# Mortgage Deals

Jesus Contreras

## Table of Contents

## Contents

# 1. Introduction

## 1.1 Statement of Problem Area (brief, non-technical)

Since most people cannot afford to buy a house in cash generally the first step to buying a house is to shop around for a mortgage. This simple process can become a real hassle once people start looking for mortgages and get into contact with lenders. Since the advent of the internet there have been many mainstream processes that have been simplified online but mortgage lending seems to be one that is lagging. The technologies in use today to shop for mortgages are just like the looking through a rolodex, where you look for lenders, contact them and hope for a good mortgage loan. Considering how much technology has advanced in recent years, the models of today's online mortgage lending are inappropriate and inefficient.

## 1.2 Previous and Current Work, Methods and Procedures (representative)

Modern day mortgage lending practices have been virtually unchanged since the federal government created the idea. Most people will go through the motions where they will put their credit out there to prospective lenders, get quotes (the recommendation is at least 6), lock the loan, and talk to a broker in some cases.

## 1.3 Brief Project Description (overview of new, extended or different functions, structure or operation)

MortgageDeals is a webservice that makes it easier for a person looking for a loan to actually get one. Instead of shopping for mortgage loans, the user of MortgageDeals will actually be shopped by lenders, creating a competitive environment that should breed low mortgage rates for the user. The user will create a mortgage profile with all of the relevant information that is usually required to qualify for a loan (income, credit score, credit history etc.) and based off of the provided data the user will be shopped. The user will be shopped by lenders who will review the persons mortgage profile, the lender will be able to purchase a user's "contact information" if desired.

## 1.4 Purpose/Objectives/justification of Project (theoretical, practical, or educational impacts on users)

MortgageDeals isn’t a web service with extraordinary flash however it is a practical application to a real world problem. The purpose of MortgageDeals is to help people looking for a mortgage to find a really good mortgage loan, while simplifying the process of doing so. MortgageDeals will be a service where a user can have all of the mortgage rates they have been looking for in one consolidated portfolio, making it easier to compare rates and be organized. MortgageDeals is also a tool for lenders. Lenders will be able to find clients of their choosing by looking at people’s mortgage portfolios and then deciding if they wish to contact them.

# 2. System Functional Specification

## 2.1 Functions Performed (itemize and describe)

NOTE: This section is in very high level language for abstraction and clarity.

### Borrowers Portal

| Functions Performed | Description (Brief) |
|---|---|
| **Launch web application** | The web service loads and executes on website load-up. |
| **Choose loan** | A user is able to select a mortgage loan from a form. |
| **Choose property type** | A user is able to select their property type (single house, condo, etc…) |
| **Provide location information** | A user is able to input which state and city they are searching for property in. |
| **Provide lending info** | A user is able to select an amount (a range of money) they wish to borrow. They are also able to input their desired down payment. |
| **Provide credit info** | A user can provide their credit info as good, bad, or fair. |
| **Provide financial info** | A user can provide their current income. |
| **Provide contact info** | If a lender decides to purchase a user's personal information, through the lender web page the user's contact info. will be displayed. |
| **Submit request** | After user inputs all of his/her information the database will save each user unique profile, there will be a call to the server to process the information, which will then be saved to the user's unique profile. |
| **Lenders for service.** | A lender will give the user a mortgage rate, then the user will be able to shop other lender's bids (a lender mortgage offer is considered a bid). |
| **Borrowers can see all Lenders info.** | After a user logs in, if there are bids the user will be able to see that a lender has sent them a mortgage bid. |
| **Send notification if there is a new offer(s).** | There will be an email notification sent to the user if there is a new offer extended to him/her, this will also be updated in the web-service page. |
| **manage account** | There will be forms to update, edit the user information as well as the ability to close the account of a user. |

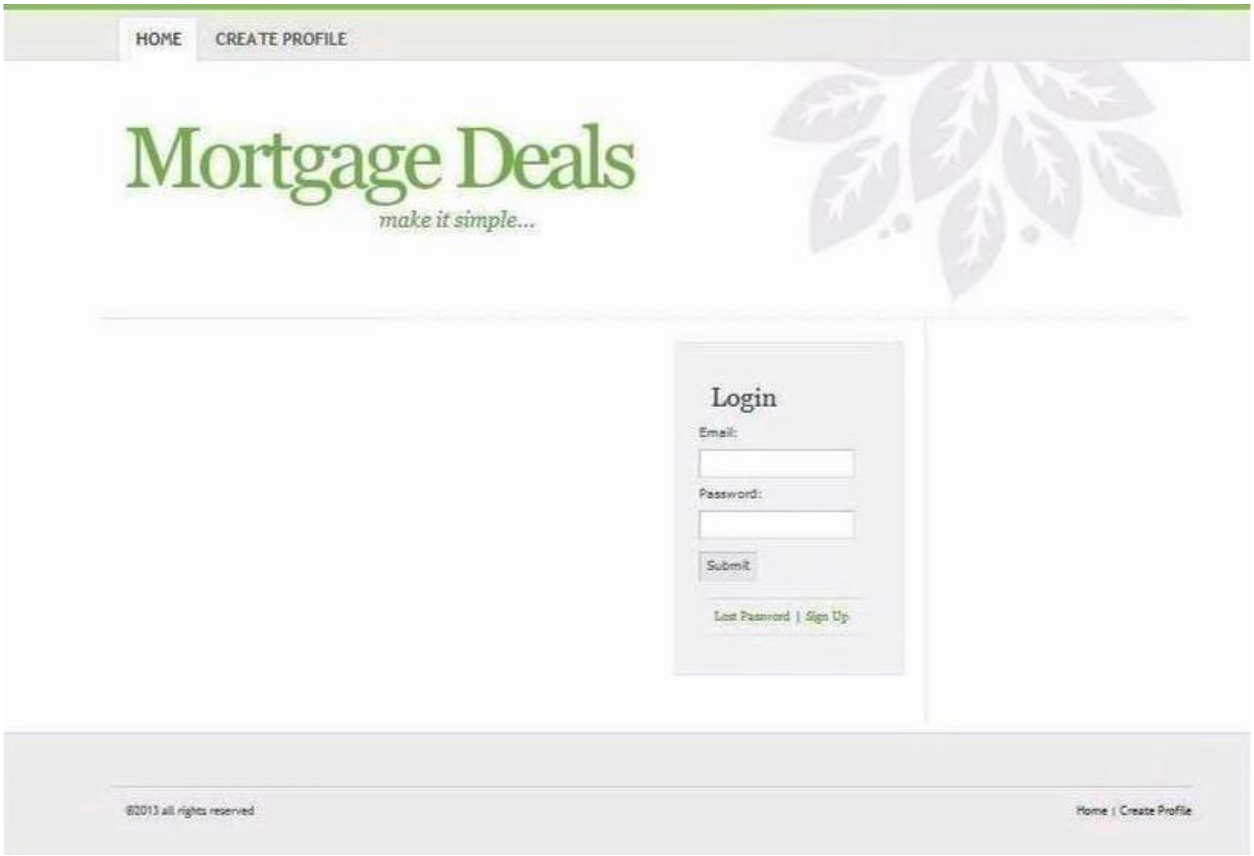
## Lender/Bank Portal

| Functions Performed | Description (Brief) |
|---|---|
| **Login to account** | After successfully having an account created the bank/lender will be able to login to their account. |
| **Manage account** | A bank/lender will be able to update, edit and close account, they will also be able to create a sub account. |
| **View Mortgage Profiles** | A bank/lender will be able to view mortgage profiles of the users looking for a mortgage. |
| **Deposit paid credit via credit card.** | A bank/lender will be able to view mortgage profiles of the users looking for a mortgage. |
| **Credit updating.** | The account amount for bank/lender is updated according to the lenders purchase activity. |
| **Credit checking.** | MortgageDeals will check automatically if a bank/lender has enough credit to purchase a user's personal contact info. |
| **Notification message(s)** | There will be a notification message letting the bank/lender that they have less than $25 credited to their account. |
| **Private offers** | Bank/lenders will not be able to see each other's competing rates/offers. |
| **Input offer** | Bank/lenders can make a mortgage offer, and with the input provided by the users mortgage profile along with the interest rate the bank/lender selects, MortgageDeals server is prompted to make a call to the Zillow API for a automated calculation of a mortgage rate (using the aforementioned data). |

## Administrator Portal

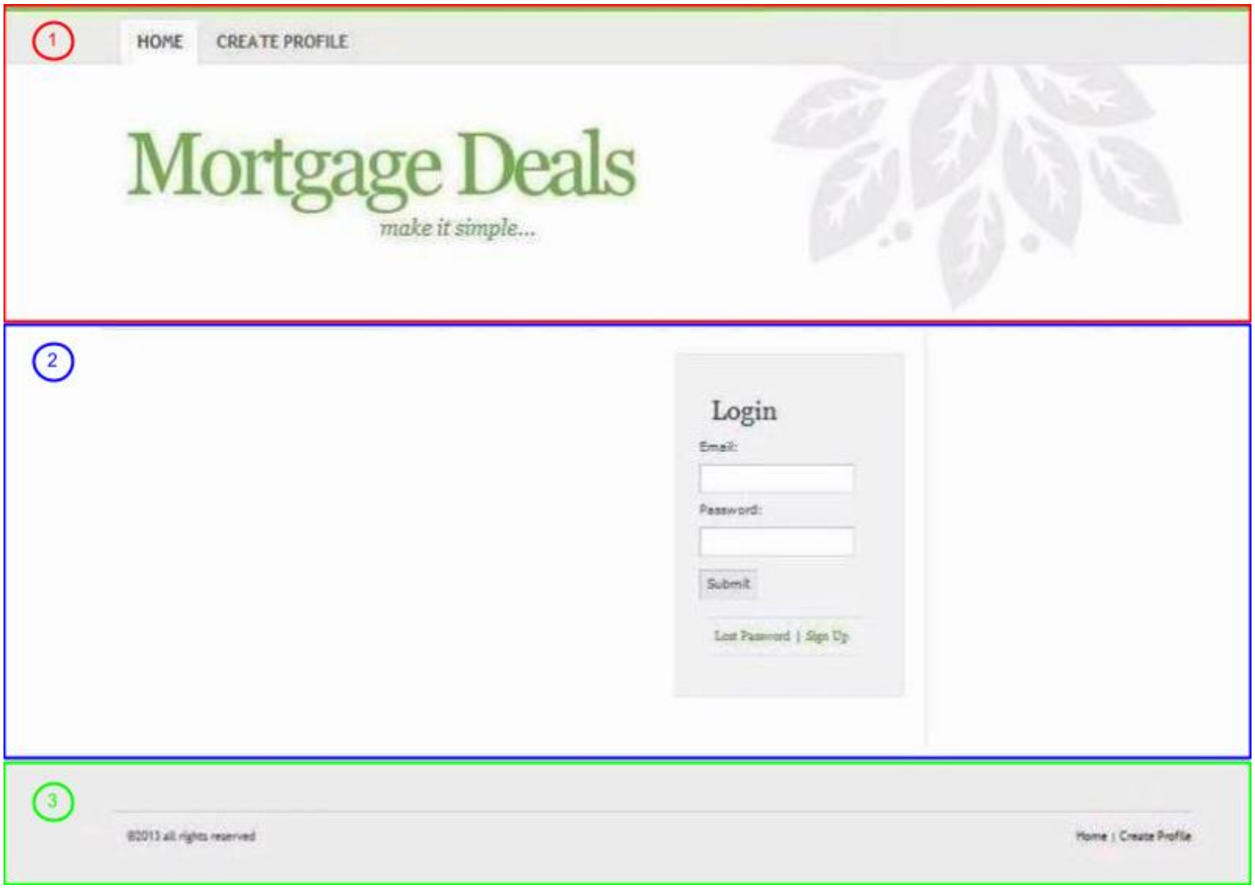
| Functions Performed | Description (Brief) |
|---|---|
| **Data Management** | Manage all user information. Ability to create, update, edit, and delete all the user information. |

## 2.2 User Interface Design



MortgageDeals has a very simple and responsive User Interface. It is clean and composed of 3 parts, a header, content section, and a footer. Our main objective was to maintain simplicity, directness, and forgiveness. Through the simplicity of our design we provide an interface that is intuitive to the users, provides the user with the easiest interaction possible which will ultimately help the users complete their tasks.

2.2 User Interface Design



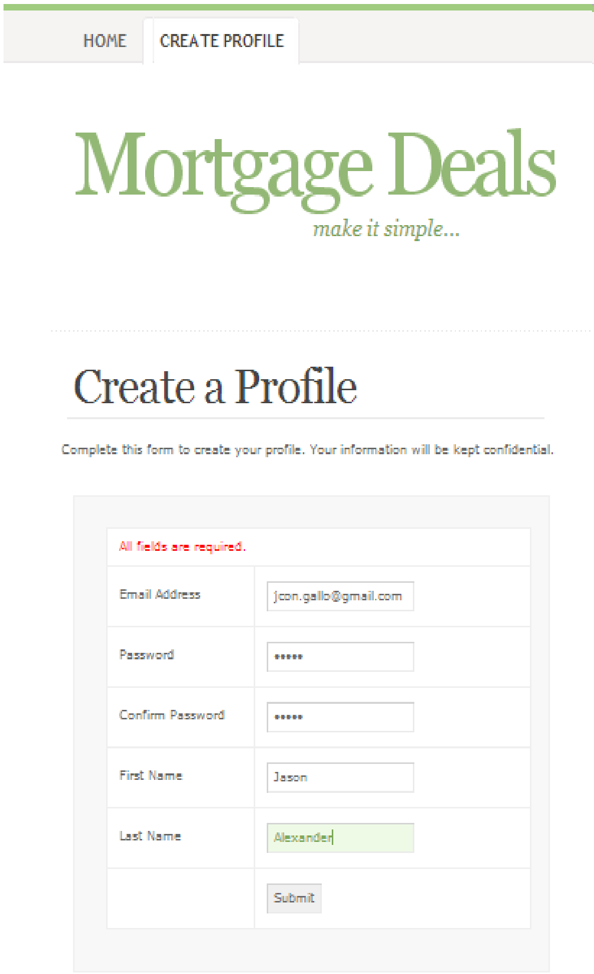
1. Header:
   The header for this particular page contains the website logo and a design. There is also a navigation bar towards the top with navigation tabs. Navigation tabs will redirect the user to a page that is indicated by the title of the tab. typically, the header has a navigation bar, it's a way for the user to easily identify how to navigate the website and separates direct navigation from the displayed content.

2. Content:
   The content sections of the pages are usually reserved for content, or data that is transferred from the server to the user. For the websites homepage the user will have the ability to login, sign up for the service or request a new password.

3. Footer:
   The footer serves as extra padding for the web page. The footer also serves an aesthetic tool although we also use it as a means to navigate back to a common page once the user has reached the bottom of the webpage.

   It's important to note that the user interface for MortgageDeals' navigation pages has a consistent header, content, footer layout. It's also important to note that these containers will also contain an array of user controls through which a user may manipulate, request and access data.

## 2.2 User Input Preview

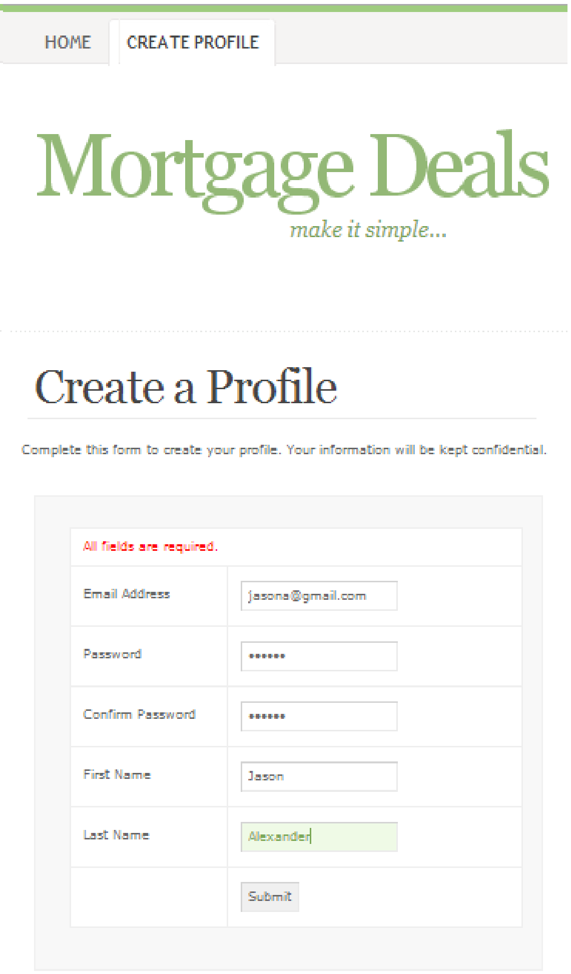


Create a Profile

Complete this form to create your profile. Your information will be kept confidential.

All fields are required.

| | |
|---|---|
| Email Address | jcon.gallo@gmail.com |
| Password | ••••• |
| Confirm Password | ••••• |
| First Name | Jason |
| Last Name | Alexander |
| | Submit |



Create a Profile

Complete this form to create your profile. Your information will be kept confidential.

All fields are required.

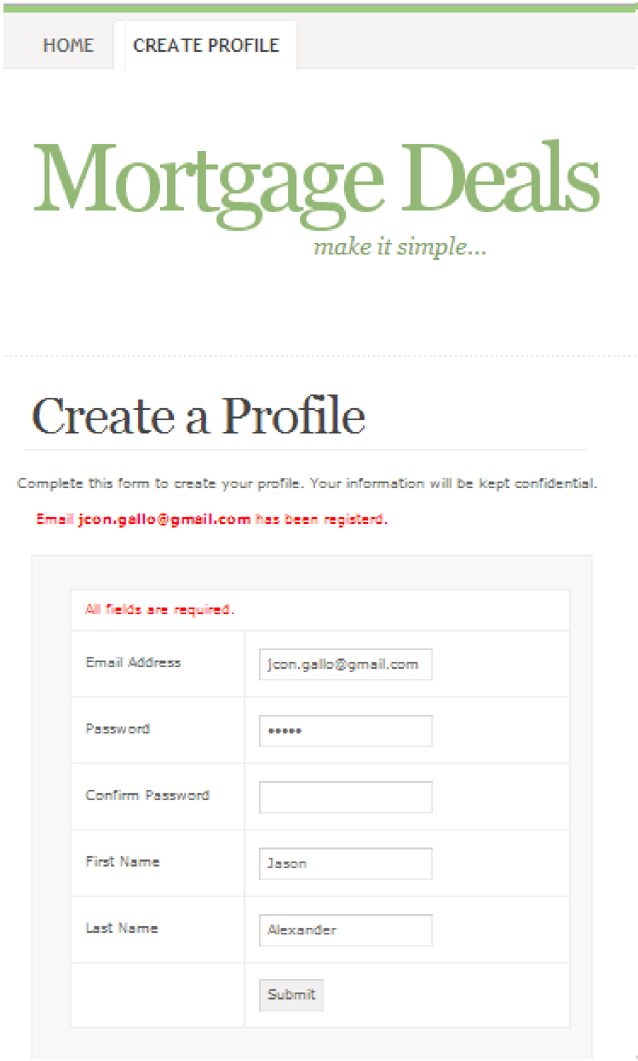
| | |
|---|---|
| Email Address | jasona@gmail.com |
| Password | •••••• |
| Confirm Password | •••••• |
| First Name | Jason |
| Last Name | Alexander |
| | Submit |

Used email account.

New email account

MortgageDeals offers many types of user inputs. Text (boxes) and dropdown list items are typical for a MortgageDeals user input. As far as our forgiveness policy is concerned there is error checking that occurs every time a submit button is clicked, there is input error checking (validation) as well as database checking (repeated use of a single email is not allowed thus will be checked for). In this particular example a user fills out a form via text boxes and a submit button, on the left the user inputs an email address that has already been registered to the MortgageDeals database, while on the right the user inputs an email that has not been registered to the MortgageDeals database.

