**Catalyst Wedding - Project Documentation**

Version: 1.0.0  
Last Updated: September 2025  
Maintainer: Development Team

# Table of Contents

1. Project Overview
2. Technology Stack
3. Project Structure
4. Features
5. User Types & Authentication
6. API Endpoints
7. Database Schema
8. Installation & Setup
9. Development Guidelines
10. Deployment
11. Future Enhancements
12. Support

# 1. Project Overview

Catalyst Wedding is a comprehensive wedding planning platform that connects couples with wedding vendors and provides tools for planning their special day. The platform serves both regular users (couples) and vendors, offering different functionalities based on user type.

## Key Objectives:

* Connect couples with verified wedding vendors
* Provide vendor management tools for service providers
* Offer wedding planning resources (photos, invitations, venues)
* Streamline the wedding planning process

# 2. Technology Stack

## Frontend

* React - Modern React with latest features
* React Router - Client-side routing
* CSS3 - Custom styling with responsive design
* Framer Motion 12.23.12 - Animation library
* React Icons 5.5.0 - Icon components

## Backend

* Node.js - Runtime environment
* Next js for backend logics
* MongoDB - NoSQL database
* bcryptjs 2.4.3 - Password hashing

## Development Tools

* Create React App - React development environment
* Nodemon 3.1.4 - Development server auto-restart
* Web Vitals 2.1.4 - Performance monitoring

# 3. Project Structure

The Catalyst Wedding project is divided into two main parts: frontend and backend.

Frontend (React application) includes components, pages, data files, and styles. Backend (Node.js application) contains models, routes, middleware, and entry point configuration.

wedding/  
├── my-app/ # Frontend React Application  
│ ├── public/ # Static assets  
│ ├── src/ # Source code  
│ │ ├── components/ # React components  
│ │ ├── pages/ # Page components  
│ │ ├── data/ # Static data files  
│ │ └── styles/ # CSS files  
│ └── package.json # Frontend dependencies  
├── server/ # Backend Node.js Application  
│ ├── src/  
│ │ ├── models/ # Database models  
│ │ ├── routes/ # API routes  
│ │ ├── middleware/ # Custom middleware  
│ │ └── index.js # Server entry point  
│ └── package.json # Backend dependencies

# 4. Features

## For Regular Users (Couples)

* Vendor Discovery: Browse vendors, filter by location/price, view profiles
* Wedding Planning Tools: Photo galleries, e-invitations, venue search, popular searches
* User Account Management: Registration, login, profile editing, order tracking

## For Vendors

* Vendor Dashboard: Add/manage services, view orders, update info
* Service Management: Detailed listings, portfolio images, pricing, categories

## General Features

* Responsive Design: Mobile-first, cross-device compatibility, modern UI/UX
* Navigation System: Mega menu, breadcrumbs, search functionality
* Content Management: Galleries, invitation templates, vendor directory, blog

# 5. User Types & Authentication

## User Roles

Regular Users (Couples): Can register, browse vendors, view photos, create invitations.

Vendors: Register with business details, manage services, view orders.

## Authentication Flow

* Login system: Separate login for users and vendors
* Role-based Access: Different menus based on user type
* Session Management: Persistent login using localStorage

# 6. API Endpoints

## Authentication Routes (/api/auth)

* POST /users/register - User registration
* POST /users/login - User login
* POST /vendors/register - Vendor registration
* POST /vendors/login - Vendor login

## Profile Routes (/api/profile)

* GET /profile - Get user profile
* PUT /profile - Update user profile

## Vendor Services Routes (/api/vendorServices)

* GET /services - Get all services
* POST /services - Create new service
* PUT /services/:id - Update service
* DELETE /services/:id - Delete service
* GET /services/vendor/:vendorId - Get vendor's services

# 7. Database Schema

The database schema is based on MongoDB with the following models: User, Vendor, VendorService.

## User Model

{  
 name: String,  
 email: String (unique),  
 mobile: String,  
 passwordHash: String,  
 role: String ("user"),  
 timestamps: true  
}

## Vendor Model

{  
 businessName: String,  
 contactName: String,  
 email: String (unique),  
 phone: String,  
 category: String,  
 city: String,  
 passwordHash: String,  
 role: String ("vendor"),  
 timestamps: true  
}

## VendorService Model

{  
 vendorId: ObjectId,  
 type: String,  
 name: String,  
 city: String,  
 location: String,  
 price: String,  
 capacity: String,  
 features: [String],  
 rating: Number,  
 reviews: Number,  
 timestamps: true  
}

## Vendor Orders

{  
 vendorName: String,  
 fromName: String,  
 fromEmail: String (unique),  
 fromPhone: String,  
 category: String,  
 message: String,  
 status: String,  
  
}

# 8. Installation & Setup

Prerequisites: Node.js (v14+), MongoDB.

## Frontend Setup

cd my-app  
npm install  
npm start

## Backend Setup

cd server  
npm install  
npm run dev

## Environment Variables

MONGODB\_URI=mongodb://localhost:27017/wedding  
DB\_NAME=wedding  
PORT=4000

# 9. Development Guidelines

* Use functional components with hooks
* Use CSS modules for styling
* Use PropTypes for props validation
* Use try-catch for API error handling
* Mobile-first responsive CSS
* Lazy loading for performance

# 10. Deployment

Frontend: Build React app (npm run build) and deploy build folder.

Backend: Deploy Node.js with MongoDB Atlas or local DB, configure CORS, environment variables.

# 11. Future Enhancements

* Payment Integration (Stripe/PayPal)
* Real-time Chat
* Review System
* Calendar Integration
* Mobile App
* Advanced Search with AI
* Analytics Dashboard

# 12. Support

For support, check documentation, review issues, or create new issue. Contact development team if needed.