </div>
 );
}
```

```
// app/dashboard/jobs/[id]/edit/page.tsx
'use client';
import { useState, useEffect } from 'react';
import { useRouter, useParams } from 'next/navigation';
import { ArrowLeft, Save } from 'lucide-react';

// Mock data for demonstration
const mockJobs: any = {
  '1': { id: 1, role: 'Loan Officer', level: 'Mid-Level', location: 'Vientiane', descripti
on: 'Process and evaluate loan applications from potential borrowers. Conduct
financial analysis and risk assessment.', requirements: 'Bachelor degree in Fin
ance or related field. 2+ years experience in banking or microfinance. Strong
analytical skills.', responsibilities: 'Review loan applications. Conduct client int
erviews. Perform credit analysis. Prepare loan documentation. Monitor loan p
ortfolio.', salary: '5,000,000 - 8,000,000 LAK', employmentType: 'Full-Time', st
atus: 'Open' },
  '2': { id: 2, role: 'Field Officer', level: 'Entry-Level', location: 'Multiple', descrip
tion: 'Conduct field visits to assess client businesses and ensure loan complia
nce.', requirements: 'High school diploma or equivalent. Good communication
skills. Motorcycle license required.', responsibilities: 'Visit client locations. Doc
ument business operations. Collect repayments. Report to loan officers.', salar
y: '4,000,000 - 6,000,000 LAK', employmentType: 'Full-Time', status: 'Open' },
  '3': { id: 3, role: 'Finance Analyst', level: 'Mid-Level', location: 'Vientiane', des
cription: 'Analyze financial data and prepare reports for management decision
```

```
making.', requirements: 'Bachelor degree in Finance or Accounting. Strong Ex
cel skills. 3+ years experience.', responsibilities: 'Financial data analysis. Repo
rt preparation. Budget monitoring. Risk assessment.', salary: '6,000,000 - 10,0
00,000 LAK', employmentType: 'Full-Time', status: 'Closed' },
};

export default function EditJobPage() {
  const router = useRouter();
  const params = useParams();
  const [loading, setLoading] = useState(true);
  const [saving, setSaving] = useState(false);
  const [formData, setFormData] = useState({
    role: '',
    level: 'Entry-Level',
    location: '',
    description: '',
    requirements: '',
    responsibilities: '',
    salary: '',
    employmentType: 'Full-Time',
    status: 'Open',
  });

  useEffect(() => {
    loadJobData();
  }, [params.id]);

  const loadJobData = async () => {
    try {
      // Simulate API call delay
      await new Promise(resolve => setTimeout(resolve, 500));

      const job = mockJobs[params.id as string];
      if (job) {
        setFormData({
          role: job.role || '',
```

```
          level: job.level || 'Entry-Level',
          location: job.location || '',
          description: job.description || '',
          requirements: job.requirements || '',
          responsibilities: job.responsibilities || '',
          salary: job.salary || '',
          employmentType: job.employmentType || 'Full-Time',
          status: job.status || 'Open',
        });
      } else {
        alert('Job not found');
        router.push('/dashboard/jobs');
      }
    } catch (error) {
      console.error('Error loading job:', error);
      alert('Failed to load job data');
      router.push('/dashboard/jobs');
    } finally {
      setLoading(false);
    }
  };

  const handleChange = (e: React.ChangeEvent<HTMLInputElement | HTMLTe
xtAreaElement | HTMLSelectElement>) => {
    const { name, value } = e.target;
    setFormData({ ...formData, [name]: value });
  };

  const handleSubmit = async (e: React.FormEvent) => {
    e.preventDefault();
    setSaving(true);

    try {
      const token = localStorage.getItem('auth_token');

      // Simulate API call
```

```
      await new Promise(resolve => setTimeout(resolve, 1000));

      alert('Job posting updated successfully!');
      router.push('/dashboard/jobs');
    } catch (error) {
      console.error('Error updating job:', error);
      alert('Failed to update job posting');
    } finally {
      setSaving(false);
    }
  };

  // Show loading spinner while data is being fetched
  if (loading) {
    return (
      <div className="flex items-center justify-center h-96">
        <div className="text-center">
          <div className="inline-block animate-spin rounded-full h-12 w-12 bor
der-b-2 border-blue-600"></div>
          <p className="mt-4 text-gray-600">Loading job data...</p>
        </div>
      </div>
    );
  }

  // Only render form after data is loaded
  return (
    <div className="space-y-6">
      {/* Header */}
      <div className="flex items-center gap-4">
        <button
          onClick={() => router.back()}
          className="p-2 hover:bg-gray-200 rounded-lg transition"
        >
          <ArrowLeft className="w-6 h-6" />
        </button>
```

```
      <div>
        <h1 className="text-3xl font-bold text-gray-900">Edit Job Posting</h1
>
        <p className="text-gray-600 mt-1">Update job posting details</p>
      </div>
    </div>

    {/* Form */}
    <form onSubmit={handleSubmit} className="bg-white rounded-lg borde
r border-gray-200 p-8">
      <div className="grid grid-cols-1 md:grid-cols-2 gap-6">
        {/* Role */}
        <div className="md:col-span-2">
          <label className="block text-sm font-bold text-gray-700 mb-2">
            Job Title / Role *
          </label>
          <input
            type="text"
            name="role"
            value={formData.role}
            onChange={handleChange}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
          />
        </div>

        {/* Level */}
        <div>
          <label className="block text-sm font-bold text-gray-700 mb-2">
            Experience Level *
          </label>
          <select
            name="level"
            value={formData.level}
            onChange={handleChange}
```

```
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
           >
            <option value="Entry-Level">Entry-Level</option>
            <option value="Mid-Level">Mid-Level</option>
            <option value="Senior">Senior</option>
            <option value="Management">Management</option>
           </select>
          </div>

          {/* Location */}
          <div>
           <label className="block text-sm font-bold text-gray-700 mb-2">
            Location *
           </label>
           <input
            type="text"
            name="location"
            value={formData.location}
            onChange={handleChange}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
           />
          </div>

          {/* Employment Type */}
          <div>
           <label className="block text-sm font-bold text-gray-700 mb-2">
            Employment Type *
           </label>
           <select
            name="employmentType"
            value={formData.employmentType}
            onChange={handleChange}
```

```
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
           >
            <option value="Full-Time">Full-Time</option>
            <option value="Part-Time">Part-Time</option>
            <option value="Contract">Contract</option>
            <option value="Temporary">Temporary</option>
           </select>
          </div>

          {/* Salary */}
          <div>
           <label className="block text-sm font-bold text-gray-700 mb-2">
            Salary Range
           </label>
           <input
            type="text"
            name="salary"
            value={formData.salary}
            onChange={handleChange}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            placeholder="e.g., 5,000,000 - 8,000,000 LAK"
           />
          </div>

          {/* Description */}
          <div className="md:col-span-2">
           <label className="block text-sm font-bold text-gray-700 mb-2">
            Job Description *
           </label>
           <textarea
            name="description"
            value={formData.description}
            onChange={handleChange}
```

```
            rows={4}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
          />
        </div>

        {/* Responsibilities */}
        <div className="md:col-span-2">
          <label className="block text-sm font-bold text-gray-700 mb-2">
            Key Responsibilities *
          </label>
          <textarea
            name="responsibilities"
            value={formData.responsibilities}
            onChange={handleChange}
            rows={4}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
          />
        </div>

        {/* Requirements */}
        <div className="md:col-span-2">
          <label className="block text-sm font-bold text-gray-700 mb-2">
            Requirements & Qualifications *
          </label>
          <textarea
            name="requirements"
            value={formData.requirements}
            onChange={handleChange}
            rows={4}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
```

```
          />
        </div>

        {/* Status */}
        <div>
          <label className="block text-sm font-bold text-gray-700 mb-2">
            Status *
          </label>
          <select
            name="status"
            value={formData.status}
            onChange={handleChange}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
          >
            <option value="Open">Open</option>
            <option value="Closed">Closed</option>
            <option value="Draft">Draft</option>
          </select>
        </div>
      </div>

      {/* Buttons */}
      <div className="flex gap-4 mt-8">
        <button
          type="button"
          onClick={() => router.back()}
          className="flex-1 px-6 py-2 border border-gray-300 text-gray-700 ro
unded-lg font-bold hover:bg-gray-50 transition"
        >
          Cancel
        </button>
        <button
          type="submit"
          disabled={saving}
```

```
          className="flex-1 px-6 py-2 bg-blue-600 text-white rounded-lg font-
bold hover:bg-blue-700 transition disabled:opacity-50 flex items-center justif
y-center gap-2"
        >
          <Save className="w-5 h-5" />
          {saving ? 'Saving...' : 'Save Changes'}
        </button>
      </div>
    </form>
  </div>
 );
}
```

```
// app/dashboard/jobs/create/page.tsx
'use client';
import { useState } from 'react';
import { useRouter } from 'next/navigation';
import { ArrowLeft, Save } from 'lucide-react';

export default function CreateJobPage() {
 const router = useRouter();
 const [loading, setLoading] = useState(false);
 const [formData, setFormData] = useState({
  role: '',
  level: 'Entry-Level',
  location: '',
  description: '',
  requirements: '',
  responsibilities: '',
  salary: '',
  employmentType: 'Full-Time',
  status: 'Open',
 });

 const handleChange = (e: React.ChangeEvent<HTMLInputElement | HTMLTe
```

```
xtAreaElement | HTMLSelectElement>) => {
    const { name, value } = e.target;
    setFormData({ ...formData, [name]: value });
  };

  const handleSubmit = async (e: React.FormEvent) => {
    e.preventDefault();
    setLoading(true);

    try {
      const token = localStorage.getItem('auth_token');

      // For now, just show success (you can add API endpoint later)
      alert('Job posting created successfully!');
      router.push('/dashboard/jobs');
    } catch (error) {
      console.error('Error creating job:', error);
      alert('Failed to create job posting');
    } finally {
      setLoading(false);
    }
  };

  return (
    <div className="space-y-6">
      {/* Header */}
      <div className="flex items-center gap-4">
        <button
          onClick={() => router.back()}
          className="p-2 hover:bg-gray-200 rounded-lg transition"
        >
          <ArrowLeft className="w-6 h-6" />
        </button>
        <div>
          <h1 className="text-3xl font-bold text-gray-900">Create Job Posting
</h1>
```

```
          <p className="text-gray-600 mt-1">Add a new job opening</p>
        </div>
      </div>

      {/* Form */}
      <form onSubmit={handleSubmit} className="bg-white rounded-lg borde
r border-gray-200 p-8">
        <div className="grid grid-cols-1 md:grid-cols-2 gap-6">
          {/* Role */}
          <div className="md:col-span-2">
            <label className="block text-sm font-bold text-gray-700 mb-2">
              Job Title / Role *
            </label>
            <input
              type="text"
              name="role"
              value={formData.role}
              onChange={handleChange}
              className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
              required
              placeholder="e.g., Loan Officer"
            />
          </div>

          {/* Level */}
          <div>
            <label className="block text-sm font-bold text-gray-700 mb-2">
              Experience Level *
            </label>
            <select
              name="level"
              value={formData.level}
              onChange={handleChange}
              className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
```

```
            required
          >
            <option value="Entry-Level">Entry-Level</option>
            <option value="Mid-Level">Mid-Level</option>
            <option value="Senior">Senior</option>
            <option value="Management">Management</option>
          </select>
        </div>

        {/* Location */}
        <div>
          <label className="block text-sm font-bold text-gray-700 mb-2">
            Location *
          </label>
          <input
            type="text"
            name="location"
            value={formData.location}
            onChange={handleChange}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
            placeholder="e.g., Vientiane"
          />
        </div>

        {/* Employment Type */}
        <div>
          <label className="block text-sm font-bold text-gray-700 mb-2">
            Employment Type *
          </label>
          <select
            name="employmentType"
            value={formData.employmentType}
            onChange={handleChange}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
```

```
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
           >
            <option value="Full-Time">Full-Time</option>
            <option value="Part-Time">Part-Time</option>
            <option value="Contract">Contract</option>
            <option value="Temporary">Temporary</option>
           </select>
          </div>

          {/* Salary */}
          <div>
           <label className="block text-sm font-bold text-gray-700 mb-2">
            Salary Range
           </label>
           <input
            type="text"
            name="salary"
            value={formData.salary}
            onChange={handleChange}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            placeholder="e.g., 5,000,000 - 8,000,000 LAK"
           />
          </div>

          {/* Description */}
          <div className="md:col-span-2">
           <label className="block text-sm font-bold text-gray-700 mb-2">
            Job Description *
           </label>
           <textarea
            name="description"
            value={formData.description}
            onChange={handleChange}
            rows={4}
```

```
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
            placeholder="Describe the role and what the job entails..."
          />
        </div>

        {/* Responsibilities */}
        <div className="md:col-span-2">
          <label className="block text-sm font-bold text-gray-700 mb-2">
            Key Responsibilities *
          </label>
          <textarea
            name="responsibilities"
            value={formData.responsibilities}
            onChange={handleChange}
            rows={4}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
            placeholder="List the main responsibilities (one per line)..."
          />
        </div>

        {/* Requirements */}
        <div className="md:col-span-2">
          <label className="block text-sm font-bold text-gray-700 mb-2">
            Requirements & Qualifications *
          </label>
          <textarea
            name="requirements"
            value={formData.requirements}
            onChange={handleChange}
            rows={4}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
```

```
            required
            placeholder="List the required qualifications and experience..."
          />
        </div>

        {/* Status */}
        <div>
          <label className="block text-sm font-bold text-gray-700 mb-2">
            Status *
          </label>
          <select
            name="status"
            value={formData.status}
            onChange={handleChange}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
          >
            <option value="Open">Open</option>
            <option value="Closed">Closed</option>
            <option value="Draft">Draft</option>
          </select>
        </div>
      </div>

