Lend.me

Team: Not Our (Seg)Fault!

Software System Requirements Document

Group Members: **Connor Black**

Damien King-Acevedo

Edward Skrod

Alexander Windelberg

Table Of Contents

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this document is to specify each necessary requirement of the Lend.me system in detail. The description of the functional and nonfunctional requirements of the system will guide the team members of Not Our Seg(Fault)! through the development of the product Lend.me. This SRS will provide engineers and other interested parties with all the information necessary to design, develop and test the product. The intended audience for this document are the developers of the Lend.me system, Not Our (Seg)Fault!, Professor Gaitros, the CEN4020-1 teaching assistants and other interested students of CEN4020.

1.2 Scope

The project's scope, in the short term, is to produce a working prototype with maximum functionality for the class, CEN4020-21, Software Engineering I and II. However, the project is scalable to the English-speaking world and beyond with minimal further requirements. To become a commercially viable product, Lend.me would need:

- A commercially viable name (Lend.me is currently owned by another party and is used only as an example name)
- An improved payment structure to monetize a percentage of loans made (such as 1-2% per loan transaction)
- Multi-lingual support
- Support for local currencies

1.3 Definitions, Acronyms and Abbreviations

1.3.1 Definitions

1.3.1.1 Backbone.js - A JavaScript library with a RESTful JSON interface that is based upon the model-view-presenter (MVP) application design paradigm. It is known for being lightweight, as its only dependency is on one JavaScript library, Underscore.js. It gives structure to web applications by providing models with key-value binding and custom events.

1.3.1.2 Balanced Payments - A third party API that facilitates card and bank transfers between customers.

1.3.1.3 Borrower - Any user who borrows money from another Lend.me user

1.3.1.4 Client - A user who accesses the Lend.me server through HTTP (i.e. a web browser, mobile phone, etc.)

1.3.1.5 Composite Architecture - A pattern that encapsulates hierarchies by providing a common superclass for aggregate and leaf nodes. New types of leaves can be added without modifying existing code.

1.3.1.6 Joyent - A third party service that provides cloud server hosting.

1.3.1.7 JSON - JavaScript Object Notation is an open standard format that uses human-readable text to transmit data objects consisting of attribute-value pairs. It is used primarily to transmit data between a server and web application.

1.3.1.8 JSON Schema - specifies a JSON-based format to define the structure of JSON data for validation, documentation, and interaction control. Provides a contract for the JSON data required by a given application and how the data can be modified.

1.3.1.9 Lender - Any user who lends money to another Lend.me user

1.3.1.10 Model-View-Controller - A software pattern for implementing user interfaces which divides an application into three interconnected parts, to separate the representations of information from the ways that the information is presented to or accepted from the user. The *model* consists of application data, business rules, logic and functions, which the *view* can be any output representation of information. The *controller* sends commands to the *model* to update the *model's* state and can send commands to the *view* to change the *view's* presentation of the *model*.

1.3.1.11 Model-View-Presenter - derivative of the model-view-controller software pattern, used mostly for building user interfaces. In MVP, the *presenter* assumes the functionality of the *controller* in MVC. The *model* is an interface defining the data to be displayed or acted upon in the user interface. The *view* is a passive interface that displays data and routes user commands to the *presenter*, *which acts upon the model and view.* The presenter is responsible for binding the model to the view.

1.3.1.12 MongoDB - A cross-platform, document-oriented database system.

1.3.1.13 MongoLab - A cloud hosted MongoDB database service.

1.3.1.14 MySQL - A widely used, open-source relational database management system. It is a popular choice of database for use in web applications.

1.3.1.15 Node.js - A software platform that is used to build scalable network (especially server-side) applications. Node.js utilizes JavaScript as its scripting language, and achieves high throughput via non-blocking I/O and a single-threaded event loop.

1.3.1.16 NoSQL - Databases that provide a mechanism for storage and retrieval of data that employs less constrained consistency models than traditional relational databases. NoSQL databases are often highly optimized key-value stores intended for simple retrieval and appending operations with the goal being significant performance benefits in terms of latency and throughput. NoSQL databases are finding significant and growing industry use in real-time web applications.

1.3.1.17 RESTful - Representational state transfer (REST) is a coordinated set of constraints applied to components, connectors and data elements, within a distributed hypermedia system. Its properties include simplicity of interfaces, modifiability of components to meet changing needs, portability of component deployment, and scalability of component interactions.

1.3.1.18 SaaS - software delivery model in which software and associated data are centrally hosted on the cloud. Typically accessed by users using a thin client via a web browser. SaaS has become a common delivery model for many business applications, including Office, Messenger, DBMS software, etc.

1.3.1.19 Server - a computer or computer program that manages access to a centralized resource or service in a network.

1.3.1.20 Underscore.js - a JavaScript library which provides utility functions for common JavaScript tasks.

1.3.1.21 Visitor - A visitor is a person who uses or visits the application for the first time. Visitors will have limited access to the application but will

have access to the FAQ and screenshots of the product in use. Additionally, visitors will have the ability to register and become users.

1.3.2 Acronyms

1.3.2.1 API - Application programming interface

1.3.2.2 DB - Database

1.3.2.3 ERD - Entity Relationship Diagram

1.3.2.4 FAQ - Frequently Asked Questions

1.3.2.5 FURPS+ - Functionality, Usability, Reliability, Performance, and Supportability. The + indicates additional subcategories.

1.3.2.6 HTTP - Hypertext Transfer Protocol

1.3.2.7 JSON - JavaScript Object Notation

1.2.2.8 MVC - Model-View-Controller architecture

1.3.2.9 MVP - Model-View-Presenter architecture

1.3.2.10 REST - Representational State Transfer

1.3.2.11 SaaS - Software as a service

1.3.2.12 SRS - Software Requirement Specification

1.3.3 Abbreviations

None

1.4 References

1.4.1 Bernd Bruegge and Allen H. Dutoit. *Object-Oriented Software Engineering: Using UML, Patterns, and Java*. 3rd ed. Boston, Prentice Hall, 2010.

1.4.1 Cash Lender, http://sockii.com/portfolio/cash-lender/

1.4.2 Lending Club, https://www.lendingclub.com/

1.4.3 Lynda.com, "Node.js First Look with Joseph LeBlanc," Joseph LeBlanc, lynda.com/Nodejs-tutorials/Nodejs-First-Look

1.4.4 Lynda.com, "Node.js Essential Training with Joseph LeBlanc," Joseph LeBlanc, lynda.com/JavaScript-tutorials/Nodejs-Essential-Training

1.4.5 Lynda.com, "JavaScript Essential Training with Simon Allardice," Simon Allardice, lynda.com/JavaScript-tutorials/Essential-Training

1.4.6 Spotme, http://www.getspotme.com/

1.4.7 Venmo, http://www.venmo.com

1.4.8 ZimpleMoney, http://www.zimpleMoney.com

1.5 Overview

The Lend.me SRS document contains a detailed analysis of the Lend.me system which explains the requirements that the developers will take into consideration in order to make the software functional. The requirements and non-functional requirements that will make the product work will be explained in this document. The SRS will be organized in the following way:

1.5.1 Section 1 - Introduction

Section 1 will be used by the developers and other interested parties to gain a brief understanding of the overall product, Lend.me. Users and developers may use this section to gain access to definitions, abbreviations and acronyms that will be used throughout the entire document. In addition, Section 1 includes the purpose of the SRS document and the Scope of the produce.

1.5.2 Section 2 - Overall Description

Lend.me will serve three main purposes:

First, it facilitates the creation and recording of the loan contract. This is important for remembering when the loan is due and for legal purposes (in the event that the lender wishes to take the borrower to court). Second, it is a reminding service. It reminds all parties of their obligation and their due. Third, it is a repayment service. It facilitates the paying of the initial loan and its subsequent repayment.

Lend.me will not authenticate identities nor will it ensure that borrowers keep their word and make repayment. Lend.me, however, will provide a

paper trail for lenders in the off-chance that the borrower does not repay the loan.

1.5.3 Section 3 - Specific Requirements

Primarily used by developers, Section 3 provides in-depth descriptions of the product's specific requirements. Section 3.1 details the products external interface requirements, while Section 3.2 outlines the application's functional requirements. Nonfunctional requirements are derived according to the FURPS+ model used by the Unified Process.[1] Performance requirements, design constraints and software system attributes are also included in Sections 3.3, 3.4 and 3.5 respectively.

1.5.4 Section 4 - Change Management Process

Section 4 contains an overview of the change management process that is used by the members of Not Our (Seg)Fault! when making, considering and implementing changes to the Lend.me application. Changes, such as making changes of implementation or additional information must first be spoken to the group and the majority must agree.

1.5.5 Section 5 - Document Approvals

Section 5 lists the team members of Not Our (Seg)Fault! that have approved of this SRS document.

