**SOFTWARE ENGINEERING II**

**PROJECT SOLUTION PROPOSAL**

1. **ABSTRACT**

The world today has been calling for software products that are not only complete in functionality, swift in performance, but also to be as scalable as possible.

The goal of the Software Engineering II course, as described in the module description*[1]*, is to arm the learners with sufficient knowledge of developing high-level architecture and large-scale software design. For that reason, the module’s project should be developed with good scalability in mind. This document will propose a way to develop the module project that does not only satisfy the given requirements*[2]*, but also gives great consideration for scalability issues.

***Keywords:*** *dynamic web, Separation of Concerns, REST API, HATEOAS*

1. **REFERENCES**

*[1]* ***Software Engineering II – Module Description****, Nguyen D.T. Long, Faculty of Information Technology, Hanoi University, 2020.*

*[2]* ***Software Engineering II – Project Description****, Nguyen D.T. Long, Faculty of Information Technology, Hanoi University, 2020.*

*[3]* ***What is REST? – Learn to create timeless REST APIs*** *(*[*https://restfulapi.net/*](https://restfulapi.net/)*), viewed in March 2020.*

*[4]* ***HATEOAS-driven APIs*** *(*[*https://restfulapi.net/hateoas/*](https://restfulapi.net/hateoas/)*), viewed in March 2020.*

1. **DEFINITIONS, ACRONYMS AND ABBREVIATIONS**

***REST*** – Representational State Transfer

***API*** – Application Programming Interface

***HATEOAS*** – Hypermedia As The Engine Of Application State

***HTTP*** – Hypertext Transfer Protocol

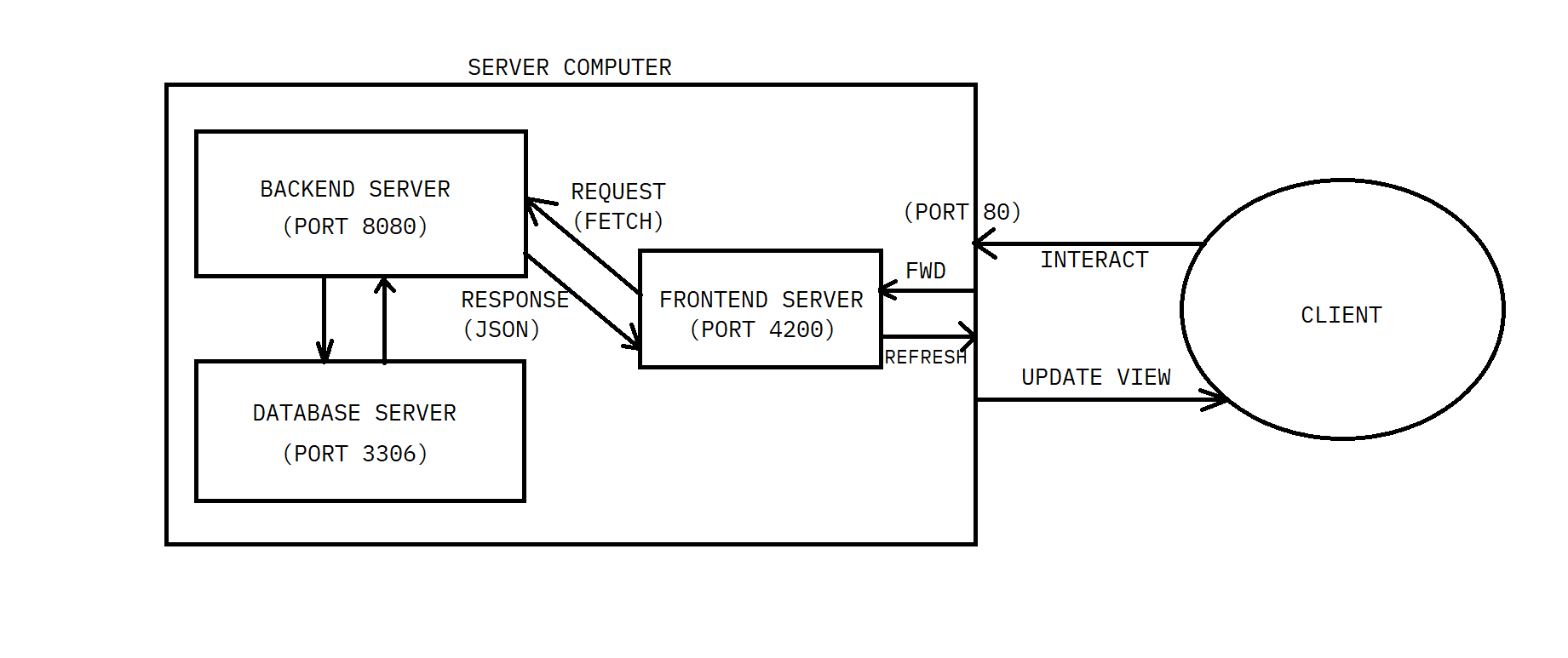
***JSON*** – Javascript Object Notation

***HTML*** – Hypertext Markup Language

1. **SOLUTION**
2. ***Overall System View***

In this solution, the system is divided into two clear components: the front-end server process and the back-end server process. Two server processes can either run on the same server computer on two different ports and different from port 80 (HTTP port) or run on two distributed server computers.

The front-end server process and the back-end server process communicate by means of HTTP requests. The back-end server need not know anything about the view technology implemented by the front-end; it only ***handles request*** from the front-end server and ***returns data*** (preferably in JSON format) or ***executes actions*** accordingly; therefore the back-end server works in the same way as a ***RESTful****[3]* ***web service***. The front-end server constructs the User Interface from whatever the back-end responses. The communication can be illustrated using the following diagram:



Organizing the project this way, the back-end server ***needn’t have any form of template engine*** (such as Java Server Pages) implemented – it can be fully operational by implementing only Servlet technology.

1. ***Evaluation***

The main benefit of this way of organizing the system is, of course, Separation of Concerns, as the View can be totally separated from the logic that drives the overall system operation. Therefore, it is easy to swap out the front-end technology with another (for instance from ReactJS to Angular) or to make another application serving the same purpose but for other platform (for example, writing an Android application for monitoring the CoViD-19 disease). Moreover, developing the system this way takes the pressure away from the back-end server, as it only has to return JSON data as opposed to resolving templated views to HTML format and return the HTML data (which is much heavier than returning JSON-only). This solution also paves the way for developing our application as a single-page application, which can make our application even more dynamic and responsive. Java WebSockets technology can be applied later on to our system to provide a real-time web application experience.

However, to efficiently organize the system using this solution, the back-end server needs to return HATEOAS-compliant*[4]* data, in which not only the object data but also the ***associated links*** must also be included. This may result in taking more time to design API efficiently, however the outcome will be a navigable API that remains consistent over time. Also, the front-end can ***dynamically render links*** based on the JSON it gets, therefore the related links returned from JSON response may change without any need to rewrite the front-end code. It may also take more effort to think of how to implement the front-end from the start (especially how to generate the links based on the response), however the result will be more open to changes, as stated above.

1. **CONCLUSION**

In this short document, I have discussed our group’s proposed way of building the system to facilitate not only the given requirements, but also as much scalability as possible. This method of developing software application by physically detach the View component from the underlying business logic can be applied in any other modern software products, especially with larger-scale products or software systems that are open to extension. Hopefully this proposal will get approved so that we can build a quality system that qualifies itself for the student science conference this year.