## 2.3 User Output Preview



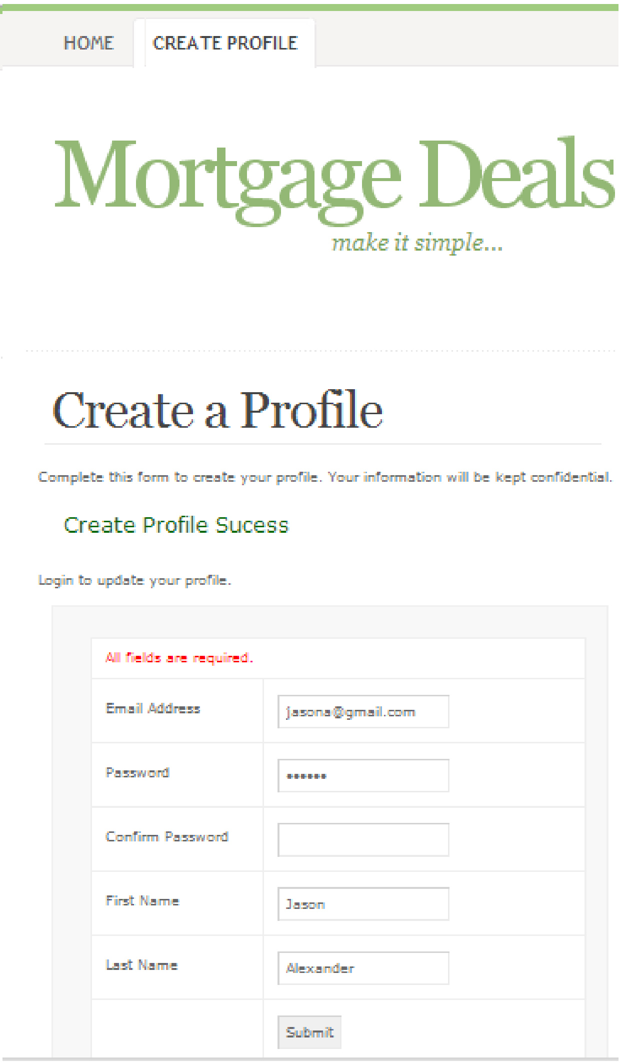
Create a Profile

Complete this form to create your profile. Your information will be kept confidential.

Email jcon.gallo@gmail.com has been registerd.

| | |
|---|---|
| All fields are required. | |
| Email Address | jcon.gallo@gmail.com |
| Password | ••••• |
| Confirm Password | |
| First Name | Jason |
| Last Name | Alexander |
| | Submit |

Used email account.



Create a Profile

Complete this form to create your profile. Your information will be kept confidential.

Create Profile Sucess

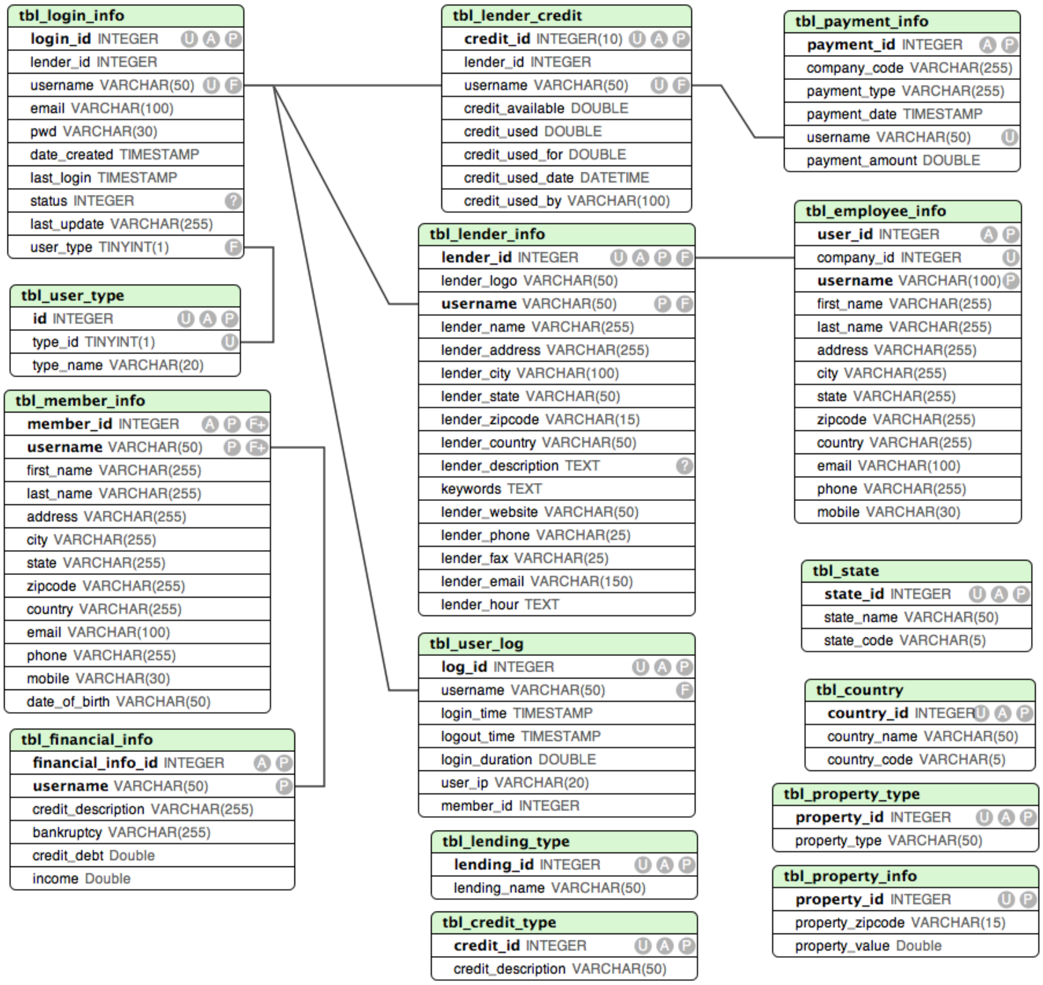
Login to update your profile.

| | |
|---|---|
| All fields are required. | |
| Email Address | jasona@gmail.com |
| Password | •••••• |
| Confirm Password | |
| First Name | Jason |
| Last Name | Alexander |
| | Submit |

New email account

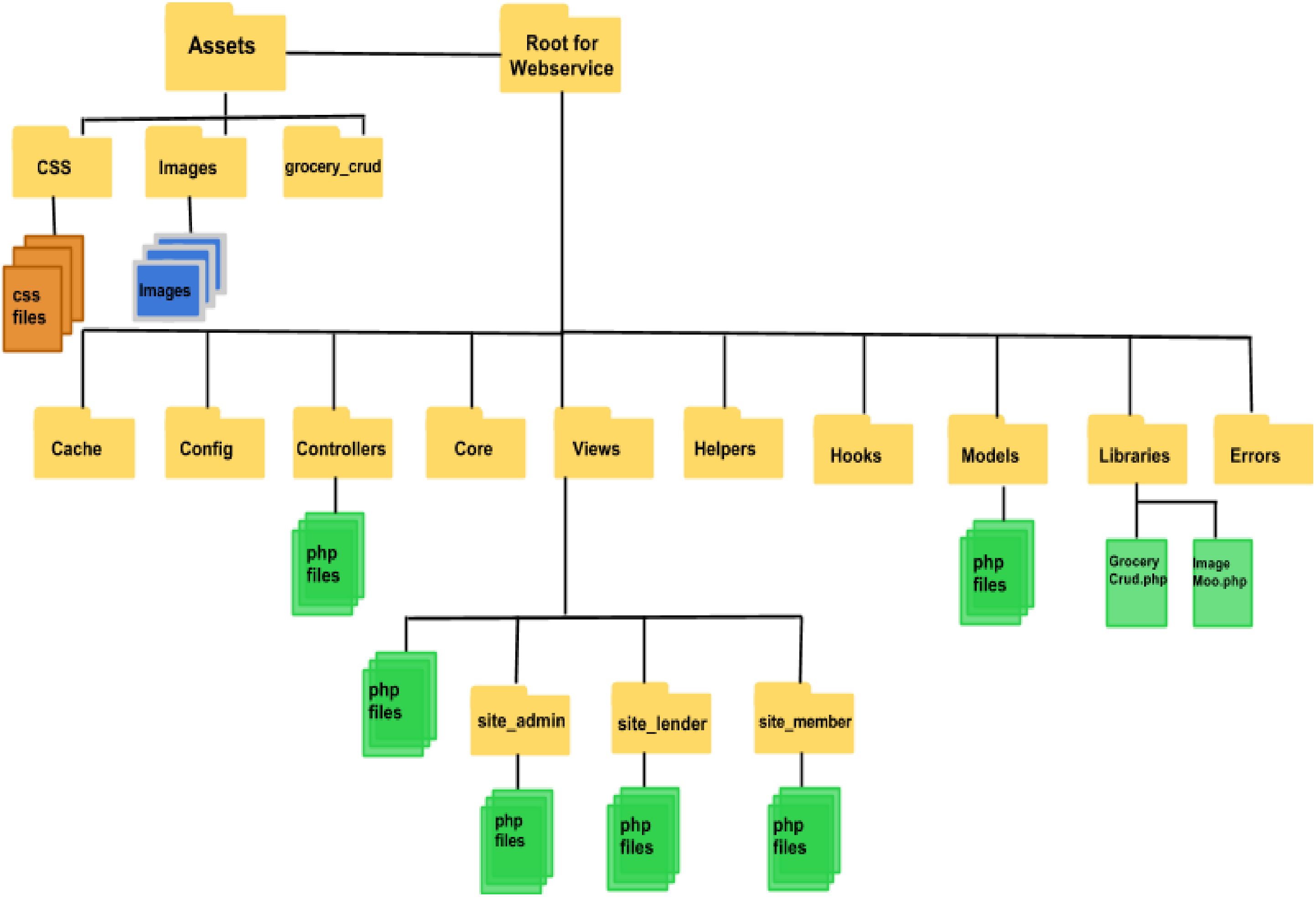
There are a variety of user outputs. User outputs are usually spelled out in text through alert message/notifications. In this particular instance to the left the MortgageDeals server found that the email used is already taken thus cannot be re-registered so the user gets a red message letting the user know that a user has already registered. Whereas in the instance to the right the server has found that the mail had not been registered before, thus a "create profile success notification".

## 2.4 System Data Base/File Structure Preview



The database schema displays all of the tables that are in use for MortgageDeals. The associations are denoted by a line connecting the tables. The P in the table rows denote a tables primary key(s), the F in the table rows denote a foreign key.

## 2.4 System Data Base/File Structure Preview



MortgageDeals' file structure follows that of a (CodeIgniter) MVC architecture. This allowed me to easily separate the business logic from user interface considerations. The web service root folder has a direct dependence to an assets folder that contains all of our pre-existing images and css designs. It's important to take into consideration that "models" represent data and establishes the business logic rules, "views" (as displayed above there is a separate child "view" folder for each of our user sites) contain elements of the UI and the "controllers" are in charge of communication between the models and views.

## 2.5 External and Internal Limitations and Restrictions

Internal Limitations:

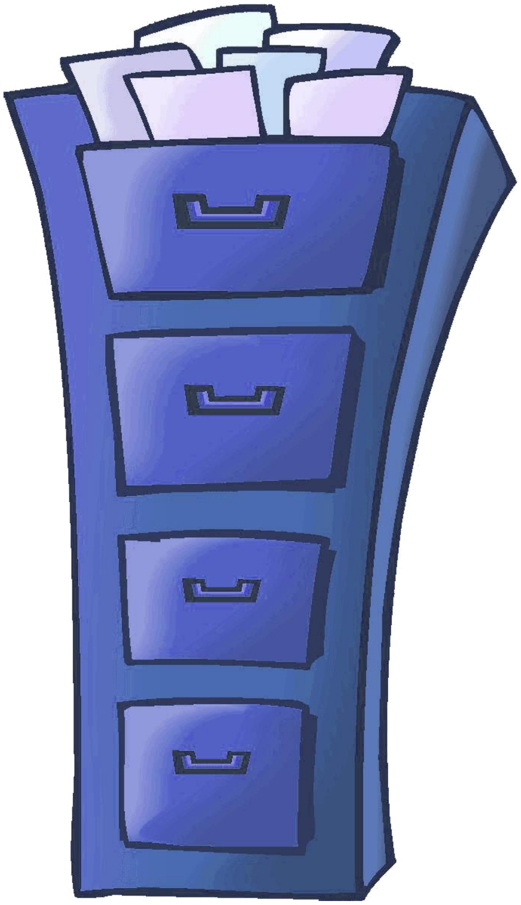
- Lack of Clarity – the code doesn’t necessarily achieve the most understandable documentation
- Concision – Code suffers duplication
    - There are duplicate files within the file structure files
- Time
    - Lack of time made it tough to complete desirable functionality, code tidiness

External Limitations:

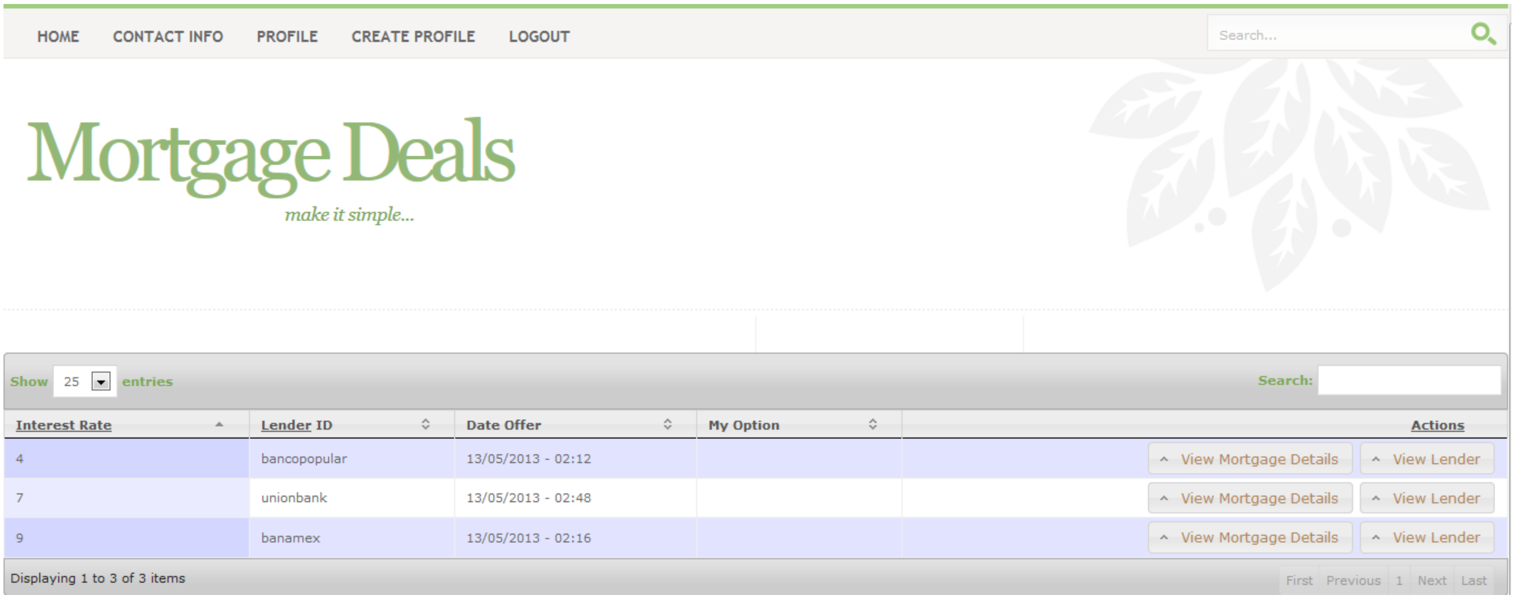
- Accuracy -No exact input of relevant data
    - No exact income input
    - No exact credit score input
    - No exact debt input
- Security
    - No data encryption for login information
    - Limited to a user’s session, once a session has expired there will be no access to the user’s data/homepage from web browser.

## 2.6 User Interface Specification

### 2.6.1 Interface Metaphor Model



MortgageDeals' metaphor model is a file cabinet, an interactive one at that. A file cabinet's files are organized by tabs, once someone reaches the intended tab, that person will look inside of the tabs designated area to access relevant content/information. MortgageDeals uses tabs for navigation as well, once the user clicks on a tab they will be re-directed and be able to access corresponding data. It's an interactive file cabinet because a user will have lenders bidding for their services, the bids occur/appear on the user end dynamically after a lender has made them an offer.



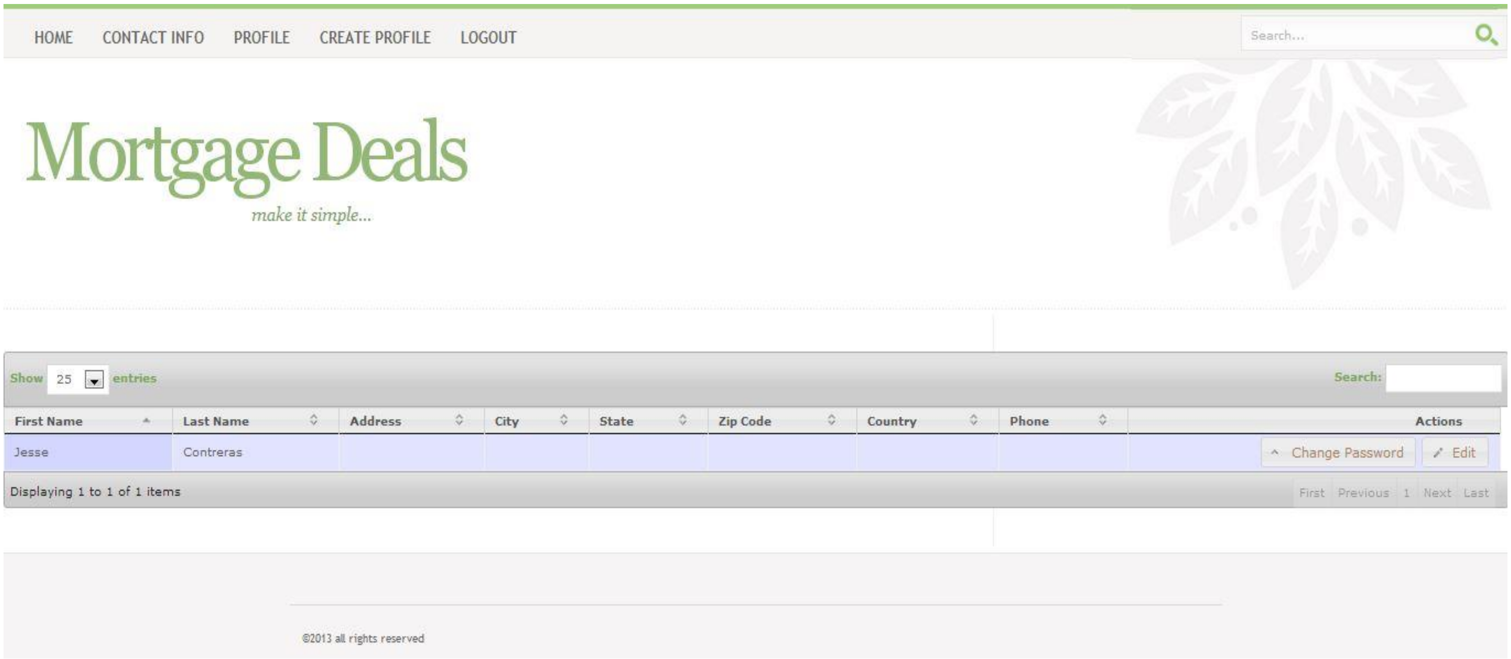
| Interest Rate | Lender ID | Date Offer | My Option | Actions |
|---|---|---|---|---|
| 4 | bancopopular | 13/05/2013 - 02:12 | | View Mortgage Details<br>View Lender |
| 7 | unionbank | 13/05/2013 - 02:48 | | View Mortgage Details<br>View Lender |
| 9 | banamex | 13/05/2013 - 02:16 | | View Mortgage Details<br>View Lender |

Displaying 1 to 3 of 3 items

Notice the navigation bar in the header has tabs just as a standard filing cabinet would. Clicking on a certain tab takes the user to a section of the site (or file cabinet) and once there the user can access a document of data.

