Introduction

This document outlines the planning and design phase for a peer-to-peer (P2P) Bitcoin lending platform. The platform operates under a 100% collateralized loan model, ensuring security without relying on traditional financial intermediaries. This documentation covers software architecture, development methodologies, client-server architecture, ethical considerations, and user stories.

---

1. Planned Software Architecture

Programming Paradigm

Our project follows an **object-oriented programming (OOP)** paradigm. OOP is a paradigm that organises code into ‘objects’, bundles of data (properties) and functions (methods) that operate on that data. It helps model real-world entities and makes programs more scalable, reusable, and maintainable. OOP uses principles like encapsulation (hiding internal data), inheritance (sharing functionality between objects), polymorphism (handling different types through a unified interface), and abstraction (simplifying complex realities).

- **Language**: JavaScript (Node.js, Express backend, and React frontend)
- **Objects**:
    - User
        - username, walletId, email, password, createdAt, isActive, isAdmin
    - LoanRequest
        - borrowerId, requestAmount, interestTermId, cryptocurrencyId, requestDate, expiryDate
    - Deal
        - lenderId, loanDetailsId (reference to LoanRequest), dealDate, isComplete
    - Collateral
        - dealId, amount
    - Transaction
        - fromUserId, toUserId, dealId, amount, isLoanRepayment, expectedDate, paymentStatus
    - InterestTerm
        - loanLength, interestRate
    - Cryptocurrency
        - symbol, name

Each object will have distinct properties and methods, reflecting real-world entities and actions.

Below is a diagram outlining how OOP is structured, using classes, instances of that class (objects), properties, and methods.

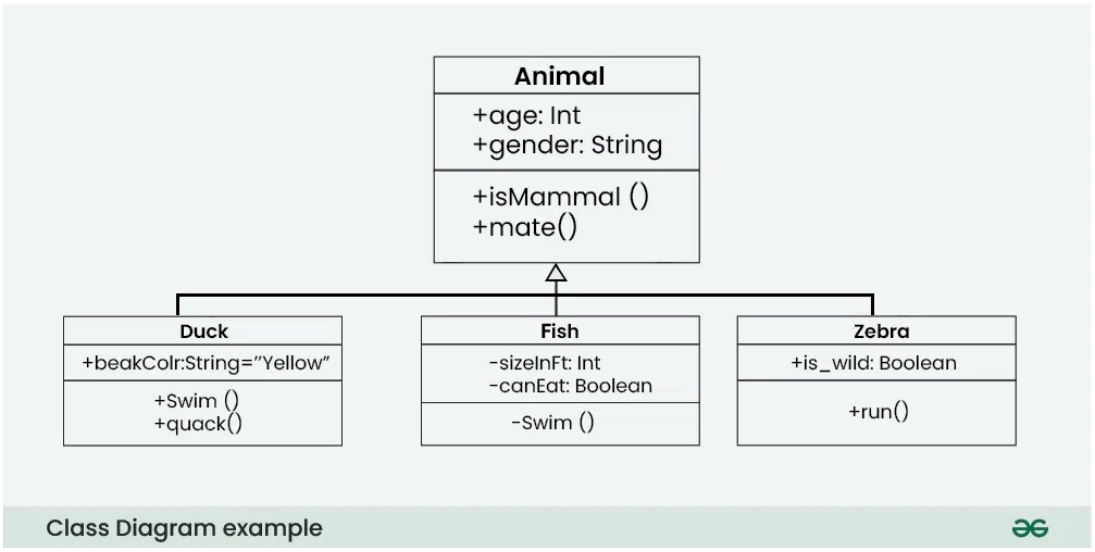
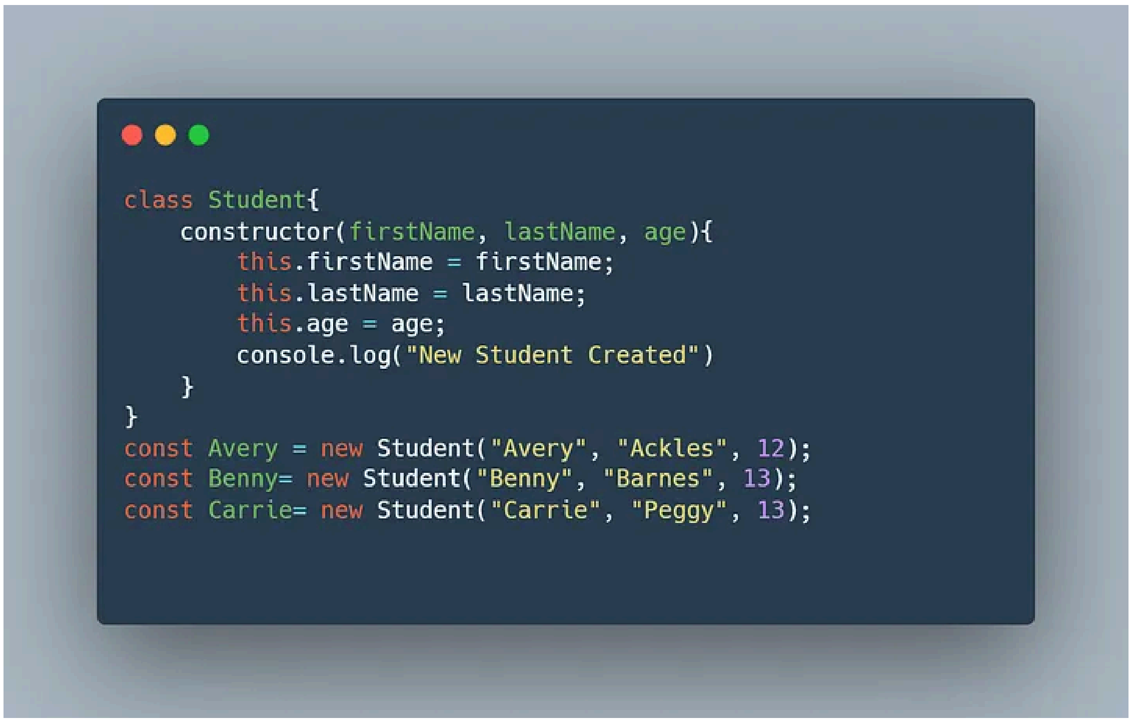


Figure 1: UML Class Diagram example (GeeksforGeeks, n.d.)

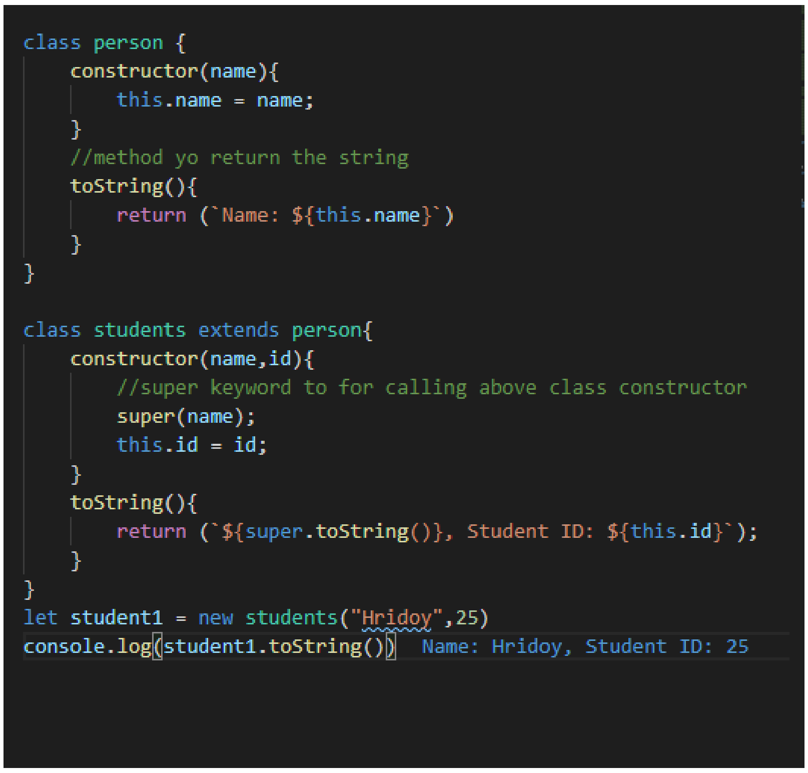
Figure 2 below demonstrates how a Student class consisting of a firstName, lastName, and age is created. Three instances of this class are created, each being a different student with different attributes.



```
class Student{
    constructor(firstName, lastName, age){
        this.firstName = firstName;
        this.lastName = lastName;
        this.age = age;
        console.log("New Student Created")
    }
}
const Avery = new Student("Avery", "Ackles", 12);
const Benny= new Student("Benny", "Barnes", 13);
const Carrie= new Student("Carrie", "Peggy", 13);
```

Figure 2: JavaScript Classes for OOP example (Dondi, 2021)

Figure 3 is an example of OOP in JavaScript, where inheritance is used. A ‘person’ class is created, and another class or object inherits its properties. A ‘students’ object inherits the ‘name’ property from ‘person’ and adds a new property, ‘id’.



```
class person {
    constructor(name){
        this.name = name;
    }
    //method yo return the string
    toString(){
        return (`Name: ${this.name}`)
    }
}

class students extends person{
    constructor(name,id){
        //super keyword to for calling above class constructor
        super(name);
        this.id = id;
    }
    toString(){
        return (`${super.toString()}, Student ID: ${this.id}`);
    }
}
let student1 = new students("Hridoy",25)
console.log(student1.toString())  Name: Hridoy, Student ID: 25
```

Figure 3: Object-Oriented Programming in JavaScript (Francis, 2020)

App Architecture

The project adopts a **Model-View-Controller (MVC)** architecture, a design pattern used in web development to separate concerns:

- **Model**: Manages the data and business logic (e.g, database, data structures). We will be using Mongoose.
- **View**: Handles the user interface and presentation (what the user sees). We will be using React.js for our view.
- **Controller**: Acts as an intermediary between the model and the view, processing user input and updating the model or view as needed. Express.js routes manage API requests and link models to the frontend.

This separation makes the code more organised, maintainable, and scalable.

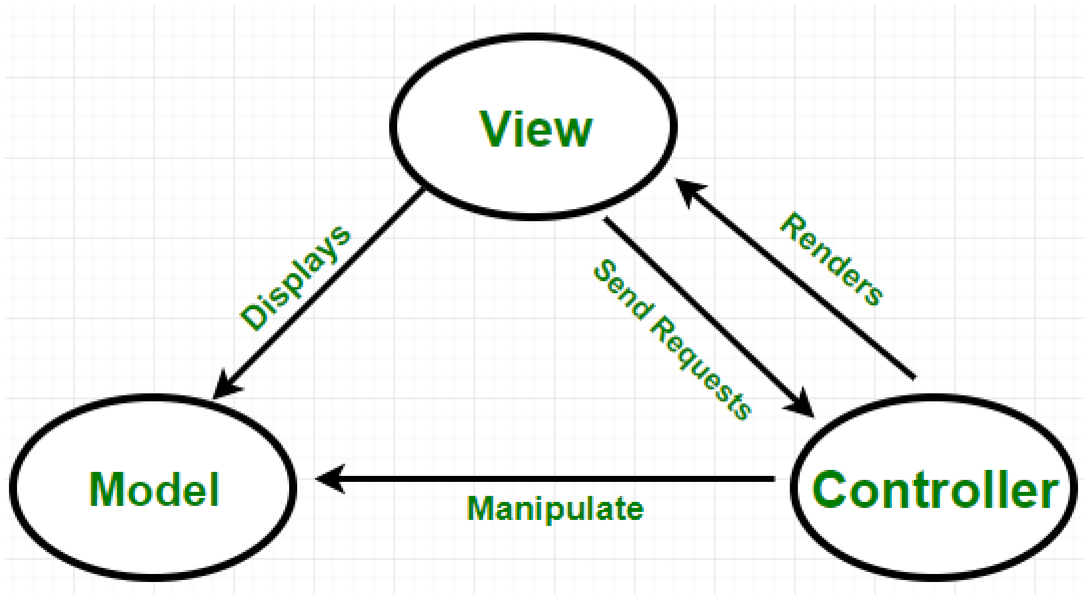


Figure 4. Model-View-Controller Diagram (GeeksforGeeks, n.d)

Additionally, MVC enhances testing and debugging by allowing each component to be developed and tested in isolation, particularly the controller. This modular structure not only supports easier maintenance but also promotes scalability and flexibility. Since components can be reused across different applications, developers can save time and effort when working on projects with similar requirements. Overall, MVC encourages a more structured, manageable, and efficient approach to application development.

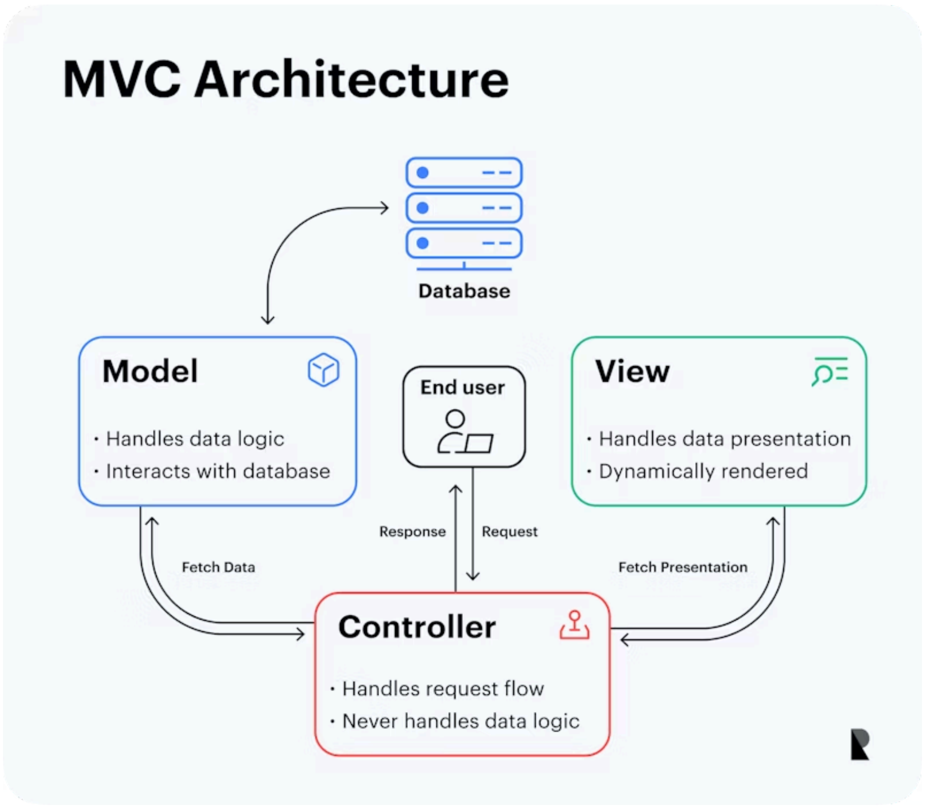
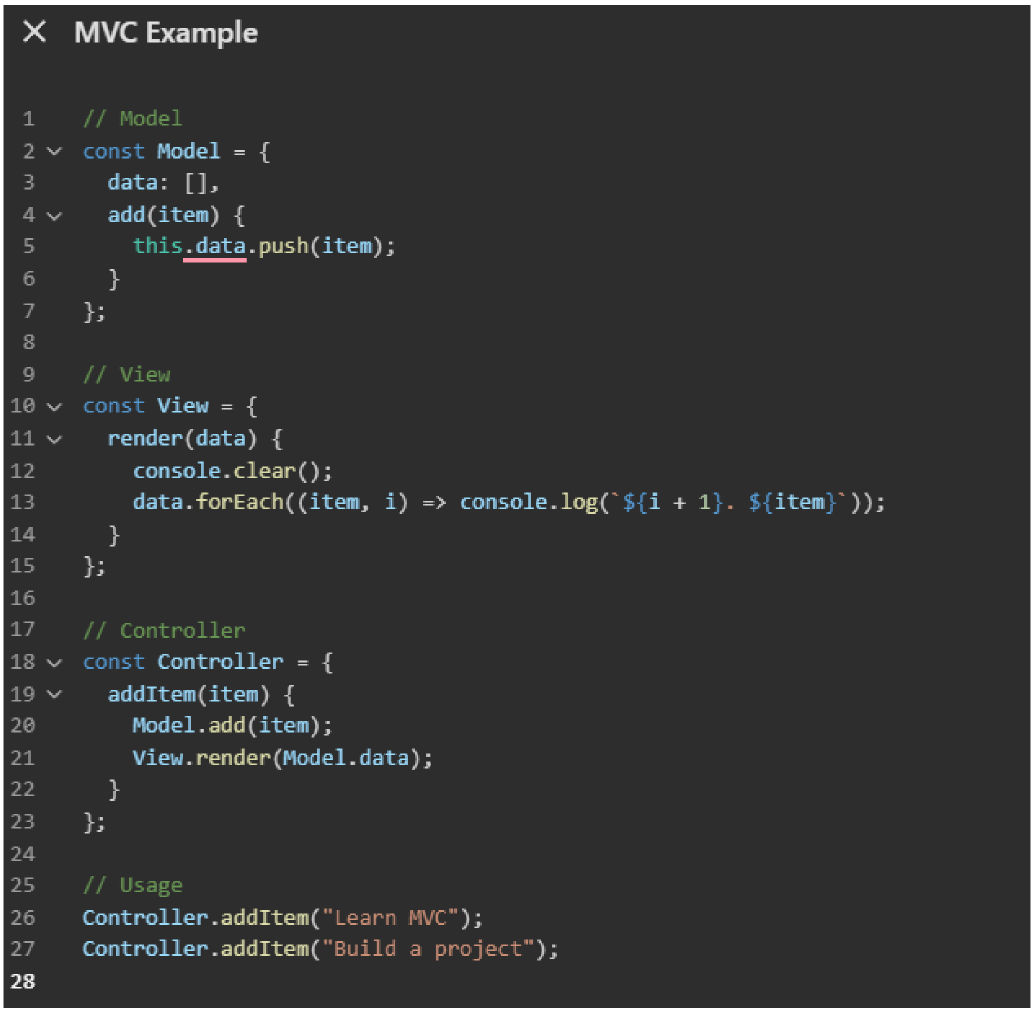


Figure 5. MVC Architecture (Ramotion, 2023)

Figure 6 is a minimal but solid illustration of the MVC pattern in action:

- **Model** stores and manages data.
- **View** handles displaying the data.
- **Controlle**r links the user action (adding items) to updating the **model** and **view.**



```
// Model
const Model = {
  data: [],
  add(item) {
    this.data.push(item);
  }
};

// View
const View = {
  render(data) {
    console.clear();
    data.forEach((item, i) => console.log(`${i + 1}. ${item}`));
  }
};

// Controller
const Controller = {
  addItem(item) {
    Model.add(item);
    View.render(Model.data);
  }
};

// Usage
Controller.addItem("Learn MVC");
Controller.addItem("Build a project");
```

Figure 6. MVC Code Snippet (Williams, 2025)

---

2. Software Development Methodologies

Project Management Methodology

We will adopt a **hybrid Waterfall-Agile methodology**:

- **Initial phases**: Planning and documentation follow **Waterfall**.
- **Development phases**: Coding adopts an **Agile** approach for iterative development and integration.