      {/* Buttons */}
      <div className="flex gap-4 mt-8">
        <button
          type="button"
          onClick={() => router.back()}
          className="flex-1 px-6 py-2 border border-gray-300 text-gray-700 ro
unded-lg font-bold hover:bg-gray-50 transition"
        >
          Cancel
        </button>
        <button
```

```
          type="submit"
          disabled={loading}
          className="flex-1 px-6 py-2 bg-blue-600 text-white rounded-lg font-bold hover:bg-blue-700 transition disabled:opacity-50 flex items-center justify-center gap-2"
        >
          <Save className="w-5 h-5" />
          {loading ? 'Creating...' : 'Create Job Posting'}
        </button>
      </div>
    </form>
  </div>
 );
}
```

```
// app/dashboard/jobs/page.tsx
'use client';
import Link from 'next/link';

import { Plus, Edit2, Trash2, Briefcase } from 'lucide-react';

const jobs = [
 { id: 1, role: 'Loan Officer', level: 'Mid-Level', location: 'Vientiane', applicants: 8, status: 'Open' },
 { id: 2, role: 'Field Officer', level: 'Entry-Level', location: 'Multiple', applicants: 12, status: 'Open' },
 { id: 3, role: 'Finance Analyst', level: 'Mid-Level', location: 'Vientiane', applicants: 5, status: 'Closed' },
];

export default function JobsPage() {
 return (
  <div className="space-y-6">
   <div className="flex items-center justify-between">
    <h1 className="text-3xl font-bold text-gray-900">Job Postings</h1>
```

```
      <Link
        href="/dashboard/jobs/create"
        className="bg-blue-600 hover:bg-blue-700 px-4 py-2 rounded-lg fon
t-medium flex items-center gap-2"
        style={{ color: '#ffffff' }}
      >
        <Plus className="w-5 h-5" style={{ color: '#ffffff' }} />
        <span style={{ color: '#ffffff' }}>Post Job</span>
      </Link>
    </div>

    <div className="bg-white rounded-lg border border-gray-200 overflow-
hidden">
      <table className="w-full">
        <thead className="bg-gray-50 border-b border-gray-200">
          <tr>
            <th className="px-6 py-3 text-left text-sm font-bold text-gray-70
0">Position</th>
            <th className="px-6 py-3 text-left text-sm font-bold text-gray-70
0">Level</th>
            <th className="px-6 py-3 text-left text-sm font-bold text-gray-70
0">Location</th>
            <th className="px-6 py-3 text-left text-sm font-bold text-gray-70
0">Applicants</th>
            <th className="px-6 py-3 text-left text-sm font-bold text-gray-70
0">Status</th>
            <th className="px-6 py-3 text-left text-sm font-bold text-gray-70
0">Actions</th>
          </tr>
        </thead>
        <tbody>
          {jobs.map((job) => (
            <tr key={job.id} className="border-b border-gray-200 hover:bg-gra
y-50">
              <td className="px-6 py-3 font-bold text-gray-900">{job.role}</td
>
```

```
            <td className="px-6 py-3 text-gray-700">{job.level}</td>
            <td className="px-6 py-3 text-gray-700">{job.location}</td>
            <td className="px-6 py-3 text-gray-900 font-bold">{job.applicant
s}</td>
            <td className="px-6 py-3">
              <span className={`px-3 py-1 rounded-full text-xs font-bold ${job.
status === 'Open' ? 'bg-green-100 text-green-700' : 'bg-gray-100 text-gray-7
00'
                }`}>
                {job.status}
              </span>
            </td>
            <td className="px-6 py-3">
              <div className="flex gap-2">
                <Link
                  href={`/dashboard/jobs/${job.id}/edit`}
                  className="p-2 text-green-600 hover:bg-green-50 rounded"
                >
                  <Edit2 className="w-5 h-5" />
                </Link>
                <button className="p-2 text-red-600 hover:bg-red-50 rounde
d">
                  <Trash2 className="w-5 h-5" />
                </button>
              </div>
            </td>
          </tr>
        ))}
      </tbody>
    </table>
  </div>
 </div>
 );
}
```

```
// app/dashboard/loan-applications/[id]/page.tsx
'use client';

import { useState } from 'react';
import { useParams, useRouter } from 'next/navigation';
import { ArrowLeft, FileText, Mail, Phone, MapPin, DollarSign, Calendar, Check
Circle } from 'lucide-react';
import Link from 'next/link';

// Mock application data
const mockApplication = {
  id: 1,
  name: 'John Sengthavanh',
  email: 'john@example.com',
  phone: '+856-20-5558888',
  address: 'Lane Xang, Vientiane',
  dob: '1985-05-15',
  gender: 'Male',
  idNumber: '1234567890',
  product: 'Micro Business Loan',
  amount: 2000,
  purpose: 'Expand my small grocery shop',
  tenure: 12,
  employment: 'Self-Employed',
  income: 800000,
  businessName: 'Lucky Store',
  yearsInBusiness: 3,
  employees: 2,
  status: 'Pending',
  submittedDate: '2024-11-20',
  documents: [
    { name: 'Valid ID', uploaded: true, date: '2024-11-20' },
    { name: 'Income Proof', uploaded: true, date: '2024-11-20' },
    { name: 'Property Document', uploaded: false },
  ],
```

```
};

export default function ApplicationDetailPage() {
  const params = useParams();
  const router = useRouter();
  const [status, setStatus] = useState(mockApplication.status);
  const [notes, setNotes] = useState('');

  const handleStatusChange = (newStatus: string) => {
    setStatus(newStatus);
    alert(`Application status changed to: ${newStatus}`);
  };

  return (
    <div className="space-y-6">
      {/* Header */}
      <div className="flex items-center justify-between">
        <div className="flex items-center gap-4">
          <button
            onClick={() => router.back()}
            className="p-2 hover:bg-gray-200 rounded-lg transition"
          >
            <ArrowLeft className="w-6 h-6" />
          </button>
          <div>
            <h1 className="text-3xl font-bold text-gray-900">Application Details
</h1>
            <p className="text-gray-600">Application ID: #{mockApplication.id}
</p>
          </div>
        </div>
        <span
          className={`px-4 py-2 rounded-full text-sm font-bold ${
            status === 'Approved'
              ? 'bg-green-100 text-green-700'
              : status === 'Pending'
```

```
            ? 'bg-yellow-100 text-yellow-700'
            : status === 'Under Review'
            ? 'bg-blue-100 text-blue-700'
            : 'bg-red-100 text-red-700'
          }`}
        >
          {status}
        </span>
      </div>

      <div className="grid grid-cols-1 lg:grid-cols-3 gap-6">
        {/* Main Content */}
        <div className="lg:col-span-2 space-y-6">
          {/* Personal Information */}
          <div className="bg-white p-6 rounded-lg border border-gray-200">
            <h2 className="text-lg font-bold text-gray-900 mb-4">Personal Info
rmation</h2>
            <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
              <div>
                <p className="text-sm text-gray-600">Full Name</p>
                <p className="font-bold text-gray-900">{mockApplication.name}
</p>
              </div>
              <div>
                <p className="text-sm text-gray-600">Email</p>
                <p className="font-bold text-gray-900 flex items-center gap-2">
                  <Mail className="w-4 h-4" /> {mockApplication.email}
                </p>
              </div>
              <div>
                <p className="text-sm text-gray-600">Phone</p>
                <p className="font-bold text-gray-900 flex items-center gap-2">
                  <Phone className="w-4 h-4" /> {mockApplication.phone}
                </p>
              </div>
              <div>
```

```
            <p className="text-sm text-gray-600">Date of Birth</p>
            <p className="font-bold text-gray-900">{mockApplication.dob}</
p>
          </div>
          <div>
            <p className="text-sm text-gray-600">Gender</p>
            <p className="font-bold text-gray-900">{mockApplication.gende
r}</p>
          </div>
          <div>
            <p className="text-sm text-gray-600">ID Number</p>
            <p className="font-bold text-gray-900">{mockApplication.idNum
ber}</p>
          </div>
          <div className="md:col-span-2">
            <p className="text-sm text-gray-600">Address</p>
            <p className="font-bold text-gray-900 flex items-center gap-2">
              <MapPin className="w-4 h-4" /> {mockApplication.address}
            </p>
          </div>
        </div>
      </div>

      {/* Loan Details */}
      <div className="bg-white p-6 rounded-lg border border-gray-200">
        <h2 className="text-lg font-bold text-gray-900 mb-4">Loan Details
</h2>
        <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
          <div>
            <p className="text-sm text-gray-600">Loan Product</p>
            <p className="font-bold text-gray-900">{mockApplication.produc
t}</p>
          </div>
          <div>
            <p className="text-sm text-gray-600">Loan Amount</p>
            <p className="font-bold text-gray-900 flex items-center gap-2">
```

```
              <DollarSign className="w-4 h-4" /> {mockApplication.amount.to
LocaleString()}
            </p>
          </div>
          <div>
            <p className="text-sm text-gray-600">Tenure (Months)</p>
            <p className="font-bold text-gray-900">{mockApplication.tenure}
</p>
          </div>
          <div>
            <p className="text-sm text-gray-600">Purpose</p>
            <p className="font-bold text-gray-900">{mockApplication.purpos
e}</p>
          </div>
        </div>
      </div>

      {/* Employment Information */}
      <div className="bg-white p-6 rounded-lg border border-gray-200">
        <h2 className="text-lg font-bold text-gray-900 mb-4">Employment I
nformation</h2>
        <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
          <div>
            <p className="text-sm text-gray-600">Employment Status</p>
            <p className="font-bold text-gray-900">{mockApplication.employ
ment}</p>
          </div>
          <div>
            <p className="text-sm text-gray-600">Monthly Income</p>
            <p className="font-bold text-gray-900">{mockApplication.incom
e.toLocaleString()} LAK</p>
          </div>
          <div>
            <p className="text-sm text-gray-600">Business Name</p>
            <p className="font-bold text-gray-900">{mockApplication.busine
ssName}</p>
```

```
          </div>
          <div>
            <p className="text-sm text-gray-600">Years in Business</p>
            <p className="font-bold text-gray-900">{mockApplication.yearsIn
Business}</p>
          </div>
          <div>
            <p className="text-sm text-gray-600">Number of Employees</p>
            <p className="font-bold text-gray-900">{mockApplication.employ
ees}</p>
          </div>
        </div>
      </div>

      {/* Documents */}
      <div className="bg-white p-6 rounded-lg border border-gray-200">
        <h2 className="text-lg font-bold text-gray-900 mb-4">Documents</
h2>
        <div className="space-y-2">
          {mockApplication.documents.map((doc, idx) => (
            <div key={idx} className="flex items-center justify-between p-3 b
g-gray-50 rounded-lg border border-gray-200">
              <div className="flex items-center gap-3">
                <FileText className="w-5 h-5 text-blue-600" />
                <div>
                  <p className="font-bold text-gray-900">{doc.name}</p>
                  {doc.uploaded && <p className="text-xs text-gray-600">{doc.
date}</p>}
                </div>
              </div>
              <span
                className={`px-3 py-1 rounded-full text-xs font-bold ${
                  doc.uploaded
                    ? 'bg-green-100 text-green-700 flex items-center gap-1'
                    : 'bg-red-100 text-red-700'
                }`}
```

```
              >
                {doc.uploaded ? (
                  <>
                    <CheckCircle className="w-3 h-3" /> Uploaded
                  </>
                ) : (
                  'Missing'
                )}
              </span>
            </div>
          ))}
        </div>
      </div>

      {/* Notes */}
      <div className="bg-white p-6 rounded-lg border border-gray-200">
        <h2 className="text-lg font-bold text-gray-900 mb-4">Review Notes
</h2>
        <textarea
          value={notes}
          onChange={(e) => setNotes(e.target.value)}
          placeholder="Add your review notes here..."
          className="w-full px-4 py-3 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
          rows={4}
        />
      </div>
    </div>

    {/* Sidebar */}
    <div className="space-y-6">
      {/* Application Status */}
      <div className="bg-white p-6 rounded-lg border border-gray-200">
        <h2 className="text-lg font-bold text-gray-900 mb-4">Change Statu
s</h2>
        <div className="space-y-2">
```

```
            {['Pending', 'Under Review', 'Approved', 'Rejected'].map((s) => (
              <button
                key={s}
                onClick={() => handleStatusChange(s)}
                className={`w-full px-4 py-2 rounded-lg font-bold transition ${
                  status === s
                    ? 'bg-blue-600 text-white'
                    : 'bg-gray-100 text-gray-700 hover:bg-gray-200'
                }`}
              >
                {s}
              </button>
            ))}
          </div>
        </div>

        {/* Timeline */}
        <div className="bg-white p-6 rounded-lg border border-gray-200">
          <h2 className="text-lg font-bold text-gray-900 mb-4">Timeline</h2
>
          <div className="space-y-3">
            <div className="flex gap-3">
              <div className="w-2 h-2 bg-blue-600 rounded-full mt-2 flex-shrin
k-0"></div>
              <div>
                <p className="text-sm font-bold text-gray-900">Application Sub
mitted</p>
                <p className="text-xs text-gray-600">{mockApplication.submitte
dDate}</p>
              </div>
            </div>
            <div className="flex gap-3">
              <div className="w-2 h-2 bg-gray-400 rounded-full mt-2 flex-shrin
k-0"></div>
              <div>
                <p className="text-sm font-bold text-gray-900">Document Verifi
```

```
cation</p>
              <p className="text-xs text-gray-600">Pending</p>
             </div>
            </div>
            <div className="flex gap-3">
             <div className="w-2 h-2 bg-gray-400 rounded-full mt-2 flex-shrin
k-0"></div>
             <div>
              <p className="text-sm font-bold text-gray-900">Final Decision</
p>
              <p className="text-xs text-gray-600">Pending</p>
             </div>
            </div>
           </div>
          </div>

