Kodak Facebook Collage Project

Team TBD
Peter Bergeron  Kevin Kuchta  Matt Olenik
Bernard Stern  Chris Tosswill

Project Sponsor
Robert Janiszewski
Lawrence Wolfe

Faculty Coach
Kenn Martínez

Kodak Facebook Collage Project
Team TBD
Project Sponsor
Robert Janiszewski
Lawrence Wolfe
Faculty Coach
Project Overview
Basic Requirements
Constraints
Development Process
Project Schedule: Planned and Actual
System Design
  Major Design Choices
Process and Product Metrics
Product State at Time of Delivery
  Completed Features
  Future Work
Project Reflection
References
Project Overview

The goal of this project was to provide a solution that enables the creation of photo collages from a set of source images, and a Facebook application that integrates this solution with the photo content in a user's profile. The source images for the collage come from photos that either the user or their friends have uploaded to Facebook. Users are allowed to select which images are added to the collage as well as modify their layout in order to create something unique and personalized. Users are able to save their collage as an image file, as well as share it with their friends using the Facebook news feed. This will help the application spread virally through the Facebook community. The solution which generates the collages is separate from the Facebook application so that this functionality can be reused in future Kodak software projects. In addition, this solution may, in the future, support tie-ins for Kodak's fulfillment services.

In addition to the core functionality, several other features have been added to the application, such as automatic tagging and image cropping, which were not initially required. There was a very long list of features that were desired by the sponsors, but not required in the final product. Of these, a small number of those features were implemented based on their importance to the sponsor and the time available to the developers. The team chose Scrum as their development process. The process was very effective in allowing implementing features based on available time, and allowing the sponsor plenty of opportunities to tell the team which features they next wanted to include, or which parts of the system needed improvement. The process chose also yielded high visibility to the sponsors, and allowed them plenty of opportunity to give feedback.

Basic Requirements

The project sponsors indicated their desire to have a collage application which would attract users. The most basic requirements were to have an application which would allow users to choose pictures from their Facebook albums, arrange the pictures in a collage, and post the final collage image to their wall.

As a Facebook application, the system would operate in an iframe in Facebook, and integrate with the Facebook API. The application would need to retrieve images from Facebook and use to generate a collage. It would need to upload the image of the generated to Facebook, and post to the user's wall.

Users of the application would range from experienced computer users to users possessing only the most basic computer skills required to use Facebook. The application would need to support Internet Explorer version 7 and newer, and Mozilla Firefox.

Constraints

Constraints originated from a number of sources. The team was limited in its size, resources, and amount of time available. Several features that the project sponsors originally wanted to include in the application would have required a significant amount of work, which the team did not have the time or resources to accomplish. For example, the sponsors expressed an interest in a feature where the application would analyze images and automatically create an aesthetically and logically appealing collage based entirely on image content and metadata. This is a very difficult and complicated problem involving numerous challenges in artificial
intelligence and image analysis. After holding discussions between the development team and the project sponsors, it was decided that this would be one of several “stretch goals” the team would work on only if they had time.

One source of constraints was the Facebook terms of service. The terms of service are located at [http://www.facebook.com/terms.php](http://www.facebook.com/terms.php). Section 9 of those terms is relevant to application developers. In addition, Facebook lists additional restrictions for Facebook applications on their Facebook Platform Policies page, at [http://developers.facebook.com/policy/](http://developers.facebook.com/policy/). For example, Facebook places limitations on what can be done with data (including images) obtained from Facebook, and states that such data should not be stored long-term.

**Development Process**

The team decided on scrum as a development methodology. The project sponsors had requested the team use an agile process that yielded high visibility for the customer. In addition, the flexibility of the development process was very beneficial. There were many stretch goals for the project that would only be accomplished if there was time. The process allowed the team to analyze its progress and accomplish the optimum amount of our project goals, while avoiding spending time working toward goals that the team would not have time to accomplish. In addition, the project sponsors would be able to provide their own input, modify priorities of planned goals and features, and add additional requirements at their own discretion. The flexibility of the process was also useful for dealing with many unknowns early on in the project. Technologies and API's that would be chosen or used, such as OpenCV, the Facebook API, and VML/SVG, were unfamiliar to the team, making it difficult to make early estimations on the difficulty of related tasks.

