Agreed.

Agreed.

Purpose

Agreed. is a full-stack web application designed to simplify and formalize person-to-person loan agreements.

It addresses the lack of legal structure, traceability, and accountability in informal lending.

Most people rely on verbal agreements or casual messaging, which can lead to disputes or missed payments.

The goal is to give everyday users an accessible, legally-respectable way to draft and manage loans digitally.

While some platforms attempt to solve this, they are often too complex or designed for enterprise use.

Industry / Domain

Domain: Fintech and Legal-Tech

The industry is filled with tools for business lending, but few focus on casual peer-to-peer loans.

There is a growing need for digital contract management among non-professionals.

Core industry concepts include contract automation, digital identity, and payment transparency.

This project could also serve freelance work, rentals, or service-based contracts.

This project could also serve freelance work, rentals, or service-based contracts.

Stakeholders

Primary stakeholders:

Individual lenders and borrowers

Secondary stakeholders:

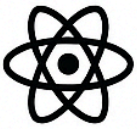
Legal advisors, small business owners, and personal lenders

Stakeholder goals

- Simplify agreement drafting
- Enable digital signatures and proof of agreement
- Make repayments easy to track and verify
- Provide a legal record of the transaction

Product Description

Architecture

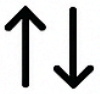


Frontend built with React and Vite, styled using Bootstrap

Backend developed in Node.js with Express, structured using MVC

Data is stored in JSON files (users.json, contracts.json, archived Contracts.json)

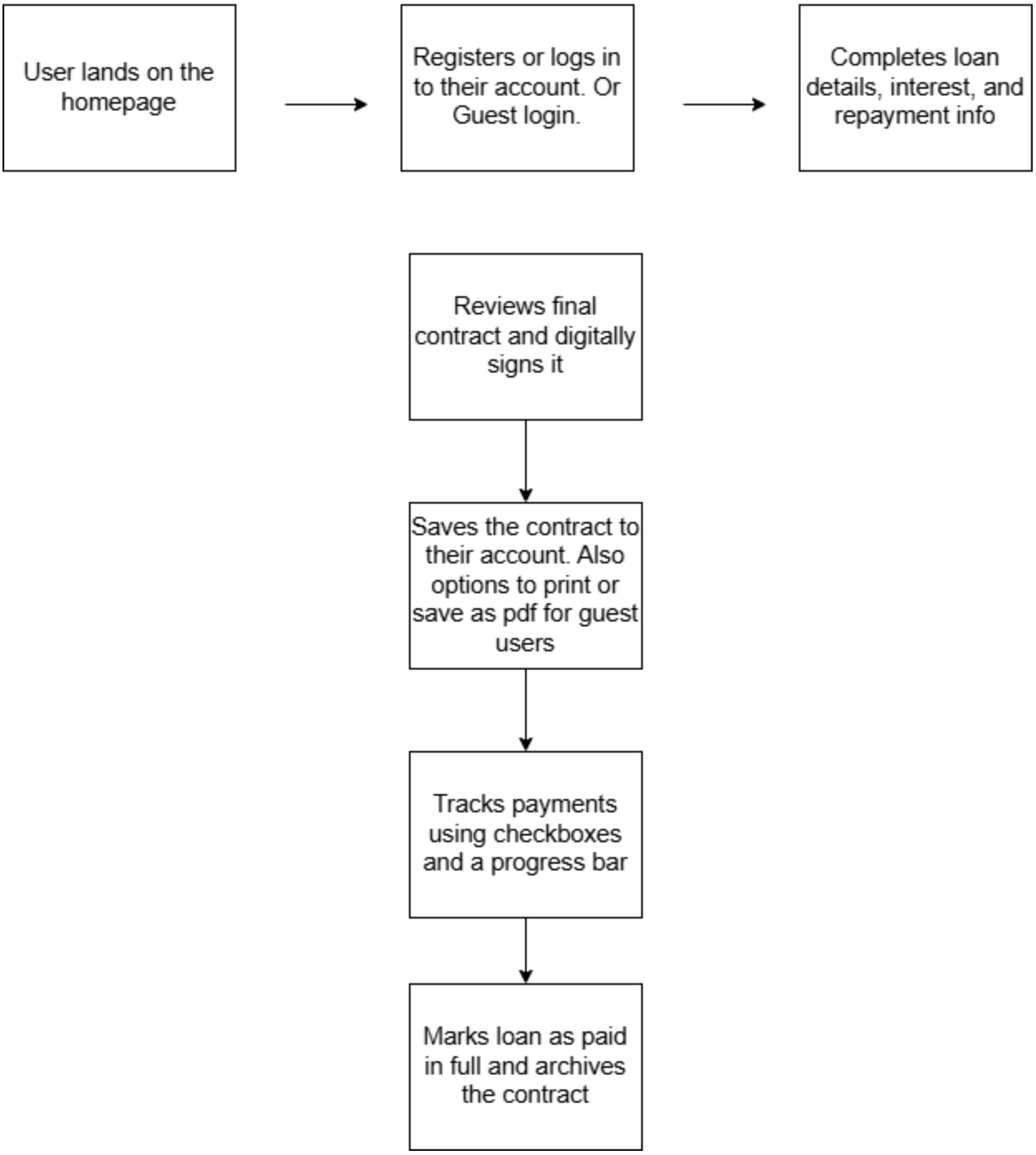
RESTful API manages user accounts contract saving, and archived via fetch-based HTTP requests



