PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
وزارة التعـليـم العـالـي و البـحـث العـلـمي
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH
جـامعـة عمـار ثليـجـي بالاغـواط
UNIVERSITY OF AMAR TELIDJI LAGHOUAT
كـليـة العـلـوم
FACULTY OF SCIENCES
COMPUTER SCIENCE DEPARTMENT

MASTER THESIS

Domain: Mathematics and Computer Science

Field: Computer Science

Option: Networks, Systems and Distributed Applications

By:

Azizi Fadila **Gourine Khadidja Narimane**

THEME

Load Testing and Load Balancing For Alumni Web application Using JMeter

Thesis defended publicly on 20-06-2023

Jury Members

Dr. BENSAAD Mohamed Lahcen	**Associate Professor**	**President**
Dr. GUIBADJ Fatna	**Associate Professor**	**Examiner**
Dr. BOUKEHILA Ali	**Associate Professor**	**Examiner**
Dr. HAMINI Narjes	**Associate Professor**	**Supervisor**

N ..... University Year 2022/2023

AKNOLEDGEMENT

We want to thank God for all the blessings and for giving us the patience and dedication that have us to accomplish this humble job.

This endeavor would not have been possible without our supervisor Dr. Nardjes Hamini, a professor in the Department of Computer Science at the Faculty of Science, University of Aghouat for her invaluable feedback and most for her kindness.

and i would thanks every single personne who helped and advices us through this hard but beautiful journey, we could not have undertaken this without the help or the sympathy of them.

Lastly, I would like to express my deepest appreciation to all my Professors at Computer Science Department in Laghouat University who earnestly contributed in my education and training.

DEDICATION

First of all I thank Allah the Almighty for having given me the will, the health and the courage to carry out this work.

I dedicate this work: To the one who gave me life, the symbol of tenderness, who sacrificed himself for my happiness and my success, to my dear mother and my dear father may God have mercy on him, school of my childhood, who watched throughout my life to encourage me, to give me help and protection.

To my dear sisters: Manal, Hadjer.
To my dear brothers: Nabil, Walid.
To my nephews: Mohammed, Younes and Mohammed, Ines, Meriem.

To my supervisor: Dr Hamini Nardjes, To my dear friends and work colleagues And to all those who contributed to the realization of this work.

Gourine Khadidja Narimane

DEDICATION

I want to start by thanking Allah for giving me the ability to finish this work.

I dedicate this work: to my lovely mother Slimi Dawia(may Allah keep her safe) how gives me all the love, support, and unwavering encouragement during my live.

To My father .
To My brother Hossam and My lovely sisters Imane, Fatima, Hafsa, Karima.

A greate thanks to my teacher, Hamini Narjes and project partner, Khadija Narimane Gourine, for ther invaluable help and significant contribution in completing this work.
Finally, I had like to express my gratitude to My close friends, and everyone who helped me directly or indirectlys in completing this project.

Azizi Fadila

ملخص

تسلط هذه الرسالة الضوء على تطوير تطبيق الويب الخاص بالخريجين لجامعة الأغواط باستخدام Angular و MySQL كأدوات تطوير. الهدف الرئيسي لموقع الخريجين هو منح الخريجين منصة للتواصل ودعم بعضهم البعض والوصول إلى الخدمات بما في ذلك ملفات تعريف الخريجين وصفحات الأحداث ولوحة الوظائف. لاختبار وضمان الأداء العالي لتطبيق الخريجين لخدمة مجتمع الخريجين بشكل فعال والتعامل مع أحمال المستخدمين المختلفة. استخدمنا Apache Jmeter أداة لإجراء اختبار التحميل من أجل تقييم أداء التطبيق في ظل العديد من سيناريوهات التحميل التي تشبه إلى حد كبير أنماط الاستخدام في العالم الحقيقي ، ويتم قياس الأداء وفقًا لمعايير معينة مثل وقت الاستجابة والإنتاجية ومعدل الخطأ. كانت نتيجة اختبار الحمل في سيناريوهات الحمل العادية مرضية ، وكان أداء النظام موثوقًا به دون أخطاء أو أعطال ، بينما لم تكن نتيجة اختبار الحمل في سيناريوهات الحمل العالي مرضية ، لم يتمكن التطبيق من التعامل مع ارتفاع الحمل المفاجئ ، مما أدى إلى زيادة أوقات الاستجابة ومعدل الأخطاء وتقليل أداء التطبيق. مع هذه المشكلات ، يتم استخدام موازنة التحميل لتحسين أعلى مستوى من الأداء والتوافر والموثوقية لتطبيق Alumni من خلال توزيع حركة مرور الشبكة بين الخوادم باستخدام محاكي Apache Jmeter .

الكلمات الرئيسية: الخريجون ، تطبيق الويب ، التطوير ، Angular ، موازنة الحمل ، اختبار الحمل ، JMeter ، قياس الأداء ، خادم الويب.

