SAN FRANCISCO STATE UNIVERSITY

SW ENGINEERING CSC648/848 SECTION 02 SPRING 2017

GATORLIST – TEAM 11

HASAN NIFTIYEV
hasan.niftiyev@yahoo.com

JORDAN SCANDLYN
DARYL ROBBIN
ANDRE SIMPELO
ANTHONY SEARLES
HIN VONG

MILESTONE 2
REVISION 1 – INITIAL STATE
MARCH 16, 2017

1. Use Cases

2.1 Guest User

Jeremy is a student at SFSU with very limited technological skills who is browsing the website for the first time looking to buy an Item. Jeremy starts browsing the available items, he can also choose to specify a category. To keep his purchase within his price limit Jeremy could sorts the items by price or category. As a ***Guest User*** Jeremy is able to access the descriptions and pictures of items. Jeremy sees a book that he likes, he wants to buy it but at this point he can only browse ***Items***. When he attempts to proceed with his purchase he must either log in to his account or register a new one. As a guest user with no ***Verified User*** account Jeremy must enter SFSU email along with a password to create his account, as well as opt in to our terms and services agreement. His e-mail submission must have a SFSU suffix. After the account has been created Jeremy is redirected to the Item page he was previously viewing, where he may use the messaging system to contact the seller, or use any of the sellers selected contact information where they can discuss price, meeting times, or deliver options.

2.2 Admin

Sally is a skilled ***Admin*** of GatorList who has opened the website login page and logged into her ***Admin*** console, where she sees a message in the support messages from a ***Verified Users*** about a dispute that has occurred involving the website. Sally is able to message both the disputing users and during that process she is able to determine that the seller was being misleading about the quality of the ***Item***. Seeing a breach in the terms and services agreement, Sally now has the option to block the seller's account from making sales, as well as remove the page of the related item. After resolving this issue, Sally chooses to go over posts from couple days ago, and check the content of those newly posted ***Items*** and their media for violation of terms and services agreement. If she finds an irrelevant picture/post for an ***Item***, she is able to hide the post and notify ***Verified User*** per agreement of contract.

2.3 Verified User

Alexander is a student at SFSU who has previously used the website to make a purchase and now would like to post his own calculator for sale. Upon opening the website, he proceeds to post his calculator for sale, but he is prompted to log in or register for an account. Since Alexander is a previously ***Verified User*** he enters his email and password to be logged in. Alexander enters the name of his ***Item*** for sale, selects a category and must upload at least one image of his ***Item*** (He has the option to upload additional images if he wishes to). Here he is also able to leave contact information for those interested in this ***Item***. After he is done Alexander can add his new ***Item*** for sale to the list of available items to purchase. When a **Verified User** wishes to communicate with Alexander about his post he will receive a notification via our messaging system.

If Alexander would like to buy something instead of selling it, all he needs to do is browse the item he is interested in, log-in using his SFSU credentials, interact with the seller via our messaging system and after both parties are satisfied with the transaction he will be prompted to rate the seller. If there is a problem with the site or if the seller does something reportable he can contact support and the message will reach a site Admin. The records of the messages between Alexander and the seller are recorded and saved for admin mediation purposes, as stated in the terms and services agreement.

2. Data Definition

2.1 Guest User

Person who can only browse and do nothing else on the website

2.2 Verified User

Person who can browse. They can post new items and even edit their own items if they wanted to. They are able to communicate with other verified users about their item for sale. And if there is a privacy violation they can report it to the system administrator. Verified Users shall be able arrange a safe place to meet up.

2.3 Admin

Has the power to edit, delete and block a user's account if that person violated the terms and services agreement. Admin can monitor and filter any unusual activity. If an Admin detects any posting violations they enforce preventive precautions, accordingly.

2.4 Item:

- Books or class notes.
- Service related to education.
- Electronic gadgets.
- Accessories or clothing.
- Furniture.
- Mobility / Commuting device.

3. Functional Requirements

Application Functions:

Priority 1:

1. Login
2. Browsing
3. Sorting items
4. Searching items by percent-like algorithm
5. Posting
6. Contact Sellers
7. Upload up to 5 pictures and check format
8. Display posting date
9. Remove item
10. User Agreement on registration
11. Responsiveness

Priority 2:

1. Set expiration for items
2. Rating for Users

Priority 3:

