Team Empiricist

"I don't believe in no-win scenarios" -James T Kirk

Purpose

The purpose of this document is to provide evidence for the different roles of the team members.

Introduction

This term, our team went through a long journey to reach our final product. While we initially were on track, and felt we had the tools we needed, we came up against challenges which seemed out of our control. Much like the original Star Trek's "final project", the Kobayashi Maru, the problem seemed unsolvable. Our team stuck together, completely altering our team organization strategy and responsibility structure, to overcome the issues we came up against. We switched from dividing work to group programming, working with our professor and TA's, redoing as much as needed to attempt to overcome our issue. Once we had solved our roadblock, we met almost every day for many hours to catch up in time to meet our final goal. Tasks were allotted fluidly and flexibly, with members working to their strengths but also shifting to assist each other when needed. By the end of the project, every member had worked at least briefly on every part of the code. Thankfully, unlike the Kobayashi Maru, we did not need to change the rules to be successful- just the way we worked together.

Our process focused on flexibility and communication. Meetings were organized via group chat on a daily basis, and daily tasks were assigned based on preference and experience at each meeting. Our collaboration style also depended on the problem at hand; some meetings we worked individually, others we would work as a group or in pairs to talk through a particular error.

Our main tools of development were Eclipse, AWS, and Git. The first two were fundamental to the development of the project, as they substantiated the assignment, but Git proved to be incredibly useful (though occasionally challenging) in allowing us to branch off and test new functionality. When we hit our biggest roadblock, we developed test branches to determine if a certain section of the code was causing problems. While it ended up being an issue with our Amazon account permissions that was causing us problems, we were glad to have a way to eliminate certain possibilities.

While we did not accomplish everything we had hoped to, we made remarkable progress given our setbacks. Our goal was to fulfill 70% of system functionality, and have 60% code coverage. Our actual achievement of about 70% of system functionality and 70% of code coverage, which exceeded our goal. We feel we worked very well as a team, and while some improvements could have been made, we are happy with our accomplishments overall.

Team organization, members, and responsibilities

Our team had the idea of utilizing a divide and conquer method which was extremely efficient in the first stages of the team project. We were able to get the first few iterations done in a timely fashion and with fairly solid grades. However, this method started to become inconvenient when git coding become existent. Dividing and conquering was no longer a desired method since it meant that code needed to be spoken about alongside push and pull requests. We began to meet more frequently and take on more and more code as a team as opposed to dealing with individual assignments within our group project. We all became equals. There was no clear leader when it came to decisions or work. In fact, when life got in the way for any one of our team members, there was another member there to take their place. With that being said, our team stuck together like glue. We faced errors that would deter any team from the main course and we handled it as responsibly as one could. Whether this meant living in office hours, pulling all nighters, or running into numerous walls, we did it together. Despite our group not necessarily meeting deadlines for reasons that I don’t have enough pages for, we continued to persevere and march forward through this journey of Software Engineering.

Meetings

In the first Iteration, our group would meet twice a week to assign and then check tasks. While this worked well initially, as we encountered difficulties meetings became more frequent and consisted of actually working on the project goals rather than discussing and assigning them. From our second iteration onwards, meetings were organized on an almost daily basis via group chat, and would consist of usually 1-3 (or sometimes more) hours of the group working through problems.

Subteams were also initially helpful to ensure no one was overwriting each other's progress, but as we moved forwards most difficulties held up the entire group, so these subteams disappeared. For our first iteration, two members worked on the API while the other two learned about lambda functions and established database infrastructure. When we came across issues that stopped the whole group from progressing, we would pair (or group) program to find the error, sometimes switching back and forth between computers but only editing code on one. When we moved past our main obstacle, we would alot individual tasks in a given session, but we were flexible on those tasks and often switched or changed from session to session. We selected this method of organization simply because it was natural; some days, certain things needed to be done, and that list of tasks would change from day to day or moment to moment.

Our group was extremely flexible when it came down to the AWS portion of this course. For the first portion we appealed toward the strength of each individual member and what they felt comfortable doing. For example, Shannon and Vinit felt comfortable with Database stuff where Jerry and Stefano felt more comfortable with code-driven tasks. However, once we started coding for the larger picture, it didn't become so clear-cut with who had what tasks. In fact, if we had to give a responsibility, I would say that each one of us has fulfilled each responsibility at least once. For example, depending on who was working on code at the time had the responsibility of communicating a push/pull request. Each one of us has had that responsibility at different points in this final deliverable. Just like each one of us has had the responsibility of working on the HTML, Javascript, the Handlers, or even the database, and so on.

Process

As previously mentioned, our group was flexible about the methods used to progress through the project. While we initially started with individual work, as we came up against major problems our group would all work around one computer and suggest different potential changes, splitting up only to implement them. Once we had passed our major challenge, we went through each item of the assignment and made a to-do list. We began with the simplest tasks to establish fundamental functionalities, then progressed to the more complex challenges.