          {/* Actions */}
          <div className="bg-white p-6 rounded-lg border border-gray-200">
           <h2 className="text-lg font-bold text-gray-900 mb-4">Actions</h2>
           <div className="space-y-2">
            <button className="w-full px-4 py-2 bg-green-600 text-white roun
ded-lg font-bold hover:bg-green-700 transition">
             Approve
            </button>
            <button className="w-full px-4 py-2 bg-red-600 text-white rounde
d-lg font-bold hover:bg-red-700 transition">
             Reject
            </button>
            <button className="w-full px-4 py-2 border border-gray-300 text-g
ray-700 rounded-lg font-bold hover:bg-gray-50 transition">
             Send Message
            </button>
            <button className="w-full px-4 py-2 border border-gray-300 text-g
ray-700 rounded-lg font-bold hover:bg-gray-50 transition">
             Schedule Call
            </button>
```

```
          </div>
        </div>
      </div>
    </div>
  </div>
 );
}
```

```
// app/dashboard/loan-applications/page.tsx
'use client';

import { useState, useEffect } from 'react';
import { Search, Filter, Plus, Eye, Edit2, Trash2, Download } from 'lucide-reac
t';
import Link from 'next/link';

export default function ApplicationsPage() {
  const [applications, setApplications] = useState<any[]>([]);
  const [loading, setLoading] = useState(true);
  const [searchTerm, setSearchTerm] = useState('');
  const [statusFilter, setStatusFilter] = useState('All');

  useEffect(() => {
    fetchApplications();
  }, [statusFilter, searchTerm]);

  const fetchApplications = async () => {
    try {
      setLoading(true);

      // Get token from localStorage
      const token = localStorage.getItem('auth_token');

      let url = '/api/applications';
      const params = new URLSearchParams();
```

```
    if (statusFilter !== 'All') {
      params.append('status', statusFilter);
    }
    if (searchTerm) {
      params.append('search', searchTerm);
    }

    if (params.toString()) {
      url += '?' + params.toString();
    }

    const response = await fetch(url, {
      headers: token ? { 'Authorization': `Bearer ${token}` } : {},
    });

    if (response.ok) {
      const data = await response.json();
      // Ensure it's an array
      if (Array.isArray(data)) {
        setApplications(data);
      } else {
        console.warn('Applications API returned non-array:', data);
        setApplications([]);
      }
    } else {
      console.error('Failed to fetch applications:', await response.text());
      setApplications([]);
    }
  } catch (error) {
    console.error('Error fetching applications:', error);
    setApplications([]);
  } finally {
    setLoading(false);
  }
};
```

```
  const getStatusColor = (status: string) => {
    switch (status) {
      case 'Approved':
        return 'bg-green-100 text-green-700';
      case 'Pending':
        return 'bg-yellow-100 text-yellow-700';
      case 'Under Review':
        return 'bg-blue-100 text-blue-700';
      case 'Rejected':
        return 'bg-red-100 text-red-700';
      default:
        return 'bg-gray-100 text-gray-700';
    }
  };

  return (
    <div className="space-y-6">
      {/* Header */}
      <div className="flex items-center justify-between">
        <div>
          <h1 className="text-3xl font-bold text-gray-900">Loan Applications</
h1>
          <p className="text-gray-600 mt-1">Manage and review all loan applic
ations</p>
        </div>
        <button className="bg-blue-600 text-white px-4 py-2 rounded-lg font-
bold hover:bg-blue-700 transition flex items-center gap-2">
          <Plus className="w-5 h-5" /> New Application
        </button>
      </div>

      {/* Filters & Search */}
      <div className="bg-white p-6 rounded-lg border border-gray-200 space
-y-4">
        <div className="flex items-center gap-4 flex-wrap">
```

```
        {/* Search */}
        <div className="flex-1 min-w-64">
          <div className="relative">
            <Search className="absolute left-3 top-1/2 -translate-y-1/2 w-5 h-5
text-gray-400" />
            <input
              type="text"
              placeholder="Search by name or email..."
              value={searchTerm}
              onChange={(e) ⇒ setSearchTerm(e.target.value)}
              className="w-full pl-10 pr-4 py-2 border border-gray-300 rounded
-lg focus:outline-none focus:ring-2 focus:ring-blue-500"
            />
          </div>
        </div>

        {/* Filter */}
        <select
          value={statusFilter}
          onChange={(e) ⇒ setStatusFilter(e.target.value)}
          className="px-4 py-2 border border-gray-300 rounded-lg focus:outli
ne-none focus:ring-2 focus:ring-blue-500"
        >
          <option>All</option>
          <option>Pending</option>
          <option>Under Review</option>
          <option>Approved</option>
          <option>Rejected</option>
        </select>

        {/* Export */}
        <button className="px-4 py-2 border border-gray-300 rounded-lg hov
er:bg-gray-50 transition flex items-center gap-2">
          <Download className="w-5 h-5" /> Export
        </button>
      </div>
```

```
    </div>

    {/* Table */}
    <div className="bg-white rounded-lg border border-gray-200 overflow-
hidden">
      <div className="overflow-x-auto">
        <table className="w-full">
          <thead className="bg-gray-50 border-b border-gray-200">
            <tr>
              <th className="px-6 py-3 text-left text-sm font-bold text-gray-70
0">Applicant</th>
              <th className="px-6 py-3 text-left text-sm font-bold text-gray-70
0">Product</th>
              <th className="px-6 py-3 text-left text-sm font-bold text-gray-70
0">Amount</th>
              <th className="px-6 py-3 text-left text-sm font-bold text-gray-70
0">Status</th>
              <th className="px-6 py-3 text-left text-sm font-bold text-gray-70
0">Date</th>
              <th className="px-6 py-3 text-left text-sm font-bold text-gray-70
0">Actions</th>
            </tr>
          </thead>
          <tbody>
            {loading ? (
              <tr>
                <td colSpan={6} className="px-6 py-8 text-center text-gray-60
0">
                  Loading...
                </td>
              </tr>
            ) : applications.length === 0 ? (
              <tr>
                <td colSpan={6} className="px-6 py-8 text-center text-gray-60
0">
                  No applications found. Start by creating a new application.
```

```
              </td>
            </tr>
          ) : (
            applications.map((app: any) => (
              <tr key={app._id} className="border-b border-gray-200 hover:bg
-gray-50 transition">
                <td className="px-6 py-3">
                  <div>
                    <p className="font-bold text-gray-900">{app.fullName}</p>
                    <p className="text-xs text-gray-600">{app.email}</p>
                  </div>
                </td>
                <td className="px-6 py-3 text-sm text-gray-700">{app.loanTyp
e}</td>
                <td className="px-6 py-3 text-sm font-bold text-gray-900">{ap
p.loanAmount?.toLocaleString() || 0}</td>
                <td className="px-6 py-3">
                  <span className={`px-3 py-1 rounded-full text-xs font-bold ${g
etStatusColor(app.status)}`}>
                    {app.status}
                  </span>
                </td>
                <td className="px-6 py-3 text-sm text-gray-700">
                  {app.createdAt ? new Date(app.createdAt).toLocaleDateString() :
'N/A'}
                </td>
                <td className="px-6 py-3">
                  <div className="flex items-center gap-2">
                    <Link
                      href={`/dashboard/loan-applications/${app._id}`}
                      className="p-2 text-blue-600 hover:bg-blue-50 rounded-lg t
ransition"
                    >
                      <Eye className="w-5 h-5" />
                    </Link>
                    <button className="p-2 text-green-600 hover:bg-green-50 ro
```

```
unded-lg transition">
                    <Edit2 className="w-5 h-5" />
                  </button>
                  <button className="p-2 text-red-600 hover:bg-red-50 rounde
d-lg transition">
                    <Trash2 className="w-5 h-5" />
                  </button>
                </div>
              </td>
            </tr>
          ))
        )}
      </tbody>
    </table>
  </div>

  {/* Pagination */}
  <div className="px-6 py-4 border-t border-gray-200 flex items-center j
ustify-between">
    <p className="text-sm text-gray-600">
      Showing {applications.length} result{applications.length !== 1 ? 's' : ''}
    </p>
    <div className="flex gap-2">
      <button className="px-3 py-2 border border-gray-300 rounded-lg ho
ver:bg-gray-50">Previous</button>
      <button className="px-3 py-2 bg-blue-600 text-white rounded-lg">1
</button>
      <button className="px-3 py-2 border border-gray-300 rounded-lg ho
ver:bg-gray-50">Next</button>
    </div>
  </div>
 </div>
 );
}
```

```
// app/dashboard/loan-products/[id]/edit/page.tsx
'use client';

import { useState, useEffect } from 'react';
import { useRouter, useParams } from 'next/navigation';
import { ArrowLeft, Save } from 'lucide-react';

export default function EditProductPage() {
  const router = useRouter();
  const params = useParams();
  const [loading, setLoading] = useState(true);
  const [saving, setSaving] = useState(false);
  const [formData, setFormData] = useState({
    name: '',
    description: '',
    minAmount: '',
    maxAmount: '',
    minTenure: '',
    maxTenure: '',
    minInterest: '',
    maxInterest: '',
    processingFee: '',
    status: 'Active',
    features: '',
    eligibility: '',
  });

  useEffect(() => {
    fetchProduct();
  }, []);

  const fetchProduct = async () => {
    try {
      const token = localStorage.getItem('auth_token');
      const response = await fetch(`/api/products/${params.id}`, {
```

```
        headers: token ? { 'Authorization': `Bearer ${token}` } : {},
      });

      if (response.ok) {
        const data = await response.json();
        setFormData({
          name: data.name || '',
          description: data.description || '',
          minAmount: data.minAmount?.toString() || '',
          maxAmount: data.maxAmount?.toString() || '',
          minTenure: data.minTenure?.toString() || '',
          maxTenure: data.maxTenure?.toString() || '',
          minInterest: data.minInterest?.toString() || '',
          maxInterest: data.maxInterest?.toString() || '',
          processingFee: data.processingFee?.toString() || '',
          status: data.status || 'Active',
          features: Array.isArray(data.features) ? data.features.join(', ') : '',
          eligibility: Array.isArray(data.eligibility) ? data.eligibility.join(', ') : '',
        });
      }
    } catch (error) {
      console.error('Error fetching product:', error);
    } finally {
      setLoading(false);
    }
  };

  const handleChange = (e: React.ChangeEvent<HTMLInputElement | HTMLTe
xtAreaElement | HTMLSelectElement>) => {
    const { name, value } = e.target;
    setFormData({ ...formData, [name]: value });
  };

  const handleSubmit = async (e: React.FormEvent) => {
    e.preventDefault();
    setSaving(true);
```

```
try {
  const token = localStorage.getItem('auth_token');

  const payload = {
    ...formData,
    minAmount: parseFloat(formData.minAmount),
    maxAmount: parseFloat(formData.maxAmount),
    minTenure: parseInt(formData.minTenure),
    maxTenure: parseInt(formData.maxTenure),
    minInterest: parseFloat(formData.minInterest),
    maxInterest: parseFloat(formData.maxInterest),
    processingFee: parseFloat(formData.processingFee),
    features: formData.features.split(',').map(f => f.trim()).filter(f => f),
    eligibility: formData.eligibility.split(',').map(e => e.trim()).filter(e => e),
  };

  const response = await fetch(`/api/products/${params.id}`, {
    method: 'PUT',
    headers: {
      'Content-Type': 'application/json',
      ...(token && { 'Authorization': `Bearer ${token}` }),
    },
    body: JSON.stringify(payload),
  });

  if (response.ok) {
    alert('Product updated successfully!');
    router.push('/dashboard/loan-products');
  } else {
    const error = await response.json();
    alert(`Error: ${error.error || 'Failed to update product'}`);
  }
} catch (error) {
  console.error('Error updating product:', error);
  alert('Failed to update product');
```

```
    } finally {
      setSaving(false);
    }
  };

  if (loading) {
    return (
      <div className="flex items-center justify-center h-96">
        <div className="text-center">
          <div className="inline-block animate-spin rounded-full h-12 w-12 bor
der-b-2 border-blue-600"></div>
          <p className="mt-4 text-gray-600">Loading product...</p>
        </div>
      </div>
    );
  }

  return (
    <div className="space-y-6">
      {/* Header */}
      <div className="flex items-center gap-4">
        <button
          onClick={() => router.back()}
          className="p-2 hover:bg-gray-200 rounded-lg transition"
        >
          <ArrowLeft className="w-6 h-6" />
        </button>
        <div>
          <h1 className="text-3xl font-bold text-gray-900">Edit Loan Product</
h1>
          <p className="text-gray-600 mt-1">Update product details</p>
        </div>
      </div>

      {/* Form - Same as create form */}
      <form onSubmit={handleSubmit} className="bg-white rounded-lg borde
```

```
r border-gray-200 p-8">
      <div className="grid grid-cols-1 md:grid-cols-2 gap-6">
        {/* Same fields as create form */}
        <div className="md:col-span-2">
          <label className="block text-sm font-bold text-gray-700 mb-2">Prod
uct Name *</label>
          <input
            type="text"
            name="name"
            value={formData.name}
            onChange={handleChange}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
          />
        </div>

        <div className="md:col-span-2">
          <label className="block text-sm font-bold text-gray-700 mb-2">Des
cription *</label>
          <textarea
            name="description"
            value={formData.description}
            onChange={handleChange}
            rows={3}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
          />
        </div>

        <div>
          <label className="block text-sm font-bold text-gray-700 mb-2">Mini
mum Amount *</label>
          <input
            type="number"
```

```
            name="minAmount"
            value={formData.minAmount}
            onChange={handleChange}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
          />
        </div>

        <div>
          <label className="block text-sm font-bold text-gray-700 mb-2">Maxi
mum Amount *</label>
          <input
            type="number"
            name="maxAmount"
            value={formData.maxAmount}
            onChange={handleChange}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
          />
        </div>