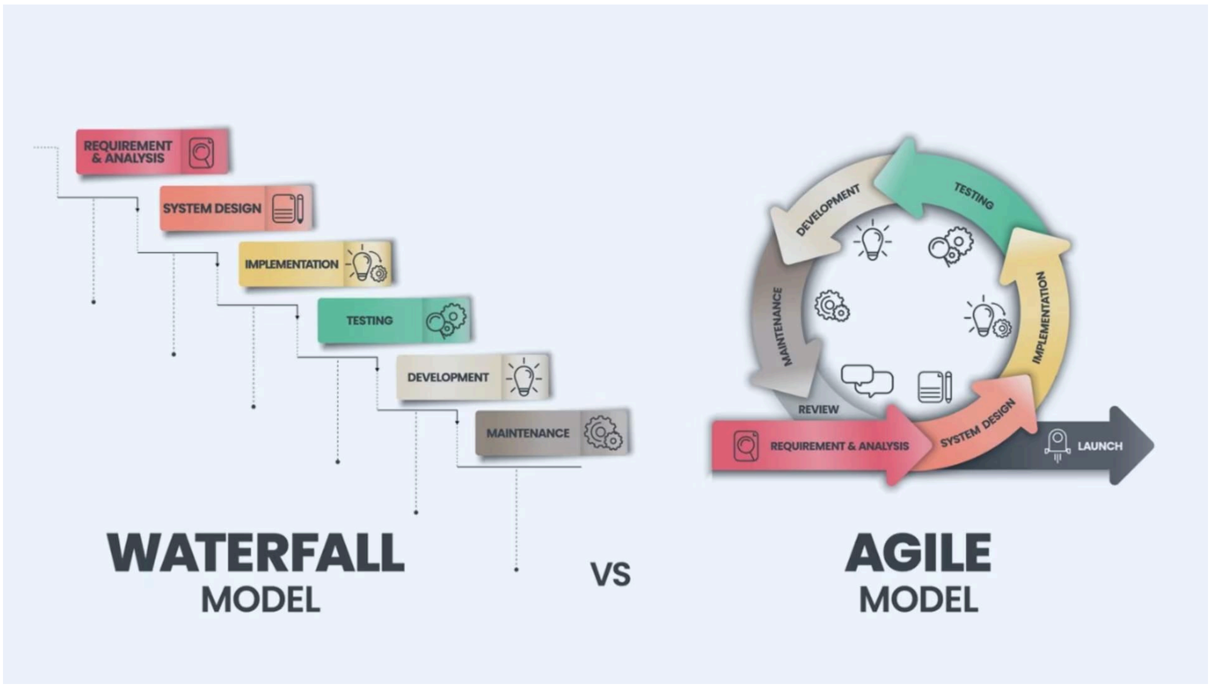


Figure 7: A side-by-side comparison of Waterfall vs Agile (BairesDev Editorial Team, n.d.)

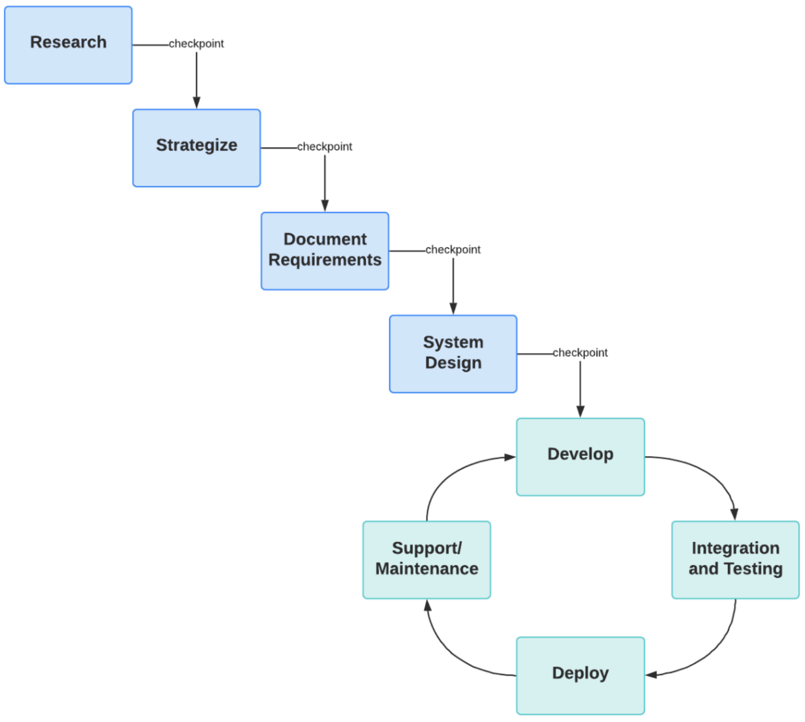


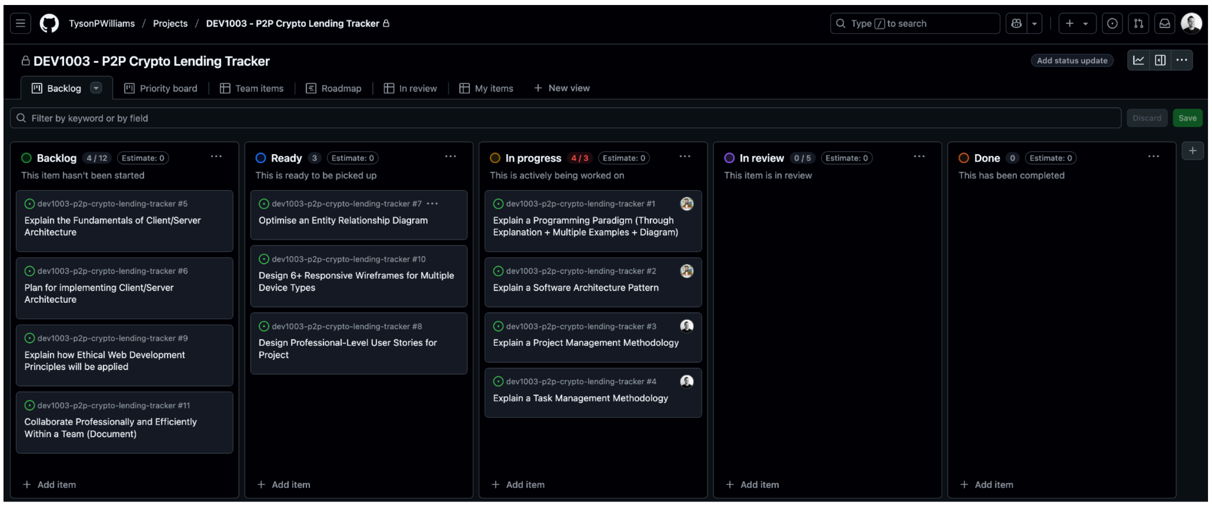
Figure 8: A hybrid Waterfall-Agile approach (Lucidchart, n.d.)

We have chosen a **hybrid methodology** (visualised in Figure 8) to match the natural progression of the project. For the planning and documentation phase, a **Waterfall approach** is ideal, as tasks are largely sequential: we must finalise architecture decisions, user stories, and data models before meaningful development can begin. Each component builds upon the last, making linear execution efficient. However, once we reach the development phase, we will benefit from **Agile practices**, allowing us to work in parallel. With two developers, we can independently build and iterate on different functionalities, using short feedback loops to stay aligned and resolve blockers quickly.

Task Management Methodology

A **dual approach** will be used:

- **Kanban Board**: For task visualisation and progress tracking.
- **Scrum elements**: Short stand-ups during coding phases to address blockers and maintain alignment.



TysonPWilliams / Projects / DEV1003 - P2P Crypto Lending Tracker

DEV1003 - P2P Crypto Lending Tracker

Backlog 4 / 12 Estimate: 0
This item hasn't been started
- dev1003-p2p-crypto-lending-tracker #5
  Explain the Fundamentals of Client/Server Architecture
- dev1003-p2p-crypto-lending-tracker #6
  Plan for implementing Client/Server Architecture
- dev1003-p2p-crypto-lending-tracker #9
  Explain how Ethical Web Development Principles will be applied
- dev1003-p2p-crypto-lending-tracker #11
  Collaborate Professionally and Efficiently Within a Team (Document)

Ready 3 Estimate: 0
This is ready to be picked up
- dev1003-p2p-crypto-lending-tracker #7
  Optimise an Entity Relationship Diagram
- dev1003-p2p-crypto-lending-tracker #10
  Design 6+ Responsive Wireframes for Multiple Device Types
- dev1003-p2p-crypto-lending-tracker #8
  Design Professional-Level User Stories for Project

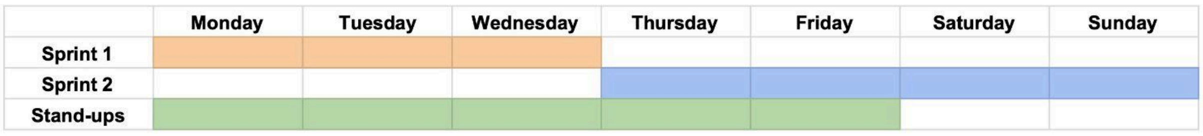
In progress 4 / 3 Estimate: 0
This is actively being worked on
- dev1003-p2p-crypto-lending-tracker #1
  Explain a Programming Paradigm (Through Explanation + Multiple Examples + Diagram)
- dev1003-p2p-crypto-lending-tracker #2
  Explain a Software Architecture Pattern
- dev1003-p2p-crypto-lending-tracker #3
  Explain a Project Management Methodology
- dev1003-p2p-crypto-lending-tracker #4
  Explain a Task Management Methodology

In review 0 / 5 Estimate: 0
This item is in review

Done 0 Estimate: 0
This has been completed

Figure 9: A screenshot from this assignment's GitHub project kanban board (Authors, 2025)

To manage tasks effectively throughout the project, we are adopting a two-pronged approach that combines **Kanban** and **Scrum** methodologies. The Kanban board, hosted via GitHub Projects (see Figure 9), provides a visual overview of task progress using clearly defined columns: *Backlog*, *In Progress*, *Review*, and *Complete*. This structure helps us prioritise tasks, track ownership, and maintain momentum.



	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
Sprint 1	🟧	🟧	🟧				
Sprint 2				🟦	🟦	🟦	🟦
Stand-ups	🟩	🟩	🟩	🟩	🟩		

Figure 10: An example of a week in the development cycle (Authors, 2025)

Alongside Kanban, we will integrate **Scrum elements** during the development phases. Specifically, we plan to run **bi-weekly sprints**, with short daily **stand-up meetings** between team members to discuss progress, surface blockers, and reassign tasks if needed (see

Figure 10). This combination ensures we stay responsive to project needs while maintaining clarity and accountability across the workload.

The flexibility of Scrum supports our Agile development cycles, while the structure of Kanban keeps our planning grounded and transparent.

3. Client-Server Architecture

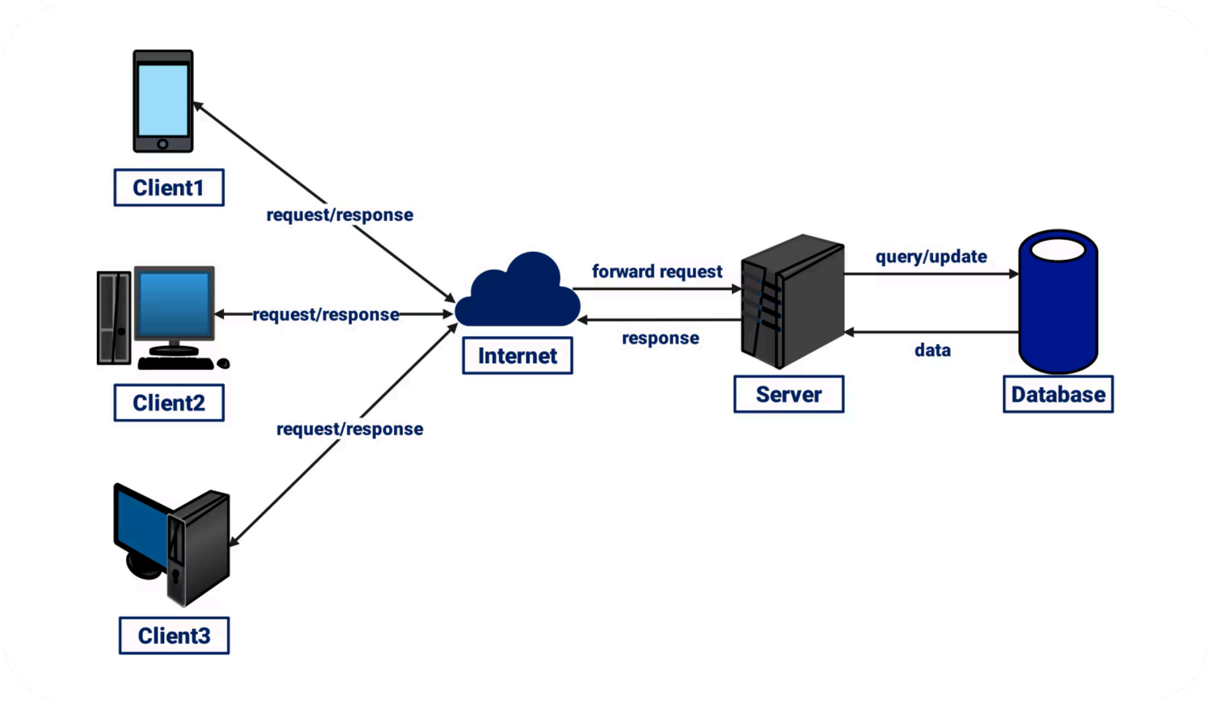


Figure 10. Network Diagram for Client-Server (EdrawMax, n.d)

Fundamentals of Client-Server Architecture

Client/Server Communication: In web applications, the client (e.g., a web browser) sends requests to the server using protocols like HTTP or HTTPS. The server processes these requests and returns responses, facilitating data exchange and functionality. (IOSR Journal of Computer Engineering, 2022)

Data Distribution: Data distribution involves the allocation of data across different servers or locations to enhance performance, scalability, and reliability. Strategies include centralized, decentralized, and distributed systems, each with its trade-offs. (Ceglar, Rak and Jezic, 2007)

Feature Distribution: Feature distribution pertains to assigning application functionalities between the client and server. Client-side features (e.g., input validation) enhance responsiveness, while server-side features (e.g., database operations) ensure data integrity and security. (Farrell, 2020)

Authorization: Authorization determines user permissions for accessing resources. Implementations often use Role-Based Access Control (RBAC) and tokens like JSON Web Tokens (JWT) to manage and verify permissions securely. (Pardede, Nayak and Sain, 2022)

Validation: Validation ensures that input data meets predefined formats and constraints, preventing errors and security vulnerabilities. It is performed both on the client side for immediate feedback and on the server side for security enforcement. (Sule, Haruna and Dogo, 2014)

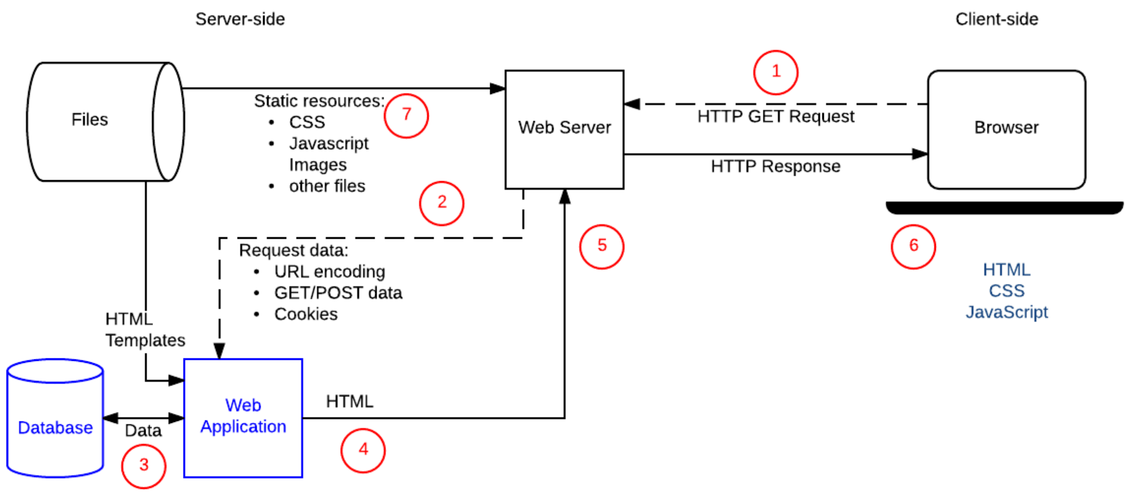


Figure 11. Client-server overview (MDN Web Docs, n.d.)

