National College of Ireland

BSc in Computing

2014/2015

Sean Keeley

x12383858

x12483858@student.ncirl.ie

PhotoPrompt - Technical Report

Github Link: https://github.com/s3an/photoprompt2

Table of Contents

Executive Summary

This document holds the purpose of portraying the progress of the Final Year Project - PhotoPrompt. Found in the appendix of this document are the previous required documents that have been submitted along with updated information in the main document. The mobile market is constantly growing and smartphones are now standard in peoples lives. Facebook claims to have 350 million photos shared a day on it's platform - not including Instagram uploads. Many of those are reposts, or self pictures and hold no real meaning. The idea behind PhotoPrompt is simple. Log in, check your prompt and go take the best picture you can of that prompt. Other users are the deciding factor of what is good or not. There is no metric or formula for a good photo, just practice. Using PhotoPrompt will provide you with inspiration to take photos that you normally wouldn't take and fill the world up with photos that actually mean something all while becoming a better photographer. I chose a hybrid web application because it allows anyone with a browser to access the site and engage in the functionality. No matter your operating system you will be able to use PhotoPrompt as long as you have access to a web browser.

The prompts themselves will range from a variety of things such as nature, buildings, traffic etc..all things that most people will be able to find. I think it will be interesting to see how so many people interpret the same topic and seeing the difference in photography style will urge people on to trying new methods or solutions to photography problems. With people taking photographs on smartphones everyday, and apps such as Instagram and Snapchat now engrained as parts of peoples daily routine, I think that mobile users with an interest in photography will be very interested in PhotoPrompt.

1 Introduction

1.1 Background

Everyone has their own device. People want to take photographs on the go. It is the reason that smartphones with cameras were invented. Sometimes there may be a lack of inspiration or drive to take a picture and what to take that picture of. PhotoPrompt will solve that problem. Bringing back the excitement to take a photograph of something that is not yourself. Using PhotoPrompt will give the user a topic to base a picture on, along with feedback about how good that photograph is based on user opinion and relevance to the the topic. PhotoPrompt aims to give the user a reason to take a picture again.

1.2 Aims

- Clean 'Material Design' UI
- Easy to use functionality
- Accessibility on all devices
- Use of forward thinking technology
- Minimal network impact on uploading and downloading images

1.3 Technologies

HTML5

The current standard in web markup will be used allowing for the most up-to-date functionality. HTML templating will be the base of the markup allowing for rapid development and modular code.

CSS3

As above I will use the most up-to-date style functionality to deliver a clean and simple UI. CSS3 is now the web-standard in delivering stylesheets.

SCSS

SCSS is a CSS pre-processor built and compiled using Ruby. Using a pre-processor has many benefits over regular CSS. These include a responsive design outlook from the beginning of your project. The ability to use 'Mixins' which act like functions within your CSS that allow you to make calculations within the stylesheets. One of the bigger uses of using the SCSS pre-processor is the ability to save variables within your stylesheet so you don't have to write the same CSS code many times. SCSS files are then compiled using a ruby program to export pure CSS code.

Javascript

Standard stuff really. Javascript for general functionality and interactivity on the backend. The whole of the framework is built and written on the assumption of using Javascript as both a front and back end language. The latest Javascript standards will be adhered to along with using the latest stable release of the language, ECMAScript 6.

AWS

Amazon Web Services or AWS for short, is a cloud based platform for hosting, integrating and storing complex systems on a easy to use, scalable, reliable and well documented platform. I chose AWS for hosting because it is becoming the industry standard along with having some prior experience on it.

The part of AWS I will be using is the EC2 (Elastic Cloud), which creates a virtual machine based on your configuration. In it, it gives full control of what type of systems and programs you want to utilize to make your system functional. I will just use AWS for additional testing instances as the main backend will be contained within MeteorJS. There will be a need for multiple instances of ec2 severs to deploy and test the application.

Another aspect of AWS that will be used will be S3. S3 is an online file storage service that provides storage through web service interfaces such as REST and SOAP. S3 will store pointers to the images that are uploaded through the application, and it will serve as a pseudo content delivery network allowing for faster streaming from the cloud service.

MeteorJS

MeteorJS is an open-source JavaScript application framework built using Node.js. Using Meteor gives me the ability to control front-end and back-end technology, along with the ability for rapid prototyping of cross-platform, hybrid code. It is basically a collection of Node.js libraries and packages bundled together to make web application development easier and more straight-forward.

MeteorJS is split into a CLI (Command Line Interface), a server and a client. The browser uses packages like Tracker and Spacebars to receive and display the dynamic front-end code you write. Your server code is written in javascript using a templating language and provides quick and easy server-side code.

You only need to create one code base to service many platforms and the code is cross-platform compatible. This enables the developers to hotfix problems in the application by simply updating the code-base, removing the need for App-Store updates and validations.

MongoDB

MongoDB is the packaged database within Meteor and is the premier NoSQL database system currently available. MongoDB is written in code similar to Javascript and uses a flexible document model that is similar to JSON. (Javascript Object Notation).

Using the flexibility of MongoDB opens up for Rapid application development and simple evolution of your system because you are not tied down to one set of fields or

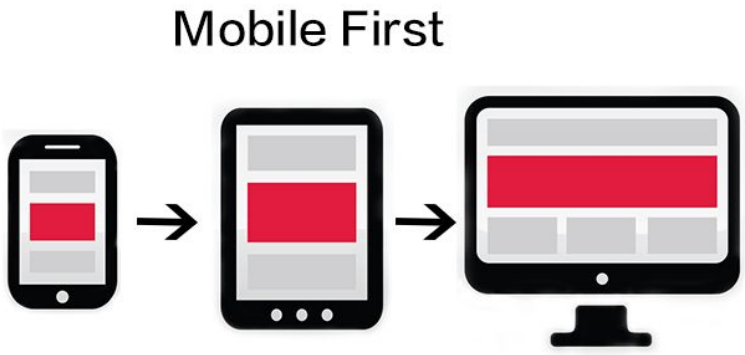
one way of doing things. With built in systems for scaling and database failure, along with being based and written again in javascript style syntax, it fit's with the rest of the project.

BootStrap 3

BootStrap3 (Built by the lovely people @ Twitter) is a HTML, CSS and JS framework for building responsive, mobile first web applications. I will use bootstrap for it's out of the box responsive functionality so that I can begin 'Mobile-first' design. At it's core it is helpful to build quick and easy user-interface components.

Mobile First Design

As an extension of the above explanation of BootStrap, Mobile-first design is as it sounds - designing for the mobile experience first. The previous standard of web design employed Graceful Degradation - which would design for the biggest experience and 'gracefully degrade' down to the small viewpoints, often resulting in lackluster experiences. Mobile-first takes the opposite approach where it starts at the smaller viewpoints and employs Progressive Enhancement - a seamless transition into the larger viewpoints with added benefits occasionally. This is a core responsive design and should give an application like feel the web app.



2 System

2.1 Requirements

2.1.1 Functional requirements

2.1.2 Requirement 1: Uploading a photo

Description & Priority

This is a vital, important requirement as the application is powered by the ability to check a topic and upload a photograph. PhotoPrompt wouldn't work without the ability to upload a photo from your device. The user takes a photo, it is stored to your devices internal memory, the user clicks an 'Upload' button which accesses the file system and allows the user to upload the chosen photograph.

Assumption & Risk

A person may upload a photo that is irrelevant to the current topic. This cannot really be stopped by the application but will hopefully be voted against by the user base. Again, a malicious user may upload an offensive photograph which cannot really be stopped on the application back-end without some intensive monitoring process beyond the means of this project.

Detailed Use Case for Uploading a Photo:

Scope

The scope of this use case is to evaluate the needs of the user and the system in order to upload a photo and interact with the application as a whole. This is the main flow of the whole application.

Description

This use case describes the processes behind uploading a new photograph to the system.

Flow Description

Precondition

The user has logged in successfully and the system is waiting at the homepage. There is a keyword and a timer displaying on the homepage of the system. The user can view already uploaded images and can clearly see the upload functionality to interact with.