1. Live Chat
2. Rotate Items like Carousel in Home Page
3. Renew item
4. Real-time Search
5. list items as gird or list

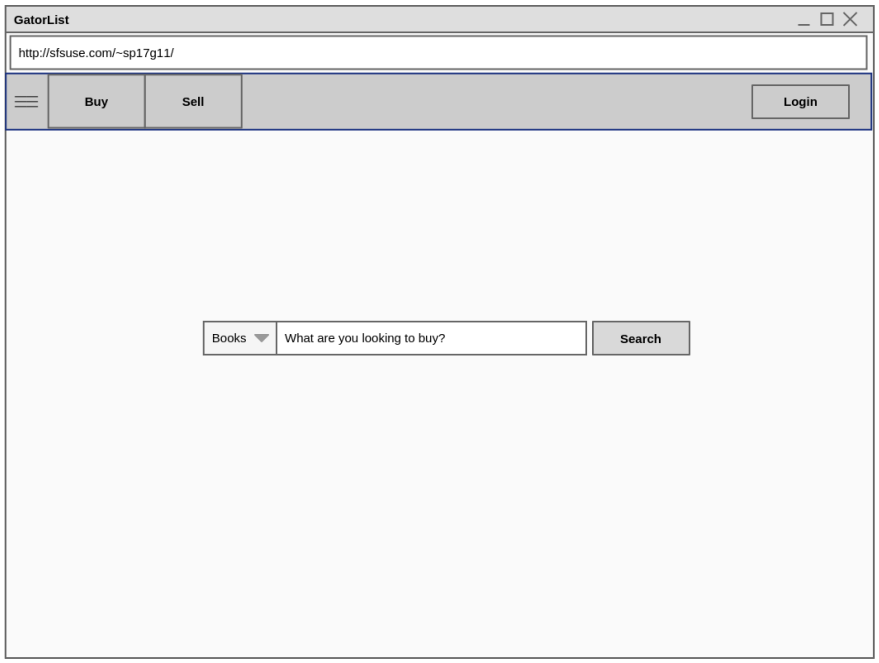
4. Non-functional Specs

4.1 Application shall be developed using class provided LAMP stack.
4.2 Application shall be developed using pre-approved set of SW development and collaborative tools provided in the class. Any other tools or frameworks must be explicitly approved by Anthony Souza on a case by case basis.
4.3 Application shall be hosted and deployed on Amazon Web Services as specified in the class
4.4 Application shall be optimized for standard desktop/laptop browsers, and must render correctly on the two latest versions of all major browsers: Mozilla, Safari, Chrome.
4.5 Application shall have responsive UI code so it can be adequately rendered on mobile devices but no mobile native app is to be developed.
4.6 Data shall be stored in the MySQL database on the class server in the team's account.
4.7 Application shall be served from the team's account.
4.8 No more than 50 concurrent users shall be accessing the application at any time.
4.9 Privacy of users shall be protected and all privacy policies will be appropriately communicated to the users.
4.10 The language used shall be English.
4.11 Application shall be very easy to use and intuitive. No prior training shall be required to use the website.
4.12 Google analytics shall be added.
4.13 Messaging between users shall be done only by class approved methods to avoid issues of security with e-mail services.
4.14 Pay functionality (how to pay for goods and services) shall not be implemented.
4.15 Site security: basic best practices shall be applied (as covered in the class).
4.16 Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development.
4.17 The website shall prominently display the following text on all pages "SFSU Software Engineering Project, Spring 2017. For Demonstration Only". (Important so as to not confuse this with a real application).
4.18 Application shall provide user with adequate information on its functionality.
4.19 Application shall not crash, and if it does, not constantly.
4.20 Application, in the event of a crash, shall be back up within 24 hours.
4.21 Application shall be available during all hours except in the following cases: the website is taken down after the semester, the website has crashed, contract with web-server was expired, web server access right was violated by third parties, or there is a serious problem/bug that needs immediate addressing.
4.22 Application shall load all pages fast enough to satisfy the average user.
4.23 Application shall not confuse the user with unnecessary information.
4.24 Application shall provide users with all information necessary to acquire their product.
4.25 All username and password shall be adequately protected.
4.26 All data shall be adequately stored.
4.27 User account and information shall not be compromised in any way.
4.28 Application shall provide elegant, simplistic design that looks appealing and fully functional.
4.29 All data and access permissions can only be changed by system administrator.
4.30 All data should be backed up and secured at least 1 time(s) a week.
4.31 General coding standards will be maintained; modular, readable, all brackets and semicolons in proper positions.
4.32 Documentation of code will be easy and clear to understand.