The diagram above illustrates the dynamic HTTP request-response cycle, showing how a client sends a request to the server, which processes it and returns a response, forming the core of client-server communication

Plan for Implementing Client/Server Architecture in Our Project

Tech Stack: JavaScript, Node.js, Express, MongoDB, Mongoose, React.js, HTTP

1. Client/Server Communication

Client-side (React) communicates with the server (Express) via RESTful HTTP requests using axios or fetch. JSON is used for all data exchange. CORS and HTTPS are configured for secure and cross-origin communication. WebSocket/Socke.IO may be introduced for real-time transaction updates.

2. Data Distribution

MongoDB stores wallets, transaction logs, and user data. Mongoose manages schemas and relationships. Data is structured for performance using indexing. Future scalability may include replication and sharding.

3. Data Security

All traffic uses HTTPS. Passwords are hashed (bcrypt). JWT is used for secure, stateless user authentication. API endpoints are protected with rate limiting, input validation, and security middleware (e.g., Helmet, CORS policies).

4. Feature Distribution

- **Client:** UI rendering, API consumption, local validation, token storage.
- **Server:** Business logic, DB operations, authentication, input validation. Environment variables control config for production and development environments.

5. Authorisation

JWT handles token-based user sessions. Protected routes verify tokens via middleware. Role-based access control restricts sensitive operations. Tokens are stored in HTTP-only cookies or memory.

6. Validation

- **Client:** Uses React validation tools or custom functions.
- **Server:** Validates via Mongoose schemas and custom middleware. Invalid requests return structured errors; logs are maintained for monitoring.

---

4. Entity Relationship Diagram (ERD)

Key Entities:

- Users
- LoanRequests
- InterestTerms
- Cryptocurrencies
- Deals
- Collateral
- Transactions

The database schema for our P2P Bitcoin lending platform has been normalised to **Third Normal Form (3NF)**. Each table includes a single-column primary key, and all non-key attributes fully depend on that key, satisfying **1NF** and **2NF**. No transitive dependencies exist between non-key attributes, meeting the requirements of **3NF**. Foreign keys are used appropriately to model relationships between entities, ensuring data integrity and minimising redundancy.

For example, LoanRequests references Users, Cryptocurrencies, and InterestTerms, while Deals links to both borrowers and lenders via their respective user_id. The Collateral, Transaction, and Repayment data have been separated into dedicated tables, which further supports modularity and maintainability of the system's data model.

Please refer to **Appendix A - Entity Relationship Diagram** for the full diagram.

---

5. User Stories

Persona 1: The Borrower – "Alex"

- **Need:** Access fast, trustless loans without involving traditional financial institutions.
- **User Story:** *As a borrower, I want to securely request a Bitcoin loan using my wallet, so that I can access funds without relying on banks or credit checks.*
- **Want/Feature:**
  - Wallet-based authentication (e.g., MetaMask)
  - Smart contract-backed loan request and repayment process
- **Justification:** Enables financial inclusivity and autonomy through decentralised borrowing, reducing dependency on banks.

Persona 2: The Lender – "Jordan"

- **Need:** Grow their Bitcoin holdings by passively earning interest through lending.
- **User Story:** *As a lender, I want to view and select loan requests to fund based on risk and return, so I can earn yield from my Bitcoin.*
- **Want/Feature:**
  - Dashboard to view loan offers with borrower credit scores or risk indicators
  - Auto-lend feature for small, diversified loans
- **Justification:** Empowers lenders to use idle assets for passive income, incentivising liquidity on the platform.

Persona 3: The Admin – "Riley"

- **Need:** Maintain platform integrity and prevent abuse or fraud.
- **User Story:** *As an admin, I want to oversee transactions and user behavior via an admin panel, so that I can ensure platform stability and compliance.*
- **Want/Feature:**

    - Admin dashboard for transaction logs and user reports
    - Ability to suspend accounts or flag suspicious activity
- **Justification:** Critical for trust, compliance, and maintaining a secure ecosystem for users.

Consolidated Wants/Functionalities Summary

1. **Wallet-Based Authentication** – For secure, decentralised access.
2. **Loan Request/Review System** – Supports both borrower and lender experiences.
3. **Smart Contract Automation** – Ensures trustless, enforceable loan agreements.
4. **Admin Monitoring Tools** – Maintain security, trust, and compliance.

---

6. Ethical Web Development Principles

Our project will adhere to the following principles from ethicalweb.org (Scott, 2016):

- **Accessibility**: Usable by people with disabilities.
- **Privacy**: Minimal data collection, wallet-based login.
- **Security**: Secure encryption for transactions.
- **Transparency**: Clear terms and user rights.

Our project is guided by ethical web development principles prioritising accessibility, privacy, security, and transparency. To ensure inclusivity, we will follow the *Web Content Accessibility Guidelines (WCAG) 2.1* by using semantic HTML and accessibility-focused tools such as eslint-plugin-jsx-a11y. This supports screen readers and keyboard navigation for users with disabilities.

While we intend to implement a wallet-based login system in the future, aligned with decentralised finance (DeFi) values and user privacy, for this assignment's minimum viable product (MVP), user accounts will require personal information such as email and passwords. Passwords will be securely hashed using bcrypt before storage.

Security is a key focus, with all client-server communication protected by HTTPS. We will also follow best practices from the *Open Worldwide Application Security Project (OWASP, 2025)*, mitigating risks such as cross-site scripting (XSS) and cross-site request forgery (CSRF). Tools like Helmet.js will secure HTTP headers, and Zxcvbn.js may be integrated to support password strength validation in future iterations.

Finally, transparency will be maintained by clearly communicating all loan terms, platform features, and user rights within the application's interface, ensuring users understand how their data and interactions are handled.

---

7. Wireframes

Wireframes illustrate the layout and page relationships for the following views:

- Landing page
- Log in/sign up page
- User dashboard
- Admin dashboard
- Browse loans
- Request a loan
- Loan terms
- Transactions

Each is presented in three formats:

- Desktop
- Tablet
- Mobile

Please refer to Appendix B - Wireframes for diagrams.

References

BairesDev Editorial Team (n.d.) *Agile vs. Waterfall: Choosing the Right Project Management Style for Your Team*. Available at: https://www.bairesdev.com/blog/agile-vs-waterfall/ (Accessed: 1 May 2025).

Ceglar, A., Rak, M. and Jezic, G. (2007) ‘Effective data distribution and reallocation strategies for fast query processing’, in Matoušek, R. (ed.) *Proceedings of the 16th International Conference on Information Systems Development*. Berlin: Springer. Available at: https://link.springer.com/chapter/10.1007/978-3-540-78849-2_55 (Accessed: 4 May 2025).

Dondi, J. (2021) *How to use JavaScript Classes for Object-Oriented Programming*. Available at: https://javascript.plainenglish.io/how-to-use-javascript-classes-for-object-oriented-programming-8f300d5f71a5 (Accessed: 2 May 2025).

EdrawMax (n.d.) *Network Diagram for Client-Server*. Available at: https://edrawmax.wondershare.com/templates/network-diagram-for-client-server.html (Accessed: 2 May 2025).

Farrell, B. (2020) ‘Web application architecture’, in *Practical Node.js: Building Real-World Scalable Web Apps*. 2nd edn. Apress. Available at: https://link.springer.com/chapter/10.1007/978-1-4842-8142-0_2 (Accessed: 4 May 2025).

Francis, A. (2020) *Intro to Object-Oriented Programming in JavaScript*. Medium. Available at: https://medium.com/swlh/intro-to-object-oriented-programming-in-javascript-fe90c70ab316 (Accessed: 29 April 2025).

GeeksforGeeks (n.d.) *Model-View-Controller (MVC) Architecture for Node Applications*. Available at: https://www.geeksforgeeks.org/model-view-controllermvc-architecture-for-node-applications/ (Accessed: 2 May 2025).

GeeksforGeeks (n.d.) *Unified Modeling Language (UML) Class Diagrams*. Available at: https://www.geeksforgeeks.org/unified-modeling-language-uml-class-diagrams/ (Accessed: 29 April 2025).

IOSR Journal of Computer Engineering (2022) *A Study on the Client Server Architecture and Its Usability*. Available at: https://www.iosrjournals.org/iosr-jce/papers/Vol24-issue4/Ser-1/G2404017376.pdf (Accessed: 4 May 2025).

Lucidchart (n.d.) *Agile-Waterfall Hybrid: Is It Right for Your Team?* Available at: https://www.lucidchart.com/blog/is-agile-waterfall-hybrid-right-for-your-team (Accessed: 1 May 2025).

MDN Web Docs (n.d.) *Client-server overview*. Available at: https://developer.mozilla.org/en-US/docs/Learn_web_development/Extensions/Server-side/First_steps/Client-Server_overview (Accessed: 4 May 2025).

OWASP Foundation (2025) *OWASP: Open Worldwide Application Security Project*. Available at: https://owasp.org/ (Accessed: 7 May 2025).

Pardede, E., Nayak, R. and Sain, S.L. (2022) 'Enhancing JWT authentication and authorization in web applications', *Computers*, 12(4), p.78. Available at: https://www.mdpi.com/2073-431X/12/4/78 (Accessed: 4 May 2025).

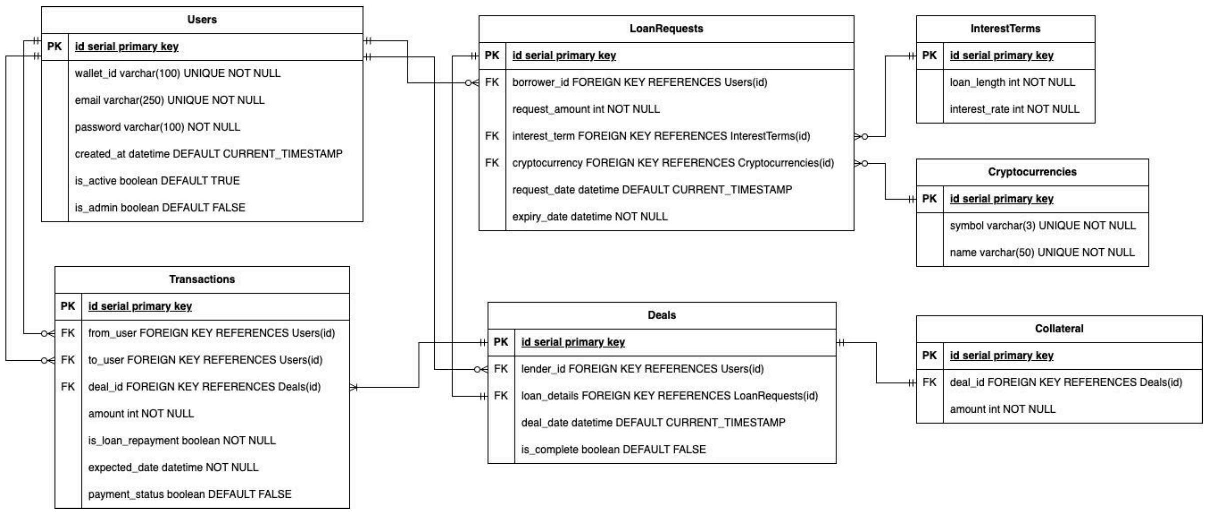
Ramotion (2023) *MVC Architecture: Simplifying Web Application Development*. Available at: https://www.ramotion.com/blog/mvc-architecture-in-web-application/ (Accessed: 2 May 2025).

Scott, A. (2016) *Ethical Web Development*. Available at: https://www.ethicalweb.org/ (Accessed: 1 May 2025).

Sule, A.S., Haruna, A. and Dogo, E.M. (2014) 'Input validation vulnerabilities in web applications', *Journal of Software Engineering*, 8(2), pp.116–126. Available at: https://scialert.net/fulltext/?doi=jse.2014.116.126 (Accessed: 4 May 2025).

World Wide Web Consortium (W3C) (2018) *Web Content Accessibility Guidelines (WCAG) 2.1*. W3C Recommendation, 5 June. Available at: https://www.w3.org/TR/WCAG21/ (Accessed: 1 May 2025).

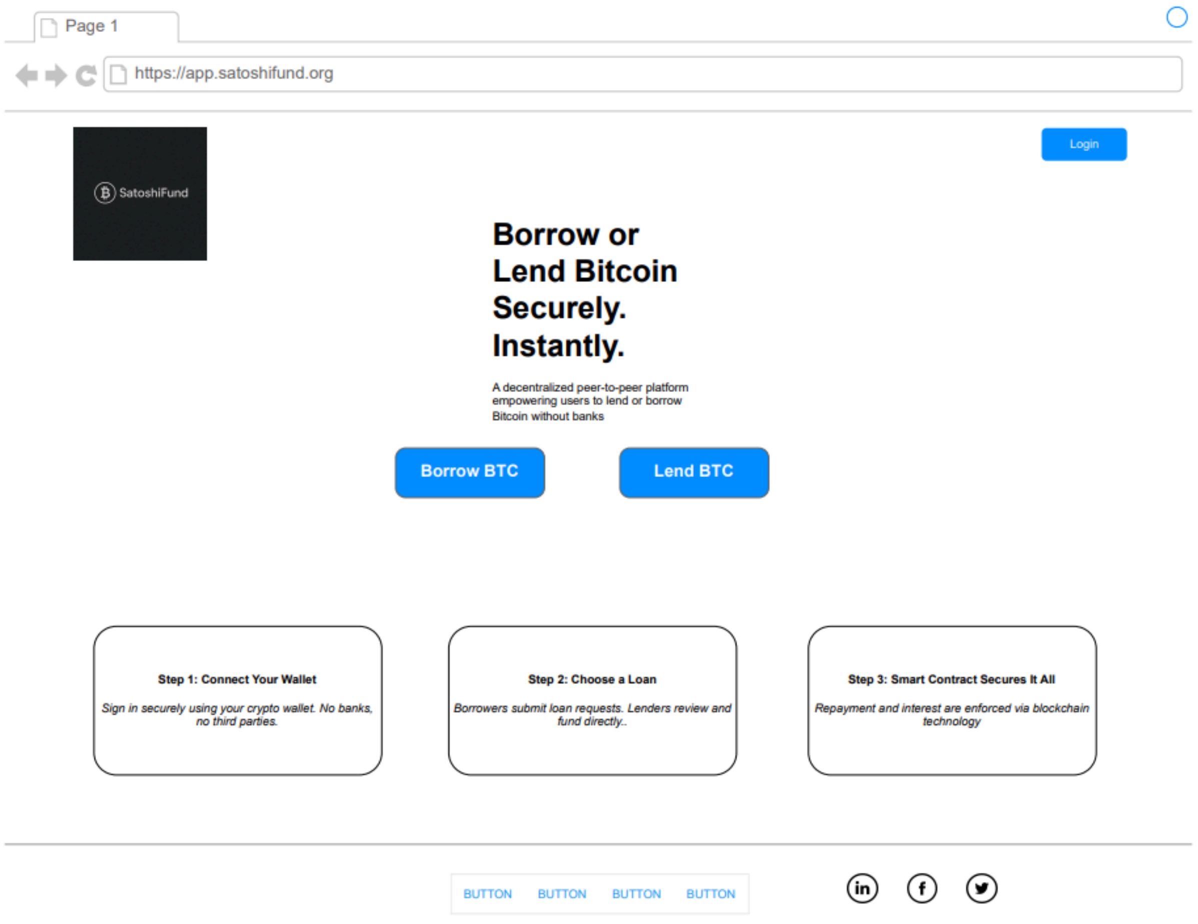
Appendix A - Entity Relationship Diagram (ERD)


