1. **Database Design**: Design your database schema. Determine what tables you need, their fields, relationships, etc. Choose a database management system (DBMS) such as MySQL, PostgreSQL, MongoDB, etc., depending on your requirements.
2. **Backend Development**:
   * Choose a backend framework like Django (Python), Express.js (Node.js), Ruby on Rails (Ruby), etc.
   * Set up your backend server to handle HTTP requests from both the website and the application.
   * Implement RESTful APIs or GraphQL endpoints to interact with your database.
3. **Frontend Development**:
   * For the website, you can use HTML, CSS, and JavaScript along with frontend frameworks like React.js, Angular, or Vue.js.
   * For the application, you can use native development (Swift for iOS, Kotlin for Android) or cross-platform frameworks like React Native, Flutter, or Xamarin.
4. **Connecting Backend with Database**:
   * Use appropriate database drivers or ORM (Object-Relational Mapping) libraries to connect your backend server with the database.
   * Implement CRUD (Create, Read, Update, Delete) operations to manipulate data in the database.
5. **Authentication and Authorization**:
   * Implement user authentication and authorization mechanisms to secure your application.
   * Use techniques like JWT (JSON Web Tokens), OAuth, or sessions for authentication.
6. **Deployment**:
   * Deploy your backend server to a hosting provider like Heroku, AWS, Google Cloud Platform, etc.
   * Deploy your website and application to appropriate platforms like web hosting services, app stores, etc.
7. **Testing**:
   * Perform thorough testing of your website and application to ensure functionality, performance, and security.
   * Write unit tests, integration tests, and end-to-end tests to cover all aspects of your system.
8. **Monitoring and Maintenance**:
   * Set up monitoring tools to track the performance and health of your website and application.
   * Regularly update and maintain your codebase to fix bugs, add new features, and improve performance.
9. **Continual Improvement**:
   * Gather feedback from users and iterate on your website and application to make improvements over time.
   * Stay updated with the latest technologies and trends in web and mobile development to keep your project competitive.

**Website Development:**

1. **Backend**:
   * **Django (Python**): Known for its simplicity and rapid development. It has built-in authentication, admin panel, and ORM for interacting with the database.
   * **Express.js (Node.js):** Lightweight and flexible framework for building APIs. It's well-suited for real-time applications and microservices.
2. **Frontend**:
   * **React.js:** A powerful library for building interactive user interfaces. It's component-based and offers great performance.
   * **Angular:** A full-fledged framework maintained by Google. It provides features like two-way data binding, dependency injection, and a powerful CLI.
   * **Vue.js:** A progressive framework known for its simplicity and ease of integration. It's beginner-friendly and offers great performance.
3. **Database**:
   * **PostgreSQL:** A powerful open-source relational database known for its reliability, ACID compliance, and support for advanced features like JSONB data type and indexing.
   * **MongoDB:** A popular NoSQL database that stores data in flexible, JSON-like documents. It's well-suited for projects with unstructured or rapidly changing data.

Based on simplicity, scalability, and widespread adoption, a popular combo you might consider is:

* **Backend**: Django (Python)
* **Frontend**: React.js
* **Database**: PostgreSQL

This combination provides a robust and scalable solution with Django's built-in features, React's flexibility and performance, and PostgreSQL's reliability and advanced features. However, you should assess your specific project requirements and team capabilities before making a decision.

**App Development:**

1. **React Native**:
   * React Native allows you to build native mobile applications for iOS and Android using JavaScript and React.
   * You can reuse much of the logic and components from your React.js web frontend in your React Native mobile app.
   * React Native supports making HTTP requests to your Django backend APIs to fetch and manipulate data stored in your PostgreSQL database.
   * This approach offers code reuse between your web and mobile apps, reducing development time and effort.
2. **Native Development**:
   * You can also build separate native mobile apps using Swift (for iOS) and Kotlin/Java (for Android).
   * These apps can communicate with your Django backend via HTTP requests to consume data from your PostgreSQL database.
   * While this approach doesn't offer as much code reuse as React Native, it allows for fine-tuning and optimization for each platform.
   * Native development may be preferred if you require advanced platform-specific features or performance optimizations.