🚀 MERN Template

A comprehensive, production-ready MERN stack template with authentication, file uploads, email services, and extensive debugging tools.

✨ Features

🔐 Authentication & Security

- **JWT Authentication** with refresh tokens
- **OAuth Integration** (Google, GitHub)
- **Email Verification** and password reset
- **Role-based Access Control** (User, Admin, Moderator)
- **Rate Limiting** and security headers
- **Input Validation** and sanitization

📁 File Management

- **File Upload** with Multer
- **Cloud Storage** support (Cloudinary)
- **Image Processing** and thumbnail generation
- **File Type Validation** and size limits
- **Progress Tracking** for uploads

📧 Email Services

- **Email Templates** with Handlebars
- **Email Queue** with job tracking
- **Multiple Providers** (SMTP, SendGrid, Mailgun)
- **Email Verification** and notifications

🛠️ Developer Experience

- **TypeScript** throughout the stack
- **Hot Reloading** for development
- **Comprehensive Debugging** dashboard
- **API Documentation** and testing tools
- **Database Seeding** and migrations

- **Testing Setup** (Jest, Cypress)

🎨 Modern UI

- **Chakra UI** component library
- **Dark/Light Mode** support
- **Responsive Design** for all devices
- **Accessibility** compliant
- **Smooth Animations** with Framer Motion

🔍 Debugging & Monitoring

- **Debug Dashboard** for system monitoring
- **API Testing** interface
- **Database Query** builder
- **Live Logs** viewer
- **Performance Metrics** tracking
- **WebSocket** connection testing

🏗️ Tech Stack

Frontend

- **React 18** with TypeScript
- **Vite** for fast development
- **Chakra UI** for components
- **Zustand** for state management
- **React Query** for server state
- **React Router** for navigation
- **Framer Motion** for animations

Backend

- **Node.js** with Express
- **TypeScript** for type safety
- **MongoDB** with Prisma ORM
- **JWT** authentication

- **Passport.js** for OAuth
- **Nodemailer** for emails
- **Socket.io** for real-time features

DevOps & Tools

- **Docker** for containerization
- **Railway** deployment ready
- **ESLint & Prettier** for code quality
- **Jest & Cypress** for testing
- **Winston** for logging
- **Redis** for caching

🚀 Quick Start

Prerequisites

- **Node.js** 18+
- **npm** 8+
- **MongoDB** (local or cloud)
- **Git**

1. Clone the Repository

```bash
git clone https://github.com/yourusername/mern-template.git
cd mern-template
```

2. Install Dependencies

```bash
npm run setup
```

This will install dependencies for the root, client, server, and shared packages.

3. Environment Configuration

Copy the environment template and configure your settings:

```bash
cp .env.example .env
```

Update the `.env` file with your configuration:

```bash
# Database
DATABASE_URL=mongodb://localhost:27017/mern_template

# JWT Secrets (generate strong secrets for production)
JWT_SECRET=your-super-secret-jwt-key-change-in-production
JWT_REFRESH_SECRET=your-super-secret-refresh-key-change-in-production

# OAuth (optional - get from Google/GitHub)
GOOGLE_CLIENT_ID=your-google-client-id
GOOGLE_CLIENT_SECRET=your-google-client-secret

# Email (configure one provider)
EMAIL_HOST=smtp.gmail.com
EMAIL_PORT=587
EMAIL_USER=your-email@gmail.com
EMAIL_PASS=your-app-password

# File Upload (optional - for cloud storage)
CLOUDINARY_CLOUD_NAME=your-cloudinary-cloud-name
CLOUDINARY_API_KEY=your-cloudinary-api-key
CLOUDINARY_API_SECRET=your-cloudinary-api-secret
```

4. Database Setup

```bash
# Generate Prisma client
npm run db:generate

# Run migrations
npm run db:migrate

# Seed the database with sample data
npm run db:seed
```

5. Start Development

```bash
npm run dev
```

This starts:

- **Frontend**: http://localhost:3000
- **Backend**: http://localhost:5000
- **Debug Dashboard**: http://localhost:8080

📋 Default Accounts

After seeding, you can log in with these accounts:

Role	Email	Password
Admin	admin@example.com	Admin123!
Moderator	moderator@example.com	Moderator123!
User	user1@example.com	User123!

🐳 Docker Development

For a containerized development environment:

```bash
# Start all services
npm run docker:up

# View logs
npm run docker:logs

# Stop services
npm run docker:down
```

🧪 Testing

```bash
# Run all tests
npm run test

# Run specific test suites
npm run test:client    # Frontend tests
npm run test:server    # Backend tests
npm run test:e2e       # End-to-end tests

# Coverage report
npm run test:coverage
```

📦 Production Build

```bash
# Build all packages
npm run build

# Start production server
npm start
```

🚀 Deployment

Railway Deployment

1. **Connect Repository**: Link your GitHub repository to Railway
2. **Environment Variables**: Add all required environment variables
3. **Database**: Add MongoDB service or connect external database
4. **Deploy**: Railway will automatically build and deploy

Manual Deployment

1. **Build the application**:

   ```bash
   npm run build
   ```

2. **Set environment variables** for production
3. **Start the server**:

```bash
npm start
```

🔧 Configuration

OAuth Setup

Google OAuth

1. Go to Google Cloud Console
2. Create a new project or select existing
3. Enable Google+ API
4. Create OAuth 2.0 credentials
5. Add authorized redirect URIs:
   - `http://localhost:5000/api/auth/google/callback` (development)
   - `https://yourdomain.com/api/auth/google/callback` (production)

