Lend and Lease

Group 3

March 24, 2017

UPPSALA
UNIVERSITET

Abstract

A web application for lending and leasing items in your area.

Contents

1 Motivation for our Website **3**

2 Related Work **3**

3 Business Model **3**

4 System Requirement Specification **4**

5 System Architecture **5**

6 Use Case **5**
6.1 User Interaction with Items 5
6.2 User Registration & Profile 6

7 ER model **7**

8 System Implementation & Evaluation **8**
8.1 Implementation of functionality 8
8.1.1 Add Items : 8
8.1.2 Profile : 9
8.1.3 Browse Items : 9
8.2 Development Tools 10

9 Technical Problems Encountered **11**

10 Testing and Security **11**
10.1 Testing 11
10.1.1 Siege performance test result 11
10.1.2 Persona 1 11
10.1.3 Persona 2 12
10.2 Security 12
10.2.1 Password Encryption 12
10.2.2 Session Implementation 12

11 Group Management **12**

12 Future Work **13**
12.1 Chat Functionality 13
12.2 Premium Account 13
12.3 Enhanced Filtering 13
12.4 Location Loading 13
12.5 Testing 13

13 Conclusion **13**

1 Motivation for our Website

Most people have valuable items lying around their home that they rarely or never use. Many expensive items may only serve a purpose once, or at least very few times. These items will not serve a purpose daily, but they are still not something that they want to sell or get rid off. It is for items like these that this website exists. With our website people can upload an image and input some information about the item and then lend or lease it out to people in their area. It can also work to promote neighborliness and bring people closer to each other since this is a system based on trust between the users.
Much like any other service like eBay or Blocket, the laws being applied when acting against the policy of the site is in most cases covered by the country's laws.

2 Related Work

There are similar services out there. However we felt that those services are relying too heavily on people's goodwill. We thought that for people to lend out more valuable and useful items they might want some money as compensation. This would mean that our site would have items of great value(to rent) and of lesser value(to lend). This would make the diversity of the items in the system a lot more desirable for the common user, we thought.

3 Business Model

The vision of what the site should be is rather simple: to let everyone that visits the site have access to the core of functionalities of the site. This means allowing the users to browse the available items in their area without logging in. Since this is the initial page, the user will not have to navigate through the site to find out what they can do. The user will be prompted to use their current location and see the items available where they are, and if they decline, it will go to the default area of Sweden (where the website was created). With that map, they can filter items based on categories, and click on items to see more details.

Our business is to populate the site with users that appreciate the site's simplicity. Everything is handled between the two people involved in the renting or lending, leaving the company outside of legal issues. When the user-base is large enough, we would start to put advertisements on the page. For example, when a user is looking at a screwdriver that he or she wants to borrow, there could be an ad for Amazon, showing a similar screwdrivers they have for sale, etc.

We would also offer a premium membership for people that rent-out their things, allowing them to be featured on the site so that people can access their personal page and browse what items they offer. The premium account would also remove advertisements.

Currently, the website does not enable users to put an item to be leased, nor is there a premium account. With the way the website is structured, adding an option for leasing an item would not be difficult.

4 System Requirement Specification

- **Create Account**: This Function would allow users to Register themselves and start using the site, in a broader way than just browsing items on the map.
- **Log in & Log out** :This function allows pre-existing users to log into their account, and log out when they are done.
- **Notify users by sending emails**: When a user requests an item from the map-page, an email is sent to the owner of the item telling that person the item has been requested. In that email the contact information of the requester will be appended so that they can interact and decide how to carry on.
- **Adding new Items to the map**: Adding new items that users want to lend to other people can be done with this "Add new Items" form . This requires the user to be logged in so that the item can be linked to that user.
- **Send request for item to user**: Allow signed in users to send a request (via email) to the holder of the requested item.
- **Filter Items by Categories**: Able to filter items by Categories.
- **Search for items by name and categories** : Able to search items by its name and categories. This functionality was not implemented due to problems with the Google Maps API. The markers information was not visible for the search function.
- **Filter by Location**: Able to Filter the items by location was not implemented but would be a simple addition in the future.
- **View Item Requests**: Able to view Item Requests is supported by the back-end. We have a table storing requests linking them to the user that sent the request and the user it was meant for. What's left is to implement it in the front-end.
- **View posted items**: Able to view all posted items. Displayed as markers on the map.
- **Regarding Requests**: As previously mentioned the full support for requests has been implemented in the database, as represented in the ER diagram in