Abstract

This thesis highlights the development of the alumni web application for the university of laghouat using Angular and MySQL as development tools. The main goal of the alumni website is to provide graduates with a platform to communicate, support each other, and access various services, including alumni profiles, event pages, and a job board. To test and ensure the high performance of the alumni application in effectively serving the alumni community and handling different user loads, we used Apache JMeter, a tool for load testing. This allowed us to evaluate the application's performance under various load scenarios that closely resemble real-world usage patterns. Performance was measured based on parameters such as response time, throughput, and error rate. The results of the load testing in normal load scenarios were satisfactory, as the system performed reliably without errors or failures. However, the results of the load testing in high load scenarios were unsatisfactory. The application was unable to handle the sudden load spike, resulting in increased response times, error rates, and degraded performance of the application. To address these issues, load balancing was employed to improve the overall performance, availability, and reliability of the alumni application. This involved distributing network traffic among servers using the Apache JMeter simulator.

Keywords: Alumni, Web application, development, Angular, Load balancing (LB), load testing, JMeter, Performance Measurement, Web Server.

Résumer

Cette thèse met en lumière le développement de l'application web Alumni pour l'université de laghouat en utilisant Angular et MySQL comme outils de développement. L'objectif principal du site Web des diplômés est de fournir une plate-forme permettant aux diplômés de communiquer, de se soutenir mutuellement et d'accéder à des services, notamment des profils d'anciens étudiant, des pages d'événements et un tableau des offres d'emploi. Nous avons effectué des tests et garanti les hautes performances de l'application pour servir efficacement la communauté des anciens étudiant et gérer différentes charges d'utilisateurs. Nous avons utilisé Apache JMeter, un outil de test de charge, pour évaluer les performances des applications dans plusieurs scénarios de charge qui ressemblent étroitement aux modèles d'utilisation réels. Les performances sont mesurées sur certains paramètres tels que le temps de réponse, le débit et le taux d'erreur. Les résultats des tests de charge dans des scénarios de charge normale étaient satisfaisants, le système a fonctionné de manière fiable sans erreurs ni pannes. Cependant, les résultats des tests de charge dans les scénarios de charge élevée n'étaient pas satisfaisants. L'application n'a pas été en mesure de gérer le pic de charge soudain, ce qui a entraîné une augmentation des temps de réponse, du taux d'erreur et une dégradation des performances de l'application. Pour résoudre ces problèmes, nous avons utilisé l'équilibrage de charge pour améliorer les performances, la disponibilité et la fiabilité de l'application Alumni. Celle-ci consiste à répartir le trafic réseau entre les serveurs à l'aide du simulateur Apache JMeter.

Mots clés: Alumni, Application Web, développement, Angular, Load balancing (LB), test de charge, JMeter, Performance Measurement, Web Server.

Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

- LB : Load Balancing
- TCP : Transmission Control Protocol
- FTP : File Transfer Protocol
- DNS : Domain Name System
- SSL : Secure Socket Layer
- UUID : Universally Unique Identifier
- HTML : Hypertext Markup Language
- TLS : Transport Layer Security
- VMs : Virtual Machines
- IIS : Internet Information Services
- API : Application Programming Interface
- VS Code : Visual Studio Code
- UI: User Interface
- SPA: Single Page Application
- UML: Unified Modeling Language
- OS: Operating System
- SMTP: Simple Mail Transfer Protocol

- IP: Internet Protocol
- GUI: Graphical User Interface
- CSV: Comma-Separated Values
- PC: Personal Computer
- RMI: Remote Method Invocation
- XSS: Cross-Site Scripting
- SQL: Structured Query Language

CHAPTER 1

GENERAL INTRODUCTION

1.1 Introduction

We live in a digital era where information is available at our fingertips, Also the internet usage has been growing at a staggering rate. Internet users are browsing tremendously in search for information. Over the past few years, the web world has transformed the ways we live and work, It has increased convenience and enhanced our lifestyle.

In this context, to accomplish various goals, having a website that offers a seamless user experience emerged, as a powerful tool. The universities worldwide have recognized the importance of web development services to connect with a massive number of students and enhance communication. A website allows students and alumni to easily access the necessary information in real time, offering convenience and efficiency.

In nowadays, having a feature rich website is crucial for universities to effectively organize events, facilitate job searches, and enable seamless communication through social networks. So the development of an efficient and high-performing alumni website, has become a essential concern for educational institutions globally.
In order to encourage connections, collaboration, and involvement among former students, discuss professional possibilities, and contribute to the expansion and development of their local communities, the alumni platform offers a valuable area for alumni.

This project focuses on the development of an alumni website for the university of laghouat and explore the conceptualization, design, and implementation of an alumni site. The purpose of this research is to examine how an alumni site can effectively connect graduates, provide

valuable services, and support their personal and professional growth. The creation of an alumni website makes use of MySQL, a popular database management system, and Angular, a popular frontend framework known for its performance, robustness and adaptability.

As the alumni website gains popularity and more users access it at once, it becomes more crucial to guarantee excellent performance and responsiveness. As a result, managing the growing workload and dividing it equally among the resources at hand presents a challenge. This is where load balancing enters the picture. A key component of the speed and scalability of online applications is load balancing, which is the process of evenly dividing incoming network traffic among a number of backend servers.

This thesis main goal is to examine load balancing techniques and how they affect the alumni website's performance. Using Apache JMeter, an open source Java application created to load test functional behavior and analyze performance metrics like response time, throughput, and error rates of the load-balanced system, one may simulate various load scenarios.

This study has a good reflection for the design and improvement of alumni websites because it addresses the need for an effective load balancing solution in the context of the websites by evenly distributing network traffic among a variety of resources that improve the website's responsiveness, availability, and user satisfaction.

