**CHAPTER TWO: LITERATURE REVIEW**

**2.1 INTRODUCTION**

This chapter reviews existing literature and studies on salon booking systems, decentralized applications (dApps), blockchain integration in web services, and the use of Firebase for off-chain storage. The aim is to understand current solutions, identify gaps, and justify the approach taken in this project.

**2.2 REVIEW OF RELATED WORKS**

Several systems and platforms have been developed to enable salon appointment bookings. Traditional systems primarily use centralized databases to manage appointments. These systems often require trust in a central server and can suffer from data manipulation, security risks, or lack of transparency.

Decentralized booking systems are less common but are emerging in industries requiring transparency and immutability. Some dApps leverage Ethereum smart contracts to handle payments and confirmations without relying on intermediaries.

**2.3 RESEARCH GAPS**

Despite the existence of centralized salon booking platforms, few solutions utilize blockchain technology to ensure transparency and trust in bookings. Many systems do not integrate both on-chain (smart contracts) and off-chain (user metadata) effectively. Additionally, there is limited application of lightweight frontend stacks (HTML/CSS/JS) in such integrations.

**2.4 SYSTEM REQUIREMENTS SPECIFICATION (SRS) FRAMEWORK**

-This project aligns with a functional and non-functional specification framework:

-Functional Requirements:

1. User registration and login
2. Appointment creation and confirmation
3. On-chain recording of bookings
4. Storage of user and booking metadata

-Non-Functional Requirements:

1 . Real-time booking feedback

2. Secure transaction and data handling

3. Scalable and maintainable codebase

4. Accessible and responsive user interface

**2.5 SUMMARY OF LITERATURE REVIEW**

Existing works focus heavily on centralized solutions with limited transparency. Blockchain-based booking systems remain niche, with few that combine Firebase and smart contracts. This project's hybrid approach addresses the research gap by offering a decentralized yet user-friendly solution using accessible tools and technologies.

**CHAPTER THREE: METHODOLOGY**

**3.1 INTRODUCTION**

This chapter outlines the methodology used to design, develop, and test a decentralized booking ledger system for a salon website. The project integrates blockchain technology for transaction integrity and Firebase for off-chain data and authentication. The goal is to create a hybrid system where essential transaction records are securely written on-chain while user and session-related data are managed off-chain for efficiency and scalability.

**3.2 SYTEM DESIGN**

The system follows a hybrid web3 architecture, integrating both blockchain and cloud technologies. The smart contract forms the ledger for bookings and payments, while Firebase handles user data, authentication, and booking metadata. The frontend is built using HTML, CSS, and JavaScript and communicates with the Ethereum blockchain through Ethers.js and MetaMask. Firebase services (Authentication and Firestore) handle user login and data storage.

**Components:**

-Smart Contract Ledger (On-chain): Handles booking confirmation and records.

- Frontend (HTML, CSS, JavaScript): Interfaces with smart contracts using Ethers.js and integrates Firebase services for backend functionality.

-Firebase Services (Off-chain): Manages authentication, user metadata, and auxiliary data for bookings.

-MetaMask: User wallet for signing blockchain transactions.

**3.3 TOOLS AND TECHNOLOGIES**

-Tabulated form of the tools I used to develop this project

|  |  |
| --- | --- |
| **Tools** | **Purpose** |
| Solidity | Writing the smart contract |
| Hardhat | For testing, and deploying smart contracts (locally) |
| Ether.js | Interfacing the frontend with smart contracts |
| Metamask | Wallet to test transactions (e.g., booking confirmations) |
| Firebase Firestore | Stores user profiles and booking metadata (off-chain) |
| Firebase Auth | Manages user login (email) |
| Firebase sdk | Connects frontend to Firebase services |
| HTML/CSS/Javascript | Web application development and integration with smart contracts |

**3.4 DATA COLLECTION/REQUIREMENTS**

-Functional Requirements:

1. Users can create and manage accounts (via Firebase Auth).

2. Users can make bookings (written to both Firebase and blockchain).

3. Admins can monitor the on-chain ledger.

4. Non-Functional Requirements:

5. Security of user data and transactions.

6. Efficient load handling for booking data.

7. Responsive and intuitive frontend interface.

-Data Sources:

1. User inputs (like booking details).

2. On-chain booking confirmations.

3. Firebase user metadata.

**3.5 DEVELOPMENT APPROACH**

-The project uses an agile methodology, structured in iterative sprints. Each sprint targets key milestones such as smart contract development, Firebase integration, frontend interface, and user testing.

-Development Phases:

1. Smart Contract Design and Testing (Solidity + Hardhat)

2. Blockchain Integration (Ethers.js + MetaMask)

3. Firebase Integration (Firestore + Auth)

4. Frontend Development and Styling

5. Fullstack Integration and User Flow Testing

**3.6 TESTING AND VALIDATION**

-Unit Testing (Hardhat): Ensures smart contract functions work as expected.

-Integration Testing: Verifies interactions between frontend and smart contracts/Firebase.

-User Testing: Conducted with sample users to simulate bookings and account creation.

-Validation: Smart contracts will be deployed on a testnet to ensure functionality before mainnet deployment.

**3.7 PROJECT TIMELINE**

|  |  |
| --- | --- |
| **Week** | **Activity** |
| 1-2 | Requirements gathering and system design |
| 3-4 | Smart contract development (Solidity + Hardhat) |
| 5 | Firebase setup (Firestore & Auth) |
| 6-7 | Frontend development and blockchain integration |
| 8 | Testing (Unit, Integration, User) |
| 9 | Final validation and documentation |
| 10 | Project presentation and submission |

**CHAPTER FOUR: BUDGET AND RESOURCES**

1. **Hardware Requirements**

-My laptop (CPU- i7-8865U 2.11 GHz, Ram 8gb)

-Internet access

1. **Software Requirements**

**-Most are open source and don’t cost anything to operate and use, they include:**

**1. Metamask wallet**

**2. Hardhat and Ether.Js**

**3. Firebase Services (used the freemium services)**

**4. Code Editor used the Visual code**

**5. Node.js and React framework**