5 System Architecture

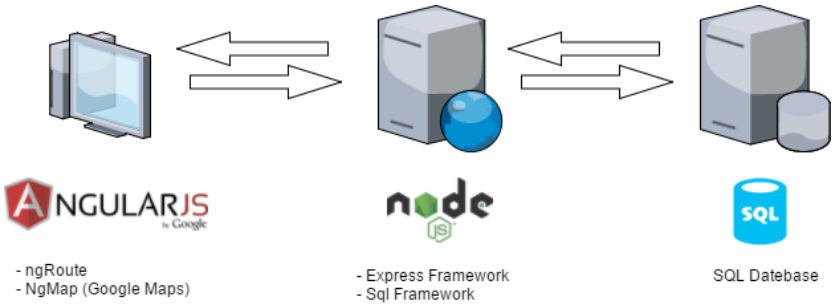

Figure 1: System Architecture

We have chosen a 3-tier architecture consisting of SQL database, Node.JS server and AngularJS client. The SQL database contains the data for the items and users. The Node.JS server performs queries against the database server and delivers JSON data and web pages to the client. The database was hosted on a live server enabling us to easily share the data across development machines. The server was hosted on the live server eventually, but during development phase, it was a local server
Several frameworks were used in implementing the server. Passport JS was used to handle authentication and security. "SQL" framework was used to handle SQL queries and database security. Express framework was used to handle HTTP actions (GET and POST)
To implement the client side pages, modules were used to handle server calls, connect to Google Maps, and other AngularJS specific actions like page routing and global variables creation.

6 Use Case

As previously mentioned our user-experience is built on the philosophy that a user should be able to access the core functionalities of the site without having to register and logging in. The user should get a good idea of what the site is about and what it offers before he or she is forced to register their email and come up with a password for the site.
For convenience, we have separated Use Case Diagram into two :
1) Users Interaction with Items
2) User Registration and Profile

6.1 User Interaction with Items

- **Request Items** : Borrower should be able to send request for Items that he/she sees on the Item lists. The request is in the form of an email to the lender.
- **Browse Items**: The user can browse different Items that are available.

- **Filter Available Items**: Able to filter items by type.
- **Place an Item** : Lender is able to place items he/she wishes to lend.
- **View list of personal items** : The user can view a list of personal items uploaded.

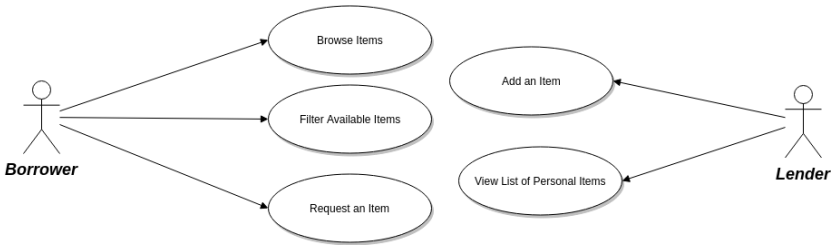

Figure 2: Users interaction with Items

6.2 User Registration & Profile

- **Sign up**: New User Registration
- **Sign In** : Sign In for Existing Users
- **View Profile** : View one's profile.
- **Sign out** : Sign out for logged in users.

