Project Roadmap for Artist-Brand Web Application

1. Introduction

This web application aims to connect brands with artists such as photographers, developers, graphic designers, and video editors. Brands can hire artists for specific roles based on their portfolio and requirements. The technology stack includes Express.js, Node.js, MongoDB Atlas, and React.js.

2. Features

User Types

1. **Artists**: Register and upload their portfolios, respond to booking requests, and edit profiles.
2. **Brands**: Register, search for artists, send booking requests, and view artist contact details after deal verification.
3. **Admin**: Verify deals, approve profile edits, and oversee the platform.

3. Backend Roadmap

3.1 Schema Design

- **User Model**:
  - `user_id`: Unique identifier for each user.
  - `user_type`: Either 'artist' or 'brand'.
  - `name`: Full name of the user.
  - `email`: Email address, unique.
  - `password`: Encrypted password.
  - `profile_pic`: URL for the profile picture.
  - `registered_date`: Timestamp of account creation.

- **Artist Model** (extends User Model):
  - `portfolio`: Array of project URLs or file links.
  - `specialization`: Role like 'Photographer', 'Developer', etc.
  - `availability_status`: Boolean to indicate if open for bookings.
  - `edit_request_status`: Boolean to show if the artist has a pending profile edit request.

- **Brand Model** (extends User Model):
  - `verified`: Boolean to indicate if verified by the admin.
  - `company_name`: Brand or company name.
  - `company_address`: Address of the brand.

- **Booking Model**:
  - `booking_id`: Unique identifier for each booking.
  - `brand_id`: Reference to the `User` model (brand user).
  - `artist_id`: Reference to the `User` model (artist user).
  - `role`: Role for which artist is booked.
  - `price`: Agreed-upon price.
  - `duration`: Duration of the booking.
  - `status`: Status like 'Pending', 'Accepted', 'Rejected', 'Verified'.
  - `created_at`: Timestamp when the booking request was made.

3.2 Authentication & Authorization

1. **JWT-based Authentication**:
   - Implement using `jsonwebtoken` library for creating and verifying tokens.
   - Generate tokens during user login and validate them for protected routes.
   - Store tokens securely in HTTP-only cookies for session management.
2. **Role-Based Access Control**:

- Use custom middleware to restrict access to specific routes based on user roles (`artist`, `brand`, `admin`).
- Only verified brands can book artists, and only artists can respond to booking requests.

3.3 Backend Routes & Controllers

Auth Routes

- `POST /auth/signup`: Register a new artist or brand.
- `POST /auth/login`: Login for both user types.
- `POST /auth/logout`: Logout and clear session cookies.

Artist Routes

- `GET /artists`: List all artists based on categories (e.g., photographers, developers).
- `GET /artists/:id`: Get specific artist details.
- `POST /artists/update-profile`: Update artist profile (sends approval request to admin).
- `POST /artists/respond-booking`: Respond to a booking request (`Accepted` or `Rejected`).
- `GET /artists/bookings`: View all booking requests made by brands.

Brand Routes

- `POST /brands/book`: Book an artist for a specified role.
- `GET /brands/bookings`: View all bookings made by the brand.
- `GET /brands/search`: Search for artists by specialization or name.

Admin Routes

- `GET /admin/pending-verifications`: View all pending deals requiring approval.
- `POST /admin/verify-booking`: Approve a booking request.
- `POST /admin/approve-profile-edit`: Approve artist profile changes.

3.4 Database Models & Security

- **Database Models**:
  - Use Mongoose to define schema models for `User`, `Artist`, `Brand`, and `Booking`.
  - Implement cascading delete for related bookings when a user is deleted.
- **Security Measures**:
  - Use `bcryptjs` for hashing and salting user passwords.
  - Validate inputs using middleware to prevent SQL injection and other attacks.
  - Implement HTTPS with `helmet` for added security.
  - Use CORS policy to restrict API access to known domains.

3.5 Notification System

- **Nodemailer Integration**:
  - Send email notifications for booking requests, deal verifications, and profile edit approvals.
  - Implement email templates for better readability and structure.
- **WebSocket Integration**:
  - Use WebSockets for real-time notifications when a booking request is accepted, rejected, or approved.

3.6 Middleware Functions

- **Authentication Middleware**: Validate JWT tokens and user roles for all protected routes.
- **Error Handling Middleware**: Centralized error handling for catching and logging errors.
- **Request Validation Middleware**: Validate incoming request bodies using `express-validator`.

3.7 Backend Logic & Workflows

1. **User Registration**:
   - Validate the registration data and assign default role (`artist` or `brand`).
   - For brands, set `verified` status to `false` until admin verification.

2. **Artist Profile Management**:
   - Allow artists to create and update portfolios (after approval).
   - Track profile edit requests with `edit_request_status` flag.

3. **Booking Process**:
   - Allow verified brands to book artists with role, price, and duration details.
   - Update booking status based on artist's response (`Accepted` or `Rejected`).
   - Trigger admin email for deal verification when an artist accepts a booking.

4. **Admin Verifications**:
   - Manage and verify all pending booking deals.
   - Send email confirmation with `Verification Done` button to finalize deals.
   - Allow admin to approve or reject profile edit requests.

4. Frontend Roadmap

- **React Components**: Separate pages for Login, Registration, Artist Profile, Brand Search, and Admin Verification.
- **State Management**: Use Context API or Redux for managing user state and bookings.
- **Responsive Design**: Ensure all pages are mobile-friendly.

5. Testing & Optimization

- Implement comprehensive testing for all routes and database models.
- Optimize API calls and reduce response time.