GitHub OAuth

1. Go to GitHub Settings > Developer settings > OAuth Apps
2. Create a new OAuth App
3. Set Authorization callback URL:
   - `http://localhost:5000/api/auth/github/callback` (development)
   - `https://yourdomain.com/api/auth/github/callback` (production)

Email Configuration

Gmail SMTP

1. Enable 2-factor authentication
2. Generate an App Password
3. Use these settings:

```bash
EMAIL_HOST=smtp.gmail.com
EMAIL_PORT=587
EMAIL_USER=your-email@gmail.com
EMAIL_PASS=your-app-password
```

SendGrid

```bash
SENDGRID_API_KEY=your-sendgrid-api-key
EMAIL_FROM=noreply@yourdomain.com
```

File Upload Configuration

Cloudinary (Recommended)

1. Sign up at Cloudinary
2. Get your cloud name, API key, and secret
3. Configure environment variables

Local Storage

Files are stored in the `uploads/` directory by default.

🛠️ Development Guide

Project Structure

```
mern-template/
├── client/                 # React frontend
│   ├── src/
│   │   ├── components/    # Reusable components
│   │   ├── pages/         # Page components
│   │   ├── hooks/         # Custom hooks
│   │   ├── stores/        # Zustand stores
│   │   ├── services/      # API services
│   │   ├── utils/         # Utility functions
│   │   └── theme/         # Chakra UI theme
│   └── package.json
├── server/                 # Express backend
│   ├── src/
│   │   ├── controllers/   # Route controllers
│   │   ├── middleware/    # Custom middleware
│   │   ├── routes/        # API routes
│   │   ├── services/      # Business logic
│   │   ├── utils/         # Utility functions
│   │   ├── config/        # Configuration
│   │   └── validation/    # Input validation
│   ├── prisma/           # Database schema & migrations
│   └── package.json
├── shared/               # Shared types & utilities
├── debug-dashboard/      # Debugging interface
├── docker-compose.yml    # Docker configuration
└── package.json         # Root package.json
```

Adding New Features

1. **Update shared types** in `/shared/src/types/`
2. **Add database models** in `/server/prisma/schema.prisma`
3. **Create API routes** in `/server/src/routes/`
4. **Add frontend components** in `/client/src/components/`
5. **Update validation** in `/server/src/validation/`
6. **Add tests** for new functionality

API Routes

Method	Endpoint	Description
POST	`/api/auth/login`	User login
POST	`/api/auth/register`	User registration
GET	`/api/auth/me`	Get current user
POST	`/api/auth/logout`	User logout
GET	`/api/users/profile`	Get user profile
PUT	`/api/users/profile`	Update profile
POST	`/api/files/upload`	Upload file
GET	`/api/files`	List user files
DELETE	`/api/files/:id`	Delete file
GET	`/api/debug/health`	Health check

Database Schema

Key models include:

- **User**: User accounts with authentication
- **UserProfile**: Extended user information
- **File**: File upload records
- **EmailJob**: Email queue and tracking
- **Notification**: User notifications
- **AuditLog**: System activity logs

🐛 Debugging

Debug Dashboard

Access the comprehensive debugging dashboard at `http://localhost:8080`:

- **System Overview**: Server stats, user metrics, performance
- **API Tester**: Test all endpoints with custom payloads
- **Database Explorer**: Query and inspect database collections
- **File Manager**: View and test file uploads
- **Email Tester**: Send test emails and view templates
- **Live Logs**: Real-time log monitoring
- **WebSocket Tester**: Test real-time connections

Common Issues

Database Connection

```bash
# Check MongoDB is running
brew services start mongodb-community@7.0  # macOS
sudo systemctl start mongod                # Linux

# Verify connection
npm run db:studio
```

Environment Variables

```bash
# Verify all required variables are set
node -e "console.log(process.env.DATABASE_URL)"
```

Port Conflicts

```bash
# Kill processes on ports 3000, 5000, 8080
sudo lsof -ti:3000,5000,8080 | xargs kill -9
```

🧪 Testing Guide

Frontend Testing

- **Component Tests**: React Testing Library
- **Hook Tests**: Custom hook testing
- **Integration Tests**: API integration
- **E2E Tests**: Cypress for user workflows

Backend Testing

- **Unit Tests**: Service and utility functions
- **Integration Tests**: API endpoints
- **Database Tests**: Model operations
- **Authentication Tests**: Auth flows

Running Tests

```bash
# Watch mode for development
npm run test:watch

# Specific test files
npm run test -- --testNamePattern="auth"

# Generate coverage
npm run test:coverage
```

📚 Additional Resources

Documentation

- React Documentation
- Chakra UI Components
- Prisma Documentation
- Express.js Guide

Learning Resources

- TypeScript Handbook
- JWT Best Practices
- Node.js Security

🤝 Contributing

1. **Fork the repository**
2. **Create a feature branch**: `git checkout -b feature/amazing-feature`
3. **Commit changes**: `git commit -m 'Add amazing feature'`
4. **Push to branch**: `git push origin feature/amazing-feature`
5. **Open a Pull Request**

Development Standards

- Use **TypeScript** for type safety
- Follow **ESLint** configuration

- Write **tests** for new features
- Update **documentation** as needed
- Use **conventional commits**

📄 License

This project is licensed under the MIT License - see the LICENSE file for details.

💖 Acknowledgments

- **React Team** for the amazing framework
- **Chakra UI** for beautiful components
- **Prisma** for excellent database tools
- **Vercel** for deployment platform
- **Railway** for easy deployment

🆘 Support

If you need help or have questions:

1. **Check the documentation** above
2. **Search existing issues** on GitHub
3. **Create a new issue** with detailed information
4. **Join our community** discussions

Happy coding! 🎉

Made with ❤️ by [Your Name]