1.5.6 Section 6 - Supporting Information

Section 6, supporting information begins with information on four applications that are either similar to Lend.me or have functionality that Lend.me would like to employ. Specifically, Section 6.1 shows the research of Cash Lender, SpotMe, ZimpleMoney and Venmo. Each sub-section lists the main functions of each similar application, the functions that Lend.me will not employ, and the way in which Lend.me differentiates itself from the similar application. Additionally, screenshots from each application are included.

[1] Bernd Bruegge and Allen H. Dutoit, *Object-Oriented Software Engineering: Using UML, Patterns, and Java*, 3rd ed., (Boston, Prentice Hall, 2010), 126.

Section 6.2 provides UML use case diagrams of each actor that will use Lend.me, such as visitor, user, lender and borrower. Detailed diagrams relating to login and loan are featured as well.

Section 6.3 contains a UML class diagram of the application's back end while section 6.4 provides a detailed model of the database, both as a MySQL Workbench ERD and as a JSON document.

2. Overall Description

2.1 Product Perspective

The software this team plans to create is meant to be a self sustained product that is web-based. If there is enough time in the semester, Not Our (Seg)Fault! will develop Lend.me into a mobile application as well. The product will be able to connect to a third-party payment system to give users the option to send money directly to the borrower as a loan or to facilitate the repayment of the loan.

2.1.1 System Interfaces

The Lend.me system interfaces with two major external services, which are Balanced Payments and PhoneGap. The Balanced Payments service exposes an API for handling all customer-to-customer payments, and manages all customer banking and card information so the application does not have to be PCI compliant. The PhoneGap service exposes an API that allows developers to interface with mobile systems. This allows developers to compile our codebase into different mobile apps which work on all major mobile operating systems.

2.1.2 User Interfaces

The Lend.me system has two interfaces from which users can access. The first interface is a GUI that can be accessed by any modern web browser such as Internet Explorer, Firefox or Chrome. The second interface is a mobile application that can be downloaded and installed on any modern mobile operating system.

2.1.3 Communications Interfaces

The system will use standard protocol.

2.1.4 Memory Constraints

As Lend.me may be developed for mobile devices and will need to interface with them, the Lend.me system needs to make sure its memory footprint stays under 1 Gigabyte, the current average memory capability of a modern mobile device.

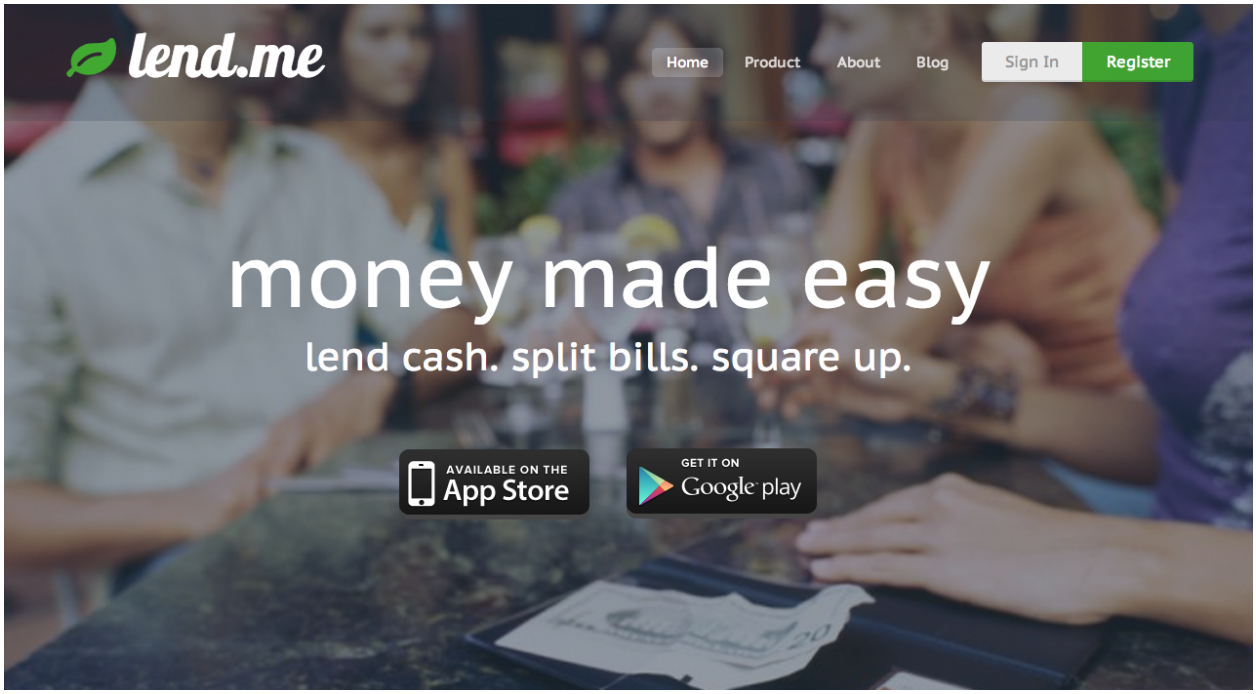
3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User interfaces

3.1.1.1 GUI Screenshots

3.1.1.1.1 Landing Page



lend.me

Sign In Register

Meet Lend.me

easily manage money with friends

Take Tour

OR

Get Started

lend.me

Sign In Register

Using lend.me: Moves

The Front

At vero eos et accusamus et iusto odio dignissimos ducimus qui blanditiis praesentium voluptatum deleniti atque corrupti quos dolores et quas molestias excepturi sint occaecati cupiditate non provident, similique sunt in culpa qui officia deserunt mollitia animi, id est laborum et dolorum fuga. Et harum quidem rerum facilis est et expedita distinctio.