        <div>
          <label className="block text-sm font-bold text-gray-700 mb-2">Min
Tenure (months) *</label>
          <input
            type="number"
            name="minTenure"
            value={formData.minTenure}
            onChange={handleChange}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
          />
        </div>
```

```
        <div>
          <label className="block text-sm font-bold text-gray-700 mb-2">Max
Tenure (months) *</label>
          <input
            type="number"
            name="maxTenure"
            value={formData.maxTenure}
            onChange={handleChange}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
          />
        </div>

        <div>
          <label className="block text-sm font-bold text-gray-700 mb-2">Min
Interest (%) *</label>
          <input
            type="number"
            step="0.01"
            name="minInterest"
            value={formData.minInterest}
            onChange={handleChange}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
          />
        </div>

        <div>
          <label className="block text-sm font-bold text-gray-700 mb-2">Max
Interest (%) *</label>
          <input
            type="number"
            step="0.01"
```

```
            name="maxInterest"
            value={formData.maxInterest}
            onChange={handleChange}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
          />
        </div>

        <div>
          <label className="block text-sm font-bold text-gray-700 mb-2">Proc
essing Fee (%) *</label>
          <input
            type="number"
            step="0.01"
            name="processingFee"
            value={formData.processingFee}
            onChange={handleChange}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
          />
        </div>

        <div>
          <label className="block text-sm font-bold text-gray-700 mb-2">Stat
us *</label>
          <select
            name="status"
            value={formData.status}
            onChange={handleChange}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
          >
            <option value="Active">Active</option>
```

```
            <option value="Inactive">Inactive</option>
          </select>
        </div>

        <div className="md:col-span-2">
          <label className="block text-sm font-bold text-gray-700 mb-2">Feat
ures (comma-separated)</label>
          <input
            type="text"
            name="features"
            value={formData.features}
            onChange={handleChange}
            placeholder="Quick approval, Flexible repayment"
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
          />
        </div>

        <div className="md:col-span-2">
          <label className="block text-sm font-bold text-gray-700 mb-2">Eligi
bility (comma-separated)</label>
          <input
            type="text"
            name="eligibility"
            value={formData.eligibility}
            onChange={handleChange}
            placeholder="Valid ID, Income proof"
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
          />
        </div>
      </div>

      <div className="flex gap-4 mt-8">
        <button
          type="button"
```

```
          onClick={() ⇒ router.back()}
          className="flex-1 px-6 py-2 border border-gray-300 text-gray-700 ro
unded-lg font-bold hover:bg-gray-50 transition"
        >
          Cancel
        </button>
        <button
          type="submit"
          disabled={saving}
          className="flex-1 px-6 py-2 bg-blue-600 text-white rounded-lg font-
bold hover:bg-blue-700 transition disabled:opacity-50 flex items-center justif
y-center gap-2"
        >
          <Save className="w-5 h-5" />
          {saving ? 'Saving...' : 'Save Changes'}
        </button>
      </div>
    </form>
  </div>
 );
}
```

```
// app/dashboard/loan-products/create/page.tsx
'use client';

import { useState } from 'react';
import { useRouter } from 'next/navigation';
import { ArrowLeft, Save } from 'lucide-react';

export default function CreateProductPage() {
 const router = useRouter();
 const [loading, setLoading] = useState(false);
 const [formData, setFormData] = useState({
   name: '',
   description: '',
```

```
    minAmount: '',
    maxAmount: '',
    minTenure: '',
    maxTenure: '',
    minInterest: '',
    maxInterest: '',
    processingFee: '',
    status: 'Active',
    features: '',
    eligibility: '',
  });

  const handleChange = (e: React.ChangeEvent<HTMLInputElement | HTMLTe
xtAreaElement | HTMLSelectElement>) => {
    const { name, value } = e.target;
    setFormData({ ...formData, [name]: value });
  };

  const handleSubmit = async (e: React.FormEvent) => {
    e.preventDefault();
    setLoading(true);

    try {
      const token = localStorage.getItem('auth_token');

      // Convert features and eligibility from comma-separated to arrays
      const payload = {
        ...formData,
        minAmount: parseFloat(formData.minAmount),
        maxAmount: parseFloat(formData.maxAmount),
        minTenure: parseInt(formData.minTenure),
        maxTenure: parseInt(formData.maxTenure),
        minInterest: parseFloat(formData.minInterest),
        maxInterest: parseFloat(formData.maxInterest),
        processingFee: parseFloat(formData.processingFee),
        features: formData.features.split(',').map(f => f.trim()).filter(f => f),
```

```
      eligibility: formData.eligibility.split(',').map(e => e.trim()).filter(e => e),
    };

    const response = await fetch('/api/products', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
        ...(token && { 'Authorization': `Bearer ${token}` }),
      },
      body: JSON.stringify(payload),
    });

    if (response.ok) {
      alert('Product created successfully!');
      router.push('/dashboard/loan-products');
    } else {
      const error = await response.json();
      alert(`Error: ${error.error || 'Failed to create product'}`);
    }
  } catch (error) {
    console.error('Error creating product:', error);
    alert('Failed to create product');
  } finally {
    setLoading(false);
  }
};

return (
  <div className="space-y-6">
    {/* Header */}
    <div className="flex items-center gap-4">
      <button
        onClick={() => router.back()}
        className="p-2 hover:bg-gray-200 rounded-lg transition"
      >
        <ArrowLeft className="w-6 h-6" />
```

```
        </button>
        <div>
          <h1 className="text-3xl font-bold text-gray-900">Create Loan Produc
t</h1>
          <p className="text-gray-600 mt-1">Add a new loan product to the sys
tem</p>
        </div>
      </div>

      {/* Form */}
      <form onSubmit={handleSubmit} className="bg-white rounded-lg borde
r border-gray-200 p-8">
        <div className="grid grid-cols-1 md:grid-cols-2 gap-6">
          {/* Product Name */}
          <div className="md:col-span-2">
            <label className="block text-sm font-bold text-gray-700 mb-2">
              Product Name *
            </label>
            <input
              type="text"
              name="name"
              value={formData.name}
              onChange={handleChange}
              className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
              required
            />
          </div>

          {/* Description */}
          <div className="md:col-span-2">
            <label className="block text-sm font-bold text-gray-700 mb-2">
              Description *
            </label>
            <textarea
              name="description"
```

```
            value={formData.description}
            onChange={handleChange}
            rows={3}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
          />
        </div>

        {/* Min Amount */}
        <div>
          <label className="block text-sm font-bold text-gray-700 mb-2">
            Minimum Amount *
          </label>
          <input
            type="number"
            name="minAmount"
            value={formData.minAmount}
            onChange={handleChange}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
          />
        </div>

        {/* Max Amount */}
        <div>
          <label className="block text-sm font-bold text-gray-700 mb-2">
            Maximum Amount *
          </label>
          <input
            type="number"
            name="maxAmount"
            value={formData.maxAmount}
            onChange={handleChange}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
```

```
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
          />
        </div>

        {/* Min Tenure */}
        <div>
          <label className="block text-sm font-bold text-gray-700 mb-2">
            Minimum Tenure (months) *
          </label>
          <input
            type="number"
            name="minTenure"
            value={formData.minTenure}
            onChange={handleChange}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
          />
        </div>

        {/* Max Tenure */}
        <div>
          <label className="block text-sm font-bold text-gray-700 mb-2">
            Maximum Tenure (months) *
          </label>
          <input
            type="number"
            name="maxTenure"
            value={formData.maxTenure}
            onChange={handleChange}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
          />
        </div>
```

```
        {/* Min Interest */}
        <div>
          <label className="block text-sm font-bold text-gray-700 mb-2">
            Minimum Interest (%) *
          </label>
          <input
            type="number"
            step="0.01"
            name="minInterest"
            value={formData.minInterest}
            onChange={handleChange}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
          />
        </div>

        {/* Max Interest */}
        <div>
          <label className="block text-sm font-bold text-gray-700 mb-2">
            Maximum Interest (%) *
          </label>
          <input
            type="number"
            step="0.01"
            name="maxInterest"
            value={formData.maxInterest}
            onChange={handleChange}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
          />
        </div>

        {/* Processing Fee */}
```

```
        <div>
          <label className="block text-sm font-bold text-gray-700 mb-2">
            Processing Fee (%) *
          </label>
          <input
            type="number"
            step="0.01"
            name="processingFee"
            value={formData.processingFee}
            onChange={handleChange}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
          />
        </div>

        {/* Status */}
        <div>
          <label className="block text-sm font-bold text-gray-700 mb-2">
            Status *
          </label>
          <select
            name="status"
            value={formData.status}
            onChange={handleChange}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
          >
            <option value="Active">Active</option>
            <option value="Inactive">Inactive</option>
          </select>
        </div>

        {/* Features */}
        <div className="md:col-span-2">
```

```
          <label className="block text-sm font-bold text-gray-700 mb-2">
            Features (comma-separated)
          </label>
          <input
            type="text"
            name="features"
            value={formData.features}
            onChange={handleChange}
            placeholder="Quick approval, Flexible repayment, No collateral"
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
          />
          <p className="text-xs text-gray-600 mt-1">Separate each feature wi
th a comma</p>
        </div>

        {/* Eligibility */}
        <div className="md:col-span-2">
          <label className="block text-sm font-bold text-gray-700 mb-2">
            Eligibility Requirements (comma-separated)
          </label>
          <input
            type="text"
            name="eligibility"
            value={formData.eligibility}
            onChange={handleChange}
            placeholder="Valid ID, Proof of income, Business registration"
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
          />
          <p className="text-xs text-gray-600 mt-1">Separate each requireme
nt with a comma</p>
        </div>
      </div>

      {/* Buttons */}
```

```
        <div className="flex gap-4 mt-8">
          <button
            type="button"
            onClick={() ⇒ router.back()}
            className="flex-1 px-6 py-2 border border-gray-300 text-gray-700 ro
unded-lg font-bold hover:bg-gray-50 transition"
          >
            Cancel
          </button>
          <button
            type="submit"
            disabled={loading}
            className="flex-1 px-6 py-2 bg-blue-600 text-white rounded-lg font-
bold hover:bg-blue-700 transition disabled:opacity-50 flex items-center justif
y-center gap-2"
          >
            <Save className="w-5 h-5" />
            {loading ? 'Creating...' : 'Create Product'}
          </button>
        </div>
      </form>
    </div>
  );
}
```

```
// app/dashboard/loan-products/page.tsx
'use client';

import { useState, useEffect } from 'react';
import { Plus, Edit2, Trash2, Eye, Search } from 'lucide-react';
import Link from 'next/link';

interface Product {
  _id: string;
  name: string;
```

```
  minAmount: number;
  maxAmount: number;
  minTenure: number;
  maxTenure: number;
  minInterest: number;
  maxInterest: number;
  status: string;
}

export default function ProductsPage() {
  const [products, setProducts] = useState<Product[]>([]);
  const [searchTerm, setSearchTerm] = useState('');
  const [loading, setLoading] = useState(true);
  const [deleting, setDeleting] = useState<string | null>(null);

  useEffect(() => {
    fetchProducts();
  }, []);

  const fetchProducts = async () => {
    try {
      setLoading(true);
      const response = await fetch('/api/products');
      const data = await response.json();
      setProducts(data);
    } catch (error) {
      console.error('Error fetching products:', error);
    } finally {
      setLoading(false);
    }
  };

  const filteredProducts = products.filter((product) =>
    product.name.toLowerCase().includes(searchTerm.toLowerCase())
  );
```

```
  const handleDelete = async (id: string) => {
    if (!confirm('Are you sure you want to delete this product?')) return;

    try {
      setDeleting(id);
      const response = await fetch(`/api/products/${id}`, {
        method: 'DELETE',
      });

      if (response.ok) {
        setProducts(products.filter((p) => p._id !== id));
        alert('Product deleted successfully');
      }
    } catch (error) {
      console.error('Error deleting product:', error);
      alert('Failed to delete product');
    } finally {
      setDeleting(null);
    }
  };

  return (
    <div className="space-y-6">
      {/* Header */}
      <div className="flex items-center justify-between">
        <div>
          <h1 className="text-3xl font-bold text-gray-900">Loan Products</h1>
          <p className="text-gray-600 mt-1">Manage loan products and their d
etails</p>
        </div>
        <Link
          href="/dashboard/loan-products/create"
          className="bg-blue-600 hover:bg-blue-700 px-4 py-2 rounded-lg fon
t-medium transition flex items-center gap-2"
          style={{ color: '#ffffff' }}
        >
```

```
        <Plus className="w-5 h-5" style={{ color: '#ffffff' }} />
        <span style={{ color: '#ffffff' }}>New Product</span>
      </Link>
    </div>

    {/* Search */}
    <div className="bg-white p-4 rounded-lg border border-gray-200">
      <div className="relative">
        <Search className="absolute left-3 top-1/2 -translate-y-1/2 w-5 h-5 te
xt-gray-400" />
        <input
          type="text"
          placeholder="Search products..."
          value={searchTerm}
          onChange={(e) => setSearchTerm(e.target.value)}
          className="w-full pl-10 pr-4 py-2 border border-gray-300 rounded-l
g focus:outline-none focus:ring-2 focus:ring-blue-500"
        />
      </div>
    </div>

    {/* Products Grid */}
    <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-6">
      {loading ? (
        <div className="col-span-full text-center py-8 text-gray-600">Loadin
g...</div>
      ) : filteredProducts.length === 0 ? (
        <div className="col-span-full text-center py-8 text-gray-600">No pro
ducts found</div>
      ) : (
        filteredProducts.map((product) => (
          <div key={product._id} className="bg-white rounded-lg border bord
er-gray-200 p-6 hover:shadow-lg transition">
            <h3 className="font-bold text-gray-900 mb-2">{product.name}</h
3>
            <span
```