1.2 Problem Statement

Nowadays, high-traffic websites must serve hundreds of thousands, if not millions of simultaneous request from users and delivering the appropriate reply. Those replies consist of images, texts, videos, audio and application data, all in a fast and reliable manner. Therefore, web developers try to generate powerful structures and efficient systems for web servers in order to satisfy internet users and the web servers from being overworked[1].

As the alumni network grows and demand increases, the performance and scalability of the alumni website become critical factors in providing a seamless user experience. Therefore, it becomes challenging to manage the total traffic with growing users. In order to handle this high traffic one simple solution is to add number of servers to handle requests. However, this approach can lead to an uneven workload distribution, with some servers being heavily loaded while others are underutilized. Achieving a balanced distribution of user requests across all

servers becomes a challenge here.

The absence of a load balancing mechanism poses several problems for the alumni website These are some of the main problems:

1. **Inefficient Workload Distribution:** This arises from the absence of a load balancing mechanism, leading to an uneven distribution of user requests across the available servers. This may lead to some servers become overloaded, while others are underutilized, leading to degraded performance and increased response times.

2. **Scalability Challenges:** As the user number increases or during peak periods, the website's ability to effectively handle a larger volume of concurrent users becomes limited due to the lack of load balancing.

3. **Risk of Downtime and Poor User Experience:** The absence of a load balancing mechanism increases the risk of server failures and downtime, which can disrupt the website's availability and negatively impact the user experience and limit their satisfaction.

To address these issues, the implementation of a load balancing mechanism is necessary. The main purpose is preventing a server overload and maintaining a healthy amount of activity in each server and to ensure the alumni website's optimal performance, scalability, and availability, the load balancing mechanism will help maximize resource utilization, reduce response times, and reduce the risks associated with sudden increases in traffic and server failures. Therefore, the primary objective of this thesis is to design and implement a robust load balancing strategy for the alumni website using Angular, MySQL, and JMeter simulator.

1.3 Thesis Organization

- **Chapter 1:**
  the first chapter provides an introduction to the thesis and presents an overview of the research objectives, problem statement, and the organization and structure of the thesis.

- **Chapter 2: Generalities on load balancing**
  The The Generalities on load balancing chapter illustrates the research on load balancing concepts, techniques, and their application in web servers and the cloud. It also explores different type of load balancers and algorithms.

- **Chapter 3: Design of the Alumni application**

  This chapter, provides an overview of the front-end implementation in Angular, including the user interface design.

- **Chapter 4: Development of the Alumni application**

  It discusses the integration of the MySQL database as the back-end, outlining the database schema.

- **Chapter 5: The Methodology, Results and Analysis**

  In this chapter, by using Apache JMeter as test tool we can describes the load testing methodology, including the configuration of test scenarios. Also, the results gained from the load testing experiments are presented and analyzed. It includes a test of performance metrics such as response time, throughput, and error rates. The chapter also describes the load balancing techniques, providing a look into the efficiency of the load balancing in improving performance and scalability.

- **Chapter 6: Conclusion and Future Work**

  In this chapter, the importance of the study and its contributions the load balancing in web applications are highlighted. By summarizes the main findings of the research and draws conclusions from the facts and analysis. The chapter advises changes to the load balancing technology utilized by the alumni website and offers suggestions for prospective new study areas.

CHAPTER 2

GENERALITIES ON LOAD BALANCING

2.1 Introduction

Modern high traffic websites face the challenge of handling numerous concurrent requests from users or clients and delivering the appropriate content quickly and reliably. To effectively manage such high volumes, a recommended approach in modern computing involves adding more servers. To optimize performance and assure even workload distribution, a load balancer is typically deployed in front of these servers. The load balancer routes client requests across all available servers, maximizing speed and capacity utilization whilst forbidding any single server from becoming overloaded and degrading performance. In the case of a server fail, the load balancer automatically redirects traffic to the residual online servers. Also, when a new server is added to the server group, the load balancer seamlessly starts directing requests to it. In simpler terms, the load balancer acts as a traffic manager, evenly distributing requests among servers for efficient and reliable website performance [2]. By gaining a clear understanding of load balancing concepts and its role, we can truly appreciate the significant impact it has on website.

In this chapter, we will take a look at load balancing concepts and techniques. First, we will see the load balancing in web applications and its benefits. Lastly,load balancing in cloud.

2.2 Load Balancing Concepts and techniques

A fundamental procedure that involves distributing network traffic among several servers is load balancing. Its main objective is to ensure that no single server is overloaded with demand.[3] By distributing the workload, load balancing enhances the performance and accessibility of apps, websites, databases, and other computing resources. It is also commonly

used for other services such as File Transfer Protocol (FTP) sites and Domain Name System (DNS) servers[4].
From the user's perspective, load balancing acts as an imperceptible intermediary between a client and a collection of servers. It operates at different layers of the network, typically at layers 4-7.

The layer 4 directing traffic based on network data and transport layer protocols,(IP address and TCP port). Layer 7 expands load balancing with content switching, enabling routing choices based on elements including HTTP headers, UUIDs, SSL session IDs, and HTML form data. L4 and L7 capabilities are expanded to servers at other sites through Global Server Load Balancing (GSLB)[5].

