Process Report

Team Hyena Crossing

Version 1.3

Alina Baciu

Yousef Abu Zahra

Bozhidar Bonev

Borislav Pavlov

Ghazi Abdul Fattah

Table of Contents

Version History

- **Mar 16, 2020:** Created the document, added work division.
- **Apr 6, 2020:** Updated the work division, added iteration 1 reflections.
- **May 25, 2020:** Updated the work division, added iteration 2 reflections.
- **Jun 12, 2020:** Final update to the work division, added iteration 3 reflections, combined everything, added every other segment for the processes.

Introduction

Our group consists of five members: Yousef Abu Zahra, Alina Baciu, Borislav Pavlov, Bozhidar Bonev and Ghazi Abdul Fattah.

The purpose of this project is to deliver a fully functional traffic lights simulator as requested by our client, Andrius Kuprys.

The following document shows a series of actions which are carried out each week in order to achieve the particular wanted results for the Procp project during the study weeks, together with the problems, challenges and decisions made during that period.

Another part of this document is represented by the „Personal Reflection" chapter, where we individually describe our experience gained while working on this project and „Extra Achieved & Completed" chapter where we describe what tasks we successfully completed.

Kick off phase

Tasks list for this phase

In this phase, we had to deliver the following:

- Proposal for an application.
- Draft version of the project plan.
- Final version of the project plan.
- Concept version of the user requirements document.

Challenges

As seen from the previous list, we had multiple questions we needed to answer before we could start implementing:

- Which topic should we choose for the project?
- How to decide which features are we going to include in the application?
- What type of algorithm are we going to implement in the application?

All these questions were challenges we needed to overcome, which we did with our research.

Research and research methods

We used DOT framework methods in order to find possible solutions to our problems, some of the methods we used were:

Available product analysis

We investigated already finished projects, and we gathered a good idea on what to expect.

Usability testing

We tried multiple styling libraries in order to find one which provides the best interface.

Document analysis

We read the "PROCP – Workbook" multiple times, which helped us understand what's expected from us.

Brainstorm

We had multiple meetings to come up with ideas for features for our applications.

Requirements prioritization

We had a clear list of what essential things we needed to do first, and what comes next.

Decisions, justifications and expected effects:

Based on the research we did; we made the following decisions in this iteration:

1. Changed the topic idea from "Airport" to "City planner"

Justification: We initially choose the airport luggage simulator as our idea, however after doing some research, and some brainstorming, we realized that we preferred the "City Planner" idea; because we could relate to it a lot easier, and we did not have to investigate a lot on how they work (Since some of us already have some knowledge on the subject).

Expected effects: The project will be about a topic that's more interesting to us.

2. Chose Dijkstra and Ford-fulkerson as our algorithms to implemented

Justification: After doing our research on the algorithms, we realized we could utilize two types of algorithms:

a. Shortest path algorithm: to find the shortest path between two points.
b. Maximum-flow algorithm: to find the maximum flow of cars a road layout can handle.

Upon further research, it turned out that despite Dijkstra's algorithm being greedy (requires more resources) it performs the best. The resources part was not relevant to our project scale (Dijkstra could find the shortest path between 50.000 nodes within milliseconds, the most our application has is 1200 nodes) which is why we chose it for the shortest path.
For Maximum flow, there are only two used algorithms. Ford-fulkerson was the one we learned previously in our Fontys courses, so we went for that one.

Expected effects: N/A.

3. Made a MOSCOW list and put the "Must" features in the first iteration, "Should" in the second and "Could" in the third.

Justification: We wanted to prioritize the most important features first, so that we have a solid core application to build on.

And because we promised a proof of concept at the end of iteration 1, having all the “Must” features by then would make the application a lot more conviving to our client.

Expect effects: Finishing the core of the application early on.

4. Deciding on Windows Forms as our platform

Justification: We had Windows Forms, WPF and Unity as our candidates.
Our team lacked experience in Unity and WPF, and we did not have much time to get to learn them. Therefore, we went with Windows Forms.
We did more research on the topic, and it came to our attention that Windows Forms isn’t the ideal platform for visuals-heavy applications, but we made the decision that the time and effort we could spend on learning different platforms would be better spent on improving our application, even if it had performance issues due to Windows Forms not being ideal.

Expected effect: Being able to start working on the application immediately, and to have more time to perfect it.

5. Working in SCRUM framework

Justification: We chose SCRUM for multiple reasons:

1. We needed to work on the application simultaneously and not sequentially, because of the short time given to us.
2. We are inexperienced, meaning a lot of the decisions we make can possibly be wrong. SCRUM allows us to be adaptive and change the scope of our project.
3. SCRUM allows us to release what is important first, meaning we can have a proof of concept a lot earlier than if we worked within a different framework.
4. Our team is small, which is considered ideal for SCRUM development.
5. Upon doing more research, we realized that SCRUM clients are often more satisfied, because they get usable portions of the applications quickly.

Expected effect: Application deliver on time, flexibility, finishihg what’s important first and client’s satisfaction.

Work division

Week 1

Nr.	Task	Total Time	Yousef	Alina	Bozhidar	Borislav	Ghazi	Task Completed
1.	Proposal for the project	2h	25%	25%	(Wasn’t in the team at the time)	25%	25%	100%
2.	Concept version of project plan	3h						
	Total time	5 h	1.25 h	1.25 h		1.25 h	1.25 h	

Week 2

Nr.	Task	Total Time	Yousef	Alina	Bozhidar	Borislav	Ghazi	Task Completed
1.	Research the to-be implemented application	1h	20%	20%	20%	20%	20%	100%
2.	Update Concept version of Project Plan	1h	20%	20%	20%	20%	20%	
3.	Create concept version of URS – Use Cases:							
a.	Start simulation	1h			100%			100%
b.	Stop simulation	1h			100%			
c.	Pause simulation	1h			100%			
d.	Create custom layout	1h	100%					
e.	Reset the map layout	1h	100%					
f.	Add/Remove one-way roads	2h					100%	
g.	Add/Remove two-way roads	1,5h					100%	
h.	Generate report	1h	100%					
i.	Save a file	1h		100%				
j.	Save as file	1h		100%				
k.	Specify number of cars generated	1h		100%				
l.	Add/Remove intersections	1,5h				100%		
m.	Add/Remove roads with traffic lights & Zebra Crossing	1,5h				100%		
4.	Create Meeting Minutes document	1h		100%				
5.	Create Meeting Agenda	30m	100%					
	Total time		3.5 h	4.2 h	3.3 h	3.3 h	3.8 h	

Reflections on this phase

- Going with Windows Forms was a bad idea, because later we realized that it is not a good platform for visuals heavy applications. The correct call would have been to go with either Unity project or Windows Presentation Foundation (WPF).