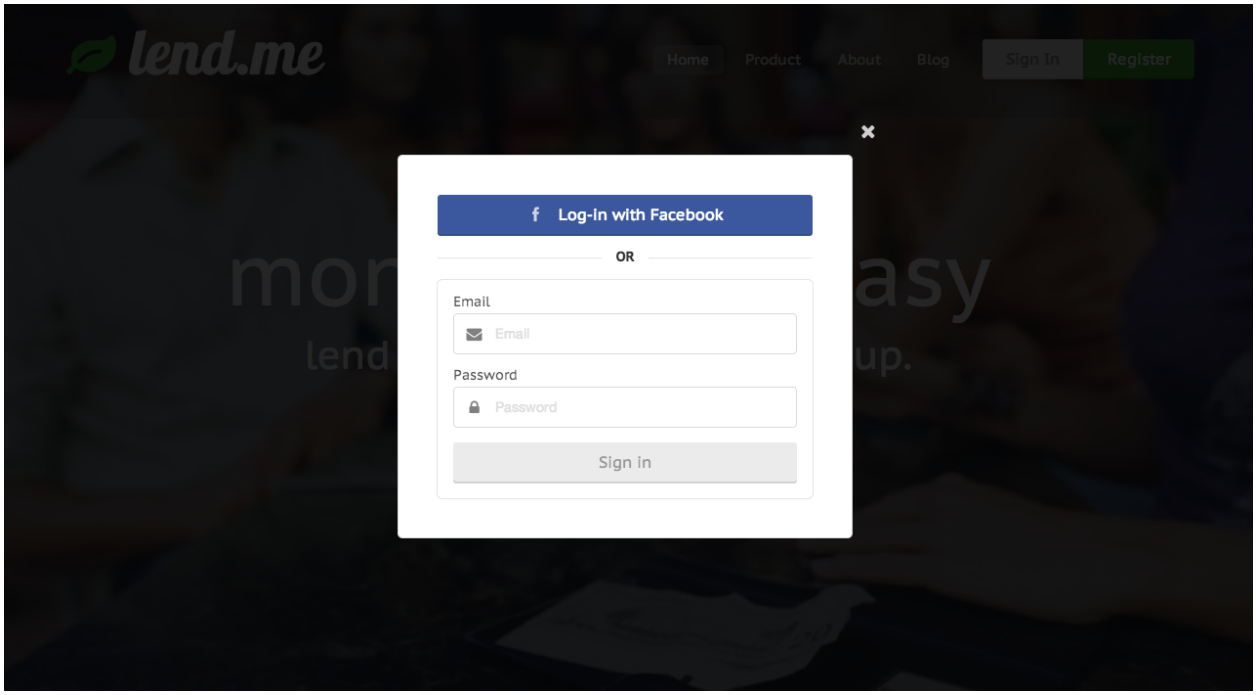
1 The Cover

2 The Front

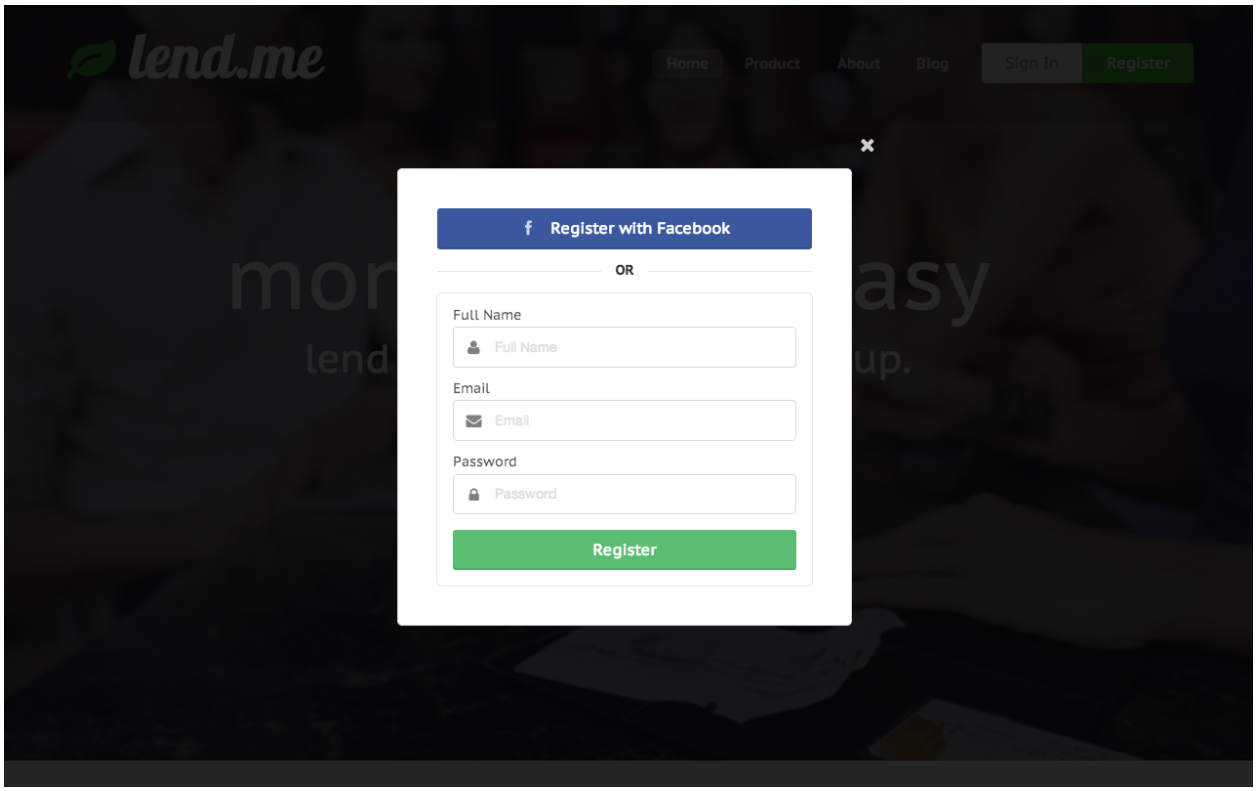
3 The Split

4 The Square Up

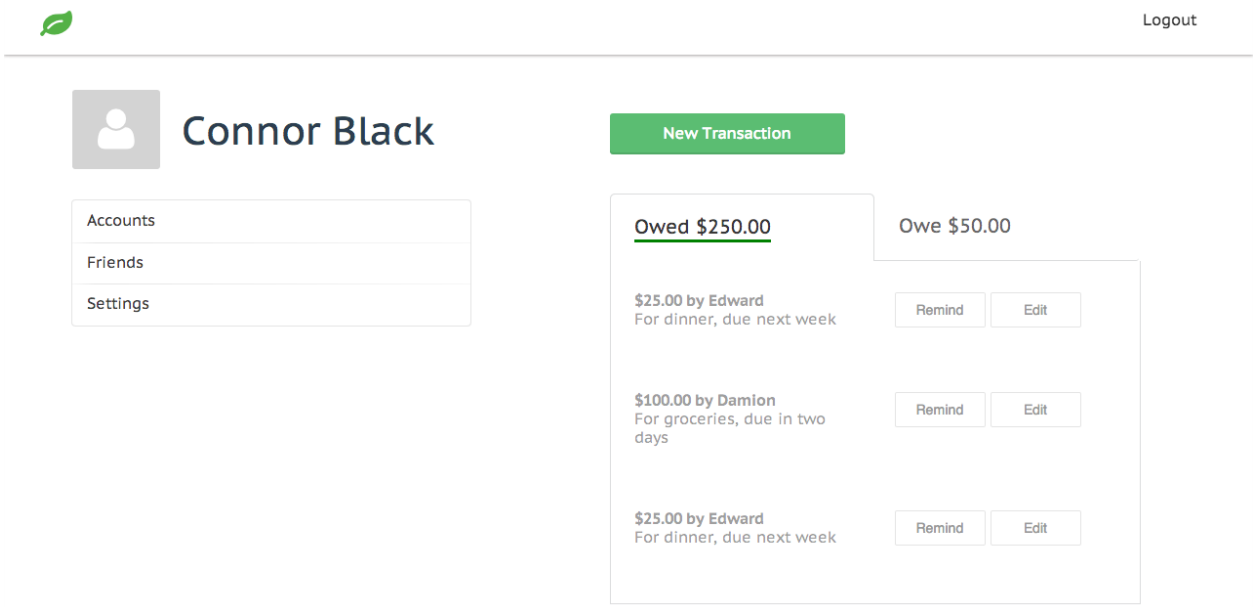
3.1.1.1.2 Login Page



3.1.1.1.3 Register Page



3.1.1.1.4 Home Page



3.1.2 Hardware Interfaces

None applicable.

3.1.3 Software Interfaces

3.1.3.1 MongoLab - MongoDB

Lend.me hosts its data persistence layer in the cloud with the SaaS company MongoLab. MongoLab provides a library interface that the server layer of Lend.me can connect with to pass JSON objects over a RESTful interface for persistence.

3.1.3.2 Joynet - SmartOS

Lend.me host the server application logic with the SaaS company Joynet. They provide us with a Linux-based distribution called SmartOS, that it tailored specifically toward NodeJS development.

3.1.3.3 Balanced Payments API

To expose an electronic payment method to Lend.me users, Lend.me interacts with the SaaS company Balanced Payments. Balanced Payments exposes an API that the Lend.me client and the Lend.me server layers can make use of. Balanced Payments keeps all user banking information and remains PCI compliant so that Lend.me does not have to.

3.1.3.4 Client

A client is a user who interacts with Lend.me through a browser and has downloaded the client JavaScript application, which allows him or her to interact with the application's server layer.

3.1.3.5 Server

Users who have logged in through our client have access to all the public-facing functionality of the Lend.me system.

3.1.4 Communications Interfaces

3.1.4.1 Browser

Lend.me allows users to access the system through a modern web browser such as Internet Explorer, Chrome, or Firefox.

3.4.4.2 Email

Lend.me interfaces with Gmail SMTP for all email sending capabilities.

3.4.4.3 Message Formatting

Lend.me formats all over the wire communcation in JSON.

3.4.4.4 Communication Standard

The standard for communication within, in and out of the Lend.me system is HTTP.

3.2 Functional requirements

3.2.1 Visitor

3.2.1.1 Visitors must register for an account to use the system by providing a unique email and password.

3.2.2 User

3.2.2.1 User is able to recover a lost username by entering their email addresses.

3.2.2.2 User must login to the system using his/her unique username and password.

3.2.2.3 User is able to change his/her password.

3.2.2.4 User is able to recover a lost password.

3.2.2.5 User is able to query for up-to-date transaction information.

3.2.2.6 User is notified by email of a reminder of loan due.

3.2.2.7 User can add a friend.

3.2.2.8 User can add a bank and/or card account.

3.2.3 Lender

3.2.3.1 Lender can start a new loan.

3.2.3.2 Lender can set terms of new loan.

3.2.3.3 Lender can agree to new terms of the loan.

3.2.3.4 Lender can disagree to new terms of the loan.

3.2.3.5 Lender can access a list of loans outstanding.

3.2.3.6 Lender can click on loan outstanding to access more information about it.

3.2.3.7 Lender can agree or disagree to a loan extension.

3.2.3.8 Lender can forgive the loan (clear it out).

3.2.3.9 Lender can lend via application or with cash.

3.2.3.10 Lender can send a personalized reminder to the borrower.

3.2.4 Borrower

3.2.4.1 Borrower can agree or disagree to terms of new loans.

3.2.4.2 Borrower can set new terms of the loan.

3.2.4.3 Borrower can ask for a loan extension.

3.2.4.4 Borrower ask for loan forgiveness.

3.2.4.5 Borrower can pay back loan by using the application or cash.

3.3 Performance Requirements

Since Lend.me is run in modern web browser, for all intents and purposes, there are no performance requirements.