### 2.6.2 User Screens/Dialog (There are 3 perspectives that MortgageDeals Provides)

## 1. User

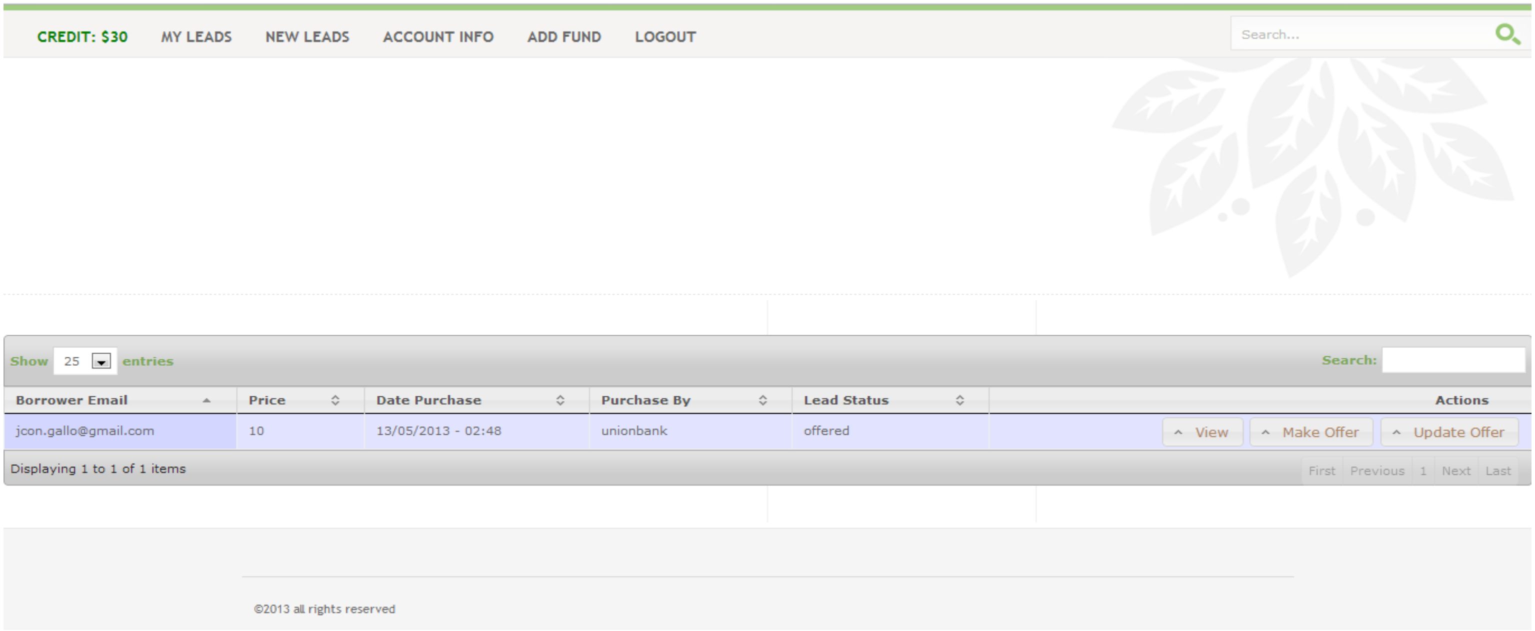


| First Name | Last Name | Address | City | State | Zip Code | Country | Phone | Actions |
|---|---|---|---|---|---|---|---|---|
| Jesse | Contreras | | | | | | | Change Password Edit |

Displaying 1 to 1 of 1 items

©2013 all rights reserved

## 2. Lender

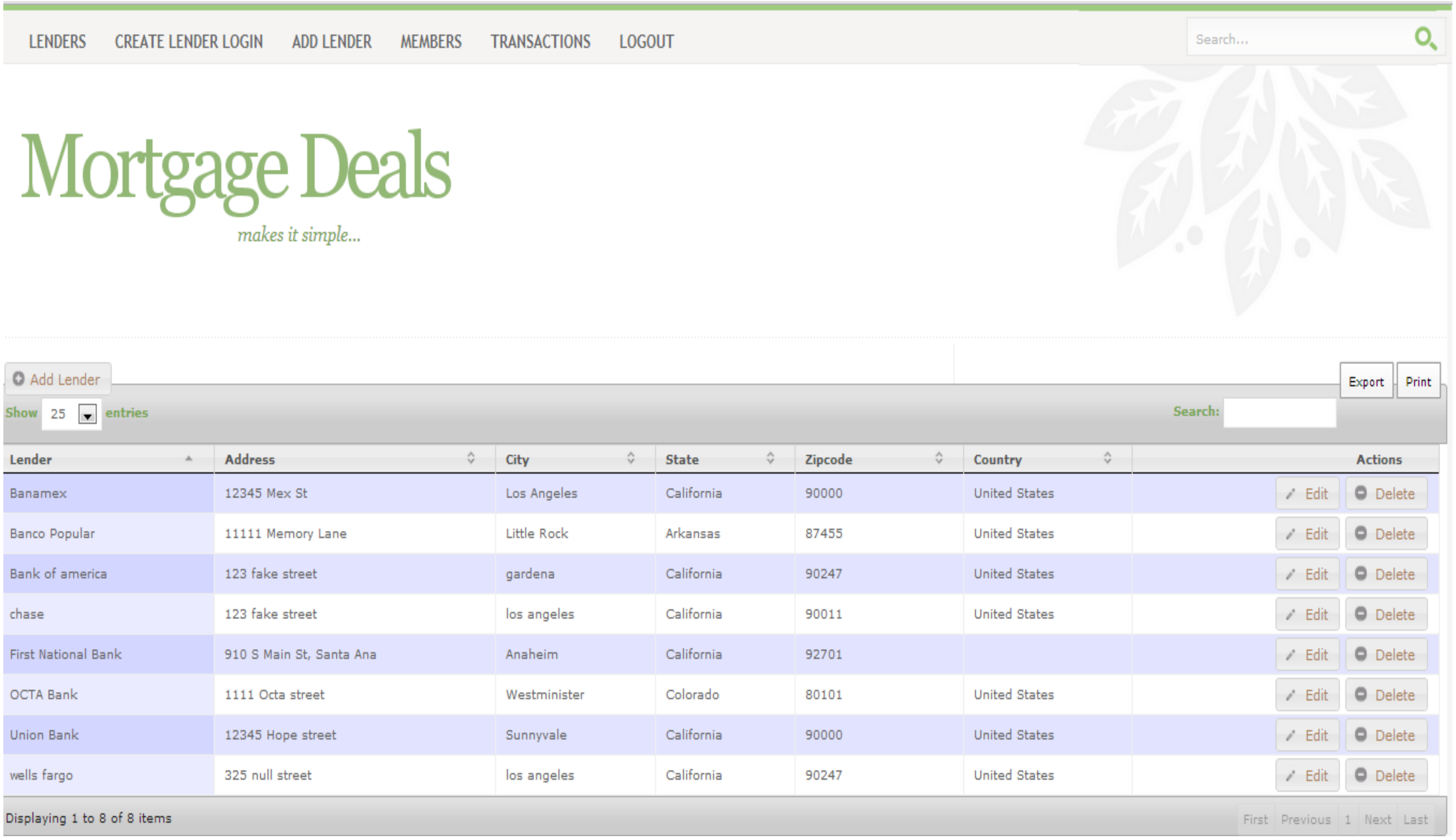


| Borrower Email | Price | Date Purchase | Purchase By | Lead Status | Actions |
|---|---|---|---|---|---|
| jcon.gallo@gmail.com | 10 | 13/05/2013 - 02:48 | unionbank | offered | View Make Offer Update Offer |

Displaying 1 to 1 of 1 items

©2013 all rights reserved

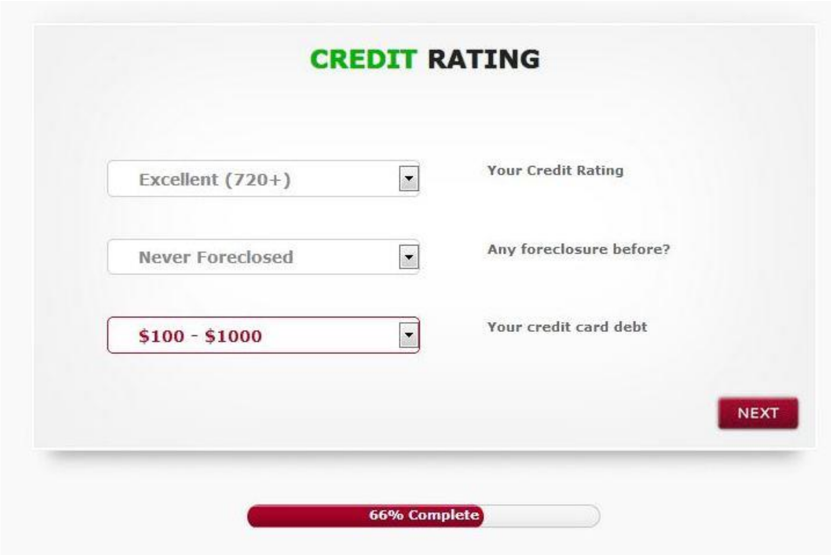
## 3. Administrator



| Lender | Address | City | State | Zipcode | Country | Actions |
|---|---|---|---|---|---|---|
| Banamex | 12345 Mex St | Los Angeles | California | 90000 | United States | Edit Delete |
| Banco Popular | 11111 Memory Lane | Little Rock | Arkansas | 87455 | United States | Edit Delete |
| Bank of america | 123 fake street | gardena | California | 90247 | United States | Edit Delete |
| chase | 123 fake street | los angeles | California | 90011 | United States | Edit Delete |
| First National Bank | 910 S Main St, Santa Ana | Anaheim | California | 92701 |  | Edit Delete |
| OCTA Bank | 1111 Octa street | Westminister | Colorado | 80101 | United States | Edit Delete |
| Union Bank | 12345 Hope street | Sunnyvale | California | 90000 | United States | Edit Delete |
| wells fargo | 325 null street | los angeles | California | 90247 | United States | Edit Delete |

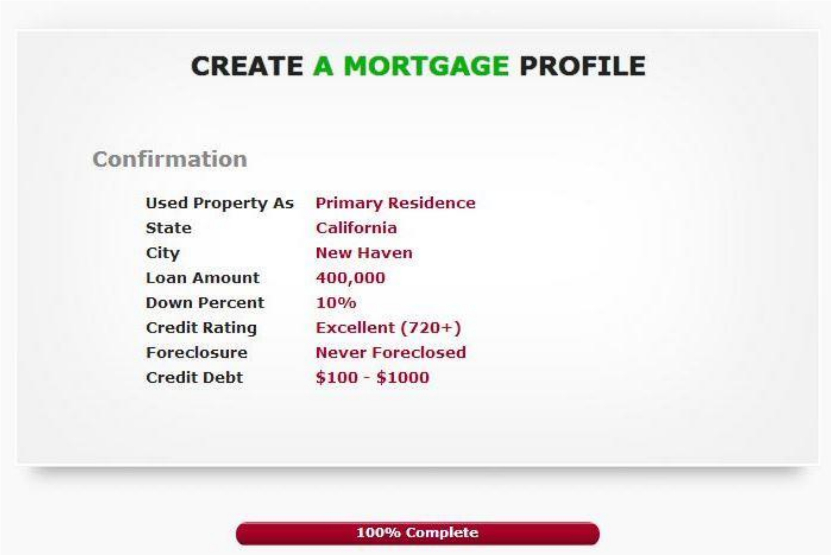
Displaying 1 to 8 of 8 items

## 2.6.2 User Screens/Dialog : User1



CREDIT RATING

| | |
|---|---|
| Excellent (720+) | Your Credit Rating |
| Never Foreclosed | Any foreclosure before? |
| $100 - $1000 | Your credit card debt |



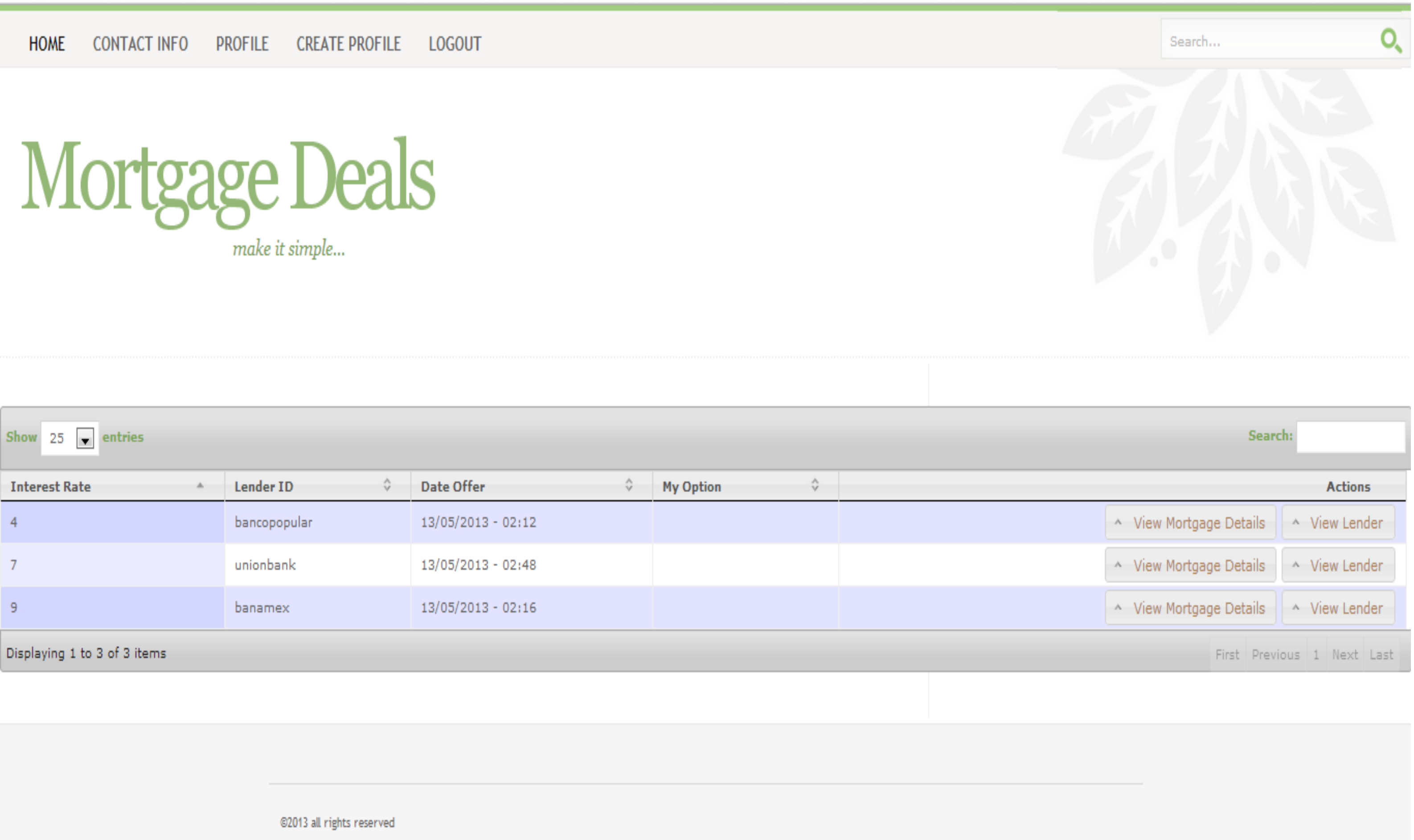
CREATE A MORTGAGE PROFILE

Confirmation

| | |
|---|---|
| Used Property As | Primary Residence |
| State | California |
| City | New Haven |
| Loan Amount | 400,000 |
| Down Percent | 10% |
| Credit Rating | Excellent (720+) |
| Foreclosure | Never Foreclosed |
| Credit Debt | $100 - $1000 |

Above are shown a couple of pages of the user’s Mortgage Profile creation process. When a user creates a mortgage profile they will go through a series of dropdown boxes that serve as answers to relevant mortgage information. These pages consist of text labels, next buttons,  dropdown boxes and a progress bar at the bottom of the pages.

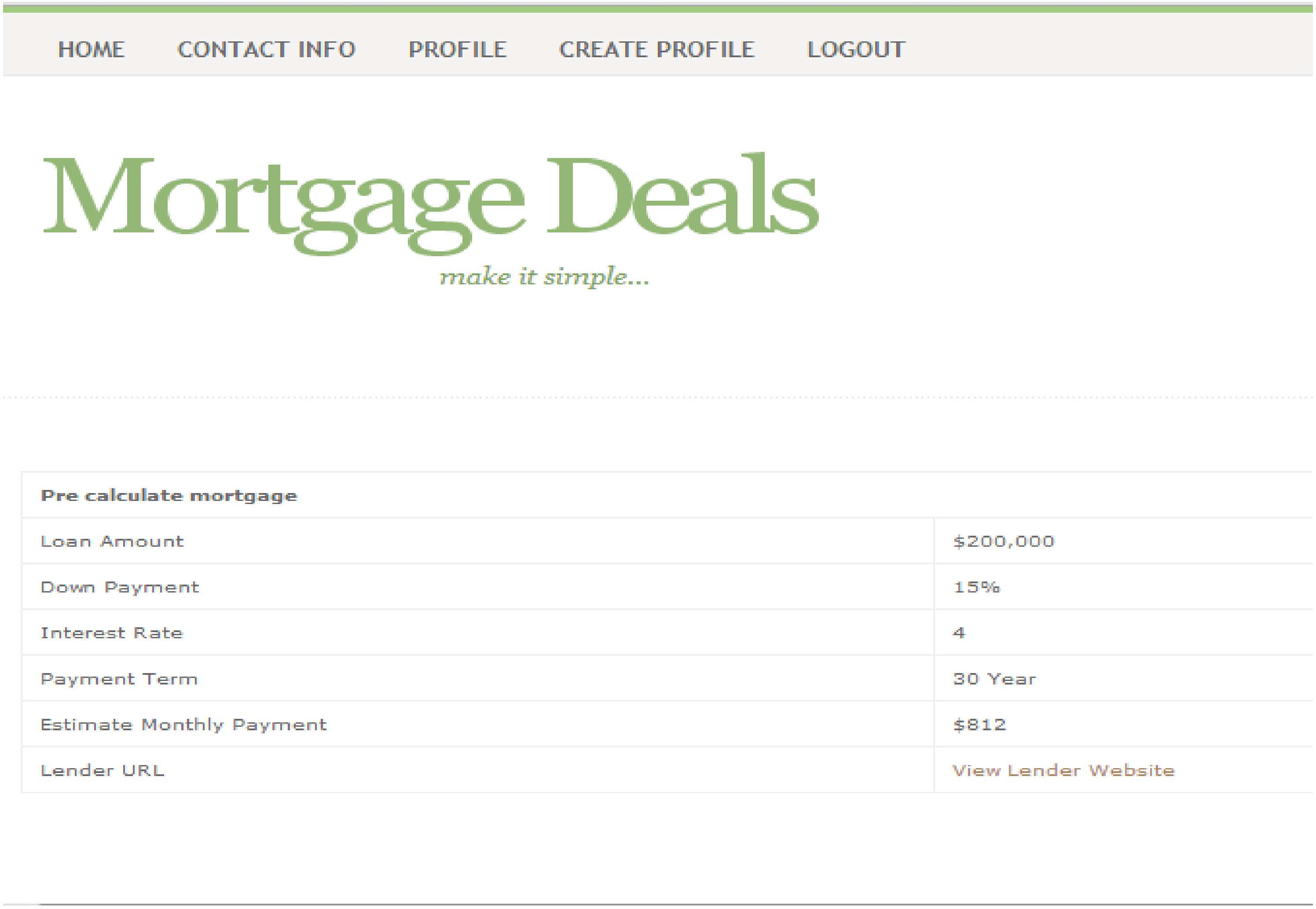
### 02.6.2 User Screens/Dialog : User2



| Interest Rate | Lender ID | Date Offer | My Option | Actions |
|---|---|---|---|---|
| 4 | bancopopular | 13/05/2013 - 02:12 | | View Mortgage Details<br>View Lender |
| 7 | unionbank | 13/05/2013 - 02:48 | | View Mortgage Details<br>View Lender |
| 9 | banamex | 13/05/2013 - 02:16 | | View Mortgage Details<br>View Lender |

