SYNOPSIS OF EXPLORATORY PROJECT (EP)

ON

NEXTSTEP

submitted in partial fulfilment of the requirements for the award of degree of

BACHELOR OF ENGINEERING

In

COMPUTER SCIENCE AND ENGINEERING

Submitted by:

Abhishek Duggal (2210991163)

Akshit Kansal (2210990092)

Aargya Sharma (2210991116)

Swastik Chauhan (2210992431)

Supervised By:

Dr. Ajay Kumar

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CHITKARA UNIVERSITY INSTITUTE OF ENGINEERING AND TECHNOLOGY
CHITKARA UNIVERSITY, PUNJAB, INDIA

CONTENTS

ABSTRACT

The Destination Wedding Planner is a comprehensive web-based platform designed to streamline and digitize the end-to-end process of planning a destination wedding. The system bridges the gap between clients (couples), planners, and vendors, offering a centralized solution for browsing exotic wedding destinations, customizing wedding packages, hiring trusted vendors, managing bookings, and tracking wedding planning progress.

Couples can filter destinations by style, location, price, and guest capacity, and can personalize wedding packages to fit cultural, traditional, or luxurious preferences. Planners manage bookings, suggest packages, assign vendors, and handle custom requests. Vendors, on the other hand, manage their offerings, availability, and invoices. An admin panel facilitates operational oversight, enabling the management of users, packages, destinations, payments, and analytics.

The system is designed using a role-based architecture, ensuring personalized dashboards, secure interactions, and efficient coordination across roles. With MongoDB as the primary database and Express.js/Node.js powering RESTful APIs, this platform delivers a scalable and modular backend. Through this tool, destination wedding planning is transformed from a tedious offline process into a seamless and joyful digital experience.

1) INTRODUCTION

The Destination Wedding Planner is a comprehensive digital platform designed to revolutionize the way couples plan and execute their dream weddings at beautiful destinations. Traditional wedding planning often involves excessive coordination, manual bookings, and limited access to reliable vendors. This platform addresses these challenges by offering an all-in-one solution that connects couples with professional planners and verified vendors, while also allowing real-time tracking, customization, and secure payments.

This system introduces four key user roles—Users (Couples), Planners, Vendors, and Admins—each with distinct capabilities and responsibilities. Users can browse curated destinations, select or customize wedding packages, choose from shortlisted vendors, and manage all aspects of their wedding planning from a single dashboard. Planners act as coordinators who manage bookings, suggest packages, assign vendors, and oversee customization requests. Vendors can list their services, manage availability, and upload deliverables such as sample designs or menus. Admins maintain the integrity of the platform by managing users, destinations, transactions, and analytics.

By integrating destination browsing, vendor selection, service customization, and financial management into one seamless experience, the platform ensures that weddings are not only beautifully planned but also efficiently executed. The project's architecture is built using a role-based approach with well-defined database schemas and RESTful API controllers to ensure scalability, modularity, and ease of maintenance.

Overall, this platform transforms the wedding planning experience by combining personalization, automation, and professional coordination—making dream weddings both attainable and stress-free.

2) METHODOLOGY

2.1. Requirement Analysis

The project began with identifying core user roles: User (Couple), Planner, Vendor, and Admin. Each role was mapped to specific use cases such as wedding customization, vendor management, feedback collection, and analytics tracking. Functional and non-functional requirements were documented based on user expectations and industry benchmarks.

2.2. System Architecture & Technology Stack

The platform follows a Modular MVC architecture, using:

Backend: Node.js with Express for API services

Database: MongoDB for flexible schema and scalability

Frontend: React.js

Authentication: JWT-based secure login for different roles

2.3. Schema Design

Twelve core models were designed to support all business logic. These include User, Planner, Vendor, Wedding Plan, Destination, Package, and Booking. The schema relationships are built using references to allow normalized storage and efficient querying.

2.4. API Layer Design

RESTful APIs are developed for each controller, grouped by roles:

User Controller enables destination browsing, wedding planning, and vendor selection.