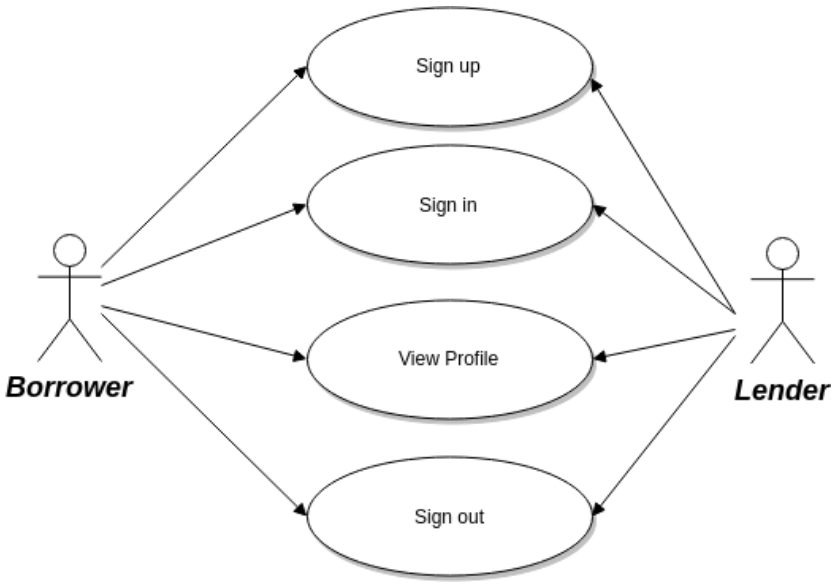

Figure 3: User Registration & Profile

7 ER model

All user-data and item information is stored in our MySQL database. It was also intended that the requests would be stored there as well. This however was, as previously mentioned, never implemented in the server-code or the front-end. The main entities of our System are : User, Items and Requests as shown in our ER Diagram generated by the backwards engineering function in MySQL workbench. Associated attributes are displayed within the entity blocks in the diagram.

- **Table Specification and Relationship Explanations**: The tables in our database are related and implemented in the following way :
  - **Requests** : A request in the requests table has a primary key that is the request id which uniquely identifies each request. There are also foreign keys to user, from user and requested item. Which all point to distinct rows in the users table and the items table. Specifying what item the request is regarding, what user sent the request and what user received the request.
  - **Items including tables such as Books, Tools, Etc** : All items stored in our database can be found in the items table. If it has a specific category it will also be stored in its respective table. For example, if the item is a book, it is stored in the items table and the books table with the same primary key. This means that adding additional categories is simply a matter of adding another table using the same primary key as the items table.
  - **Item subcategories tables such as book categories, tool categories etc**: These tables' primary keys are foreign keys in each category table. For example in the table Books there is a foreign key called book category id which in book categories specifies what that subcategory is called. A horror book is then stored in items, books and in books it is specified that the subcategory is Horror.
  - **Users** : The users table is fairly straight-forward. There the users' password, email and other information is stored. The password is encrypted before being stored in the table so that a if anyone gained access to the users table they would still not get hold of any sensitive information.