Displaying 1 to 3 of 3 items

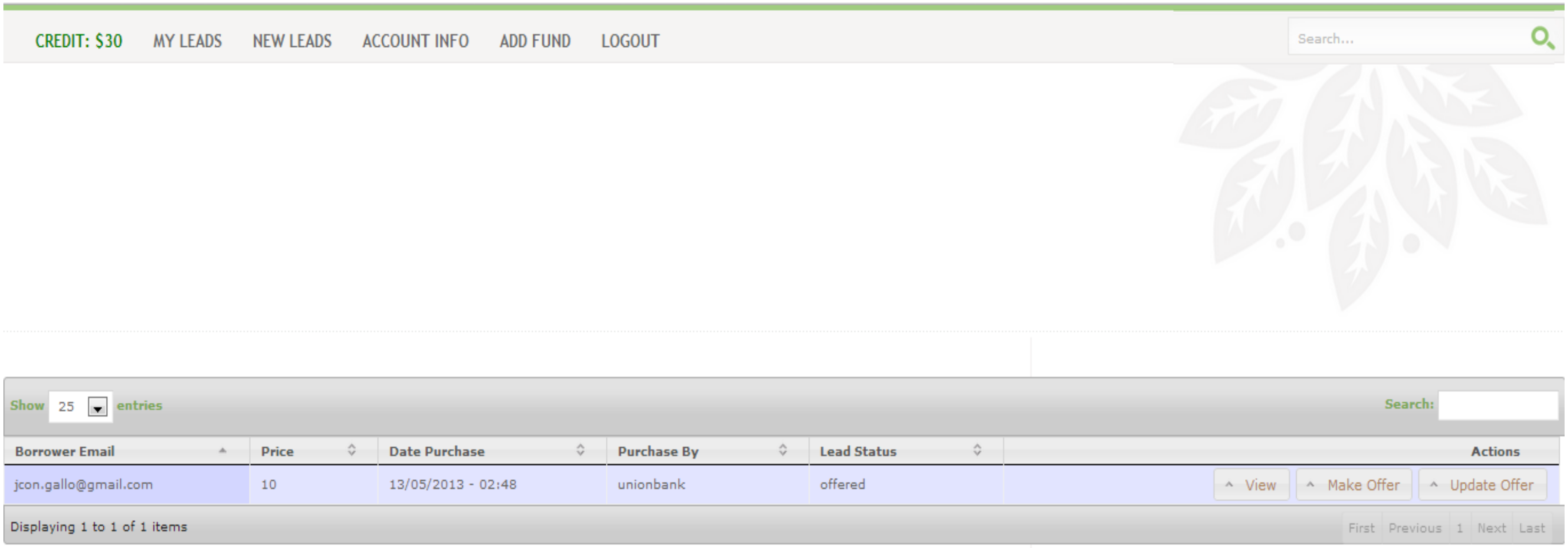
**Above are offers that a lender has extended to a user, the user can see details about the offer, such as the time and date that the offer was made as well as the offer details.**



| Pre calculate mortgage | |
|---|---|
| Loan Amount | $200,000 |
| Down Payment | 15% |
| Interest Rate | 4 |
| Payment Term | 30 Year |
| Estimate Monthly Payment | $812 |
| Lender URL | View Lender Website |

**Above the user has clicked on an offer that a lender has extended to him/her. The user is able to see the terms of the offer as well as a url to the lender's website.**

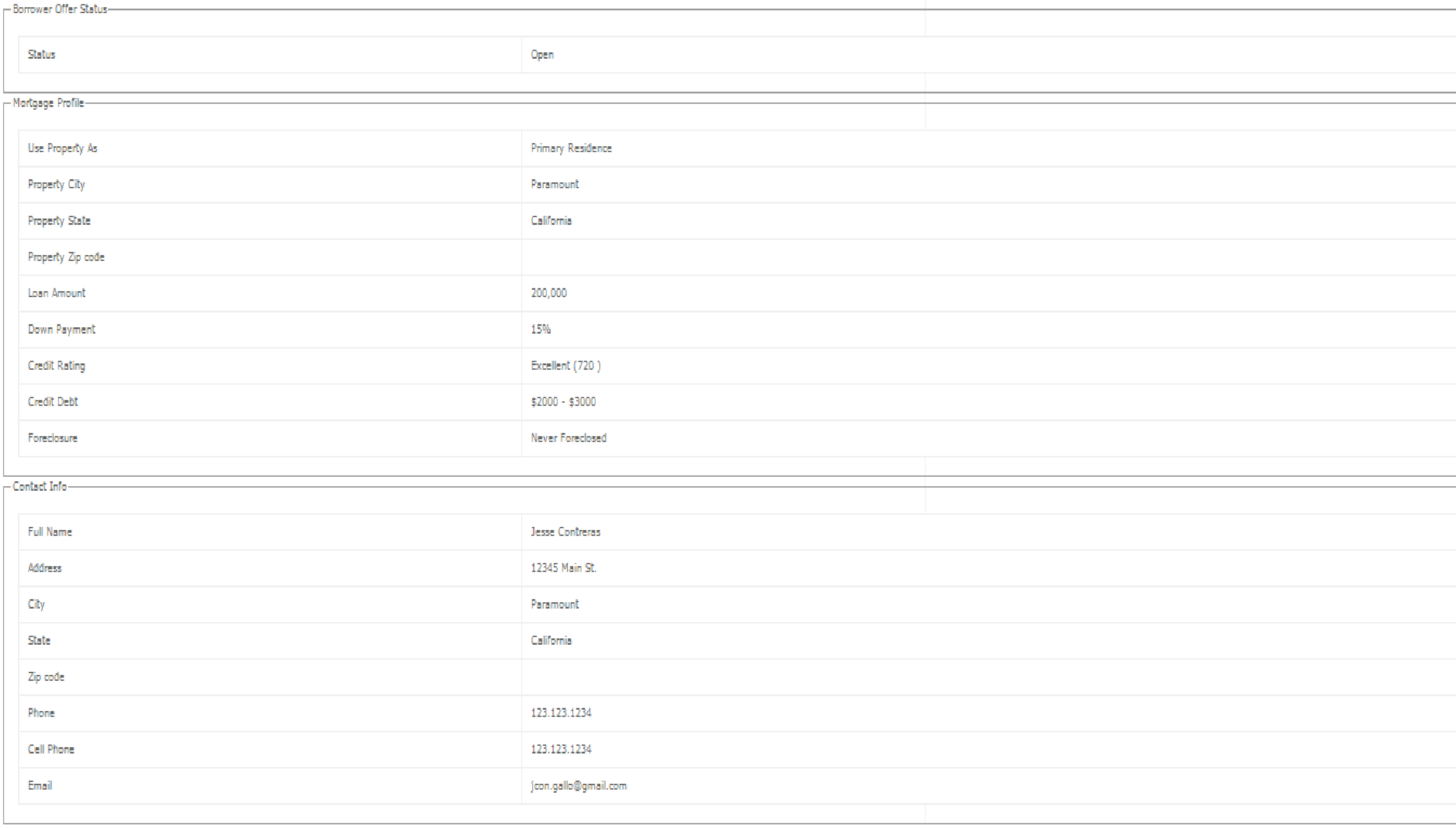
### 2.6.2 User Screens/Dialog : Lender1



| Borrower Email | Price | Date Purchase | Purchase By | Lead Status | Actions |
|---|---|---|---|---|---|
| jcon.gallo@gmail.com | 10 | 13/05/2013 - 02:48 | unionbank | offered | View Make Offer Update Offer |

Displaying 1 to 1 of 1 items

**Above is a lender page, what is being displayed is the lenders "leads", at this point the lender only has one lead. The lender will be able to view his/her offer, make one, or update an existing mortgage offer.**



Borrower Offer Status

| | |
|---|---|
| Status | Open |

Mortgage Profile

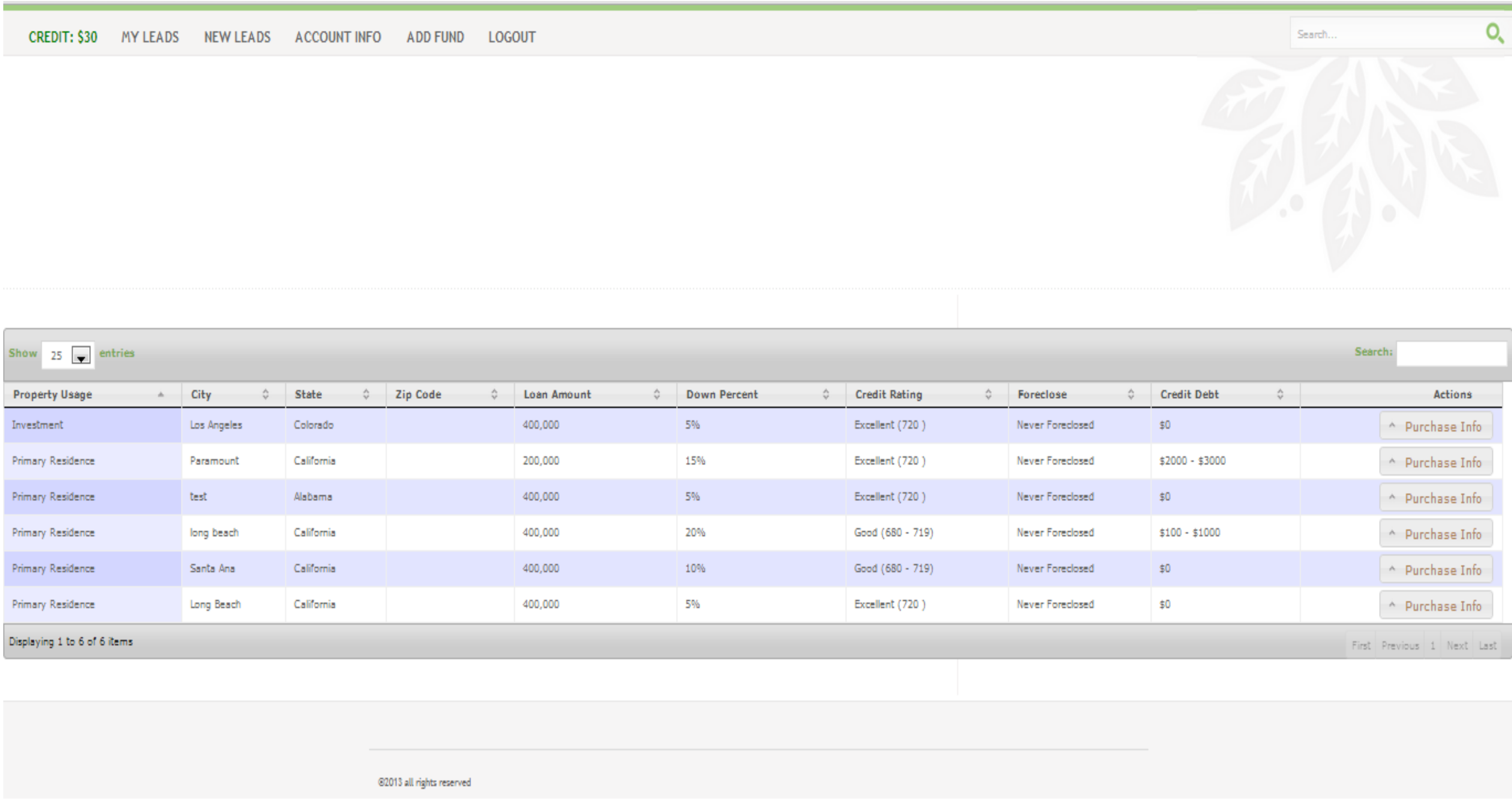
| | |
|---|---|
| Use Property As | Primary Residence |
| Property City | Paramount |
| Property State | California |
| Property Zip code | |
| Loan Amount | 200,000 |
| Down Payment | 15% |
| Credit Rating | Excellent (720 ) |
| Credit Debt | $2000 - $3000 |
| Foreclosure | Never Foreclosed |

Contact Info

| | |
|---|---|
| Full Name | Jesse Contreras |
| Address | 12345 Main St. |
| City | Paramount |
| State | California |
| Zip code | |
| Phone | 123.123.1234 |
| Cell Phone | 123.123.1234 |
| Email | jcon.gallo@gmail.com |

**Above the lender has chosen to view the offer and also visible will be the potential client's personal information.**

### 2.6.2 User Screens/Dialog : Lender2



Show 25 entries

Search:

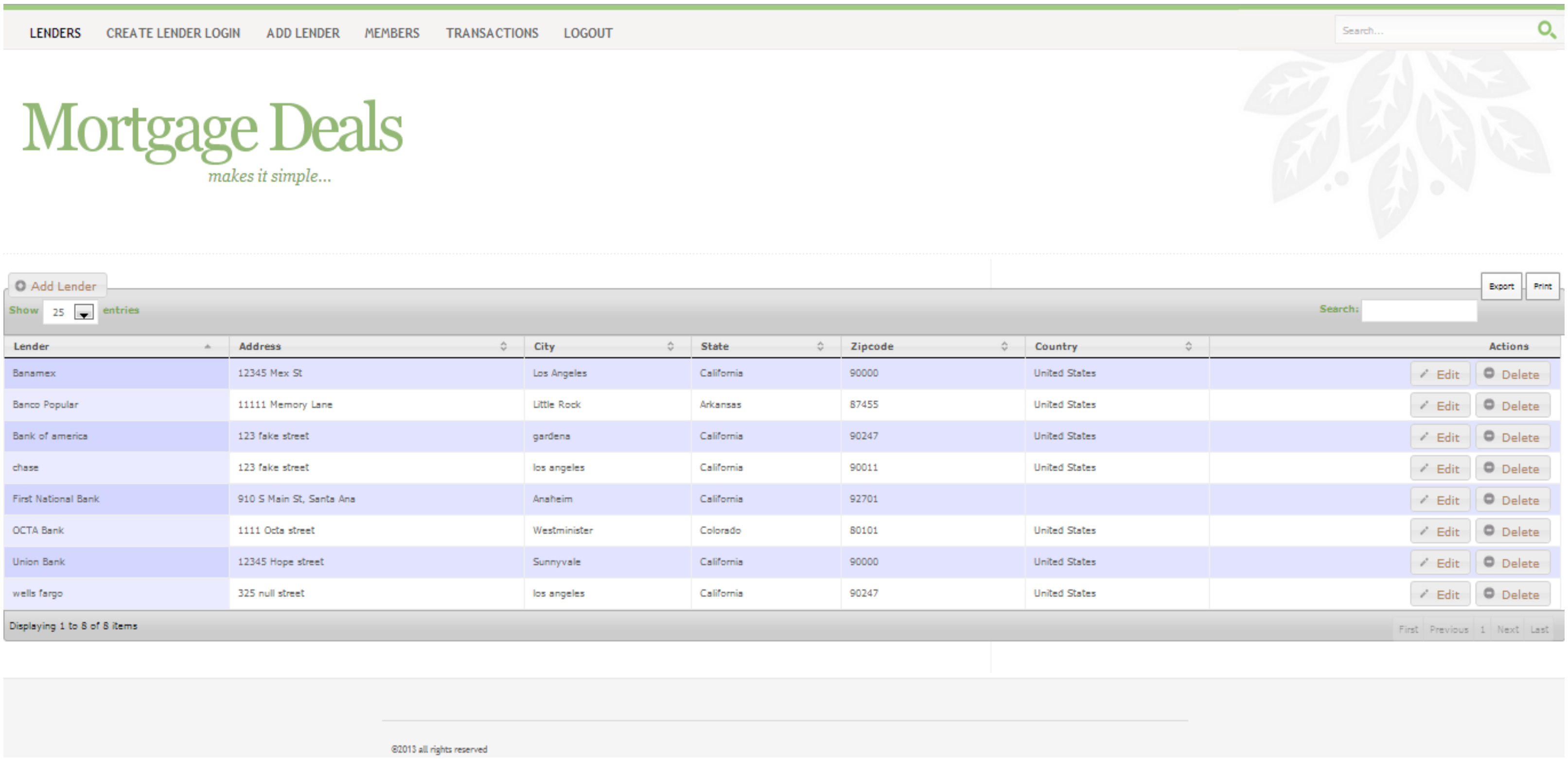
| Property Usage | City | State | Zip Code | Loan Amount | Down Percent | Credit Rating | Foreclose | Credit Debt | Actions |
|---|---|---|---|---|---|---|---|---|---|
| Investment | Los Angeles | Colorado | | 400,000 | 5% | Excellent (720 ) | Never Foreclosed | $0 | Purchase Info |
| Primary Residence | Paramount | California | | 200,000 | 15% | Excellent (720 ) | Never Foreclosed | $2000 - $3000 | Purchase Info |
| Primary Residence | test | Alabama | | 400,000 | 5% | Excellent (720 ) | Never Foreclosed | $0 | Purchase Info |
| Primary Residence | long beach | California | | 400,000 | 20% | Good (680 - 719) | Never Foreclosed | $100 - $1000 | Purchase Info |
| Primary Residence | Santa Ana | California | | 400,000 | 10% | Good (680 - 719) | Never Foreclosed | $0 | Purchase Info |
| Primary Residence | Long Beach | California | | 400,000 | 5% | Excellent (720 ) | Never Foreclosed | $0 | Purchase Info |

Displaying 1 to 6 of 6 items

©2013 all rights reserved

**Above is a lenders new leads, the lender could choose to purchase any of these lead's personal information. From the start the lender can view the potential borrower's mortgage profile information.**

### 2.6.2 User Screens/Dialog : Administrator



| Lender | Address | City | State | Zipcode | Country | Actions |
|---|---|---|---|---|---|---|
| Banamex | 12345 Mex St | Los Angeles | California | 90000 | United States | Edit Delete |
| Banco Popular | 11111 Memory Lane | Little Rock | Arkansas | 87455 | United States | Edit Delete |
| Bank of america | 123 fake street | gardena | California | 90247 | United States | Edit Delete |
| chase | 123 fake street | los angeles | California | 90011 | United States | Edit Delete |
| First National Bank | 910 S Main St, Santa Ana | Anaheim | California | 92701 |  | Edit Delete |
| OCTA Bank | 1111 Octa street | Westminister | Colorado | 80101 | United States | Edit Delete |
| Union Bank | 12345 Hope street | Sunnyvale | California | 90000 | United States | Edit Delete |
| wells fargo | 325 null street | los angeles | California | 90247 | United States | Edit Delete |

Displaying 1 to 8 of 8 items

**The administrator is able to view/edit/delete all lenders, members and transactions. It is up to the administrator to create an account for all of the lenders. An administrator cannot create a member.**

### 2.6.5 Error Conditions and System Messages

| Error Condition/System Message | Description/Corrective Action |
|---|---|
| **Incorrect input**<br>**System Message: "Please enter a valid value into the textbox."** | The user has input a non alpha-numeric value in a textbox that only takes in alpha-numeric characters.<br>Corrective Action: User must input a valid character into the textbox. |
| **Blank/Incomplete input**<br>**System Message: "The * field is required"** | A user leaves a necessary input container blank<br>Corrective Action: User must input necessary data into input container. |
| **Invalid Email**<br>**System Message: "The email field must contain a valid email address"** | Upon login or account creation user does not enter a valid email address i.e. ("dantheman$my.hey").<br>Corrective Action: User must input valid email address into input . |
| **Taken Email**<br>**System Message: "Email * has already been registered"** | Upon account creation a user inputs a registered email as their new account email address.<br>Corrective Action: User must input valid email (non registered) address into input . |
| **Insufficient Funds**<br>**System Message: "You do not have enough funds, please add more funds to your account"** | A lender will try to purchase a user's contact info with insufficient funds credit to his/her account<br>Corrective Action: There will be an automated or manual deposit of funds depending on the lenders preference. |
| **Character length check**<br>**System Message: "The * field must be at least * characters long"** | When a lender attempts to deposit money into his account without provided 16 characters for the credit card number or 3 numbers for the secure code.<br>Corrective Action: User must input valid number of characters into form. . |
| **Invalid offer rate**<br>**System Message: "The rate offer field must contain a value lower than 11"** | When a lender inputs a value for an interest rate, the lender inputs more than a maximum value for the rate.<br>Corrective Action: User must input valid value into form . |
| **Non-matching password fields**<br>**System Message: "The password field does not match the confirmation password field"** | When creating an account or updating a password, the user will enter distinct passwords to password textbox and confirm password textbox.<br>Corrective Action: User must input valid matching passwords into both password and confirmation fields. |

Typically error messages will occur in the event of an error. A majority of the time an error message will occur due an incorrect input of the user. Messages are displayed on the webpage itself, so there will not be a separate floating message box or other notification system.