```
            className={`text-xs font-bold px-2 py-1 rounded-full inline-block m
b-4 ${product.status === 'Active'
              ? 'bg-green-100 text-green-700'
              : 'bg-red-100 text-red-700'
              }`}
            >
              {product.status}
            </span>

            <div className="space-y-2 text-sm text-gray-700 mb-4">
              <p><strong>Amount:</strong> {product.minAmount.toLocaleString
()} - {product.maxAmount.toLocaleString()}</p>
              <p><strong>Tenure:</strong> {product.minTenure} - {product.maxT
enure} months</p>
              <p><strong>Interest:</strong> {product.minInterest}% - {product.m
axInterest}%</p>
            </div>

            <div className="flex gap-2 border-t border-gray-200 pt-4">
              <button className="flex-1 p-2 bg-blue-50 text-blue-600 rounded-l
g hover:bg-blue-100 transition flex items-center justify-center gap-2">
                <Eye className="w-4 h-4" /> View
              </button>
              <Link
                href={`/dashboard/loan-products/${product._id}/edit`}
                className="flex-1 p-2 bg-green-50 text-green-600 rounded-lg ho
ver:bg-green-100 transition flex items-center justify-center gap-2"
              >
                <Edit2 className="w-4 h-4" /> Edit
              </Link>
              <button
                onClick={() => handleDelete(product._id)}
                disabled={deleting === product._id}
                className="flex-1 p-2 bg-red-50 text-red-600 rounded-lg hover:b
g-red-100 transition flex items-center justify-center gap-2 disabled:opacity-5
0"
```

```
            >
              <Trash2 className="w-4 h-4" /> Delete
            </button>
          </div>
        </div>
      ))
    )}
  </div>
 </div>
);
}
```

```
// app/dashboard/reports/page.tsx
'use client';

import { useState, useEffect } from 'react';
import { BarChart, Bar, LineChart, Line, XAxis, YAxis, CartesianGrid, Tooltip, L
egend, ResponsiveContainer } from 'recharts';
import { Download, TrendingUp } from 'lucide-react';

const reportData = [
  { month: 'Jan', applications: 40, approved: 24, rejected: 8, pending: 8 },
  { month: 'Feb', applications: 52, approved: 35, rejected: 10, pending: 7 },
  { month: 'Mar', applications: 48, approved: 28, rejected: 12, pending: 8 },
  { month: 'Apr', applications: 61, approved: 42, rejected: 15, pending: 4 },
  { month: 'May', applications: 55, approved: 38, rejected: 12, pending: 5 },
  { month: 'Jun', applications: 67, approved: 45, rejected: 18, pending: 4 },
];

interface Stats {
  totalApplications: number;
  approvedApplications: number;
  pendingApplications: number;
  rejectedApplications: number;
  totalDisbursed: number;
```

```
  averageLoan: number;
  approvalRate: number;
}

export default function ReportsPage() {
  const [stats, setStats] = useState<Stats | null>(null);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    fetchStats();
  }, []);

  const fetchStats = async () => {
    try {
      const response = await fetch('/api/stats');
      const data = await response.json();
      setStats(data);
    } catch (error) {
      console.error('Error fetching stats:', error);
    } finally {
      setLoading(false);
    }
  };

  if (loading || !stats) {
    return (
      <div className="flex items-center justify-center h-96">
        <div className="text-center">
          <div className="inline-block animate-spin rounded-full h-12 w-12 bor
der-b-2 border-blue-600"></div>
          <p className="mt-4 text-gray-600">Loading reports...</p>
        </div>
      </div>
    );
  }
```

```
  return (
    <div className="space-y-6">
      <div className="flex items-center justify-between">
        <h1 className="text-3xl font-bold text-gray-900">Reports & Analytics</
h1>
        <button className="bg-blue-600 text-white px-4 py-2 rounded-lg font-
bold hover:bg-blue-700 flex items-center gap-2">
          <Download className="w-5 h-5" /> Export Report
        </button>
      </div>

      {/* Key Metrics */}
      <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-4">
        <div className="bg-white p-6 rounded-lg border border-gray-200">
          <p className="text-gray-600 text-sm">Total Applications</p>
          <p className="text-3xl font-bold text-blue-600 mt-2">{stats.totalAppli
cations}</p>
          <p className="text-xs text-green-600 mt-2 flex items-center gap-1">
            <TrendingUp className="w-3 h-3" /> All time
          </p>
        </div>
        <div className="bg-white p-6 rounded-lg border border-gray-200">
          <p className="text-gray-600 text-sm">Approved Loans</p>
          <p className="text-3xl font-bold text-green-600 mt-2">{stats.approve
dApplications}</p>
          <p className="text-xs text-gray-600 mt-2">{stats.approvalRate}% ap
proval rate</p>
        </div>
        <div className="bg-white p-6 rounded-lg border border-gray-200">
          <p className="text-gray-600 text-sm">Funds Disbursed</p>
          <p className="text-3xl font-bold text-purple-600 mt-2">{(stats.totalDi
sbursed / 1000000).toFixed(2)}M</p>
          <p className="text-xs text-gray-600 mt-2">Total amount</p>
        </div>
        <div className="bg-white p-6 rounded-lg border border-gray-200">
          <p className="text-gray-600 text-sm">Average Loan</p>
```

```
        <p className="text-3xl font-bold text-orange-600 mt-2">{stats.averag
eLoan.toLocaleString()}</p>
        <p className="text-xs text-gray-600 mt-2">Per applicant</p>
       </div>
     </div>

     {/* Charts */}
     <div className="grid grid-cols-1 lg:grid-cols-2 gap-6">
      {/* Application Trends */}
      <div className="bg-white p-6 rounded-lg border border-gray-200">
       <h2 className="text-lg font-bold text-gray-900 mb-4">Application Tr
ends</h2>
       <ResponsiveContainer width="100%" height={350}>
        <LineChart data={reportData}>
         <CartesianGrid strokeDasharray="3 3" />
         <XAxis dataKey="month" />
         <YAxis />
         <Tooltip />
         <Legend />
         <Line type="monotone" dataKey="applications" stroke="#2563eb" st
rokeWidth={2} />
         <Line type="monotone" dataKey="approved" stroke="#16a34a" strok
eWidth={2} />
         <Line type="monotone" dataKey="rejected" stroke="#ef4444" stroke
Width={2} />
        </LineChart>
       </ResponsiveContainer>
      </div>

      {/* Status Distribution */}
      <div className="bg-white p-6 rounded-lg border border-gray-200">
       <h2 className="text-lg font-bold text-gray-900 mb-4">Application St
atus Distribution</h2>
       <ResponsiveContainer width="100%" height={350}>
        <BarChart data={reportData}>
         <CartesianGrid strokeDasharray="3 3" />
```

```
            <XAxis dataKey="month" />
            <YAxis />
            <Tooltip />
            <Legend />
            <Bar dataKey="approved" fill="#16a34a" />
            <Bar dataKey="rejected" fill="#ef4444" />
            <Bar dataKey="pending" fill="#f59e0b" />
          </BarChart>
        </ResponsiveContainer>
      </div>
    </div>

    {/* Summary Stats */}
    <div className="bg-white p-6 rounded-lg border border-gray-200">
      <h2 className="text-lg font-bold text-gray-900 mb-6">Summary Statis
tics</h2>
      <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-4">
        <div className="border-l-4 border-blue-600 pl-4">
          <p className="text-gray-600 text-sm">Total Applications</p>
          <p className="text-2xl font-bold text-gray-900">{stats.totalApplicati
ons}</p>
        </div>
        <div className="border-l-4 border-green-600 pl-4">
          <p className="text-gray-600 text-sm">Approved</p>
          <p className="text-2xl font-bold text-gray-900">{stats.approvedApp
lications}</p>
        </div>
        <div className="border-l-4 border-yellow-600 pl-4">
          <p className="text-gray-600 text-sm">Pending Review</p>
          <p className="text-2xl font-bold text-gray-900">{stats.pendingAppli
cations}</p>
        </div>
        <div className="border-l-4 border-red-600 pl-4">
          <p className="text-gray-600 text-sm">Rejected</p>
          <p className="text-2xl font-bold text-gray-900">{stats.rejectedAppli
cations}</p>
```

```
        </div>
       </div>
      </div>
    </div>
  );
}
```

```
// app/dashboard/users/[id]/edit/page.tsx
'use client';
import { useState, useEffect } from 'react';
import { useRouter, useParams } from 'next/navigation';
import { ArrowLeft, Save, AlertCircle } from 'lucide-react';

export default function EditUserPage() {
  const router = useRouter();
  const params = useParams();
  const [loading, setLoading] = useState(true);
  const [saving, setSaving] = useState(false);
  const [error, setError] = useState('');
  const [formData, setFormData] = useState({
    name: '',
    email: '',
    role: 'officer',
    status: 'Active',
  });

  useEffect(() => {
    fetchUser();
  }, []);

  const fetchUser = async () => {
    try {
      setLoading(true);
      setError('');
```

```
      const token = localStorage.getItem('auth_token');
      const response = await fetch(`/api/users/${params.id}`, {
        headers: token ? { 'Authorization': `Bearer ${token}` } : {},
      });

      if (response.ok) {
        const data = await response.json();
        setFormData({
          name: data.name || '',
          email: data.email || '',
          role: data.role || 'officer',
          status: data.status || 'Active',
        });
      } else {
        const errorData = await response.json().catch(() => ({ error: 'Failed to loa
d user' }));
        setError(errorData.error || 'Failed to load user data');
      }
    } catch (error) {
      console.error('Error fetching user:', error);
      setError('Network error: Could not connect to the server');
    } finally {
      setLoading(false);
    }
  };

  const handleChange = (e: React.ChangeEvent<HTMLInputElement | HTMLSe
lectElement>) => {
    const { name, value } = e.target;
    setFormData({ ...formData, [name]: value });
  };

  const handleSubmit = async (e: React.FormEvent) => {
    e.preventDefault();
    setSaving(true);
```

```
    try {
      const token = localStorage.getItem('auth_token');
      const response = await fetch(`/api/users/${params.id}`, {
        method: 'PUT',
        headers: {
          'Content-Type': 'application/json',
          ...(token && { 'Authorization': `Bearer ${token}` }),
        },
        body: JSON.stringify(formData),
      });

      if (response.ok) {
        alert('User updated successfully!');
        router.push('/dashboard/users');
      } else {
        const error = await response.json();
        alert(`Error: ${error.error || 'Failed to update user'}`);
      }
    } catch (error) {
      console.error('Error updating user:', error);
      alert('Network error: Failed to update user');
    } finally {
      setSaving(false);
    }
  };

  // Show loading spinner while data is being fetched
  if (loading) {
    return (
      <div className="flex items-center justify-center h-96">
        <div className="text-center">
          <div className="inline-block animate-spin rounded-full h-12 w-12 bor
der-b-2 border-blue-600"></div>
          <p className="mt-4 text-gray-600">Loading user data...</p>
        </div>
      </div>
```

```
  );
 }

 // Show error state with retry option
 if (error) {
  return (
   <div className="space-y-6">
    <div className="flex items-center gap-4">
     <button
      onClick={() => router.back()}
      className="p-2 hover:bg-gray-200 rounded-lg transition"
     >
      <ArrowLeft className="w-6 h-6" />
     </button>
     <div>
      <h1 className="text-3xl font-bold text-gray-900">Edit User</h1>
     </div>
    </div>

    <div className="bg-white rounded-lg border border-red-200 p-8">
     <div className="flex items-start gap-4">
      <AlertCircle className="w-8 h-8 text-red-600 flex-shrink-0 mt-1" />
      <div className="flex-1">
       <h3 className="text-lg font-bold text-red-900 mb-2">Failed to Loa
d User</h3>
       <p className="text-red-700 mb-4">{error}</p>
       <div className="flex gap-4">
        <button
         onClick={() => fetchUser()}
         className="px-6 py-2 bg-blue-600 text-white rounded-lg font-bol
d hover:bg-blue-700 transition"
        >
         Retry
        </button>
        <button
         onClick={() => router.push('/dashboard/users')}
```

```
              className="px-6 py-2 border border-gray-300 text-gray-700 roun
ded-lg font-bold hover:bg-gray-50 transition"
            >
              Back to Users List
            </button>
          </div>
        </div>
      </div>
    </div>
  );
 }

 // Only render form after data is loaded successfully
 return (
  <div className="space-y-6">
   {/* Header */}
   <div className="flex items-center gap-4">
    <button
     onClick={() => router.back()}
     className="p-2 hover:bg-gray-200 rounded-lg transition"
    >
     <ArrowLeft className="w-6 h-6" />
    </button>
    <div>
     <h1 className="text-3xl font-bold text-gray-900">Edit User</h1>
     <p className="text-gray-600 mt-1">Update user details</p>
    </div>
   </div>

   {/* Form */}
   <form onSubmit={handleSubmit} className="bg-white rounded-lg borde
r border-gray-200 p-8">
    <div className="grid grid-cols-1 md:grid-cols-2 gap-6">
     {/* Name */}
     <div className="md:col-span-2">
```

```
          <label className="block text-sm font-bold text-gray-700 mb-2">
            Full Name *
          </label>
          <input
            type="text"
            name="name"
            value={formData.name}
            onChange={handleChange}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
          />
        </div>

        {/* Email */}
        <div className="md:col-span-2">
          <label className="block text-sm font-bold text-gray-700 mb-2">
            Email Address *
          </label>
          <input
            type="email"
            name="email"
            value={formData.email}
            onChange={handleChange}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
          />
        </div>

        {/* Role */}
        <div>
          <label className="block text-sm font-bold text-gray-700 mb-2">
            Role *
          </label>
          <select
```