Activation

This use case starts when a user interacts the 'Upload Photo' button

Main flow

1. The system identifies the button click or interaction and sends a request to the user device.
2. The user interacts with the pop up screen and chooses a photo to upload. (The user may also drag a file directly onto the system)
3. The system then takes that information and uploads the photo to a database and stores a CDN pointer.
4. The system validates the photo is of correct format and is within the size regulations.
5. The system sends a request to upload the photo
6. The system displays the photo on the users homepage

Alternate flow

A1 - The user uploads a non-photo format

1. The user chooses a non-photo format file to upload - such as .mp3 or .java
2. The system attempts to validate this upload
3. An error message is returned alerting the user of a failure.

4. The use case continues at position 2 of the main flow

Exceptional flow

E1- Invalid photo format.

5. The user chooses to upload a file that is not of photo format. This could me .mp3 or .java et
6. The system cannot upload this file because of pre-determined restraints.
7. The use case continues at position 2 of the main flow

Termination

The system then presents the homepage again, with the users newly uploaded photo being displayed.

Post condition

The system goes into a wait state, for another photo to be uploaded or another functionality to be invoked.

2.1.3 Requirement 2: User Voting

Description & Priority

This is another important requirement. The ability to vote for and against what you think is a relevant photo is important to the success of PhotoPrompt. There will only be a positive vote. Giving the user the option to negatively vote often leads to a larger % of negative voting, purely for the sake of voting against someone.
https://medium.com/the-physics-arxiv-blog/data-mining-reveals-how-the-down-vote-leads-to-a-vicious-circle-of-negative-feedback-aad9d49da238#.ej2979gg3

This is referenced in the above article but it covers points that a

"Users who are down-voted produces lower quality content in future that is valued even less by others on the network. What's more, people are more likely to down-vote others after they have been down voted themselves. The result is a vicious spiral of increasingly negative behaviour that is exactly the opposite of the intended effect."

Assumption & Risk

Risks lie in this requirement about a lack of user interaction. Users are not required to vote so therefore we cannot make them vote. I make the assumption that the user is actually interested in seeing the better content that the application has to offer therefore the user will vote according. As said above, users can only be encouraged to vote through prompts on the site.

2.2 Data requirements

The data requirements of PhotoPrompt was research into the optimization and compression of the photo files in order to save the most server space as possible. There will also be a limit on the size of the photo that can be uploaded by the user in order to stop malicious users uploading gigabytes of data and slowing down the experience for all users. Using the cloud storage as a pointer system over a traditional database will provide a faster connection and less need for users to download the images, rather just redirect the devices screen to the content. Upon initial testing, this method saves data and is not in any way detrimental to a users data plan.

2.3 Portability requirements

PhotoPrompt is a mobile-friendly, responsive web application meaning that the portability requirement is quite high. The experience will be the same and fluid across any device that you may use. This means that the application has a very portable aspect to its creation. With a mobile-first principle in place, the application should deliver a consistent experience across any device regardless of platform.

2.4 Maintainability requirements

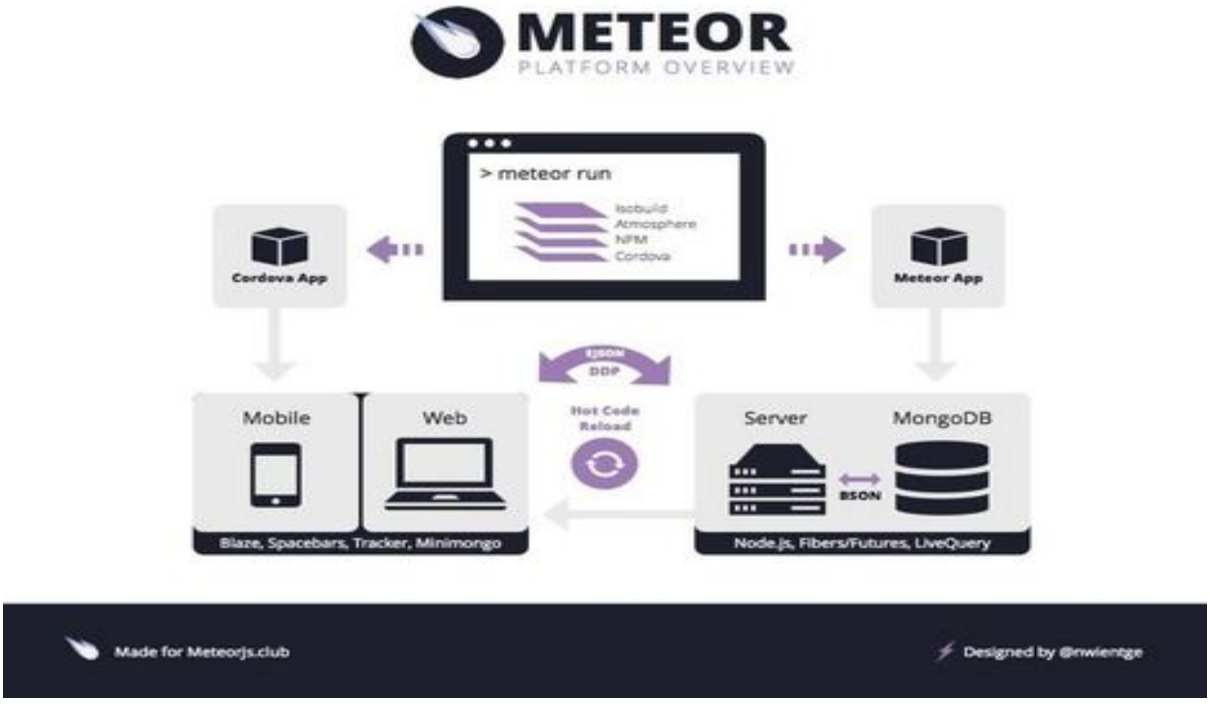
As PhotoPrompt is a web application, there can be a quick turnaround time for new features, bug fixing and testing. I chose the web application method because of this because of the rigid testing and approval processes that go along with native applications and applications stores. Having the ability to make quick changes was really important in developing this project as I wanted to be able to develop in an agile work methodology. MeteorJS being cross platform allows for the hot-fixes to skip the approval process and for new features to be developed quickly and efficiently.

2.5 Usability requirements

The user will need a constant internet connection, as well as either 1) a device with a camera or 2) a device with accessible and readable internal storage that has the desired photo stored on there. The internet connection will be for uploading the photos from the device as well as loading the content of the web application. Baring the physical requirements of the devices the user will need access to input the text of the usernames and passwords. There are no other usability requirements upon the application.

3 Design and Architecture

The architecture of PhotoPrompt will be based on the simple Client/Server stack that is contained within MeteorJs. The Picture below will explain further.



Barring the architecture, there will be three main algorithms in this application. First of all the algorithm for returning a random topic for the user to take a picture of. For testing purposes I will have a sample database to pull random topics from. I eventually want this to search the web for things that are trending and being talked about, collate them together and then return them as topics for people to use.

The second algorithm will be for uploading images. This will take the users image either taken from the camera on the device, or from within the file system and upload it to an Amazon S3 bucket where it waits to be returned to the client application. This upload algorithm will take advantage of many Meteor Packages such as Collections:FS and CFS:S3. These packages leverage Meteors built in methods for easy manipulation of files and uploading to S3 by just inserting the information that Amazon provides. I'll need to make sure this information is hidden correctly so no malicious users can use the Amazon account.

The third algorithm is the voting one. The application will just return the most recent images uploaded by users. There will be an ability to have a positive vote, showing the uploader your enjoyment of the photograph. There will be no negative voting. As referenced above, negative voting leads to a cycle of negative impact on the user and on the site as a whole. A positive vote may inspire people to use the application more and in turn create more and better content. A user who has been negatively voted may feel victimised and that they shouldn't continue to use the application for fear of further negative voting.

4 Implementation

Using Meteor JS as the main implementation framework will have its benefits and it's challenges. Using things like Blaze to make reactive templates will speed up development substantially. Blaze is Meteor's built-in reactive rendering library. Usually, templates are written in Spacebars, a variant of Handlebars designed to take advantage of Tracker, Meteor's reactivity system. These templates are compiled into JavaScript UI components that are rendered by the Blaze library. Handlebars is just a templating syntax language.

