**NMU Senior Project Proposal Revised**

Joe Bonino

2/14/2017

Project Intention

My senior project revolves around consumerism and connection. Through my project, I hope to establish a holistic connection between the consumer and the product. Consumers will have the ability to look up a product they are considering purchasing and receive information about the products environmental, ethical, and personal health impact. From the client’s (smartphone/tablet/web), they will be able to search/scan the product to find the above information to help them more fully connect with the product.

For instance, if I wanted to purchase salmon, I would look up the product on this system and it would show me that the product was a 90/100 for my health, 70/100 for humanity, and 20/100 for the environment. Below each rating would be a short review describing why this rating was chosen. Similar to Wikipedia, the review could be edited by any registered user who has sourced their facts. I could read why salmon is healthy for me and that salmon fisherman are paid well but many die on the job. I would also read that salmon are overfished and harvested before they lay their eggs for reproduction. The system will be flexible so that new ratings may appear at any moment based on new information.

Project Details

The system will operate on a web server and be coded in small talk. At the core of the system lies an object I call **consumerItem**. This will contain all the information about the product (name, make, nutritional facts). Each consumerItem will also a list of review objects.

Users can create consumerItems if they don’t exist already by utilizing a **constructor class**. You may **manually** enter an item or scan the **UPC code** to quickly build a consumerItem (web scraping). The problem arises where there can be **duplicate** consumerItems, one entered manually and one created with a UPC code. When creating a new item, I must find like consumerItems and ask user if this is the item they were looking for.

**Review objects** will contain an in depth description of a review. Users will be able to vote on these reviews determined by quality. A review with more upvotes will appear at the beginning of the list. This will require a basic list implementation where a node that is upvoted/downvoted will have to check its two nearest nodes to see if they shall switch places.

**A translator object** is an interface between server and client. All requests flow through this object and it parses requests. If formatting is wrong in request, then client will be notified. If request is a valid, the ip address, mac address, and request are placed in a **Request Object**. The request object will then complete the request (creating/editing/getting consumerItem, editing/creating user, creating/editing review) and return completed request to translator.

**The officer object** is responsible for coordinating all users. The officer object must handle new, edit, and remove requests. It is also responsible for user timeout.

**The user object** will contain all the information of a user. This includes username, history, saved, shopping list, and email. Newly registered users will verify their account by a facebook API or email link. This will require two states, guest and verified.

A **data structure** will exist so that a consumer may use a keyword(s) or UPC code to find a consumerItem.

Instances of all these objects will exist and work harmoniously. Agents will visit data structures when information is needed. A blackboard architecture could be implemented for scaling.

Client

A basic web client will be made using **seaside or AidiaWeb**. The web client will have a **news page** which will require a ‘smart’ up vote system.

**A navigation page** will also exist so users may find an item by clicking by category.

The web client will have a **responsive design** so you may access it on a phone, tablet, and computer.

A page will exist for user registration that will contain the **Facebook/email** verification system.

A user will have a user profile consisting of an **account page**, **history page**, **my reviews page**, and a **“shopping list” page**. A shopping list page will hold local storage of consumerItems.

**Registered users** can write reviews and **guests** may only view them.

This webclient will be optimized by building **offline first**. With this in mind, a “**service worker**” will exist on the client’s machine. All requests from client are sent to service worker. Service worker will first check if request can be completed by local lookup. If it doesn’t exist locally then it will make a request to the server.

An **offline page** is stored locally in the case of losing internet.

An “**app shell**” will be implemented and stored on user’s local storage. All content and pages will fit in this shell. This way only fresh content is being retrieved from server to put into the “app shell”.

The user profile will be stored on client side. When user wants to store user profile on server, he must **encrypt** it first before sending. The server will know nothing about the user profiles history etc. so one may not mine their information for commerce. Since server can’t read user profile, a recovery option is very hard to implement. This is because is user forgets password, then user profile can’t be decrypted. **Server should never be able to decrypt user profile for client.**

One may still view his user profile (shopping list, history, etc) while offline. This will require an **offline mode.**

What I hope to learn

I am hoping to learn a lot more about cross referencing data structures. This may be a harder task than I think and I will find out soon enough. I want to learn how to build a responsive web design to my page and have a very basic interface that’s easy and powerful to use. I hope to learn a design that will let all ages use my system. I am curious to know how general I can make the system and still have it contain appropriate functionality. I also hope to learn lots of environmental effects of items I normally purchase. I want to learn more about black board architecture. This may be the architecture for me and am ready to try and implement it on my system.

Rubric

**All bolded features make up core of system.**

The Server (118)

|  |  |
| --- | --- |
| **ConsumerItem object** (information holder) +1   * Constructor   + UPC, scrap web for info +5   + **Manual entry +3**   + Duplicate item prevention +7   ReviewList object (coordinator for review objects)   * + **Coordinates all review objects +5**   + **Review object** (information holder) +3   + Unlimited amount of ratings option +3   + **Upvote system +3** | 30 |
| Translator object (interface between entire system and client)   * **Socket stuff +5** * **Parsing requests**   + **Notify client of invalid request +2**   + Accepts JSON requests +8   + **Accepts seaside/Aidiaweb** **requests (**My web client) **+5** | 20 |
| ­Request object (service provider)   * **Holds ip address, mac address, and request +1** * **Completes request**   + **Create/edit/remove user +3**   + **Create/edit/remove review +3**   + **Create/edit consumerItem +1** | 8 |
| Officer object (coordinator for user objects)   * **Editing user information+4** * **Default user profile +1** * **Authenticating users +8** * **Data Structure for user objects +2** * Timeout +3 * User Object (information holder)   + **Verified +5**     - Facebook verified +10     - **Email verified +8**   + **Guest +2** | 43 |
| Data structures   * **UPC code +2** * Keyword search +10 * **Personal data structure +5** | 17 |

Basic Web Client (122)

|  |  |
| --- | --- |
| News page +5   * Use reddit API for new page +10 | 15 |
| **Friendly cross devices (responsive design)** | 20 |
| Navigation page (browse items by category) | 5 |
| Search bar (keyword search) | 2 |
| **UPC code lookup page** | **5** |
| User account profile   * **Log in/out page +2.5** * **User account page +5** * History page +2.5 * Shopping list feature +5 * Slim user profile +5 * Encrypting before storing on server   + **Encrypt user profile +5**   + Client can pick type of encryption +5 * Recovery option +10 | 40 |
| Implement Offline cache   * App shell +5 * Service worker +10 * User profile +5 * Offline mode +10 * Local data structure +5 | 35 |

Grading scale

|  |  |
| --- | --- |
| A | 165 |
| B | 135 |
| C | 105 |
| D | 90 |
| E | 60 |

Im at 137