### 2.6.6 Control Functions

**Note: To avoid redundancy some control functions were generalized.**

| Control Functions | Description (brief) |
|---|---|
| **SignUpLink/CreateAccountTab** | Will direct the user to a "Create Account" Page where the user will be able to input the necessary information to create an account. |
| **LostPasswordLink** | Will direct user to page where the user can input email in a textbox, submit a "new password" request, this prompts the server to email user a new password. |
| **Submit Button(data to server/database)** | Submits data to server for processing. |
| **NavigationTabs** | Links that assist user navigating through web-page content. |
| **Show*EntriesDropDownLists** | Any user of the service will be able to display a certain number of entries in a data table according to the value selected on the current drop down list. |
| **ViewDetailsButtons** | Will display a form with corresponding data. |
| **EditButtons** | Will display a form and allow a user to edit existing data. |
| **UpdateChangesButtons** | Will save the edited changes a user has made. |
| **UpdateGoBackButtons** | Will save the edited changes a user has made and return user to last page before edit. |
| **CancelButtons** | Will cancel an edit. |
| **PurchaseInfoButton** | Will display a form giving lender an option to purchase a user's personal information. |
| **SearchEntriesSearchBar** | Searchbar atop of data tables will allow for table search. |
| **MakeOfferButton** | Will process a request for a lender to make a "borrower" an offer. |
| **UpdateOfferButton** | Will allow updating and processing a request for a lender to make a "borrower" an offer. |
| **LogOut** | Will log out current user from their service page and end their session. |
| **ChangePasword** | Will display a form that will allow a user to change password. |
| **AddLenderButton** | An Administrator will be able to add a lender via a form submission. |
| **ExportButton** | Will export all of the data in a table to a spreadsheet onto localhost. |
| **DeleteButton** | Will delete a record from the MortgageDeals database. |

# 3. System Performance Requirements

## 3.1 Efficiency & Reliability

| Functional Requirements | Administrator | Lender | Borrower |
|---|---|---|---|
| **SignUpLink/CreateAccountTab** | | X | X |
| **LostPasswordLink** | | | X |
| **Submit Button(data to server/database)** | X | X | X |
| **NavigationTabs** | X | X | X |
| **Show*EntriesDropDownLists** | X | X | X |
| **ViewDetailsButtons** | X | X | X |
| **EditButtons** | X | X | X |
| **UpdateChangesButtons** | X | X | X |
| **UpdateGoBackButtons** | X | X | X |
| **CancelButtons** | X | X | X |
| **PurchaseInfoButton** | | X | |
| **SearchEntriesSearchBar** | X | X | X |
| **MakeOfferButton** | | X | |
| **UpdateOfferButton** | | X | |
| **LogOut** | X | X | X |
| **ChangePasword** | | X | X |
| **AddLenderButton** | X | | |
| **ExportButton** | X | X | X |
| **DeleteButton** | X | X | X |
| **AddLender** | X | X | |
| **ViewTransactions** | X | X | |

### 3.1.1 Description of Reliability Measures (accuracy, precision, consistency, reproducibility, etc.)

Considering this is primarily a desktop web service, MortgageDeals accomplishes the following:

1. Navigating menus takes less than 10 seconds
2. Extra leeway acceptable for exporting spreadsheet documents
3. Pages fit on many resolution desktop screens
4. For email (for lost password):
    a. Display should be intuitive i.e. it should be obvious where subject, recipient and message body go.
5. For tables search-bar search results return accurate results and are near-instantaneously cached.
6. For edit content user should have complete control of modifiable content.
7. Scalabity is scaled down for optimized use of MortgageDeals
8. Easy to use.
9. Clean, direct and responsive UI.
10. Data displayed on content sections of webpages are useful to all stakeholders.
11. Information is created upon submission of data
    a. Editing data easy 2 step process
        i. Modify data
        ii. Submit modified data
12. All web controls do exactly as they are labeled/abstracted.

Note: The consistency of the web pages is high, nevertheless there are a few web pages that deviate from the standard design.

### 3.1.2 Error/Failure Detection and Recovery (failure modes, failure consequences, error logging and reporting, manual and automatic recovery procedures)

- The framework used, CodeIgniter, contains a “errors” and a “form_validation” folder whose functions help deliver error messages to the user using appropriate logic.
- After error is logged the server prompts CodeIgniter to display a message based on pre-defines rules in a functions parameter.
- Errors include but are not limited to
    - Incorrect user input
    - Invalid user input

**3.2 Security**

**3.2.1 Data Security**

- A user’s private data is protected by a unique session, initiated by the user once successfully logged on
    - A user’s session is terminated once they log out of MortgageDeals
    - Data is not visible to public or unauthorized users.

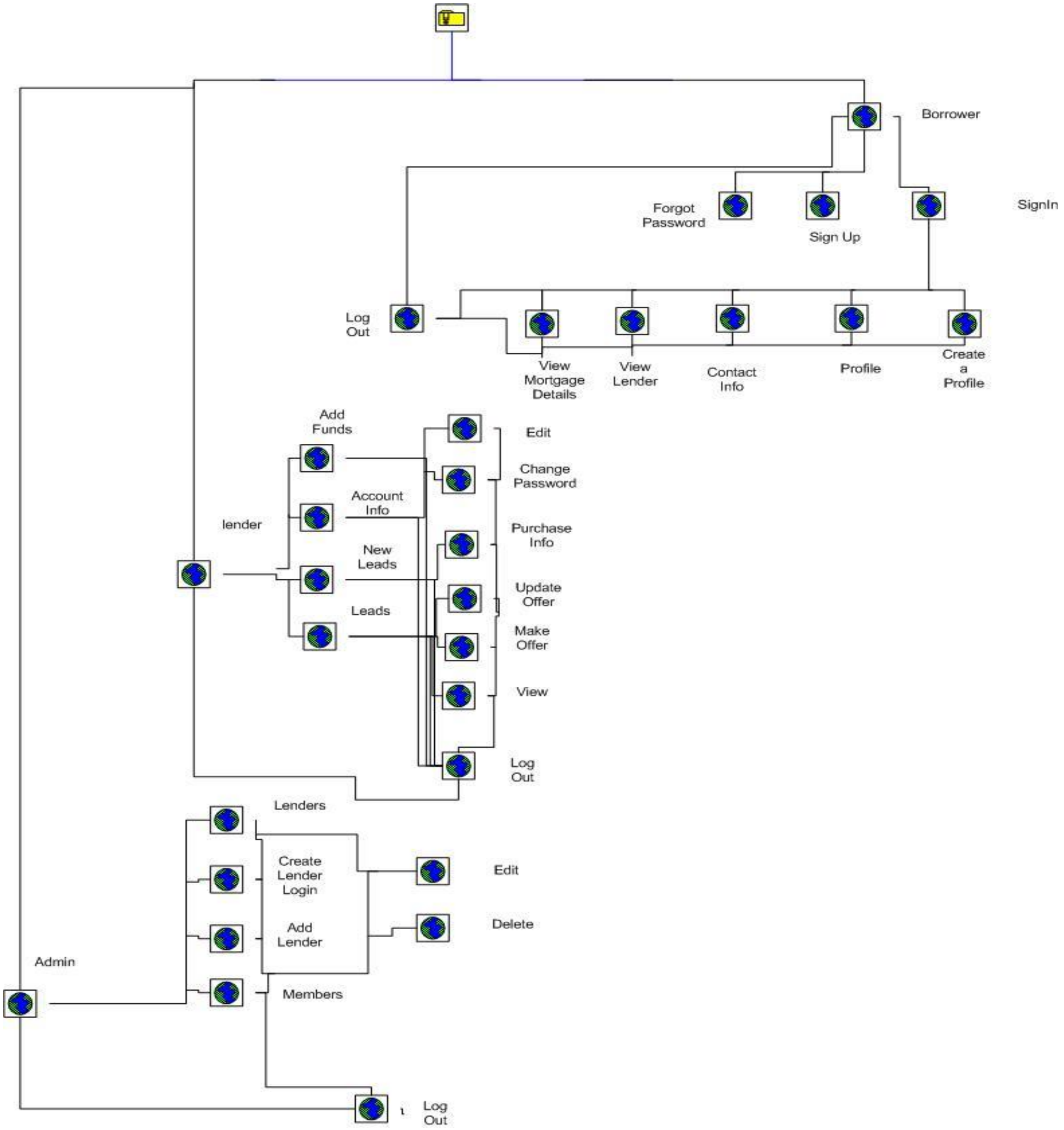
**3.2.2 Execution Security (user validation)**

- User validation occurs in login, MortgageDeals checks database for matching information.
- A user cannot create 2 or more account with the same email.

**3.4 Maintainability & Modifiability**

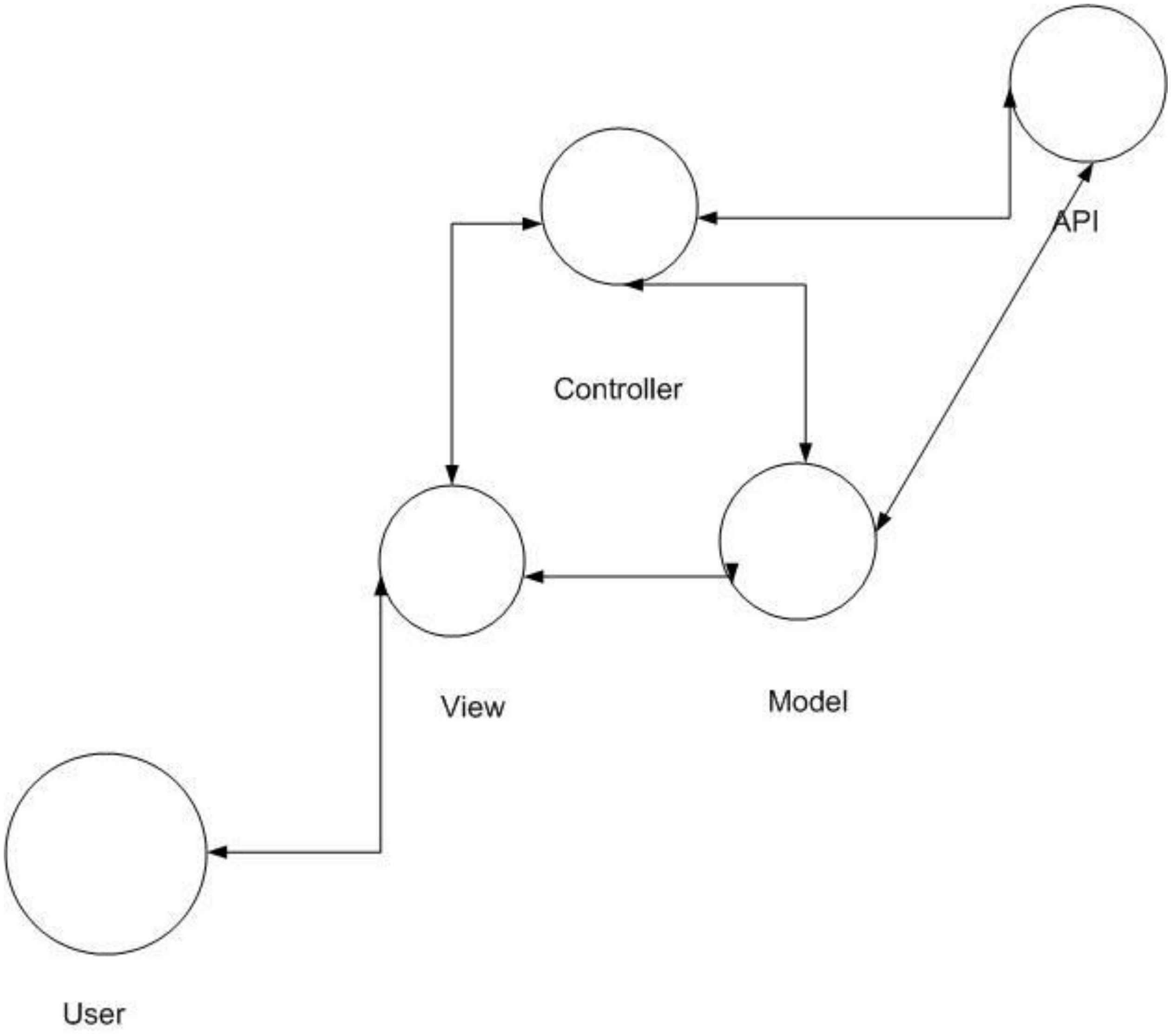
- The MVC model for MortgageDeals allows for just the right amount of modularity for a web-service of i’s size.
    - The logic and design could be maintained and modified independently
    - Addition of any Model, View or Controller is simple and straightforward.

# 4. System Design Overview



This is the site map of the web application. We have three sub site maps that break down the three different users (admin, lender, and borrower). The site map shows the different pages each user can go to based off of the pages they are located in. This map can be a guide to follow for users that would like to see the steps to get to a certain page on the application to understand some of the flow of the application. For example, if you would like to edit the lenders information for the admin we would have to go to the admin home page. After that we would have to go to the lenders page and select the edit button to go to the edit page.

## 4.1 Description of System Operation (high level)



The user interacts with the application by viewing the applications UI design.  The user enters data through the UI which is sent to the controller which is sent to the model to retrieve from the database.  Depending on which function the user is looking for the API will then be called by the model and the controller to retrieve data from the Zillow API (API used mainly for mortgage calculation). Zillow's API sends any data back to the model which then goes to view and essentially on the computer monitor for user to see.

**4.2 Equipment Configuration (diagram and description)**

- Netbeans – This was used to implement codeIgniter and PHP
- Heidi – Heidi was used to create the tables and database files in MySQL
- Github – was used as the repository
- Photoshop - was used to edit pictures and create pictures for the web application
- Zillow API – was used to retrieve mortgage rates and monthly payment rates for the borrowers.
- Godaddy host server – was used to host web application
- Internet connection – establishes connection with the web application server and database.
- Google Docs – was used to organize all of documentation for the application.

**4.3 Implementation Languages (which and why)**

- PHP – was implemented for the web application dynamic functionalities.
- HTML – was used to code the basic web application display.
- CSS- CSS was used to adorn the UI of the web application.
- CURL – was used to establish connection with the API.
- MySQL – was used to create the database.
- XML- the Zillow API used XML.
- Javascript / JQuery – was used to give dynamic functionality on the client side of the web application.

**4.4 Required Support Software (pre-existing)**

- Update browser support(Firefox and Chrome' s)
- Internet Connection
- Zillow API server needs to be running
- A basic computer that support browsers

# 5. System Data Structure Specifications

## 5.1 Other User Input Specification

The input specification is the Zillow API. The Zillow API fetches borrower's data form database, as well as the interest rate offered by the lender. It computes the information in the API's server. Once the API compute's the borrower (user) and lender it sends back output data and makes it visible to borrower.

## 5.2 Other User Output Specification

The output given by the Zillow API is in XML format. The XML file is all sent back to the user to view, with all of the data it computes. However, the user is most likely more concerned with seeing a monthly mortgage rate/payment. The XML file is parsed through to output the monthly mortgage payment that is then stored back into the user's database instead of the API's database. This is stored so the borrower can see how much they would be paying monthly based out of the offer given to them by the lender.

## 5.3 System Data Base/File Structure Specification

The data base structure in SQL is made to create the tables which our application needs to save the data input by the user. These tables are created and stored in the server for the application to access when a user is communicating with our web application. Our SQL code contains 19 tables and 8 alter tables.

- The country table contains the names of the countries.
- The credit type saves the borrower's information if they have excellent or bad credit.
- Property type table is used for the user to choose what kind of property they have in the form they are filling out.
- The lending type stores the name of the type of loan they are going to give you.
- The property info is to describe the area by zip code is located in and the value of the home
- The state table is created to save all the names of the states for the United States.
- The transaction table is the table that stores the offer the lender gave to the borrower with information of whether the borrower accepted or denied the offer the lender gave them.
- Mortgage Profile table is to store the information of the borrower to obtain a mortgage by filling in that information as a form.
- Price calculation table is the table used by the API to output the calculation for the borrower's monthly payment.
- The purchase info table is used to show the lender the information of the borrower to see if the lender would like to purchase and view the borrower as a lead that the lender may want to work with to offer the borrower a rate.
- The employee info table stores the information of the lender's that are going to be in the application to offer rates to potential leads they decide to purchase.
- The payment info table stores the information for the credit card the lender is going to use to add funds to purchase leads they would like to view as potential customers.
- The user log table is to keep track of the user while he is logged in to the application. This will allow for the user to stay logged in throughout the session.
- Lender info table stores all the information of the lender that a borrower can view such as address, phone, fax, and website of the lender.
- The lender credit table is used to save the amount the lender has in their funds to purchase new leads they would like to work with as potential customers.
- The financial info is a table that is used on the borrower's side that would describe and store all the financial information of the borrower the application ask for them to fill out.
- The member info table is the table in which stores all the contact information of the borrower.
- Login info table stores all the important information that a user needs to log into their account such as email and password.

### 5.3.1 Identification of Data Base/Files

The database is created is to maintain the data entered by three different types of users. The three types of users are administrator, lender, and borrower. The administrator can create and delete users of the application.

The lender is the person that can offer an interest rate, the borrower can accept his offer to be his customer. The database will store the entire lender's information and data that he enters for borrower's to view. The lender can use the table to add funds with their credit card which will then be stored in the database. This will allow for the application to use this data to know if the lender can interact with the new leads. Also, the application will use the funds table the lender used to know if the lender has enough funds to purchase leads (borrower's information) to offer an interest rate.

The borrower is another user in which they interact with the application by filling out a simple form that would be viewed by lenders in the site. After filling out the form the borrower just needs to wait for offers by lender's that are interested in working with them for a mortgage. Once a lender sends an offer the borrower would either accept or deny the offer. This will trigger the data table that stores that decision for that offer so the lender can view later on if the lead accepted his offer.

### 5.3.2 (Sub) systems accessing the Data Base (creating, updating, using; frequency) describe the methods used to update, delete, and submit the data

The sub systems in our application are the models that interact with the controllers. They load specific methods for deleting, updating, reading specific data tables on the server. The administrator can only create a login for lender's and delete the lender profile. The administrator does not have the power to manipulate the information within the profile which is another functionality that only the lender can use.

The lender can update, delete, and add funds in their profile. The functionalities gives the lender the possibility to access the data table in which their information is being stored to edit or remove their information. Also, it gives the lender access to the funds table to add more funds so they can purchase new leads. The model links the lender functions with the data table so they can interact and store the information in the database.