The main classes within the application revolve around uploading images to the cloud service, logging in and registering and voting on the uploaded images.

Uploading images code will combine the use of multiple packages that come with Meteor JS. Namely it will use the Dropzone package, giving the users an area to upload to on the front-end. There is another package called Collections-FS that will allow manipulation of the images to be put into temporary mongo collections before using an S3 extension of Collections-FS to push those images to Amazon S3. Below is a code snippet for uploading:

```
if (Meteor.isServer){
    var imageStore = new FS.Store.S3("images",{
    /* REQUIRED */
    accessKeyId: Meteor.settings.private.AWSAccessKeyId,
    secretAccessKey: Meteor.settings.private.AWSSecretAccessKey,
    bucket: Meteor.settings.private.AWSBucket
    });

    Images = new FS.Collection("Images", {
        stores: [imageStore],
        filter: {
            allow: {
                contentTypes: ['image/*']
            }
        }
    });
}
```

For user accounts , the built in Accounts packages in Meteor will be used allowing for quick accounts integration. Firstly to use the accounts-ui package initially, along with accounts-password for user account creation. This combined with accounts-bootstrap for better design that is more in line with the rest of the application. MeteorJS has built in oAuth functionality that allows extensions to various other sites to use those as login credentials. The oAuth packages that are included in Meteor include but are not limited to Facebook, Twitter, Google, Github and many more. This will allow users to enter the application from whatever chose medium they want - if for some reason they do not want to make an account using the account creation functionality.

The voting system will make use of a reactive-var package. A ReactiveVar holds a single value that can be get and set, such that calling set will invalidate any Computations that called get, according to the usual contract for reactive data sources. It's similar to a session variable in traditional web development with the added benefit of holding any data type, not just JSON like a session would hold. An important property of ReactiveVars — which is sometimes a reason for using one — is that setting the value to the same value as before has no effect; it does not trigger any invalidations. So if one autorun sets a ReactiveVar, and another autorun gets the ReactiveVar, a re-run of the first autorun won't necessarily trigger the second. This allows for consistent returns of voting numbers across all users.

5 Testing

Selenium is the premium web test suite. It works based on the theory of browser automation. You create classes based on what you want to test so if you wanted to test the upload function - You would navigate to the url, click the button and tell Selenium the desired outcome and if the outcome is the correct one the test will pass.

Meteor JS has many built in testing functionality packages to enable faster integration of testing scripts. The most popular and widely used framework for testing Meteor is Velocity - https://github.com/meteor-velocity/velocity - which is a reactive test runner for Meteor. It in itself is not the actual tests, rather a means to run those tests. Velocity

needs to be paired with one of the forerunning Javascript testing frameworks, such as Cucumber or Jasmine. Once the relevant packages are installed, Velocity provides you with an interface to interact with on the browser, in order to run the tests by hand. This process will eventually be automated.

The use of these frameworks for feature testing will be combined with some webDriver style testing in order to check the live status of the site. This will probably be completed using an EC-2 instance, with Jenkins, running Selenium Web Driver classes.

Below is the Jasmine code for a sample test of testing a user uploading a photograph. You can see in the code snippet that it is straightforward. Tell the test what to do and what to expect in the case of a successful test.

```
1
2  describe('Uploading a photo', function () {
3
4    //This is to clear the slate before each test
5    beforeEach(function(){});
6
7      it('interacting with the button uploads a photograph', function () {
8        //click the image upload button
9        $('button#image-upload').click();
10       // gather the newly created divs
11       $newDivs = $('div.image-container');
12       // assert the number of new divs
13       expect($newDivs.length).toEqual(1);
14     });
15 });
```

6 Conclusion

There are many advantages and disadvantages to the direction I chose to take PhotoPrompt thus far. The advantages mainly lie in the speed department where I have a development stack that allows me to quickly prototype functionality with live updates to the code. MeteorJS has been a life-saver in this regard. It has sped up development

many times over. The ability for countless packages to be installed easily containing many things that are needed has allowed quick development and progression. The development of the front-end, back-end and database functionality within the one language is highly appealing and surprisingly powerful.

I felt I might be hindered by the use of Meteor as an independent framework but the advantages have far outweighed the disadvantages. The speed of development really helped in the high pressure situation that is the software project. The disadvantages lie in the innate theory of having a hybrid application. I have to code for every little nuance within each operating system. For example, to upload a picture from a laptop, and Android Device and an iOS Device are three separate methods because they all have independent file structures and conventions independent to the operating system.

7 Further development or research

I feel like this is an incredibly marketable idea for an application. People taking photos on the go using smartphones is happening more and more and this application will only grow and blossom as that does. With further development I would like to implement geo-location tags for the uploaded photographs. I feel this is beyond the minimum viable product. There should also be a consideration to move the hybrid web application away from it's current architecture and move to 3 separate instances of PhotoPrompt -

1) A fully fledged, mobile web application

2) A native Android application using Java within Android Studio

3) A native iOS application using Swift in xCode.

I want native applications for the inherit optimisation benefits they provide, but currently I am without the resources to make native applications - especially without the time needed to learn a new Language in Swift.

If the application were to reach a commercial status this would have to be weighed up against where Meteor is in its current development cycle. It is an ongoing framework

and the needs might outweigh its capabilities if it were to become highly, commercially successful. Mobile-first is the future and it would remain the future of this project but research would be done around upcoming innovations in the mobile technology space in order to take better advantage of the trends that appear and are upcoming. The performance difference isn't really noticable within the scope of this project but if it were to reach large commercial potential, it would have to be considered.

8 Appendix

8.1 Project Proposal

Background

I wanted to make an application that would show off my skills that I learned during my work placement. I worked in a web development company where I was developing mobile first websites and applications using a myriad of front end languages and principles. I wanted to take this mobile focused, front end driven experience into my final year project as this is the aspect of programming and computer science I enjoy the most. I am a design geek at heart and I love making something look pixel perfect and flow well. User experience is a big deal to me and I want to bring that into my project as a future career possibility.

I decided on a mobile application that somehow incorporates pictures as I enjoy taking pictures on the go with my smartphone. My main idea would be when you log in to the app, the app presents you with a keyword and this keyword is the basis of the next photo you take. You have to find something near you that is related to that keyword and capture that image. Once you have that image you can upload it to the app and it will be categorized with the other entries of that same keyword. People can then vote on the picture they like the most (or is best representative of that keyword). The winning photo(s) will be featured on the main screen of the app. The keyword will change on a timer meaning that during the day there will be many different keywords and many different subjects of which to take a picture of. I like the idea of something prompting you to go out and take a picture providing you a myriad of different areas and different perspectives. I think this is an interesting approach to a photography app and it will allow me to express my strengths in a way that is forward thinking and future driven.

Objectives

I will break the objectives into 3 main objectives for the application.

The first objective is to create a fully responsive, fluid web application that runs and works well on any device or screen size. The industry is headed towards this kind of development with a lot of companies beginning to develop in a 'Mobile First' mindset. Developing a mobile first application requires you to first finish the mobile design and then scale back up to the desktop design, assuming most of your users will be coming from a mobile device. I chose a web application because it allows me to harness the skills I learned during my industry placement and create a quality, finished product.

The second objective is to set up my own hosting platform for the application. This is something I have little experience with and therefore want to challenge myself in creating a fast, reliable and secure cloud based hosting platform. I will use Amazon Web Services, using the suite of services and applications on there. There are many hosting services out there but the industry standards are leaning towards using Amazons impressive proprietary service for all things hosting, with many companies replacing physical server farms with intricate AWS deployments. I want to gain more experience in using this technology so the project is a perfect place to start.

My third objective is to create an engaging, seamless experience that makes the user want to return and keep using the app. This is in part achieved by having a fluid design allowing you to use the app on any device you have. The other part is having a clean looking design, that is easy to navigate and understand, along with a rewarding experience in using it.

Technical Approach

There are some aspects of this project that will require some research on my part. I do not have extensive experience in the hosting element of an application. I will need to research the ins and outs of using Amazon Web Services for hosting.

My target audience are users that like to take pictures on the go and who are always connected. The app will require a constant connection to the internet.