- Deciding to go with “City Planner” simulation was a correct call; other options would have been out of our domain and would have caused more complications down the line.
- Scrum made planning our work a lot easier, because it simplified all the planning aspects of working in a team.
- Putting our most important features in our first iteration was a good idea, because we managed to finish all the promised “Must” featured from our list early on.
- Including that many road types was a mistake, we thought it would bring value at the time but as it turns out later: it did not.

Initial phase

Tasks list for this phase

In this phase, we had to deliver the following:

- Concept version of iteration 1 plan.
- URS
- Work division report
- Final version of iteration 1 plan

Challenges

During this phase, we did not have many challenges.
Most of the work done here was already decided in the Project plan. All we needed to do was to provide the expected documentation by our client.
We only had to make two important decisions:

- How are we going to divide the work?
- How are we going to design the layout?

Research and research methods

We used DOT framework methods in order to find possible solutions to our problems, some of the methods we used were:

Decisions, justifications and expected effects:

Based on the research we did; we made the following decisions in this iteration:

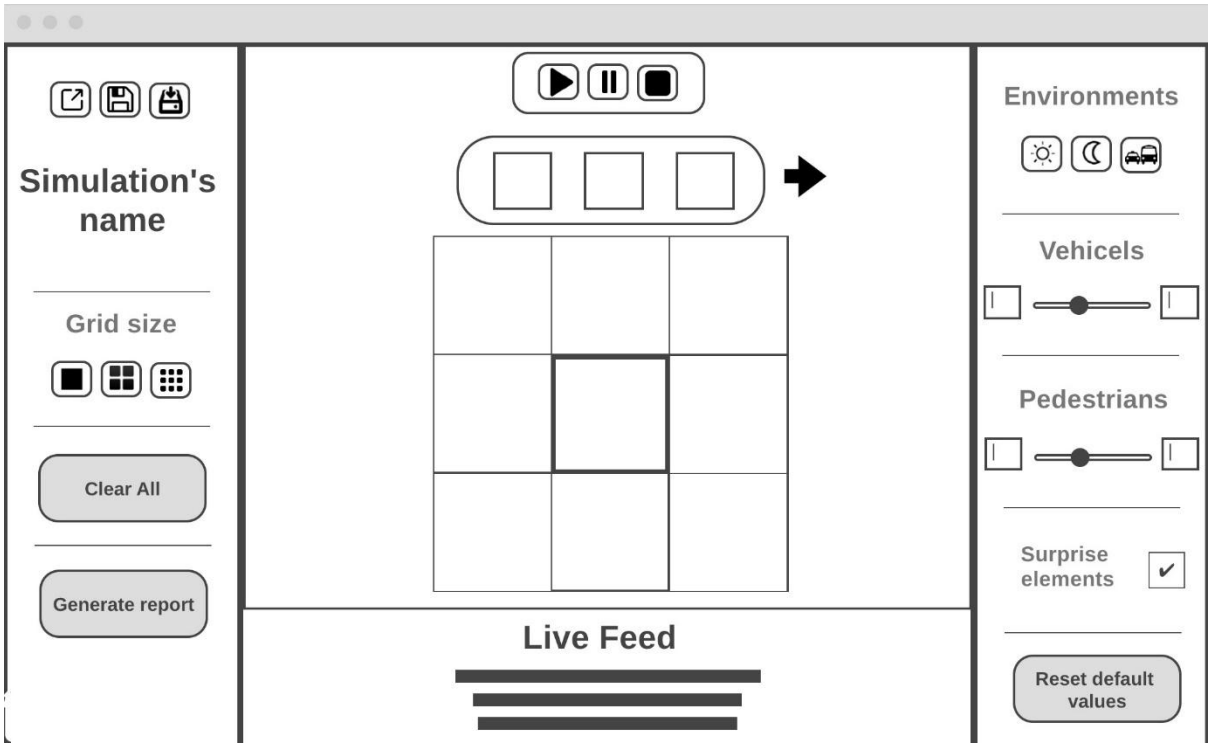
1. Splitting the tasks into five independent, equal groups in every phase:

Justification: This gave us multiple advantages:

- We avoided having any member having too little or too much to do.
- Because the tasks were independent for the most part, we avoided dependency, allowing everyone to work simultaneously.
- Everyone knows exactly what everyone else is doing and would know who to ask in case they need help.

Expected effects: The advantages listed above.

1. Creating our GUI design (Concept shown in the image below)



Justification: We chose to design our GUI this way for these reasons:

- Our application is complex, so we must keep the GUI as simple as possible to not intimidate the user, which is why we chose to use common icons instead of text for the majority of the buttons.
- We chose to divide our application buttons in categories while keeping the most essential buttons on top of the screen (which is what the user pays attention to the most).
- We kept consistency in similar components (for example, the pedestrians and vehicles section).
- The user always knows what is going on; indicated by enabling and disabling certain sections during different phases. (For example, the user cannot adjust vehicles count while the simulation is running).

Expected effects: The user interacting with our application seamlessly.

Work division

Week 3

Nr.	Task	Total Time	Yousef	Alina	Bozhidar	Borislav	Ghazi	Task Completed
1.	Discuss project plan	1h	20%	20%	20%	20%	20%	
2.	Concept plan for iteration 1	3h					100%	

3.	Update Project Plan & URS	1h	20%	20%	20%	20%	20%	100%
4.	Use Case Diagram	2h				100%		
5.	User Interface Sketches	7h	60%	40%				
6.	Create Meeting Minutes document	1h		100%				
7.	Create Meeting Agenda	30 min	100%					
	Total time		4.5 h	4.1 h	30 m	2.3 h	3.2 h	

Nr.	**Task**	**Total Time**	**Yousef**	**Alina**	**Bozhidar**	**Borislav**	**Ghazi**	**Task Completed**
1.	Discuss URS & plan for iteration 1	2h	20%	20%	20%	20%	20%	100%
2.	Updated URS & plan for iteration 1	1,5h	20%	20%	20%	20%	20%	
3.	Create work division report	1h		100%				
4.	Create meeting minutes	1h		100%				
5.	Create Meeting Agenda	30 min	100%					
	Total time		1.1 h	2.4h	40m	40m	40m	

Week 4

Nr.	**Task**	**Total Time**	**Yousef**	**Alina**	**Bozhidar**	**Borislav**	**Ghazi**
1.	Discuss URS & plan for iteration 1	2h	100%	90%	90%	95%	90%
2.	Updated URS & plan for iteration 1	1,5h	100%	100%	100%	100%	100%
3.	Create work division report	1h		100%			
4.	Create meeting minutes	1h		100%			
5.	Create Meeting Agenda	30 min	100%				
	Total time		4h	5.5h	3.5h	3.5h	3.5h

Reflections on this phase

- Some of the GUI decisions we took were not ideal, for example, the trackbar for pedestrians and vehicles. It would have been much better to use a simple textbox.
- The way we split the workload worked for us, but only because we kept good communication. If the project was on a bigger scale, or if the team was larger, we would have had a lot of conflicts.