5. UI Mockups and Storyboards

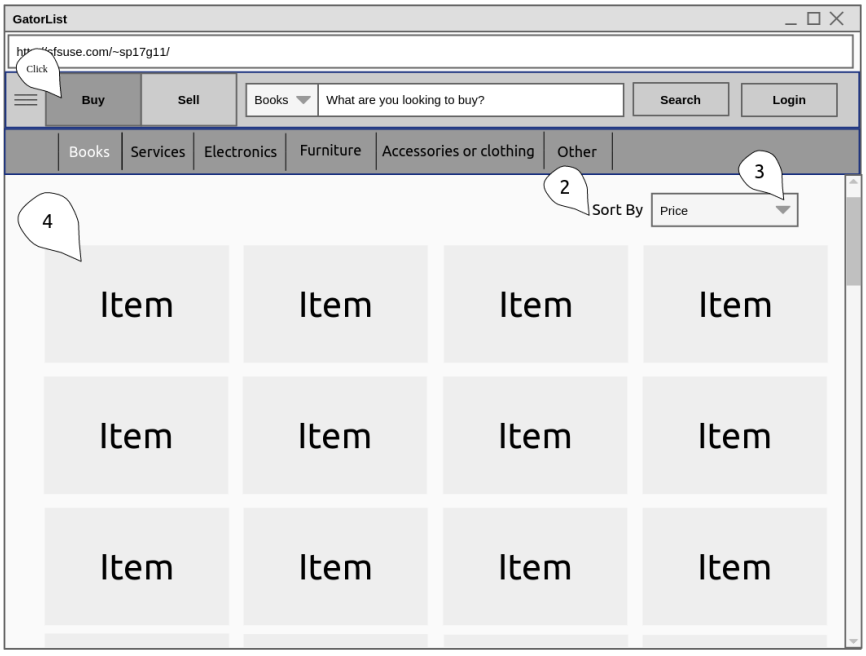
Guest User

Use Case: 2.1 Guest User



Guest User Jeremy is a student at SFSU with very limited technological skills who is browsing the website for the first time looking to buy an Item.

Use Case: 2.1 Guest User



Jeremy starts browsing the available items, he can also choose to specify a category.

To keep his purchase within his price limit Jeremy could sort the items by price or category.

Functional Specs:

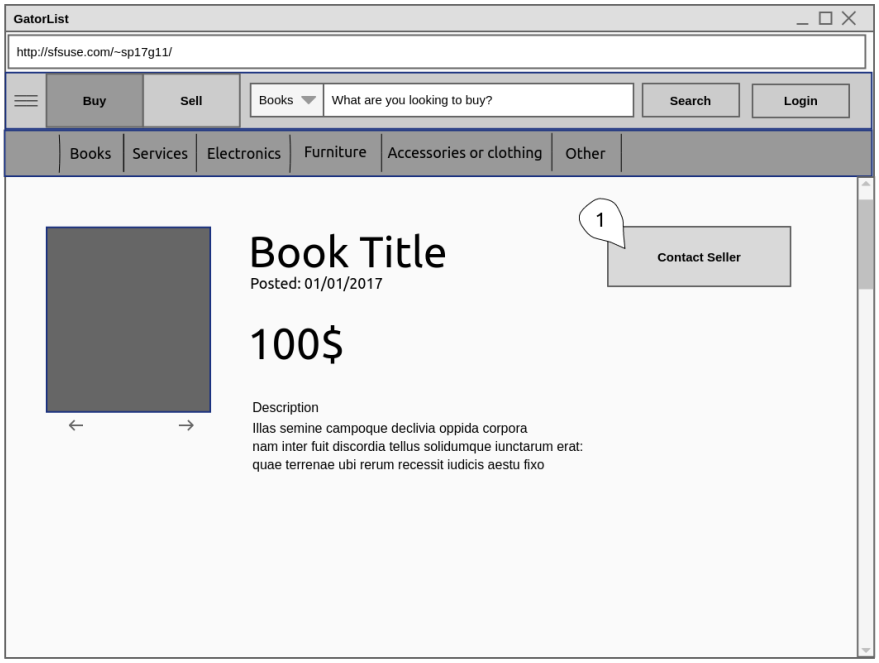
1. Browse items without having to register.

2. Sort items by category.

3. Sort similarly categorized items by price, alphabetically, or by date.

4. Show images of products.

Use Case: 2.1 Guest User



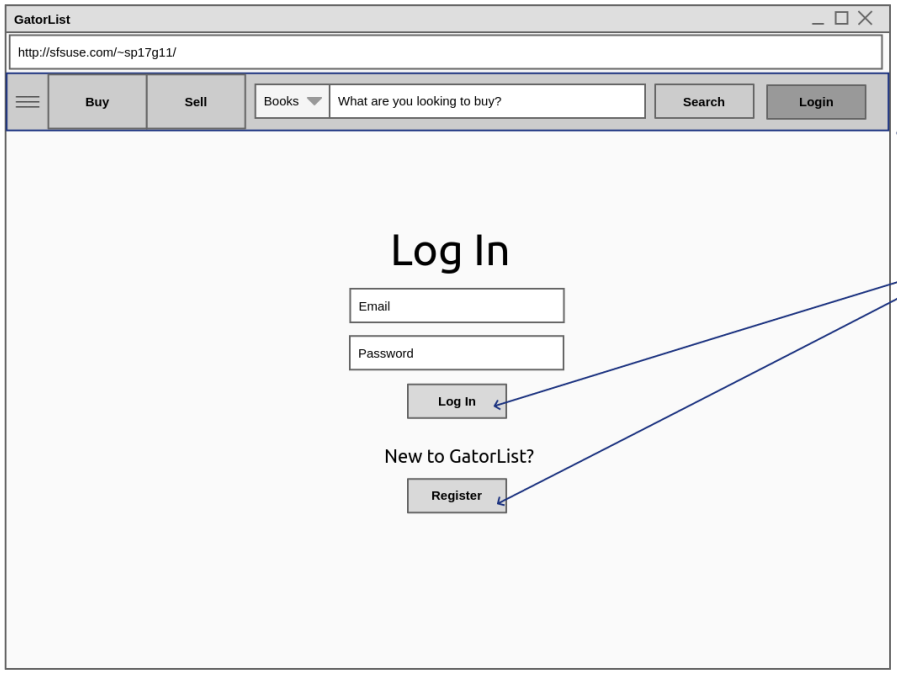
Guest user Jeremy sees a book that he likes, he wants to buy it but at this point he can only browse Items.