This application will most likely been based in HTML5/PHP/JS. A mainly web based front end with a cloud hosted back end. I want to use the cloud as a hosting platform because it is the way of the future and AWS(Amazon hosting) is extremely popular right now.

I picked a HTML 5 webapp because of the skills I gained during my work placement. I wanted to apply the front end engineering experience I gained during the placement into the project and I feel using HTML5 and making a web application is the best way to do that.

I will approach this in a Scrum development practice. This requires me to break work into week long increments and achieve something start to finish in that week period. The week can be increased if something is more difficult or will require more man power. When I say something is done from start to finish it means a basic implementation and a start at the testing. I want to test features as I go along so that hopefully I can have an extensive test suite by the time the project is completed.

Special resources required

As this is a mobile web application it will need to be tested on as many devices as possible in order to make sure that the styling and design is consistent across the application.

At a minimum you need a desktop view, an iPad view, a generic tablet view and a mobile view. The mobile view could be broken down into the larger phone screens and

smaller ones. Having all of these multiple devices will be useful for testing and designing. Websites react differently on different devices and different software versions and having some devices that differ will help in capturing bugs and problems.

Project Plan

Gantt Chart in Microsoft Project

Technical Details

Implementation Languages include;

Front End -

- HTML5 for markup and some functionality
- Javascript for interactivity and functionality
- CSS3/SCSS for styling and design
- Grunt for Task running

Back End -

- PHP5 for database connectivity
- PHP for functionality
- Simple MySQL database for data I/O
- Java/Jenkins/Selenium for testing

Evaluation

I plan to simulate a user base manually by uploading pictures and upvoting/downvoting them myself to show that functionality. There will be an automated test suite that runs tests to make sure everything is running correctly. I want to set up a jenkins to do this and run some extra selenium jobs to validate the front ends and test the functions.

The app won't have that much backend infrastructure to keep reliable but I will automatically test anything as it does come up. Automated testing is turning into an extremely important part of software development as manual testing and QA is becoming more and more rare. Companies are hiring automation engineers over manual QA engineers so I want to improve my skillset in these areas and this project will be a great chance to do just that.

__Sean Keeley 28/09/2015____

Signature of student and date

8.2 Project Plan

Gantt Chart embedded here - Gantt Chart in Microsoft Project

8.3 Monthly Journals

Sean Keeley Journal - x12483858

This is my collective reflection journal for the software project module my final year. Here I will type the musings that went on in my head and the thoughts I've reached to complete my final year project. This is an evolving document. Each month I will upload the changes to the document made for that month along with the previous entries. The first upload will contain the journal for September 2015.

September 2015

Week 1:

Starting back into the routine of college is something I never get used to. This year is even worse coming back from 8 months of industry placement. The routine I got into there is shattered now and I have to readjust. There is not a lot of time for that readjustment period this year. Normally there is a 'grace' week where lectures are introductions and tutorials are empty due to no lectures but not final year! It was straight in, with a whole day based on the project module. After the guidelines were given I began to break down the few project ideas I had and try and figure out one that I felt I could best deliver. I need to ask some clarification questions around some of my ideas and the platforms I want to use,

Week 2:

This week I started off productive and narrowed my ideas down to a smaller list. It is genuinely frustrating when you have an idea and you feel this lightbulb go off in your head only to be shot down with a quick google search, confirming your brilliant Silicon Valley-Esque startup has been going for years. I want to bring the skills I learned during my work placement into the project. I feel this would be beneficial for me and beneficial for the project as a whole. I emailed and met with my work placement supervisor, Lisa Murphy, about some of the guidelines needed. I recall her saying she was from a design

background and this is where I see myself headed in the future. I wanted to do something that would look good on a portfolio for a company I wanted to work for. I met with her and we spoke about how design could be included in a project and it gave me some insight on what I have to do.

Week 3:

This week I had some questions about the platforms we could use. I took the Networking and Mobile stream meaning the was more geared towards a native application rather than a web application. I could make a native iOS app or an android app but I don't think I could get the application to be as polished as I could make a web application because of all the web technology work I did during my work placement. I have a more focused idea on what I want to do as a project now too. I want to incorporate photos somehow into the project. Maybe some kind of photo taking application. I love Instagram and I want to create a UI that is similar and easy to use. User Experience is a huge part of my development style. I have an affinity for making things seamless and easy to use. I need to make a decision this week about the actual content of the project as the proposal and supervisor allocation is next week.

Week 4:

This week is pretty much dedicated to creating a project proposal. I need to do a Gantt chart which means I have to use the college computers (:( ) The Gantt chart should help me stay on track this week. I spent the weekend drawing up design documents and plan to start setting up the back end of this project next week. I've finished off the project proposal with an idea I'm happy enough to start on!

October 2015

Week 1:

This week I am swaying back and forth on the idea of making an actual native application. I don't know if I possess the skills to perform on that stage where as I know I could create a polished application if it were a web based app. I will email Sam Cogan about this problem. He might have some more impact on what is really expected of me. I hope that using a web application will be enough because this will allow me to focus on my strengths and build a great product.

Week 2:

This week I came to the conclusion that I can indeed use a web application framework for the project. This is fantastic news. This allows me to use cloud platform hosting, quick turnaround times and the ability to focus somewhat on my passion of front end work. I started working on the requirements specification document which is a large enough piece of work. I drew the mock ups and class diagrams on paper and need to transfer them to some form of online program.

Week 3:

This week was reading week, so I spent most of my time finishing off the requirements specification documents. I had a bunch of other assignments to do to so it was a hard week of week. I got good progress on the RS document though, so that is good news. There is a lot in this document but I used my old RS from the software engineering courses during second year as a basis for what has to be done. I'd like to get an upload function done next week. I've done one before so it shouldn't be that big of a deal to replicate it.

Week 4:

This week I finished off the RS document and my change management documents. I also started to write some of the functionality of the application, mainly the image uploading to a database PHP functions. I don't want to delve too deep into the actual development of the application until there is a clear direction from the documentation because that is what this college teaches you to do! Can’t break the rules kids.

November 2015

Week 1:

I continued to finalise what I wanted to achieve with the project and mainly how I wanted to go about getting to that goal. I've ditched the original idea for a LAMP stack in favour of trying to learn Meteor JS, a rapid application development framework that runs completely in Javascript. Perfect, I can avoid all of the stuff I don't like by sticking to the Javascript plus LAMP stacks are boring. I haven't completely decided if this is a good idea or not yet but I do want to try and challenge myself. I'd like to narrow down the framework choices by next week. I pretty much only know Angular JS. I'll need to investigate further.

Week 2:

This week I read a few journals and books around NoSQL and specifically MongoDB. MongoDB is the base DB used in MeteorJS but based on the plans I have for the project, I doubt I'll be using it much bar pushing the information to some cloud based database, probably in Amazon Web Services. Meteor has a lot of functionality built in that you can just install in your application, like user accounts and packages to help with file uploads. I've found a book on Meteor that I'll read to confirm it's usefulness in the project. I don't have much time next week with the upcoming assignments I want to finish but I'll try do more research.

Week 3:

This was a busy week finishing up assignments and assessments for the semester. Not a lot of project work done. I'm honestly beginning to wonder how they expect anyone to deliver a decent project and good assignments. Where is the assigned time?! Next week I want to start planning the mid-point deliverables. There seem to be a lot of deliverables at the mid-point and I don't want them to creep up on me.

Week 4:

Began planning for the prototype for the upcoming midpoint as I'm planning on not touching the project for most of the winter break. I want to focus on exams as much as possible. Last week was a busy week with assignments and such but now my focus is on project for a few weeks to prepare some mid-point deliverables in lieu of the winter exams. The documentation seems sizeable but manageable and now I at least have a better understanding of the direction I want to go, technology wise!

December 2015

Week 1:

Finalised what I needed to deliver for a midpoint presentation and began work on the document. I really dislike documentation. I've modified my initial plan and proposals to better match the current state of my project. I also think I've now fully committed to using Meteor JS. I love it, it makes making an application fun. It will also allow me to easily make this into a hybrid application by using PhoneGap to compile it into native mobile device software without any extra coding.