Iteration 1

Tasks list for this phase

According to our project plan, we had to deliver the following:

Features

- Start simulation.
- Stop simulation.
- Pause simulation.
- Reset the map layout.
- Add/Remove one-way roads.
- Add/Remove two-way roads.
- Add/Remove intersections.
- Add/Remove traffic lights.
- Save the simulation as a file.
- Specify the number of cars generated.
- Generate reports with information about current simulation.

Documentation:

- Iteration 1 User Requirements Specifications.
- Iteration 2 Plan.
- Proof of concept.
- Source code for proof of concept.
- Updated version of the work division report.
- UML class diagrams.
- Non-trivial sequence diagrams.

Challenges

As seen from the previous list, we had multiple questions we needed to answer before we could start implementing:

- How are we going to connect roads together?
- How are we going to move the vehicles?
- How are we going to implement the road rules?
- How are we going to allow the user to allow the user to control the flow of the simulation?
- What is the best way to save the simulation (serialization or text-based)?
- How can we load a previously saved simulation?
- What could be useful to include in the report?

All of these questions were challenges we needed to overcome, which we did with our research.

Research and research methods

We used DOT framework methods in order to find possible solutions to our problems, some of the methods we used were:

Brainstorm

We needed to find new ideas on how we can do our application, which is why we came together and made a list of possible implementations of the application.

Available Product Analysis

We did this type of research because we realized that there are previous projects that had delivered similar applications to ours, and we could make use of what they've already done, instead of "reinventing the wheel".

Design Pattern Research

We came to realize that some problems in our application (e.g. communication between vehicles) could be solved by an already established design pattern(s).

Before we could start implementing we needed to design the application, which is why we had to use this research method.

Decisions, justifications and expected effects:

Based on the research we did; we made the following decisions in this iteration:

1. Implemented the "Composite" design pattern, but scrapped it later:

Justification: We initially found the composite fitting for our application, because it's a tree-based pattern (like our application, we have classes such as "Simulation – Grid – Cell – Road – Lane" in a hierarchy). We scrapped this idea later on because of two reasons: Extra unneeded complexity in our UML design and no real added value, despite it fitting the "mold" of what we're trying to do.

Expected effects: Less complexity in our design.

2. Implemented the "Mediator" design pattern:

Justification: After researching design patterns and knowing elements in our application will have to interact with each other (Cars and other cars, cars and traffic lights, pedestrians and cars), we realized that the "Mediator" pattern is a perfect solution.

One of the advantages that the "Mediator" pattern solve is that we don't need to implement the communication logic inside the classes themselves, because it would make them a lot less reusable, it encapsulates all the communication within that class.

Expected effects: Reduced coupling between components, making said components more reusable, having the communication code contained within one class makes it a lot easier to maintain.

One expected drawback is that we knew our "Mediator" would evolve into a "God-object", or in other words, an object that does too much. Because in the future iterations, we will have a lot of other dynamic elements using the "Mediator" for communication (such as "Pedestrian" objects).

We made the decision that the advantages outweigh the disadvantages.

3. Completely removed "One-way roads":

Justification: Since the goal of the application is to try simulating real city traffic, and indirectly reduce traffic jams in real life, we looked into real-life traffic planning. Upon doing that, we realized that city planners very rarely make use of one-way roads, because they cause "bottleneck effect" and traffic jams.

The only way it would be effective in a road plan is if the destination of the car moving was clear, and the one-way road is a shortcut to that destination.

Since cars move randomly in our application, one-way roads would only create bottlenecks, because cars will randomly choose to take the one-way roads and might have to wait for the other cars to enter first.

Expected effect: Reduce "artificial" traffic jams during the simulation.

4. Moved "Two-way roads" from "Must" to "Should" (And from iteration 1 to iteration 2)

Justification: Upon researching real-life traffic, we came to realize that intersections are the most important piece of road to implement, because it's where most of the traffic issues (that our application is trying to tackle) are born (traffic jam, accidents...).
Therefore, we decided to have intersections to be the first road option in our application.
A city that has 2-way roads that do not intersect is not realistic and wouldn't serve the purpose of our application (Simulating real-life traffic).

Expected effects: More time to perfect intersections, leading to the application delivered at the end of iteration 1 to be more realistic, and closer to the end goal (Reducing traffic issues in real life), because intersections capture most of these issues.

5. Built multiple demo applications to get familiar with our design

Justification: After we reached a conclusion on how our application is going to be designed, we decided to build few demos for two reasons:

1. To have a soft "Proof of concept".
2. To get familiar with different components of the application.

We eventually decided on one version, and we built our design upon that prototype.

Expected effects: All the members getting familiar with the different aspects of the application, testing whether our concept works or not, understanding the advantages and disadvantages of our design.

6. Drag and drop is how the user is going to place road pieces on the grid

Justification: The interaction with the GUI seemed more natural to the user this way; The user could see all the road types available immediately. They would also see the empty grid and would get the feeling of a "Puzzle game".
We didn't want to clutter the screen with instructions either, so we went with the most forward way.

Expected effects: Easy to understand interactions between the user and the application.

Work division

The general task breakdown for this iteration was:

Yousef

- How cars move in lanes, turning points, stopping points.
- Where are cars generated (On the edges of the map).
- Provide live feedback about the cars on the map. (What lane they entered, whether they crash or reach their destination.etc) in the “Live feed” section.
- Design Document.
- Iteration 2 reflections.

Alina

- Name, save and load the simulation.
- Start, pause, stop and restart the simulation.
- Generate a report after the simulation is over.
- Work division report.

Bozhidar

- Design the UI (Windows, buttons).
- Create the layout of each window (as shown in the URS)
- Disable/enable some windows depending on what the application is doing (e.g. the user cannot change the grid size while the application is running).