When he attempts to proceed with his purchase he must either log in to his account or register a new one.

Functional Specs:

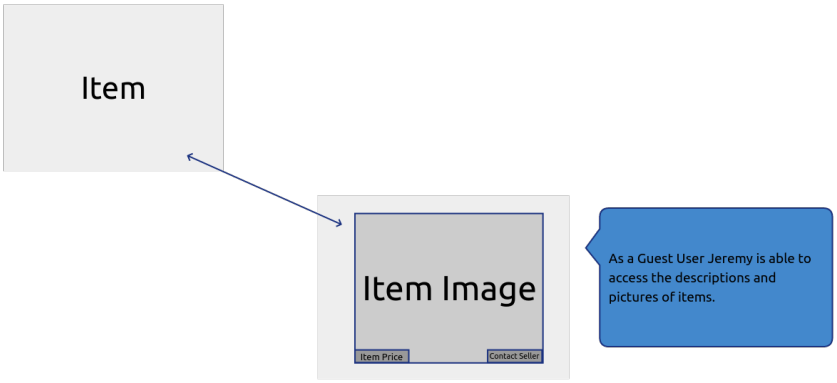
1. Message between sellers and buyers.

Use Case: 2.1 Guest User



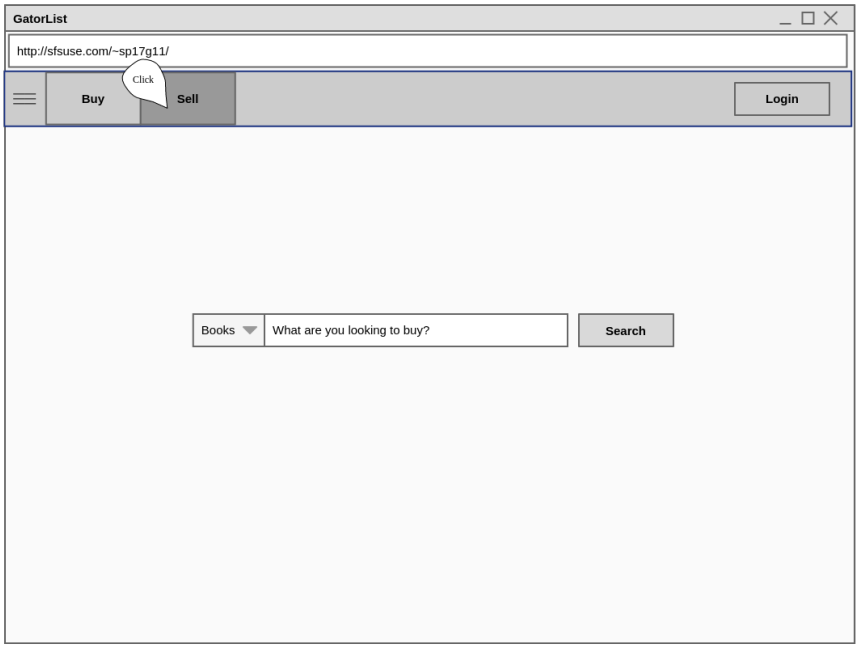
Guest user Jeremy sees a book that he likes, he wants to buy it but at this point he can only browse Items.

When he attempts to proceed with his purchase **he must either log in to his account or register a new one.**


Item
Item Image
Item Price
Contact Seller
As a Guest User Jeremy is able to access the descriptions and pictures of items.

