Second Year Internship Report

Deploying a desktop application online

Submitted by

Kévin AFFRAIX

ENSTA Bretagne Engineering School

2 Rue François Verny, 29200 BREST, FRANCE

Under the guidance of

Walid TAHA & Sundas MUNDIR

Department of Effective Modeling

Halmstad University

Kristian IV:s väg 3, 301 18 HALMSTAD, SWEDEN

Acknowledgment

I would like to thank Professor Walid TAHA for giving me the opportunity to work on this project. I also would like to thank Sundas MOUNDIR and Sotiris TZAMARAS for their support thorough this project.

Abstract

This report deals with the implementation of an online application for modelling and simulation of cyber-physical systems. As another, more technical document has been produced for the future developers of this application, this report will deliberately focus more on the strategies envisaged for the development, as well as the tools and concepts used. It aims to give an overall vision of the project and the reasons that led it to be what it is today. Throughout the report, we will see how the web overlay of the project has been structured, the main features that have been integrated on the web, as well as the avenues for improvement that are to be explored for the future of the project.

Résumé

Ce rapport traite de la mise en ligne d'une application de modélisation et de simulation de systèmes cyber-physiques. Un autre document, plus technique, ayant été produit pour les futurs développeurs de cette application, ce rapport sera volontairement plus axé sur les stratégies envisagées pour le développement, ainsi que les outils et concepts utilisés. Il a pour but de donner une vision globale du projet et des raisons qui l'ont mené à être ce qu'il est aujourd'hui. Tout au long du rapport, nous verrons comment a été structurée la sur-couche web du projet, les principales fonctionnalités qui ont été intégrées sur le web, ainsi que les pistes d'amélioration qui sont envivagées pour la suite du projet.

Contents

1 Context **1**
- 1.1 Laboratory presentation . . . 1
- 1.2 Acumen Project . . . 1
- 1.3 Objectives . . . 2

2 Problem definition **3**
- 2.1 Main tasks of the internship . . . 3
- 2.2 Constraints of the project . . . 3

3 Work Done **4**
- 3.1 Structure the project . . . 4
  - 3.1.1 Choose a hosting service . . . 4
  - 3.1.2 Strategies . . . 4
  - 3.1.3 Final structure of Web Acumen . . . 5
- 3.2 Modify Acumen source code . . . 6
  - 3.2.1 Find a new communication solution . . . 6
  - 3.2.2 Enable Headless Run . . . 7
- 3.3 Implement Acumen features online . . . 8
  - 3.3.1 Code editor . . . 8
  - 3.3.2 2D Plot . . . 8
  - 3.3.3 3D Animation rendering . . . 8
- 3.4 front-end Test . . . 9

4 Future Work **10**
- 4.1 Optimize Acumen intern communication . . . 10
- 4.2 Make Web Acumen a multi-user application . . . 10
  - 4.2.1 Make Acumen a multi-user tool . . . 10
  - 4.2.2 Enable Server to run one Acumen instance per web client 11
- 4.3 Implement online database . . . 12

5 Conclusion **13**

References **14**

Chapter 1

Context

1.1 Laboratory presentation

During this internship, I worked in teleworking for one of the research centers of the Halmstad University. Founded in 1983, this university is divided into four schools: the School of Business, Engineering and Science, the School of Health and Welfare, the School of Education, Humanities and Social Sciences and the School of Information Technology. The Effective Modeling Group, belonging to the School of Information Technology, was the research team offering me the opportunity to take part in one of their projects: Acumen.

The Effective Modeling Group directed by Professor Walid TAHA, specializes in Cyber-Physical Systems, which are, according to the group's definition " systems where embedded cyber components interact closely and in complex ways with their physical environment ". Concerning a vast majority of future innovations, Cyber-Physical Systems needs more than ever new tools allowing us to better understand, simulate and visualize their complex behaviour.

1.2 Acumen Project

Quoting the definition in the reference manual:

> "Acumen is an experimental modeling language and integrated development environment (IDE) for model-based design of cyber-physical system." [1]

The desktop IDE (1.1) contains:

- A code editor with integrated text highlighting.
- A library of examples with an integrated browser to access it.
- A console printing messages generated by the computation core of Acumen.
- Three display windows actualized in real time. The *Plot* window will display charts of all variables as a function of time. The *Table* window will show a table containing the values of each variable at every single step. Finally, the *_3D* window will show the 3D animation corresponding to the model we created.

