**Post-Mortem Report**

Class: 3307A – Object Oriented Design and Analysis

Professor: Michael Katchabaw

By: Team 08

Team Members: Shin Chang, Kirk Vander Ploeg,

Christopher Steward, Jacob Stirling,

Pedro Ortega Santos

December 8, 2017

**Project summary**

Technology is an continuously expanding and flourishing in all aspects of everyday life. From its many applications in medicine, communication and military, our society has become more intertwined with computers and gadgets to make out lives easier and more comfortable. One aspect of this effect is home automation, the use of remote or interactive programs that can be used to alter the conditions in one’s own home. The use ranges from a heating / cooling system, security or even lighting controls. For our assignment, we were tasked with interacting with lighting systems, specifically the Philips Hue brand lighting system by creating a web based service that could interact with the system, and allow users to access the Philip Hues many features from virtually anywhere inside or outside of their homes through a simple web browser.

The project required various parameters for us to follow. We had to assure that the program we made would be implemented in C++ as well as apply a Wt framework based on version 3.3.8. The overall website design had to be created by the members of our group, with both user interactive content (buttons, check boxes, text areas, etc.) as well as RESTful web services ( to connect and signal the app controlling the user home lights). In addition to functional code, the format of the code had to be made in a robust and efficient manner, with good coding practices and proper documentation. In addition to general parameters, we had to consider the Restful interface between the Hue emulator and our own website. As noted before, each group had to apply a Wt C++ framework, which we had previously acquired on the first individual projects at the beginning of this course. Unlike the project, however, we would need to utilize the Wt’s Http client as our connection point between the Hue emulator and our site.

Aside from the general parameters, specific features had to be completed including; a welcome/login, user account management, bridges, lights, groups, schedules and persistence sections. The welcome/login would contain a brief greeting and a location for the user to input their account information send them to their own page to alter their information as well as the conditions of their lights and bridges. The user management is a section a logged in user would see who they are once logged into our page, they could also alter their name in our database, their email and even their password if desired. The bridges section would entail letting users create bridges from the website onto the emulator and once created allow the specific user to edit, add, or remove said bridges from the website, and effectively affecting the bridge on the Hue emulator remotely.

The lights feature required that we allow the user to manipulate a specific lightbulb’s settings like its on/off status, brightness, and color. In addition to these, a user was to be able to set a timer on how long the color transition should take. The groups feature required that the lights be able to be grouped together and allow for the user to simultaneously alter the state of many lights they added to the specific group. The schedule feature required that the user be able to create a set timeframe for a light, group or bridge so that when it reaches 0, a desired change would take effect and alter the specified light or lights. Lastly, for the persistence feature, our program was to be able to maintain every user data even after they have logged out of our website. Fortunately, our data persistence only needed to maintain the user information as the information regarding the lights, groups and schedule could be obtained from the bridge (in the emulator) when necessary. As well, the program should be able to handle and respond to errors in a light state change (e.g. a light color change is made but then the user decides to remove a lightbulb from the bridge on the site). Finally, there was the option to use file storage or Wt database services to control and maintain user information, and for simplicity we opted to use the file storage method as this was a small scale and would only be utilized by the members of the group and the TA or professor Katchabaw.

We also had to consider some additional features for an added touch, and we decided to add the following as possible extras; repeating events, location of the bridge, and specific effects on single or multiple groups of lights. For repeating events, we wanted to add the ability for the user to set an event to repeat itself at the same time, once or multiple times. The location of the bridge feature was more about the exact location of the bridge the user would be interacting with, as a way of maintaining general order of the bridges they would be using in their home and modifying through our site. And for the specific effects settings, we were aiming to give control to their user to be able to change lightbulbs in a bridge as individuals, or an assigned group.

**Key accomplishments**

The group, worked well together. Being computer science majors, we had some general experience with object oriented programming (from 2212 course) as well as hobby and summer time projects in our spare time. Being given the liberty to pick our groups gave us the chance to form with like minded people as well as group with those of different skill levels. While working together, we weren’t all able to be physically present 100% of the time, so we relied on Slack as a common tool to maintain communication and give updates in the progress, so the other members would be aware what tasks were near completion and what other ones required attention, testing or revising. Working with C++ was a good experience for our group, as we had all had prior exposure to it through first and second year courses requiring C++ language individual programming.