Week 2:

More work on the document. There is not a lot of time between the exams finishing and the mid-point presentation stuff happening. I'll need to get some concise work done and code during study breaks. I spent half of this week studying for the upcoming exams. Getting the hang of this Meteor JS thing. In a nutshell it's based on Node.JS and is used for rapid prototyping and cross-platform application development. It covers a lot of the requirements I have for the project so I'm going to continue to make little test applications in order to garner a knowledge of the functionality.

Week 3: Study (Christmas)

Started studying properly this week. It's also Christmas this week so I'll take some time off that. I'm constantly planning in my head how I'll manage study and assignments next semester, I don't want it all to pile up on me again. (I swear I've said that every year for the last 4 years now..)

Week 4: Study (Christmas)

Nothing exciting this week. Exams start next week so the majority of my attention is on that.

January 2016

Week 1:

Exams (Study) The exams start this week. Not only do I not have much time for any project work, I don't have much motivation due to the workload of preparation for these exams.

Week 2:

Exams (Study) The exams start continue and finish week. Not only do I not have much time for any project work, I don't have much motivation due to the workload of preparation for these exams. It's kind of strange to look back on this time last year and being nervous to go into my internship!

Week 3:

The exams went well. Well, they felt like they went well. I'll just have to wait and see for the results to come out. I went straight from exams and back into the midpoint deliverable work. There is a rather sizeable document to produce as well as the hope for some kind of working prototype. The document is coming along well. Meteor JS is handy for just starting a fresh project and having something substantial quite quickly due to the use of packages and built in features. The documentation is where I feel like I'll be struggling for the mid-point. I also am not the biggest fan of doing presentations either, although people seem to tell me I'm quite good at them. Being good at something doesn't translate to me liking to do something. If that was the case I'd be doing some back-end Java project where I tracked data or something snooze-worthy.

Week 4:

I spent most of this week working on the prototype itself and is actually the first time I've really delved deep into the code of the project! With all the exams and assignments and documentation - it's the first time I've had free to focus on it! I started a new Meteor project because I was learning through the old one and it had become a little bit cluttered and uneven. I can easily rebuild what I have from just copying the templates I wrote. One of the reasons I like Meteor, I can just slot stuff together fairly handily. The midpoint is next week and the document is coming together nicely. I'm worried about some of the functionality though. I want to push user image uploads to a cloud based DB for quick streaming but I can't get it to work consistently. It'll be a tough grind to get it out but I have a fair idea of how it could be done.

February 2016

Week 1:

MidPoint - The biggest downfall of Meteor JS is that in it's infancy, there isn't really a lot of people out there who can answer your question if you run into a problem. There are a few packages for what I want to do but they can't really be leveraged without proper documentation or examples on how to use them. I've prepared a backup plan for the mid-point anyway - an interactive mockup that explains the premise of what I want to achieve. The word prototype is subjective anyway.

Week 2:

½ Midpoint This week I had my actually midpoint presentation. I felt it went fine. It is always hard to judge how those things go especially at this early stage of the project. I spoke well and clearly and stayed on topic. I think I did a good presentation. I was happy enough with the standard of the document too. The prototype may let me down because I couldn't figure out that problem with the cloud database. I mean, I could use the local MongoDB database but that's kind of useless for what I need to achieve. I'll have to just properly break it down and figure it out whenever I get some time. I want to step away from the project now for a while, in order to get a fresh perspective on it. This will be helpful with the incoming presentation marks too.

Week 3:

Spent some more time getting into that cloud error. It seems to crash my meteor each time I run it. Exit Code 8. I'm going to get exit code 8 tattooed on me by the end of this endeavour. The problem with that exit code is that it seems to be based within some of the packages that come installed with meteor that you never touch, so everytime I google the error it's a different set of error messages that all relate to exit code 8. Also having some trouble with database locking issues. The use of MongoDB in depth is challenging but I'm enjoying it. I've never really delved too far into it in a practical sense

as I only became familiar with it through a report I did during a database module. I'm becoming an expert at UNIX commands at this stage. Especially killing ports.

Week 4:

Not much done this week, still some troubleshooting in order to try and get rid of that error code. Just settling back into the final set of lectures of the degree. This semester is a short one so there will be a tonne of assignments and assessments coming up soon! If only we had a module on 'Exit Code 8'. Regardless I will continue to think about the problem along with alternate ways to solve it.

March 2016

Week 1:

Lsof -i :3000. Find the node PID. kill -9 Node PID. This is my life now. I spent half a day trying to write a shell script to do this process for me. It works sometimes. It's actually just quicker once you factor in all the times it gets it wrong to write the commands rather than running the script. I just wanted to be fancy. Not much done in the sense of exit code 8 from above so I just started building the application around the error, leaving out the user uploads to a cloud database feature. Not like it's the most important one or anything.

Week 2:

I'm finishing up some of the other features. I have a front end built and some user account done. Some of the smaller functionality is all done, I just need to figure out the uploading consistently part. I've also started researching image moderation but I can't really find a way to do it without paying a service for the use of their api's per image you upload. I don't know how fast that is and if it would factor into the latency of images appearing on the site or not. I know there needs to be some kind of moderation but I'd love to be able to just trust the world. Maybe a reporting feature.

Week 3:

This week was spent mainly finishing up assignments and doing some presentations. It's a weird feeling that the degree is pretty much over now. (bar this project..) I've also finished the 200 word profile about the project for the showcase booklets and got my photograph taken. It all seems so official now. My word profile was approved and I can move on to studying with coding breaks.

Week 4: Study

Full study mode has begun for these exams. I used to take study breaks between coding sessions but now i'm doing the opposite. There is so much to learn in these final weeks and I'm still figuring out the ideal levels of functionality for the project. Error code 8 seems to be less consistent now but I still can't figure out what causes it or even replicate it on command. This is incredibly frustrating when trying to trouble shoot a problem.

April 2016

Week 1:

More study this week. The final set of exams are next week and they are pretty important. In my spare time I managed to stop the exit code 8 from happening. I couldn't tell you what it was from but it stopped happening so I guess that's good enough. For the record I deleted a bunch of packages that I wasn't using within the projects and it seems to work now. I've also spent the week finishing off some assignments and projects that were due. Not a lot of time for project work.

Week 2:

The exams start this week. The modules this year were fairly heavy on theory so require a lot of remembering and some rote learning. I don't particularly enjoy those modules but they also have to be done. I'm looking forward to the exams clearing up so that I can continue working on the project.

Week 3:

Exams. During some downtime for the exams this week I began working on the project poster for the showcase on the 25th of next month. Just following the template but these things always perplex me as to how much information is relevant. I don't want people spending 20 minutes trying to read an abstract on the project but I don't want it to be a one-line affair either. I'll have a look at some older posters.

Week 4:

The exams are over now and full focus is back on the project. I have a few things to finish up this week but they all revolve around cleaning stuff up. I've had to change my approach to hosting because Meteor JS have switched to paid-only hosting services and I don't particularly want to pay too much to host the app. There are a few meteor packages that deal with hosting the app on free platforms such as AWS. I'll probably spin up an ec-2 instance as I'm already using s3 for database storage. May aswell keep that in

the same platform. I've had to update some of the older functions for meteor 1.3, as they were a tad redundant. I've been testing as I go along, but I've not really but in some official testing standards yet. I'll do that next week.

8.4 Analysis Design Document

PHOTOPROMPT

PRODUCT DESIGN SPECIFICATION

Version *<1.0>*

<01/10/2016>

VERSION HISTORY

Version #	Implemented By	Revision Date	Approved By	Approval Date	Reason
1.0	*Sean Keeley*	*01/12/15*	*<name>*	*<mm/dd/yy>*	Initial Design Definition draft
1.1	Sean Keeley	04/05/16			Refinement

TABLE OF CONTENTS

1 Introduction

1.1 Purpose of The Product Design Specification Document

The Product Design Specification document documents and tracks the necessary information required to effectively define architecture and system design in order to give the project developer the outline of system design and architecture during testing and developing.

2 General Overview and Design Guidelines/Approach

This section describes the principles and strategies to be used as guidelines when designing and implementing the system.