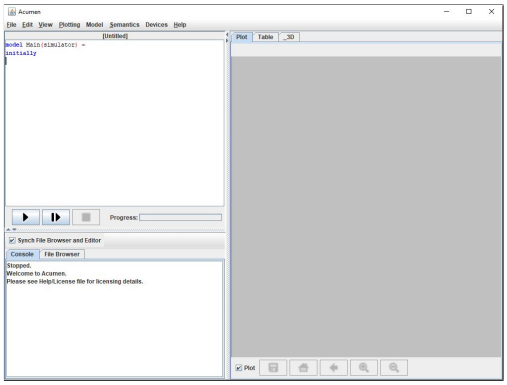

Figure 1.1: Acumen desktop IDE

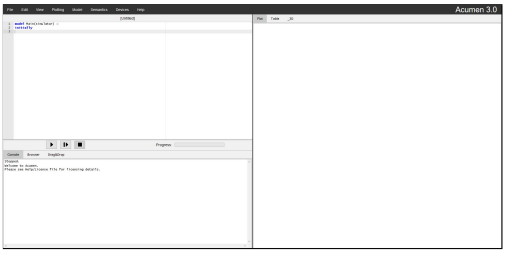

Figure 1.2: Acumen Web Interface

Acumen always had three goals: to be rigorous, but also practical and accessible, so that it would suit both researchers, teachers and students. The desktop application, requiring the user to have programming experience for the installation, did not perfectly complete those goals. Therefore, a project to transform Acumen into a web application was born, the aim being to improve greatly Acumen's accessibility.

1.3 Objectives

This internship had two main objectives:

- First, I had to deploy Acumen on a web server. The aim here was to explore the first attempt of the team to make a browser-based version of Acumen, and to find a working structure integrating Acumen and making it accessible on the web.
- Then, I was asked to work on a new Acumen semantic that would use set-based analysis to provide a new way to integrate randomness in Acumen language.

Unfortunately, the pandemic compelled us to work remotely, and the first part has fallen behind. We chose to concentrate our efforts on the first task, so that we could implement most of Acumen features online.

Chapter 2

Problem definition

2.1 Main tasks of the internship

Deploying Acumen online can be divided into tasks as follows:

- Acquire basic knowledge on the languages that will be used during the internship: Scala (main language used for Acumen development), HTML/CSS (web page rendering) and Javascript (script language for web pages).
- Create or find a front-end / back-end template that can be deployed on Heroku, the hosting service we chose.
- Build the communication between Acumen and the new front-end.
- Implement Acumen features on the front-end.
- Write a documentation for both users and future developers that will work on the project.
- Create a test-bench for the additional code developed during the internship.

2.2 Constraints of the project

I identified three main constrains restraining the project during my internship:

- First, as Acumen is an open source application, all added libraries add to be open source, or at least provide a sufficient free service. This proved to be particularly restraining for the hosting service.
- The second main constraint of the project was that Acumen source code had to be left untouched unless absolute necessity. All the additional code had to be a software overlay handling the web IDE.
- Finally, the last main constraint of the internship was the choice of Heroku as a hosting service. This must be nuanced as we were not compelled to use this tool for the project, but as we discovered its limits once the application was built especially to deploy it on Heroku, we then just did not have the time to built a new project from scratch again.

Chapter 3

Work Done

3.1 Structure the project

3.1.1 Choose a hosting service

We chose Heroku to host our application online. This choice was made mainly because one team member already had experience with this tool. This service provides:

- 512MB of RAM
- One free dyno (linux container): contrary to other services, Heroku free dynos can always be activated (the hour limit was plently sufficient), so we did not had to define offline hours. However, Heroku will put to sleep all dynos inactive for more than 30 minutes, and therefore slow down a bit the first access to the front-end to wake it up again.
- Facilitated deploy thought the use of Git.
- A custom domain with access to three ports (basic HTTP and HTTPS ports and a third port to connect to our server).