```
            name="role"
            value={formData.role}
            onChange={handleChange}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
          >
            <option value="officer">Officer</option>
            <option value="manager">Manager</option>
            <option value="admin">Admin</option>
          </select>
        </div>

        {/* Status */}
        <div>
          <label className="block text-sm font-bold text-gray-700 mb-2">
            Status *
          </label>
          <select
            name="status"
            value={formData.status}
            onChange={handleChange}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
          >
            <option value="Active">Active</option>
            <option value="Inactive">Inactive</option>
          </select>
        </div>
      </div>

      {/* Info Box */}
      <div className="mt-6 p-4 bg-yellow-50 border border-yellow-200 roun
ded-lg">
        <p className="text-sm text-yellow-700">
```

```
          <strong>Note:</strong> Password cannot be changed through this for
m.
          Users can reset their password through the password reset flow.
        </p>
      </div>

      {/* Buttons */}
      <div className="flex gap-4 mt-8">
        <button
          type="button"
          onClick={() => router.back()}
          className="flex-1 px-6 py-2 border border-gray-300 text-gray-700 ro
unded-lg font-bold hover:bg-gray-50 transition"
        >
          Cancel
        </button>
        <button
          type="submit"
          disabled={saving}
          className="flex-1 px-6 py-2 bg-blue-600 text-white rounded-lg font-
bold hover:bg-blue-700 transition disabled:opacity-50 flex items-center justif
y-center gap-2"
        >
          <Save className="w-5 h-5" />
          {saving ? 'Saving...' : 'Save Changes'}
        </button>
      </div>
    </form>
  </div>
 );
}
```

```
// app/dashboard/users/create/page.tsx
'use client';
import { useState } from 'react';
```

```
import { useRouter } from 'next/navigation';
import { ArrowLeft, Save, Eye, EyeOff } from 'lucide-react';

export default function CreateUserPage() {
  const router = useRouter();
  const [loading, setLoading] = useState(false);
  const [showPassword, setShowPassword] = useState(false);
  const [formData, setFormData] = useState({
    name: '',
    email: '',
    password: '',
    confirmPassword: '',
    role: 'officer',
    status: 'Active',
  });

  const handleChange = (e: React.ChangeEvent<HTMLInputElement | HTMLSe
lectElement>) => {
    const { name, value } = e.target;
    setFormData({ ...formData, [name]: value });
  };

  const handleSubmit = async (e: React.FormEvent) => {
    e.preventDefault();

    // Validate passwords match
    if (formData.password !== formData.confirmPassword) {
      alert('Passwords do not match!');
      return;
    }

    setLoading(true);

    try {
      const token = localStorage.getItem('auth_token');
      const response = await fetch('/api/users', {
```

```
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
        ...(token && { 'Authorization': `Bearer ${token}` }),
      },
      body: JSON.stringify({
        name: formData.name,
        email: formData.email,
        password: formData.password,
        role: formData.role,
        status: formData.status,
      }),
    });

    if (response.ok) {
      alert('User created successfully!');
      router.push('/dashboard/users');
    } else {
      const error = await response.json();
      alert(`Error: ${error.error || 'Failed to create user'}`);
    }
  } catch (error) {
    console.error('Error creating user:', error);
    alert('Failed to create user');
  } finally {
    setLoading(false);
  }
};

return (
  <div className="space-y-6">
    {/* Header */}
    <div className="flex items-center gap-4">
      <button
        onClick={() => router.back()}
        className="p-2 hover:bg-gray-200 rounded-lg transition"
```

```
        >
          <ArrowLeft className="w-6 h-6" />
        </button>
        <div>
          <h1 className="text-3xl font-bold text-gray-900">Create User</h1>
          <p className="text-gray-600 mt-1">Add a new staff user to the syste
m</p>
        </div>
      </div>

      {/* Form */}
      <form onSubmit={handleSubmit} className="bg-white rounded-lg borde
r border-gray-200 p-8">
        <div className="grid grid-cols-1 md:grid-cols-2 gap-6">
          {/* Name */}
          <div className="md:col-span-2">
            <label className="block text-sm font-bold text-gray-700 mb-2">
              Full Name *
            </label>
            <input
              type="text"
              name="name"
              value={formData.name}
              onChange={handleChange}
              className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
              required
              placeholder="John Doe"
            />
          </div>

          {/* Email */}
          <div className="md:col-span-2">
            <label className="block text-sm font-bold text-gray-700 mb-2">
              Email Address *
            </label>
```

```
          <input
            type="email"
            name="email"
            value={formData.email}
            onChange={handleChange}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
            placeholder="john@ndtmfi.la"
          />
        </div>

        {/* Password */}
        <div>
          <label className="block text-sm font-bold text-gray-700 mb-2">
            Password *
          </label>
          <div className="relative">
            <input
              type={showPassword ? 'text' : 'password'}
              name="password"
              value={formData.password}
              onChange={handleChange}
              className="w-full px-4 py-2 border border-gray-300 rounded-lg fo
cus:outline-none focus:ring-2 focus:ring-blue-500"
              required
              minLength={6}
              placeholder="••••••••"
            />
            <button
              type="button"
              onClick={() => setShowPassword(!showPassword)}
              className="absolute right-3 top-1/2 -translate-y-1/2 text-gray-500
hover:text-gray-700"
            >
              {showPassword ? <EyeOff className="w-5 h-5" /> : <Eye classNa
```

```
me="w-5 h-5" />}
            </button>
          </div>
          <p className="text-xs text-gray-600 mt-1">Minimum 6 characters</
p>
        </div>

        {/* Confirm Password */}
        <div>
          <label className="block text-sm font-bold text-gray-700 mb-2">
            Confirm Password *
          </label>
          <input
            type={showPassword ? 'text' : 'password'}
            name="confirmPassword"
            value={formData.confirmPassword}
            onChange={handleChange}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
            minLength={6}
            placeholder="••••••••"
          />
        </div>

        {/* Role */}
        <div>
          <label className="block text-sm font-bold text-gray-700 mb-2">
            Role *
          </label>
          <select
            name="role"
            value={formData.role}
            onChange={handleChange}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
```

```
            required
          >
            <option value="officer">Officer</option>
            <option value="manager">Manager</option>
            <option value="admin">Admin</option>
          </select>
        </div>

        {/* Status */}
        <div>
          <label className="block text-sm font-bold text-gray-700 mb-2">
            Status *
          </label>
          <select
            name="status"
            value={formData.status}
            onChange={handleChange}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg foc
us:outline-none focus:ring-2 focus:ring-blue-500"
            required
          >
            <option value="Active">Active</option>
            <option value="Inactive">Inactive</option>
          </select>
        </div>
      </div>

      {/* Info Box */}
      <div className="mt-6 p-4 bg-blue-50 border border-blue-200 rounded
-lg">
        <p className="text-sm text-blue-700">
          <strong>Note:</strong> The user will be able to log in with the provide
d email and password.
          Make sure to communicate the credentials securely.
        </p>
      </div>
```

```
        {/* Buttons */}
        <div className="flex gap-4 mt-8">
          <button
            type="button"
            onClick={() ⇒ router.back()}
            className="flex-1 px-6 py-2 border border-gray-300 text-gray-700 ro
unded-lg font-bold hover:bg-gray-50 transition"
          >
            Cancel
          </button>
          <button
            type="submit"
            disabled={loading}
            className="flex-1 px-6 py-2 bg-blue-600 text-white rounded-lg font-
bold hover:bg-blue-700 transition disabled:opacity-50 flex items-center justif
y-center gap-2"
          >
            <Save className="w-5 h-5" />
            {loading ? 'Creating...' : 'Create User'}
          </button>
        </div>
      </form>
    </div>
  );
}
```

```
// app/dashboard/users/page.tsx
'use client';

import { useState, useEffect } from 'react';
import { Plus, Edit2, Trash2, User } from 'lucide-react';
import Link from 'next/link';

interface StaffUser {
```

```
  _id: string;
  name: string;
  email: string;
  role: string;
  status: string;
}

export default function UsersPage() {
  const [users, setUsers] = useState<StaffUser[]>([]);
  const [loading, setLoading] = useState(true);
  const [deleting, setDeleting] = useState<string | null>(null);

  useEffect(() => {
    fetchUsers();
  }, []);

  const fetchUsers = async () => {
    try {
      setLoading(true);
      const response = await fetch('/api/users');
      const data = await response.json();
      setUsers(data);
    } catch (error) {
      console.error('Error fetching users:', error);
    } finally {
      setLoading(false);
    }
  };

  const handleDelete = async (id: string) => {
    if (!confirm('Are you sure you want to delete this user?')) return;

    try {
      setDeleting(id);
      const response = await fetch(`/api/users/${id}`, {
        method: 'DELETE',
```

```
      });

      if (response.ok) {
        setUsers(users.filter((u) => u._id !== id));
        alert('User deleted successfully');
      }
    } catch (error) {
      console.error('Error deleting user:', error);
      alert('Failed to delete user');
    } finally {
      setDeleting(null);
    }
  };

  return (
    <div className="space-y-6">
      <div className="flex items-center justify-between">
        <h1 className="text-3xl font-bold text-gray-900">Staff Users</h1>
        <Link
          href="/dashboard/users/create"
          className="bg-blue-600 hover:bg-blue-700 px-4 py-2 rounded-lg fon
t-medium flex items-center gap-2"
          style={{ color: '#ffffff' }}
        >
          <Plus className="w-5 h-5" style={{ color: '#ffffff' }} />
          <span style={{ color: '#ffffff' }}>Add User</span>
        </Link>
      </div>

      <div className="bg-white rounded-lg border border-gray-200 overflow-
hidden">
        <table className="w-full">
          <thead className="bg-gray-50 border-b border-gray-200">
            <tr>
              <th className="px-6 py-3 text-left text-sm font-bold text-gray-70
0">Name</th>
```

```
          <th className="px-6 py-3 text-left text-sm font-bold text-gray-70
0">Email</th>
          <th className="px-6 py-3 text-left text-sm font-bold text-gray-70
0">Role</th>
          <th className="px-6 py-3 text-left text-sm font-bold text-gray-70
0">Status</th>
          <th className="px-6 py-3 text-left text-sm font-bold text-gray-70
0">Actions</th>
         </tr>
        </thead>
        <tbody>
         {loading ? (
          <tr>
           <td colSpan={5} className="px-6 py-8 text-center text-gray-60
0">
            Loading...
           </td>
          </tr>
         ) : users.length === 0 ? (
          <tr>
           <td colSpan={5} className="px-6 py-8 text-center text-gray-60
0">
            No users found
           </td>
          </tr>
         ) : (
          users.map((user) => (
           <tr key={user._id} className="border-b border-gray-200 hover:bg-
gray-50">
            <td className="px-6 py-3 font-bold text-gray-900">{user.name}
</td>
            <td className="px-6 py-3 text-gray-700">{user.email}</td>
            <td className="px-6 py-3 text-gray-700 capitalize">{user.role}</t
d>
            <td className="px-6 py-3">
             <span className="bg-green-100 text-green-700 px-3 py-1 round
```

```
ed-full text-xs font-bold">
                {user.status}
               </span>
              </td>
              <td className="px-6 py-3">
               <div className="flex gap-2">
                <button className="p-2 text-green-600 hover:bg-green-50 rou
nded">
                 <Link
                  href={`/dashboard/users/${user._id}/edit`}
                  className="p-2 text-green-600 hover:bg-green-50 rounded"
                 >
                  <Edit2 className="w-5 h-5" />
                 </Link>
                </button>
                <button
                 onClick={() => handleDelete(user._id)}
                 disabled={deleting === user._id}
                 className="p-2 text-red-600 hover:bg-red-50 rounded disabl
ed:opacity-50"
                >
                 <Trash2 className="w-5 h-5" />
                </button>
               </div>
              </td>
             </tr>
            ))
           )}
          </tbody>
         </table>
        </div>
       </div>
  );
}
```

```
// app/dashboard/layout.tsx (UPDATED)
'use client';

import { useEffect } from 'react';
import { useRouter } from 'next/navigation';
import { useAuth } from '@/lib/auth-context';
import Sidebar from '@/components/layout/Sidebar';
import TopBar from '@/components/layout/TopBar';

export default function DashboardLayout({
  children,
}: {
  children: React.ReactNode;
}) {
  const router = useRouter();
  const { isAuthenticated, loading } = useAuth();

  useEffect(() => {
    if (!loading && !isAuthenticated) {
      router.push('/login');
    }
  }, [isAuthenticated, loading, router]);

  if (loading) {
    return (
      <div className="min-h-screen flex items-center justify-center bg-gray-5
0">
        <div className="text-center">
          <div className="inline-block animate-spin rounded-full h-12 w-12 bor
der-b-2 border-blue-600"></div>
          <p className="mt-4 text-gray-600">Loading...</p>
        </div>
      </div>
    );
  }
```

```
  if (!isAuthenticated) {
    return null;
  }

  return (
    <div className="flex bg-gray-50">
      {/* Sidebar */}
      <Sidebar />

      {/* Main Content */}
      <div className="flex-1 md:ml-64">
        {/* Top Bar */}
        <TopBar />

        {/* Page Content */}
        <main className="mt-14 md:mt-0 p-4 md:p-8">
          <div className="max-w-7xl mx-auto">
            {children}
          </div>
        </main>
      </div>
    </div>
  );
}
```

```
// app/dashboard/page.tsx
'use client';

import { useState, useEffect } from 'react';
import { BarChart, Bar, LineChart, Line, XAxis, YAxis, CartesianGrid, Tooltip, L
egend, ResponsiveContainer } from 'recharts';
import { TrendingUp, Users, FileText, DollarSign, AlertCircle } from 'lucide-rea
ct';
import Link from 'next/link';
```