The borrower can only update and delete their profile; also, they can deny offers from lenders. The borrower can only interact with the table that stores their information in the database. This table stores basic information the borrower is required to fill in for the application to interact properly with the API that would output the proper information they are looking for on a mortgage monthly payment with the offer given to them. The other table they interact with is the offer table in which they can accept or deny the offer the lender gave them. By interacting with this table they can then store their response into the database which the lender can view later on when they login.

### 5.3.3 Logical File Structure (record formats, file organization, access methods, rationale, examples)

The logical file structure for our application are the forms the users interact with while creating profiles. These forms work as a temporary structure that saves the information of the user during the session. This logical file structure gives the user the power to create an object (profile) for others to be able to view. Once the user decides to submit the form the objects information (user information) is then sent to the physical file structure.

The controller will give access to the forms so the user can input their information through the client side of the application. These methods will allow for the user to interact with the application while still not communicating with the physical file structure. Once the user finishes inputting the information and submits it then the methods change to access new methods that will save the information in the database.

### 5.3.4 Physical File Structure (storage device, blocking, organization, access, etc.)

The physical file structure is the database that is saved in the server of our application. The database is the one that organizes all the information that is sent from the logical file structure to the server. By using database tables the database will organize all the information making is simpler to access this information at a later time when the application needs to read it from the physical structure. This will make it easier on the API to obtain the information from the physical structure because is organized in a manner that makes it easy to call for the API to function properly when using the database information of a user.

### 5.3.5 Data Base Management Subsystems Used (internal or external)

The database is being managed externally by using MySQL. The model section of the MVC is managing the data going through it by interpreting the tables of MySQL. Once a model interprets the data and dictates where the data has to go in the database the model will insert the data in the proper place of each table in the database.

### 5.3.6 Data Base Creation and Update Procedure (if NOT by system)

The database was created by using the program Heidi. This program allowed me to create the tables for the web service. Once the MySQL coding for the database structure was complete I used the controller side of the MVC to update the information. The controller would allow the user for the information in the data tables to be updated with the new data entered by the user in the application to over write the old data. For example, when the borrower needs to edit his information from previous mortgage profile they created the controller accesses his old information and allow the user to edit/ change previous data. Once the user submits the changes the controller will over write the old data with the new data that would replace it in the table with the updated information.

## 5.4 Modules Accessing Structures (creating, updating, using)

The controllers that access the data structures are the lender's controller to input their information. The forms that the lenders needs to create for their profile accesses the data structure to store their information. Their information can be adding funds, profile information, or offering a rate to a lead. The addition of funds would update the funds table. The profile form will be update/create new information of the lender that was not previously inputted into the table. The controllers for the borrowers are to update and create the mortgage profile they created. They access the database table to create the mortgage profile as well as the contact information table for the borrower. The administrator creates and deletes profiles for the application. The controllers accesses the database in which it stores the login information for the login profiles the administrator created for users (lenders).

# 6. Module Design specifications (for each module)

## 6.1 Module Functional specification

In the web service I am using the Model View Controller with CodeIgniter framework. I have multiple controllers in the web service. There is a default controller for the web service that allows users to log in. There are also controllers for the main users (the barrowers), the lenders, and for the admin side.

### 6.1.1 Functions Performed

The default controller allows new members to register and old register to log in. Once are members are login depending on type user they are they can perform certain actions.

site controller function perform

- register
- login

member (barrower) controller functions perform

- edit change contact info
- change password
- create one mortgage profile
- edit the mortgage profile
- logout
- view offers from lender

Lender Controller functions perform

- view barrowers information
- purchase barrowers information
- edit profile info
- add fund so they can purchase more barrowers
- send offers only to barrowers leads that they purchase

Admin Controller function perform

- login
- create a lender account
- create a login for the lender
- view transaction
- edit/delete barrowers/lenders information
- login/logout

### 6.1.3 Module Limitations and Restrictions

All functions are restricted to work unless they are login. The only functions that have no restrictions are the login functions and the register functions. Once the users are login they can perform more functions.

site controller function perform

- register
- login

member (barrower) controller functions perform

- edit change contact info(require login)
- change password(require login)
- create one mortgage profile(require login)
- edit the mortgage profile(require login)
- logout(require login)
- view offers from lender(require login)

l

Lender Controller functions perform

- view barrowers information(require login)
- purchase barrowers information(require login)
- edit profile info(require login)
- add fund so they can purchase more barrowers(require login)
- send offers only to barrowers leads that they purchase(require login)

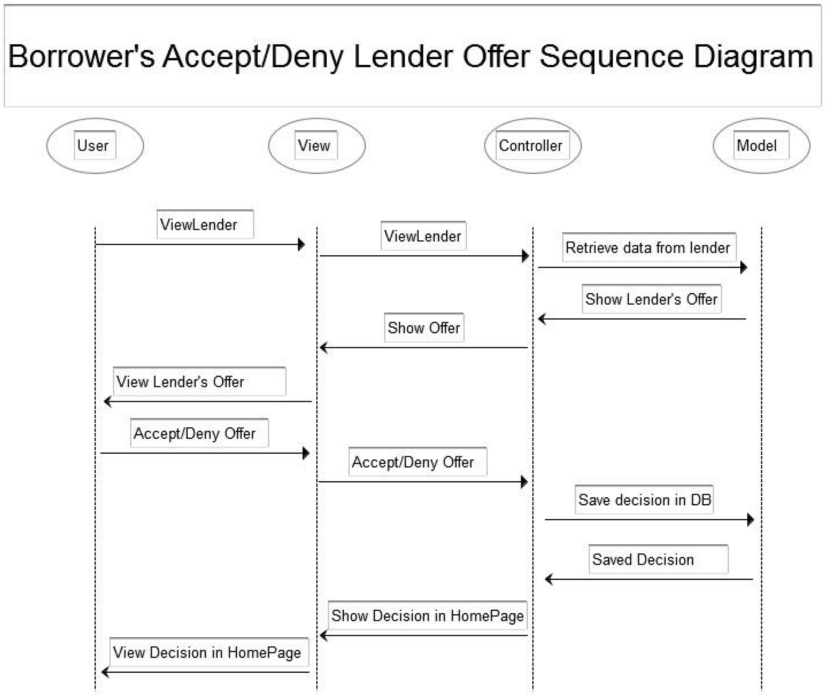
Admin Controller function perform

- create a lender account(require login)
- create a login for the lender(require login)
- view transaction(require login)
- edit/delete barrowers/lenders information(require login)
- login
- logout(require login)

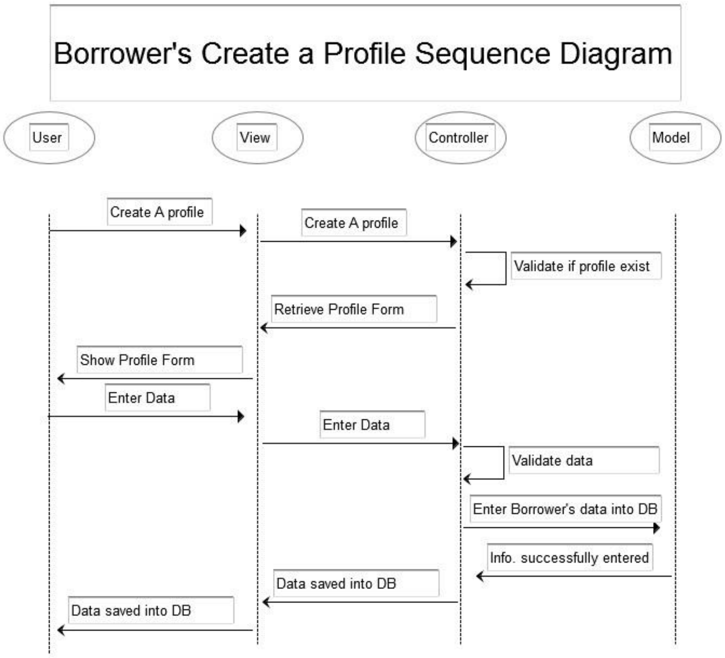
## 6.2 Module operational Specification

The following are UML diagrams describe the interactions between users and the web service

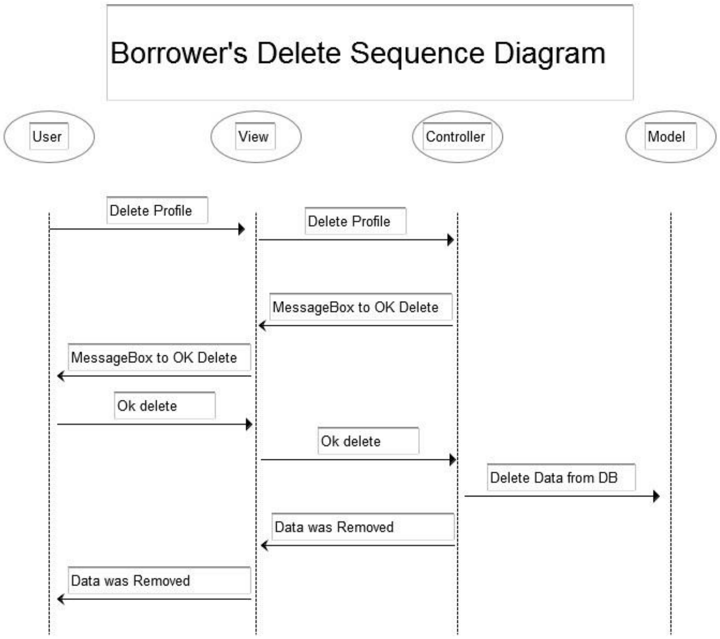
### Barrowers UML Diagrams



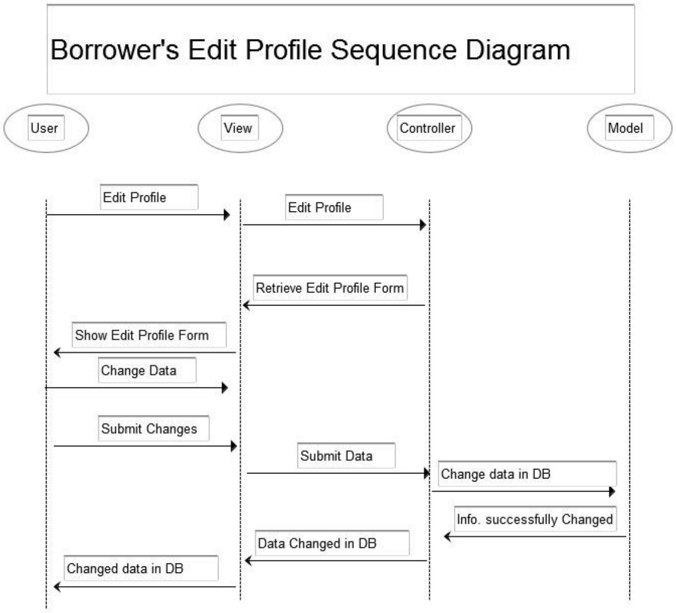
The borrower's accept/deny diagram describes the flow of what the borrower's decision for the offer goes through. The borrower should be signed in to their profile to be able to view the lender that is giving them an offer. By selecting the view Lender button the borrower can view the offer the lender gave them as well as basic information of the lender. Once the borrower views this page and is satisfied with the information they see then the borrower makes a decision about the offer that was given to them. The borrower then selects their choice of either accepting or denying the offer. This selection is sent through the application into the database to be saved for the lender to be able to view the choice the borrower made about the offer.



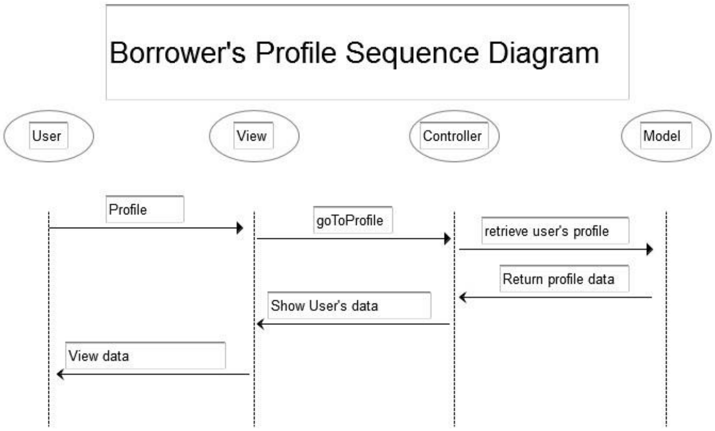
The sequence diagram to create a profile is to create a mortgage profile. The mortgage profile sequence is by selecting the tab to create a profile. The application will validate if a profile already exist. If it does not exist the controller will send the form to be viewed by the user. Once the user sees the form they can fill out the form with the appropriate information. This information is being validated through the process of the form. The user submits the information which is saved into the database and the user will be shown a page with the data they input into the database.



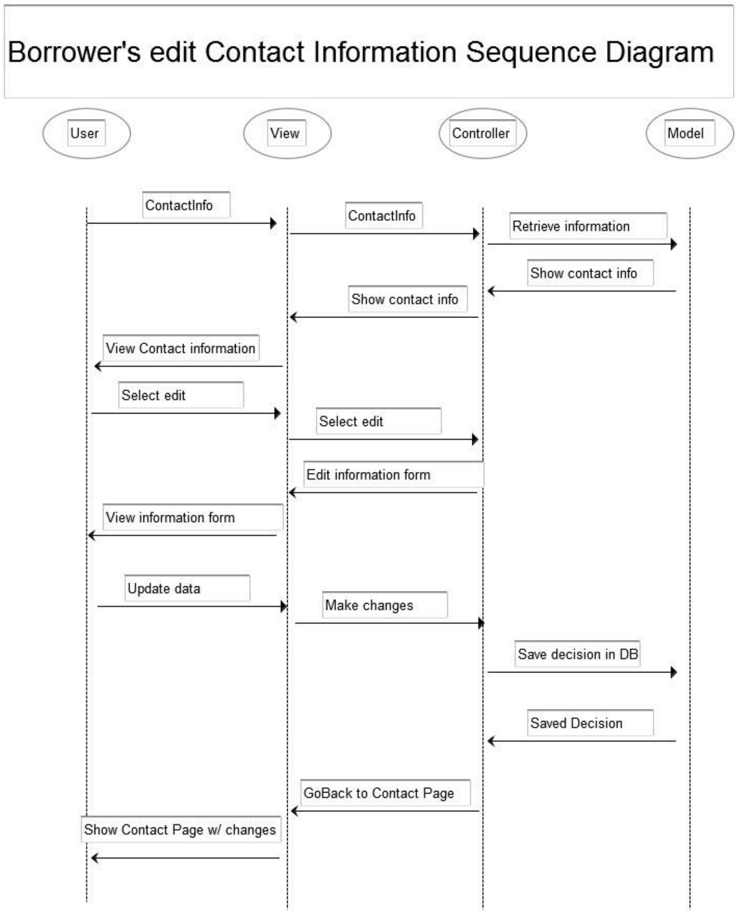
The delete diagram is the steps taken to remove a mortgage profile the user created. We'll assume the user is signed into their page. The user will see a delete profile button on the right they will select the button which will send a message to the controller. Once the controller receives that message it will send a message box to the user asking if they are sure they want to delete the profile. If the user okays the action then the mortgage profile will be deleted. This means all the information that was saved in the database will be removed from the database. The user will see their profile removed on the website as proof that they no longer have a mortgage profile was removed.



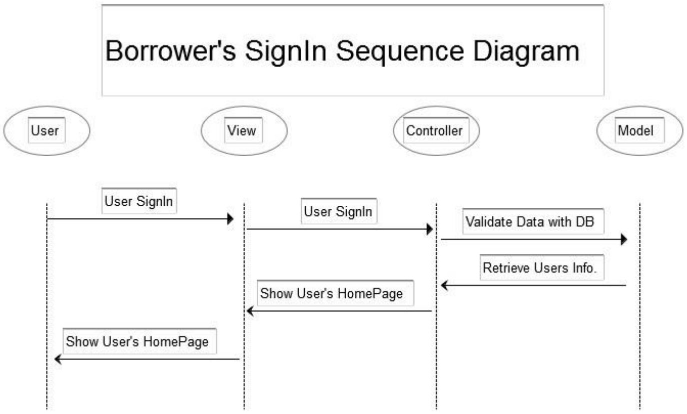
The edit  mortgage profile diagram gives the user the ability to change their information.  The diagram shows that they need to select the edit profile button from the profile page.  When they select the button the controller will send the user the form to edit the information from their previous mortgage profile.  The user then changes the information and submits the data that goes through the controller.  The controller sends the data which will change the information in the model for the database.  The user will obtain a message saying the data was changed successfully.



The user can go view their profile they created by choosing the profile tab.  The controller view will send the message to the controller when the user selects the profile they want to view.  Once the controller sends the model to retrieve the information of the users profile to be viewed.  The model sends the information through the controller to organize the information for the user to view properly.

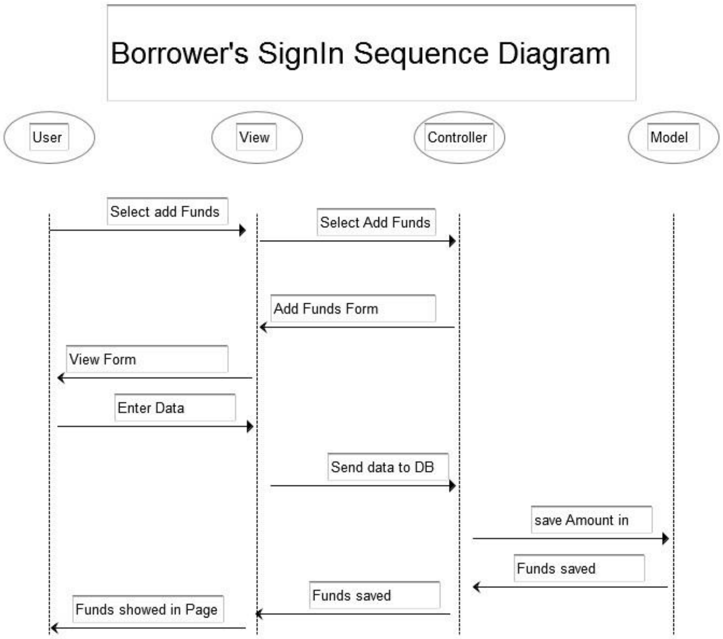

Borrower's edit Contact Information Sequence Diagram

The user selects the contact info tab which the controller shows the contact info page.  The controller will send a message to the model to retrieve the users contact information that goes through the controller to display it properly to the user to view.  The user then can choose to edit the information by selecting the edit button that will send the controller to show the form with the information filled in to edit or change it to the user’s specifications.  The user updates the information they want to change.  When they submit the information the controller will receive it to break it down to send it to the database to be saved.  The database sends message to the controller that the data was saved.  The controller then sends the new information that was saved to the view so the user can see their information was changed.