Borislav and Ghazi (This list of tasks is too dependant and large to be done by a single member)

- Draw & design roads.
- Add/Delete roads.
- Create grids with different sizes
- Connect neighbor roads together.
- Change green light timing
- Reset map layout

Ghazi

- Iteration 2 plan (Minus the reflections).

Borislav

- URS updates.

Week 5

Nr.	Task	Total Time	Yousef	Alina	Bozhidar	Borislav	Ghazi	Task Completed
1.	Discuss future changes	1h	20%	20%	20%	20%	20%	

2.	Update URS	1,5 h	20%	20%	20%	20%	20%	100%
3.	Update work division report	1h		100%				
4.	Create meeting minutes	30 min		100%				
5.	Create Meeting Agenda	30 min	100%					
	Total time		1h	2h	30m	30m	30m	

Week 6

Nr.	Task	Total Time	Yousef	Alina	Bozhidar	Borislav	Ghazi	Task Completed
1.	Search Programming Design Patterns	2h	30%	15%	20%	20%	15%	100%
2.	Create demo application	25h	70%			15%	15%	
3.	Create Concept Version of plan per Iteration 2	2h					100%	
4.	Code							
a.	Cars logic (Mediator included)	8h	100%					70%
b.	Roads/Lanes/ Grid/Cells	10h				50%	50%	20%
c.	Simulation & file options	2h		100%				30%
d.	Design the GUI	1.5h			100%			100%
	Documentation							
5.	Update Work Division Report	1h		100%				100%
6.	URS Update	1h				100%		
7.	Design Document							
8.	Create meeting minutes	1h		100%				
9.	Create Meeting Agenda	30 min	100%					

10.	Concept version of the UML diagram	1.5h	100%					
	Total hours		30h	4.1h	2h	3h	8.5h	

Week 7

Nr.	Task	Total Time	Yousef	Alina	Bozhidar	Borislav	Ghazi	Task Completed
1.	Code							
a.	Cars logic	5h	100%					90%
b.	Roads/Lanes /Grid/Cells	5h				60%	40%	50%
c.	Simulation File options	2h		100%				20%
d.	Design the GUI	1.5h			100%			40%
2.	Documentation							
3.	Update Work Division Report	1 h		100%				100%
4.	URS Update	1 h				100%		
5.	Create Meeting minutes	30 m		100%				
6.	Create Meeting Agenda	30 m	100%					
7.	Final Version of plan for iteration 2	1h					100%	
8.	Iteration 1 reflection	1h	100%					
	Total hours		6.5h	2.5h	1.5h	3.3h	3h	

Week 8

Nr.	Task	Total Time	Yousef	Alina	Bozhidar	Borislav	Ghazi	Task Completed
1.	Code							

	Cars Logic	2h	100%					100%
a.	Roads/Lanes/ Grid/Cells	6h	40%			40%	20%	60%
b.	Simulation File Options	3h	25%	20%		30%	25%	40%
c.	Design the GUI	1h			100%			70%
	Documentation							
5.	Update Work Division Report	1 h		100%				100%
6.	URS Update	30				100%		
7.	Create Meeting minutes	40 m		100%				
8.	Create Meeting Agenda	30 m	100%					
	Total hours		5.6 h	2.2 h	1 h	4 h	2 h	

Week 9

Nr.	**Task**	**Total Time**	**Yousef**	**Alina**	**Bozhidar**	**Borislav**	**Ghazi**	**Task Completed**
1.	Code							
a.	Roads/Lanes/ Grid/Cells	7h	50%				50%	80%
b.	Simulation File Option							
B.1	Save & Save as file Options	3h		100%				90%
B.2	Open file	3h		100%				40%
c.	Design the GUI	2h		30%	70%			100%
	Documentation							
	Class Diagram	3h	100%					80%
5.	Update Work Division Report	1 h		100%				100%
6.	URS Update	30m				100%		
5.	Create Meeting minutes	30 m		100%				
6.	Create Meeting Agenda	30 m	100%					
	Total hours		7.5 h	7.5 h	1.5 h	30 m	4h	

Reflections on this phase

- Using the mediator pattern was a good idea, however one expected drawback is that we cannot multithread it, which affected performance, especially in later iterations.
- Windows Forms is not the ideal platform to code an application like this, because of the bad visual performance (for example: The main thread lags whenever it has to move too many elements, or cars move with “stuttering”).
- Scrapping the one way roads completely was perhaps not the most ideal decision, we could have moved that feature to a “Could” and put in the third iteration.
    - Reason being is that we introduced vehicles that know its destination (emergency vehicles) and one-way roads could have been beneficial in finding the shortest path.
- Regarding saving and loading the simulation: the decision to serialize our objects instead of saving via text seemed correct the, however, in retrospect, saving the simulation values inside a text file and loading it might have required less maintenance in the future (Since we had to serialize everything we needed to save from that point onwards).
- Assigning complete documents to each person was a mistake, it’s better to have each person contribute a part of each document.

Calibration Week

Tasks list for this phase

- Present our application.
- Compare our progress with the other groups.
- Finalize some of the documents.

Work division

Week 10

Nr.	Task	Total Time	Yousef	Alina	Bozhidar	Borislav	Ghazi	Task Completed
1.	**Calibration session**							
		2h	100%					
2.	Code							
a.	Simulation File Options							
A.1	Created Actual Runtime for the application in the Generate Report	2h		100%				100%
3.	Sequence Diagrams							
	Generate Report	2h	100%					
	Specify the numbe r of cars generated	1.5h	100%					
	Add intersection with/without zebra crossing	3h					100%	
	Remove intersections without crossing	2h					100%	100%

	Start Simulation	1.5h			100%			
	Stop Simulation	2 h			100%			
	Save & Save As	2.5 h		100%				
	Pause Simulation	1.5h				100%		
	Reset Map Layout	1.5h				100%		
3.	Update Work Division Report	1 h		100%				
4.	Create Meeting minutes	30 m		100%				
5.	Create Meeting Agenda	30 m	100%					
	Total hours		6h	6h	3.5h	3h	5h	

Reflections on this session

- We realized that our progress with the application is really good: we nearly have all the essential “Must” features, which made us think we have more room to be creative with what we’re doing.
- We realized that we needed more statistics and numbers in our application, which was prevalent in the other group’s application.

