GRAND VALLEY STATE UNIVERSITY

Honors Project Proposal: *Offline Solutions to High Network Latency*

Mitchell Couturier

Advisor: Dr. Hans Dulimarta

Date of Submission: April 1st, 2017

Introduction:

Millions of people use the internet every day. It may be used for communicating, researching information, running a business, providing entertainment, shopping, and countless other purposes. But while the internet has many opportunities, it can be very limiting in cases of slow internet connection. Not only that, but it can also cause serious problems to businesses or any work that heavily relies on efficiency.

Problem Statement:

Depending on a user's internet connection and network latency, webpages or applications using the internet may take multiple minutes to load at a time. In an environment where internet usage is crucial to a user, slow speeds can be detrimental to both efficiency and productivity. This is especially apparent in situations of business, where it is extremely important to complete tasks in a timely manner.

For example, running an eBay business is a popular form of self-employment in the United States. For a regular consumer using eBay, the time it takes to list an item for sale is meaningless. However, when listing massive numbers of items on a daily basis, every second of the process counts toward efficiency. One chunk of wasted time in the listing process is the multiple periods of webpage loads that occur throughout each individual listing. Depending on internet quality, these load times could range anywhere between 10 to 30 seconds per listing. When running an eBay business where the number of listings per day ranges in the hundreds, this amount of time adds up quickly.

Luckily, there are scripts called *service workers* that can run in the background of web browsers, separate from a webpage, that don't need webpage or user interaction. These scripts can be used to support offline experiences, which can greatly increase productivity and decrease time spent waiting for webpage loads. Unfortunately, service workers are only useful if you understand exactly where their services should be used. More specifically, they are useful in specific areas of a webpage where network latency can cause problems and inefficiencies.

Objectives:

I propose to review the main areas on websites, such as eBay, where network latency can cause inefficiency. In this review I will achieve the following two goals:

1. Research areas of webpages and online applications where:
    a. services workers can best be utilized, or
    b. operations can be refactored and optimized for minimal internet usage.
2. Test methods of improvement in a program of my development, and strive to achieve minimal wait time in areas that usually heavily rely on the internet. I will then test for improvements in performance.

Plan of Action:

The first goal of my research is to study the areas where internet speed can cause inefficiencies. This research will be done for websites of multiple uses, to discover the most common areas where improvements can be made. The majority of this research will be attempting to discover the best locations to utilize service works, but additional areas for optimization may be researched as well.

The second goal is to test improvement methods by designing and developing a program to show that the methods reduce time spent loading. I plan to do this by using the eBay example previously described. The program will be an eBay-listing application, allowing for the creation of listings all local to the computer without the need to constantly load data from the internet. This could utilize a background process that would do all of the interfacing with eBay so the main UI would see no delays between listings. I will use the specific areas found in my research to implement the use of service workers and maximize efficiency of creating eBay listings.

The current areas of inefficiency in the eBay listing process that are currently of most concern were timed are listed as follows:

- 0:17.2 - to load eBay and log in
- 0:12.5 - to load the sell page
- 0:42.1 - to go to create listing page and find specific item
- 1:04.3 - to enter title and description
- 0:47.6 - to upload photos
- 0:13.9 - for the listing to submit

The goal is to complete the tasks above while minimizing the amount of time where possible.

Application to Test Methods

Core Feature List:

- **User-friendly application**
  - Easy-to-use user interface with visual appeal
- **eBay Listing Creation**
  - Must be able to create and submit listings, similar to the listing process on eBay.
- **Very minimal Wait Time**
  - Minimal (or no) wait time between different phases of listing process (creation to submission)
- **Account Login**
  - Login specifications at start-up, using eBay API to handshake authenticity of accounts

Additional Feature List:

(Will be added to as more research is done)

- **Configurable listing templates**
  - Stored values in specified fields of listing that are consistent through common listings
- **Check-box description builder**
  - User-specified check-boxes that build the description section of the listing when selected.
- **Configurable layout / ordering**
  - Suggested to have description-builder come up first to make electronic testing easier and more conveniently flow into listing

- **Offline-mode**
  - Allow the user to create many listings without actually sending them, storing them in an internal database (or other storage method), pushing them all to eBay at a user-specified time

Schedule

	5/08	5/15	5/22	5/29	6/5	6/12	6/19	6/26	7/3	7/10	7/17	7/24	7/31
Research													
Core Features													
Additional Features													
Final Report													

Preliminary Bibliography

- eBay Trading API Source Code :
  - "Call Index." Call Index - API Reference - Trading API. N.p., n.d. Web. 30 Mar. 2017. http://developer.ebay.com/devzone/xml/docs/Reference/eBay/index.html
- Service Worker API
  - "Service Worker API." Mozilla Developer Network. N.p., n.d. Web. 31 Mar. 2017. https://developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API

Acknowledgements

The main idea and some decisions behind the project were developed by a close friend of mine, Trevor Ekin. He will also be tracking the progress of this project.

Feasibility Study

Software Decisions

Originally, we were torn between using Java or Javascript to write the eBay application in. While writing the program in Java could provide a much simpler, executable end product, we decided that with Javascript it would be easier to create an adjustable form for creating listings and it would also be simpler to connect with the eBay API. In addition, we could simulate an executable by writing a script to run a local server as well as the javascript application, which would make things much easier for the end user.

Software Languages and Libraries

Chosen Languages, Frameworks and Libraries

Language, Framework or Library	Purpose
Javascript	Used for creating the main form used to create an eBay listing.
eBay Trading API	Used to transfer information between the application and eBay.
Service Worker API	Used to run background scripts for the application and create an effective offline experience.
node.js	Used to run a web server for the application to run on.

Remarks

We will create an executable by creating a bash script to first start the node.js server and then run the javascript program.

Feature Requirements

Account Login

User will need to be able to log into the application so that they can connect their Ebay account and create listings. This will be done by using the eBay API to handshake authenticity of accounts.

eBay Listing Creation

The listing creation can simply be a mimic of the already existing listing process on eBay's website, but with more efficient and time-saving mechanics. This portion will be mostly writing HTML to create an identical form.

Minimal Wait Time

The main goal of this project is to find areas in which to minimize the amount of time spent waiting due to internet loading. Achieving minimal wait time will be done by both finding locations where the Service Worker API can be utilized, as well as any areas where operations can be optimized for offline use.

Tools

Communication

- Email

Source Code Control

- Github

Issue Organization

- Github Issues

Development Environments

- Language : Javascript → IDE : JetBrains WebStorm

Time-tracking

- Toggl