3.4 Design Constraints

The design constraints for Lend.me are average-sized computer screens, 11 – 17 inches.

3.6 Non Functional Requirements

3.6.1 Usability

3.6.1.1 Lend.me should be easy to use and should not require any special training. A simple understanding of how to use a web browser should be sufficient.

3.6.1.2 A FAQ should be available to help visitors sign up for an account.

3.6.1.3 An FAQ should be available to help users lend, borrow or make a payment on a loan.

3.6.1.4 The system should prevent users from changing terms in a contract without the opposite party's consent.

3.6.1.5 The system should have a unit test for each module.

3.6.1.6 Each module should be attached to the application object.

3.6.1.7 The system should decouple each module.

3.6.1.8 The primary language of Lend.me is English. If possible, Lend.me should be available in Spanish.

3.6.1.9 Users should be able to access Lend.me with a web browser supporting cookies, JavaScript, and Java applets

3.6.1.10 User should be made aware of liability of not paying through the system.

3.6.2 Reliability

3.6.2.1 There are no exact constraints on the maximum acceptable time for restarting the system after a failure. However, the system should restart in a reasonable time.

3.6.3 Performance

3.6.3.1 There are no exact speed, throughput or response time constraints, but the system must perform all functions in a reasonable time.

3.6.4 System Interface

3.6.4.1 The system should be written in English.

3.6.4.2 The system should support all mobile and modern browsers.

3.6.4.3 The color scheme should be white and green.

3.6.4.4 The system should have the option to sign up/log in with Facebook.

3.6.4.5 The system should enforce that modules only communicate through events and the application object.

3.6.4.6 The website should have a modern, Web 2.0 look and feel

3.6.5 Operation

3.6.5.1 The system should verify a user's email before allowing them to continue registration.

3.5.6.2 The records returned to lenders and borrowers during a session should be updated in real time. The system should be capable of updating the displayed record count every few seconds.

3.5.6.3 Records of loans, payments and repayments should be backed up in real time.

3.5.6.4 The system server should run in a Unix environment.

The system should maintain a sessions object.

3.5.6.5 Each module should have its own API.

3.6.6 Security

3.6.6.1 The database should never store plain-text passwords.

3.6.6.2 The system should serve an HTTPS server.

3.6.6.3 The system should protect API requests by requiring either an email/password or a cookie.

4. Change Management Process

The Lend.me team has a structured method of enacting change to the document. First, changes are suggested freely within the Lend.me Facebook group or group meetings. Possible changes are logged automatically by Facebook or logged by the group secretary. Once everyone has had a chance to give his or her input on the proposed change, we take a consensus check on the proposed change. If a majority is in favor of the proposed change, we

officially enact the change. In the event of a tie, we will call in a third party, such as Professor Gaitros.

5. Document Process

The following members of Not Our (Seg)Fault! approve this SRS document:

Name	Date	Signature
Connor Black		
Damien King-Acevedo		
Edward Skrod		
Alex Windelberg		

6. Supporting Information

6.1 Research - Similar Applications to Lend.me

6.1.1 Cash Lender

Cash Lender is a debt management application that allows users to track incoming and outgoing debts.[2] It is iCloud compatible, supports loan management from multiple people, has built-in passcode protection, the ability to set due dates with reminders and even supports local currencies. Cash Lender is available for Mac, Ipad and Iphone. Each version can sync with its counterparts. Cash Lender is $14.95 for Mac, $4.99 for Ipad and $2.99 for the IPhone.

What Cash Lender Offers that Lend.me does not:

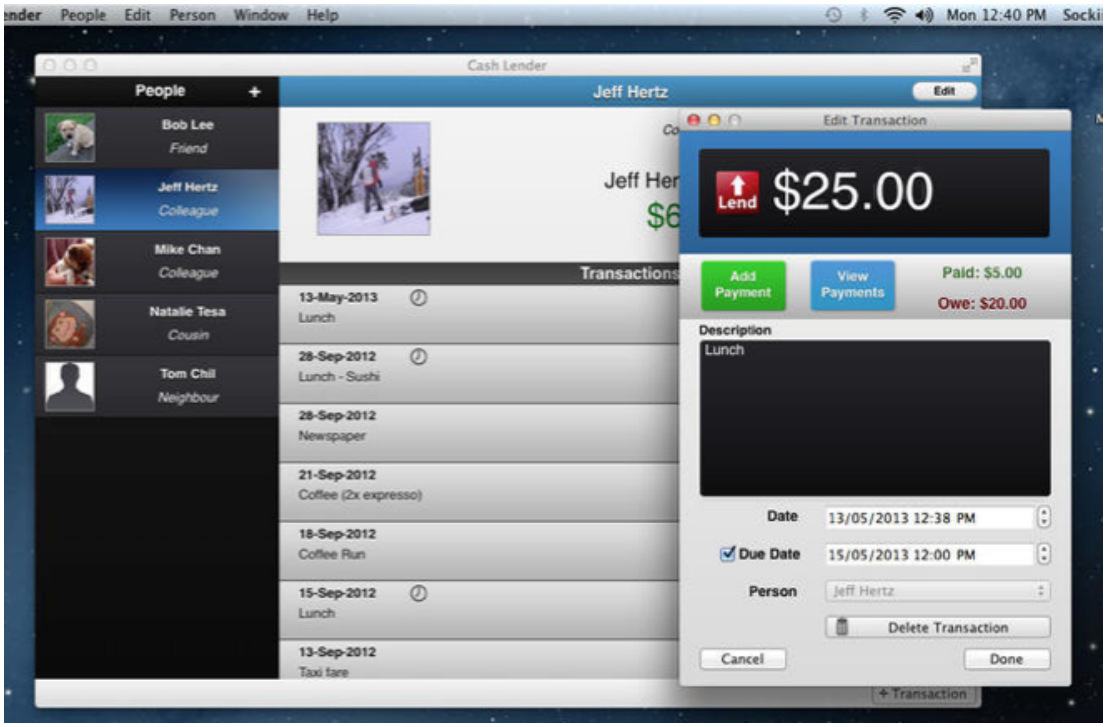
- iCloud compatibility
- Local Currency Support
- Passcode protection (not necessary for a web application like Lend.me)

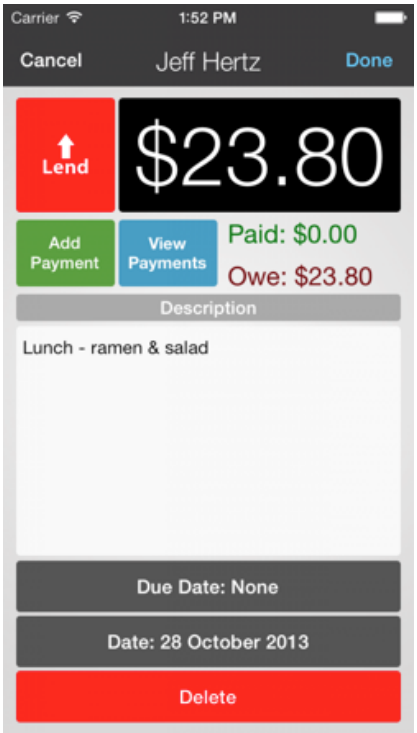
What Lend.me will offer that Cash Lender does not:

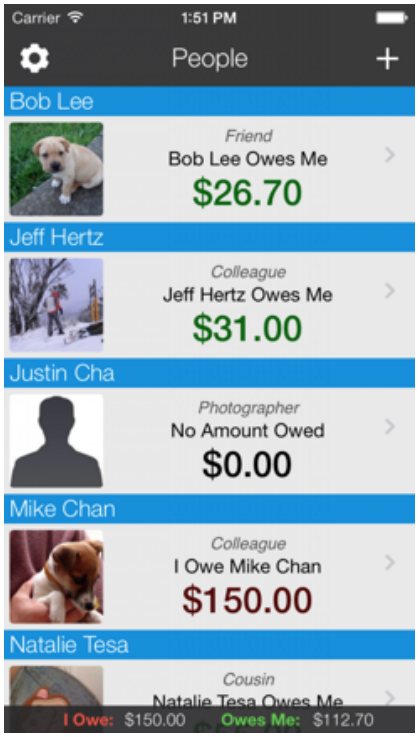
- Ability to make payments via a 3rd party website such as PayPal

[2] "Cash Lender," http://sockii.com/portfolio/cash-lender/, accessed November 27, 2013.