Iteration 2

Tasks list for this phase

According to our project plan, we had to deliver the following:

Features

- Add/remove highways
- Add/Remove roundabouts
- Add/Remove non-priority roads
- Add/Remove turn right signals
- Change the traffic light time
- Open saved file
- Add/Remove pedestrians
- Cars follow Ford-Fulkerson algorithm for maximum flow
- Cars follow Dijkstra algorithm for shortest path
- Save Simulation report
- Compare simulation reports

Documentation:

- Iteration 2 User Requirements Specifications.
- Iteration 3 Plan.
- Design Document.
- Prototype.
- Source code for prototype.
- Unit testing and test document for the prototype.
- Updated version of the work division report.

Challenges

As seen from the previous list, we had multiple questions we needed to answer before we could start implementing:

- How are we going to implement the extra roads?
- How to open a previously saved serialized file?
- Are there any other alternative ways to test our app?
- How to perform good unit testing?
- What are the steps to implement the Ford-Fulkerson algorithm?
- What are the steps to implement the Dijkstra algorithm?

All of these questions were challenges we needed to overcome, which we did with our research.

Research and research methods

We used DOT framework methods in order to find possible solutions to our problems, some of the methods we used were:

Expert interview

None of us has done unit testing before, so we had to talk to one of our teachers who guided us through the process.

Best good and bad practices

Additionally, we looked into previously done testing online to figure out the best practices out there when it comes to unit testing.
Using this method, we also came across another type of testing: Manual testing. Which we put to use in this iteration.

Literature study

In order to implement the algorithms, we had to look into their theory. With the help of our previous college courses, and online sources, we managed to get a grasp on how to apply the theories into our application.

Decisions, justifications and expected effects:

Based on the research we did; we made the following decisions in this iteration:

1. Completely removed Non-Priority Roads

Justification: The reason for removing Non-Priority roads is that they would not reduce the traffic in our system at all, which is our goal. We realized that in order for the cars to cross lanes they have to follow the rule for non-priority roads, which say that the car on the right is always with priority. In our opinion the non-priority road will create insanely big traffic jams exactly because of this rule.

Expected effects: Less artificial traffic jams in our application.

2. Completely scrapped highways, turn right signals, bridges and joins.

Justification: We removed the Highway type of road because our system is made for city traffic and there is no sense in adding a highway inside the city. This would be really dangerous for the people in the city, because usually the limit in a highway is around 90 km/h and the limit inside a crowded place is 50 km/h which makes this type of road absolutely incompatible. However, for instance, if we decide to reduce the limit in our system and make it the same as the limit in a crowded place - 50 km/h, this again will create traffic jams and will not have any positive outcome.

Moreover, the team decided that the additional types of roads that were planned for iteration 3 such as bridges and joins should also be removed. Since the programming environment used for the project is Visual Studio and all of the graphic visualization is in 2D, the implementation and the visualization of a bridge or a join cannot happen.

Expected effects: More time to focus on features that actually matter, less artificial traffic jam, more room to add new features which are helpful to the client.

3. Added all the remaining planned iteration 3 features to iteration 2 (par the removed roads)

These features include:

- Add pedestrian road-sensors.
- Add special types of vehicles (Emergency).
- Add Surprise elements (Random crashes).
- Specify numbers of pedestrians.
- Change simulation speed.
- Display tips.

Justification: After removing all the unnecessary roads, we came to realize that iteration 2 and 3 were both very light, which meant that we can combine them both into one iteration, and add new features for iteration 3.
The reason that we didn't keep the iterations the way they are and then added new features in place of the removed roads is that we wanted to stay as close as possible to the features we promised the client, our idea was clear: First we finish we promised, then we add new features.

Expected effects: Delivering more promised features (even the ones in the "Could" category), freeing up time for new features in iteration 3.

4. Emergency vehicles have to follow Dijkstra-algorithm

Justification: This algorithm is for the shortest path. The way Dijkstra algorithm works is:

a. Start by specifying the start node and the end node.
b. From the start node, find all the adjacent nodes.
c. Calculate cost (distance) between those nodes and choose the closest one to start.
d. From the new node, find all the adjacent nodes, pick one that's closest to start.
e. repeat until you find the end node.
f. All the nodes and edges connecting them represent the shortest path.

Dijkstra is resource greedy but is considered the most accurate algorithm for shortest path. We chose it over other available algorithms because the maximal number of nodes in our application is really small, and the calculations for that would take milliseconds at most. Therefore, the performance didn't matter as much and we could make use of the best accuracy possible.

We decided that we could make use of this algorithm in moving emergency vehicles, the start and end nodes (chosen by user) could represent a real life emergency department, and when you plan a city, you will need to make sure that the emergency vehicles take the minimum amount of time to arrive at crashes/accidents.

Expected effects: The user can plan emergency points that spawn emergency vehicles, which move to the crash points with the best accuracy possible which allows the user to compare different settings for the emergency points placements, and find out which one is the best.

5. Implemented Ford-Fulkerson algorithm to find the correct maximal flow of vehicles.

Justification: Explained in the document "Algorithm - Calculating maximum capacity of road network" in the "Design document" folder on our gitlab.

Expected effects: Explained in the document “Algorithm - Calculating maximum capacity of road network” in the “Design document” folder on our gitlab.

6. Saving the report as text as opposed to serialized object

Justification: We wanted the user to be able to view the report from outside the application, and from iteration 1, we realized it’s easier to maintain “Save” features if they’re text based, because we don’t have to serialize the new objects we add to the application.

Expected effects: User is able to view the reports without opening the application.
Application maintenance is easier in the future.

7. Comparing simulation reports moved to iteration 3

Justification: We decided to revamp how comparison between reports works: instead of comparing only one report with the current, the user can add as many reports as he would like to a list (including the current one) and then he can choose how to compare them.
However this implementation might take too much time for this iteration, so we decided to move it to the next one.

Expected effects: More freedom to the user when it comes to comparing simulation results.

8. Split the unit testing between team members

Justification: Initially we had one member doing all the testing, however we realized that it’s a big task, and each team member knows how to test the features they developed the best. Therefore, we split the workload.

Expected effects: Testing done on time and more in-depth.

9. Added night-mode

Justification: This mode is to allow the user to simulate traffic at nighttime (Low traffic), we thought it would be handy to have pre-set environments to compare to in any simulation.

Expected effects: Users can test their road piece in different environments without having to directly change the settings.

10. Pedestrians automatically use road sensors if the road is empty

Justification: This is done to simulate reality, in which you can cross the road if there are no cars nearby, contributes to reducing traffic jams.