User Stories

User Story	Description	Importance
User Story and Login Register an	As a user, I want to securely register and log in so that I can create and access my loan agreements.	5
Create a Loan Contract Sign Contract	As a user, I want to enter loan terms (role, amount, interest, repayment) so I can generate a clear agreement.	5
Sign Contract Digitally	As a user, I want to sign the agreement using a digital signature so that the contract is legally valid.	5
Track Payments Track Pay-	As a user, I want to check off each payment and see a progrest bar so thȧt I can track loan repayment status.	4
Archive Completed Contracts	As a user, I want to move fully paid contracts into an archive so I can keep my dashboard clean and organized.	3

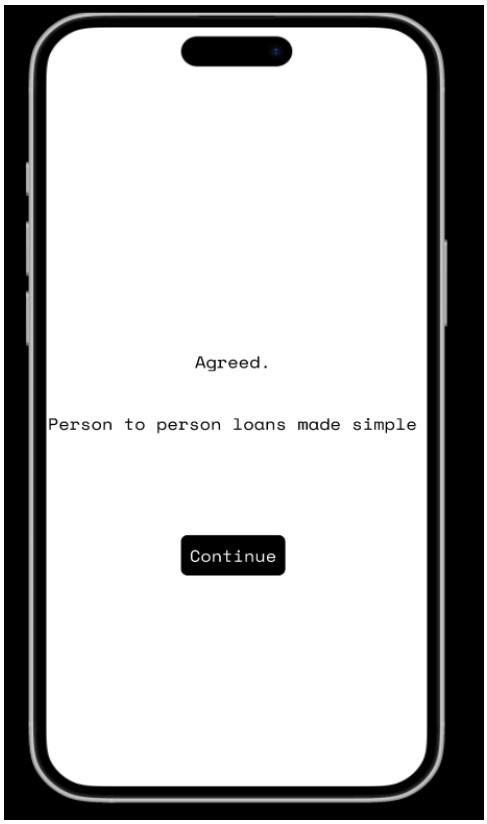
User Flow


User lands on the homepage
Registers or logs in to their account. Or Guest login.
Completes loan details, interest, and repayment info
Reviews final contract and digitally signs it
Saves the contract to their account. Also options to print or save as pdf for guest users
Tracks payments using checkboxes and a progress bar
Marks loan as paid in full and archives the contract

Wireframe Design

Link to original prototype:

https://www.figma.com/proto/4STfdY8mK37FOa8aGgD8RN/MiniProject?node-id=1-3&starting-point-node-id=1%3A3



Open Questions / Out of Scope

Integration with actual payment providers (e.g. Stripe, PayPal)

Blockchain or notarized contract verification

PDF import or document scanning for external contracts

QR code payment functionality (considered but not implemented)

Project Planning

Phase 1: Initial wireframe/Figma Design

Phase 2: Frontend multi-step contract flow

Phase 3: Add digital signature and interest calculations

Phase 4: Account page, Contract saving and archiving functionality

Phase 5: UI polish, testing, README creation, and documentation

Unit Test Overview:

Technologies Used:
• Jest – JavaScript testing framework
• Supertest – HTTP testing for Express.js

Test Coverage:

User API:
• POST /user/signup – Test user registration
• POST /user/login – Test invalid login response

Contract API:
• POST /contracts/:username – Test contract creation
• GET /contracts/:username – Test retrieval of user's contracts

Test Files & Status:
• users.test.js
  - Signup test – Pass
  - Login test – Pass

• contracts.test.js
  - Create contract test – Pass
  - Get contracts test – Pass

Terminal Output:
• Test Suites: 2 passed, 2 total
• Tests: 4 passed, 4 total

How to Run:
• In terminal, navigate to backend folder
• Run `npm install`
• Run `npm test`

Importance Ratings:
• User Signup API Test – 5 (Critical)
• User Login API Test – 5 (Critical)
• Contract Creation Test – 5 (Critical)
• Contract Retrieval Test – 4 (Important)

Implementation

App runs locally with separate frontend and backend servers

Frontend served via Vite (npm run dev)

Backend served via Express (node server.js)

Contracts and user data saved as JSON

Designed for easy deployment to Render, Railway, or Heroku

Future-ready for migration to MongoDB

End-to-End Solution

The app allows users to create and formalize legally structured loans

Both parties can digitally sign the agreement

Users can view, update, and archive contracts

The project meets its functional and design goals, providing an end-to-end working solution

References

GitHub repository:
https://github.com/aesmithiv/Capstone

Frontend tools: React, Vite, Bootstrap, jsPDF, Signature Pad

Backend tools: Node.js, Express

Utilities:

- date-fns – for formatting and calculating repayment dates
- Git and GitHub – for version control and project collaboration
- VS Code – used as the primary code editor for frontend and backend development
- Jest & Supertest – for writing and running backend unit and API tests