```
const dashboardData = [
  { month: 'Jan', applications: 40, approved: 24, rejected: 8 },
  { month: 'Feb', applications: 52, approved: 35, rejected: 10 },
  { month: 'Mar', applications: 48, approved: 28, rejected: 12 },
  { month: 'Apr', applications: 61, approved: 42, rejected: 15 },
  { month: 'May', applications: 55, approved: 38, rejected: 12 },
  { month: 'Jun', applications: 67, approved: 45, rejected: 18 },
];

interface Stats {
  totalApplications: number;
  approvedApplications: number;
  pendingApplications: number;
  rejectedApplications: number;
  totalDisbursed: number;
  averageLoan: number;
  approvalRate: number;
}

interface Application {
  _id: string;
  fullName: string;
  loanType: string;
  loanAmount: number;
  status: string;
  createdAt?: string;
}

export default function DashboardPage() {
  const [stats, setStats] = useState<Stats | null>(null);
  const [recentApplications, setRecentApplications] = useState<Application[]>
([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState('');
```

```
useEffect(() => {
  fetchDashboardData();
}, []);

const fetchDashboardData = async () => {
  try {
    setLoading(true);
    setError('');

    // Get token from localStorage
    const token = localStorage.getItem('auth_token');

    // Fetch stats
    const statsResponse = await fetch('/api/stats', {
      headers: token ? { 'Authorization': `Bearer ${token}` } : {},
    });

    if (statsResponse.ok) {
      const statsData = await statsResponse.json();
      setStats(statsData);
    } else {
      console.warn('Stats API failed:', await statsResponse.text());
      // Set default stats if API fails
      setStats({
        totalApplications: 0,
        approvedApplications: 0,
        pendingApplications: 0,
        rejectedApplications: 0,
        totalDisbursed: 0,
        averageLoan: 0,
        approvalRate: 0,
      });
    }

    // Fetch recent applications
    const appsResponse = await fetch('/api/applications', {
```

```
      headers: token ? { 'Authorization': `Bearer ${token}` } : {},
    });

    if (appsResponse.ok) {
      const appsData = await appsResponse.json();
      // Ensure it's an array
      if (Array.isArray(appsData)) {
        setRecentApplications(appsData.slice(0, 5)); // Take first 5
      } else {
        console.warn('Applications API returned non-array:', appsData);
        setRecentApplications([]);
      }
    } else {
      console.warn('Applications API failed:', await appsResponse.text());
      setRecentApplications([]);
    }
  } catch (error) {
    console.error('Error fetching dashboard data:', error);
    setError('Failed to load dashboard data');
    // Set defaults
    setStats({
      totalApplications: 0,
      approvedApplications: 0,
      pendingApplications: 0,
      rejectedApplications: 0,
      totalDisbursed: 0,
      averageLoan: 0,
      approvalRate: 0,
    });
    setRecentApplications([]);
  } finally {
    setLoading(false);
  }
};

if (loading) {
```

```
    return (
      <div className="flex items-center justify-center h-96">
        <div className="text-center">
          <div className="inline-block animate-spin rounded-full h-12 w-12 bor
der-b-2 border-blue-600"></div>
          <p className="mt-4 text-gray-600">Loading dashboard...</p>
        </div>
      </div>
    );
  }

  if (!stats) {
    return (
      <div className="flex items-center justify-center h-96">
        <div className="text-center">
          <AlertCircle className="w-12 h-12 text-red-600 mx-auto mb-4" />
          <p className="text-gray-600">{error || 'Failed to load dashboard'}</p
>
          <button
            onClick={fetchDashboardData}
            className="mt-4 bg-blue-600 text-white px-6 py-2 rounded-lg hove
r:bg-blue-700"
          >
            Retry
          </button>
        </div>
      </div>
    );
  }

  const statCards = [
    {
      icon: <FileText className="w-8 h-8" />,
      label: 'Total Applications',
      value: stats.totalApplications.toString(),
      change: '+12% from last month',
```

```
      bgColor: 'bg-blue-50',
      iconColor: 'text-blue-600',
    },
    {
      icon: <DollarSign className="w-8 h-8" />,
      label: 'Funds Disbursed',
      value: stats.totalDisbursed > 0 ? `${(stats.totalDisbursed / 1000000).toFixe
d(2)}M` : '0',
      change: '+8.5% from last month',
      bgColor: 'bg-green-50',
      iconColor: 'text-green-600',
    },
    {
      icon: <Users className="w-8 h-8" />,
      label: 'Approved Loans',
      value: stats.approvedApplications.toString(),
      change: '+5.2% from last month',
      bgColor: 'bg-purple-50',
      iconColor: 'text-purple-600',
    },
    {
      icon: <TrendingUp className="w-8 h-8" />,
      label: 'Approval Rate',
      value: `${stats.approvalRate}%`,
      change: '+2.1% from last month',
      bgColor: 'bg-orange-50',
      iconColor: 'text-orange-600',
    },
  ];

  return (
    <div className="space-y-8">
      {/* Page Header */}
      <div>
        <h1 className="text-3xl font-bold text-gray-900">Dashboard</h1>
        <p className="text-gray-600 mt-2">Welcome to NDTMFI Admin Panel
```

```
</p>
    </div>

    {/* Stats Cards */}
    <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-6">
     {statCards.map((stat, idx) => (
       <div
         key={idx}
         className={`${stat.bgColor} p-6 rounded-lg border border-gray-200
hover:shadow-lg transition`}
       >
         <div className="flex items-start justify-between">
           <div>
             <p className="text-gray-600 text-sm font-bold">{stat.label}</p>
             <p className="text-3xl font-bold text-gray-900 mt-2">{stat.value}
</p>
             <p className="text-xs text-green-600 mt-2 flex items-center gap-
1">
               <TrendingUp className="w-3 h-3" /> {stat.change}
             </p>
           </div>
           <div className={`${stat.iconColor}`}>{stat.icon}</div>
         </div>
       </div>
     ))}
    </div>

    {/* Charts */}
    <div className="grid grid-cols-1 lg:grid-cols-2 gap-6">
     {/* Line Chart */}
     <div className="bg-white p-6 rounded-lg border border-gray-200">
       <h2 className="text-lg font-bold text-gray-900 mb-4">Application Tr
ends</h2>
       <ResponsiveContainer width="100%" height={300}>
         <LineChart data={dashboardData}>
           <CartesianGrid strokeDasharray="3 3" />
```

```
            <XAxis dataKey="month" />
            <YAxis />
            <Tooltip />
            <Legend />
            <Line type="monotone" dataKey="applications" stroke="#2563eb" st
rokeWidth={2} />
            <Line type="monotone" dataKey="approved" stroke="#16a34a" strok
eWidth={2} />
          </LineChart>
        </ResponsiveContainer>
      </div>

      {/* Bar Chart */}
      <div className="bg-white p-6 rounded-lg border border-gray-200">
        <h2 className="text-lg font-bold text-gray-900 mb-4">Monthly Comp
arison</h2>
        <ResponsiveContainer width="100%" height={300}>
          <BarChart data={dashboardData}>
            <CartesianGrid strokeDasharray="3 3" />
            <XAxis dataKey="month" />
            <YAxis />
            <Tooltip />
            <Legend />
            <Bar dataKey="approved" fill="#2563eb" />
            <Bar dataKey="rejected" fill="#ef4444" />
          </BarChart>
        </ResponsiveContainer>
      </div>
    </div>

    {/* Alerts & Recent Applications */}
    <div className="grid grid-cols-1 lg:grid-cols-3 gap-6">
      {/* Alerts */}
      <div className="bg-white p-6 rounded-lg border border-gray-200">
        <h2 className="text-lg font-bold text-gray-900 mb-4">System Alerts
</h2>
```

```
        <div className="space-y-3">
          <div className="flex items-start gap-3 p-3 bg-yellow-50 rounded-lg
border border-yellow-200">
            <AlertCircle className="w-5 h-5 text-yellow-600 flex-shrink-0 mt-
0.5" />
            <div>
              <p className="text-sm font-bold text-yellow-900">Pending Revie
ws</p>
              <p className="text-xs text-yellow-700">{stats.pendingApplication
s} applications awaiting review</p>
            </div>
          </div>
          <div className="flex items-start gap-3 p-3 bg-blue-50 rounded-lg b
order border-blue-200">
            <AlertCircle className="w-5 h-5 text-blue-600 flex-shrink-0 mt-0.
5" />
            <div>
              <p className="text-sm font-bold text-blue-900">Approved Today
</p>
              <p className="text-xs text-blue-700">{stats.approvedApplication
s} loans processed successfully</p>
            </div>
          </div>
          <div className="flex items-start gap-3 p-3 bg-green-50 rounded-lg
border border-green-200">
            <AlertCircle className="w-5 h-5 text-green-600 flex-shrink-0 mt-0.
5" />
            <div>
              <p className="text-sm font-bold text-green-900">System Status
</p>
              <p className="text-xs text-green-700">All systems operational</p
>
            </div>
          </div>
        </div>
      </div>
```

```
      {/* Recent Applications */}
      <div className="lg:col-span-2 bg-white p-6 rounded-lg border border-gray-200">
        <div className="flex items-center justify-between mb-4">
          <h2 className="text-lg font-bold text-gray-900">Recent Applications</h2>
          <Link href="/dashboard/loan-applications" className="text-blue-600 hover:text-blue-700 text-sm font-bold">
            View All →
          </Link>
        </div>
        <div className="overflow-x-auto">
          <table className="w-full text-sm">
            <thead className="bg-gray-50 border-b border-gray-200">
              <tr>
                <th className="px-4 py-3 text-left font-bold text-gray-700">Name</th>
                <th className="px-4 py-3 text-left font-bold text-gray-700">Product</th>
                <th className="px-4 py-3 text-left font-bold text-gray-700">Amount</th>
                <th className="px-4 py-3 text-left font-bold text-gray-700">Status</th>
              </tr>
            </thead>
            <tbody>
              {recentApplications.length === 0 ? (
                <tr>
                  <td colSpan={4} className="px-4 py-8 text-center text-gray-600">
                    No applications yet. <Link href="/dashboard/loan-applications" className="text-blue-600 hover:underline">Add your first application</Link>
                  </td>
                </tr>
              ) : (
```

```
            recentApplications.map((app) => (
              <tr key={app._id} className="border-b border-gray-200 hover:b
g-gray-50">
                <td className="px-4 py-3">{app.fullName}</td>
                <td className="px-4 py-3">{app.loanType}</td>
                <td className="px-4 py-3">{app.loanAmount.toLocaleString()}
</td>
                <td className="px-4 py-3">
                  <span
                    className={`px-2 py-1 rounded-full text-xs font-bold ${
                      app.status === 'Approved'
                        ? 'bg-green-100 text-green-700'
                        : app.status === 'Pending'
                        ? 'bg-yellow-100 text-yellow-700'
                        : app.status === 'Under Review'
                        ? 'bg-blue-100 text-blue-700'
                        : 'bg-red-100 text-red-700'
                    }`}
                  >
                    {app.status}
                  </span>
                </td>
              </tr>
            ))
          )}
        </tbody>
      </table>
    </div>
  </div>
 </div>
 );
}
```

```
// app/login/page.tsx
'use client';

import { useState, useEffect } from 'react';
import { useRouter } from 'next/navigation';
import { useAuth } from '@/lib/auth-context';
import { LogIn, Eye, EyeOff, AlertCircle } from 'lucide-react';

export default function LoginPage() {
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const [showPassword, setShowPassword] = useState(false);
  const [error, setError] = useState('');
  const [loading, setLoading] = useState(false);
  const [shouldRedirect, setShouldRedirect] = useState(false);
  const router = useRouter();
  const { login, isAuthenticated } = useAuth();

  // Handle redirect after authentication
  useEffect(() => {
    if (shouldRedirect || isAuthenticated) {
      router.push('/dashboard');
    }
  }, [shouldRedirect, isAuthenticated, router]);

  const handleLogin = async (e: React.FormEvent) => {
    e.preventDefault();
    setError('');
    setLoading(true);

    try {
      await login(email, password);
      setShouldRedirect(true);
    } catch (err: any) {
      setError(err.message || 'Login failed');
```

```
      setLoading(false);
    }
  };

  // Show loading while redirecting
  if (shouldRedirect || isAuthenticated) {
    return (
      <div className="min-h-screen bg-gradient-to-br from-blue-600 to-blue-
800 flex items-center justify-center p-6">
        <div className="text-center">
          <div className="inline-block animate-spin rounded-full h-12 w-12 bor
der-b-2 border-white"></div>
          <p className="mt-4 text-white">Redirecting to dashboard...</p>
        </div>
      </div>
    );
  }

  return (
    <div className="min-h-screen bg-gradient-to-br from-blue-600 to-blue-
800 flex items-center justify-center p-6">
      <div className="w-full max-w-md">
        {/* Card */}
        <div className="bg-white rounded-lg shadow-2xl p-8">
          {/* Logo/Title */}
          <div className="text-center mb-8">
            <div className="inline-flex items-center justify-center w-16 h-16 bg-
blue-100 rounded-full mb-4">
              <LogIn className="w-8 h-8 text-blue-600" />
            </div>
            <h1 className="text-3xl font-bold text-gray-900">NDTMFI Admin</h
1>
            <p className="text-gray-600 mt-2">Microfinance Management Syst
em</p>
          </div>
```

```
        {/* Error Message */}
        {error && (
          <div className="mb-6 p-4 bg-red-50 border border-red-200 rounde
d-lg flex items-start gap-3">
            <AlertCircle className="w-5 h-5 text-red-600 flex-shrink-0 mt-0.5"
/>
            <p className="text-sm text-red-700">{error}</p>
          </div>
        )}

        {/* Demo Credentials */}
        <div className="mb-6 p-4 bg-blue-50 border border-blue-200 rounde
d-lg text-sm">
          <p className="text-gray-700"><strong>Demo Account:</strong></p
>
          <p className="text-gray-600">Email: admin@ndtmfi.la</p>
          <p className="text-gray-600">Password: admin123</p>
        </div>

