Wedding Planner Platform - Development Playbook

Version: 1.0 (MVP)
Last Updated: October 1, 2025
Status: Successfully Deployed to Production

Table of Contents

Project Overview

The Wedding Planner Platform is a comprehensive web application designed to help couples plan their perfect wedding. Users can browse and search for wedding vendors (venues, photographers, caterers, florists, decorators, musicians), manage their wedding checklist, track budgets, and organize their planning process.

Production URL: https://wedding-planner-platform-seven.vercel.app

GitHub Repository: sfadda-dotcom/wedding-planner-platform

Technology Stack

Frontend

- **Framework:** Next.js 14.2.28 (App Router)
- **UI Library:** React 18.2.0
- **Styling:** Tailwind CSS 3.3.3
- **UI Components:** Radix UI + Custom shadcn/ui components
- **State Management:** React Hooks (useState, useEffect), SWR for data fetching

- **Icons:** Lucide React

Backend

- **Framework:** Next.js API Routes
- **Database ORM:** Prisma 6.7.0
- **Database:** PostgreSQL (Neon.tech)
- **Authentication:** NextAuth.js 4.24.11 (Credentials Provider)
- **Password Hashing:** bcryptjs

Deployment

- **Platform:** Vercel
- **Database Host:** Neon.tech (PostgreSQL with connection pooling)

Project Structure

```
wedding_planner_platform/
├── app/
│   ├── api/                      # API routes
│   │   ├── auth/                # NextAuth configuration
│   │   ├── register/            # User registration endpoint
│   │   ├── vendors/             # Vendor CRUD endpoints
│   │   ├── checklist/           # Checklist management
│   │   └── budget/              # Budget tracking
│   ├── auth/                    # Authentication pages (login, register)
│   ├── dashboard/               # User dashboard
│   ├── vendors/                 # Vendor browsing and details
│   ├── checklist/               # Wedding checklist page
│   ├── budget/                  # Budget tracking page
│   ├── layout.tsx               # Root layout with SessionProvider
│   ├── page.tsx                 # Homepage
│   └── globals.css              # Global styles
├── components/                  # React components
│   ├── ui/                      # Reusable UI components (shadcn)
│   ├── auth/                    # Auth-specific components
│   ├── vendors/                 # Vendor-related components
│   └── dashboard/               # Dashboard components
├── lib/
│   ├── db.ts                    # Prisma client singleton
│   ├── utils.ts                 # Utility functions
│   └── types.ts                 # TypeScript types
├── prisma/
│   └── schema.prisma            # Database schema
├── public/                      # Static assets
├── .env                         # Environment variables (local)
├── next.config.js               # Next.js configuration
├── tailwind.config.ts           # Tailwind configuration
├── tsconfig.json                # TypeScript configuration
├── vercel.json                  # Vercel deployment config
└── package.json                 # Dependencies
```

Initial Setup

1. Project Initialization

The project was built using Next.js 14 with TypeScript and Tailwind CSS. Key dependencies were installed including:
- Prisma for database management
- NextAuth.js for authentication
- Radix UI components for accessible UI elements
- bcryptjs for password hashing

2. Database Schema Design

The Prisma schema includes the following models:

- **User:** Stores user account information
- **Vendor:** Wedding service providers (venues, caterers, etc.)
- **ChecklistItem:** Wedding planning tasks
- **BudgetItem:** Budget tracking entries
- **Account/Session:** NextAuth.js session management

Database Configuration

Initial Setup (Internal Database - Development)

Initially, the app used an internal Abacus.AI PostgreSQL database for development:

```
DATABASE_URL="postgresql://[internal_connection]"
```

This worked perfectly for local development and testing.

Migration to Neon.tech (Production)

Issue Encountered: When deploying to Vercel, the internal database was not accessible from Vercel's servers, causing connection timeout errors during user registration.

Solution: Migrated to Neon.tech, a cloud-hosted PostgreSQL provider with a generous free tier.