Expected effects: Less traffic jams and more realism.

11. Tips-on-hover instead of a dedicated “Help” section

Justification: We chose to display tips as our sort of way to help the user instead of having a complete tutorial section.
We made this decision because it is more GUI friendly and less intimidating.

Expected effects: Less cluttered, complex GUI.

12. introduced weights to Dijkstra algorithm

Justification: After doing unit testing, we realized that the Dijkstra algorithm sometimes chooses a node that is technically the closest to the start, but the actual path to it wouldn't be (because of the way roads are drawn).
To mitigate this, we introduced weights to these types of nodes, making it so that the algorithm would prefer nodes that are theoretically further (in x's and y's) , but visually are closer (on the road pieces themselves), which provides the true shortest path.

Expected effects: Finding the actual, practical shortest path.

Work division

General task division for this iteration is:

Yousef

- Add Surprise elements (such as random crashes, cars not following road rules)
- Add new types of cars (Police cars) that must move to a certain point on the map.
- Police cars must follow Dijkstra algorithm for shortest path.
- Update the class diagram and add short description of the new functions
- Sequence diagram and use cases for:
    - Enable surprise elements
    - Add special cars
    - Remove special cars
    - Generate a report
    - Specify numbers of cars generated

Bozhidar

- Change the speed of the simulation (Speed up the simulation)
- Set a timer on the simulation
- Simulation automatically ends when the timer hits 0.
- Maximum flow mode (Ford-Fulkerson algorithm).
- Display tips (On hover).
- Sequence diagram and use cases for:
    - Add a timer to the simulation
    - Display tips
    - Enable maximum flow mode

- Start the simulation
- Stop the simulation

Ghazi

- Add Pedestrians to roads that have zebra crossings:
- Pedestrians cross the road when the traffic light is green (Sensors).
- Pedestrians move on the sidewalks.
- When the road is empty, pedestrians can use the sensors to turn the lights green.
- Sequence diagram and use cases for:
    - Add pedestrians
    - Remove pedestrians
    - Enable/disable road sensors.
    - Add/Remove intersections without zebra crossings
    - Add/Remove intersections with zebra crossings

Borislav

- Add/Remove Roundabouts
- Add/Remove two-way roads (Vertical and horizontal)
- Add pre-set environments (Night traffic, day traffic).
- Sequence diagram and use cases for:
    - -Add roundabouts
    - -Remove roundabouts
    - -Change simulation speed
    - Pause simulation
    - Reset map layout

Alina

- Unit testing.
- Create a test report.
- Improve application visual performance.
- Sequence diagram and use cases for:
    - Add two-way roads
    - Remove two-way roads
    - Add pre-set environment
    - Save
    - Save as

Week 11

Nr.	Task	Total Time	Yousef	Alina	Bozhidar	Borislav	Ghazi	Task Completed
1.	Code							
a.	Working on Pedestrians	10h					100%	
b.	Working on Surprise Elements	1h	100%					80%
c.	Working on Dijkstra Algorithm	5h	100%					
d.	Working on Open file feature	3h		100%				60%
e.	Working on generated values for report	1.5h		100%				30%
f.	Iteration 3 Plan	1.5h				100%		
	Documentation							
5.	Update Work Division Report	1 h		100%				100%
5.	Create Meeting minutes	30 m		100%				
6.	Create Meeting Agenda	15 m	100%					
	Total hours		6h	5.8h	h	1.5h	10h	

Week 12

Nr.	Task	Total Time	Yousef	Alina	Bozhidar	Borislav	Ghazi	Task Completed
1.	Code							
a.	Working on Dijkstra	10h	100%					
b.	Working on Emergency vehicles	5h	100%					

c.	Working on enable road sensors	10h					100%	
d.	Working on open file	2h		80%		20%		100%
e.	Reimplement the way cars are moving	10h				100%		
f.	Working on Generate Report	3h		100%				
g.	Working on the addition of the new lanes	10h				100%		
h.	Working on Simulation Speed	4h			100%			
	Documentation							
5.	Update Work Division Report	1 h		100%				100%
6.	Working on class diagram	2h	100%					
5.	Create Meeting minutes	30 m		100%				
6.	Create Meeting Agenda	15 m	100%					
	Total hours		17h	4h	4h	20h	10h	

Week 13

Nr.	Task	Total Time	Yousef	Alina	Bozhidar	Borislav	Ghazi	Task Completed
1.	Code							
a.	Add Night mode Design	6h				100%		100%

b.	Add Randbout type of rode	3h				100%		100%
c.	Working on Dijkstra	10h	100%					
d.	Working on sequence diagrams	2h	50%			50%		
e.	Implement Save and Close functionality of generate report	2h		100%				100%
f.	Fix Save & Saves as functionalities	1h		100%				
g.	Working on use cases	1h	100%					
h.	Implement Max/Current Flow	7h			100%			
i.	Working on pedestrians	5h					100%	
j.	Enabled Road Sensors	5h					100%	
	Documentation							
2.	Update Work Division Report	1 h		100%				100%
3.	Create Meeting minutes	30 m		100%				
4.	Create Meeting Agenda	15 m	100%					
	Total hours		12h	5h	7h	9h	10h	

Reflections on this phase

- Handing the testing to just one member was a mistake, we made the decision to split it in the last week of the iteration, but it was too late by then. Because of this, we missed the deadline for unit testing by a couple of days.
- All the scrapped roads being removed was the right call. They were unnecessary and would have not added any value to the application while taking a lot of effort to implement.
- Splitting the unit testing allowed us to figure out a few hidden bugs (for example, the dijkstra nodes issue) that are much clearer to the person who made the logic of the unit.
- “Pedestrian sensors” is a nice feature to have but it does not add that much value, since cars have to wait for greenlight anyway. It would have been smarter to move pedestrians to iteration 3.
- We should have started with the comparison process between reports in iteration 1, because it took more time than we originally planned for it.

- Generally, everything went to plan in this phase, and we are very pleased with the work we have done so far.

Iteration 3

Tasks list for this phase

Since we merged this iteration with the previous one, we had no features left to provide here. Therefore, we decided to add new additional features:

Features

- New graphic view of statistics (visual display of statistics).
- Reports comparison.
- Additional shortest-path algorithm.
    - A*
    - Greedy Best-first Search
    - Breadth-first Search
    - Depth-first Search
- Export statistics as PDF.
- New theme stylization.

Documentation:

- URS.
- Design Document.
- Final product.
- Source code of the final product.
- Unit testing of the final product.
- Process report (Including work division report).

