



UNIVERSITY OF GOTHENBURG

MiniProject: Distributed Systems Software Architecture Document

Group 8

Final Report

Munatsir Adam

Mila Mehrvarz

Leila Bencheikh

Ramzi Abu Zahra

Maab Mohammedali

2021

1- Program Management brief

- **Describe the project management practices used**

We have used Agile development methodology for our project management. We have identified the project requirements and broke them down into tasks which we included in the sprint backlog. Furthermore, in our project development process we had sprint planning and sprint review meetings to consolidate the team's efforts by unifying our vision and demonstrating our results to each other. In addition to defining the nature of our meetings, the team members have been assigned to different roles such as Scrum master, Developers and Product owner (TA) which helped us in organizing our contributions more clearly.

- **Report on important project management decisions regarding schedule and scope**

During each sprint we set deadlines for the tasks that need to be completed individually, as a subgroup and as a team. The communication between team members has been conducted via Discord, Zoom and Slack. In this project, Trello was the main management and planning tool that has been used. Moreover, we decided to use software which allows online collaboration such as Draw.io for creating various software diagrams, and Google drive for easy collaboration on project documents and assignments.

2- Strategy & design decision

- **Provide your overall strategy to manage and execute the project**

We have come to some project management decisions. First of all, we decided the scope of the project which was defining four milestones and dividing each requirement to specific milestones. According to agile program management methodology, we selected a scrum master and defined the roles for each member in the group-8-project component. In order to complete specific requirements, each task is assigned to different members or to subgroups in our sprint planning meetings. In the first two sprints the main goal was to just come up with a Minimal Viable Product, which meant that all suggestions to improve the looks of the UI, or add features that enhance the experience were postponed to the end of the project. Each task defined by the group had a specific deadline which was usually set to be done before the scrum Review meeting, the reason is that we wanted to give ourselves time to find if there were issues such as conflicting codes or even codependent codes between subgroups. And at the sprint review meeting we presented to each other the tasks to the group.

There would then be two possibilities; If the task was successfully accomplished it would move to the "Solved" section on Trello. But if there were some places for modification of the tasks it would be replaced in the "Needs Improvement" section on Trello. The assessment of tasks that need to be completed is based on the criteria that we defined on the checklist in each Trello card, which defines exactly what is needed to be accomplished so that it is considered Done. To prioritize tasks visually we used different colored labels for each card, as an example red was the color that refers to a task which has a high priority to be completed whereas blue is for less urgent tasks.

- **Identify your Architectural Significant Requirement (ASR) that impacted your design decisions**
 1. The system should unsubscribe from the MQTT broker once a topic is sent through MQTT to a different component (15 in srs)
 2. The system should have at least 4 distributed components (02 in constraint)
 3. Upon choosing the date and the time of the appointment, the system shall display all available dentists locations on the map.(19 in srs)

4. The system shall send booking requests and receive booking responses in a predefined format. (05 in srs)
5. Upon clicking on the booking button, If the booking is rejected the system shall send a rejection message (17 in srs)
6. The system shall send an error message when booking a time outside of the opening hours of each dentist (12 in srs)

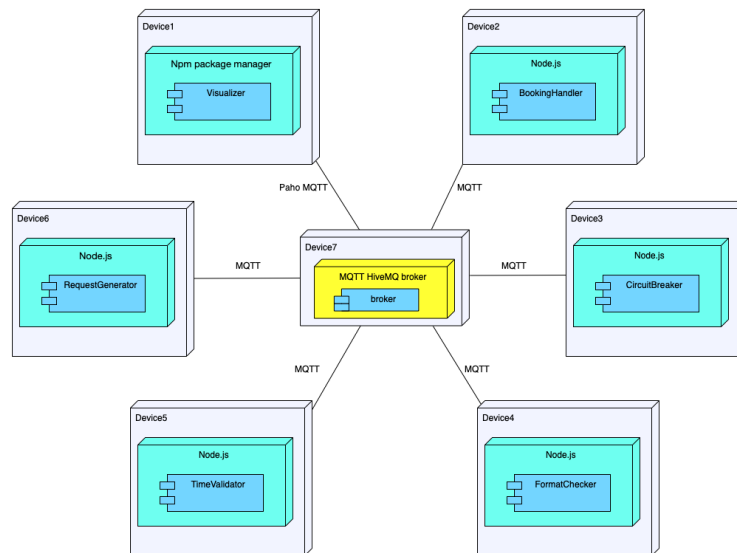
3- Architecture, styles, and tactics

- **Include a clear description of the conceptual design of the architecture, including architectural styles**