2.2.1 load balancers

The load balancing is facilitated by an application or device known as a load balancer. Load balancer role is to manage the flow of requests between the server and devices and it's used to increase capacity (concurrent users) and reliability of applications[7]. The LB can take the form of either hardware-based or software-based . Hardware load balancers necessitate the installation of a specialized device dedicated to load balancing tasks. Conversely, software-based load balancers can be deployed on various platforms such as servers, virtual machines, or within cloud environments.
LB's employs a predetermined pattern, referred to as a load balancing algorithm or method, to distribute traffic evenly among servers. Various algorithms utilize distinct techniques to manage this process[8].

In today's modern application, load balancers have become indispensable components, as they not only ensure efficient traffic distribution but also offer functionalities such as security and application acceleration.

2.2.2 Basics of load balancer

Load balancers sit between the application servers and the users on the internet.Once the load balancer receives a request, it determines which server in a pool is available and then routes the request to that server[4].

- The client(application or browser) initiates a request and tries to connect with a server.

- A load balancer receives the request and based on one of the algorithms, It routes the request to one of the servers that is available in a servers group.The load balancer implements this routing operation at the same time for all incoming requests.
- The selected server receives the connection request and responds to the client through the load balancer.
- The load balancer receives the response and matches the client's IP address with that of the specified server. Then the response is forwarded to the client[4].

2.2.3 Hardware vs software load balancers

Hardware load balancers are physical devices or appliances that direct traffic to servers based on criteria such as existing connections, processor utilization, and server performance.

Hardware load balancers include proprietary firmware that requires maintenance and updates as new versions and security patches are released. Hardware load balancers typically offer better performance and control, but require some level of proficiency for proper management and maintenance. These load balancers are less flexible and scalable, so there is a tendency to over provision hardware load balancers[4].

Software load balancers usually are easier to deploy than hardware versions. They also tend to be more cost effective and flexible, and they are used in conjunction with software development environments. The software approach gives you the flexibility of configuring the load balancer to your environment's specific needs. The boost in flexibility may come at the cost of having to do more work to set up the load balancer. Compared to hardware versions, which offer more of a closed-box approach, software balancers give you more freedom to make changes and upgrades[4].

2.2.4 Types of load balancers

1. **Network Load Balancer (Layer 4 Load Balancer)**

   Based on the network variables like IP address and destination ports[9][10], network load balancing is the distribution of traffic at the transport level through the routing decisions. Based on the packet header. Specifically, they looked at the 5-tuple source IP, destination IP, source port, destination port, and IP protocol. This is the entry of the network server load balancer or Layer 4 load balancer. Network Load Balanc-

ing cares only about the network layer information and directs the traffic on this basis only.

2. **Application Load Balancer (Layer 7 Load Balancer)**

   These load balancers distributes the requests based on multiple parameters at the application level[9][10]. These load balancers look at the content such as the URL, HTTP header, and other things to make load balancing decisions and distributes the server load based on the decision arising from a combination of several variables.

3. **Global Server Load Balancer(layer 4-7 Load Balancer)**

   Global server load balancing (GSLB) is actually a different technology than the traditional layer 4-7 load balancer[9][10]. GSLB is based on DNS and acts as a DNS proxy to provide responses based on GSLB load balancing algorithms in real time. A global load balancer is designed to distribute traffic across multiple data centers or regions. It considers factors like user location, latency, and server availability to direct requests to the nearest or least loaded data center.

2.3 Load balancing algorithms

A load balancing algorithm serves as the logic implemented in a load balancer to distribute network traffic across servers.Two primary methods exist for load balancing algorithms, dynamic and static. Dynamic load balancing algorithms consider the current state of each server and distribute traffic accordingly. On the other hand , static load balancing algorithms distribute traffic without factoring in server states. Some static algorithms send an equal amount of traffic to each server in a group[14][8].

1. **Static load balancing algorithms**

   - Round robin load balancing:

     spreading client information across a group of servers .Is the approach of Round robin algorithm ,this technique is followed in a small network where servers are limited.
     In this algorithm we have multiple clients and servers, when a client initiates request and passes it to the server, the first server is allocated to the request, when the second request is initiated it is allocated to the second server and third request come

to the third server, but if server receive fourth request then it is allocated to the first server and create a ring type allocationv, the number of requests that can be processed is a limitation of the server[11].

- Weighted round robin:

  Allows an administrator to assign different weights to each server. Servers deemed capable of handle more traffic will receive slightly more[12], the weights can be assigned based on factors such as server processing power or total bandwidth or can be configured in DNS records.

- source IP hash:

  The source IP hash load balancing algorithm uses the client's source and destination IP addresses to generate a unique hash key to tie the client to a particular server[13]. As the key can be regenerated if the session is interrupted, this method of load balancing can ensure that the client is directed to the same server that it was using previously.

- fixed weighting load balancing:

  In the fixed weighting algorithm, The administrator assigns a weight to each server based on criteria representing each server's relative traffic handling capability[14]. The server with the highest weight will receive all of the traffic, if the highest weight server fails, all traffic will be passed to the next highest weight application server. The algorithm does not consider the current state or capacity of the servers, instead, it follows a predetermined distribution pattern based on the assigned weights.