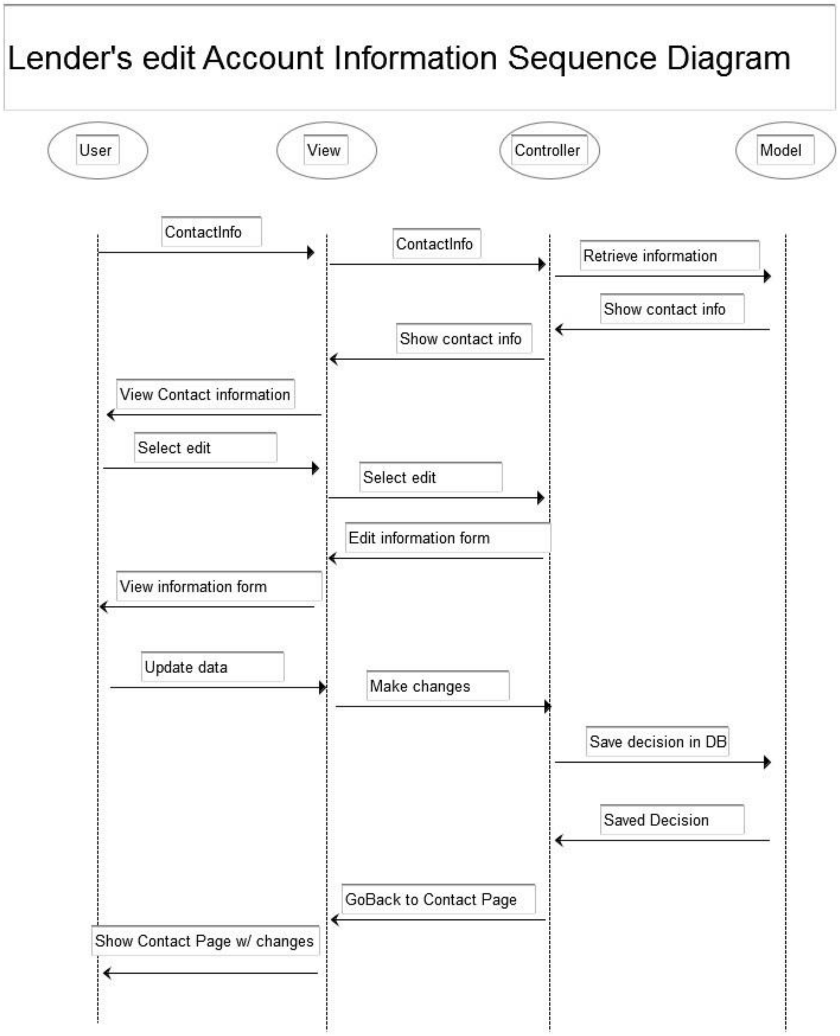


The borrower's sign in diagram is before the user is logged in.  They are in the main page of the website and they will see a form that will ask for their login information.  The user will fill in the form and choose sign in button.  The button will send the controller to retrieve the users page information to the model.  The model will return the data to the controller to show the user in the view their home page after the logged in.

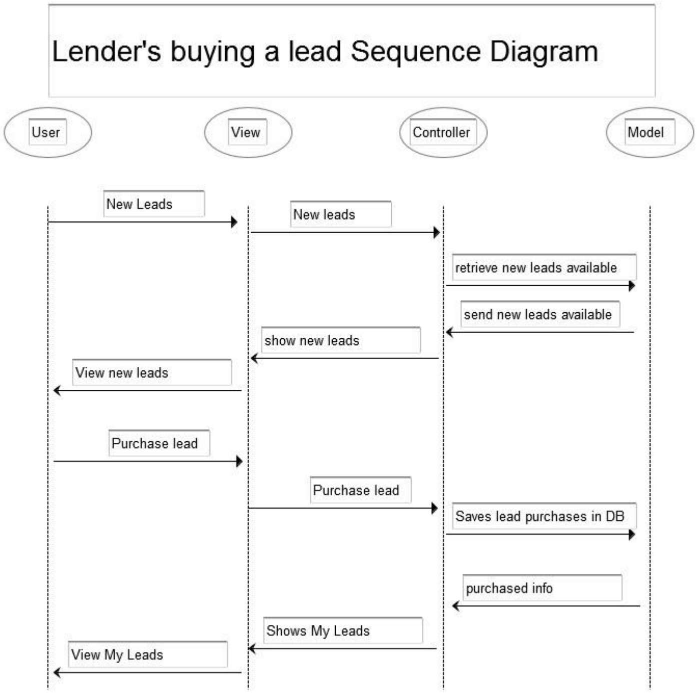
## Lender UML Diagrams



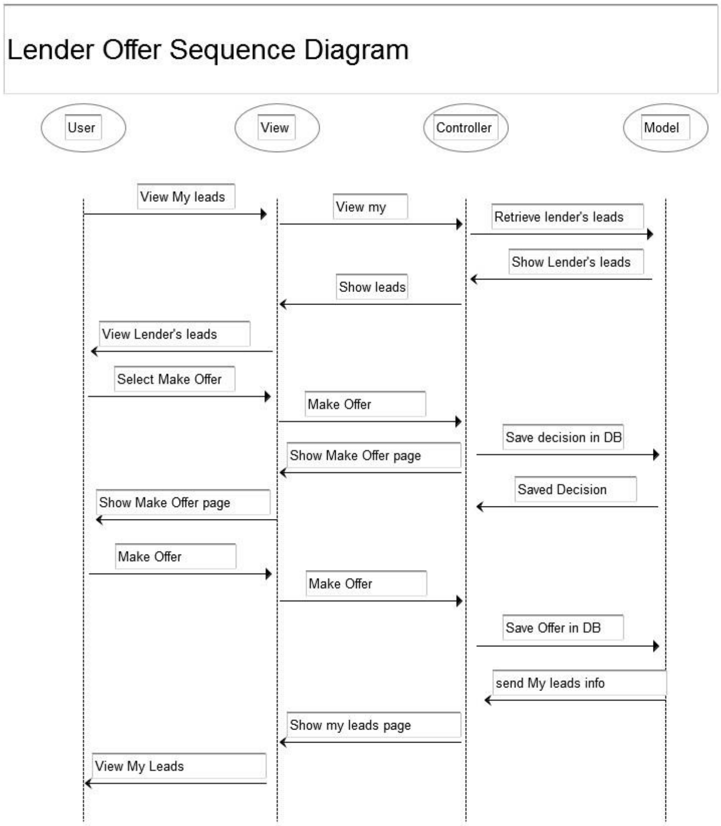
This diagram is the funds diagram which allows the lender to add funds to their account to purchase new leads. The lender selects to add new funds tab. The controller will return the add funds page to the view for the user to view. The user fills in the form to add the funds they want to their account. Once they submit their funds the controller will send this information to the database to be saved as the amount of funds they have in the account. The user will be able to confirm the funds were added by seeing the amount of funds they have on their home page on the top left side of the page.



The lenders can edit their account by selecting the contact info tab for their page. The controller will send a message to the model to retrieve the data from the database. The controller will organize the data to be viewed properly by the user. The user can then select to the edit button to update the information on their contact info. The controller will return the edit form with pre filled in data from the previous time to the user. The user may change the data and submit the data. The controller will breakdown this information to send to the model to save it in the database in the appropriate tables. The controller will then show the contact info to the lender with the new changes of information.

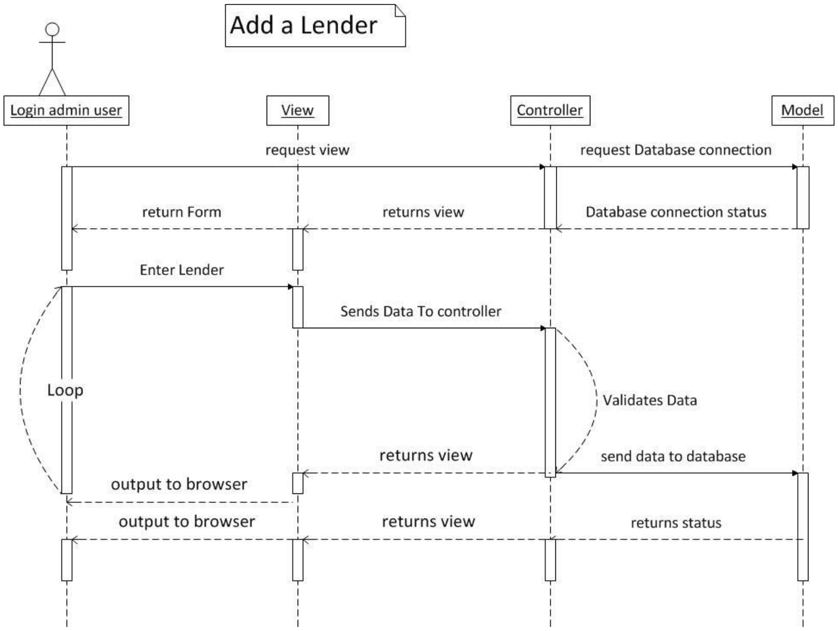


To buy a lead the lender needs to go to new leads. The controller will flow to the model to retrieve the new leads(borrowers) that they can view from the database. This information is then sent back to the view by the controller to display and for the lender to view. The lender can then decide if they can purchase a lead they would like to purchase. However, there is an exception, the lender needs to have enough funds to be able to purchase the lead. If they do the controller will send the information to the database to be able to save the leads information on My Leads page for the lender to view.

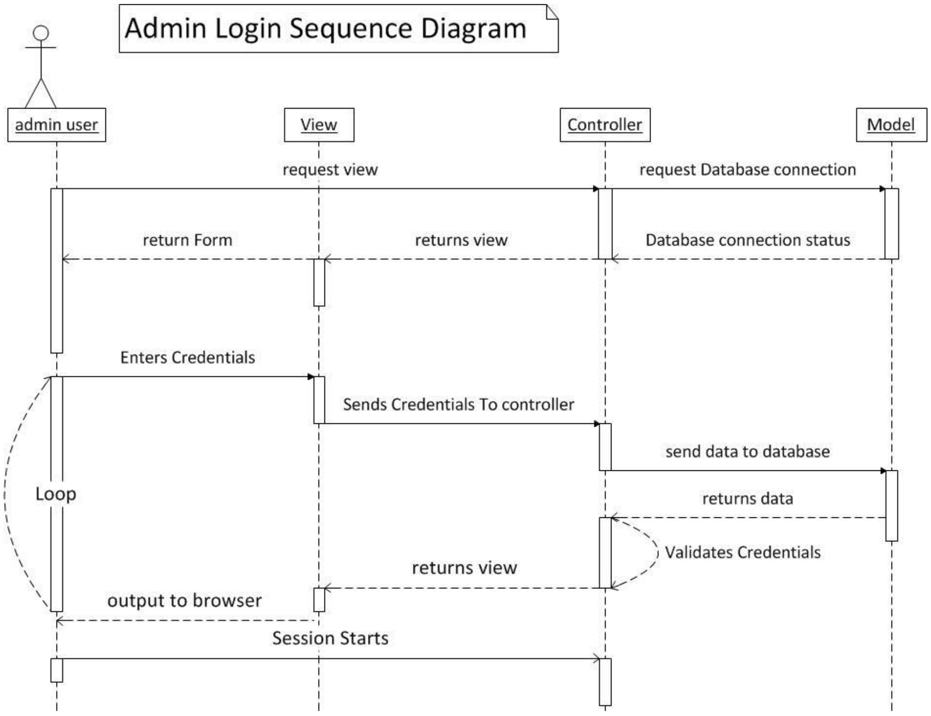


The offer diagram is to show how the lender gives an offer to a lead they purchased. The lender accesses My Leads tab. The controller will retrieve their leads from the model so the user can view the leads they have. Once the lender views their leads they can choose to make an offer to one of the leads by selecting the “make an offer” button. This will open a page up for the lender with the borrower's information so the lender can view and give the borrower an offer in that same page. Once they make an offer to the borrower it is sent to the controller to send that message to the model so it can be stored in the database. Then the controller will send the lender back to the My Leads page to view their leads.

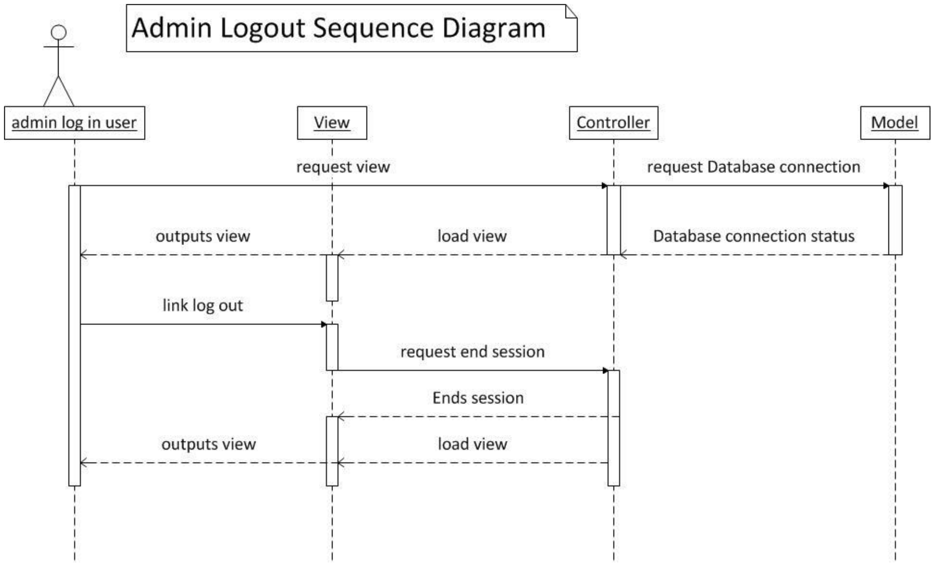
## Admin UML Diagrams



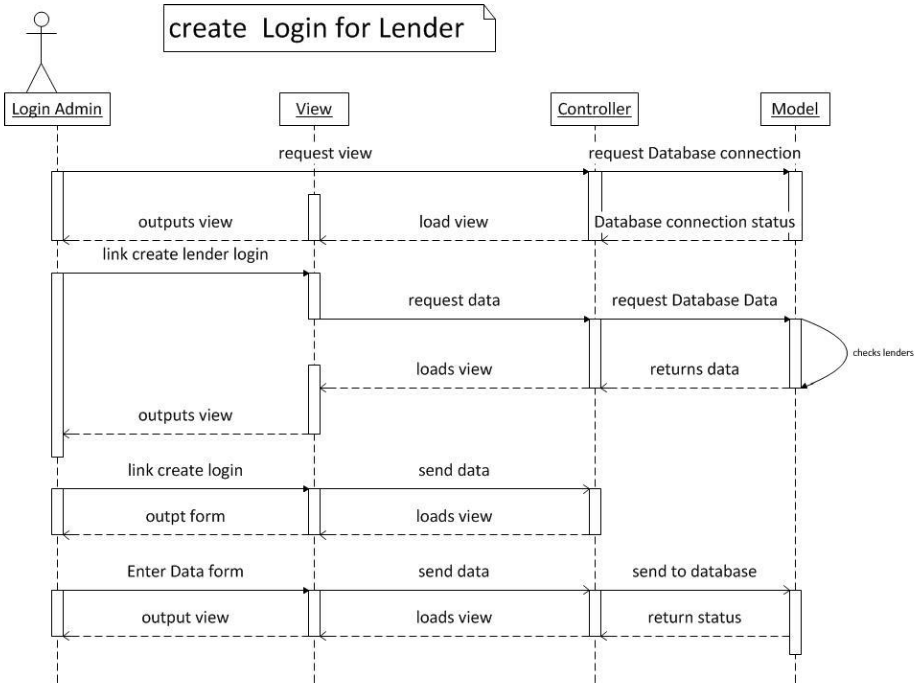
“Add a lender” diagram displays how a login admin user creates lender. The Admin clicks “create a lender” link located in the header section or clicks on the “add lender” button. Once the form is loaded the admin can enter the information of the lender and if it validates it will send the information obtain from the form and save it to the database. Once the information is store it will reload the page with a message if it was success or an error message.



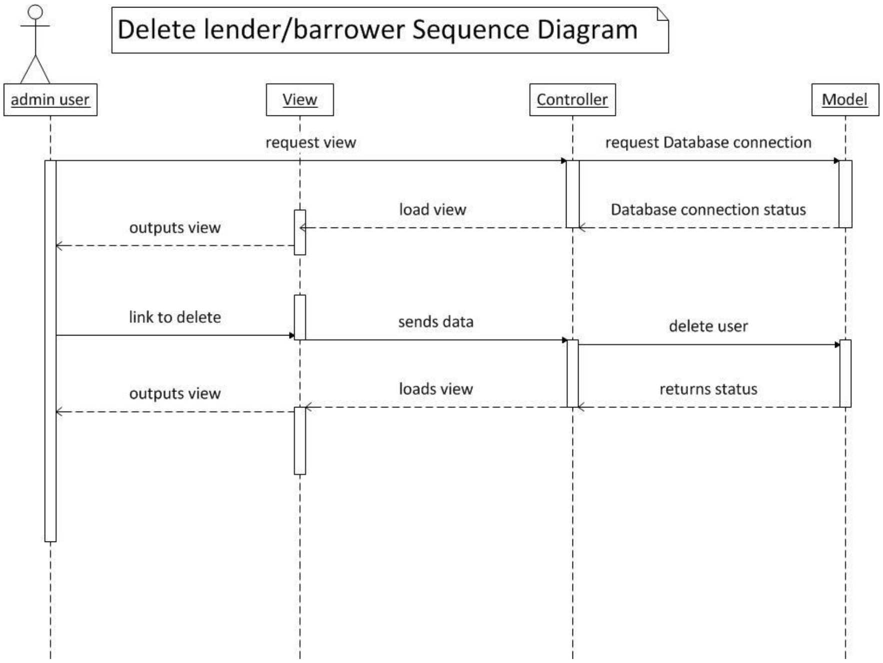
This diagram shows the process login for the administrator side. The administrator enters his credentials and then it goes to the database checks the credentials are correct. If the information is correct it would start the session else it would load the form again with error message.



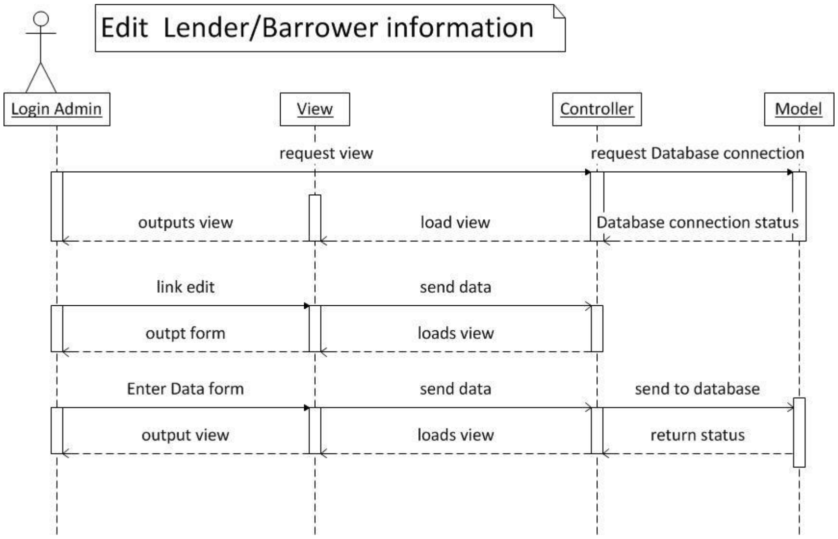
This diagram demonstrates the process for the admin to log out. The administrator logouts by clicking the link in the header to log out. By clicking the logout link the admin ends the session and the admin is route back to login page for the administrator.