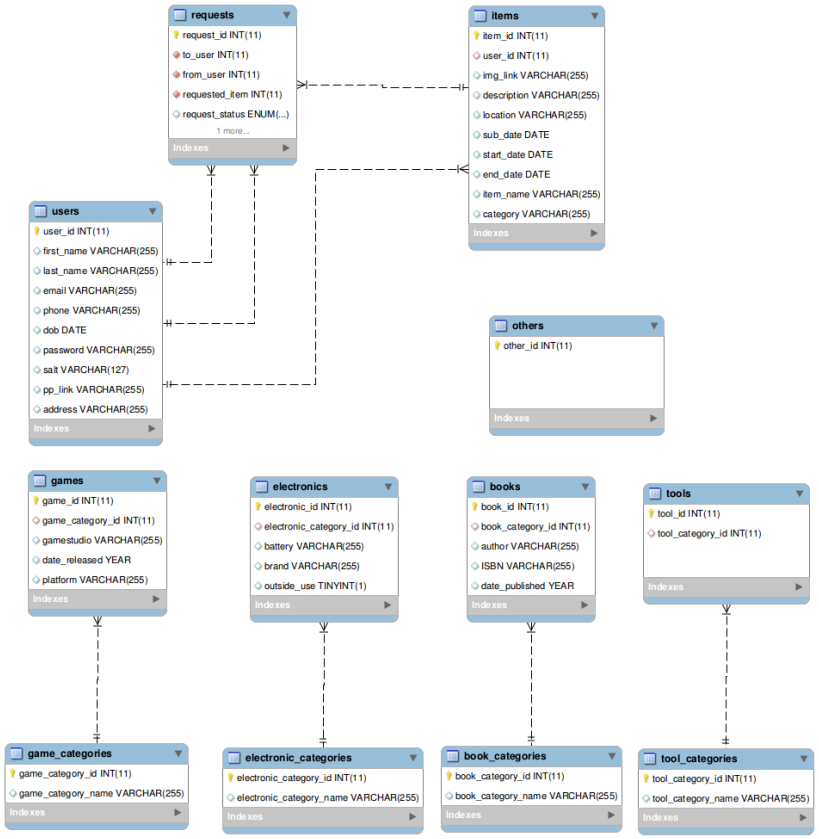

Figure 4: ER Diagram

8 System Implementation & Evaluation

8.1 Implementation of functionality

The functionalities we were able to implement were:

8.1.1 Add Items :

This page allows users to add items to the database. The user can select from a variety of categories, and each category contains their own subcategory. Each item is required a name and description, and depending on the subcategory, additional information can be added. The user can use their current location to add to the item, or they can type in an area and use that location. To ensure the right location is shown, a marker will be displayed on a map. The last part of add item is adding an image to go along with the item.
The technical steps include loading AngularJS view for the add item page which will check if the user is logged in. Then the user enters the information which will be used to construct a JSON object that is sent to the server using a POST method. The location of the user will be taken from Google Maps and added to the JSON object. The server will then insert the information into the database

according to the item type. Then the server will return a confirmation message to the client, or an error message in case a server or SQL error occured.

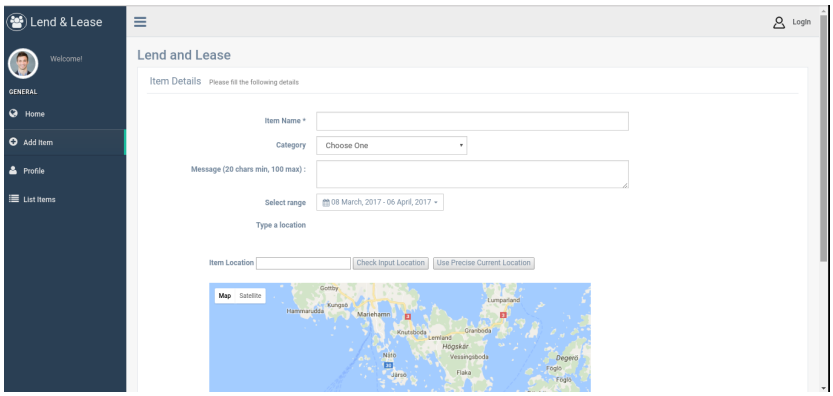

Figure 5: Add items

8.1.2 Profile :

To be able to view user's profile.
The technical steps include loading AngularJS view for the profile page witch will make HTTP GET request to the server. The server will then check the logged in user and returns information about that user. The information is then displayed in the corresponding fields.

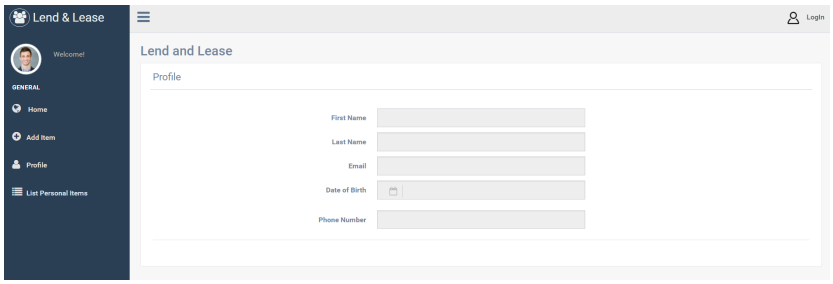

Figure 6: user Profile

8.1.3 Browse Items :

Users of the website don't need to create accounts to view the items available on the website. This is done so that users can explore the website before deciding to create accounts. However, if a user wants to request an item from the owner, he/she must have an account. On the main page, there is a Google Map with markers corresponding to the items in the database. The user can filter the items using the filter options above the map. By clicking on a marker, information about that item will be displayed in a small pop up. The user, if logged in, can choose to request that item which will send an email to the owner of the item. The technical steps include loading the AngularJS view first. Then an HTTP GET request is sent to the server which will return all the items contianed in