Another aspect that worked well for the group was acting as an equal collective, with no appointed or self enrolled leader. This gave everyone more equal footing and allowed for discussions to be less pressured to speak about their ideas or suggestions, giving the opportunity for group members to collaborate more with each others code more understandably. In part with this collective, we allowed each member to undertake a role in a section of the project so as to help standardize the coding practice, use of functions and even comments. To maintain consistency in coding and commenting to help retain a general coding pattern and assurance that the comments stayed within a concise and consistent form that a programmer could view and understand fully.

Wt was to be our website backbone, what would allow the user to interact with our code and ultimately, their own at home bridges and control their lightbulbs how they saw fit. Moreover, we were new to the use of it, and had only some basic understanding of how it functioned alongside C++, but our use of one helpful tool gave us the opportunity to develop a better understanding of this. One of our team members recommended we try to learn through the Wt documentation page, and while it was time consuming, it allowed us to get a better grip on the tools Wt had at its disposal and how we would need to code them on our files in order to make them work properly. We also had members of the group view the Hue light api documentation, which proved to be especially useful when working on the third deliverable of the assignment, as we began facing problems. Another useful tool we applied was our make file, as it gave us a simple and generic way of compiling and preparing our program to run and test on out individual computers, so we would be able to see the projects stability and cross system compatibility. As well, whenever we added new files, we simply added a few lines of code to the make file and it would automatically apply a new compiled file that would include the new class.

As a group, we made sure that ideas about the project and its design were agreed by all members, so we could work the most effectively and with enthusiasm. For example, for the second stage of the project, we decided to tackle the log in, account management and data persistence parts of our website, as we all admitted that the other features would require more dedication and to properly make those features would have caused delays for developing other features, and could result in a late submission and cost us our grading on that part of the assignment. We also decided against overinvesting time and manpower into the CSS portion of the site until the last part of the assignment, as we wanted to ensure that the features and core components of our site would be operational prior to creating an appealing and interactive outlook for potential users. This strategy worked to our favor, as nearing the final submission date, we required extra time to work on the features and by minimizing how much time we spent on the CSS we were able to complete nearly all features and create our functioning website.

In summary, we were a good team. We took time and made sure each person took part in the project, worked on specific role, but had other members be knowledgeable of various aspects of the project, so as to provide added support and help with the flow and completion of the websites features for their given due date. While we did not completely finish and include all we wanted, we made sure the site ran all the features we added to it, the code was written well, organized and properly and consistently documented, and made sure to have all members voice their ideas and concerns. In the end, our perseverance was the core factor to the overall completeness of the project. And while there were some things we could not reach, we all ended up satisfied with our overall performance and final product.

**Key problem areas**

Every great goal comes with great obstacles that must be overcome to reach the finish line, and in our case, get the final product. Our group experienced different issues throughout the development, testing and even the final phases.

In the beginning for the initial design, though we planned the UML of our project with care and detail, we paid more attention to coherency in our class hierarchy, and overlooked how to undertake those components and implement them properly using Wt. This became more apparent when we began coding, with the limited experience in Wt, we were sometimes coding in a very complicated manner, and whenever we tried introducing new features to the program, our previous code would create conflicts. Some of these resulted in rendering problems and errors in parts we considered stable. In turn, we had to spend more time to read over the Wt and Hue documentations, redo the code and reimplement into our projects so that the features would collaborate rather than crash on each other. We also did try having meetings to better order ourselves and the tasks for each deliverable, but due to different school schedules, work, or personal emergencies, we end up with missing members and had to relay discussed material with other members through slack, using up more time.

We did attempt to implement strategies to structure and organize our project and our team. For example, we did have and set time to stage our project to undergo testing for prior to pushing or merging, or even submitting but due to the size of the project and the overlapping dependencies between each feature in order to function, we sometimes had to delay those testing times until the missing dependency was resolved. Sometimes we had some rudimentary mock code to suffice for the missing dependency, but without the proper testing we had urgency to return to that part of the program and spend more time to make sure it was stable and that it would run properly.

More specifically, we faced an issue with error responses from HTTP calls. We would use JSON tools, in theory, to analyze the returning JSON body, as success or error. But for some reason, the error message would not be received by the default Json message. Another obstacle we had was that since we were still fairly new to Wt signaling and the Http client, we could not immediately render a message sent by the Hue API and had to resort to manually refreshing the page to view the changes

Multiple users pushing and pulling through git with a team of five meant that we had to be very coordinated in how we made and pushed each update for new versions, as to prevent conflicts in merging on other user computers and their own functions created in different class files. We would sometimes stumble and accidentally push to the wrong branch or at the same time as another member, so we would need to revert changes on bitbucket, while the members with affected files would need to revise and correct their code before pushing again. To correct this error, we would message on slack or if we were present in the same room we would warn about the push, then the other members would ask if they had tested yet; depending on the answer we would allow the member to proceed with the push. This required more attention to detail on who made the more recent pushes and which member was behind on the branch development, but we reduced program issues and spent the time more efficiently in developing the project.