2.1 Assumptions / Constraints / Standards

The applications design should be simple to implement but there is a large learning curve with a framework in such infancy. There are some small details to hammer down with latency between the client side and the server returning the images in a fast manner. The biggest factor for a smooth operation of this system would be a quick connection to the server from the users application. This is in order to ensure the smallest amount of downtime.

The use of cloud services such as Amazon Web Services will require quite a classified approach to setting up buckets for image storage. There is a problem with configuration information being leaked through unsecure project uploads, leading to huge unwarranted bills for services that you never started.

Due to the infancy of the framework and it's constantly updating processes, constant upkeep with forum documentation, research of capabilities and a certain level of contingency will be needed to bring this application to full functionality. The use of AWS cloud services and built in framework API's will be needed and have to be researched.

3 Architecture Design

This section outlines the system and hardware architecture design of the system that is being built.

3.1 Hardware Architecture

With the use of a web application platform, there is no extensive hardware need baring a connection to the internet. With the intention of serving the application as a cross platform experience, there will be no inherent need for a certain version of Android or iOS. With a mobile-first approach quickly becoming standard, a consistent experience across all devices and breakpoints will be important.

The user's input will be taken in on the phones touch screen or using a mouse on a computer, and the images will be displayed on the screen or the monitor - dependent on the device in use.

3.2 Software Architecture

The main client side application will consist of a front end that allows the user to interact with the overall system by checking the current topic, and then deciding if they want to upload a picture of that topic. It will require internet access for the upload and device file system access in order to pick a photo to upload. The main GUI of this web application will display

the topic, the timer of when that topic changes and any pictures that have been uploaded by users. This will be done in a call to the cloud database to retrieve the images for that topic.

The upload functionality will be handled using packages within Meteor. There are packages such a dropzone javascript or ok-grow. It will limit the file types and sizes of the uploads to allow for quicker and more consistent uploads.

Login systems and mobile-first front ends will be built using meteor packages such as accounts-ui and bootstrap. A mobile first design philosophy is key as it will deliver a consistent experience across all devices.

The server will contain the upload scripts, along with the topic that is currently displayed. The topic must be consistent amongst all users. The topic will be tagged within the database, along with a tag on each picture. This will allow for searching as well as history of topics to be checked.

3.3 Security Architecture

The main security aspect of this application lies within the uploading from the client applications, through upload scripts and onto the server database. The user image has to travel safely with the required information, such as tag, time, userID, preferably somewhat encrypted to protect information. User accounts will be secure as per standard. While this information is not necessarily that classified and important to remain private and anonymous, it is still important to introduce adequate security measures.

The use of Amazon Web Services is a contentious issue. It's an incredible service but it's not that secure. There have been many reported cases of server credentials being left in the public parts of the project and that

information being compromised for the personal gain of hackers. The credentials and configuration of the project must not be public in any capacity.

3.4 Communication Architecture

The main communication protocol for the project will be the standard HTTP protocol, which is the standard in any Node.JS system. NodeJS HTTP POST is built for high concurrency so user volume won't be a problem. I'd like to implement HTTPS if at all possible because it would increase another level of security for the users of the application.

There is no plan for a user messaging system, or user interaction in any capacity past a report button. This removes the need for any extra communication features such as chats and equivalent technologies. I can't foresee a need for user chat to be important in the context and scope of the project.

3.5 Performance

Performance is an issue due to the application constantly loading large image files from an online service. It has to be lightweight as to not slow down the network and usage of the application. If the user has to wait too long for an image to load, they will lose interest and exit the application - rendering a high level of importance on quick and efficient loading. This will require decent compression and limitation on what the user can actually upload. Using a cloud service will bring some speed to the process too.

The bulk of the slow down will come from the downloading of these images when the user loads the system. Rather than loading each image individually, it would make sense to load a small bulk of images for the

user to see initially and infinitely scroll through more images, loading a new bulk when the user requests new images. The user upload feature will also cause some slowdown but that is to be expected as upload speeds are not as fast as download speeds. The use of a cloud service like Amazon S3 will allow for scalability to not be an issue, along with the use of pointers to almost turn it into a content delivery network. So rather than pulling the images from a database each time, you are simply just telling the application where to look for the images and returning that location.

4 System Design

4.1 Use-Cases

The use cases can be found in the project requirements specification document.

4.2 Database Design

The database will primarily function as an NoSQL, MongoDB database as that is packaged within the framework. The only database requirements will be for user account information, random topic information and pointers to the images on the cloud storage. For early development, pointers to a local directory for the images will work just as well. The user account database structure comes prepackaged with Meteor and is done using MongoDB syntax. There will be a store that saves pointers to the images that are on the cloud service, turning whatever cloud service database is chosen into a psuedo-CDN (content delivery network)

4.3 Data Conversions

There are not too many data conversions needed in this project. The JSON encoding from the NoSQL Database will extract the relevant information and the images will just be stored as pointers and downloaded from the cloud service. There is no need to convert anything from SQL for example because MongoDB uses JSON as default. The image will be stored as a pointer link and then sent to the client in order to be downloaded and displayed. The use of NoSQL is quite proactive in this regard because there is not a lot of information, especially sensitive information, that needs to be stored locally.

4.4 Application Program Interfaces

The system will incorporate multiple API's for a complete suite of functionality. Namely they will be the accounts API built into Meteor for accounts, the oAuth API for further login functionality and the image API for uploading and downloading images.

The accounts and oAuth API's are both for user account creation and use. The image API will handle the majority of the functionality and point to the project. It will allow a user to upload an image from the chosen and current device they are on, send that image to a cloud stored database and then return the image once that transfer has occurred. Another call will be made everytime the user loads the application in order to return the images to the homepage.

4.5 User Interface Design

There are two main screens for the user client UI to show. A login screen where a user enters valid credentials in order to see the main home screen. Once valid credentials are entered, the user will see the home screen where the current topic is displayed, the time remaining and the most recently uploaded pictures from other users.

Mobile first design calls for this to be one screen, with all of the relevant and valid information available within one glance. Uploading, viewing, deleting and logging out can all be accessed within one screen from an easily determined point. If you think of a website accessed on mobile as a graceful degradation from the desktop standpoint, mobile first aims for a progressive enhancement from the mobile to the desktop.

4.6 Performance

Performance should be fairly quick in regards to downloading and uploading images and will only depend on the speed and strength of the connection on the client device. The use of cloud based storage as a pointer over a true data store allows for less load time on the client side, as there is less need to load new data, rather just redirect the user to where the existing data already is. Amazon S3 will probably be the chosen cloud service which has features such as geographic redundancy and built in scalability protocol should prevent slowdown from the usage of a cloud service. This will also provide reliability for the application.

This bodes well for client data usage, as there will be little reason to download large amounts of data. The images themselves will not downloading onto the device, your devices browser will just be redirected to the chosen location. Further testing is required but in estimation there should not be any large spikes in data usage from using the application.

Appendix A: References

The following table summarizes the documents referenced in this document.

Document Name and Version	Description	Location
<Document Name and Version Number>	*[Provide description of the document]*	*<URL or Network path where document is located>*

Appendix B: Key Terms

The following table provides definitions for terms relevant to this document.

Term	Definition
[Insert Term]	*[Provide definition of the term used in this document.]*
[Insert Term]	*[Provide definition of the term used in this document.]*
[Insert Term]	*[Provide definition of the term used in this document.]*

BSHC4

Requirements Specification (RS)

PhotoPrompt

Sean Keeley
Upload Date: November 6th 2015

Requirements Specification (RS)

Document Control

Distribution List

Name	Title	Version
Eamon Nolan	Lecturer	
Paul Hayes	Project Supervisor	

Related Documents

Title	Comments
Title of Use Case Model	
Title of Use Case Description	

Table of Contents

Introduction

Purpose

The purpose of this document is to set out the requirements for the development of the mobile web application, PhotoPrompt. PhotoPrompt is a mobile friendly web application that gives users topics to which they can go out and take a picture of with their smartphone. The topic changes at a given interval and then users can begin to vote on which they think is the best photograph to represent a given topic. Users can view previous topics and the photos that are associated with that topic. This application will follow mobile-first design pattern meaning that it will be designed with the end product looking the best on the smaller devices, and flowing up to the larger screen sizes.