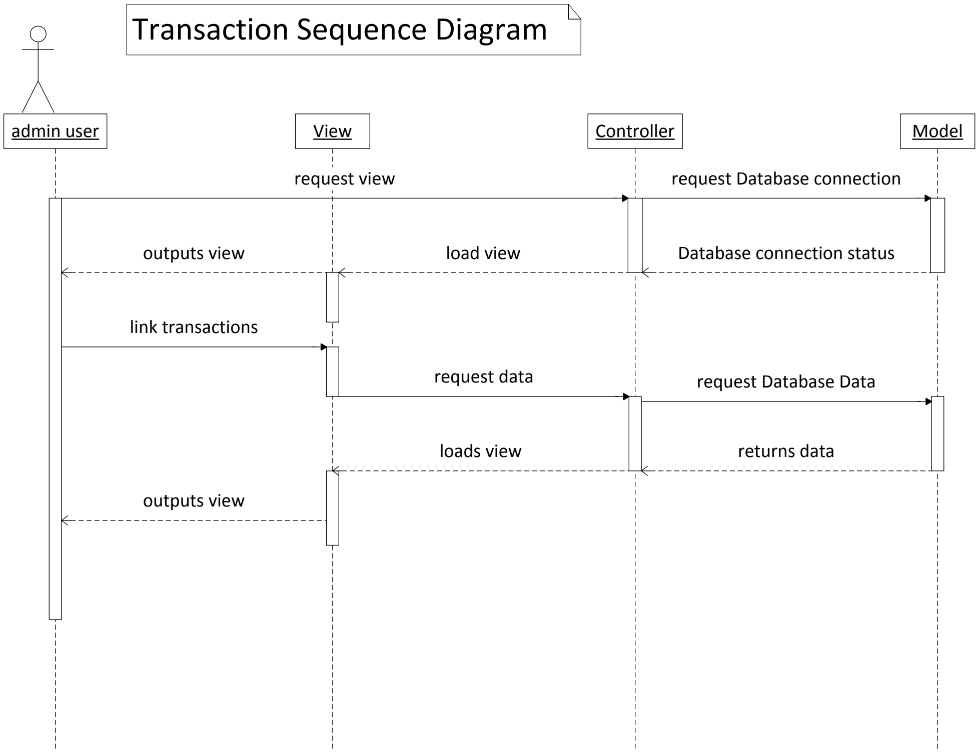
This diagram demonstrates the process of creating a login for the lender that administrator already created. If there are new lenders by clicking create login button they will be send to form for creating the login. In the form the administrator enters the username and if everything is correct it send the information in the database and store the information.



This diagram demonstrates how the administrator user can delete barrowers or lenders. By clicking “delete link” in view transfers data to the controller. The controller sends the information to the database it would delete the user. Finally the controller deletes the information on the view and sends it to the user to see the new list of barrowers or lenders.



This diagram demonstrates how the administrator user can edit barrowers or lenders. By clicking “edit link” the administrator controller sends a form for to the admin user. Once the administrator completes the form the form data is sent to the controller and the controller sends the information to the database where it will be stored.



This Diagram shows how the administrator can view the transactions from the lenders and borrowers. When the admin is login and by clicking the transactions link in the header, the admin can view transactions.

# 7. System Verification

The system verification process was consolidated into one step via test cases, I performed test cases on the Administrative, Lender, and user side of the web service.  I tested the system for all functionality and documented the functionality of the test cases which yielded the most trouble starting off. All test cases were tested multiple times and I was able to collect data on what makes parts of the system pass and what makes parts of the system fail. After all of the testing and debugging the system has a fairly low fail-rate.  All system verification is documented using the template below, followed by our test cases.

| | |
|---|---|
| **Test Case Number** | Unique Identifier for traceability |
| **Test Item** | Brief Description of the test items exercised by this test case |
| **Pre-Conditions** | Conditions must be established before the test |
| **Post-Conditions** | Conditions must be established after the test |
| **Input Specifications** | Bullet points to describe each input require for this test case |
| **Expected Output Specifications** | Bullet items to describe each expected output for this test case |
| **Pass/Fail Criteria** | Bullet items |
| **Assumption and Constraints** | Bullet items |
| **Dependencies** | List of the test cases depended on or are depend by this case |

### Administrators Test Cases

| | |
|---|---|
| **Test Case Number** | Admin test #1 |
| **Test Item** | This case test the functionality log in for the admin |
| **Pre-Conditions** | • One of the following browsers: FireFox,Chrom,IE<br>• Latest update for browsers<br>• visits www.vietatoz.com/site_admin |
| **Post-Conditions** | Establish session routes to home page |
| **Input Specifications** | • Enters username<br>• Enters password |
| **Expected Output Specifications** | • Outputs errors if errors<br>• No Errors routes to home page |
| **Pass/Fail Criteria** | • Pass test for invalid inputs<br>• pass routes |
| **Assumption and Constraints** | Assuming the server of the Database is running<br>Assuming Internet connection is establish |
| **Dependencies** | None |

| | |
|---|---|
| **Test Case Number** | Admin test case #2 |
| **Test Item** | Test case for log out functionality ends it ends the session |
| **Pre-Conditions** | Login as admin user and Database is working |
| **Post-Conditions** | Ends session |
| **Input Specifications** | • clicks link on logout |
| **Expected Output Specifications** | • routes to login page |
| **Pass/Fail Criteria** | Pass |
| **Assumption and Constraints** | • Assume session works |
| **Dependencies** | Test case admin #1 pass |

| | |
|---|---|
| **Test Case Number** | Admin test case #3 |
| **Test Item** | Test case admin can view the transactions |
| **Pre-Conditions** | Login as admin user and Database is working |
| **Post-Conditions** | Ends session |
| **Input Specifications** | • clicks link to transaction |
| **Expected Output Specifications** | • displays all transactions<br>• search transactions<br>○ email<br>○ price<br>○ lender<br>○ status<br>• search transactions |
| **Pass/Fail Criteria** | • pass for searches<br>• pass for displaying |
| **Assumption and Constraints** | • Assume session works |
| **Dependencies** | Test case admin #1 pass |

| | |
|---|---|
| **Test Case Number** | Admin test case #4 |
| **Test Item** | Test case admin can edit members information saves it to database |
| **Pre-Conditions** | Login as admin user and Database is working |
| **Post-Conditions** | Saves changes to database |
| **Input Specifications** | • clicks link to members/lenders<br>• clicks link to edit<br>• Enters Name<br>• Enters Last Name<br>• Enters Address<br>• Enters City<br>• Enters State<br>• Enters Zipcode<br>• Enters Phone/Mobile |
| **Expected Output Specifications** | • outputs success message or fail message |
| **Pass/Fail Criteria** | • pass |
| **Assumption and Constraints** | • Assume database is working |
| **Dependencies** | Test case admin #1 pass |

| | |
|---|---|
| **Test Case Number** | Admin test case #5 |
| **Test Item** | Test case for deleting a user |
| **Pre-Conditions** | Login as admin user and Database is working |
| **Post-Conditions** | Deletes users from database |
| **Input Specifications** | • clicks link to delete |
| **Expected Output Specifications** | • removes user from the table |
| **Pass/Fail Criteria** | Pass |
| **Assumption and Constraints** | • Database has members/lenders |
| **Dependencies** | Test case admin #1 pass |
| **Test Case Number** | Admin test case #6 |
| **Test Item** | Test case for creating lender |

| | |
|---|---|
| **Pre-Conditions** | Login as admin user and Database is working |
| **Post-Conditions** | Saves new lender to database |
| **Input Specifications** | • clicks link create lender<br>• Enters Lenders Name<br>• Enters Address<br>• Enters City<br>• Enters State<br>• Enters Zipcode<br>• Enters phone/fax<br>• Enters Email<br>• Enters website |
| **Expected Output Specifications** | • displays message create<br>• displays validation errors |
| **Pass/Fail Criteria** | Pass message create lender |
| **Assumption and Constraints** | • Database is working |
| **Dependencies** | Test case admin #1 pass |

| | |
|---|---|
| **Test Case Number** | Admin Test Case #7 |
| **Test Item** | Create login for lender |
| **Pre-Conditions** | Login as Admin and database is working<br>There are new lenders create |
| **Post-Conditions** | |
| **Input Specifications** | • clicks on create log in<br>• enters username<br>• clicks on save |
| **Expected Output Specifications** | Success message display |
| **Pass/Fail Criteria** | pass |
| **Assumption and Constraints** | Bullet items |
| **Dependencies** | Test case admin #1 pass |

## Lenders Test Cases

| | |
|---|---|
| **Test Case Number** | Lender TestCase#1 |
| **Test Item** | Editing the lender's information |
| **Pre-Conditions** | 1. Lender profile is created by admin<br>2. Lender is logged in |
| **Post-Conditions** | Lender's information is updated to the information the lender changed and added. |
| **Input Specifications** | 1. Edit lender<br>2. edit address<br>3. edit city<br>4. select state<br>5. edit zip code<br>6. select country<br>7. edit phone<br>8. edit fax<br>9. edit email<br>10. key in update button |
| **Expected Output Specifications** | The information is updated to the changes made by the lender |
| **Pass/Fail Criteria** | 1. While editing any of the aforementioned data if there is a required input that is not inputted, the submission of data will not be validated thus it will not be logged into database causing test case to fail.<br>2. If all of the required data is inputted properly the test case will pass. |
| **Assumption and Constraints** | The lender profile is created by the admin.<br>The lender is logged in. |
| **Dependencies** | No dependencies on other test cases |
| **Test Case Number** | Lender TestCase#3 |
| **Test Item** | The lender tries to purchase leads for his profile |
| **Pre-Conditions** | 1. Lender profile is created by admin<br>2. Lender is logged in<br>3. Lender chooses new leads |
| **Post-Conditions** | Funds are added to lender's account |
| **Input Specifications** | 1. Views new leads<br>2. Selects the purchase lead<br>3. key in the submit button to purchase lead |
| **Expected Output Specifications** | The lender purchases the new lead.<br>The new lead appears in the lender's leads<br>Lender can view all info for the borrower<br>lender can offer a rate to the borrower he purchased |
| **Pass/Fail Criteria** | 1.The lender test case will fail if the lender has insufficient funds in his/her account.<br>2. .The lender test case will pass if the lender has sufficient funds in his/her account. |
| **Assumption and Constraints** | The lender has not purchased the lead yet<br>The lender has enough funds to purchase leads<br>Lender is interested in purchasing lead |

| | |
|---|---|
| **Dependencies** | Depended:<br>Lender TestCase#2 |

| | |
|---|---|
| **Test Case Number** | Lender TestCase#4 |
| **Test Item** | The lender makes a interest rate offer to the borrower |
| **Pre-Conditions** | 1. Lender profile is created by admin<br>2. Lender is logged in<br>3. Lender chooses My leads<br>4. Lender chooses make offer |
| **Post-Conditions** | The lender made an offer that borrower can view and accept or deny from the lender.<br>Lender can update the offer by giving the borrower a different rate. |
| **Input Specifications** | 1. Enter the interest rate offer to the borrower the lender bought as a lead. |
| **Expected Output Specifications** | The lender sends an offer tot he borrower.<br>The borrower decides to accept or deny offer.<br>Lender can view decision of borrower and make new offer or remove the lead |
| **Pass/Fail Criteria** | The lender reserves the right to make an offer to a borrower, whether the borrower chooses to accept it is independent of the test case, test case should always pass. |
| **Assumption and Constraints** | The lender purchased the lead to offer a rate. |
| **Dependencies** | Depended:<br>Lender TestCase#2<br>lender TestCase#3<br><br>depent:<br>Borrower's TestCase to accept and deny offer |

| | |
|---|---|
| **Test Case Number** | Lender TestCase#2 |
| **Test Item** | The lender wants to add funds to his account to purchase leads |
| **Pre-Conditions** | 4. Lender profile is created by admin<br>5. Lender is logged in<br>6. Lender chooses add fund tab |
| **Post-Conditions** | Funds are added to lender's account |
| **Input Specifications** | 1. Enter name on card<br>2. enter card number<br>3. enter card type<br>4. enter expiration date<br>5. enter security code<br>6. enter amount<br>7. key in the submit button |
| **Expected Output Specifications** | Bullet items to describe each expected output for this test case |

| | |
|---|---|
| **Pass/Fail Criteria** | If the user has an account this test case should always pass. |
| **Assumption and Constraints** | The lender enters the right information<br>The lender needs to add funds to purchase leads |
| **Dependencies** | Depended:<br>Lender TestCase#2 if low on funds |

## Borrower’s Test Cases

| | |
|---|---|
| **Test Case Number** | Borrowers TestCase #1 |
| **Test Item** | Creating a Borrower user |
| **Pre-Conditions** | 1. Internet connection established<br>2. Database connection established<br>3. Select SignUp Link or Create Profile Link |
| **Post-Conditions** | 1. If all data entered was correct then a unique user record is created in the database. |
| **Input Specifications** | 1. User enters email address<br>2. User enters Password<br>3. User enters confirm Password<br>4. User enters First Name<br>5. User enters Last Name<br>6. User keys button to validate information<br>7. Information is submitted to the database |
| **Expected Output Specifications** | Borrower's profile is created and can now login |
| **Pass/Fail Criteria** | 1. If any entered data was invalid then no user record is created in the database.<br>2. If email already is registered then the new record being created will not be allowed and the User will receive a message notifying them email is already registered in the system. |
| **Assumption and Constraints** | Assuming all data passes validation even if wrong data it will create user |
| **Dependencies** | Dependent on this case:<br>1.Borrowers TestCase #2<br>2.Borrowers TestCase #3<br>3.Borrowers TestCase #4<br>4.Borrowers TestCase #5 |

| | |
|---|---|
| **Test Case Number** | Borrowers TestCase #2 |
| **Test Item** | Borrower's creating a profile |
| **Pre-Conditions** | 1. User Signed In<br>2. User selects Create Profile tab<br>3. Fill the form in the page |
| **Post-Conditions** | 1. Once data is entered correctly and selected it is saved in the database. |
| **Input Specificatio** | 1. User selects property usage2. User selects state<br>3. User enter city<br>4. User Keys Next button<br>5. User Selects Loan estimate<br>6. User Selects Down payment percentage<br>7. User Keys Next Button |

| | |
|---|---|
| | 8. User Selects Credit Rating<br>9. User selects Foreclosure answer<br>10. User selects Credit Card debt<br>11.User is shown Data entered<br>12. User sends data to database |
| **Expected Output Specifications** | A new profile will be created for the borrower |
| **Pass/Fail Criteria** | 1. If any entered data (city) was invalid then the user will not be allowed to move forward in the form. |
| **Assumption and Constraints** | Assumption that all data is entered correctly as well as constraints |
| **Dependencies** | Depende:<br>Borrower's TestCase#1<br><br>Depent:<br>1.Borrower's TestCase#3<br>2.Borrower's TestCase#4 |

| | |
|---|---|
| **Test Case Number** | Borrower's TestCase#3 |
| **Test Item** | The user can edit their profile they just created |
| **Pre-Conditions** | 1. User Signed In<br>2. User selects Profile tab<br>3. User selects edit button |
| **Post-Conditions** | 1. If all data entered was correct then the user has changed data successfully and sent back to profile page with updated information. |
| **Input Specifications** | 1. User sent to edit form<br>2. User edits Use property as<br>3. User edits city<br>4. User edits state<br>5. User edits zip code<br>6. User edit loan amount<br>7. User edits down payment percentage<br>8. User edits Credit rating<br>9.User edits foreclosure<br>10. User edits credit debt<br>11. User keys in update button or cancel button |
| **Expected Output Specifications** | 2. User updated Use property as<br>3. User updated city<br>4. User updated state<br>5. User updated zip code<br>6. User updated loan amount<br>7. User updated down payment percentage<br>8. User updated Credit rating<br>9.User updated foreclosure |

| | |
|---|---|
| | 10. User updated credit debt<br>11. User keys in update button or cancel button |
| **Pass/Fail Criteria** | Should always pass. |
| **Assumption and Constraints** | The user updates all data correctly.<br>The user updates all or specific sections from profile |
| **Dependencies** | Depended:<br>Borrower's TestCase#2 |

| | |
|---|---|
| **Test Case Number** | Borrower's TestCase#4 |
| **Test Item** | The user capable to delete their profile they created |
| **Pre-Conditions** | 1. User Signed In<br>2. User selects Profile tab |
| **Post-Conditions** | 1. User selects and ok the delete button the data is removed from the database |
| **Input Specifications** | 1. User Keys in delete button<br>2. User is show message box<br>3. User Keys in Ok button<br>4. User's profile is deleted from database<br>5. User profile removed from page |
| **Expected Output Specifications** | Profile is deleted by the user |
| **Pass/Fail Criteria** | The user should be able to edit their own profile, this should always pass. |
| **Assumption and Constraints** | Assuming the user is in profile and chooses the delete button |
| **Dependencies** | 1.Borrowers TestCase #2 |

| | |
|---|---|
| **Test Case Number** | Borrower's TestCase#5 |
| **Test Item** | Accept or deny the lender's offer |
| **Pre-Conditions** | 1. User Signed In<br>2. User is in home Page<br>3. selects view Lender link |
| **Post-Conditions** | 1. Offer decision is shown in User's home page and saved in database for user and lender to view User's decision on offer. |
| **Input Specifications** | 1. User sent to lender's offer page<br>2. User Selects offer option<br>3. User keys submit button |
| **Expected Output Specifications** | The borrower will decide to accept or deny the offer. If so they will submit their decision and show in the borrower's home page. |
| **Pass/Fail Criteria** | If there is no offer sent then there is a stalemate of this test case, otherwise the borrower should always be able to |
| **Assumption and Constraints** | Assuming the borrower received an offer by a lender |
| **Dependencies** | Dependent:<br>Lender's test case to give an offer |

| | |
|---|---|
| **Test Case Number** | Borrower's TestCase#6 |
| **Test Item** | The borrower's contact information can be edited or add missing information. |
| **Pre-Conditions** | 1. User Signed In<br>2. User selects Contact info  tab<br>3. User selects edit button |
| **Post-Conditions** | 1. If all data entered was correct then the user has changed data successfully and sent back to contact page with updated information. |
| **Input Specifications** | 1. User edits First Name<br>2. User edits Last Name<br>3. User edits address<br>4. User edits city<br>5. User selects state<br>6. User edit zip code<br>7. User selects country<br>8. User edits phone number<br>9.User edits mobile phone number<br>10. User keys in update button or cancel button |
| **Expected Output Specifications** | The output expected is the updated/ edited contact information |
| **Pass/Fail Criteria** | |
| **Assumption and Constraints** | Assuming the borrower is logged in |
| **Dependencies** | 1.Borrowers TestCase #2 |

# 8. Conclusions

## 8.1 Summary

At the commencement of the web-service development process I brainstormed a simpler way for something practical. In all actuality I did not have an idea of what I wanted to simplify, at least not one that I wanted to share with the class. A family member was discussing how difficult it was for him to find a mortgage loan for his house. Then my inception snowballed; then I had idea and another and then finally decided to create a web-service that would aid a person looking for a mortgage loan while simultaneously allowing mortgage lenders look for prospective clients. I believed that this process would be easier than it was since there was unlimited creative freedom. However I soon found that with all that freedom an immense of amount of organization and responsibility was required. I organized, researched, learned and implemented in a fast pace while balancing time between work, school and life. It was a very challenging experience and some of the technologies required were new to me, so I had to catch up while building. The toughest part of this experience was getting my ideas to work exactly as I had thought them out. MortgageDeals simplifies a practical need for many people..

## 8.2 Problems Encountered and Solved

The most prevalent “bug” in my experience was organization (lack thereof). At times I lacked the direction that students sometimes have in bland cookie cutter assignments. The alpha and beta presentations were a fantastic way to keep students from falling off track initially. I had an ambiguous understanding of what was meant by code complete for our beta, so I worked off of the professors suggestions after the Beta presentation. I initially thought I could expedite a waterfall software engineering approach and realized that I was not gaining any traction as far as the working product was concerned so I chose to change things up and develop using SCRUM. The most difficult problem was establishing a connection with the Zillow (home mortgage website) API, I was trying to “homegrow” all of the code so it became a challenge creating functions that would handle the connection, I finally stumbled upon cURL which already had functions to handle URL connections to API’s so I ended up using that

## 8.3 Suggestions for Better Approaches to Problem/Project

Perhaps an example of a code complete project would serve the class as a clear product to strive for. Maintain progress reports because they were also a good way to keep the class doing work since we didn’t want to turn in “nothing” as an update. Preserving the presentations since the presentations are another impetus for fast development. Going beyond that in between presentations pairing up people and have the people present among each other would be a good way to have an intimate presentation and perhaps receive intimate feedback. A Github repository lecture, where the professor does a walkthrough with the students in establishing a repository for their team would be of good service for all students. Other than that the development experience feels appropriate for the time and setting.

## 8.4 Suggestions for Future Extensions to Project

Having students focus more on modern technologies, I decided to create an MVC desktop web-service, but I observed all of the mobile applications were very exciting to see in development. Or perhaps include a mobile aspect to all the web-services. If I could do it again I would have built a native web-service app.