Planner Controller focuses on managing bookings, assigning vendors, and tracking customizations.

Vendor Controller handles assignments, service updates, and invoice uploads.

Admin Controller supports moderation, dispute resolution, and insights via dashboards.

2.5. Security and Role Management

Role-based access is enforced across all APIs. Sensitive operations like payments, booking approvals, and review moderation are restricted to authenticated users of specific roles.

3) Tools and Technologies

The *Destination Wedding Planner* platform is built using modern and scalable technologies to ensure seamless user experience, efficient backend processing, and maintainability. The following tools and technologies were used in the development of the system:

3.1. Frontend Technologies

- React.js – For building dynamic, component-based user interfaces for users, planners, and admins.
- Tailwind CSS – For responsive design and prebuilt UI components.

3.2. Backend Technologies

- Node.js – JavaScript runtime environment used to build a scalable backend.
- Express.js – A minimal and flexible Node.js framework for routing and middleware handling.
- Multer / Cloudinary – For handling file uploads such as invoices, images, and deliverables.

3.3. Database

- MongoDB – NoSQL database used for flexible, document-oriented data storage.

3.4. Payment Integration

- Razorpay / Stripe – For secure online payment processing, invoice generation, and refund handling.

3.5. Deployment

- Vercel – For deploying the frontend.
- Render – For backend deployment.
- Git & GitHub – Version control and collaborative development.

4) PROJECT PLAN

Week 1:

The initial phase involves gathering requirements to understand the core needs and expectations of the project. Based on these insights, user flows and detailed use cases are drafted to outline how different users will interact with the system. Once the functionalities are clearly mapped, the next step is to finalize the technology stack that best supports the project's goals and scalability. Simultaneously, an Entity Relationship (ER) diagram is designed to model the database structure and relationships among entities. Finally, UI wireframes are sketched to visualize the user interface layout and ensure a seamless user experience before moving into the development phase.

Week 2: ‘

The database design phase begins with defining MongoDB schemas that accurately represent the data structure required for the application. Each schema outlines the fields, data types, and constraints needed for storing relevant information. To establish relationships between different collections, ObjectId references are used, enabling efficient linking and data retrieval. Once the schemas and relationships are in place, Mongoose models are set up to interact with the MongoDB database, providing a structured and organized way to perform CRUD operations within the application.

Weeks 3–5:

The backend REST API development begins with implementing secure user authentication using JSON Web Tokens (JWT), ensuring only authorized access to protected routes. Controllers are then built for key roles including User, Planner, Vendor, and Admin, each handling specific business logic and data operations. The core functionalities such as booking and payment processing are also implemented, managing transactions and reservations seamlessly. Throughout the development process, Postman is used to test and validate API endpoints, ensuring they respond correctly and reliably across all scenarios.

Weeks 6–8:

The frontend is built using React, featuring role-based dashboards tailored to Users, Planners, Vendors, and Admins, each with customized views and functionalities. The interface seamlessly integrates with backend REST APIs to fetch and display real-time data. Key UI components are implemented for booking services, managing calendars, and facilitating real-time chat, ensuring a smooth and interactive user experience across all roles.

Week 9:

The testing phase involves conducting unit tests on core APIs to ensure each function performs as expected in isolation. Role-based access validation is implemented to verify that each user type—User, Planner, Vendor, and Admin—can only access their authorized features and data. Additionally, real-world workflows, such as an end-to-end wedding booking process, are simulated to test the system's behavior under realistic conditions and ensure all integrated components work seamlessly together.

Week 10:

The deployment phase involves hosting the backend on Render, ensuring reliable server performance and scalability. Continuous Integration and Continuous Deployment (CI/CD) pipelines are set up using GitHub to automate the build and deployment processes, enabling faster and error-free updates. The frontend is deployed on Vercel, providing a high-performance, globally distributed environment for delivering a fast and responsive user interface.