Verified User

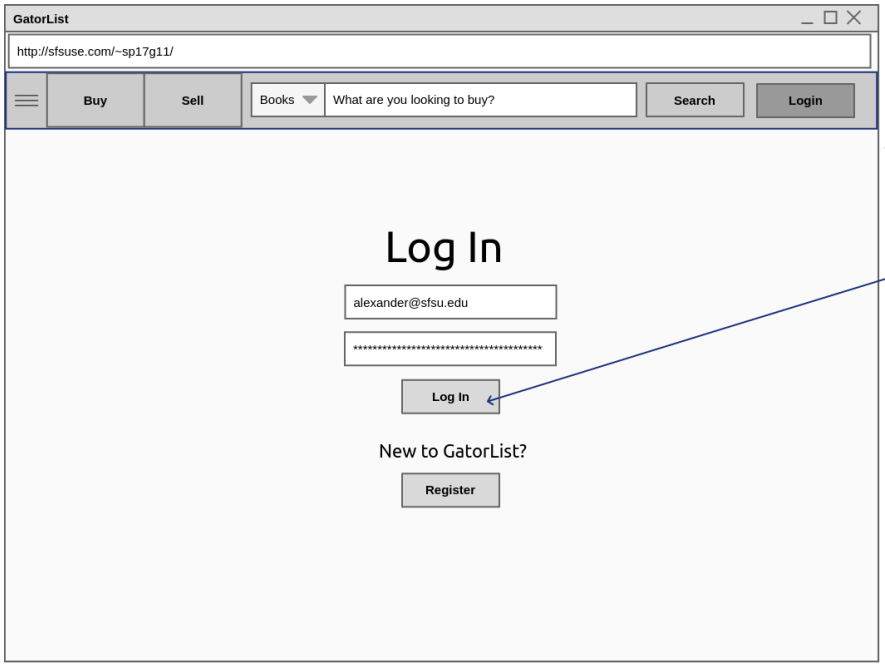
Use Case: 2.3 Verified User



Prospective seller Alexander is a student at SFSU who has previously used the website to make a purchase and now would like to **post his own calculator for sale.**

Upon opening the website, he proceeds to post his calculator for sale, but he is prompted to log in or register for an account.

Use Case: 2.3 Verified User

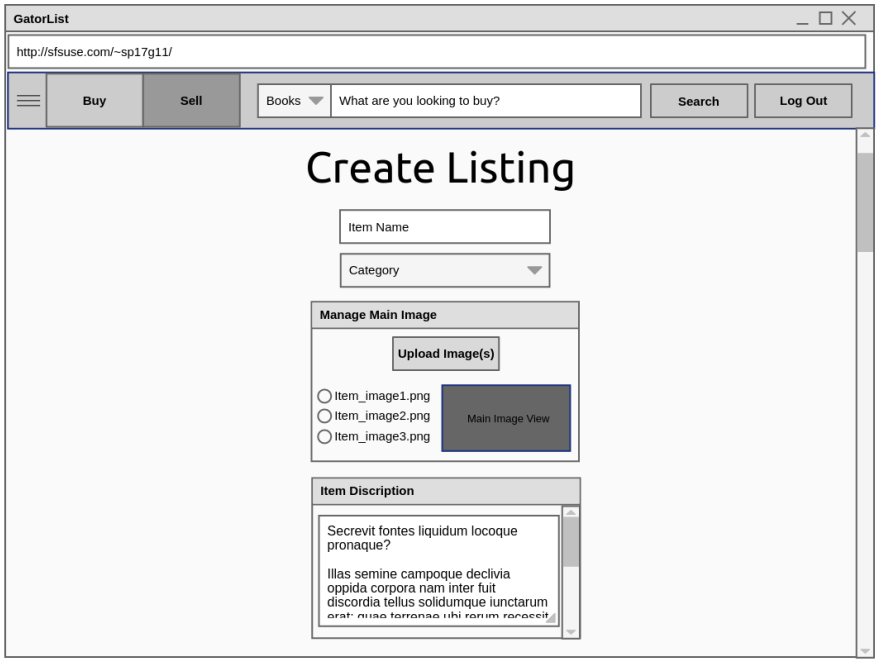


Prospective seller Alexander is a student at SFSU who has previously used the website to make a purchase and now would like to post his own calculator for sale.

Upon opening the website, he proceeds to post his calculator for sale, but he is **prompted to log in or register for an account.**

Since Alexander is a previously Verified User he enters his email and password to be logged in.

Use Case: 2.3 Verified User

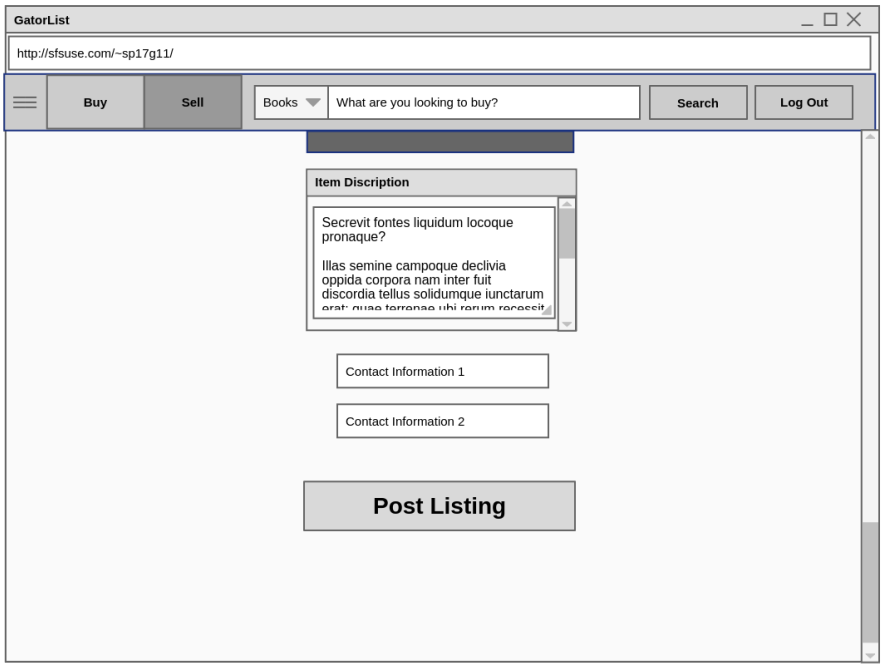


Alexander enters the **name of his Item for sale, selects a category and must upload at least one image of his Item** (He has the option to upload additional images if he wishes to).