Neon.tech Setup Steps

1. **Create Neon Account**
   - Signed up at https://neon.tech/
   - Created a new project: "wedding-planner-platform"

2. **Neon Integration with Vercel**
   - Used Neon's Vercel Integration (Integrations tab in Neon dashboard)
   - This automatically created environment variables in Vercel
   - Note: The integration created `DATABASE_URL_UNPOOLED` but the app requires `DATABASE_URL` (pooled connection)

3. **Connection String Format**
   - **Pooled Connection (Required):** Uses `@pooler` subdomain with port 5432
   `postgresql://neondb_owner:PASSWORD@ep-red-pine-agvs1t5u-pooler.us-east-2.aws.neon.tech/`

`neondb?sslmode=require&channel_binding=require`
- **Unpooled Connection:** Direct connection without pooling (not suitable for serverless)

4. **Common Issues & Solutions**

Issue 1: Incorrect Password
- Symptom: “password authentication failed”
- Solution: Ensured the complete password was copied (check for special characters)

Issue 2: Missing `-pooler` in hostname
- Symptom: Connection timeout or “endpoint idle”
- Solution: Confirmed the hostname includes `-pooler` before `.us-east-2.aws.neon.tech`

Issue 3: Wrong port
- Symptom: Connection refused
- Solution: Used port 5432 (default PostgreSQL port) for pooled connections

1. **Database Synchronization**

After configuring the correct connection string in `.env`:

```
bash
   cd /home/ubuntu/wedding_planner_platform/app
   yarn prisma db push
```

This command synchronized the Prisma schema with the Neon database without requiring migrations.

Authentication Setup

NextAuth.js Configuration

- **Provider:** Credentials (email + password)
- **Session Strategy:** JWT-based
- **Password Security:** bcryptjs with salt rounds

Key Files

- `app/api/auth/[...nextauth]/route.ts` - NextAuth configuration
- `app/api/register/route.ts` - User registration endpoint
- `app/auth/login/page.tsx` - Login UI
- `app/auth/register/page.tsx` - Registration UI

Authentication Flow

1. User registers with email, password, name, and partner name
2. Password is hashed using bcryptjs
3. User record is created in the database
4. User can then log in using credentials
5. JWT token is issued and stored in session
6. Protected pages check for active session using `useSession()` hook

Session Management

All protected pages wrap content with session checks:

```
const { data: session, status } = useSession() || {};

if (status === 'loading') return <div>Loading...</div>;
if (!session) return redirect('/auth/login');
```

Deployment to Vercel

GitHub Integration

1. Pushed code to GitHub repository: `sfadda-dotcom/wedding-planner-platform`
2. Connected Vercel to GitHub account
3. Imported the repository into Vercel

Build Configuration

Vercel Settings:
- **Framework Preset:** Next.js
- **Root Directory:** `/` (project root)
- **Build Command:** `yarn build` (handled by Vercel automatically)
- **Output Directory:** `.next` (default)

Custom Build Script:

Created `vercel-build.sh` to ensure Prisma client generation during build:

```
#!/bin/bash
cd app
yarn prisma generate
yarn build
```

Vercel Configuration (`vercel.json`):

```
{
  "buildCommand": "bash vercel-build.sh",
  "installCommand": "cd app && yarn install"
}
```

Deployment Process

1. Push code to GitHub
2. Vercel automatically detects changes and triggers build
3. Build process:
   - Installs dependencies (`yarn install`)
   - Generates Prisma client (`yarn prisma generate`)
   - Builds Next.js app (`yarn build`)
4. Deployment completes and app is live

Multiple Deployments

The app was deployed multiple times while fixing issues:
- Initial deployment: Database connection issues

- Second deployment: Fixed environment variables
- Final deployment: Successfully working with Neon database

Environment Variables

Local Development (`.env`)

```
DATABASE_URL="postgresql://neondb_owner:PASSWORD@ep-red-pine-agvs1t5u-pooler.us-
east-2.aws.neon.tech/neondb?sslmode=require&channel_binding=require"
NEXTAUTH_SECRET="1enoIqXPKX15RGkpydVkAA8v0KoVPiIv"
NEXTAUTH_URL="http://localhost:3000"
```