The intended customers are smartphone users who enjoy taking photographs on the go with their phones. It has become a common thing to do now, when you see something interesting you can capture that moment easily and in high quality on your smartphone. I often find myself wanting to take pictures without any inspiration so by using this application you can gain that little push of inspiration.

Project Scope

The scope of this project is to develop a mobile friendly web application. The project was decided to be a mobile friendly web application because that is where I see the future of development going. Rather than native applications which require a large amount of effort to update or fix bugs, a website that is mobile friendly, works across all devices and has similar functionality to an application gives you the ability to push new features and fixes out quickly. During my work placement I did some work with responsive websites and even made a web application for the office with an eye for responsive and mobile-first design.

The system will have a login and registration system, a HTML, Javascript and PHP front end experience with CSS to deliver the experience across all devices. There will be a PHP inclusive backend experience for any back end needs the application may come across. I will store user details and photo information into a database for quick I/O when the photos are needed for display. I plan to host the whole application on an EC-2 instance of Amazon Web Services Cloud Hosting system. This will give me a reliable uptime and dependable scalability if needed. EC-2 acts like a mini cloud server and allows me to run many server instances for multiple purposes to have faster load times and reliable uptimes. I plan to package the whole application together using some kind of deployment manager such as Cordova or PhoneGap in order to create an Android .apk file for anyone who does want to use the application as a pseudo native application. Creating an .apk file seems like a slight contradiction from what I wrote above but it's not. The idea behind create a web application and then creating an .apk file avoids the more difficult use of the native android language and development systems. By making a mobile-first web application it allows me to focus on my web development strengths.

There are no outlying cost requirements for the development of this application bar the chance that one may go over the free-usage tier while using AWS (Amazon Web Services). This free-usage tier should cover all the needs of the application quite comfortably but there is a chance that running beyond the free tier is possible. This could arise from too many requests being sent or too much data being stored. This constraint will have to see this project code be efficient in the amount of traffic it generates for the server.

Definitions, Acronyms, and Abbreviations

AWS - Amazon Web Services

HTML - HyperText Markup Language

(S)CSS - (Sassy) Cascading Style Sheets

JS - Javascript

PHP - PHP: HyperText Processor

EC-2 - Elastic Cloud

User Requirements Definition

From a target market perspective and the perspective of someone who uses smartphone applications daily - the main user requirement lies with ease of use. A well thought out and planned user experience is key to the success of any mobile endeavour. As PhotoPrompt is a mobile web application over a native application, the need for a clean flow similar to the experience of a native mobile application is key.

Requirements Specification

The user shall go through a welcome screen once they have been registered. This should allow the user to be completely familiar with the workings of the application and therefore the ability to use the application. This means that once the user has completed this they should be able to use all of the system functions within a few minutes. This information will be available to the user even after the 'tutorial' has been completed.

Functional requirements

Some of the basic functional requirements within this project are that the device you are using either has a camera or a hard drive in which you can store pictures.

1.1.1 Requirement 1 <Upload A New Photo>

1.1.1.1 Description & Priority

This is a description of the main functionality of PhotoPrompt, the ability to upload a photo to the main website. This is an incredibly essential functionality of the system as it is the main USP of the project.

1.1.1.2 Use Case

Scope

The scope of this use case is to evaluate the needs of the user and the system in order to upload a photo and interact with the application as a whole.

Description

This use case describes the processes behind uploading a new photograph to the system.

Flow Description

Precondition

The user has logged in successfully and the system is waiting at the homepage. There is a keyword and a timer displaying on the homepage of the system

Activation

This use case starts when a user clicks the 'Upload Photo' button.

Main flow

1. The system identifies the button click and sends a request to the user device.
2. The user interacts with the pop up screen and chooses a photo to upload.
3. The system then takes that information and uploads the photo to a database and stores a CDN pointer.
4. The system validates the photo is of correct format and is within the size regulations.
5. The user fills out a description and then confirms the upload.
6. The system sends a request to upload the photo
7. The system displays the photo on the users homepage

Alternate flow

A1 : The user uploads a non-photo format

1. The user chooses a non-photo format file to upload - such as .mp3 or .java
2. The system attempts to validate this upload.
3. The use case continues at position 4 of the main flow

Exceptional flow

E1 : Invalid photo format.

4. The user chooses to upload a file that is not of photo format. This could me .mp3 or .java et
5. The system cannot upload this file because of pre-determined restraints.
6. The use case continues at position 2 of the main flow

Termination

The system then presents the homepage again, with the users newly uploaded photo being displayed.

Post condition

The system goes into a wait state, for another photo to be uploaded or another functionality to be invoked.

1.1.2 Requirement 2 <User Registration>

1.1.2.1 Description & Priority

This is a description of the requirement of User Registration. This has a high priority because the user will not be able to interact with the application without going through the account creation and registration process.

1.1.2.2 Use Case

Scope

The scope of this use case is to evaluate the needs of a user registration system and the flows held within

Description

This use case describes the user registration process for PhotoPrompt.

Flow Description

Precondition

The user navigates to the website and is met with a Login Screen. A new user will need to follow to the 'Create an Account' screen in order to access the application as they will not have any login details.

Activation

This use case starts when the User clicks 'Create an Account'

Main flow

1. The system identifies the user trying to create an account
2. The User enters the details into the form
3. The system validates that these details fill out the security criteria
4. The user receives a validation or failure message
5. The system sends an email to the user on a successful request

Alternate flow

A1 : The User does not enter a valid email

1. The user enters a non-valid email address (without @ or .com etc)
2. The system will not validate this request
3. The use case continues at position 4 of the main flow

Exceptional flow

E1 : No Internet Connection

1. The user attempts to create an account
2. The system cannot complete the valid requests because of a non-existent connection to the internet and therefore the database.
3. The user reconnects to the Internet
4. The use case continues at position 1 of the main flow

Termination

Upon successful form validation, the system will then present the PhotoPrompt login screen, where the user will then log in with their new login details. Otherwise, upon the failure of validation, the system remains on the account creation page.

Post condition

The system goes into a wait state

Non-Functional Requirements

Performance/Response time requirement

As a web application the performance and response time requirement is an important one. Being built on an EC-2 instance allows the web application to be quick, responsive and easily scalable.

Availability requirement

Using AWS as a hosting medium is a paradigm to availability. In 2014 Amazon EC-2 was only down for 2.14 hours across 20 different outages. This means that the average site had an uptime of 99.9974% during 2014. This is the highest uptime statistic of any cloud hosting platform in 2014. This was partially the reason behind my choice of using Amazon Web Services.

Security requirement

As there is a user account system, there will have to be sufficient security requirements. There will be password salting and hashing and no passwords will be stored in plaintext. Nothing will be transmitted in plaintext and there will always be some form of encryption.

Reliability requirement

As stated above using Amazon Web Services has an incredibly reliably service. There was a 99.99974% uptime for 2014 - which is amongst the highest in cloud hosting platforms. This garners a very safe and high sense of reliability within the project. I will also run a second instance for testing alongside the client facing one. This will mean that I cannot break the user experience through the introduction of new features of bug fixing.

Maintainability requirement

As PhotoPrompt is a web application, there can be a quick turnaround time for new features, bug fixing and testing. I chose the web application method because of this because of the rigid testing and approval processes that go along with native applications and applications stores.