Users
PK id serial primary key
wallet_id varchar(100) UNIQUE NOT NULL
email varchar(250) UNIQUE NOT NULL
password varchar(100) NOT NULL
created_at datetime DEFAULT CURRENT_TIMESTAMP
is_active boolean DEFAULT TRUE
is_admin boolean DEFAULT FALSE
LoanRequests
PK id serial primary key
FK borrower_id FOREIGN KEY REFERENCES Users(id)
request_amount int NOT NULL
FK interest_term FOREIGN KEY REFERENCES InterestTerms(id)
FK cryptocurrency FOREIGN KEY REFERENCES Cryptocurrencies(id)
request_date datetime DEFAULT CURRENT_TIMESTAMP
expiry_date datetime NOT NULL
InterestTerms
PK id serial primary key
loan_length int NOT NULL
interest_rate int NOT NULL
Cryptocurrencies
PK id serial primary key
symbol varchar(3) UNIQUE NOT NULL
name varchar(50) UNIQUE NOT NULL
Transactions
PK id serial primary key
FK from_user FOREIGN KEY REFERENCES Users(id)
FK to_user FOREIGN KEY REFERENCES Users(id)
FK deal_id FOREIGN KEY REFERENCES Deals(id)
amount int NOT NULL
is_loan_repayment boolean NOT NULL
expected_date datetime NOT NULL
payment_status boolean DEFAULT FALSE
Deals
PK id serial primary key
FK lender_id FOREIGN KEY REFERENCES Users(id)
FK loan_details FOREIGN KEY REFERENCES LoanRequests(id)
deal_date datetime DEFAULT CURRENT_TIMESTAMP
is_complete boolean DEFAULT FALSE
Collateral
PK id serial primary key
FK deal_id FOREIGN KEY REFERENCES Deals(id)
amount int NOT NULL

Appendix B - Wireframes

Landing Page

Desktop



Borrow or Lend Bitcoin Securely. Instantly.

A decentralized peer-to-peer platform empowering users to lend or borrow Bitcoin without banks

Step 1: Connect Your Wallet

Sign in securely using your crypto wallet. No banks, no third parties.

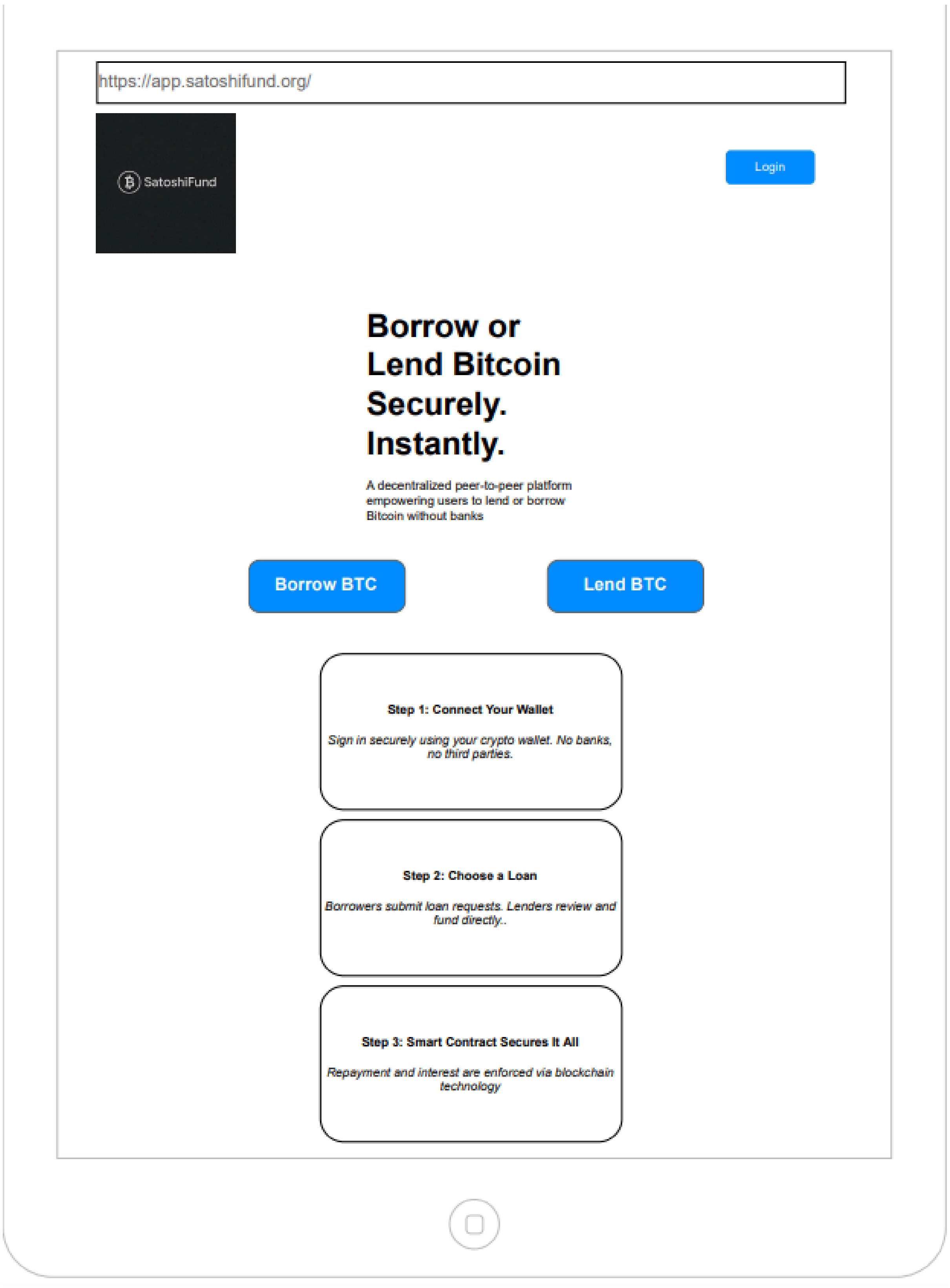
Step 2: Choose a Loan

Borrowers submit loan requests. Lenders review and fund directly..

Step 3: Smart Contract Secures It All

Repayment and interest are enforced via blockchain technology

Tablet



https://app.satoshifund.org/

SatoshiFund

Login

Borrow or Lend Bitcoin Securely. Instantly.

A decentralized peer-to-peer platform empowering users to lend or borrow Bitcoin without banks

Borrow BTC

Lend BTC

Step 1: Connect Your Wallet

Sign in securely using your crypto wallet. No banks, no third parties.

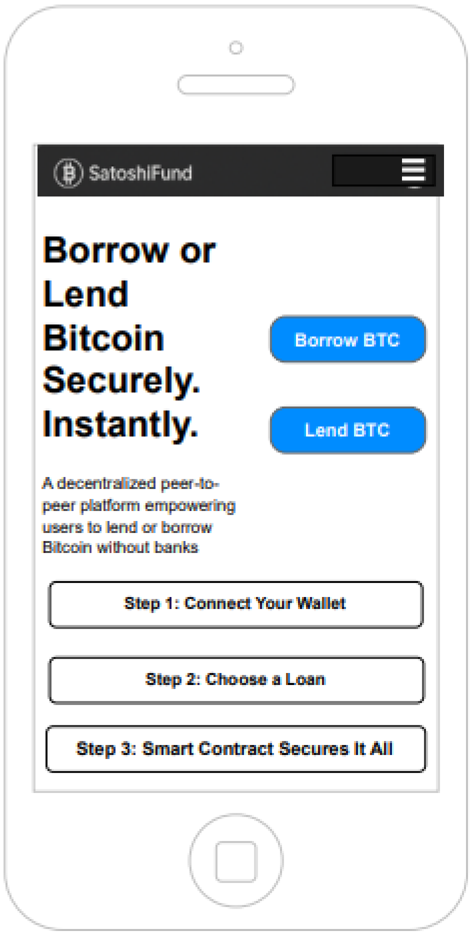
Step 2: Choose a Loan

Borrowers submit loan requests. Lenders review and fund directly..

Step 3: Smart Contract Secures It All

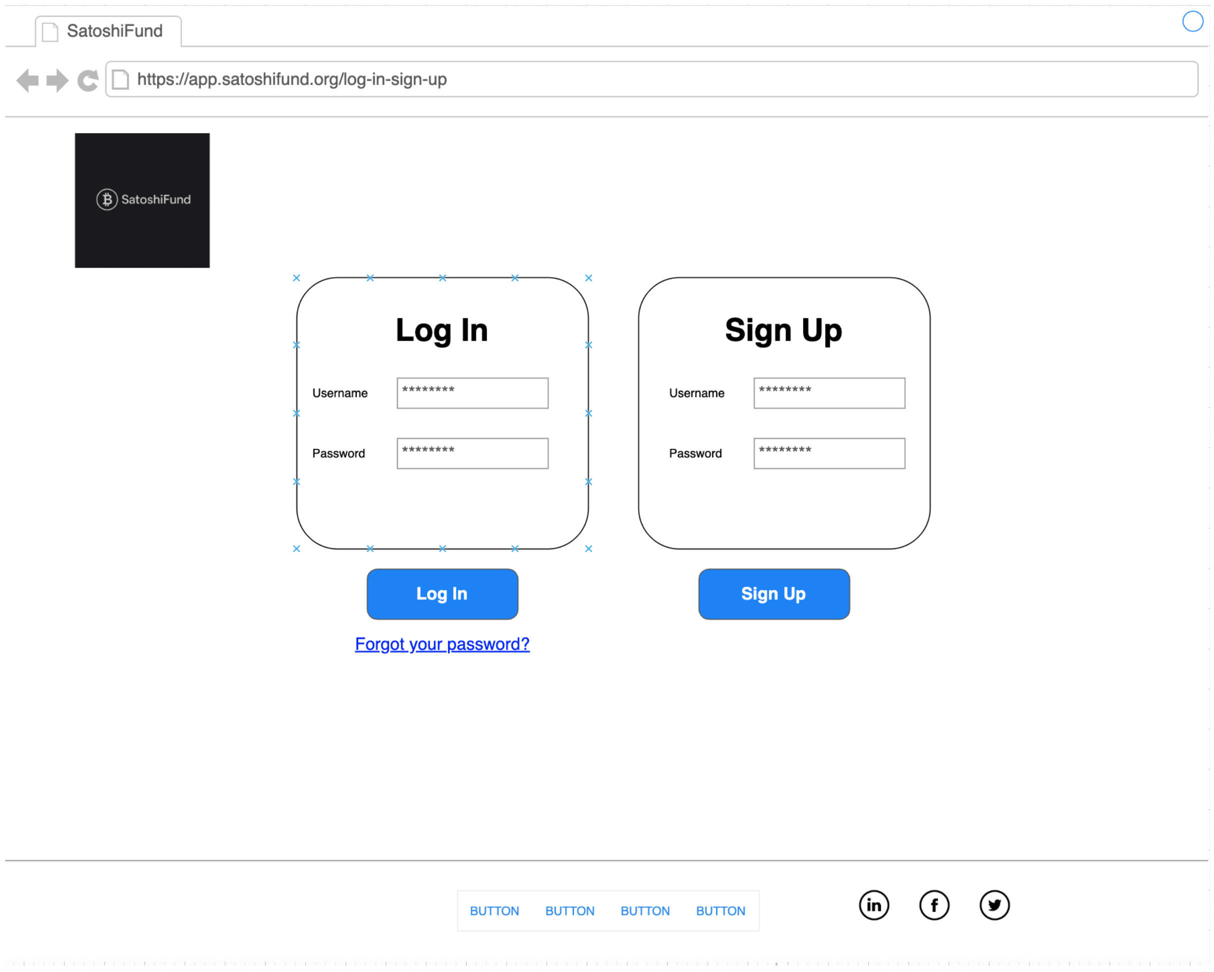
Repayment and interest are enforced via blockchain technology

Mobile

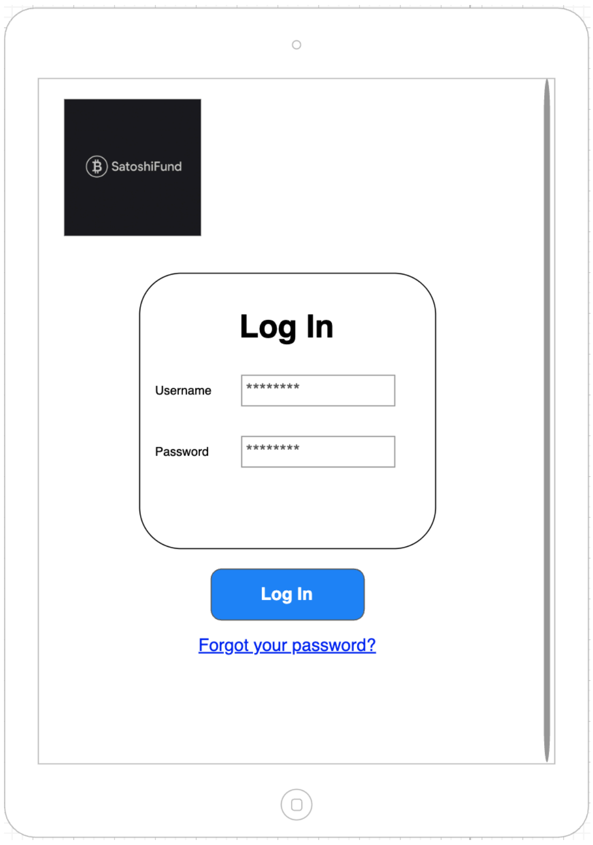

SatoshiFund
Borrow or Lend Bitcoin Securely. Instantly.
Borrow BTC
Lend BTC
A decentralized peer-to-peer platform empowering users to lend or borrow Bitcoin without banks
Step 1: Connect Your Wallet
Step 2: Choose a Loan
Step 3: Smart Contract Secures It All

