1. Understand the Requirements

Gather Requirements: Communicate with stakeholders to clearly understand what they want from the application, including the features, user roles, and functionalities.

Define Objectives: Clarify the business goals and user needs the application should address.

2. Create a Wireframe or Mockup

Low-Fidelity Wireframes: Use tools like Figma, Sketch, or Adobe XD to create a basic wireframe or mockup. This would include:

Layout of different pages

Navigation structure

Positioning of key components (buttons, forms, images, etc.)

Focus on User Flow: Ensure the design supports a logical and intuitive user flow.

3. Design Database Schema

ER Diagrams: Draft an entity-relationship diagram (ERD) to model how data will be structured in the application.

Identify Key Entities and Relationships: For example, if building an e-commerce app, entities could include users, products, orders, etc.

Define Data Models: Consider tables, fields, and relationships between different entities.

4. Choose a Tech Stack

Front-end: Select frameworks like React.js, Vue.js, or Angular based on the project needs.

Back-end: Choose a server-side language such as Node.js, Django (Python), or Ruby on Rails.

Database: Depending on requirements, choose SQL (e.g., PostgreSQL, MySQL) or NoSQL (e.g., MongoDB) databases.

Hosting: Consider deployment platforms such as AWS, Google Cloud, or Heroku for hosting the application.

5. Develop the Prototype

Set Up a Basic Front-End: Create a basic front-end that reflects the wireframes. For example:

Navigation menu

Home page with placeholder data

Sample forms for user input

Set Up Back-End API: Create a simple back-end using REST or GraphQL APIs that handle CRUD (Create, Read, Update, Delete) operations.

Database Integration: Set up database tables and connect them to your back-end to store and retrieve data.

6. Add Core Functionalities

User Authentication: Implement simple user login and registration features.

CRUD Operations: Develop basic create, read, update, and delete functionalities for key data entities.

Basic Interactivity: Add simple forms, buttons, and feedback mechanisms for users.

7. Testing

Unit Testing: Write simple test cases for key functionalities.

Usability Testing: Share the prototype with stakeholders or users for feedback on usability and design.

Bug Fixing: Identify and fix any major bugs or issues.

8. Iterate and Refine

Get Feedback: Present the prototype to stakeholders, gather feedback, and iterate on design or functionality based on their input.

Adjust Based on Testing: Make necessary refinements after performing usability and functionality tests.

9. Document the Prototype

User Stories: Document how different user roles will interact with the application.

Technical Documentation: Include descriptions of the tech stack, API endpoints, data models, and architecture.

10. Prepare for Full Development

Once the prototype is approved, I would start expanding it to a full-featured web application by integrating more advanced features, enhancing security, improving performance, and adding scalability.