3.1.2 Strategies

The first huge task to complete during this internship was to find or create an application sample structure that could both be deployed easily on server and integrate Acumen. In order to achieve this, we explored numerous possible solutions:

- Deploy Acumen jar file: the first idea was to deploy directly the executable jar file, which would have left Acumen source code unchanged. However, Heroku server did not support Acumen jar file as it was not running in headless mode. Moreover, each attempt to modify Acumen source code to update it later would require to recompile and deploy the whole file, which was not ideal. From this unsuccessful experience, we learn that we would have to bypass Acumen graphic features, and add it as a dependency to a bigger project. We then searched to divide front-end services and back-end ones, and link Acumen source code as a back-end dependency.
- The second idea we came up with was to build a front-end application and a back-end server application separately, and then make them communicate through Heroku. The reason was that Acumen required

an old version of SBT (which is the build tool used to compile scala code) which was fine with the back-end requirements but would restrain what we could do on the front-end side. We then tried to use the PM2 Module to handle both applications. However, because we had to use Heroku free service, we were limited to one dyno (linux container) to make the whole application work. Therefore, we had to create a structure that would integrate in the same application both the front-end and the back-end.

- We then discovered that SBT allowed us to build sub-projects separately and then merge them within a bigger one. Therefore, we tried to create our own back-end and front-end projects, and merge them together before deploying the application. Unfortunately, our lack of knowledge on Scala and ScalaJS greatly hindered our progress. We concluded that we either had to spend more time to familiarize ourselves with the language, or to find another solution. As we began to be short on time, we chose to search for sample applications that would adopt a similar structure, so as to have little work to do to adapt it to our problem.

3.1.3 Final structure of Web Acumen

The overall structure of the application was built relying on a tutorial made by Antoine Doeraene [2] . First, this structure allows us to deploy easily our work on Heroku, which was the main goal of the tutorial's author. Next, it contains both the front-end and back-end of the application in the same global SBT project, but clearly separate both the code and the dependencies of each one. This allows us to better adapt each dependencies' version to make it compatible with Acumen ones, which can sometime be old. As for Acumen code source, it can be find on the official GitHub repository of the project [3]. Finally, the work I completed during the internship can be found on my personal GitHub account [4].

The tool used by the back-end of the tutorial is Play Framework. Not many changes were made on this part of the application, but improvements must be made here to find a better way to retrieve Acumen messages efficiently

Three main parts form the front-end:

- First, a block using ScalaJS and the Laminar library. This part implements the overall display of the webpage. The display was written with Scalatags. The part building http requests' structure is no longer used

but was left in case future developers prefer Scala language instead of Javascript.

- Next, a block handling the style of all elements, using CSS language. Some of this part has also been implemented in Scalatags, but because the library is far from being as rich and documented as CSS, it is advised to stick to CSS for that kind of work.

- Finally, most of the features implemented in the front-end rely on Javascript. The main reason for using this language is its popularity on the web developers' community. Numerous libraries can be found, and the support cannot compare to ScalaJS one, which is, in the end, only a facade transforming Scala code into Javascript. This part mainly implements what type of actions to launch when the user interacts with the website.

3.2 Modify Acumen source code

3.2.1 Find a new communication solution

The hosting service we chose to work with is Heroku. It provides free hosting and domain, if the application can work on a single dyno (Linux container). However, that means we must give up on the *Web Socket* solution that was implemented previously.

Now, we can identify two different types of communication in Web Acumen:

- First, the retrieval of Acumen messages (3.1). This kind of communication is for the time being necessary to handle all messages generated by Acumen when an action is required by the user. Those messages are first and foremost stored in a buffer. On the front-end side, the client periodically sends a request to the back-end to see if the buffer contains some message. In this case, the back-end will access the buffer variable and send back the oldest message sent by Acumen. Empiric test showed that Heroku server begins to show errors when the rate of these request exceeds 3 times a second, while, locally, it can go up to 15 times a second. The length of request data that can be send being limited, in case of very long messages, we split them into chunks of data, and then send them in chronological order for the front-end to receive them in the right order.