Though we all agreed to work together and focus the Wt network calls and return features near the end of the final part of the project, we did not take to account the difficulty and complexity of programming that section. having a large GUI class for example and not being able to divide it into subclasses to be called separately made it difficult to debug well without spending a great deal of time on it. As well, we were unaware of the asynchronous nature of the Wt client, and as such designed our bridge controller class, we had them send an HTTP call and receive a response, but we usually received the value after the function executed and could not assign the response to return string, which meant we needed to return an empty string.

Depending on the shortcoming, it would either be caught early on or sometimes would build up until it became big and problematic or just not possible to handle as time became a bigger factor and functionality became a higher priority. One of these was our GUI class and its overall size; we ended up with a big and messy class. Though it would have had multiple alternatives into downsizing it and optimizing this into multiple class files to for example force Wt to render at command, this was not possible we did not have time allotted to correct this. Another one was how much time we spent on the HTTP client. Once we had deciphered how it works, we were forced to re-write the old bridge controller version to a proper format, but we were low on time by then, so final testing and few user stories could not be completed, and we could only do rough implementations of select features we could work on with whatever time remained.

Despite these issues, did our best to resolve them. Our main solution source was brute force programming, a crude but eventually effective way to remedy some of our problems. Even though it was not a very advisable method, it did give us the opportunity to better understand Wt through their documentations, as well as trial and error practice. What was important was that in the large scope of the project, that at its core we wanted to project to function and undergo Wt Resful API communication with A Hue Lights emulator on a user account base website, and through our efforts we had gotten very close to that goal despite the numerous struggles and setbacks.

**Lessons learned**

From every experience, new lessons are learned and saved for future endeavours of similar calibre. Some things we thought we could reimplement in a similar way, while others we all felt best to leave behind and remind ourselves that other solutions exist instead of some we chose.

A lesson we all agree to remember to implement early on in projects like this is familiarity research. We were confident with C++, as we all had used it to a certain level, but problems stemmed from our lack of knowledge and implementation of Wt, and when we began coding we had a slow incline to our deliverables. We did eventually read and study the documentation of it online, though we all acknowledge that if we had taken more time at the start of the semester to view, interact and test Wt’s features, many long nights could have been prevented.

Another lesson we learned was to be more rigid with design patterns while working on projects like this. We learned these techniques over the course of the program, and understandably could be able to implement them so effectively while we were in the middle of our project development, but for further project in team based environments, we could very likely see ourselves using these methods to help organize and effectively complete tasks in the allotted times.

While we did have a group with members who had abilities and skills that were used well throughout the project, we sometimes felt as though we could have been more organized and distributed in our roles. We sometimes had moments where team mates could not contribute to the project because they had not done work into that area of the project before, and more time would have to be spent teaching the other member about that project segment, so some work became isolated to one or two members. For future projects, we will keep in mind the use of design patterns to help maintain a steady workflow among group members, so all members can be actively working on different areas of the project simultaneously we can complete the task on time.

In part with the use of design patterns, we also thought it would be a good idea that in future assignments, the groups would have an equal meeting time, where all members could gather and overview the work done during that week or span of days. As well, the time could be used to help other members with their struggles in certain parts of their code, as well as dedicate time to cross train the team to other features so they may be knowledgeable in another area and if issues were to arise, the team members who completed their tasks could more actively contribute with the more complicated areas and help resolve development problems faster.

Overall, this was an enjoyable project. We liked the fact that it allowed us to implement our skills and experience in computer science and programming towards a real-life application. In most of our courses, only theory or brief implementation in coding acted as our experience in Computer Science. This was more of a practical and even career oriented project. It gave us a drive to more actively pursue this assignment and complete it to the best of our abilities and even strive to learn more about the tools we were given and how to improve in our implementation of them.

Lastly, many of us felt that projects like these should be more widely used in the department. It gives the students that have not had the opportunity to work for a computer or programming based company to work and develop a project with applications that can actually have an impact on a person’s life, similar to that of a short term internship. We would like to work with a similar project again in the future, but with the opportunity to utilize tools and applications after taking more time to understand their workings and functions in detail.