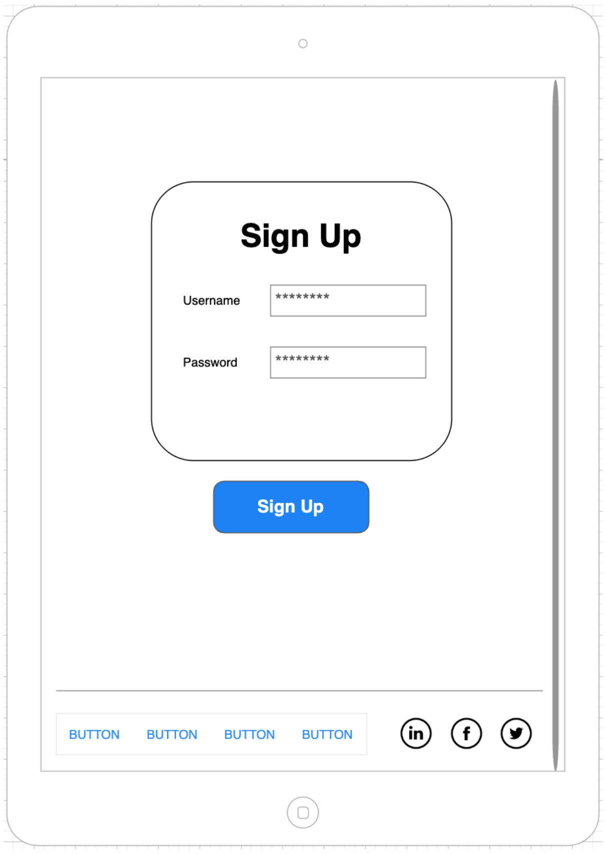
Login/Register

Desktop

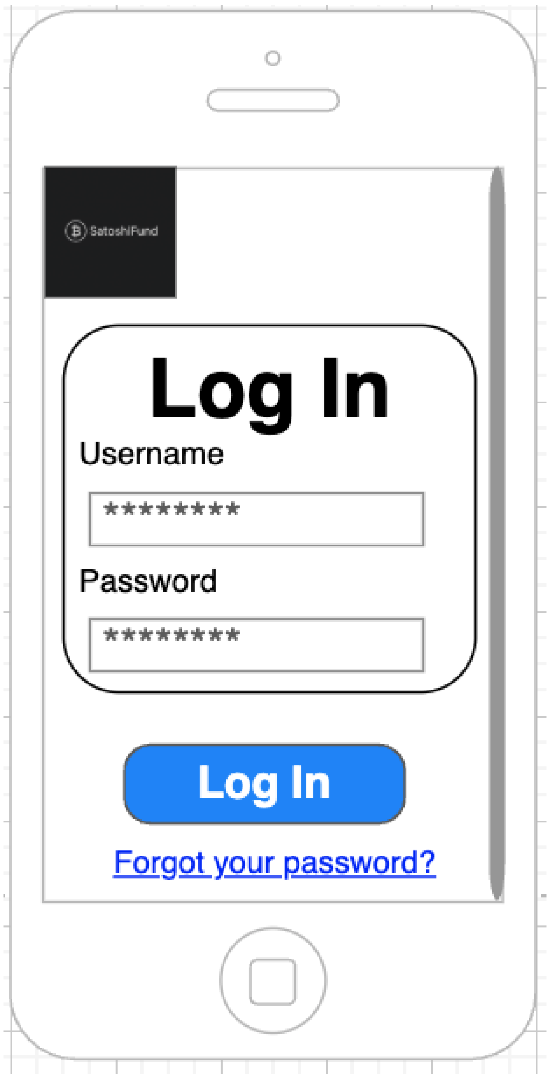


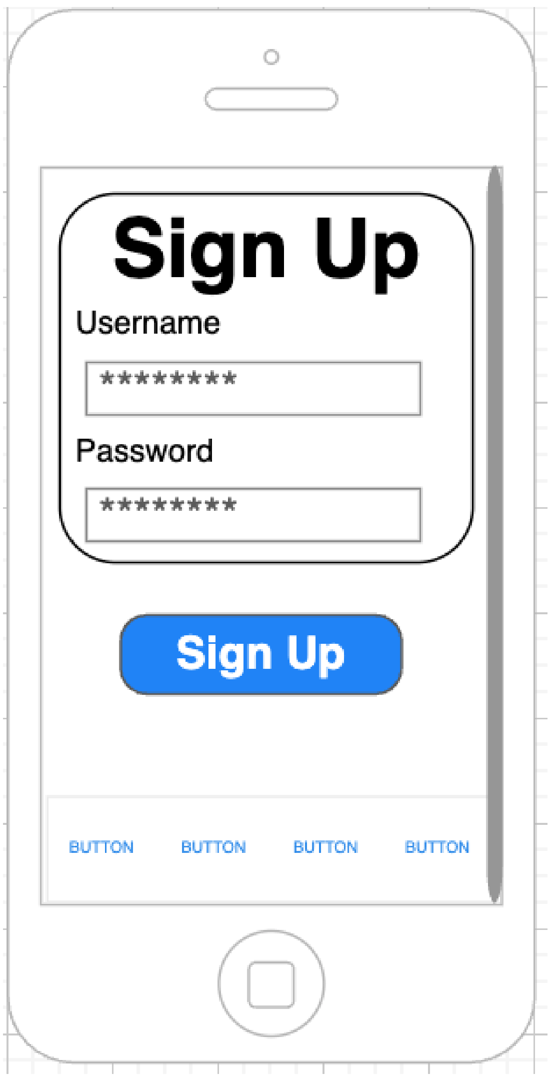
Tablet


SatoshiFund
Log In
Username
****
Password
****
Log In
Forgot your password?


Sign Up
Username
****
Password
****
Sign Up
BUTTON
BUTTON
BUTTON
BUTTON

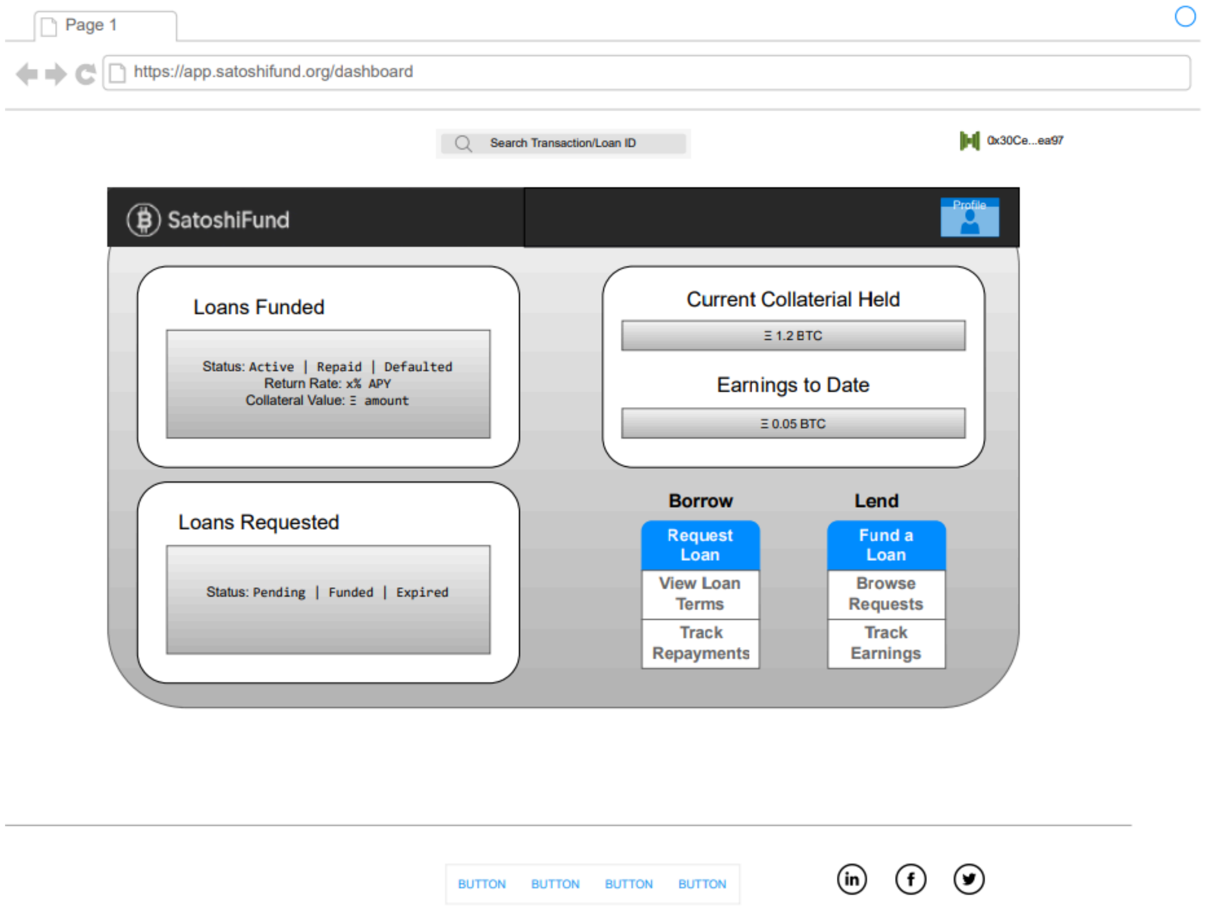
Mobile


SatoshiFund
Log In
Username
****
Password
****
Log In
Forgot your password?

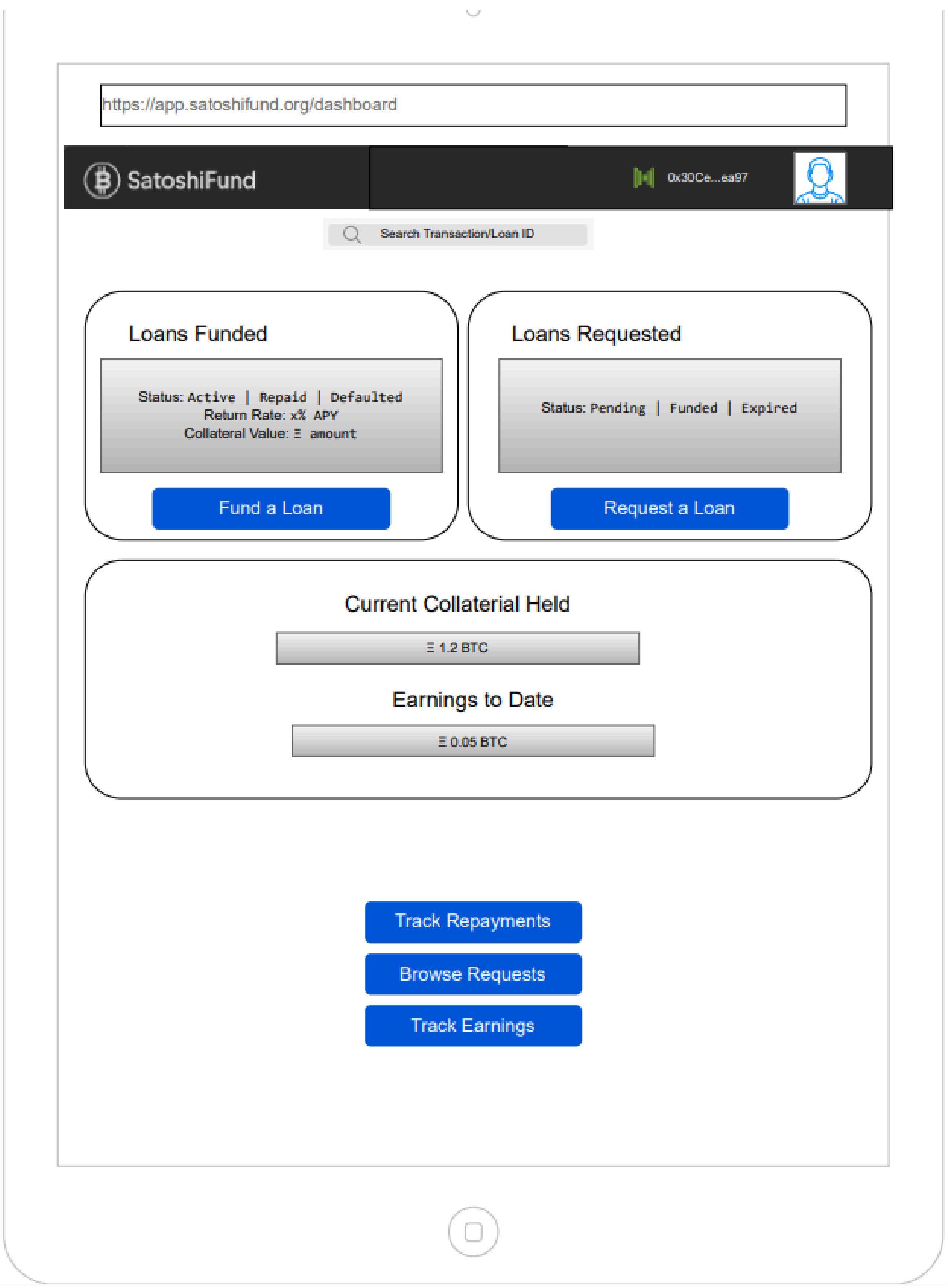

Sign Up
Username
****
Password
****
Sign Up
BUTTON
BUTTON
BUTTON
BUTTON

Dashboard (User)

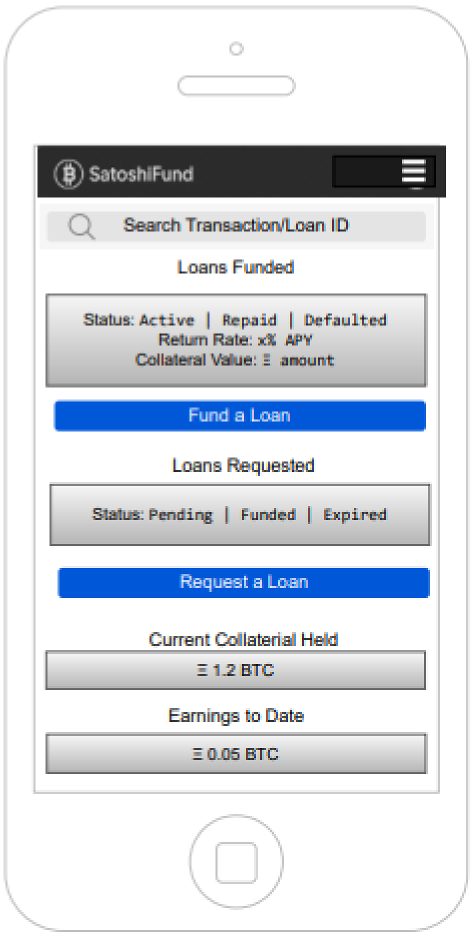
Desktop



Tablet

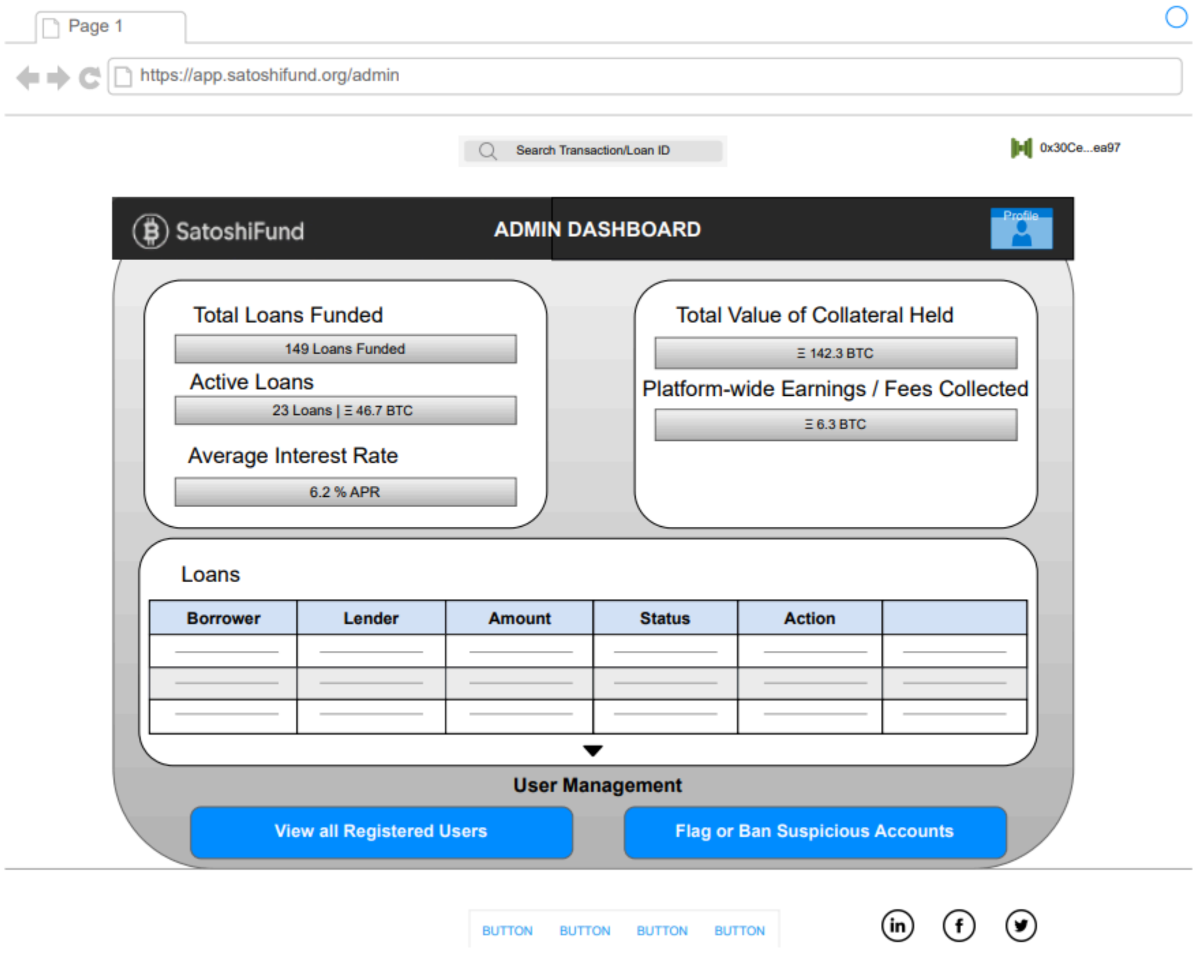

https://app.satoshifund.org/dashboard
SatoshiFund
0x30Ce...ea97
Search Transaction/Loan ID
Loans Funded
Status: Active | Repaid | Defaulted
Return Rate: x% APY
Collateral Value: Ξ amount
Fund a Loan
Loans Requested
Status: Pending | Funded | Expired
Request a Loan
Current Collateral Held
Ξ 1.2 BTC
Earnings to Date
Ξ 0.05 BTC
Track Repayments
Browse Requests
Track Earnings

Mobile


SatoshiFund
Search Transaction/Loan ID
Loans Funded
Status: Active | Repaid | Defaulted
Return Rate: x% APY
Collateral Value: Ξ amount
Fund a Loan
Loans Requested
Status: Pending | Funded | Expired
Request a Loan
Current Collaterial Held
Ξ 1.2 BTC
Earnings to Date
Ξ 0.05 BTC

Dashboard (Admin)