Here he is also able to leave contact information for those interested in this Item.

After he is done, his new Item for sale is posted and added the list of available items to purchase.

Use Case: 2.3 Verified User



Alexander enters the **name of his Item for sale, selects a category and must upload at least one image of his Item** (He has the option to upload additional images if he wishes to).

Here he is also able to leave contact information for those interested in this Item.

After he is done, his new Item for sale is posted and added the list of available items to purchase.

6. High-level system architecture

6.1 CakePHP framework.
6.2 PHP framework chosen to supplement the development of the application.
6.3 Git version control using GitHub utilization of a private repository to maintain our ongoing development.
6.4 Amazon web server used for hosting and deployment of code.
6.5 Google Analytics used to track website traffic.
6.6 Markdown used for documentation
6.7 LAMP stack utilizing:

- OS: Ubuntu Server, Version: 16.04
- MySQL Version: 5.7
- PHP Version: 7.0.13
- OpenSSH Version: 7.2
- Git Version: 2.7.4
- Python: 2.7
- Ruby: 2.3.1
- Nodejs: 4.2.6
- NPM: 3.5.2
- Less: 481
- Sass: 3.4.23

6.8 Supports Mozilla Firefox 50+, Google Chrome 55+, Internet Explorer 10+, Microsoft Edge 39.14971+ .
6.9 Application supports the second to most recent web browser versions.
6.10 Image size shall not exceed 4MB.
6.11 Filesystems have been chosen for Database Structure.
6.12 Pictures shall be displayed as 570 px wide.
6.13 Thumbnails shall be displayed as 35x35.

Schema: Team Eleven Table

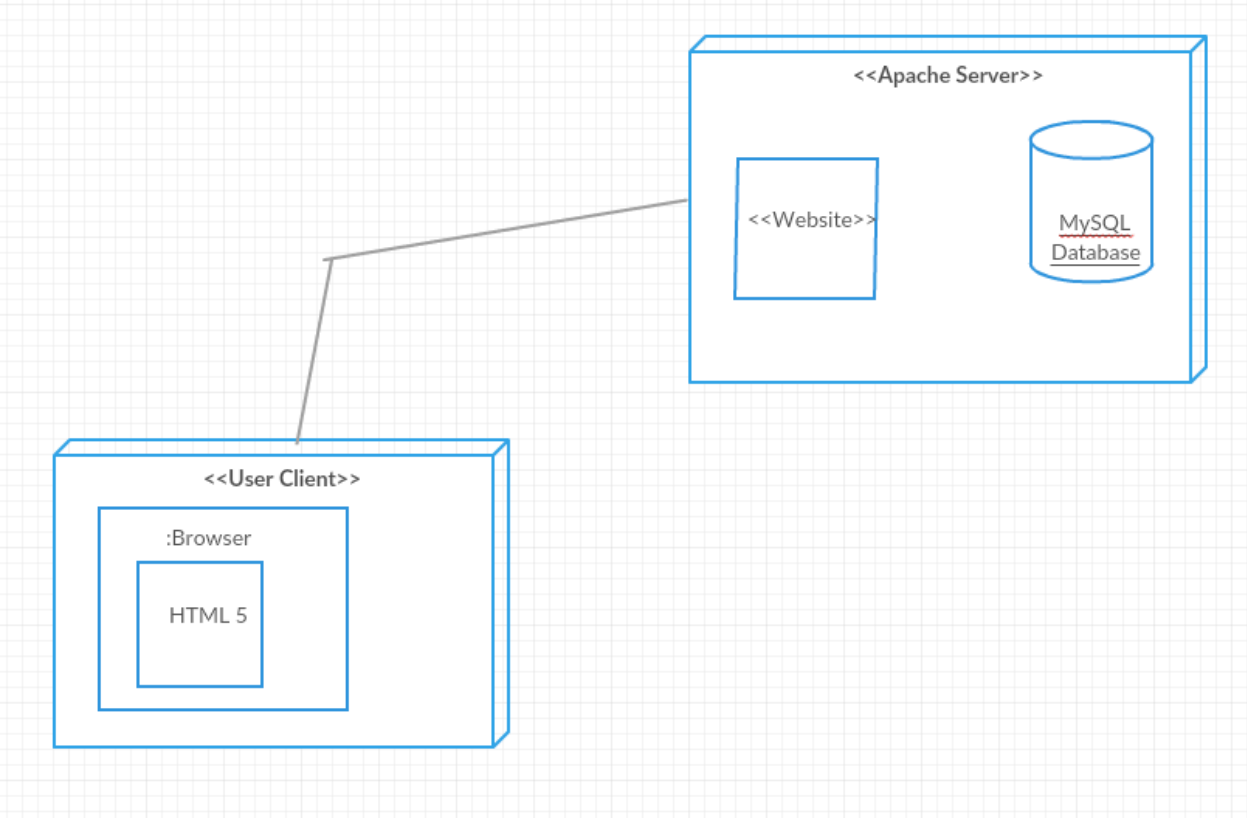
- *Table:* Books
  Items: b.Name
  Examples: The Very Hungry Caterpillar, Where the Weird things are, Goodnight Moon, The Cat in the Hat, Corduroy

