Fontys University of Applied Sciences

Software Semester 1

BRiCkeD

Figure 1: Our Team Logo

D. Ratushniuk (leader)	3874362
C. Polder	3830306
R. Arotăriţei	3767086
B. Forslund	3849791

Eindhoven, January 17, 2020

Contents

1 Introduction **3**

2 Background **4**

3 Problem statements **5**

4 Process & Results **6**
4.1 Beatrice 6
4.2 Dima 6
4.3 Robert 7
4.4 Colin 7

5 Evaluation & Reflection **9**
5.1 Beatrice 9
5.2 Dima 9
5.3 Robert 9
5.4 Colin 9

6 Conclusion & Recommendation **11**

7 References **12**

1 Introduction

In this report we will discuss our application made for Student Housing B.V.. The reason for this application is that the company has been receiving complaints from their clients and they would like to have a software solution to reduce the amount of issues. For this application we choose to use Microsoft Visual Studio 10, Arduino Uno/Genuino and Raspberry Pi to assist Student Housing B.v. For the background we have used the theory classes as given in the course P-CB-S-CMK and P-CB-ADO-CMK of Fontys school of ICT English Stream in Eindhoven as knowledge. Some members in the team had previous knowledge which has made it more convenient. We found out that the problems the company is facing are that appointed people do not clean the shared responsibilities, common groceries are not done or paid and garbage disposal is not done on time. This was solved using different techniques though of by the team members.

We are also going to bring up what every team-member was responsible for and if there were certain challenges regarding their part. Every team-member then explains in total how they think the project went and to finish of the report we are going to talk about recommendations for next projects.

2 Background

For this project we have used a lot of old and new techniques. Every member of the team has used in their part theory from the software course along with some addition theory.

The team has decided to create a client-server application[1], so that each tenant in the house would be able to receive their own version of the application. For client part the group used a Windows Forms[2] with a Tab Control[3] component as a main one. To show information for the user the team used mainly two components: User Control[4] and Table Layout[5] with panels.

The server was made using theory which was not covered during the software course. It is based on a TCP[6] connection, using a UDP[7] auto-discovery[8] to not make a connection to the server a hard thing for a user. Data is saved on the server using files in JSON[9] format. After server was completed, it was moved on the Raspberry Pi[10].

3 Problem statements

Student Housing B.V. is a Dutch housing company which owns different building where students can stay during their study in the Netherlands. For some time, they now receive complaints from their clients related to different problems in the houses. As an extra service Student Housing B.V. wants to offer their clients a software solution to better arrange day-to-day situation.

Currently, the company needs to go to every house and ask every student if everything is fine and if they need to improve something, and the students oftentimes do not do their assigned chores, but they want these problems solved. That is the main reason why Student Housing B.V. wants an IT solution.

Student Housing B.V. envisions an application where their clients can record and see agreements made between them, but also the possibility to see the house rules and file complaint anonymously.

Every so often Student Housing B.V. will have one of their employees go by the buildings to update the rules and gather any complaints. However, this company is not strict with implementation, moreover, they encourage additional functionalities and suggestions as long as it does not endanger the delivery of the application on time.

4 Process & Results

4.1 Beatrice

Beatrice was responsible for the admin form and the chat feature. The challenge for her was to make the software look like the GUI we agreed on. In Application development course, we had only covered using listboxes so far and in the GUI there was no listbox. She asked my team-leader Dima, and he said there was a control in Visual Studio called table layout and panels. She then researched using google and found out that instead of using a listbox, She can use a table layout, and panels similar to the listbox. Because with table layout, you can add rows like a spreadsheet. Inside those rows, you can add buttons and label, making it look like the GUI.

Beatrice figured out how to code the text/labels and the buttons, and she started out with dummy "rules". When Dima created the server soon after, the classes and the information. It was easy for her to adapt it to use the classes instead of the dummy rules since she had learned classes and objects in the Software lectures.

When Beatrice knew how to add and how to remove rows/objects from the table layout and at the same time remove from the server, the rest was easy. That is because it is the same techniques used, instead of a button as a control, she could change it to a checkbox and add the same remove and add but using different classes. Beatrice was quickly done with mandatory rules, house rules and complaints. The application was not the fastest, but Dima and our teacher rewrote a lot of the code to make it faster. Beatrice made the base, and then we updated it together.

When she was done with all, Beatrice wanted to do something more, which is why we came up with the chat feature. She made it from scratch on both the student form and admin form. This time, instead of Dima making all connections with the server, Beatrice decided she might be able to to make the objects, classes and everything related to the server. She did not understand at first how the server worked at all, since she have no experience before, and that was a huge challenge. Beatrice saw a pattern of how he did with the previous classes, like with mandatory rules and house rules. She then decided to try it out and to her surprise, it actually worked. Beatrice was done, but it was not looking it's best, so she asked Dima if he wanted to do the GUI and fix some minor changes and now it's looking very good.

The final result of both these features is good. Dima helped and updated her version and thus, making it way faster and a better-looking GUI.