        {/* Form */}
        <form onSubmit={handleLogin} className="space-y-4">
          {/* Email */}
          <div>
            <label className="block text-sm font-bold text-gray-700 mb-2">Em
ail Address</label>
            <input
              type="email"
              value={email}
              onChange={(e) => setEmail(e.target.value)}
              className="w-full px-4 py-2 border border-gray-300 rounded-lg fo
cus:outline-none focus:ring-2 focus:ring-blue-500 focus:border-transparent"
              placeholder="admin@ndtmfi.la"
              required
              disabled={loading}
            />
          </div>
```

```
          {/* Password */}
          <div>
            <label className="block text-sm font-bold text-gray-700 mb-2">Pa
ssword</label>
            <div className="relative">
              <input
                type={showPassword ? 'text' : 'password'}
                value={password}
                onChange={(e) => setPassword(e.target.value)}
                className="w-full px-4 py-2 border border-gray-300 rounded-lg f
ocus:outline-none focus:ring-2 focus:ring-blue-500 focus:border-transparen
t"
                placeholder="••••••••"
                required
                disabled={loading}
              />
              <button
                type="button"
                onClick={() => setShowPassword(!showPassword)}
                className="absolute right-3 top-1/2 -translate-y-1/2 text-gray-50
0 hover:text-gray-700 disabled:opacity-50"
                disabled={loading}
              >
                {showPassword ? (
                  <EyeOff className="w-5 h-5" />
                ) : (
                  <Eye className="w-5 h-5" />
                )}
              </button>
            </div>
          </div>

          {/* Remember Me */}
          <div className="flex items-center">
            <input
```

```
              type="checkbox"
              id="remember"
              className="rounded border-gray-300"
              disabled={loading}
            />
            <label htmlFor="remember" className="ml-2 text-sm text-gray-60
0">
              Remember me
            </label>
          </div>

          {/* Submit Button */}
          <button
            type="submit"
            disabled={loading}
            className="w-full bg-blue-600 text-white py-2 rounded-lg font-bold
hover:bg-blue-700 transition disabled:opacity-50 disabled:cursor-not-allowed
flex items-center justify-center gap-2"
          >
            <LogIn className="w-5 h-5" />
            {loading ? 'Signing in...' : 'Sign In'}
          </button>
        </form>

        {/* Footer */}
        <div className="mt-8 pt-6 border-t border-gray-200 text-center text-s
m text-gray-600">
          <p>© 2024 NDTMFI. All rights reserved.</p>
          <p className="mt-2 text-xs">Contact: support@ndtmfi.la</p>
        </div>
      </div>
    </div>
  </div>
 );
}
```

```
// app/layout.tsx (UPDATED)
import type { Metadata } from 'next';
import '@/styles/globals.css';
import { AuthProvider } from '@/lib/auth-context';

export const metadata: Metadata = {
  title: 'NDTMFI Admin Dashboard',
  description: 'Admin panel for managing NDTMFI microfinance institution',
};

export default function RootLayout({
  children,
}: {
  children: React.ReactNode;
}) {
  return (
    <html lang="en">
      <head>
        <meta charSet="utf-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1" /
>
      </head>
      <body className="bg-gray-50 text-gray-900 antialiased">
        <AuthProvider>
          {children}
        </AuthProvider>
      </body>
    </html>
  );
}
```

```
// app/page.tsx
'use client';
```

```
import { useEffect } from 'react';
import { useRouter } from 'next/navigation';

export default function RootPage() {
  const router = useRouter();

  useEffect(() => {
    // Check if user is authenticated
    const token = localStorage.getItem('auth_token');

    if (token) {
      // Redirect to dashboard if logged in
      router.push('/dashboard');
    } else {
      // Redirect to login if not authenticated
      router.push('/login');
    }
  }, [router]);

  return (
    <div className="min-h-screen flex items-center justify-center bg-gray-5
0">
      <div className="text-center">
        <div className="inline-block animate-spin rounded-full h-12 w-12 bord
er-b-2 border-blue-600"></div>
        <p className="mt-4 text-gray-600">Loading...</p>
      </div>
    </div>
  );
}
```

```
// lib/auth-context.tsx
'use client';

import React, { createContext, useContext, useState, useEffect } from 'react';
```

```
interface User {
  _id: string;
  email: string;
  name: string;
  role: string;
  status: string;
}

interface AuthContextType {
  user: User | null;
  token: string | null;
  loading: boolean;
  login: (email: string, password: string) => Promise<void>;
  logout: () => void;
  isAuthenticated: boolean;
}

const AuthContext = createContext<AuthContextType | undefined>(undefine
d);

export function AuthProvider({ children }: { children: React.ReactNode }) {
  const [user, setUser] = useState<User | null>(null);
  const [token, setToken] = useState<string | null>(null);
  const [loading, setLoading] = useState(true);

  // Check if user is already logged in (on mount)
  useEffect(() => {
    const storedToken = localStorage.getItem('auth_token');
    const storedUser = localStorage.getItem('auth_user');

    if (storedToken && storedUser) {
      setToken(storedToken);
      setUser(JSON.parse(storedUser));
    }
```

```
  setLoading(false);
}, []);

const login = async (email: string, password: string) => {
  try {
    setLoading(true);
    const response = await fetch('/api/auth/login', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({ email, password }),
    });

    if (!response.ok) {
      const error = await response.json();
      throw new Error(error.error || 'Login failed');
    }

    const data = await response.json();

    // Store token and user
    localStorage.setItem('auth_token', data.token);
    localStorage.setItem('auth_user', JSON.stringify(data.user));

    setToken(data.token);
    setUser(data.user);
  } catch (error) {
    throw error;
  } finally {
    setLoading(false);
  }
};

const logout = () => {
  localStorage.removeItem('auth_token');
  localStorage.removeItem('auth_user');
  setToken(null);
```

```
    setUser(null);
  };

  return (
    <AuthContext.Provider
      value={{
        user,
        token,
        loading,
        login,
        logout,
        isAuthenticated: !!token,
      }}
    >
      {children}
    </AuthContext.Provider>
  );
}

export function useAuth() {
  const context = useContext(AuthContext);
  if (!context) {
    throw new Error('useAuth must be used within AuthProvider');
  }
  return context;
}
```

```
// lib/auth.ts
import jwt from 'jsonwebtoken';
import bcrypt from 'bcryptjs';

const JWT_SECRET = process.env.JWT_SECRET || 'your-secret-key-change-i
n-production';

export interface TokenPayload {
```

```
  userId: string;
  email: string;
  role: string;
  iat?: number;
  exp?: number;
}

// Hash password
export async function hashPassword(password: string): Promise<string> {
  try {
    const salt = await bcrypt.genSalt(10);
    return await bcrypt.hash(password, salt);
  } catch (error) {
    throw new Error('Error hashing password');
  }
}

// Compare password
export async function comparePassword(password: string, hash: string): Pro
mise<boolean> {
  try {
    return await bcrypt.compare(password, hash);
  } catch (error) {
    throw new Error('Error comparing password');
  }
}

// Create JWT token
export function createToken(payload: TokenPayload): string {
  return jwt.sign(payload, JWT_SECRET, {
    expiresIn: '7d',
  });
}

// Verify JWT token
export function verifyToken(token: string): TokenPayload | null {
```

```
  try {
    const decoded = jwt.verify(token, JWT_SECRET) as TokenPayload;
    return decoded;
  } catch (error) {
    return null;
  }
}

// Extract token from request
export function extractToken(authHeader?: string): string | null {
  if (!authHeader || !authHeader.startsWith('Bearer ')) {
    return null;
  }
  return authHeader.substring(7);
}
```

```
// lib/mongodb.ts
import mongoose from 'mongoose';

if (!process.env.MONGODB_URI) {
  throw new Error('Invalid/Missing environment variable: "MONGODB_URI"');
}

const MONGODB_URI: string = process.env.MONGODB_URI;

interface Cached {
  conn: typeof mongoose | null;
  promise: Promise<typeof mongoose> | null;
}

let cached: Cached = (global as any).mongoose || { conn: null, promise: null };

if (!cached) {
  cached = (global as any).mongoose = { conn: null, promise: null };
}
```

```
async function dbConnect(): Promise<typeof mongoose> {
  if (cached.conn) {
    return cached.conn;
  }

  if (!cached.promise) {
    const opts = {
      bufferCommands: false,
    };

    cached.promise = mongoose.connect(MONGODB_URI, opts).then((mongoo
se) => {
      return mongoose;
    });
  }

  try {
    cached.conn = await cached.promise;
  } catch (e) {
    cached.promise = null;
    throw e;
  }

  return cached.conn;
}

export default dbConnect;
```

```
// models/Branch.ts
import mongoose, { Schema, Document } from 'mongoose';

export interface IBranch extends Document {
  name: string;
  city: string;
```

```
  address: string;
  phone: string;
  email: string;
  hours: string;
  status: 'Active' | 'Inactive';
  createdAt: Date;
  updatedAt: Date;
}

const branchSchema = new Schema(
  {
    name: { type: String, required: true },
    city: { type: String, required: true },
    address: { type: String, required: true },
    phone: { type: String, required: true },
    email: { type: String, required: true },
    hours: { type: String, required: true },
    status: {
      type: String,
      enum: ['Active', 'Inactive'],
      default: 'Active',
    },
  },
  { timestamps: true }
);

export const Branch =
  mongoose.models.Branch ||
  mongoose.model<IBranch>('Branch', branchSchema);
```

```
// models/FAQ.ts
import mongoose, { Schema, Document } from 'mongoose';

export interface IFAQ extends Document {
  question: string;
```

```
  answer: string;
  category: string;
  views: number;
  status: 'Active' | 'Inactive';
  createdAt: Date;
  updatedAt: Date;
}

const faqSchema = new Schema(
  {
    question: { type: String, required: true },
    answer: { type: String, required: true },
    category: { type: String, required: true },
    views: { type: Number, default: 0 },
    status: {
      type: String,
      enum: ['Active', 'Inactive'],
      default: 'Active',
    },
  },
  { timestamps: true }
);

export const FAQ =
  mongoose.models.FAQ ||
  mongoose.model<IFAQ>('FAQ', faqSchema);
```

```
// models/LoanApplication.ts
import mongoose, { Schema, Document } from 'mongoose';

export interface ILoanApplication extends Document {
  fullName: string;
  email: string;
  phone: string;
  dob: Date;
```

```
  gender: string;
  idNumber: string;
  address: string;
  loanType: string;
  loanAmount: number;
  purpose: string;
  tenure: number;
  employment: string;
  income: number;
  businessName?: string;
  yearsInBusiness?: number;
  employees?: number;
  status: 'Pending' | 'Under Review' | 'Approved' | 'Rejected';
  documents: string[];
  notes?: string;
  createdAt: Date;
  updatedAt: Date;
}

const loanApplicationSchema = new Schema(
  {
    fullName: { type: String, required: true },
    email: { type: String, required: true, unique: true },
    phone: { type: String, required: true },
    dob: { type: Date, required: true },
    gender: { type: String, required: true },
    idNumber: { type: String, required: true, unique: true },
    address: { type: String, required: true },
    loanType: { type: String, required: true },
    loanAmount: { type: Number, required: true },
    purpose: { type: String, required: true },
    tenure: { type: Number, required: true },
    employment: { type: String, required: true },
    income: { type: Number, required: true },
    businessName: String,
    yearsInBusiness: Number,
```

```
    employees: Number,
    status: {
      type: String,
      enum: ['Pending', 'Under Review', 'Approved', 'Rejected'],
      default: 'Pending',
    },
    documents: [String],
    notes: String,
  },
  { timestamps: true }
);

export const LoanApplication =
  mongoose.models.LoanApplication ||
  mongoose.model<ILoanApplication>('LoanApplication', loanApplicationSche
ma);
```

```
// models/LoanProduct.ts
import mongoose, { Schema, Document } from 'mongoose';

export interface ILoanProduct extends Document {
  name: string;
  description: string;
  minAmount: number;
  maxAmount: number;
  minTenure: number;
  maxTenure: number;
  minInterest: number;
  maxInterest: number;
  processingFee: number;
  status: 'Active' | 'Inactive';
  features: string[];
  eligibility: string[];
  createdAt: Date;
  updatedAt: Date;
```

```
}

const loanProductSchema = new Schema(
  {
    name: { type: String, required: true, unique: true },
    description: { type: String, required: true },
    minAmount: { type: Number, required: true },
    maxAmount: { type: Number, required: true },
    minTenure: { type: Number, required: true },
    maxTenure: { type: Number, required: true },
    minInterest: { type: Number, required: true },
    maxInterest: { type: Number, required: true },
    processingFee: { type: Number, required: true },
    status: {
      type: String,
      enum: ['Active', 'Inactive'],
      default: 'Active',
    },
    features: [String],
    eligibility: [String],
  },
  { timestamps: true }
);

export const LoanProduct =
  mongoose.models.LoanProduct ||
  mongoose.model<ILoanProduct>('LoanProduct', loanProductSchema);
```

```
// models/User.ts
import mongoose, { Schema, Document } from 'mongoose';

export interface IUser extends Document {
  email: string;
  password: string;
  name: string;
```

```
  role: 'admin' | 'officer' | 'manager';
  status: 'Active' | 'Inactive';
  createdAt: Date;
  updatedAt: Date;
}

const userSchema = new Schema(
  {
    email: { type: String, required: true, unique: true },
    password: { type: String, required: true },
    name: { type: String, required: true },
    role: {
      type: String,
      enum: ['admin', 'officer', 'manager'],
      default: 'officer',
    },
    status: {
      type: String,
      enum: ['Active', 'Inactive'],
      default: 'Active',
    },
  },
  { timestamps: true }
);

export const User =
  mongoose.models.User ||
  mongoose.model<IUser>('User', userSchema);
```