- Secondly, the processing of the user's actions (3.2). Each time the user asks for an action requiring Acumen to intervene, this communication will be fired. An XMLHttpRequest is generated, sending a JSON object specifying the action Acumen must launch. The request then ends and notifies the user in the console that Acumen has received the information. The result of this action, however, will generate a message that will be processed as said above.

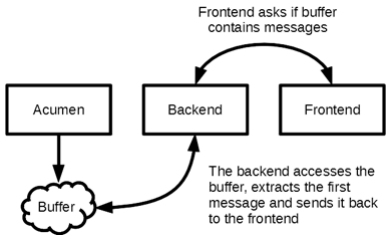

Figure 3.1: Illustration of Acumen-front-end communication

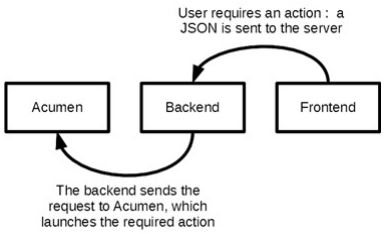

Figure 3.2: Illustration of front-end-back-end communication

3.2.2 Enable Headless Run

The desktop IDE provided with Acumen uses the Swing library to handle the Graphical User Interface (GUI). It would run once the main class of acumen was called, and load all graphic elements. However, this was now a hindrance for two reasons:

- First, now that we handled the GUI through the front-end web interface, it was now completely useless.
- Next, Heroku server required the application to run in java headless mode, that is to say without calls to display elements.

To minimize both the time we spent on this issue and the amount of code we had to modify, we chose to simply bypass only the displaying of elements, so as to keep the instances of each element, most of which being tightly linked to Acumen behaviour.

3.3 Implement Acumen features online

3.3.1 Code editor

The code editor we chose for Web Acumen is Ace Editor. To implement text highlighting with Ace, we must configure two types of files.

- Ace mode: it allows you to the syntax highlighting you want for the code written in the editor. Briefly, you choose here "what" you want to highlight. To achieve that, ace will ask you to provide both the words (language keywords, built-in functions, ...) and symbols you want to highlight. More complex expressions can also be detected using regex syntax.

- Ace theme: it enables us to choose "how" highlighted text is going to stand out (Ace themes are basically defining CSS style for elements it automatically generates).

3.3.2 2D Plot

To implement 2D plotting on Web Acumen, we use the Plotly library. When the user wants to compute his code, Acumen will send data in real time providing the active window is the *Plot* window (3.3).

3.3.3 3D Animation rendering

We tackled 3D animation rendering with the BabylonJS Library. Currently, Web Acumen does not support real time rendering. When the user has computed the whole simulation one time, he can then replay the animation, and Acumen will send all frames data at the same time once he activates the 3D window (3.4). It works as follows:

- First, we declare a Scene object that will contain all animations.We create a light mimicking the sunlight, setting its position and orientation. We also create a fixed camera and enable the user to modify both its position and the target position it will be looking at.

- Then we look, frame by frame, Acumen messages and create or transform the objects it concerns. There are two ways to create an object: simple shapes (cubes, spheres, axes, ...) can be created with BabylonJS built-in functions, but more complex shapes described by .obj and .mtl files can also be imported.