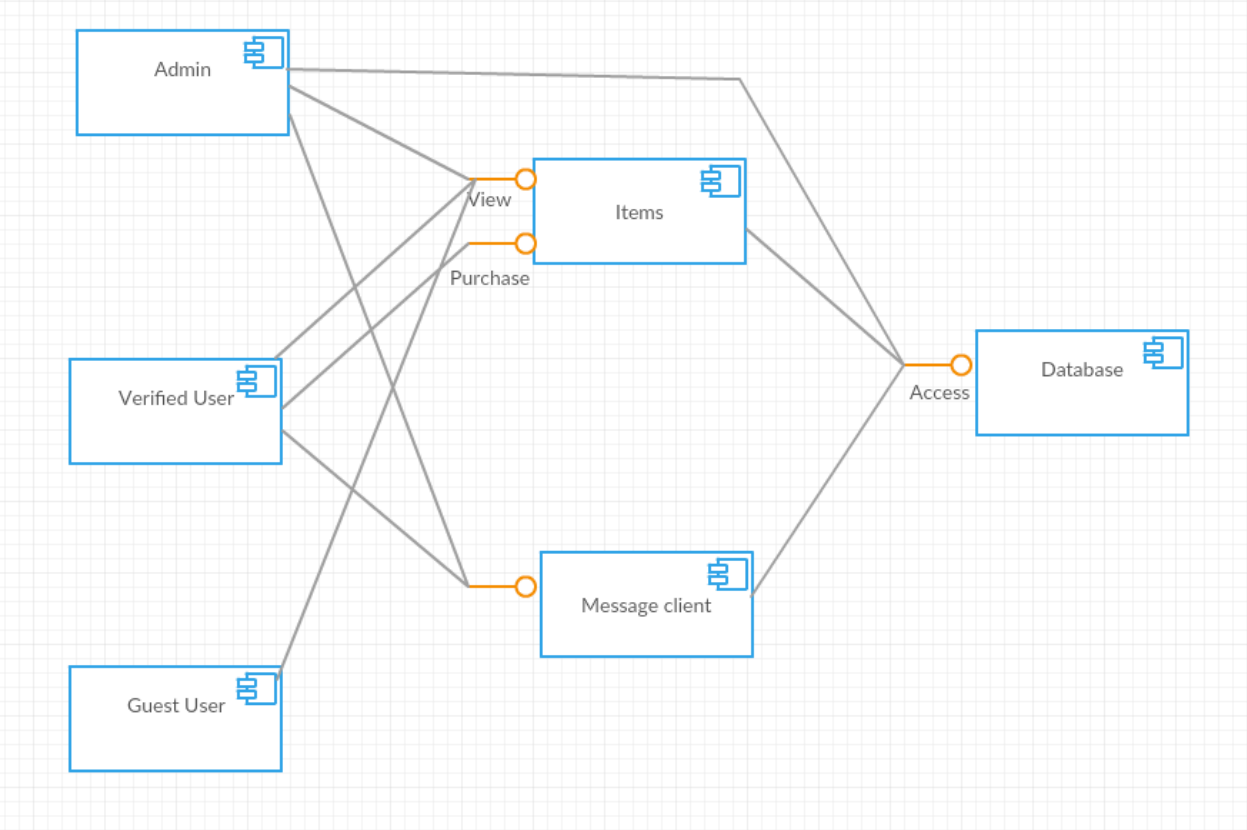
- *Table:* Electronics
  Items: e.Name
  Examples: Rolex, HP Laptop, Alienware