The team's process consisted of a series of two week sprints. Each sprint consisted of a series of tasks assigned to different members of the team. At the start of each sprint, the team would analyze their progress on previous sprints. The team met with the sponsor to discuss their progress and determine which tasks the sponsor most wanted the team to work on over the course of the next sprint. The team would then estimate the time required to complete each task, and assign tasks to users. The first sprint, started shortly after team formation, was referred to as “Sprint 0”, and focused mostly on establishment of a process and a plan for future development. To manage tasks and track progress, the team chose the software tool Agilefant.

Eventually, the team recognized the need for the ability to record and track defects, which was not supported by the current tools and process being used. To address this, the team installed the bug tracking software Trac on one of their servers, and integrated defect tracking into their process. Defects, when found, were recorded in Trac. At the start of the next sprint, tasks would be created to represent the bugs that needed to be fixed.

**Project Schedule: Planned and Actual**

As a result of the team’s choice of Scrum as the development process, development time was divided into two-week long sprints. At the beginning of each sprint, the team would decide which tasks to work on, based on discussions with the project sponsor.
The first sprint was referred to as “Sprint 0”, and began not long after team formation. It consisted of the establishment of the development process, research into technologies, and the formation of a plan.

The progress of several individual sprints was discussed more thoroughly in the “Process and Product Metrics” section of this document.

Rather than have a well-defined schedule for software development, the process used by the team dictates that the team have a backlog of high-level project requirements. The project sponsor can control what gets put in the backlog, and can determine a priority for every item in the backlog. At each sprint, those requirements were split into tasks, which were then assigned to individual developers.

**System Design**

There were several constraints and requirements that had to be taken into account when designing the architecture for the system. For example, the application is a Facebook application, which necessarily demands that part of the system be a web application. The system was designed to be modular, extensible but simple enough to be quickly worked into prototypes for customer feedback. Familiarity with technologies amongst the team members was also considered when choosing tools for implementation.

**Major Design Choices**

**Facebook application** - The application is accessed via Facebook, meaning our application had to be at least partly a web application. The application has multiple views and substantial dynamic content, which lead us to choose from popular web frameworks that are known to work well for this type of problem. The two major candidates were Ruby on Rails and ASP.NET MVC. Rails was chosen for its quick rapid prototyping capability, extensibility from community plugins and familiarity amongst team members. ASP.NET MVC, while similarly adequate in terms of capability, is not as mature and less familiar to the team. It was initially considered due to easy integration with OpenCV for image processing, a decision that became moot once image processing was moved away from the web application.

**Facebook integration** - The primary and most flexible way to create an in-Facebook application is to use Facebook’s iframe method- that is, apps.facebook.com/yourapp loads some external content in an iframe. The external content is our Rails app running on an SE VM,
and interfacing with Facebook using the Facebook Graph API. While this presented some early
difficulties involving Facebook authentication, it ultimately allowed us complete control over
whatever the user saw within our app frame.

**Rails layer** - Rails is a framework which uses a model-view-controller architecture to structure
its design. As such, our code used MVC as well. We deviated somewhat from the most common
rails design, however, by using Rails’ ActiveModel for our model, rather than ActiveRecord. This
was because we have no database in our system (which is what the traditional ActiveRecord
tool in rails (named after the active record design pattern)). Our model encapsulated calls to our
image processing service and to Facebook’s API. As such, we wanted to wrap those calls in a
way that would make them look like regular Rails model interactions, such that other tools in
Rails (like easy data validation, form building, etc.) could be used without modification.
ActiveModel allows us to hide these external data sources and treat them as any other model in
Rails.