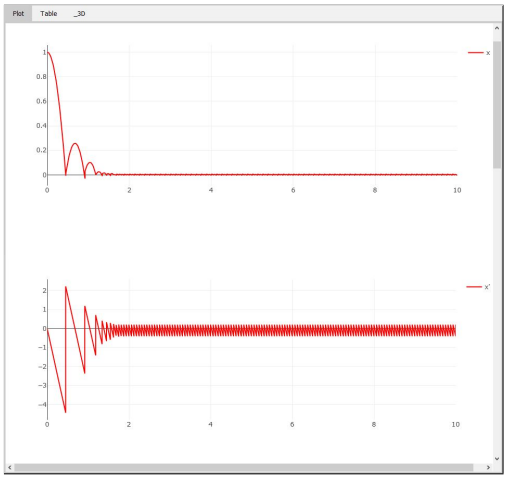

Figure 3.3: Plot window

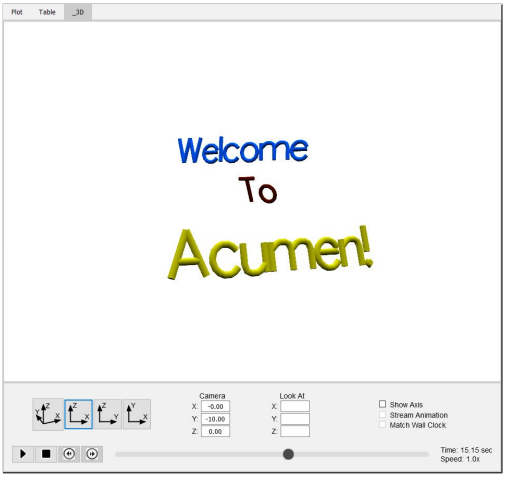

Figure 3.4: 3D window

3.4 front-end Test

Regarding tests, we chose to use the Cypress Library. The reason for this choice is that numerous test are already checking if Acumen core is working properly, so we only have to check if correct data is send to the front-end and displayed correctly. It allows us to check the application from the user perspective. A typical test would:

- Access the website
- Open a file (from the browser or through the drag and drop feature)
- Select a Semantic
- Run the simulation
- Open *Plot* and *Table* windows to see if data is correctly displayed
- Run 3D simulation
- Check if the 3D animation is correctly displayed on the *_3D* window

Chapter 4

Future Work

4.1 Optimize Acumen intern communication

Using a buffer to store information cannot be a long-lasting solution. We have to send requests be it empty or not and maintain a high rate to load huge data arrival quickly enough. What is needed here is to implement some kind of socket connection, whose server would be on the Play Framework back-end and taking Acumen and the web user as two of its clients exchanging information. The main issue here is that this connection cannot have a dedicated port, because Heroku only provides a unique port for the application back-end.

4.2 Make Web Acumen a multi-user application

Currently, Web Acumen can handle only one client at a time. Main reasons for this are:

- Currently the Server can have only one instance of Acumen at a time.
- Acumen is meant for one user at a time, so one instance of Acumen can only work for one web client.
- The communication between Acumen and the front-end works thanks to a buffer. Currently I check the buffer three times a second. More clients would mean an explosion of http requests to check the buffer, the Server would not keep up with that.

To tackle this issue, two paths should be explored: transforming Acumen so that it can handle many clients at a time, or making the server run one instance of Acumen per user connected.

4.2.1 Make Acumen a multi-user tool

Because Acumen was built as a desktop application, most of its features only support one user at a time. Be it the computation, the console or the code editor, Acumen should be able to run many of those and discriminate between each client. This can seem to require too much computation power, but actually many of Acumen actions are no longer required in the web application. The swing library that was implemented to show the GUI of the application can be removed, the jPCT library implemented to load 3D

animations should be suppressed ... Acumen should, for a web version, only be a "computation core" sending messages to the front-end. File gestion, 3D rendering, GUI are now handled by the front-end and should not appear on Acumen anymore.