2. **Dynamic load balancing algorithms**

- Least connection:

  In cases where application servers have similar specifications[15], one server may get overloaded due to longer-lived connections. This algorithm considers the dynamic connection load and doesn't send requests to servers that cannot handle them.Checks

which servers have the fewest connections open at the time and sends traffic to those servers. This assumes all connections require roughly equal processing power.

- Weighted least connection:

  Weighted least connection extends the least connection algorithm[15], to account for differing application server characteristics,it gives administrators the ability to assign different weights to each server based on relative processing power and available resources. Load balancing decisions get based on active connections and the assigned server weights

- Weighted response time:

  Calculates Averages response time for each server and combines it with the number of connections opened by each server to determine where to send traffic. By sending traffic to the servers with the fastest response time, the algorithm ensures faster service for users[14].

- Resource-based:

  load balancing makes decisions based on status indicators retrieved by the load balancer from the application servers. Utilising Specialized software called an "agent", an agent running on each server measures server's available CPU and memory. Then, The load balancer distributes load based on what resources each server has available at the time[14].

- URL Hash load balancing:

  The URL hash load balancing algorithm is similar to the source IP hashing, except that the hash created is based on the URL in the client request. This ensures that any client requests to a particular URL always go to the same back-end server[15]. By consistently routing requests to the same server, the URL Hash algorithm can be useful for maintaining session persistence or ensuring that certain resources or data associated with a particular URL are consistently served by the same server.

2.4 Load Balancing in Web Applications servers

In web application servers the load balancers are positioned between the backend servers and the firewall. This strategic placement allows the LB to distribute client requests among the available servers, enabling efficient handling of high traffic scenarios without experiencing any downtime and ensuring the efficient utilization of resources, maximize throughput and improve the performance and availability of the web application[16].

2.4.1 Benefits of Load Balancing on Web Servers

- Reducing downtime :

  By implementing load balancers, downtime can be significantly minimized[16]. Load balancers can prevent scenarios such as server overload, which can be caused by factors like DDoS attacks, excessive traffic, or malfunctioning servers. Additionally, load balancers support maintenance activities by redirecting traffic to other servers, ensuring that the application stays online. In the event of a failure, load balancers offer built-in redundancy by redirecting traffic to a backup system while the servers are being recovered.

- Improving scalability :

  For enhancing the scalability of websites the Load balancers play a crucial role in that[16], allowing them to perform well even during periods of heavy traffic. They efficiently distribute traffic across available servers, improving the capacity of the existing infrastructure. This approach offers a more effective way to enhance scalability

- Health checks :

  The load balancers check the health of individual servers on an ongoing basis. in case of one server becomes unresponsive or breakdown, it will be detected by health checks and automatically the load balancer forwards and send traffic to the healthy servers. This ensures that users experience minimal disruptions in the case of a server failure[16].

- SSL termination:

  The load balancer can handle SSL/TLS encryption and decryption, taking this computation-heavy task off the servers. So they can focus on processing application logic

and improving performance[16].

- Enhancing security :

  Load balancers provide a layer of security for network systems[15]. They filter and redirect malicious traffic from attackes. This helps protect the system from potential security breaches and strengthens the network system's security layer.

2.5 Cloud load balancing

Cloud load balancing, is a software-based load balancing service that distributes traffic between multiple cloud servers[17]. Many cloud providers allow customers to rent load balancing services on an as-needed basis, rather than configuring and maintaining dedicated on-premise appliances to route their traffic themselves, This process is often referred to as **load balancing-as-a-service** (LBaaS).

2.5.1 Load Balancer as a Service (LBaaS)

Load Balancer as a Service (LBaaS) is a cloud computing service that provides load balancing functionality. LBaaS uses advances in load balancing technology to meet the agility and application traffic demands of organizations implementing private cloud infrastructure. Using an as-a-service model[18]. LBaaS simplifies the process of implementing and managing load balancing infrastructure by providing a cloud-based solution. users can leverage LBaaS to provision and manage load balancers through an intuitive web interface or API.

Benefits of Load Balancing as a service

- Multi-cloud and reliabtility :
  The reliabtility of applications enhances by LBaaS, where it's distributing traffic across multiple cloud providers,in the case of a cloud provider is breakdown , LBaaS pass traffic to other available clouds, making the uninterrupted service and minimizing downtime [18].

- Reduced costs :
  LBaas requires less time , internal resources, than hardware appliances. It also has easy configuration. Administrators can easily define load balancing algorithms, settings, health checks, and other options[20].

- Scalability :

  LBaaS can provide the capability to add more servers to the server pool automatically, manage high traffic and ensur efficient utilization of resources [19].

2.6 Conclusion

In this chapter, we gained a clear understanding of the concepts, types and techniques of load balancing. Then, we explored the crucial role of load balancing in web applications. Next, We highlighted the load balancing in cloud environments, LBaaS and it's benefits in the cloud.
In the next chapter, we will delve into the design of the Alumni Application. Our exploration will begin by delving into the Requirements Specification. Then, we will shift our focus to the analysis and design phase, where we will utilize UML as a tool to design the Application. Finally, we will proceed to the design of our alumni application.