Screenshots:







6.1.2 SpotMe

SpotMe is an iPhone application and web service for sharing expenses within a group.[3] Users can add friends to a group and split bills as the user travels, lives or works together. SpotMe notifies friends of new expenses and updates the balance owed between users. It automatically keeps track of everyone's payments in a group. SpotMe's main functionality includes:

- Facebook registration
- Lending to an email address
- Split bill between friends with "Create Group"
- Invite friends
- Automatic and manual email notifications
- Rating: 4+ / 5 Stars on iTunes
- Free to use

What SpotMe offers that Lend.me does not:

- Borrower does not need to join app (can lend to an email address)

What Lend.me will offer that SpotMe does not:

- Facilitates payment and repayment

[3] "About SpotMe," http://www.getspotme.com/about, accessed December 4, 2013.

Screenshots:

6.1.3 ZimpleMoney

ZimpleMoney is a web-based application that makes it easy to manage and track intra-family loans, business loans or recurring payments. ZimpleMoney offers automated billing and collection, automated payment alerts and reminders, accounting and posting and the depositing of collected funds.[4]

In addition, there are multiple payment options for various types of users. Examples include the "Business Plan," for businesses or active individuals who want to automate a growing loan portfolio, the "Rich Uncle" plan, for the "go-to" person in the family for loans and financial help, or the "Family & Friends" plan for parents eager to help their children buy their first car or finance their education.

ZimpleMoney's main functions include:

- Cloud hosted
- Data is password protected
- Identity verification
- Users can track lenders and borrowers
- Multiple options for lending money
- Multiple types of loans recognized (amortized, interest only, etc.)

What ZimpleMoney offers that Lend.me does not:

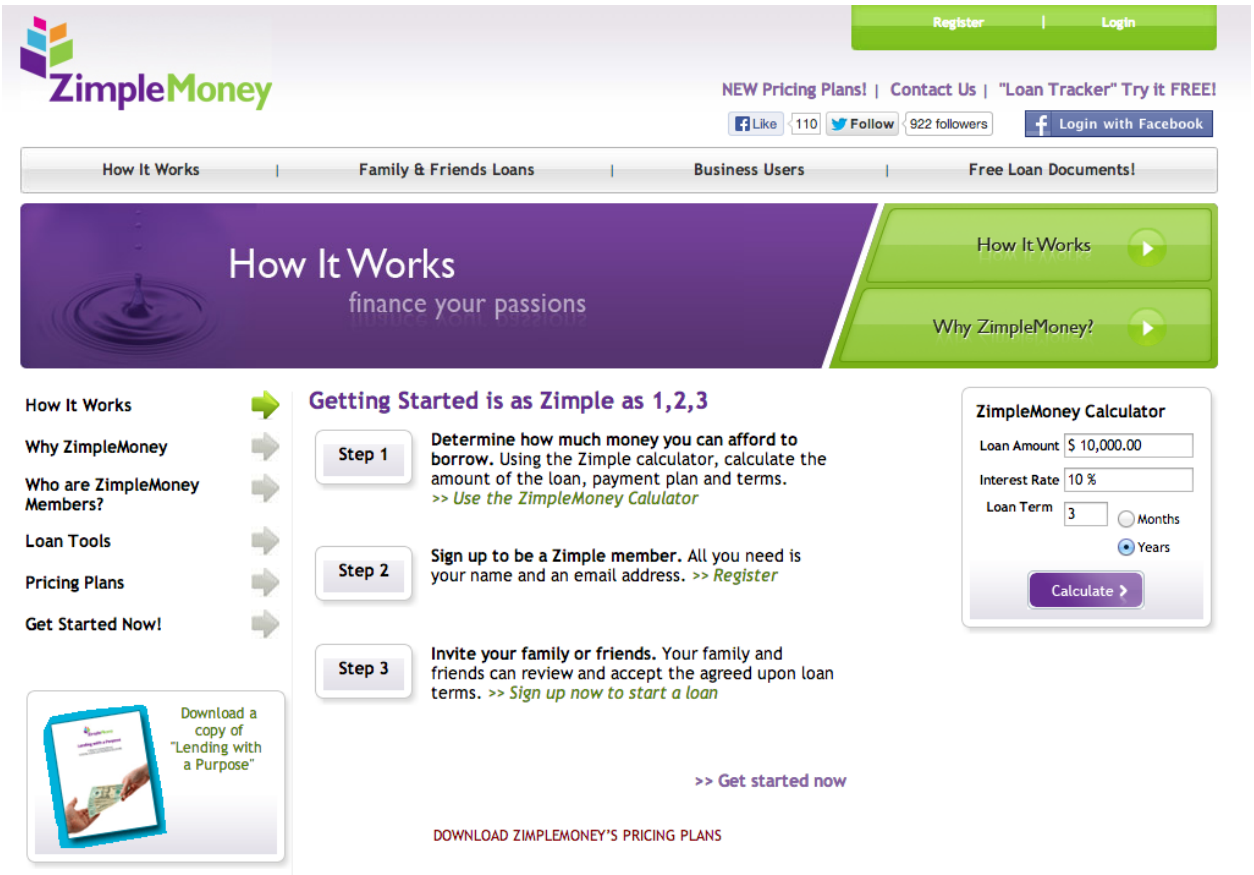
- Identity verification

What Lend.me will offer that ZimpleMoney does not:

- Elegant, web 2.0, user interface
- Mobile ready

[4] "Family & Friends Loans: helping the kids buy their first home," http://www.zimplemoney.com/Family-Friend-Loan.aspx, accessed December 4, 2013.

Screenshots:



6.1.3 Venmo

Venmo is a mobile application for the iPhone, Android, and Blackberry that allows users to pay their friends instantly. It provides bank-grade security systems and data encryption to protect user's data and prevent unauthorized transactions or access to a user's personal or financial information.[5]

Users choose a friend via phone, email or Facebook to send money to. Sending money is as easy as typing the amount and a message.

Sending money on Venmo is free, provided the user uses a bank account, supported debit card, or Venmo balance to fund the payment. Credit cards cost 3% of the transaction.

What Venmo offers that Lend.me does not:

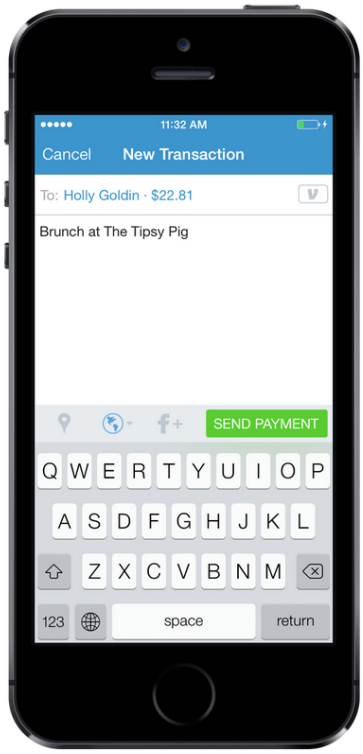
- Ability to store money in a Venmo account until the user "cashes out"

What Lend.me will offer that Venmo does not:

- All lending options such as loan tracking, notifications, etc.

[5] "Our App: Overview," https://venmo.com/info/product, accessed December 4, 2013.

Screenshots:

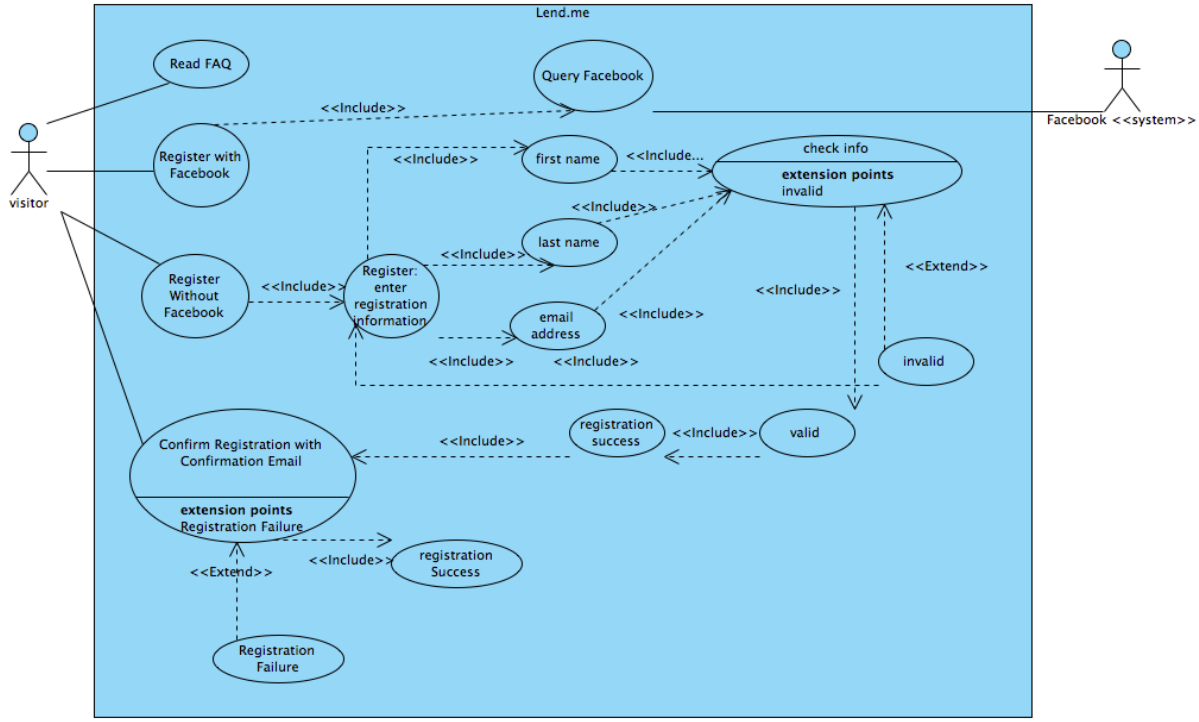


Pay your friends instantly.