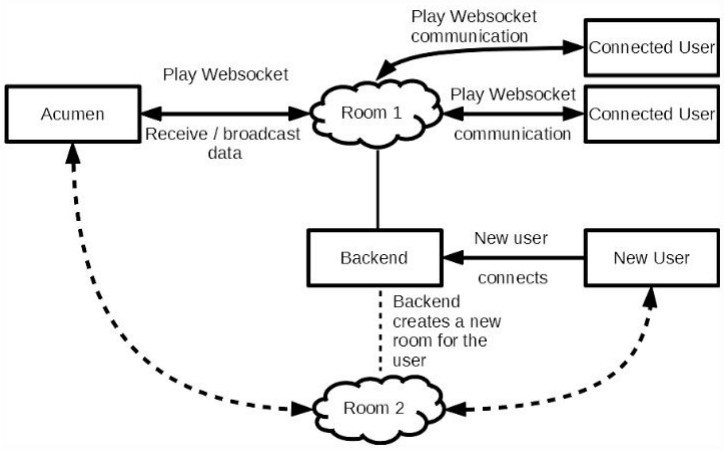

Figure 4.1: Possible structure when Acumen becomes a multi-user tool

4.2.2 Enable Server to run one Acumen instance per web client

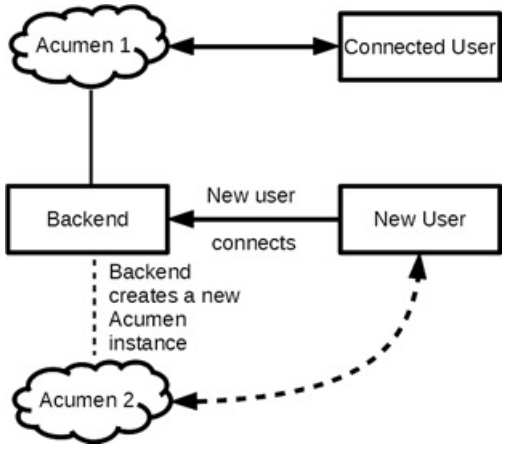

Figure 4.2: Structure implementing multiple Acumen instances

However, the first solution, even if it is the more optimal, long-lasting solution, requires colossal work. A more time-saving solution could be to

ask the back-end server to run one instance of Acumen for each web client attempting connection. The main limitation would be of course that it would generate a huge workload on the server, and Heroku free service would not be the ideal solution then. To make this solution viable, in any case Acumen should be made more lightweight and the "buffer communication" should be replaced by an optimal communication.

4.3 Implement online database

On Acumen desktop application, the user could save and open his own files from the local machine he was working on. However, a web page does not have the same rights as a desktop application. For security reasons, a web page cannot easily access nor make modifications on a local machine. That is precisely why integrating a database would be interesting. In fact, a database with a login process would enable the user to save his own files on the cloud and retrieve it from any machine, mimicking what he would do on the desktop application.

Chapter 5

Conclusion

In spite of many unforeseen events, most core features of Acumen has successfully been deployed on Heroku and is available on the web. The structure we chose allowed us to handle the web interface with minimal changes to Acumen source code, facilitating future updates. This project is far from being completed, but clear guidelines have been defined to Improve Web Acumen.

Personally, I wanted to challenge myself on this project, being very curious about web development but knowing that it deviated from my studies. The fact that a working product, although imperfect, emerged from this internship was very rewarding for me. The pandemic that I initially saw as a greatly unfortunate event has given me the opportunity to gain insights on what type of organisation and tools would be useful in this case. As more and more IT projects gather contributors all around the world, I think this project will be an important experience in my future career.

References

[1] Walid M. Taha, Abd-Elhamid M. Taha, Johan Thunberg, *Cyber-Physical Systems: A Model-Based Approach*

[2] Deploying a full stack Scala application on Heroku, Antoine Doeraene, [Online] *https://medium.com/@antoine.doeraene/deploying-a-full-stack-scala-application-on-heroku-6d8093a913b3*

[3] Acumen GitHub repository [Online] *https://github.com/maroneal/acumen*

[4] Web Acumen GitHub repository. [Online] *https://github.com/affraike/full_scale_scala_app*

List of Figures

1.1 Acumen desktop IDE 2
1.2 Acumen Web Interface 2

3.1 Illustration of Acumen-front-end communication 7
3.2 Illustration of front-end-back-end communication 7
3.3 Plot window 9
3.4 3D window 9

4.1 Possible structure when Acumen becomes a multi-user tool . . 11
4.2 Structure implementing multiple Acumen instances 11

ENSTA Bretagne

RAPPORT D'EVALUATION
ASSESSMENT REPORT

Merci de retourner ce rapport par courrier ou par voie électronique en fin du stage à :
At the end of the internship, please return this report via mail or email to:

ENSTA Bretagne – Bureau des stages - 2 rue François Verny - 29806 BREST cedex 9 – FRANCE
00.33 (0) 2.98.34.87.70 / stages@ensta-bretagne.fr