CHAPTER 3

DESIGN OF THE ALUMNI APPLICATION

3.1 Introduction

The objective for designing the Alumni application is the connection and support among graduates through an Alumni platform. Its main purpose is to provide a space where Alumnies can interact with each other and access different features. The alumni profile is one of the most important components, which allows users to create personal profiles highlighting their educational background and other relevant information. This feature alow Alumni to connect and network with graduates who share similar career path or interest. The application also involves event pages, where users can find information about upcoming reunions, social gatherings, and other events organized. Additionally, the job board provids alumnies with access to job opportunities. Through these features, the Alumni application aims to facilitate meaningful connections, enhance the Alumni experience, and support professional growth. In this chapter, we will begin by identifying the functional and non-functional requirements. Then, We will move on to the analysis and design phase, using UML as our tool for visualizing the application's structure. Finally, We will explore the design of the application.

3.2 Requirements Specification

In this part and in order to identify the services that our site must provide, we will detail the specific requirements which consist of functional needs and non-functional needs.

3.2.1 Functional Needs

- User registration : Alumni application allows users to sign up for accounts and verify them to ensure secure access.

- Email validation : The application check the validity and ownership of user email addresses to make sure they belong to the right users. The application will also verify if the email address follows the correct syntax format (user-name@univ-lugh.dz).
- Profile Management : The alumni profiles include personal information, educational background, and employment history. The application allows users to manage their profiles by provide some action (edite and publish).
- Data Validation : To make sure that data entered into the system is valid,The application verify the user inputs.
- Communication : The application allows users to communicate with other graduates.
- Event Management : Allow the administrator(alumni) the ability to organize and advertise events with a clear alumni focus, such as conferences and reunions.
- Job Board : The application provide a function on the job platform where employers can post jobs that are only open to graduates, and users can browse and apply for those jobs.
- CV Creation Interface : Application give a user-friendly interface for formatting and preparing CVs.
- Alumni Success Stories : The application allows the alumni to post their success stories to demonstrate their accomplishments and contributions to their fields.

3.2.2 Non-functional Needs

- Performance : The application program operates effectively and reacts quickly to user inputs to make an easy-to-use experience.
- Scalability : The application is create with capacity to handle an increasing volume of users, data, and concurrent activities.
- Security : To safeguard alumni data, the application stop unauthorized access, and guarantee the confidentiality, integrity, and availability of the application.
- Maintainability : With a modular and well-documented codebase(Angular) the application is designed to make it easier to maintain, update, and add new features in the future.

3.3 Analysis and design phase

3.3.1 Unified Modeling Language (UML)

UML is a standardized modeling language that can be used by all object-oriented methods, consisting of an integrated set of diagrams, developed to help system and software developers for specifying, visualizing and documenting the artifacts of software systems. UML uses mostly graphical notations to express the design of software projects, the UML goal is The creation of an abstract language that should be human-understandable and machine-interpretable[21].

UML Diagram Types

There are two main categories, structure and behavioral diagrams.

1.Structure Diagrams

- Class Diagram
- Component Diagram
- Deployment Diagram
- Object Diagram

2.Behavioral Diagrams

- Use Case Diagram
- Activity Diagram
- Sequence Diagram
- State Machine Diagram[22].

3.3.2 Design of our Alumni application

This section deals with the design phase of alumni application,this design plays an important role in shaping the functionality and user experience with the application.
To design diagrams for the application, we will utilize StarUML.
StarUML is a software modeling tool that allows us to create various types of diagrams using the Unified Modeling Language (UML)[26].

Class Diagram

Class diagram in UML describes the structure of a system by showing the system's classes, their attributes, operations, and the relationships between objects[25].