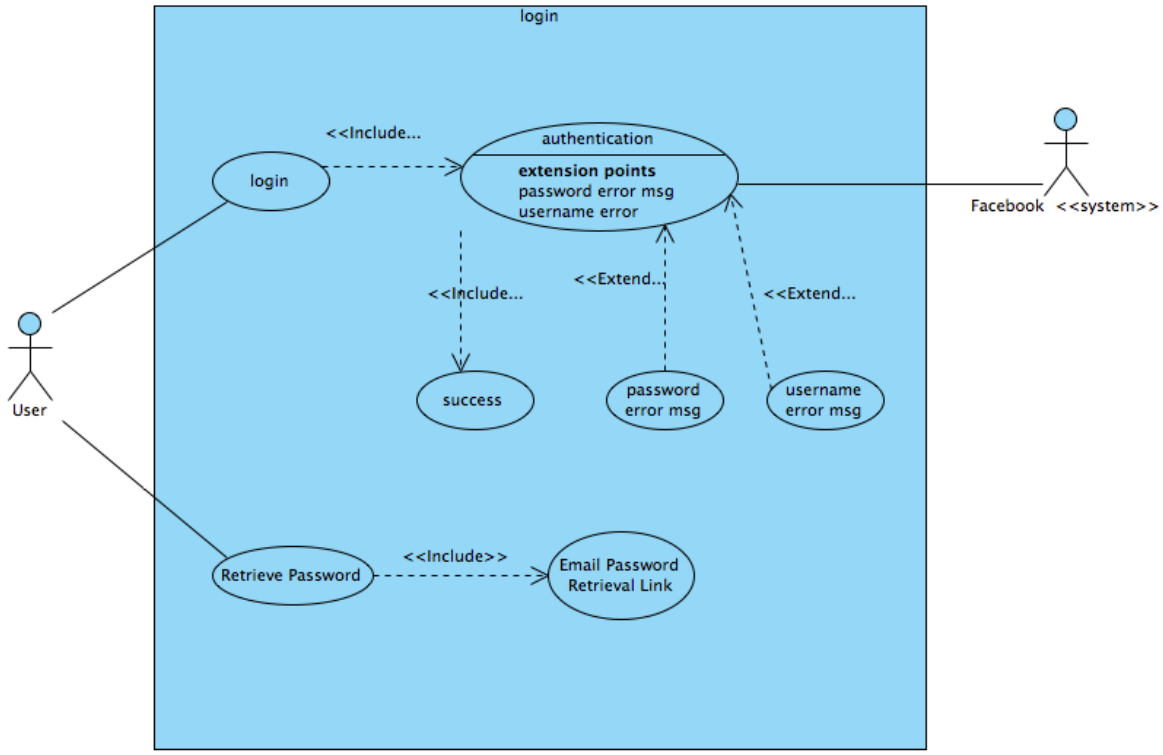
1. Choose a friend
   via phone, email, or facebook
2. Type amount and message
3. Choose sharing options