**Image processing service** - The team initially considered doing image processing directly in
the Facebook application but soon ruled out this idea for several reasons. The major rationale
centered on the sponsor’s request to have image processing potentially be compatible with
other social services, such as Flickr or MySpace. In such a scenario, tying the image processing
to Facebook does not lead to easily extending the application. Additionally, the team was
considering using OpenCV for image processing, which is not easily used from Rails, the team’s
technology of choice for the user facing website. Separating image processing as a service
allows more flexibility for image processing technology, a critical component of the application.

The team decided to use OpenCV for image processing, the most powerful and flexible image
processing library available. Other libraries were considered but OpenCV was chosen due to
fewer potential bottlenecks. The team didn’t want to worry about running into processing
limitations with simpler libraries. This decision was weighted against the use of the difficulty of
using OpenCV, a C library. High level language wrappers for OpenCV were evaluated and the
team found that EmguCV, a .NET wrapper, was among the most stable and complete.

The image processing layer consists of two components – an actual web service that exposes
behavior to the Facebook application and an image processing library that does the actual
photo manipulation. Both components were written in C# targeting .NET 4.0. This lent itself very
well to the use of OpenCV, using the EmguCV wrapper. In the event that EmguCV did not
satisfy our requirements (incomplete, broken) in any capacity, a simple interface to the C library
could be created with C++/CLI, a .NET interop technology familiar to several team members.

The image processing functionality was exposed as a web service using WCF (Windows
Communication Foundation). WCF is Microsoft’s now recommended way of writing any kind of
cross-process service for the .NET platform. It provides a powerful framework to abstract away
the details of service implementation and transport (e.g. SOAP, HTTP web service with JSON,
binary IPC) allowing the team to focus on the functionality of the service. This choice proved an
excellent one, as very little time was spent working on configuration aspects of the service.

**User Interface** - The user would need to create and edit a collage in their browser, which
demands a complex user interface with complex UI elements and drawing functionality other
than just the simple links and buttons in most web pages. Since Internet Explorer 7 and 8 were
among the required browsers, the team would not be able to use features exclusive to SVG or
HTML5. In addition, the team and sponsors decided not to be dependent on any third-party
browser add-ons, such as Flash or Silverlight, since many users do not have the necessary
plug-ins installed. Instead, the team relied on a JavaScript library called Raphael, which provides a cross-browser interface for creating and manipulating vector graphics in JavaScript applications. It does this by using SVG in browsers such as Firefox, and VML in versions of Internet Explorer that do not support SVG. However, the functionality provided by Raphael did not completely cover everything needed for the application, so in several parts of the JavaScript, there was a need to manipulate SVG or VML directly.

While the collage editor page contained the most complex user interface, the application's overall UI allowed the user to take a series of steps to create and share their collage. The flowchart to the left shows the original design for the application's UI. When a logged-in Facebook user loads the start URL (usually within a Facebook iframe), the application determines if the user has granted the application the necessary permissions, such as the ability to access photos uploaded by the user. If the user has not authorized the application to do these things, they are redirected to the Request Authorization page, where they can give those permissions to the application. Once the application is given authorization, the user is taken to the photo select page, which shows thumbnails of photos uploaded by the user, arranged by application. The user can select which photos they want to include in their collage by clicking on the thumbnails of these photos. When they are finished, they are taken to the Edit Collage page. When finished, they can view their finished collage. From there, they can do a number of things, such as save as a photo on Facebook, post to their wall, or invite their friends to view the application.

Process and Product

Metrics

The software used to track our progress was Agilefant. Agilefant provides an excellent set of tools for tracking and analyzing performance. The progress of each sprint is shown in a burndown chart, which shows the “burndown”, or amount of estimated effort required to complete all tasks assigned to the current sprint, side by side with the optimal burndown (a steady and equal decrease in effort left on each weekday of the sprint).
This image represents the burndown chart for the first sprint. The blue line represents the optimal amount of work left to, based on our initial estimates. The red line represents the amount of work left to do based on the work that was actually accomplished. As the graph shows, the two lines did not match as they should. The amount of remaining effort actually increased early on in the sprint, due to tasks being added or estimated partway through the sprint.