Desktop


Page 1
https://app.satoshifund.org/admin
Search Transaction/Loan ID
0x30Ce...ea97
SatoshiFund
ADMIN DASHBOARD
Profile

Total Loans Funded
149 Loans Funded
Active Loans
23 Loans | Ξ 46.7 BTC
Average Interest Rate
6.2 % APR

Total Value of Collateral Held
Ξ 142.3 BTC
Platform-wide Earnings / Fees Collected
Ξ 6.3 BTC

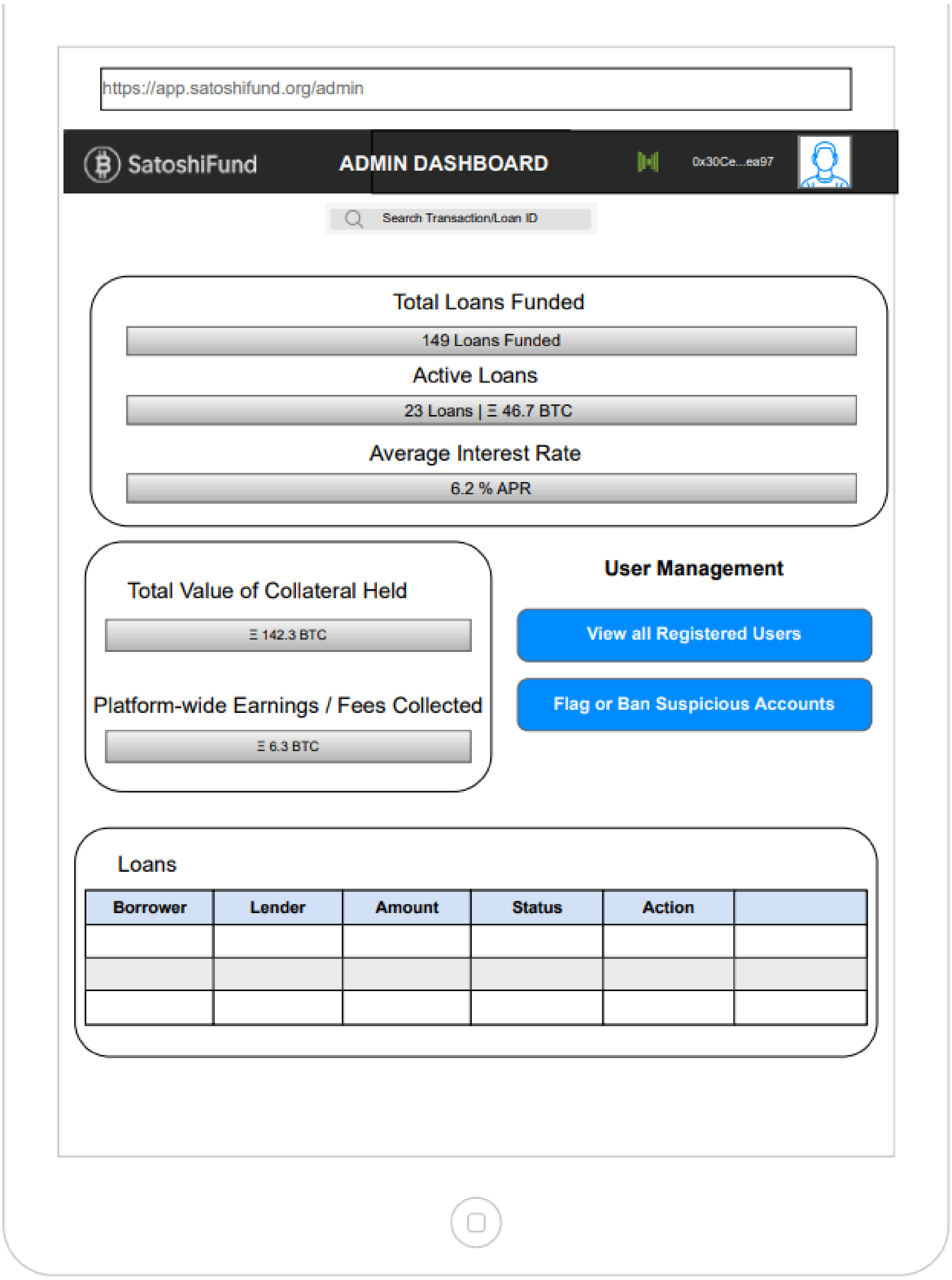
Loans

Borrower	Lender	Amount	Status	Action	

User Management

View all Registered Users
Flag or Ban Suspicious Accounts
BUTTON
BUTTON
BUTTON
BUTTON

Tablet



https://app.satoshifund.org/admin

SatoshiFund ADMIN DASHBOARD 0x30Ce...ea97

Search Transaction/Loan ID

Total Loans Funded

149 Loans Funded

Active Loans

23 Loans | Ξ 46.7 BTC

Average Interest Rate

6.2 % APR

Total Value of Collateral Held

Ξ 142.3 BTC

Platform-wide Earnings / Fees Collected

Ξ 6.3 BTC

User Management

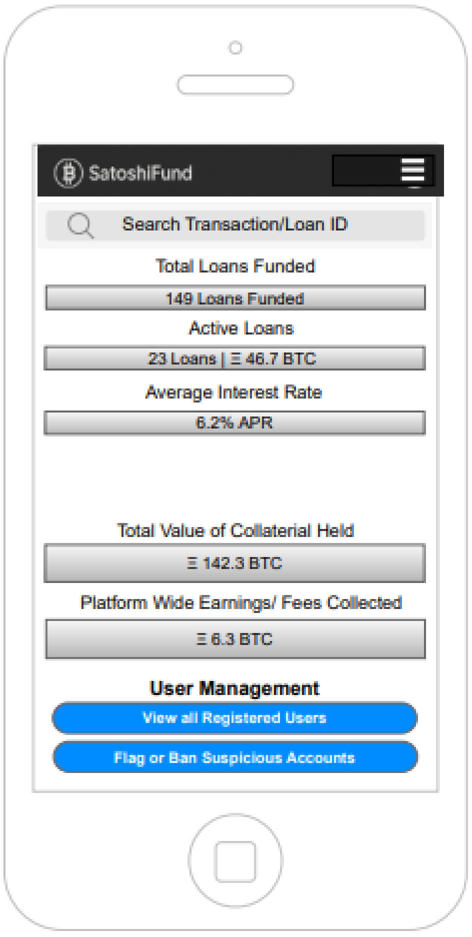
View all Registered Users

Flag or Ban Suspicious Accounts

Loans

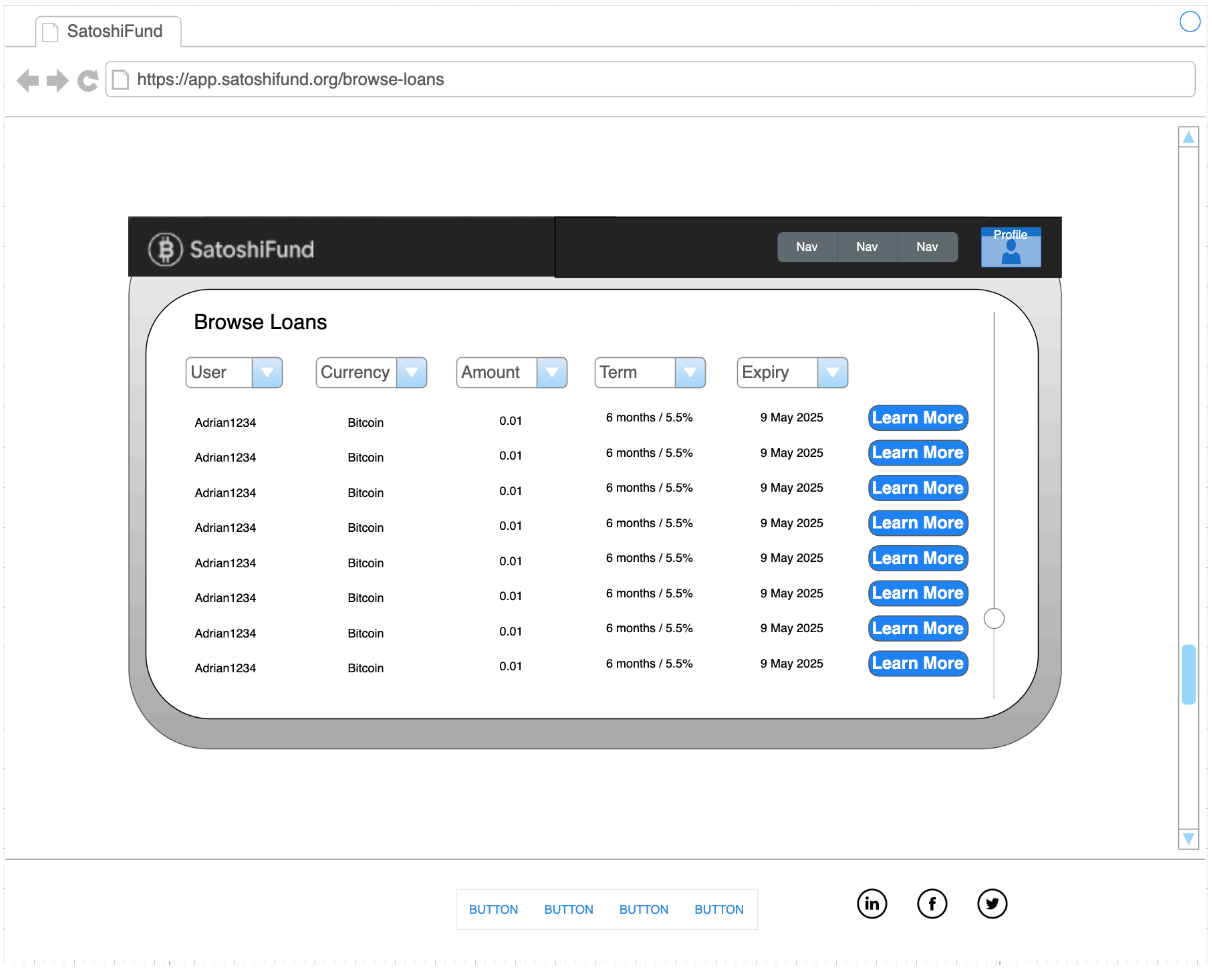
Borrower	Lender	Amount	Status	Action	

Mobile


SatoshiFund
Search Transaction/Loan ID
Total Loans Funded
149 Loans Funded
Active Loans
23 Loans | Ξ 46.7 BTC
Average Interest Rate
6.2% APR
Total Value of Collateral Held
Ξ 142.3 BTC
Platform Wide Earnings/ Fees Collected
Ξ 6.3 BTC
User Management
View all Registered Users
Flag or Ban Suspicious Accounts

Browse Loans

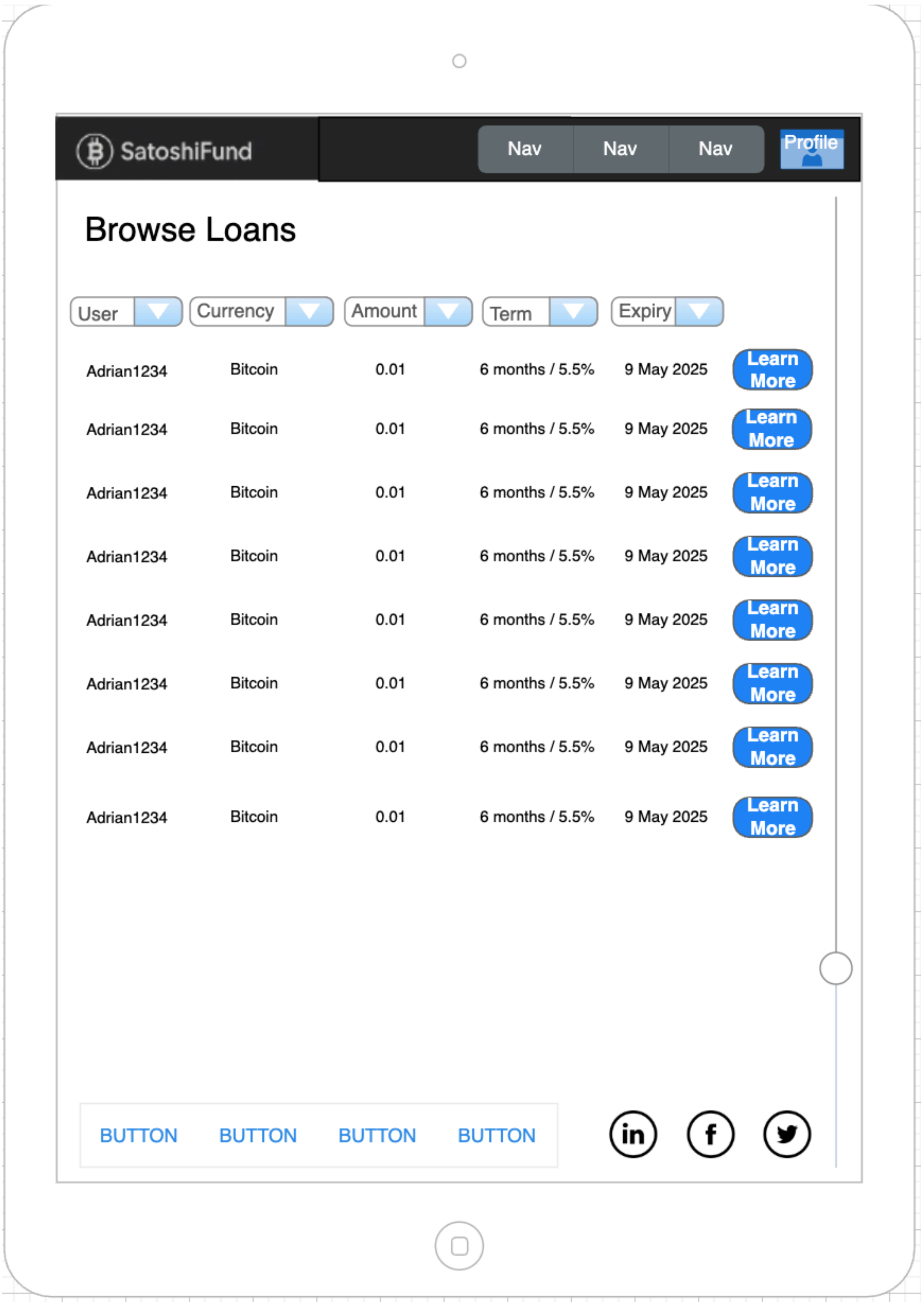
Desktop



Browse Loans

User	Currency	Amount	Term	Expiry	
Adrian1234	Bitcoin	0.01	6 months / 5.5%	9 May 2025	Learn More
Adrian1234	Bitcoin	0.01	6 months / 5.5%	9 May 2025	Learn More
Adrian1234	Bitcoin	0.01	6 months / 5.5%	9 May 2025	Learn More
Adrian1234	Bitcoin	0.01	6 months / 5.5%	9 May 2025	Learn More
Adrian1234	Bitcoin	0.01	6 months / 5.5%	9 May 2025	Learn More
Adrian1234	Bitcoin	0.01	6 months / 5.5%	9 May 2025	Learn More
Adrian1234	Bitcoin	0.01	6 months / 5.5%	9 May 2025	Learn More
Adrian1234	Bitcoin	0.01	6 months / 5.5%	9 May 2025	Learn More