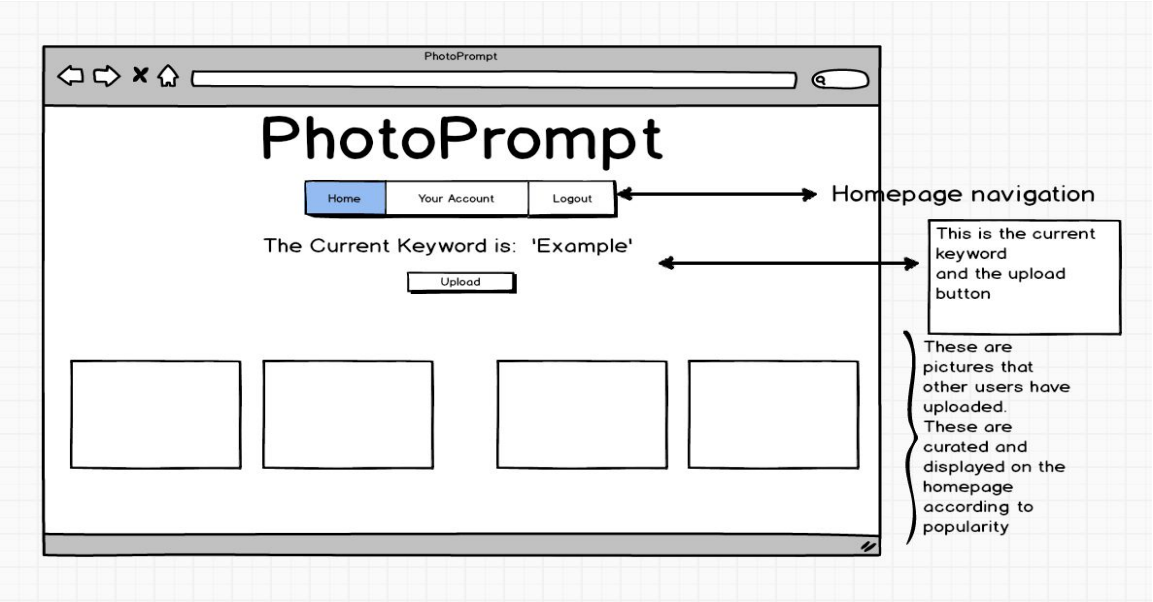
Portability requirement

PhotoPrompt is a mobile-friendly, responsive web application meaning that the portability requirement is quite high. The experience will be the same and fluid across any device that you may use. There will be a mobile view, a tablet view and a desktop view. This means that the application has a very portable aspect to its creation.

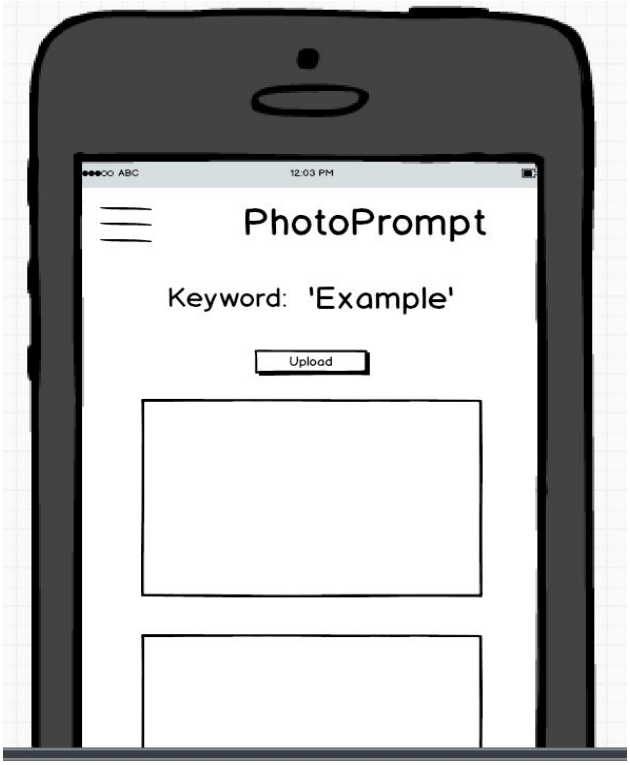
Interface requirements

GUI

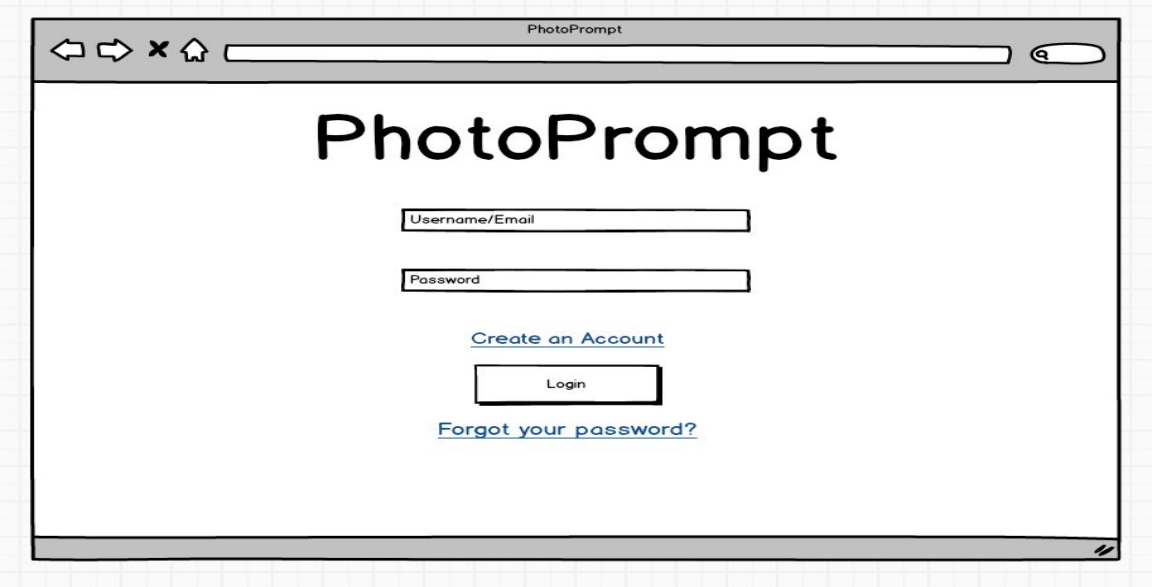
Homepage On Desktop:



Mobile View Desktop:



Desktop Login:



Application Programming Interfaces (API)

I will integrate the Twitter, Facebook and Instagram API's to allow the users to notify their other accounts when they use my application. Sending push notifications to their independent timelines will give PhotoPrompt some exposure and bring in extra users from people seeing their friends using it. Using these API's will require me to follow the steps that are available on the corresponding websites. I may in the future integrate the Google Maps API to show the location that a picture was uploaded from.

System Evolution

This system application can be easily scalable in the future. Using AWS adding more instances is an easy task. This can handle more load, and more requests per minute. As per functionality evolution, this application will be a living application. Meaning that I will constantly write new functionality and features as they come to light. The basis of the application will be the above but the application will be suspect to some scope creep as the development continues. With the nature of the application being a web application, this means that there can be quick fired bug fixes and new feature pushes with minimal downtime and no waiting around for the app stores to confirm the updates.

8.6 Other Materials Needed

The following are materials I referenced from throughout the course of the project. Any code that has been sourced from a source that is not my own will be appropriately commented in the comments section of the source files. Otherwise all code is a representation of my own work.

References:

GitHub. 2016. GitHub - meteor-velocity/velocity: A reactive test-runner for Meteor. [ONLINE] Available at: https://github.com/meteor-velocity/velocity. [Accessed 10 May 2016].

Meteor. 2016. Meteor. [ONLINE] Available at: https://www.meteor.com/. [Accessed 10 May 2016].

Documentation - Meteor. 2016. Documentation - Meteor. [ONLINE] Available at: http://docs.meteor.com/#/full/. [Accessed 10 May 2016].

The Meteor Chef. 2016. The Latest | The Meteor Chef. [ONLINE] Available at: https://themeteorchef.com/. [Accessed 10 May 2016].

GitHub. 2016. GitHub - arunoda/meteor-up: Production Quality Meteor Deployments. [ONLINE] Available at: https://github.com/arunoda/meteor-up. [Accessed 10 May 2016].

Matt Silverman. (2016). Number of mobile app downloads worldwide from 2009 to 2017 (in millions). Available: http://www.statista.com/statistics/266488/forecast-of-mobile-app-downloads/. Last accessed 2nd Feb 2015.

-. (2014). 23 DAYS A YEAR SPENT ON YOUR PHONE. Available: http://www.mobilestatistics.com/mobile-news/23-days-a-year-spent-on-your-phone.aspx. Last accessed 2nd Feb 2016.