Vercel Production Environment Variables

Set in Vercel Dashboard → Settings → Environment Variables → All Environments:

1. **DATABASE_URL**
   - Value: The **pooled** Neon connection string
   - Environment: Production, Preview, Development
   - **Critical:** Must be the pooled connection string (with `-pooler` in hostname)

2. **NEXTAUTH_URL**
   - Value: `https://wedding-planner-platform-seven.vercel.app`
   - Environment: Production

3. **NEXTAUTH_SECRET**
   - Value: `1enoIqXPKX15RGkpydVkAA8v0KoVPiIv`
   - Environment: All

Important Notes

- **Never commit `.env` file to Git** (it's in `.gitignore`)
- After updating environment variables in Vercel, **redeploy the app** for changes to take effect
- The `DATABASE_URL` must match exactly between local and Vercel for consistency

---

Key Features Implemented

1. User Authentication & Registration

- User sign-up with email, password, name, and partner name
- Secure login with NextAuth.js
- Session management across the app
- Password hashing for security

2. Vendor Directory

- Browse vendors by category:
- Venues
- Photographers
- Caterers

- Florists
- Decorators
- Musicians
- Vendor detail pages with:
- Description
- Pricing
- Location
- Contact information
- Image gallery
- Search functionality (to be fixed)

3. Wedding Checklist

- Comprehensive checklist of wedding planning tasks
- Mark tasks as complete/incomplete
- Filter by category (venue, invitations, attire, etc.)
- Track progress with visual indicators

4. Budget Tracker

- Add budget items with:
- Category
- Estimated cost
- Actual cost
- Paid status
- Visual budget summary
- Track total estimated vs. actual spending

5. User Dashboard

- Overview of wedding planning progress
- Quick access to checklist items
- Budget summary
- Upcoming tasks

Known Issues & Future Improvements

Known Issues (MVP Scope - Not Critical)

1. **Vendor Search Not Functional**
   - The search bar in vendors page doesn't filter results
   - Low priority for MVP
   - To be fixed in next iteration

2. **No Email Notifications**
   - User doesn't receive confirmation email after registration
   - No email notifications for reminders or updates
   - Would require email service integration (SendGrid, Resend, etc.)
   - Future enhancement: Welcome emails, task reminders, vendor inquiry notifications

3. **Color Scheme**
   - Current colors work but may need refinement
   - Consider brand identity adjustments in future

Future Enhancements (Backlog)

High Priority

- [ ] Fix vendor search functionality
- [ ] Add email notification system
- [ ] Implement email verification for new users
- [ ] Add password reset functionality
- [ ] Improve color scheme and branding

Medium Priority

- [ ] Add vendor favorites/bookmarks
- [ ] Implement vendor reviews and ratings
- [ ] Add guest list management
- [ ] Create seating chart planner
- [ ] Add file upload for wedding documents (contracts, receipts)
- [ ] Implement real-time budget alerts

Low Priority

- [ ] Mobile app version (React Native)
- [ ] Wedding website builder
- [ ] Registry integration
- [ ] RSVP management
- [ ] Vendor messaging system
- [ ] Wedding day timeline builder

Troubleshooting Guide

Problem: “Timed out fetching a new connection from the connection pool”

Cause: Database URL is not accessible from Vercel (using internal/local database)

Solution:
1. Use a cloud-hosted database (Neon.tech, Supabase, Vercel Postgres)
2. Update `DATABASE_URL` in Vercel environment variables
3. Ensure you’re using the **pooled** connection string
4. Redeploy the app

Problem: “password authentication failed for user”

Cause: Incorrect password in connection string or incomplete password copy

Solution:
1. Go to Neon dashboard
2. Copy the complete connection string (with full password)
3. Verify no characters are missing

4. Update `.env` locally and Vercel environment variables
5. Test locally with `yarn prisma db push`

Problem: “relation does not exist” (Prisma errors)