Challenges

As seen from the previous list, we had multiple questions we needed to answer before we could start implementing:

- What are some of the available libraries we can use to visualize our data?
- What are some of the available libraries we can use to style our application?
- What are some of the available libraries we can use to export data as a file (PDF)?
- What advantages would a new algorithm provide over Dijkstra?

All of these questions were challenges we needed to overcome, which we did with our research.

Research and research methods

We used DOT framework methods in order to find possible solutions to our problems, some of the methods we used were:

Available product analysis

We looked into already existing libraries; to see if any of them fit our needs.

Usability testing

We tried multiple styling libraries in order to find one which is the most usable.

Decisions, justifications and expected effects:

Based on the research we did; we made the following decisions in this iteration:

1. Discarded all new algorithms except A*

Justification: The new algorithms do not offer much compared to the effort required to implement them and since this iteration is short (only two weeks) it would be better if we focused on bringing value in a different way to our client.
We kept A* because it's considered an improved Dijkstra, the way A* works is:

1. Start by specifying the start node and the end node
2. From the start node, find all the adjacent nodes.
3. Calculate cost (distance) between those nodes and choose the one closest to the **end (in a straight line).**
4. From the new node, find all the adjacent nodes, pick one that's closest to the end.
5. repeat until you find the end node.
6. All the nodes and edges connecting them represent the shortest path.

A* offers an advantage over Dijkstra: faster calculations, however its accuracy is marginally worse.

Expected effects: More time to spend on the other features.
Allow the user to compare between two algorithms.

2. In place of shelved algorithms, focus more on data and visualization

Justification: Up to this point, we have not gone in depth when it comes to simulation numbers and data, despite the goal of this application is to compare different city plans and see how well they fare.
Measuring how well a simulation fares and putting it into visual representation is a great way to achieve the goal of this application.

Expected effects: More value to the client by providing better comparison and documentation of previously run simulation.

3. Allow user to export statistics as PDF

Justification: We think it is important that the statistics and the data we provide can be viewed, copied and printed; which is what we are offering by giving the option "Export to PDF".

Expected effects: Better utilization of simulation' statistics.

Work division

General task breakdown was:

Yousef

- Process report
- New Algorithms for shortest path (A*).
- Improve application performance
- Unit testing
    - Specify number of cars generated.
    - Dijkstra algorithm.
    - A* algorithm.
- Use case and sequence diagram:
    - Choose the “Emergency Vehicle” algorithm

Borislav

- Graph library research
- Generate statistics about the simulation
- Generate Graphs.
- Unit testing
    - Change simulation speed
    - Generate graph view about the statistics
- Use case and sequence diagram:
    - Generate graph view about the statistics

Bozhidar

- Modifying the style of the application.
- Unify documents (combining different versions of documentations, adding version history).
- Unit testing
    - Reset default values
    - Generate overall insights.
- Use case and sequence diagram:
    - Generate overall insights.

Ghazi

- Generate PDF file of the simulation’s report.
- Test document.
- Unit testing:
    - Selecting preset environments.
    - Generate PDF file of the simulation’s report.
- Use case and sequence diagram:
    - Generate PDF file of the simulation’s report.

Alina

- Compare statistics visually.

- Work division report update.
- UML Class diagram update.
- Unit testing:
    - Leftover features testing.
    - Statistics comparison.
- Use case and sequence diagram:
    - Compare statistics visually.

Week 14

Nr.	Task	Total Time	Yousef	Alina	Bozhidar	Borislav	Ghazi	Task Completed
1.	**Code**							
a.	Working on pedestrians	4h					100%	
b.	Working on Emergency vehicles	2h	100%					
c.	Working on generate report	5h	30%	30%		40%		
d.	Working on car flow	h				100%		
e.	Fixed bugs	2h	50%			50%		

f.	Fixed Saving feature	20m		100%				
	Unit testing							
a.	Start/Stop/Pause	3h		100%				100%
b.	Clear Layout	2h		100%				
c.	Add/Remove road	2h		100%				
d.	Save/Load file	2h		100%				
e.	Change Simulation Speed	h				100%		
f.	Change grid simulation speed exception	h				100%		
g.	Change trackbar simulation speed exception	h				100%		
h.	Reset default values	2 h			100%			

i.	Check Night Mode	2h					100%	
k.	Check Day Mode	2h					100%	
l.	Verify roads are created with no cars	3 h			100%			
m.	Verify number of cars generate d on lane	3h			100%			
n.	Check shortest path	2h	100%					
p.	Check points and lanes on grid	1h	100%					
	Documentation							
1.	Update Work Division Report	1 h		100%				
2.	Test Report	4 h		100%				

3.	Create Meeting minutes	30m		100%				100%
4.	Create Meeting Agenda	15 m	100%					
	Total hours		7.75h	16.5h	8h		8h	

Week 15

Nr.	Task	Total Time	Youssef	Alina	Bozhidar	Borislav	Ghazi	Task Completed
1.	**Code**							
a.	Added default number of cars for night mode	2h				100%		100%
b.	Added algorithm & Changes to form	2h	100%					100%
c.	Working on pedestrians	4h					100%	

	Documentation							
1.	Update Work Division Report	1 h		100%				100%
2.	Create Meeting minutes	30m		100%				
3.	Create Meeting Agenda	15 m	100%					
4.	Working on Process Report	10h	60%	40%				
	Total hours		8.25h	5.30h		2h	4h	

Week 16

Nr.	Task	Total Time	Yousef	Alina	Bozhidar	Borislav	Ghazi	Task Completed

1.	**Code**							
a.	Fixed bugs for pedestrians	4h					100%	100%
b.	Form materials	2h			100%			
c.	Added Statistics	3 h				100%		
d.	Test case Overall Insights	2h			100%			
2.	**Unit testing**							
a.	Test A* Algorithm	30 M	100%					100%
b.	Overall insights	2h			100%			
c.	Generate graph view	2h				100%		
d.	Generate PDF file	2h					100%	

3.	**Sequence diagrams and use cases**							
a.	Choose a shortest path possible algorithm	3h	100%					100%
b.	Overall insights	1h			100%			
c.	Generate graph view	2h				100%		
d	Generate PDF file	1h					100%	
4.	**Documentation**							
1.	Update Work Division	1h		100%				
2.	Update Class Diagram	2h		100%				
3.	Update Process Report	2h	100%					
	Total hours	h	5.5h	3h	7h	7h	6h	

Reflections on this phase

- As discussed before, we scrapped all the new algorithms that we planned on implementing except A*, but we should have scrapped A* too.
  Because of the small scale of our project, the expected advantage of A* being faster than Dijkstra is barely noticeable. But the lower accuracy of it is noticeable. Overall, Dijkstra is the better algorithm to find the shortest path.
- This iteration is the most important one in terms of bringing value to the client. Up to this iteration, the user could generate a report and save it. But interpreting the performance of each simulation was subjective.
  Now that we have provided visual tools and exportation options, the user can easily tell which road-plan would perform the best in real life.