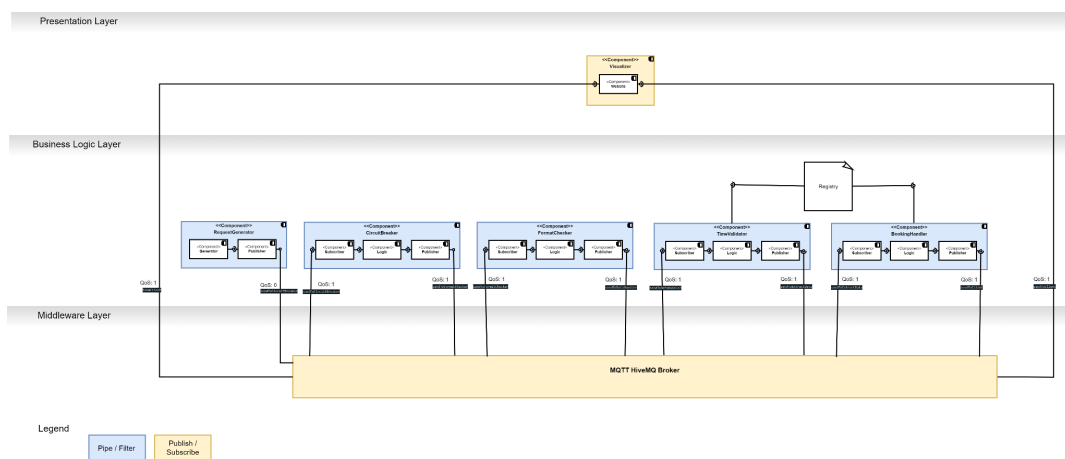
The driving forces around our early and later architectural choices were centered around two main architectural styles in our Dentistimo Booking System, those are Publish-Subscribe architecture and Pipe-Filter. The reason for choosing the Pipe-Filter style is to give our system the ability to further expand when more components are needed to be installed for further filtering or for adding more features to the system. This has proven useful especially at the end of the project when we needed to make sure the system has the ability to manage fault tolerance and stop the flow of requests when a certain threshold is reached. Which explains the additional components that were added later on, namely the circuitBreaker and RequestGenerator. Creating the Pipe and Filter system meant for us at the beginning that a component with the name Pipe should be created. At the beginning of the project the component named Pipe was initially created to manage and delegate the messages between components, but was later removed since we found out that the Mqtt broker delivers the same result thus omitting the need for redundant components like the Pipe.

Some components took the role of being a double filter such as formatChecker which did not just check the format received when a date is chosen by the client, but also the booking of a clinic was also checked within the same component in order to pass the data to the next components with validation. In order to do that in one component we decided to divide the messages sent to the same component according to the size of the message, since the booking message contains more information than the date message(which had a fixed maximum size of 27 bytes) we knew that the same filter could have more than one behaviour while receiving the same topic from the previous component. Another reusable element in our system was the Registry file, which is used by both the timeValidator and the bookingHandler, since redundancy of data is not optimal for a system that depends on updatable data, we opted for making the registry available for both components simultaneously by keeping it in the root folder of the system. In addition to that the data that the Registry depended on was collected using a url based JSON file, allowing the system to be more scalable and manageable.

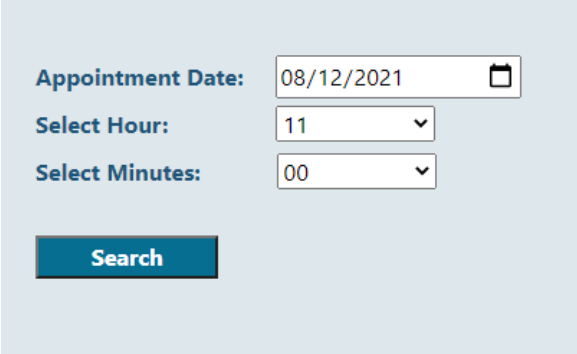
- Include a section that explains how the conceptual design is mapped onto implementation/technologies.




The whole system is centered around the Mqtt protocol, which allows each component to be run independently on a different device, or even multiple devices, where the broker remains the same. Such design choices will allow scalability of the system.



The component diagram above demonstrates how the Presentation Layer is both the starting and ending points of the data flow within our system. Through close observation of the diagram one might notice that the broker handles all data except for the Registry which is crucial when it comes to the availability of data, and avoiding delays caused by API calls, since the number one priority is to provide the data to the user in the most efficient manner.

A screenshot of a web form for booking an appointment. The form is set against a light blue background. It contains three labels in bold blue text: 'Appointment Date:', 'Select Hour:', and 'Select Minutes:'. The 'Appointment Date:' label is followed by a text input field containing '08/12/2021' and a small calendar icon to its right. The 'Select Hour:' label is followed by a dropdown menu showing '11' with a downward arrow. The 'Select Minutes:' label is followed by a dropdown menu showing '00' with a downward arrow. Below these fields is a dark blue button with the word 'Search' in white text.

Appointment Date: 08/12/2021 

Select Hour: 11 ▼

Select Minutes: 00 ▼

Search

The client's ability to customize the date of choice will trigger the TimeValidator Component that directly depends on the registry to iterate and fetch the needed data, without the need for callback functions. The instantaneous results are then delivered to the BookingHandler to notify the client of his success or failure in booking the clinic according to the components own implementation which requires accessing the Registry.

4- Lessons learned & Conclusion

- **Identify most important learning experience for your team in this project**

The most important learning experience for the team was to create an architecture that supports future changes. For example we added two more components to the distributed system in order to implement the circuit breaker. The decisions were centered around performability, availability, and deployability. During this project we had to focus on what are the systems strong and weak points, and performability was one of the most important drivers in shaping our decisions during due to the fact that adding a circuit breaker that checks the incoming messages will need to be within a reasonable frequency so that no latency would occur, and that meant that all other components would be affected before reaching the client's side.

- **Identify most significant challenges and how you mitigate these challenges**

The most significant challenges included how we divide the work between us without anyone feeling left out, or exhausted. Since each person had different visions of what the project should look like we had to always stick to the requirements document that we created in the beginning of the project to mitigate these challenges. We also faced some challenges when it came to creating a connection between the UI and the mqtt broker, which was solved by going back to the HiveMQ API documentation, and following the steps described early on in the first sprint.

- **For a project similar to this in the future, what would be the things that you wish available in order to deliver best results.**

For future projects focusing on what constitutes a minimal viable product should be defined as detailed as possible to keep the focus of the group on achieving the correct requirements. We also learned that everyone's idea of cooperation is different, so in the future we should be able to understand what each person expects of the group as a whole and what is expected of each individual when it comes to defining what cooperating means in a group.