Cause: Database schema not synchronized

Solution:

```
cd app
yarn prisma db push
```

Problem: Build fails on Vercel with “Cannot find module @prisma/client”

Cause: Prisma client not generated during build

Solution:
1. Ensure `vercel-build.sh` includes `yarn prisma generate`
2. Verify `vercel.json` points to the build script
3. Check that Prisma is in dependencies (not devDependencies)

Problem: Session not persisting (user gets logged out immediately)

Cause: `NEXTAUTH_URL` doesn’t match the actual deployment URL

Solution:
1. Update `NEXTAUTH_URL` in Vercel to match exact production URL
2. Redeploy the app

Problem: 404 on API routes

Cause: Next.js App Router structure issue

Solution:
1. Verify API routes are in `app/api/` directory
2. Ensure each route has a `route.ts` file (not `index.ts`)
3. Check that exports are named correctly (`GET`, `POST`, etc.)

Key Learnings

1. Database Connection Pooling is Critical for Serverless

Vercel uses serverless functions, which means each request spawns a new connection. Without connection pooling, you’ll quickly exhaust database connections. Always use pooled connections (e.g., Neon’s `-pooler` endpoint).

2. Environment Variables Must Match Across Environments

Inconsistencies between local `.env` and Vercel environment variables cause deployment issues. Always test locally with the exact same connection strings you’ll use in production.

3. Prisma Client Must Be Generated During Build

In serverless environments, the Prisma client must be generated fresh during each build. Include `prisma generate` in your build command.

4. NextAuth.js Requires Exact URL Matching

The `NEXTAUTH_URL` must match your deployment URL exactly, including protocol (https://) and domain. Mismatches cause session issues.

5. Test Registration and Login After Every Deployment

The most critical user journey is sign-up and login. Always test this after deploying to catch database or authentication issues early.

6. MVP Mindset: Ship First, Perfect Later

It’s okay to have known issues in an MVP. Document them, prioritize ruthlessly, and ship a working product. You can iterate based on real user feedback.

7. Use Integration Services When Available

Neon’s Vercel integration automatically set up environment variables, saving manual configuration time. Look for official integrations between services you use.

8. Keep a Deployment Checklist

Pre-Deployment Checklist:
- [ ] Code pushed to GitHub
- [ ] Environment variables set in Vercel
- [ ] Database schema synchronized
- [ ] Build succeeds locally
- [ ] Critical user flows tested locally

Post-Deployment Checklist:
- [ ] Deployment successful (check Vercel dashboard)
- [ ] Homepage loads without errors
- [ ] User registration works
- [ ] User login works
- [ ] Protected pages require authentication
- [ ] Database queries return expected data

Quick Reference Commands

Local Development

```
# Start development server
cd app && yarn dev

# Generate Prisma client
cd app && yarn prisma generate

# Sync database schema
cd app && yarn prisma db push

# Open Prisma Studio (database GUI)
cd app && yarn prisma studio

# Build for production (test locally)
cd app && yarn build

# Start production server (after build)
cd app && yarn start
```

Database Management

```
# View current database URL
cd app && cat ../.env | grep DATABASE_URL

# Test database connection
cd app && yarn prisma db pull
```

Git & Deployment

```
# Push to GitHub (triggers Vercel deployment)
git add .
git commit -m "Your commit message"
git push origin main

# View deployment logs
# Visit: https://vercel.com/your-username/wedding-planner-platform
```

Contact & Support

For issues or questions:
- Review this playbook first
- Check the Troubleshooting Guide
- Review Vercel deployment logs
- Check Neon database logs
- Consult Next.js documentation: https://nextjs.org/docs
- Consult Prisma documentation: https://www.prisma.io/docs

Version History

v1.0 - October 1, 2025
- Initial MVP deployment
- Core features: Auth, Vendors, Checklist, Budget
- Successfully deployed to Vercel with Neon database
- Known issues documented for future iterations

End of Playbook

This document should be updated as new features are added or issues are resolved.