Personal Reflections

In this section, each member is going to reflect on what went wrong, what went well and what they would have changed in this project:

- **Yousef**

This project was a great practical experience for me, the main learning points that i took away from this project were:

- Improved soft-skills: I learned how to work within a team more so than I knew before, I learned how to brainstorm, give and take feedback, and as a team-leader, I learned how to organize meetings, prepare beforehand, take initiative and be critical of what the client has to say.
- Better planning skills: whether planning the project, iterations or meetings, my team and I learned how we could plan a project realistically and stick to promised deadlines.
- Advanced documentation skills: I got familiar with the typical documentations that a developer might have to write. I learned what's essential in each one of them and what should be left out.
- Improved technical knowledge, especially in:
    - Version control.
    - Design patterns.
    - Testing (unit testing).
    - Working with new libraries.
    - Writing algorithms.
    - Coding within a team.
- I learned how to apply my previous knowledge into something practical, such as algorithms I previously picked up in a math course.
- On top of that, I got to discover some of my weaknesses as a developer, which i will be working on. These weakness are:
    - I have a tendency to tunnel-vision certain aspects of the project, while ignoring the picture as a whole.
    - I am sometimes inclined towards adding new features that are nice to have, but wouldn't bring much values (Like the A* algorithm).
    - I have a hard time getting rid of work that I have already done, even if it's better to; I would rather maintain an old code I worked on than add a new, better one.

Overall, this project was a valuable learning experience, and will be detrimental for me during my internship.

- **Alina**

I believe that this project was a bit challenging for me for the reason that I was not aware in the beginning of how a traffic lights system functions in real life. However, while working on this project I got a lot of knowledge related to the possible roads and rules that exist and the logic that cars need to have in order to simulate a real traffic situation.

From the very beginning, I consider that we worked together really well as a team. Everyone was willing to help others and tried to contribute as much as he could to the project.
Communication is the key factor for this course. Even if we couldn't meet face to face on account of the current pandemic situation, I believe that we managed to communicate quite well on online platforms.
I managed to improve my technical knowledge regarding Unit Testing, Version Control, Coding within a team and also Reporting as I learned how to write reports in a professional manner.
I also learned to ask questions and communicate whenever the code of my team members was not clear. In this way, we managed to deliver the promised features .
Next time, I need to take into consideration that even though I implement features and they work, I always need to maintain the code as future changes may affect the accuracy of it.
Overall, I believe that this project improved my technical knowledge and it taught me how to work and communicate better in a team.

- **Borislav**

The project was a really fun and valuable experience for me. I believe me and my team did the best that we could for the time that we had. Thanks to our team leader we had a great communication and a perfect task distribution supported by daily standup meetings.

During the project a lot of code was implemented but we were not only thinking about having the application to work, we were always trying to optimize and make the right decision no matter that it may be harder. We made use of design patterns, unit testing and our version control was perfect.Moreover, we managed to implement MUST,SHOULD and COULD feature and we still had 1 month left in which we added additional features such as one of the most efficient path finding algorithm A* Star and also display the results of every simulation in beautiful graphs, because we all know that everything is much easier to comprehend when it is visualized.

In the first place, I personally managed to improve my communication skills. Now I feel much more confident than before this project. Furthermore, I improved my coding skills, learned working with the Mediator design pattern, learned to do unit testing and understood how important it is to test your application. Although I did not have the task to implement the path finding algorithms, I did spend time looking at their implementation and understanding the way they work. Not on a last place, I learned integrating charts and visualizing graphs in windows forms project, which I think was a really important feature for this project because with the help of comparing different simulation reports and seeing the visualization of it, the client would be able to spot and apply different insights.

However, I still have a lot of work to do on my personal skills and this project helped me to realize and find some of the weaknesses that I have.

- **Bozhidar**

This is the second big project I have had in my ICT studies and it was bigger and more complex than the previous. When we started we had to make big decisions very quickly so there was a lot of research to do. We had to choose a platform and start straight away with brainstorming and writing documentation. During the project we had perfect communication and organization. Every iteration

the tasks were split between the team members very easily and with great organization thanks to our team leader. We had a lot of challenges but we overcame them as we finished with musts, shoulds and coulds at the end of the second iteration so we had a whole iteration to come up with extra features. I personally think that our team did very well and coped with the challenges very fast. For me this was a great experience as it was more realistic than pro-p and it was something that is near to real life project. I learned about new libraries, unit testing, algorithms and even researched how road capacities are calculated and how to apply it to our application. I learned to manage my time better as we were always involved in the project, everyday there was communication on what is going on and what is left to do and I didn't want to be late with commits or documentation. Overall I think this was a very useful project as we learned how to learn and how to do it fast. Moreover, we went through research, implementation, testing and documentation. We went through it all and we are one step closer to working on real projects. However, there is still work to be done as there are a lot more different technologies to be learned.

- **Ghazi**

In this project, I learned a lot of things more seriously compared to the previous P project. Within a group, communication is a very important part of being a successful team. I learned how to cooperate with others to make specific functionality for the app. For example, when making an app it is mandatory to understand what other member codes are. Therefore it is important to ask questions when I do not know what the code is about. I learned to be initiative by asking questions to a teammate since it is better to understand and helps me figure out what is the problem. Also, we always deal with conflict in our version control system and it is good that we have good communication so we can trace what went wrong. Furthermore, for the documentation, I learned how to make the report professionally because we are always being checked by our mentor and it is also good for a real working job later. I believe by making the documentation structured will result in a good product. Moreover, I experienced that it is also important to have a better planning of tasks that I want to finish it because it will affect team performance and slowing down our progress. Lastly, for the technical part, it should be take into account that we work with a team. So i tried to make the code more structural and more easy to understand. Mostly i learned problem solving skills or other libraries from the code by myself since in this situation is not easy to go ask to the teacher directly. Overall,i'm happy able to be contribute in this project and in my opinion this project teaches us how to work in a real team project.