the database. It is worth mentioning that for future work, scaling of database record should be handled. One simple way is to include the location of the user in the request and return only the items in close proximity to him/her. After recieving the data from the server, the map is populated with markers that contain information about each item. Clicking on an item will triger a listener that shows the pop up which has a button to request the item. Clicking on the button will then make another request to the server. The server will then send an email to the owner of the item.

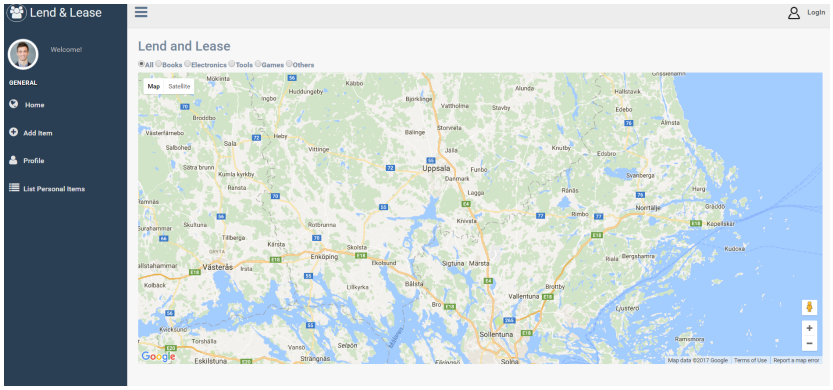

Figure 7: Maps from Home page

8.2 Development Tools

1. Github: For version control, we used Github to share code.

2. AngularJS : The Front-End of the system is using Google's own framework - AngularJS. This is a perfect language for building dynamic web-applications and it fits perfectly with what we wanted our website to be. We wanted it to be a single-page application, with a sidebar for navigation and functionalities, leaving the center of the page to display information to the user and let him or her input their own information.

3. NodeJS : The server built with NodeJS, using the express package to handle the HTTP requests and responses. What makes us satisfied with this choice is that NodeJS also has a package for communicating with MySQL databases which is our database of choice.

4. MySQL : For our database management we are using MySQL. The reason being that it is widely used and well-documented. Since AngularJS and NodeJS are somewhat new and poorly-documented we wanted something sturdy and trusted running the background if all else failed.

5. node_modules

   - bcrypt-nodejs : used for encrypting user password. [1]
   - express : used as a NodeJS web framework. [3]
   - passport : used for Authentication. [2]

9 Technical Problems Encountered

As we picked new languages and framework such as Angular and Node, it was difficult to get around with it. If we encountered some error, it was time consuming and annoying to find a fix for it. Also, there was not good enough documentation available to get a quick overview.

10 Testing and Security

10.1 Testing

We have not applied any kind of test-driven development during our progress of the project. After observing the usage of our finished product amongst users with different backgrounds, we have forged some personas which covers the typical user test cases. We have also performance tested the finished page with the framework 'siege'.

10.1.1 Siege performance test result

Below we can see a picture of our performance test during our siege. It handled requests from 10 000 users with a fairly high increase in latency from the server. However, it did not cause the server to malfunction, which is a result we are satisfied with.

Figure 8: Screenshot from the siege test

10.1.2 Persona 1

Name: Gustav Farthinder
Age: 30
Profession: IT consultant
Computer habit: High

Gustav is a frequent user of pages like Blocket and Amazon. Suddenly, it occurs to him that he has several different tools that are not used by him often. He does, however, not want to sell them as he might have a require them in the future. Gustav comes up with the brilliant idea to lend them out when he's not using them. As he is highly involved with the Internet and all of its possibilities, he searches for pages that supply this service. He stumbles upon the page 'Lend and Lease' and feels that this serves his need perfectly! As he enters the site, he immediately shares his location and sees a map with nearby objects that are