6.2 Use Case Diagrams

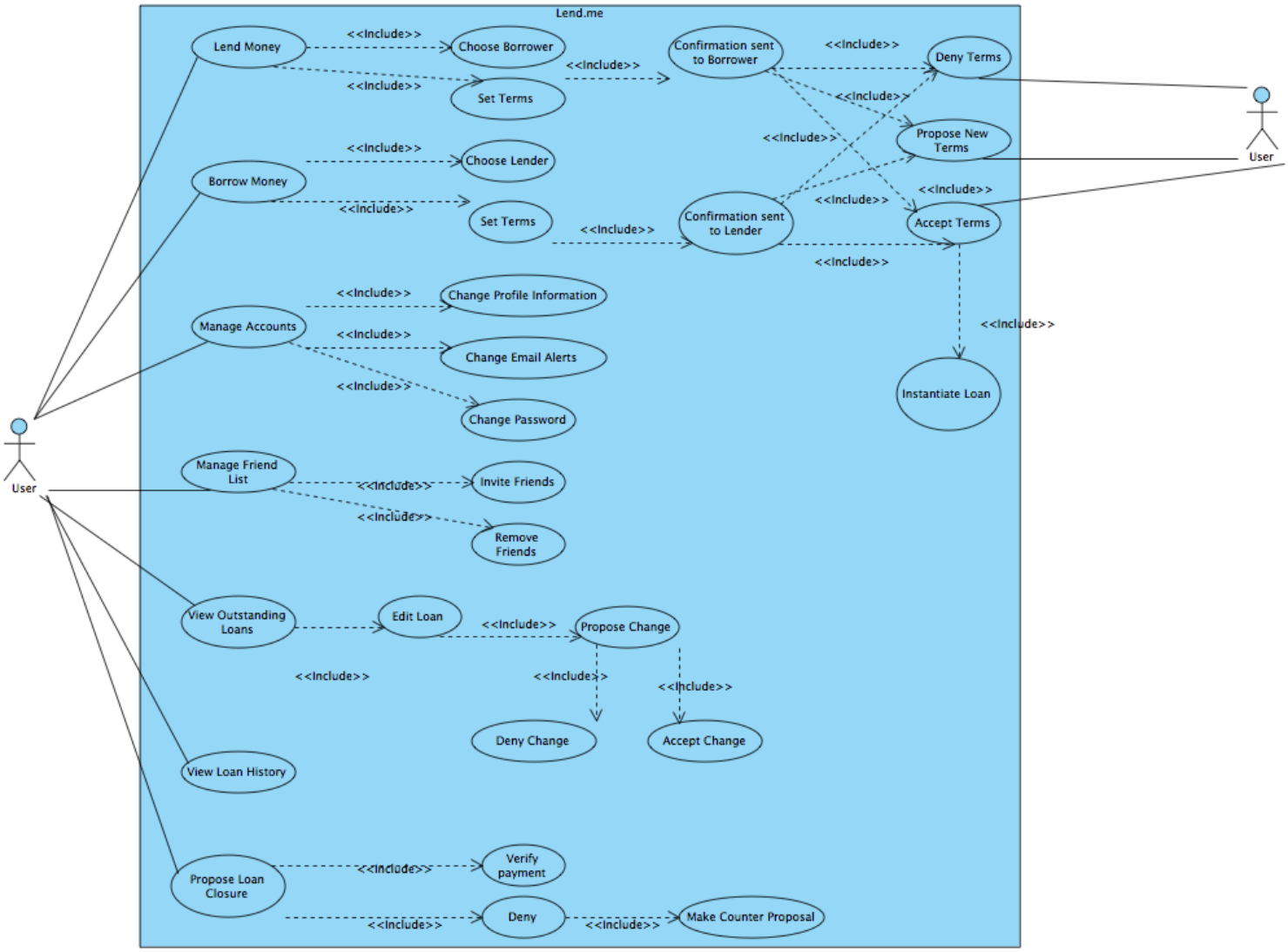
6.2.1 Visitor Use Cases



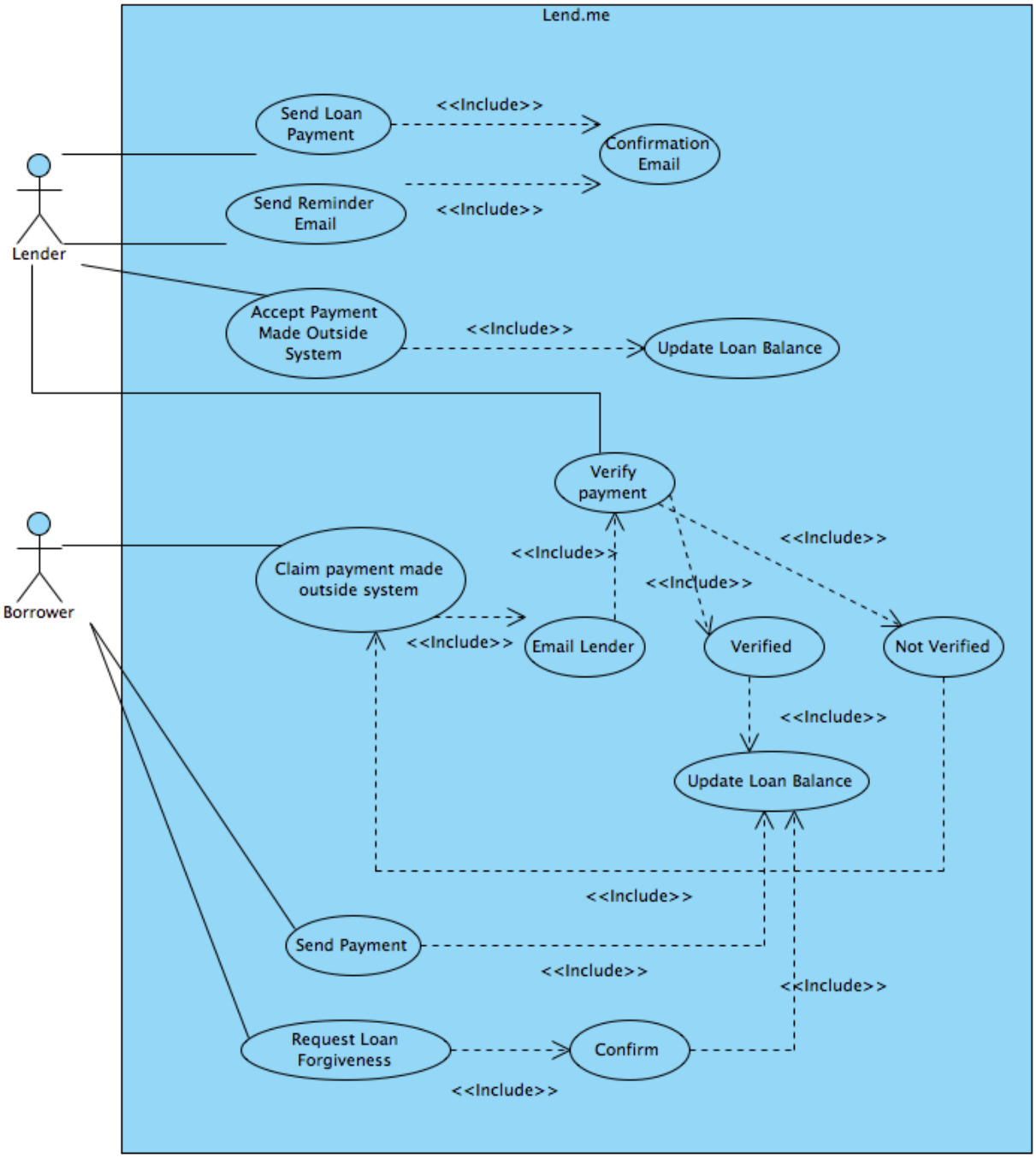
6.2.2 User Login Use Cases



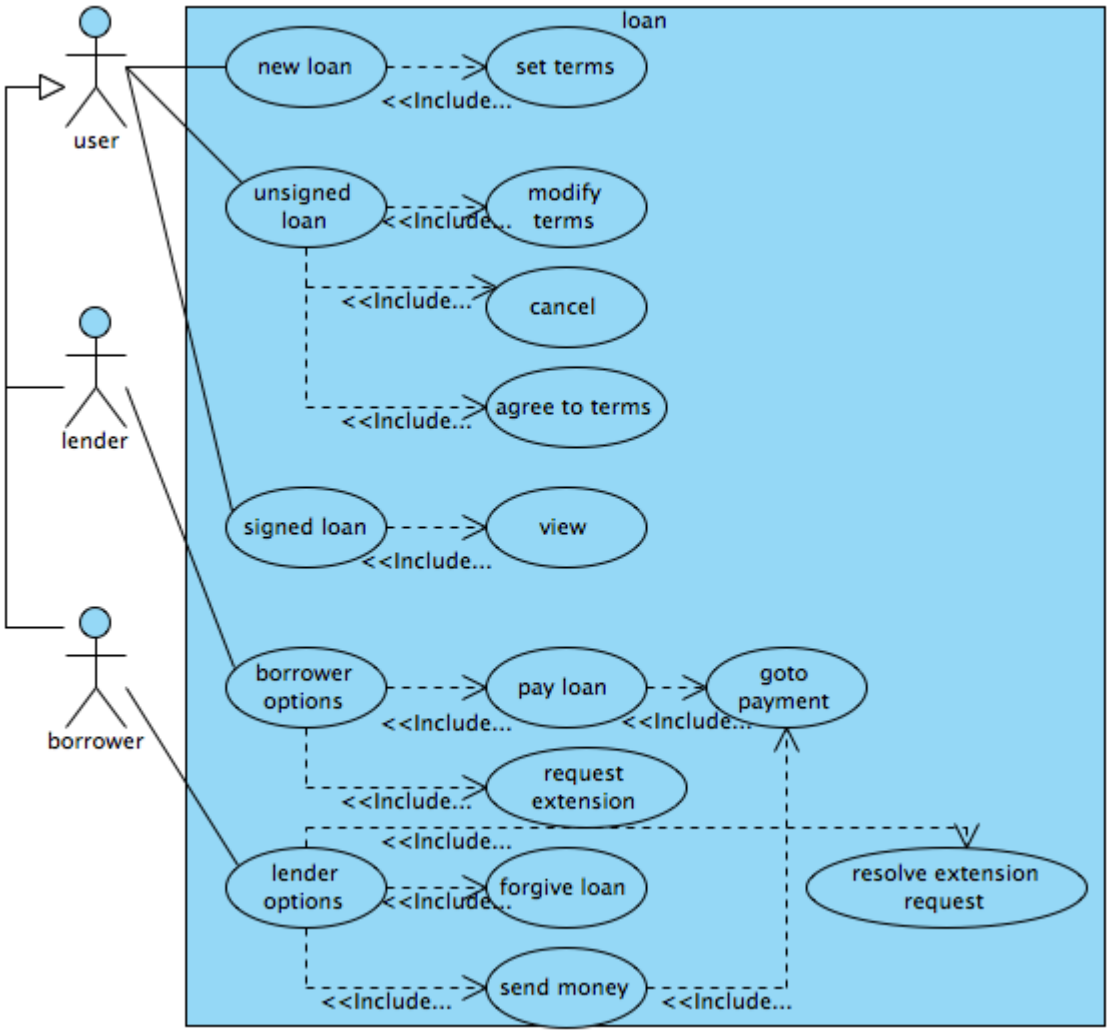
6.2.2 User Use Cases



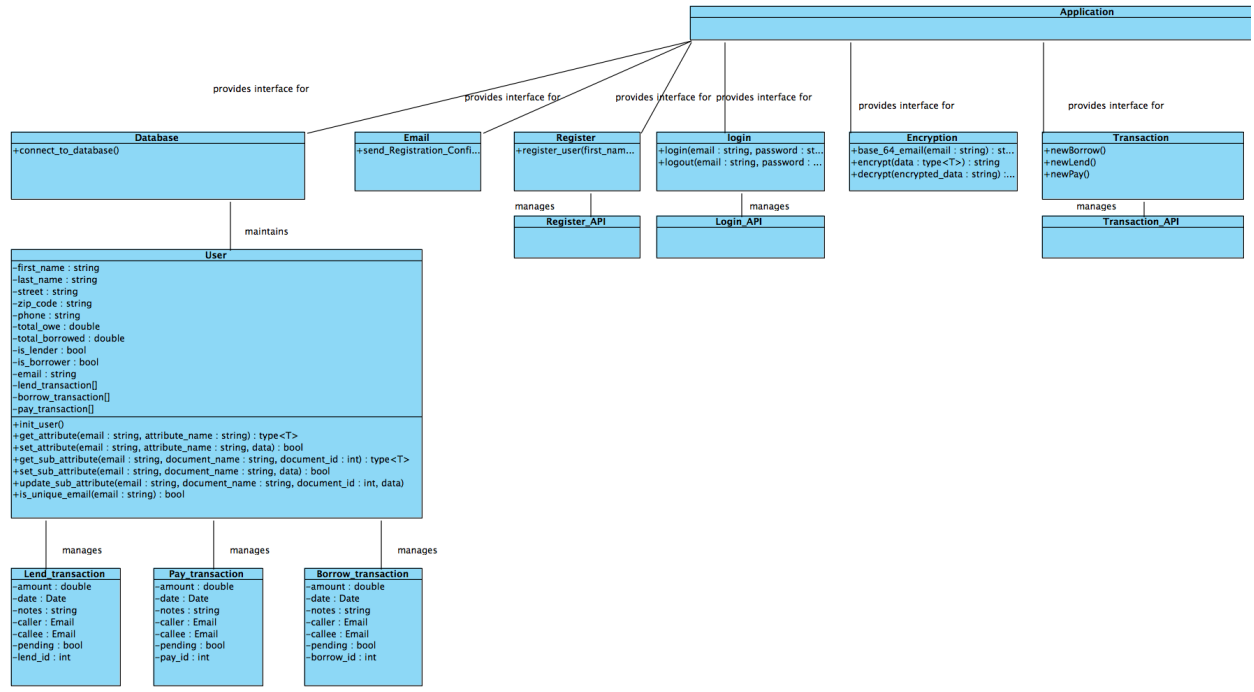
6.2.3 Lender & Borrower specific Use Cases