This chart, as well as the charts for other sprints, shows very erratic progress. This is due to the way in which the team members work. As college students, the members of the team have many other commitments and work. Unlike an employee who will work on a project for 8 hours each day, the members of the team were managing their own time in such a way that they were not working on the project a consistent amount of time each day. As a result, many of the burndown charts showed erratic progress. This was not a problem that needed to be addressed, but merely a side effect of the way the team members were allocating their own time.

In addition, the chart also estimates that no effort is spent on weekends. As college students, the team members were not confined to working during work hours, and were just as likely to spend effort on a weekend as a weekday.
The burndown chart for **Sprint 3** shows a significant improvement in the team’s working habits and estimation ability. Although still very erratic, the amount of remaining work tended to stay closer to the estimates. However, close to the end of the sprint, the remaining work increased. This is a sign that a task was added or an estimate was changed, indicating that the value of "effort left" for one or more tasks had been increased by one of the developers. This is an indication that work was not progressing as well as predicted. In addition, the remaining effort at the end of the sprint was higher than usual, which indicates that several tasks were not completed during this sprint.

The burndown chart for **Sprint 6** is an excellent example of how the team members, as college students, allocated their own effort on the project differently than typical office workers working on 40-hour work weeks. This sprint took place in the middle of RIT’s spring quarter. As a result, there is a long period of time where very little effort was spent on the project, as the students were focusing on the enormous amount of class-related work typically experienced by RIT students halfway through a quarter, such as studying for midterms and upcoming project due dates. However, the team members had not forgotten about their commitments to the project. Starting at the beginning of the next week, the burndown chart shows that the remaining effort decreased at an impressively rapid rate, indicating that team members were working harder than usual on the project.
The burndown chart for Sprint 7 shows that the team made progress faster than initially expected. There were several instances where effort left increased, but overall, the effort left tended to stay below the projected effort left. This was likely a result of less ambitious estimates, resulting from the amount of work that was not accomplished in the previous sprint. In this sprint, tasks were completed quicker than expected.

**Product State at Time of Delivery**

The product is in a state where it could be successfully used to create a collage by a Facebook user. The core functionality of the software is working, and it successfully meets the most basic requirements. In addition, a number of extra non-essential features were implemented, such as automatic tagging. The software as delivered contains a few recorded defects, but none of those inhibit a critical portion of the application’s functionality. The team believes that only a small amount of additional work would be required to make the application ready for production.

The project contained a large number of stretch goals. These features were not necessary for the final product, but would have implemented if there was time to do so. As a result, the list of unfinished or unmet goals is very large. However, this in no way indicates that the team failed to deliver the product that was promised. It also does not in any way indicate that the product is far from ready to be put in production.

**Completed Features**

**Facebook integration** - The application was successfully implemented as a Facebook application, which correctly uses the Facebook API and can be used like a typical Facebook application.

**Photo selection** - The user is able to see all photos they have uploaded, arranged by album. They are able to select the desired images, and proceed to the next step.

**Collage editing** - The user is able to edit their collage. Users are able to:
- Move images to any location on the canvas
- Change the size of images
- Rotate images
- Place overlapping images in front of or behind each other
• Crop images
• Choose a background color for the collage
• Choose the width and height of the collage

Automatic tagging - The application automatically tags Facebook users in the collage image based on which users were tagged in the images used to create the collage. This requirement was added by the project sponsor late in the development of the software, and was implemented by the team.

Future Work

Usability and appearance - Although the application is functional, it would be helpful to add polish. Additional work is recommended on modification of styles and graphics used in the user interface. Such modifications could improve the usability of the application, as well as make the application appear more professional.