up for lending and leasing. He is not interested in loaning someones items, he wants to lend his own items out - and understands that he probably has to sign up to be able to get in touch with possible loaners. He cannot find any "sign up"-button, so he presses Login. He is taken to a login-page, which also has a button to sign up. He gets taken to the sign-up menu, where he fills in all his details and adds a profile image of himself. After this, he is logged in and ready. He gets straight to his point and see the "add item" in the left bar. He clicks it. He fills in all details about his lendable item and chooses his current location (as that is where the item is). He continues to add the other items which he wants to lend out. After he is finished, he simply waits for people to send him requests. Gustav is highly satisfied with his stay and loves the page.

10.1.3 Persona 2

Name: Jeanette 'Nettish' af Oxjärpe
Age: 27
Profession: World famous painter
Computer habit: Below moderate

Nettish is not too familiar how to handle the situation, now that she realizes that he has several brushes which she does not frequently use. She wants to lend them out to people. After googling, she finds the page 'Lend and Lease'. Nettish goes straight for finding how to put up her stuff for lending, without registering first. As there is no "register here" button on the main page, the register button is on the login page. Nettish has trouble finding this, as she does not want to login, she wants to register. She finally finds it and registers. Because she is filled with rage from his time wasted on not finding his way to add an item, she misses that she has chosen the wrong location for her item when she has added it. Unfortunately, an edit item function has not yet been implemented. Nettish simply gives up and leaves the page, never to use it again. If she wanted to change it, she would have had to delete her item and make a new entry, which is a quite unnecessarily long way to go, for a pretty simple task.

10.2 Security

10.2.1 Password Encryption

We used native NodeJS library called bcrypt for encrypting and storing users password which uses the latest Security Recommendation for cryptographic algorithm. [1]

10.2.2 Session Implementation

Establish login session for user and destroy it when logged out.

11 Group Management

We initially started with everyone trying to understand how to use node.js and AngularJS. One everyone knew the basics, we were to familiarize ourselves with

the frameworks we planned to use. As the time went by, we had to get started on the project, and so we started dividing work as front end features and back end features.

As the project was being worked on, and after midterm presentation, we had to change the task division because not everyone was working at the same pace and the deadline was approaching. In the end, the final work division was done as two members on front-end, one on back-end, one on testing, and everyone working on the report.

12 Future Work

The feature that we initially planned to implement but were not able to because of limited time was:

12.1 Chat Functionality

We initially planned to implement chat functionality in our App. This feature would enable requester and requestee to interact with one another and plan to exchange the requested item(s) they have.

12.2 Premium Account

This feature would be part of our payment plan, and would enable users with a premium account to avoid advertisements and be featred on the website.

12.3 Enhanced Filtering

The current filtering leaves a lot to be desired for users. With our planned enhanced filtering function, users would be able to type and search what they want instead of just narrowing down the categories.

12.4 Location Loading

With location loading, only items with locations in a close proximity to the user would be loaded, reducing loading time and improving speed for the user.

12.5 Testing

We didn't have time to do testing in an efficient and correct way. Given more time, we would have liked to implement some testing frameworks such as mocha which is a javascript test frameworks that runs on nodejs. Also, we would have liked to use some standard testing methods like: QAcomplete, HP QC etc and create test cases and run them properly in parallel with development.

13 Conclusion

The idea of this project is to create a website that would help people share items that they don't usually use. Users can find such items on a map and can easily filter those items based on item categories. The website was deployed on

a live server at the end of the project where it can be acessed publicly. Some functionalities were initially thought of but were not implement due to the lack of time and the technologies used being new to the development team. Overall, the basic idea of the project was showcased and additional work could be done later to expand the scope of the project.

References

[1] bcrypt-nodejs, A native JS bcrypt library for NodeJS, [Online]. Available: https://www.npmjs.com/package/bcrypt-nodejs

[2] Authentication, "Easy Node Authentication", [Online]. Available: https://scotch.io/tutorials/easy-node-authentication-setup-and-local

[3] Express, "Web Framework for NodeJS", [Online]. Available: http://expressjs.com/