I - ORGANISME / *HOST ORGANISATION*

NOM / *Name* Högskolan i Halmstad

Adresse / *Address* Kristian IV:s väg 3, 30118 Halmstad Sweden

Tél / *Phone (including country and area code)*

Nom du superviseur / *Name of internship supervisor*
Sundas Munir

Fonction / *Function* Doktorand

Adresse e-mail / *E-mail address* sundas.munir@hh.se

Nom du stagiaire accueilli / *Name of intern* Kévin Affraix

II - EVALUATION / *ASSESSMENT*

Veuillez attribuer une note, en encerclant la lettre appropriée, pour chacune des caractéristiques suivantes. Cette note devra se situer entre **A (très bien)** et **F (très faible)**
*Please attribute a mark from **A (excellent)** to **F (very weak).***

MISSION / *TASK*

- La mission de départ a-t-elle été remplie ? A B C D E F
  Was the initial contract carried out to your satisfaction? Yes

- Manquait-il au stagiaire des connaissances ? ☐ oui/*yes* ☑ non/*no*
  Was the intern lacking skills?

  Si oui, lesquelles ? / *If so, which skills?*

ESPRIT D'EQUIPE / *TEAM SPIRIT*

- Le stagiaire s'est-il bien intégré dans l'organisme d'accueil (disponible, sérieux, s'est adapté au travail en groupe) / *Did the intern easily integrate the host organisation? (flexible, conscientious, adapted to team work)*

  A B C D E F

  Yes, Kevin integrated very well with the team. It was easy to communicate with him. He understood all tasks and cooperated effortlessly.

Souhaitez-vous nous faire part d'observations ou suggestions ? / *If you wish to comment or make a suggestion, please do so here*

COMPORTEMENT AU TRAVAIL / *BEHAVIOUR TOWARDS WORK*

Le comportement du stagiaire était-il conforme à vos attentes (Ponctuel, ordonné, respectueux, soucieux de participer et d'acquérir de nouvelles connaissances) ?
Did the intern live up to expectations? (Punctual, methodical, responsive to management instructions, attentive to quality, concerned with acquiring new skills)?

A B C D E F

Yes, Kevin met all the expectations and he possesses all the above-mentioned qualities.
His progress along with learning new skills is impressive.

Souhaitez-vous nous faire part d'observations ou suggestions ? / *If you wish to comment or make a suggestion, please do so here* _____

_____

INITIATIVE – AUTONOMIE / *INITIATIVE – AUTONOMY*

Le stagiaire s'est –il rapidement adapté à de nouvelles situations ? A B C D E F
(Proposition de solutions aux problèmes rencontrés, autonomie dans le travail, etc.)

Did the intern adapt well to new situations? A B C D E F
(eg. suggested solutions to problems encountered, demonstrated autonomy in his/her job, etc.)

Yes, infact he outperformed and proved to be an excellent candidate.

Souhaitez-vous nous faire part d'observations ou suggestions ? / *If you wish to comment or make a suggestion, please do so here* _____

_____

CULTUREL – COMMUNICATION / *CULTURAL – COMMUNICATION*

Le stagiaire était-il ouvert, d'une manière générale, à la communication ? A B C D E F
Was the intern open to listening and expressing himself /herself?

Yes, Kevin is open to express his concerns and listen to critics even. He takes instructions seriously and performs task efficiectly.

Souhaitez-vous nous faire part d'observations ou suggestions ? / *If you wish to comment or make a suggestion, please do so here* _____

_____

OPINION GLOBALE / *OVERALL ASSESSMENT*

❖ La valeur technique du stagiaire était : A B C D E F
Please evaluate the technical skills of the intern:

Kevin is a highly-skilled student and posses professional behavior. He can easily understand new and complicated technical tasks.

III - PARTENARIAT FUTUR / *FUTURE PARTNERSHIP*

❖ Etes-vous prêt à accueillir un autre stagiaire l'an prochain ?

Would you be willing to host another intern next year? ☑ oui/*yes* ☐ non/*no*

Fait à _____, le _____
In _____, on _____

Signature Entreprise _____ Signature stagiaire
Company stamp _____ *Intern's signature*

HALMSTAD UNIVERSITY
School of Information Technology
Box 823, 301 18 Halmstad
035-16 71 00

Merci pour votre coopération
We thank you very much for your cooperation

Version du 05/04/2019