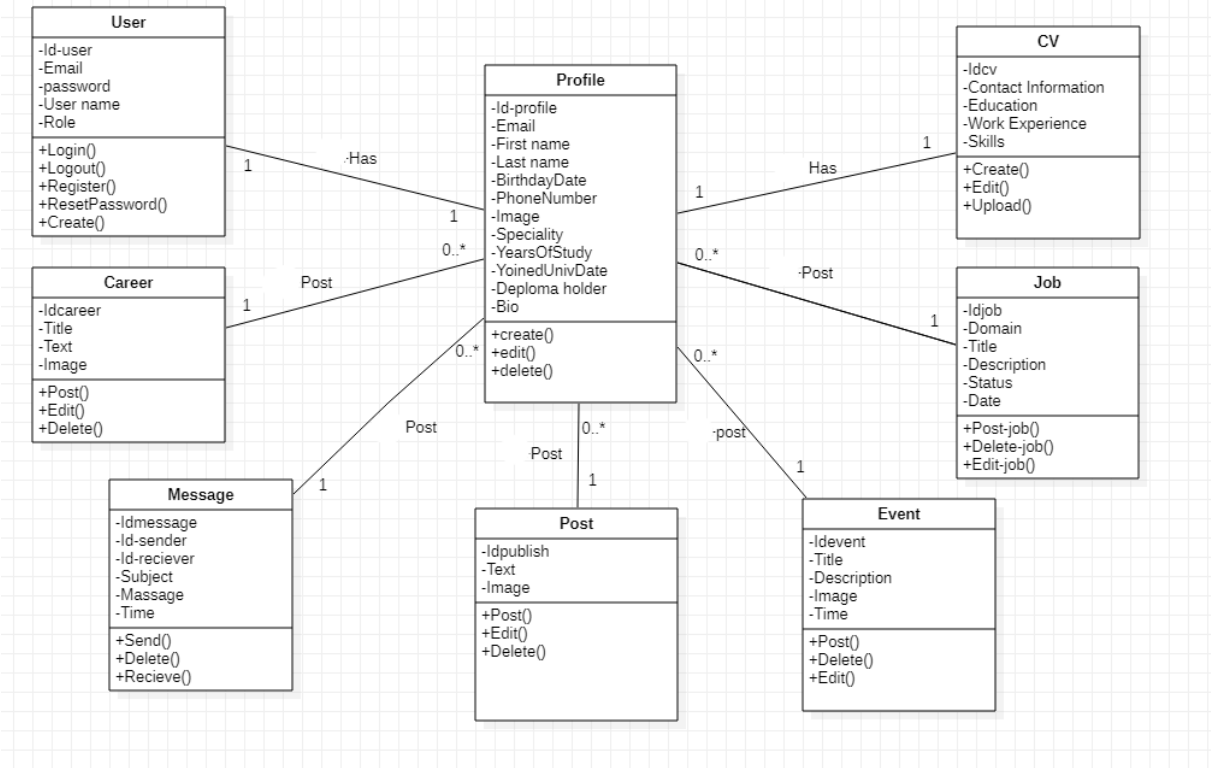

Figure 3.1: Class Diagram

Use Case Diagram

The use case diagram shows people who use the system, it gives a graphical overview of the actors involved in the system, the different functions needed by those actors and how these functions interact.

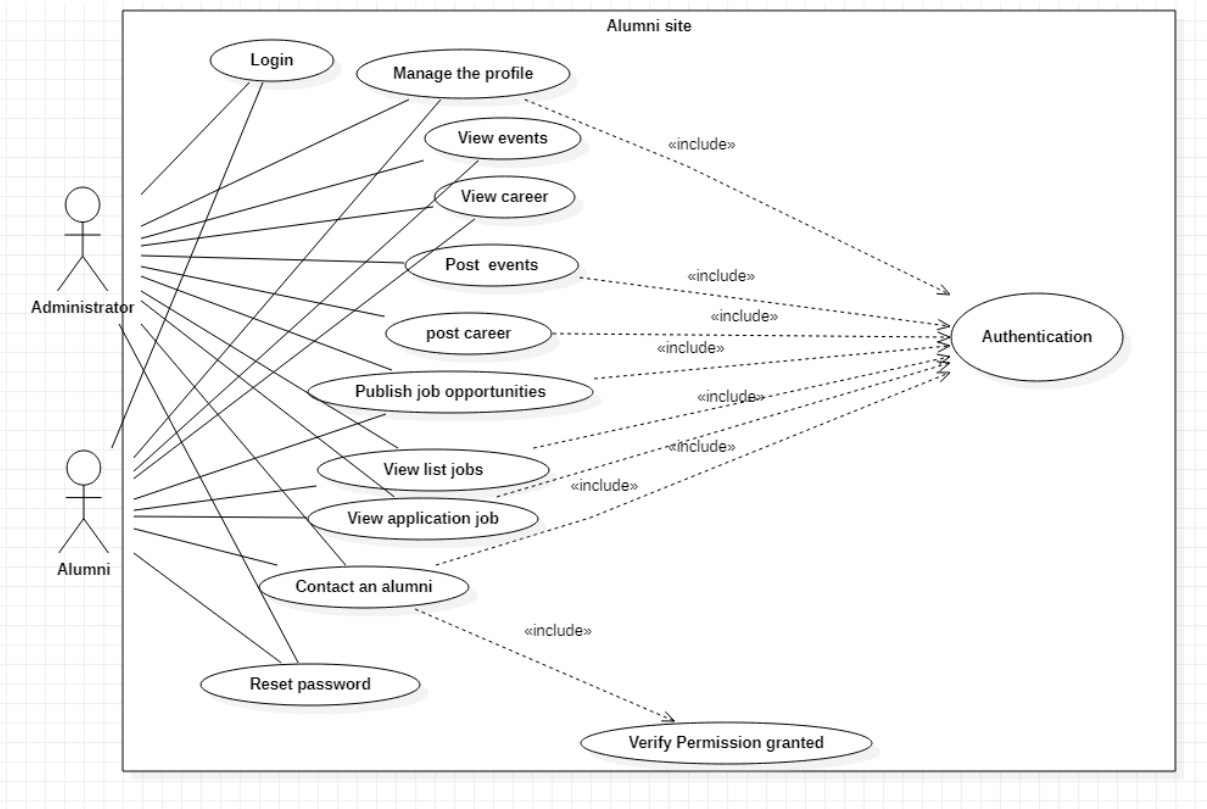

Figure 3.2: Diagram for an alumni

Sequence Diagram

A sequence diagram in UML illustrates the sequence of messages between objects in an interaction.A sequence diagram consists of objects that are represented by lifelines, and the messages that they exchange over time during the interaction[24].