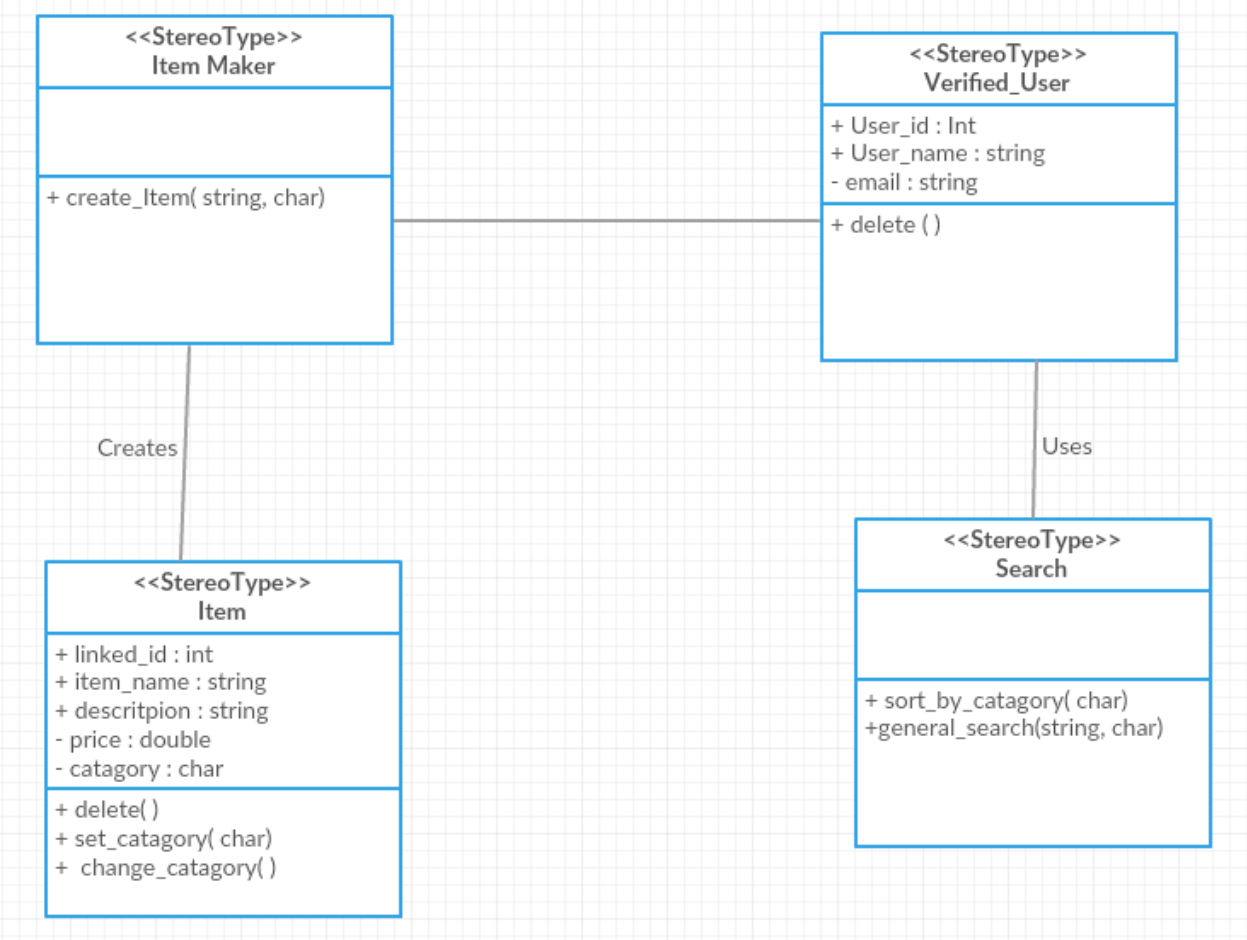
Search:
We will be using SQL Percent *Like* (%_) as our search method.
The *LIKE* operator is used in a *WHERE* clause to search for a specified pattern in a column.
Examples:

WHERE b.name *LIKE* 'T%' Returns any value in Table Books that starts with T.
WHERE e.name *LIKE* '%ol%' Returns any value in Table Electronics that has 'ol' in any position.

7. High Level UML Diagram






<<StereoType>>
Item Maker
+ create_Item(string, char)
<<StereoType>>
Verified_User
+ User_id : Int
+ User_name : string
- email : string
+ delete ( )
Creates
Uses
<<StereoType>>
Item
+ linked_id : int
+ item_name : string
+ descritpion : string
- price : double
- catagory : char
+ delete( )
+ set_catagory(char)
+ change_catagory( )
<<StereoType>>
Search
+ sort_by_catagory(char)
+general_search(string, char)

8. Key Risks

- Skill risks - We try to minimize it by additional meetings on campus and Google Hangout meeting on weekends.

- Schedule risks - We try to meet 1-2 hours before class that all the members are on campus and set Google Hangout meetings on other days.

9. Team Organization

- **Mockups and Storyboards** – Andre Simpelo
- **Database** – Daryl Robbin, Jordan Scandlyn
- **Vertical Prototype** - Hasan Niftiyev, Jordan Scandlyn, Hin Vong, Anthony Searles

ROLES

- Hasan Niftiyev (*Leader, CEO, Universal Role*) .
- Jordan Scandlyn *(CTO – Professional Player*) .
- Daryl Robbin *(Front End, GitHub*) .
- Andre Simpelo (*Performance , Design*) .
- Anthony Searles (*Front End*)
- Hin Vong (*Documentation* , Back End)