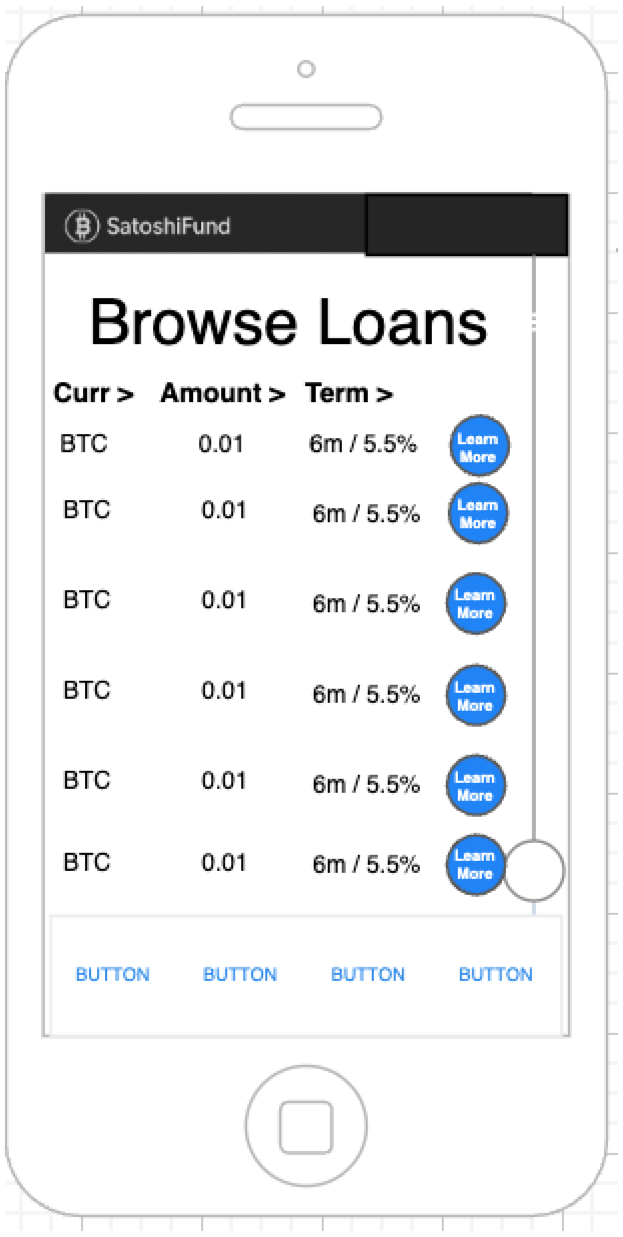
Tablet



Browse Loans

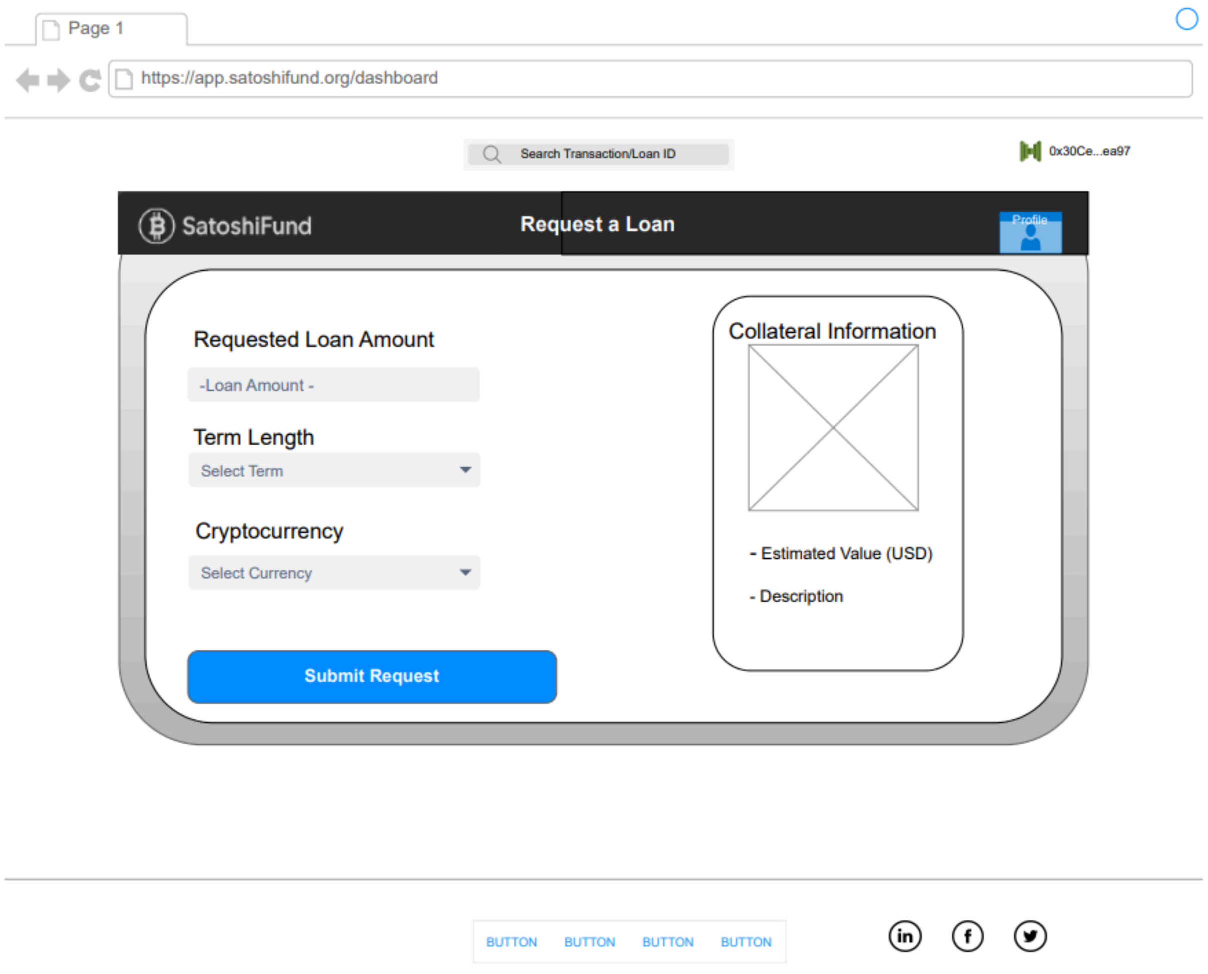
User	Currency	Amount	Term	Expiry	
Adrian1234	Bitcoin	0.01	6 months / 5.5%	9 May 2025	Learn More
Adrian1234	Bitcoin	0.01	6 months / 5.5%	9 May 2025	Learn More
Adrian1234	Bitcoin	0.01	6 months / 5.5%	9 May 2025	Learn More
Adrian1234	Bitcoin	0.01	6 months / 5.5%	9 May 2025	Learn More
Adrian1234	Bitcoin	0.01	6 months / 5.5%	9 May 2025	Learn More
Adrian1234	Bitcoin	0.01	6 months / 5.5%	9 May 2025	Learn More
Adrian1234	Bitcoin	0.01	6 months / 5.5%	9 May 2025	Learn More
Adrian1234	Bitcoin	0.01	6 months / 5.5%	9 May 2025	Learn More

Mobile

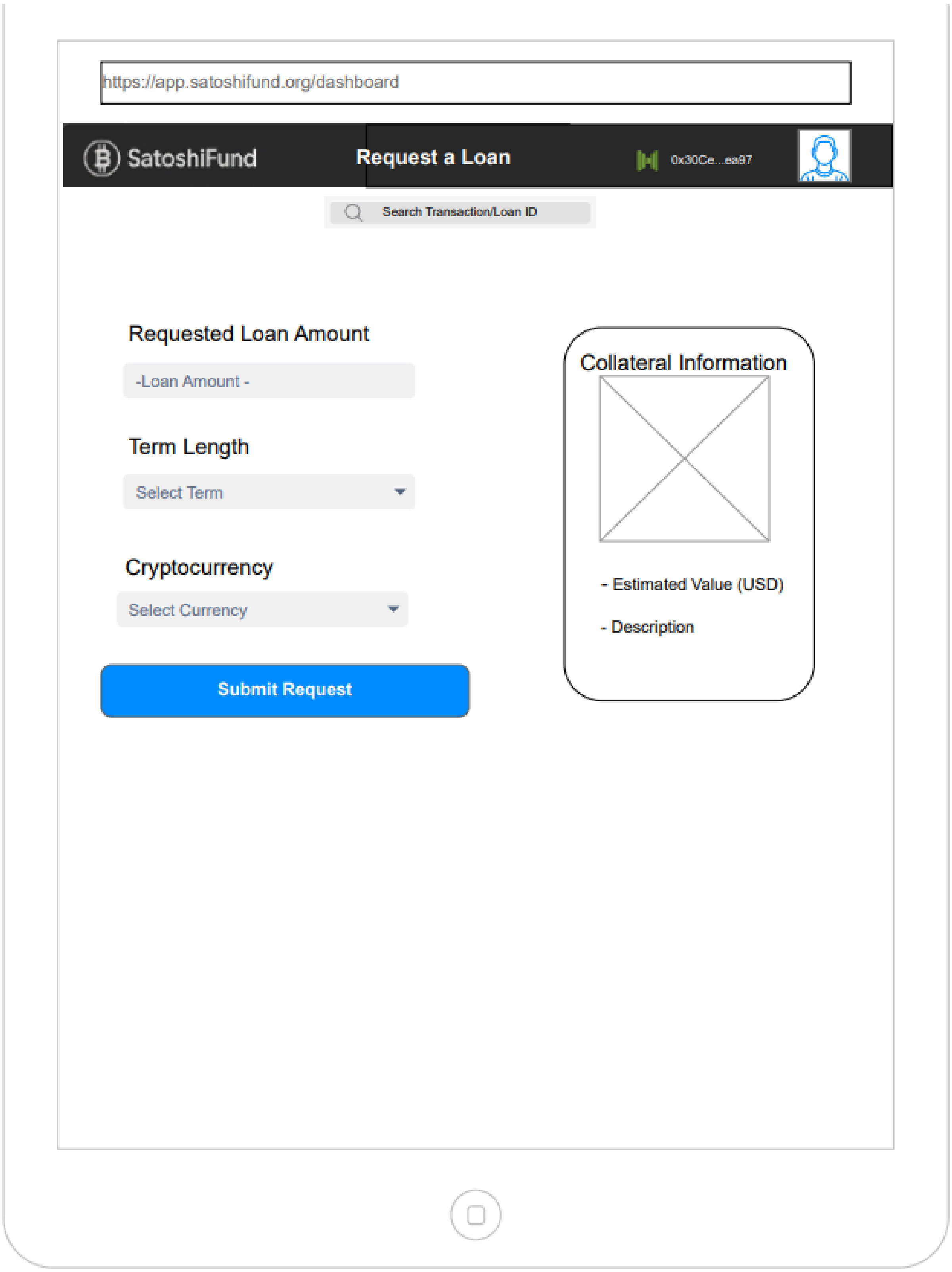

SatoshiFund
Browse Loans
Curr >
Amount >
Term >
BTC
0.01
6m / 5.5%
Learn More
BTC
0.01
6m / 5.5%
Learn More
BTC
0.01
6m / 5.5%
Learn More
BTC
0.01
6m / 5.5%
Learn More
BTC
0.01
6m / 5.5%
Learn More
BTC
0.01
6m / 5.5%
Learn More
BUTTON
BUTTON
BUTTON
BUTTON

Request A Loan

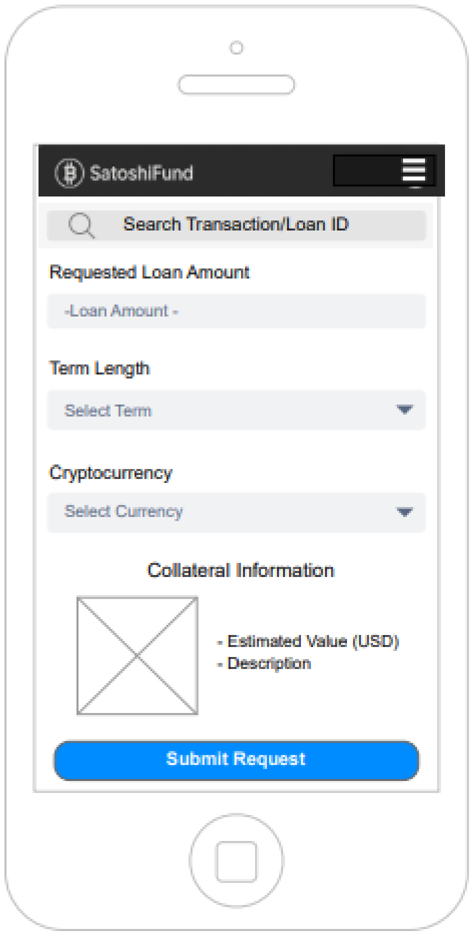
Desktop



Tablet

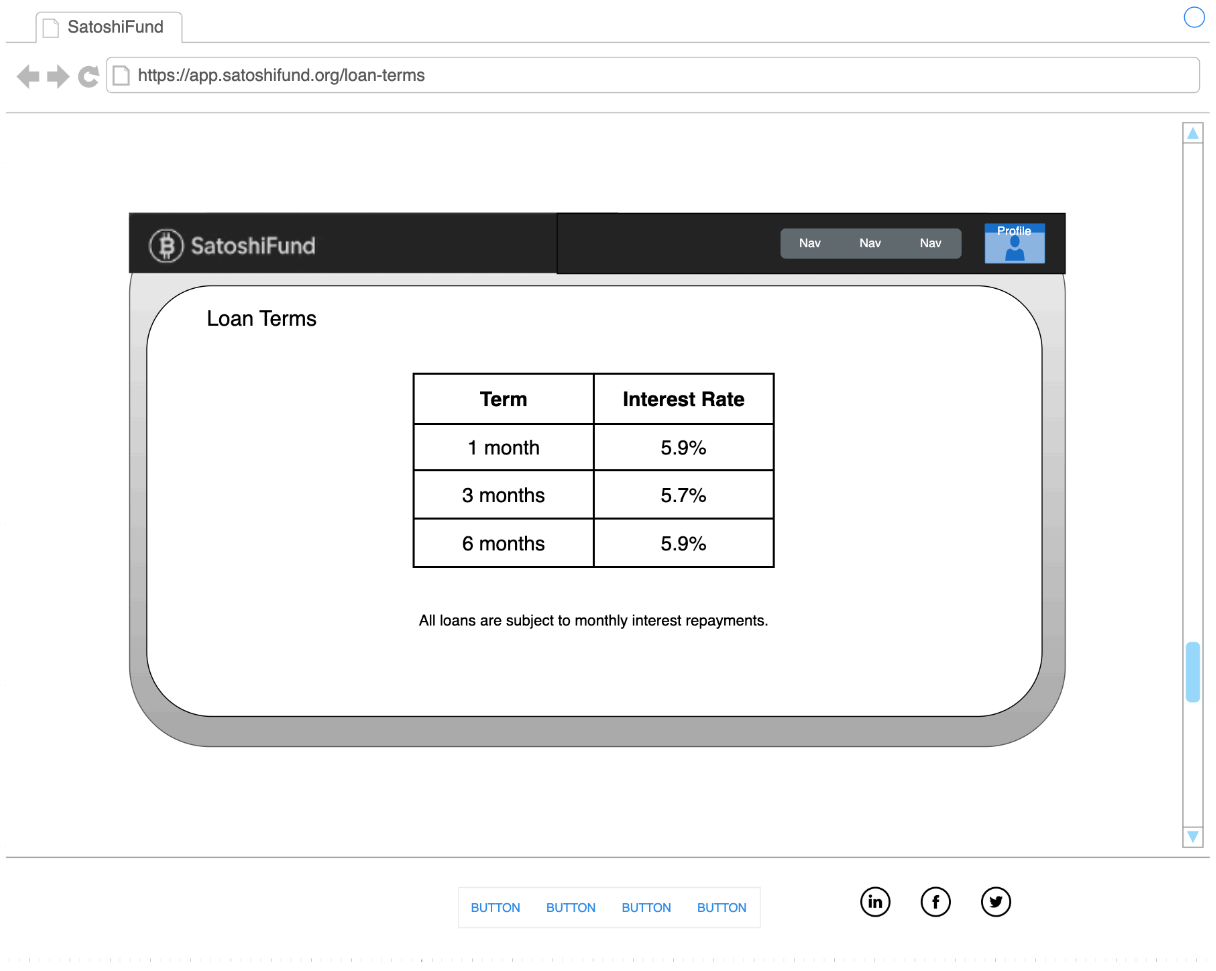

https://app.satoshifund.org/dashboard
SatoshiFund
Request a Loan
0x30Ce...ea97
Search Transaction/Loan ID
Requested Loan Amount
-Loan Amount -
Term Length
Select Term
Cryptocurrency
Select Currency
Submit Request
Collateral Information
- Estimated Value (USD)
- Description

Mobile


SatoshiFund
Search Transaction/Loan ID
Requested Loan Amount
-Loan Amount -
Term Length
Select Term
Cryptocurrency
Select Currency
Collateral Information
- Estimated Value (USD)
- Description
Submit Request