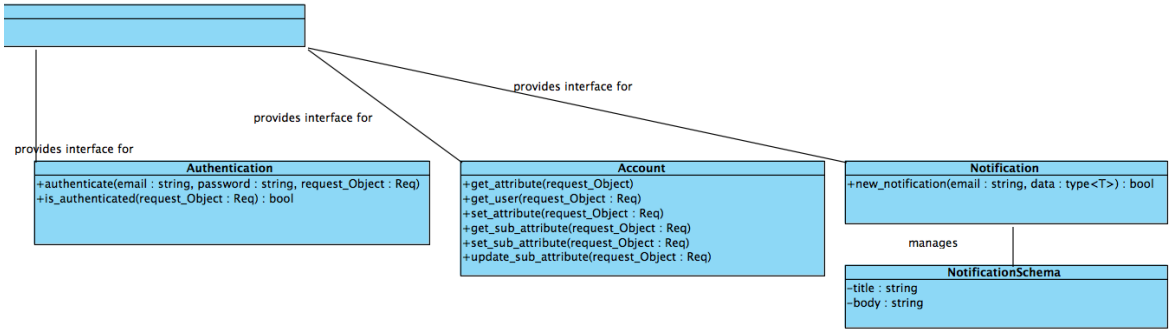
6.2.4 Loan Use Cases



6.3 Class Diagrams

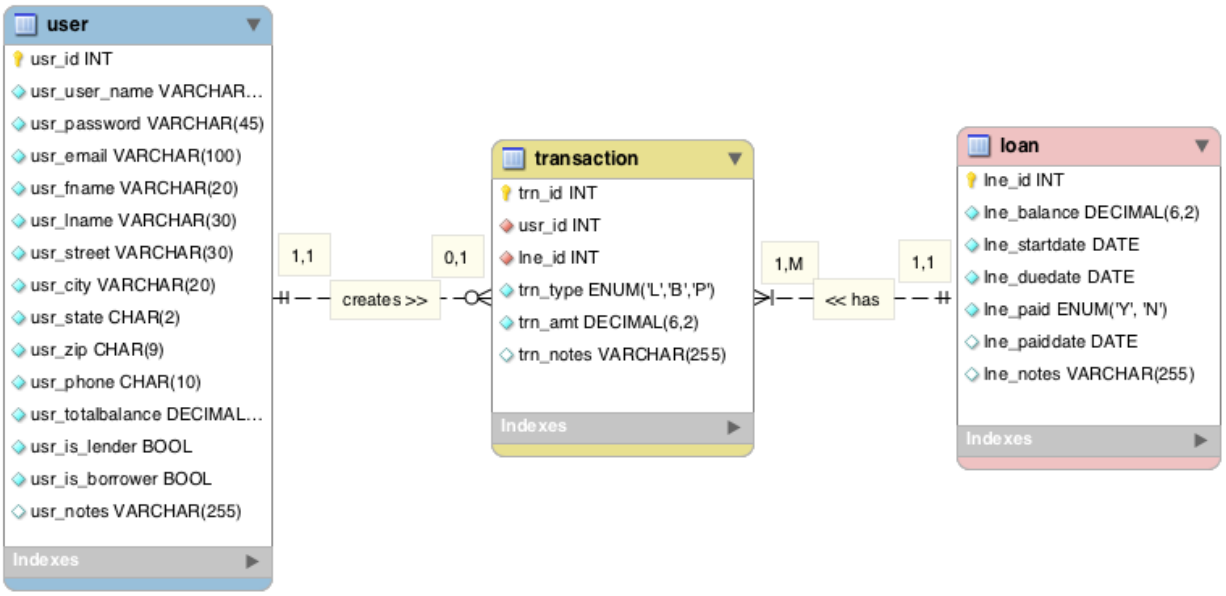


(continued)



6.4 Database Structure

6.4.1 Database modeled in MySQL Workbench



6.4.2 Data Dictionary

Table Name	Attribute Name	Contents	Type	Format	Range	Required	PK or FK	FK Reference Table
user	usr_id	User Identification	INT	123456	0-9	Y	PK	
	usr_user_name	Username	STRING	usrName	a-z, 0--9	Y		
	usr_fname	User First Name	STRING	First	a-z, 0-9	Y		
	usr_lname	User Last Name	STRING	Last	a-z, 0-9	Y		
	usr_street	User Street	STRING	123 Street	a-z, 0-9	Y		
	usr_city	User City	STRING	City	a-z, 0-9	Y		
	usr_state	User State	STRING	AZ	a-z	Y		
	usr_zip	User Zip Code	STRING	12345	0-9	Y		
	usr_phone	User Phone Number	STRING	5553211234		N		
	usr_email	User Email	STRING	xxxx@xxx.com	a-z, 0-9	Y		
	usr_totalbalance	User's Total Balance	DOUBLE	1234567.89	0-9	Y		
	usr_is_lender	User is a lender	BOOL	TRUE	TRUE or FALSE	Y		
	usr_is_borrower	User is a borrower	BOOL	TRUE	TRUE or FALSE	Y		
	usr_notes	User Notes	STRING	Notes	a-z, 0-9	N		
transaction	trn_id	Transaction ID	INT	123456789	0-9	Y	PK	
	usr_id	User Id	INT	123456789	0-9	Y	FK	user
	lne_id	Loan Id	INT	123456789	0-9	Y	FK	loan
	trn_type	Transaction Type	ENUM('L', 'B')	L	L, B	Y		
	trn_date	Transaction Date	DATE	2013-10-06	0-9	Y		
	trn_amt	Transaction Amount	DOUBLE	123456.12	0-9	Y		
	trn_notes	Transaction Notes	STRING	Notes	a-z, 0-9	N		
loan	lne_id	Loan Id	INT	123456789	0-9	Y	PK	
	lne_balance	Loan Balance	DOUBLE	1234567.12	0-9	Y		
	lne_startdate	Loan Start Date	DATE	2013-10-06	0-9	Y		
	lne_duedate	Loan Due Date	DATE	2013-10-06	0-9	Y		
	lne_paiddate	Loan Paid Date	DATE	2013-10-06	0-9	N		
	lne_paid	Loan Paid Back	BOOL	TRUE	TRUE or FALSE	Y		
	lne_notes	Loan Notes	STRING	Notes	a-z, 0-9	N		

6.4.3 Database modeled as a JSON document for MongoDB

6.4.3.1 User Schema

```
{
    "_id": {
       "$oid": ""
   },
    "modified": {
       "$date": ""
   },
   "created": {
       "$date": ""
   },
   "__v": “”

   "email": "",
   "password": "”,
   “street”: “”,
   “zipcode”: “”,
   “phone”: “”,
   “total_owed”: “”,
   “total_borrowed”: “”,
   “is_lender”: “”,
   “is_borrower”: “”,
   "phone": "",
   "last_name": "",
   "first_name": "",

   “lend_transaction”: [],
   “borrow_transaction”: [],
   “pay_transaction”: []
}
```

6.4.3.2 Lend Transaction Schema

```
{
    "_id": {
       "$oid": ""
   },
    "modified": {
       "$date": ""
   },
   "created": {
       "$date": ""
   },
   "__v": “”

   "amount": "",
   "date": "”,
   “notes”: “”,
   “caller”: “”,
   “callee”: “”,
   “pending”: “”,
   “lend_id”: “”,
}
```

6.4.3.3 Pay Transaction Schema

```
{
    "_id": {
       "$oid": ""
   },
    "modified": {
       "$date": ""
   },
   "created": {
       "$date": ""
   },
   "__v": “”

   "amount": "",
   "date": "”,
   “notes”: “”,
   “caller”: “”,
   “callee”: “”,
   “pending”: “”,
   “lend_id”: “”,
}
```

6.4.3.4 Borrow Transaction Schema

```
{
    "_id": {
       "$oid": ""
   },
    "modified": {
       "$date": ""
   },
   "created": {
       "$date": ""
   },
   "__v": “”

   "amount": "",
   "date": "”,
   “notes”: “”,
   “caller”: “”,
   “callee”: “”,
   “pending”: “”,
   “lend_id”: “”,
}
```