6. Requirements

1. Nextjs: Develop and maintain the user interface using Next.js.
2. API: Implement API calls to interact with backend services and smart contracts.
3. RWD: Ensure responsive and accessible UI/UX design.
4. Backend: Build and maintain the backend using Node.js and Prisma.
5. Database: data storage using SQLite.
6. Solidity: Develop, deploy, and audit Solidity smart contracts.
7. Deployment: Vercel

8- Architecture:

Blockchain Deployment: Smartcontract —deploy(hardhat)—> ETHSepolia

Frontend Deployment: Nextjs project —deploy(Vercel)—> Vercel —DNS—> webpage

Frontend: Database —send data to frontend—> Show on webpage

Backend: EthSepolia —getData(hardhat)—> Data —decode(hardhat)—> API —prisma(store)—> Database

FundSeekerUser: webpage —wirte&pulishRaisingProposal(pulish)—> if reach raising goal website stops allowing investers—> when need fund—requestPool—> get fund(shares shared)

FundProviderUser: search on webpage —provide fund in pool to become contributor—> when FundSeekerUser request fund —vote to accept—> provide fund to FundSeeker and get interests

8- API:

SmartContract:

JSON API Endpoints

User Authentication

- **POST /api/auth/register**
    - Description: Register a new user.
    - Request Body:`json`

```
{
  "username": "string",
  "email": "string",
  "password": "string"
}
```

    - Response:`json`

```
{
  "message": "User registered successfully",
```

```
  "userId": "string"
}
```

- **POST /api/auth/login**
  - Description: Authenticate user and return a token.
  - Request Body:`json`

```
{
  "email": "string",
  "password": "string"
}
```

  - Response:`json`

```
{
  "token": "string",
  "userId": "string"
}
```

Investment Proposals

- **GET /api/proposals**
  - Description: Retrieve all investment proposals.
  - Response:`json`

```
[
  {
    "id": "string",
    "projectDetails": "string",
    "minInvestment": "number",
    "targetAmount": "number",
    "interestRate": "number",
    "repaymentPeriod": "string"
  }
]
```

- **POST /api/proposals**
  - Description: Create a new investment proposal.
  - Request Body:`json`

```
{
  "projectDetails": "string",
  "minInvestment": "number",
  "targetAmount": "number",
  "interestRate": "number",
  "repaymentPeriod": "string"
}
```

  - Response:`json`

```
{
  "message": "Proposal created successfully",
  "proposalId": "string"
}
```

Investments

- **POST /api/investments**
  - Description: Invest in a project.
  - Request Body:`json`

```
{
  "userId": "string",
  "proposalId": "string",
  "amount": "number"
}
```

  - Response:`json`

```
{
  "message": "Investment successful",
  "investmentId": "string"
}
```

- **GET /api/investments/{userId}**
  - Description: Retrieve all investments by a specific user.
  - Response:`json`

```
[
  {
    "investmentId": "string",
    "proposalId": "string",
    "amount": "number",
    "status": "string"
  }
]
```

Fund Withdrawals

- **POST /api/withdrawals**
  - Description: Request a fund withdrawal.
  - Request Body:`json`

```
{
  "proposalId": "string",
  "amount": "number",
  "reason": "string"
}
```

  - Response:`json`

```
{
  "message": "Withdrawal request submitted",
  "withdrawalId": "string"
```

- }
-

8-testing appoarch:

Testing Approach

- **Unit Testing**: Each API endpoint will be tested independently to verify the correctness of input validation and business logic.
- **Integration Testing**: Ensure interactions between frontend, backend, and blockchain smart contracts function correctly.
- **Security Testing**: Verify authentication, authorization, and protection against vulnerabilities (e.g., SQL injection, XSS, CSRF).
- **Load Testing**: Simulate high traffic to evaluate system performance and scalability.
- **Smart Contract Audits**: Conduct security reviews of Solidity smart contracts to prevent vulnerabilities.

8- ErrorHandling:

Error Handling

- **400 Bad Request**: Invalid input, missing fields, or incorrect format.
- **401 Unauthorized**: Token is missing or invalid.
- **403 Forbidden**: User does not have permission to perform the action.
- **404 Not Found**: Requested resource does not exist.
- **500 Internal Server Error**: Unexpected server-side issues.

8-NFR:

Non-Functional Requirements (NFR)

- **Security**: Implement JWT authentication and enforce strict access control.
- **Scalability**: Support horizontal scaling by deploying backend services in a containerized environment.
- **Performance**: Ensure API response time is under 200ms for 95% of requests.
- **Availability**: Maintain 99.9% uptime by utilizing cloud-based infrastructure.
- **Data Integrity**: Utilize blockchain immutability to prevent unauthorized fund manipulations.