4.2 Dima

Dima was responsible for server creation, connecting windows form with the server and improving the performance of the app. In this project, our team would like to give every tenant a version of our app that is why a multi-client server was required. The challenging part here was to create this multi-client server. For this, Dima has decided to use a console application[11]. He knew how to implement a one-client TCP server and using this knowledge, he has started attempting to create a multiclient version.

Google helped to figure out how to wait for a connection asynchronously[12], this is a necessary thing for the server. After that, there was a need to implement a constant receiving of information from the clients. It was not a big problem; however, it took some time to implement. Next, auto-discovery was added to the server. This feature is implemented using a UDP broadcast on 50 different ports. Finally, there were only two things to add, reading information and handling received information. Reading was not a hard thing. For handling, to standardize sent information, a little class called "Server Package" was created, so that server and client knew what to do with received data. With that server was finished.

After finishing the server, there was a need to provide the team with an easy method to communicate with the server from the client-side. Work on the server connection class has begun. This class needed to provide data sending, receiving and handling. These things were already implemented on the server-side, so after a little adjustment, everything was done.

When the server and server connection class were implemented, Dima started improving the design of the application. This was done to make it re-sizable and responsible. For that, some changes on the server side were also required, so that it could react faster on incoming messages.

After everything was finished, Dima came up with an idea to move server on the Raspberry Pi. Moving it on the Raspberry was not hard because initially server was made using .NET Core[13], that is why it could be run on the Linux system. On Raspberry Pi server was very slow, thus work on improving the server has started one more time. During this improvement, one big issue regarding reading data on the client-side has been discovered. TCP data from Raspberry was sent in small packages which were not read properly. On the Internet there was very little information about this problem, however, after some time a solution which could solve this problem was found.

4.3 Robert

Together with Colin, Robert started creating the GUI and Design of the application. They decided on a colour palette and shapes for the buttons displayed and worked on the application's appearance. After the User Interface was finished, the student's side of the application needed to be implemented.

Besides seeing his schedule and the mandatory and house rules a student can file complaints and propose new house rules. A complaint ends up in a list visible only to the company's employee, while a proposed house rule goes to the Notifications tab in the student's side of the application. A proposed house rule becomes an actual house rule when more than half of the students living in the house have accepted said rule. If all students reject a proposed rule, including the person who created it, the proposed house rule is deleted.

4.4 Colin

At the start of the project Colin helped Robert with some of the designs. Afterwards he was responsible for the making of the schedule. When that was all done Colin was the one in charge about the security of the password storage. This meant that he would go an build in some encryption and decryption. This also meant that the app would get a change password setting and an add new user setting. This was created together with Dima, who made the server connections for these implementations. Colin also added a simple Arduino Script to read the temperature in the room. At the end, when the report had to be made, Colin was in charge of leading the report and making it look like a professional report, as he had done this before.

The working of the schedule came with some issues at the start. First off, the schedules were not displaying correctly, so that had to be fixed. Afterwards, we noticed that the scheduled items where the next person in line is not the logged in student would not be displayed. This also had to be fixed. Afterwards, the days until the task had to be finished was not displaying correctly. This also had to be fixed.

The encryption[14] went quite smoothly, but with the implementation a problem arose, which was fixed by Dima. During this time Colin was working on the report, including checking other people their work.

5 Evaluation & Reflection

5.1 Beatrice

I thought the project went very smooth. On the first day we found out about the project, we sat down and discussed how everything should be and I believe that helped everyone get a clear image. We also divided the work, set deadlines and gave Dima the role of group leader. Throughout working on the project, the project was structured, everyone worked on their part and we got done before the deadlines. My previous group was lacking all these things.

For the next project, I will plan and structure again in the beginning. Having a plan and split responsibilities are something I love to work with. I do not think I would do anything differently since I believe everything went excellent now. I am proud how the application turned out, I like our end product. I am also proud of my good teammates, they are dedicated like me and we all supported and helped each other.

5.2 Dima

From the start, the whole project went well. During the first meeting, our team has decided which functionality we would like to implement, divided the work and agreed on deadlines. Everyone in the team was working on the assigned part. Due to that, our team managed to finish everything before the deadline.

In the next project, I would like to stick to the division of the project parts. It helps a lot because everyone knows what he or she is responsible for. During the next project, I would like to meet with my team more often in real life, to make some decisions.

To sum up, I am proud of our teamwork. The application our team managed to create looks good and works fine.

5.3 Robert

The project went well from the beginning. We discussed how we should approach the problems we faced and came up with proper solutions. The responsibility was divided equally amongst us and everyone stuck to the deadlines. We worked well together and the project we envisioned was done on time. That being said, we did not communicate with each other as much as we could have. The lack of communication slowed down our progress and is a flaw we will have to work on in our future projects.