Collage editor UI improvements - There were several enhancements to the collage editor UI that were discussed, but not implemented. Among these were:

• Scrollbars - Although scrolling functionality was implemented in the editor UI, the act of scrolling can only be accomplished through use of a mouse wheel (or similar device), or by holding down the alt key. Therefore, the scroll feature has very poor discoverability. The addition of scrollbars would add to the discoverability of the scroll feature.
• Multiple Selection - In many user interfaces, the user is able to select and manipulate multiple objects by holding down the shift key while selecting objects. Although there was stubbed-out code in the JavaScript used for this page, the ability to select and manipulate multiple collage images was never fully implemented.

Photo selection page UI improvements - As one of the last user interfaces to be implemented, the photo selection page lacks polish. Additional work is recommended to improve the appearance and usability of this page. Recommended improvements include:

• Better next/previous buttons. The small size of the buttons means that, according to Fitt’s Law, it is more difficult to click these buttons than necessary. In addition, it may be beneficial to show whether clicking these buttons will show additional photos through the use of visual feedback, such as disabling and “greying-out” these buttons when there are no additional images will be shown as a result of clicking on them.
• Better visual indication that images are meant to be clicked on. This could be achieved through the use of the CSS “cursor” property, or the CSS “hover” pseudo-class. In addition, it is recommended that usability improvements are made so the user has a better understanding of what action will be performed when clicking on an image.

Background image - The project sponsors expressed the desire to allow the user to set a background image. The background image would differ in appearance from other images in the collage in that it would visually appear to be the background. This would be achieved through one or more effects, such as lowering the contrast, decreasing the alpha of the background image, or tinting the image.

Collage dimensions in context of printing - The sponsors from Kodak expressed dissatisfaction with the current definition of the width and height of the collage. As delivered, the dimensions are defined as width and height in terms of pixels. The sponsors indicated a desire to use other information, such as DPI and inches/centimeters, as a typical non-technical user
would not understand size in terms of pixels. It was suggested that the user be allowed to choose from among a list of image dimensions.

**Automatic photo suggestion** - One stretch goal that was discussed was the ability to automatically suggest additional photos to be added to the collage once the user had made their initial selection of photos. The software would look at several factors, such as tagged users, photo composition, comments made, and date uploaded. Using these factors, the software would find similarities between images chosen by the user and other images in the user's albums. Photos that were similar to those selected by the user would be suggested to the user before they began the process of editing their collage. This feature was omitted from the final delivered project for multiple reasons. Primarily, the time available and size of the team created limitations on what goals could be accomplished, and, as this was not an important goal, no work was done on this feature. In addition, the information required by the software to find similarities between images would require that the user grant additional permissions to the Facebook application. This would have created a greater risk of users choosing not to use the application when requested to authorize permissions to the application.

**Intelligent automatic photo arrangement** - Another stretch goal was the automatic creation of a Facebook collage where the images would be pre-arranged such that similar images were closer together, or so that the initial arrangement would look appealing to the user. This was omitted for several reasons, including the difficulty of implementing such a feature and the time and resources available to the team. In addition, similar to automatic photo selection, finding similarities between photos would require the user to authorize additional permissions to the application.

**Redeye removal** - Another stretch goal added to the project was the implementation of a redeye removal feature. To implement this feature would require the creation of a user interface to allow the user to control the redeye removal process, as well as implementation of actual redeye removal functionality in the image processing layer.

**Better automatic tagging** - Users tagged in a photo are also tagged in the collage in which the photo was included. However, currently, the location of the tag is at the center of the photo in the collage. To improve this feature, the application would have to determine where, relative to the photo, the user was tagged. It would then determine where this location is relative to the collage that includes the photo, taking into account the placement and size of the photo, and whether or not the tagged person was off screen or cropped out of the photo.

**Project Reflection**

Overall, we feel that the project went well. The project being delivered is functional, and would not require much additional work to be ready to be put into production. We believe that Scrum was a good choice for a software process, as it allowed us to work on tasks and features we had time to accomplish.