Loan Terms

Desktop

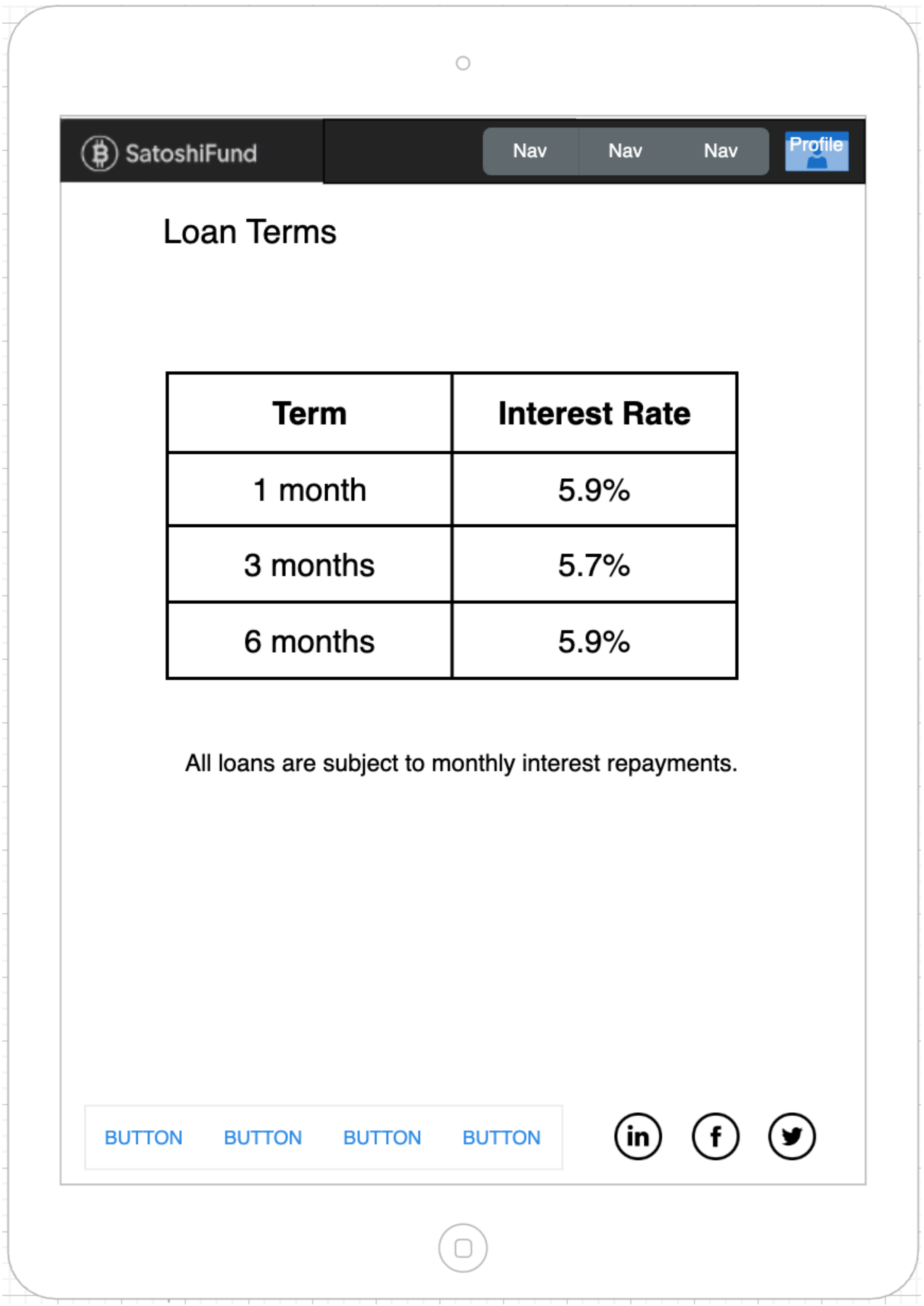


Loan Terms

Term	Interest Rate
1 month	5.9%
3 months	5.7%
6 months	5.9%

All loans are subject to monthly interest repayments.

Tablet

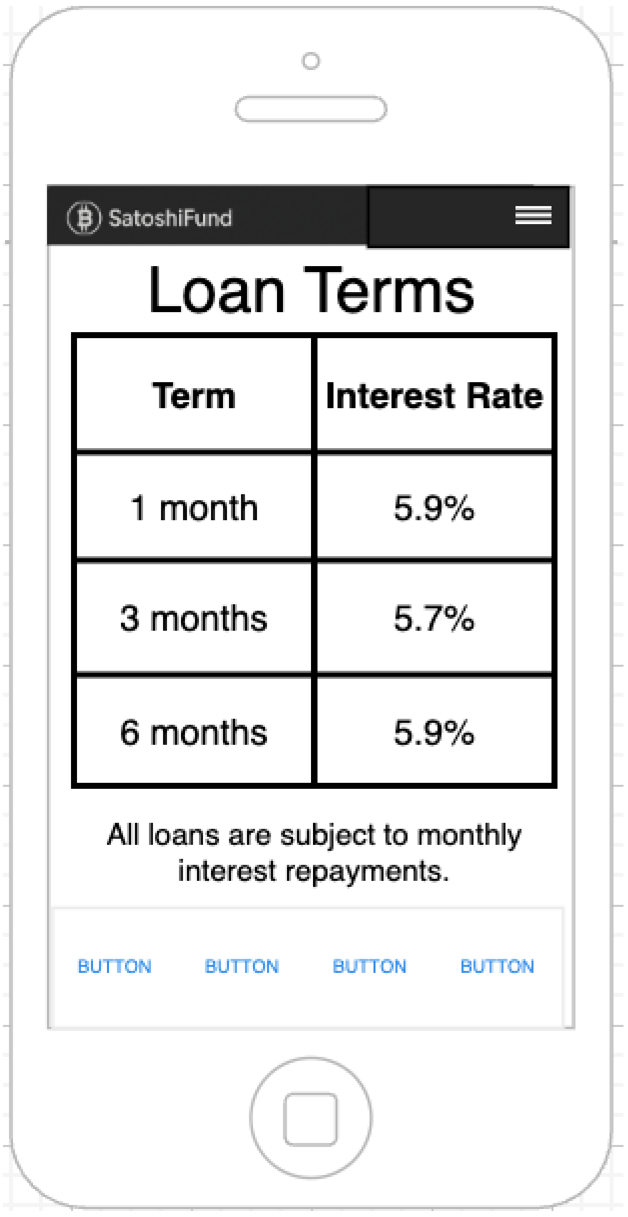


Loan Terms

Term	Interest Rate
1 month	5.9%
3 months	5.7%
6 months	5.9%

All loans are subject to monthly interest repayments.

Mobile



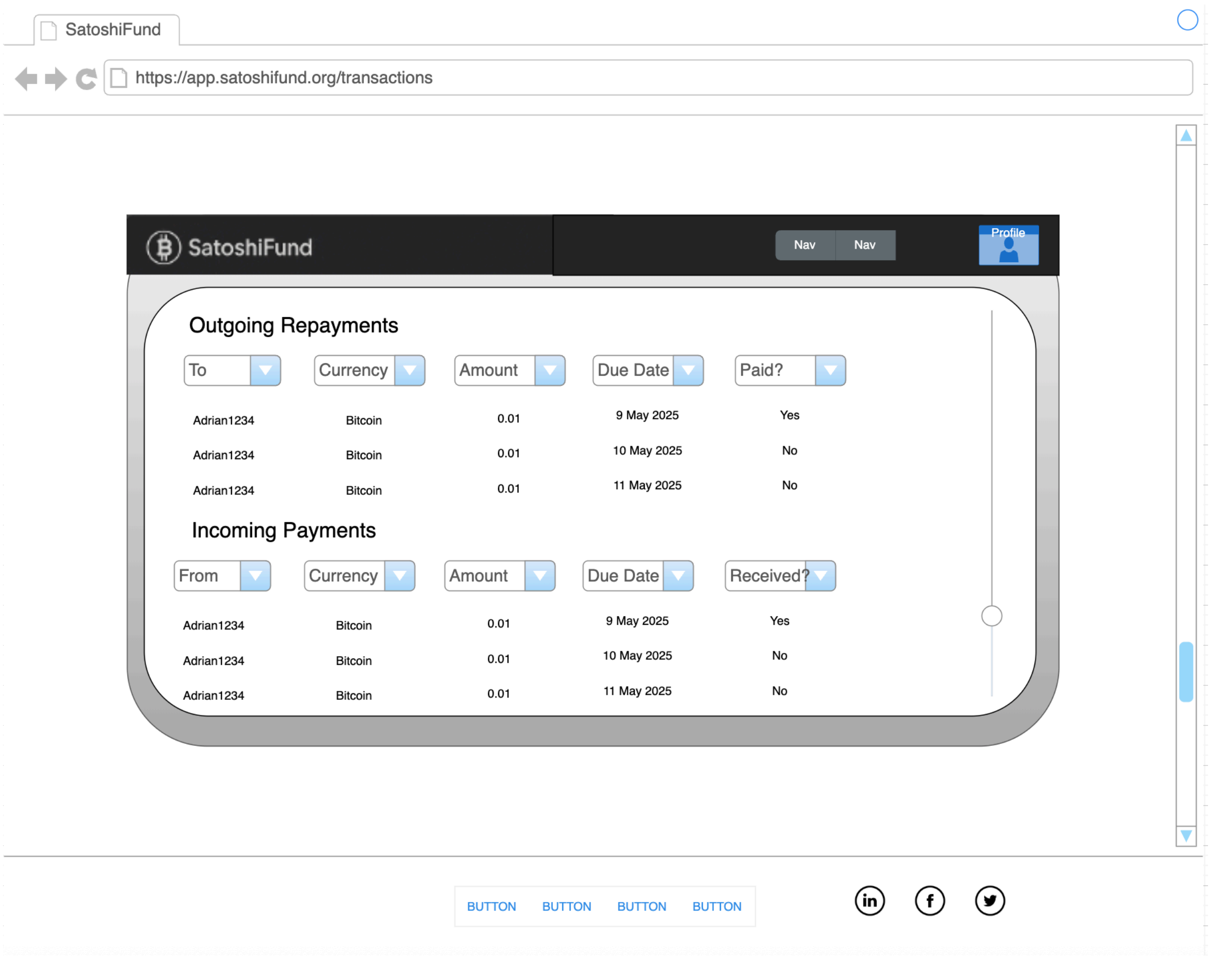
Loan Terms

Term	Interest Rate
1 month	5.9%
3 months	5.7%
6 months	5.9%

All loans are subject to monthly interest repayments.

Transactions

Desktop



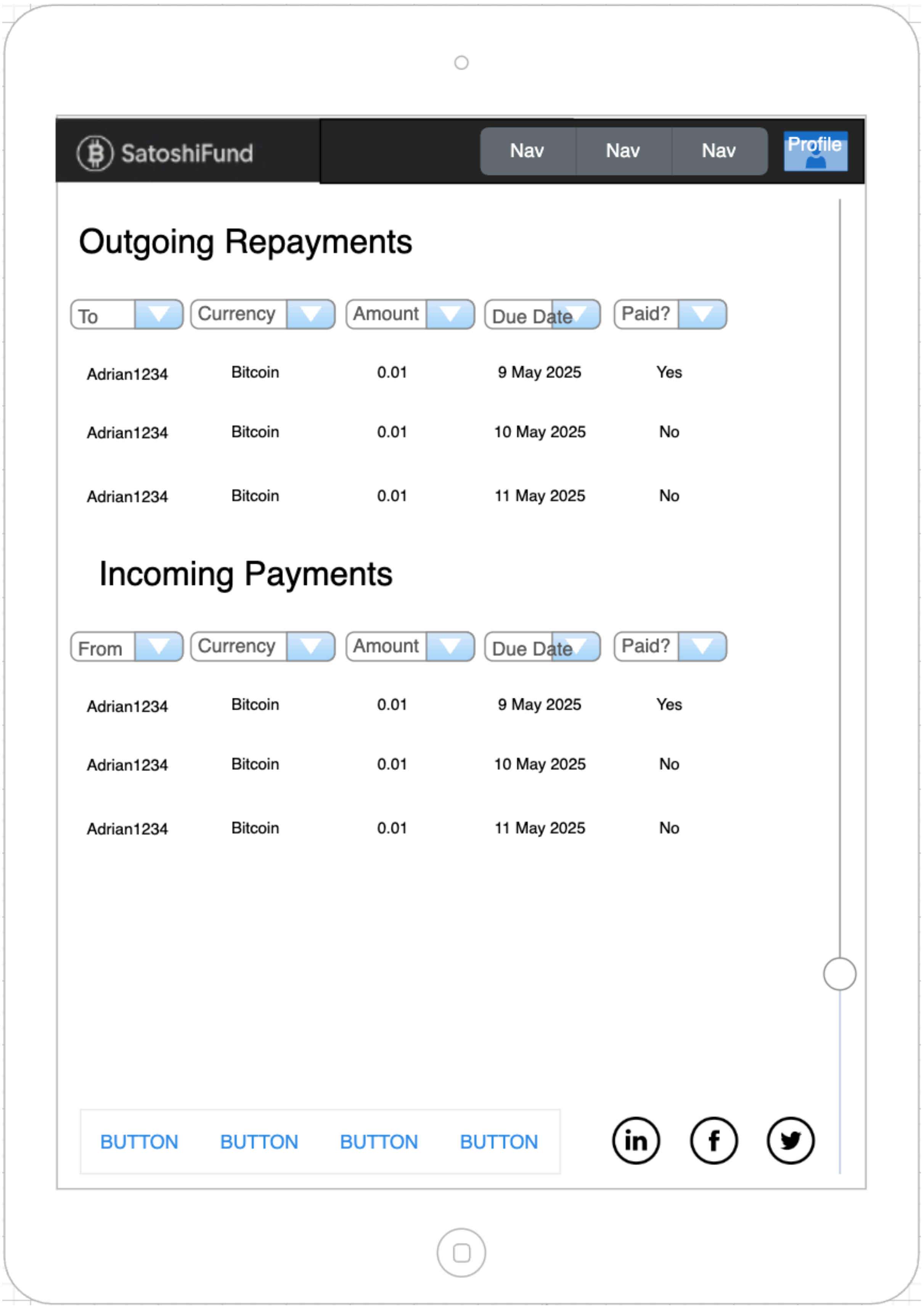
Outgoing Repayments

To	Currency	Amount	Due Date	Paid?
Adrian1234	Bitcoin	0.01	9 May 2025	Yes
Adrian1234	Bitcoin	0.01	10 May 2025	No
Adrian1234	Bitcoin	0.01	11 May 2025	No

Incoming Payments

From	Currency	Amount	Due Date	Received?
Adrian1234	Bitcoin	0.01	9 May 2025	Yes
Adrian1234	Bitcoin	0.01	10 May 2025	No
Adrian1234	Bitcoin	0.01	11 May 2025	No

Tablet



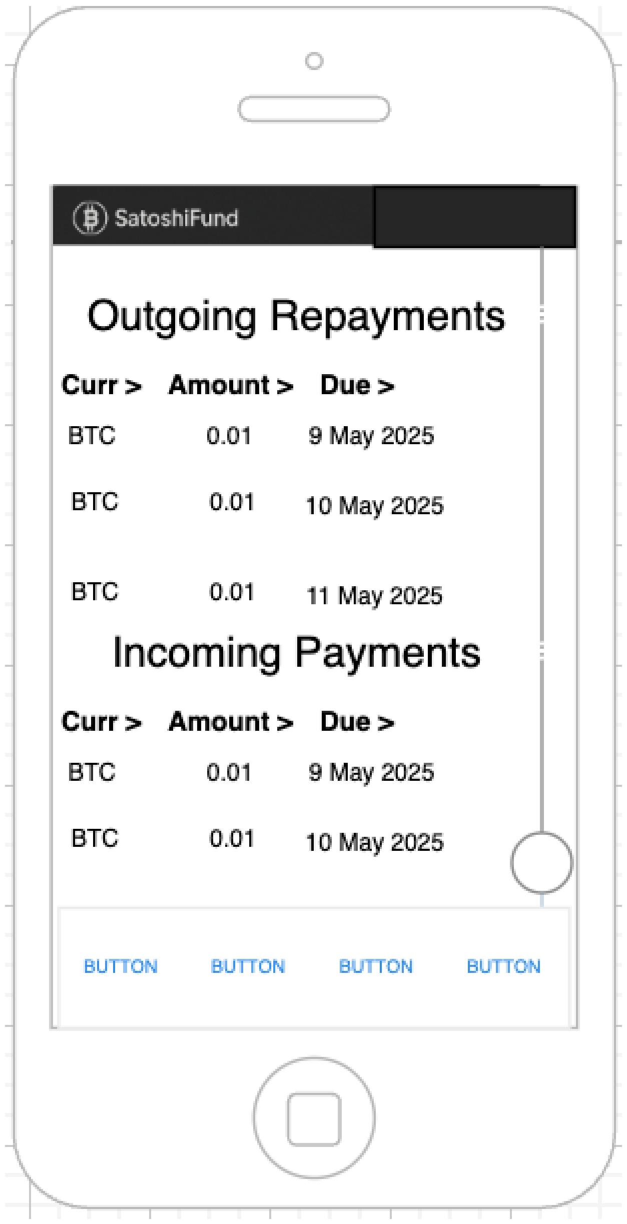
Outgoing Repayments

To	Currency	Amount	Due Date	Paid?
Adrian1234	Bitcoin	0.01	9 May 2025	Yes
Adrian1234	Bitcoin	0.01	10 May 2025	No
Adrian1234	Bitcoin	0.01	11 May 2025	No

Incoming Payments

From	Currency	Amount	Due Date	Paid?
Adrian1234	Bitcoin	0.01	9 May 2025	Yes
Adrian1234	Bitcoin	0.01	10 May 2025	No
Adrian1234	Bitcoin	0.01	11 May 2025	No

Mobile



Outgoing Repayments

Curr >	**Amount >**	**Due >**
BTC	0.01	9 May 2025
BTC	0.01	10 May 2025
BTC	0.01	11 May 2025

Incoming Payments

Curr >	**Amount >**	**Due >**
BTC	0.01	9 May 2025
BTC	0.01	10 May 2025