5.4 Colin

The project went well from the start. We started working and set reasonable but early deadlines. This meant that we were done with the base product at a very early stage. Meaning we could spend the extra time on additions. We could work well together and our communications went smooth. We didn't take any minutes because our meeting were usually intertwined with working and in between lessons. This made it hard to take minutes for everything that was being made. We could have been harder on our deadlines, because these were sometimes not met (although I must admit I was mostly the one that didn't meet the deadline with a few hours).

We could have improved by asking for more feedback from others, including what they would think would be a great addition for the product.

Over all, I am quite happy with our product, but improvements can always be made.

6 Conclusion & Recommendation

In the end, a product was made using the problem statements. The product included all the required functions, and also included some extra functionalities the group decided would be good to have in the application. The teamwork went well and the group could get our things done on time. The teamwork of the group and the leadership that was shown resulted in the success in making this product. The group held meetings regularly, multiple times a week, as to keep up to date with all the changes the others made. All possible additions were thought out by the entire team, and was consequentially implemented by an assigned member of the group.

Some things we could do to improve would be by asking feedback from users, so in this case the students, ask for feedback from the company, and implement this feedback. We would be able to do this using forms, interviews and/or observations.

Another things that can be done future projects is implementing a server that is not local, but uses the internet. Therefore, it would be available at all times everywhere, and can also be accessed by the company so they would not have to send an employee to each house.

Besides this, a database would be a great addition to a future project relating to this. A database can help a lot of the problems with saving information. Currently, everything is saved in a JSON file, and to make a database with references to each table would create referencing for example a student a lot easier.

At last, a good method to work with objects in a group would be to make a different object for each person, and make each person work in their own classes. This way, the source code will be a lot less messy and a lot better to work with using Git.

7 References

1. Client-server
Client–server model. (2020, January 11). Retrieved from
https://en.wikipedia.org/wiki/Client%E2%80%93server$_{m}odel?oldformat = true$

2. Windows Forms
Windows Forms. (2019, November 17). Retrieved from
https://en.wikipedia.org/wiki/Windows$_{F}orms?oldformat = true$

3. TabControl Class
Dotnet-Bot. (n.d.). TabControl Class (System.Windows.Forms). Retrieved from
https://docs.microsoft.com/en-
us/dotnet/api/system.windows.forms.tabcontrol?view=netframework-4.8

4. User Control Class
Dotnet-Bot. (n.d.). UserControl Class (System.Windows.Forms). Retrieved from
https://docs.microsoft.com/en-
us/dotnet/api/system.windows.forms.usercontrol?view=netframework-4.8

5. TableLayoutPanel Class
Dotnet-Bot. (n.d.). TableLayoutPanel Class (System.Windows.Forms). Retrieved from
https://docs.microsoft.com/en-
us/dotnet/api/system.windows.forms.tablelayoutpanel?view=netframework-4.8

6. TcpListener Class
Karelz. (n.d.). TcpListener Class (System.Net.Sockets). Retrieved from
https://docs.microsoft.com/en-
us/dotnet/api/system.net.sockets.tcplistener?view=netframework-4.8

7. UdpClient Class
Karelz. (n.d.). UdpClient Class (System.Net.Sockets). Retrieved from
https://docs.microsoft.com/en-
us/dotnet/api/system.net.sockets.udpclient?view=netframework-4.8

8. Automatic Discovery
Automatic server discovery. (2017, April 7). Retrieved from
https://en.wikipedia.org/wiki/Automatic$_{s}erver_{d}iscovery?oldformat = true$

9. JSON Format
Introducing JSON. (n.d.). Retrieved from https://www.json.org/json-en.html

10. Raspberry Pi 3 Model B
Raspberry Pi 3 Model B – Raspberry Pi. (n.d.). Retrieved from
https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

11. C# Console Application
BillWagner. (n.d.). Create a Hello World application with .NET Core in Visual Studio -
.NET Core. Retrieved from https://docs.microsoft.com/en-
us/dotnet/core/tutorials/with-visual-studio?tabs=csharp

12. TcpListener.AcceptTcpClientAsync Method
Karelz. (n.d.). TcpListener.AcceptTcpClientAsync Method (System.Net.Sockets). Retrieved from https://docs.microsoft.com/en-us/dotnet/api/system.net.sockets.tcplistener.accepttcpclientasync?view=netframework-4.8

13. .NET Core
.NET Core. (2020, January 15). Retrieved from https://en.wikipedia.org/wiki/.NET$_{C}ore?oldformat = true$

14. Encryption
RichardRichard 4, CraigTPCraigTP 38.1k88 gold badges6464 silver badges9595 bronze badges, A GhazalA Ghazal 1, ja72ja72 21.4k44 gold badges5757 silver badges112112 bronze badges, Sergey KolodiySergey Kolodiy 5, UlisesUlises 12.7k55 gold badges2929 silver badges4545 bronze badges, … Mike CalvertMike Calvert 10366 bronze badges. (1962, April 1). Encrypting Decrypting a String in C. Retrieved from https://stackoverflow.com/questions/10168240/encrypting-decrypting-a-string-in-c-sharp