Because of the setbacks we experienced with iteration 1, we were not able to manage code quality or releases efficiently. Once we had surpassed these obstacles, we were able to decide upon reasonable goals as a group for our progress. For our final project, we did not expect to complete the entire assignment, but aimed to get at least 70% of G3 functionality, and 60% of test case coverage.

Good communication was essential to our success as a team. Whenever a meeting finished where not all members were present, one participant would send an update to the others. We also communicated (via text or in person) what we were working on and when, so that we did not have major merge conflicts. Despite the team's flexibility in terms of task allotment, no one was ever directly working on the same task who was not pair programming.

Tools

Aside from word processors, our team had access to a few key tools that have made this group project possible. The main two being Eclipse and AWS! Without these two, this group project would have been impossible to do. Two other tools were StarUML which was responsible for creating our UML diagrams, and Notability on iPad which gave us the ability to put together a pretty awesome Storyboard! Lastly was the use of GitHub, which is what allowed us to branch off of master and create new versions of code for testing that didn't conflict with working code.

Accomplishments

Deliverables

G.Analysis **Accomplished**	Throughout this deliverable, the team worked to go through the project definition and requirements and create the foundation for the rest of the project. This deliverable did not require any implementation but it did require the team to create use cases defining the actionable events that were displayed in the storyboard that was a mock-up of our GUI. In addition to the use cases, and the storyboard we also created a UML diagram as a way to layout all the functionality that would be present in the classes that were to come in the future deliverables.
G.API **Accomplished**	In this deliverable, our team worked to create an API for the system that would be used to define our requests which connected the front-end and the back-end together. In addition to the API, this deliverable included a sketch of the database that described how our tables were going to be set up and what would be included in the database.
G.2 Implementation 1 **Accomplished**	For this deliverable, our team worked to create a template of the HTML which was essentially bringing our storyboard to life where it was clear to see where all functionality would come into play when we implemented the back-end and Javascript. As far as the actionable elements in the HTML, the only thing that should have been implemented was the capability to see the list of segments and playlists that were to be put into our database as well as our segments should have been put into our bucket. Finally for this deliverable we had to show the data in our database and we have to provide a URL for a public video segment in the bucket as well as URLs for our landing pages.
G.3 Implementation 2 **Partly Accomplished**	In this deliverable, our team worked to make sure the segments appeared on the website without hardcoding them. Apart from this we

	also worked on the Handler and the functionality that allowed the user to create a playlist and also make sure that those playlists appeared on the front end HTML part of the website after ensuring that the back end javascript was implemented.
G.4 Implementation 3 **Partly Accomplished**	For this deliverable the goal was to focus on getting 80% code coverage for our src/main folder as well as to wrap up a git report for all the commits that were done in eclipse as well as include a README file that described how the project was to be run which included the final URLs.

Reflection

What worked, what didn't work

Our group itself had a very strong dynamic and developed a strong chemistry as the project progressed, so as far as things that didn't work for our group it was more-so complications with the system.

Our biggest mistake

I don't believe our team made any technical mistake that could have lead to a drastic setback. However, one thing that could have been done for lack of better things to list was to reset our AWS and start from scratch sooner than what we did. Our reasoning for not starting over at the first sight of internal server errors were embedded in the belief that something else might have been wrong and the fact that we just didn't want to get rid of our code. This is precisely what needed to be done, and our only regret is that we should have done it sooner.

Changes we would make

As far as changes to be made in respect to the project, restarting and creating a new AWS account earlier on in the process would have fixed our problems with the 500 errors however we did not know that the problem originated with the AWS account itself. In addition to this, running through the whole entirety of the AWS tutorials earlier on in the term would have been more helpful in the long run because some of the stuff we were learning as we progressed along with the project.

What We Have Learned

The most important lesson that was learned was teamwork. Despite all of our team's setbacks, we still found the ability to remain strong and continue marching on. There was not a belief in our mind to give up and so we hammered on. We had setbacks, and we had discouraging moments but we found value in them and progressed despite all challenges. We all come from competitive backgrounds, and **just** passing the class wasn't satisfactory to any of us. We wanted to raise the bar on all of the assignments, and we wanted to learn most importantly. We were able to do both of those whilst learning the value of teamwork, hard work, and dedication.

Advice to future teams

I would say to learn outside of the firehose of information. Despite the slides giving us all of the information necessary to complete the project, it is essential to take what the professor gave us and take it to the next level. There are many things that the slides don't teach you (not in a bad way), like internal server errors, and the only way to recuperate is to scrounge the web to see if anyone else has experienced the same problem. Self-teaching yourself more information throughout the course about Amazon Web Services can really ensure a positive AWS experience and overall success.