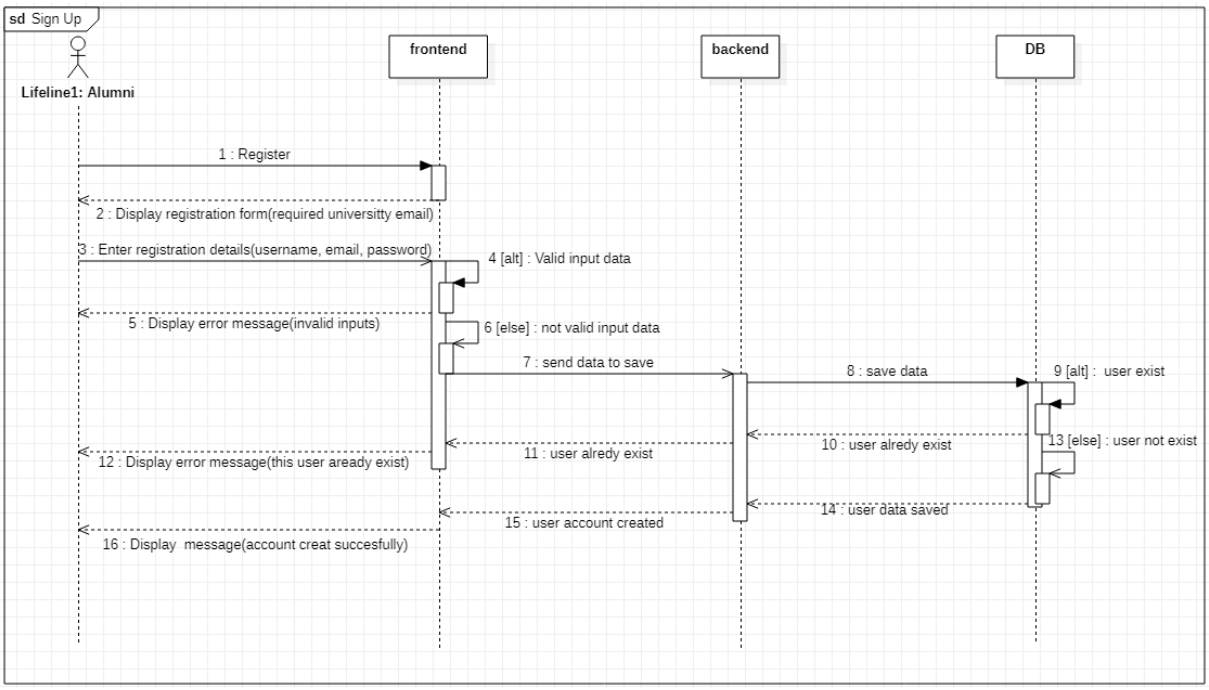

Figure 3.3: Sign up Sequence Diagram

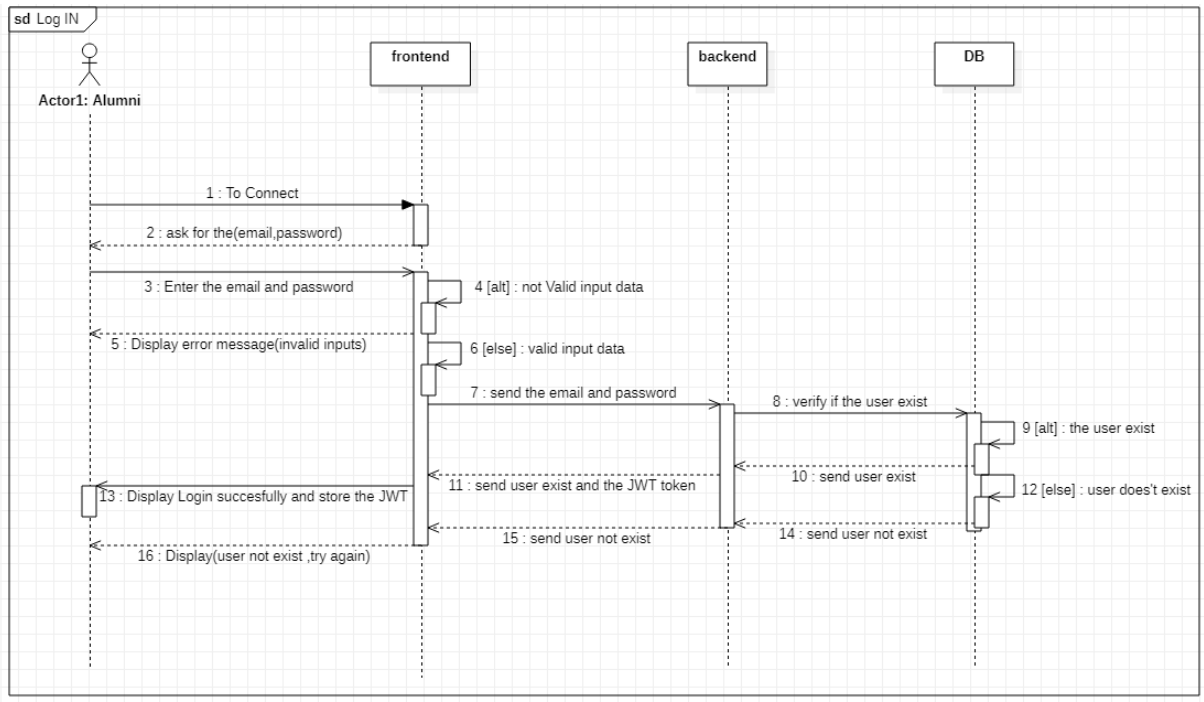

Figure 3.4: Login Sequence Diagram

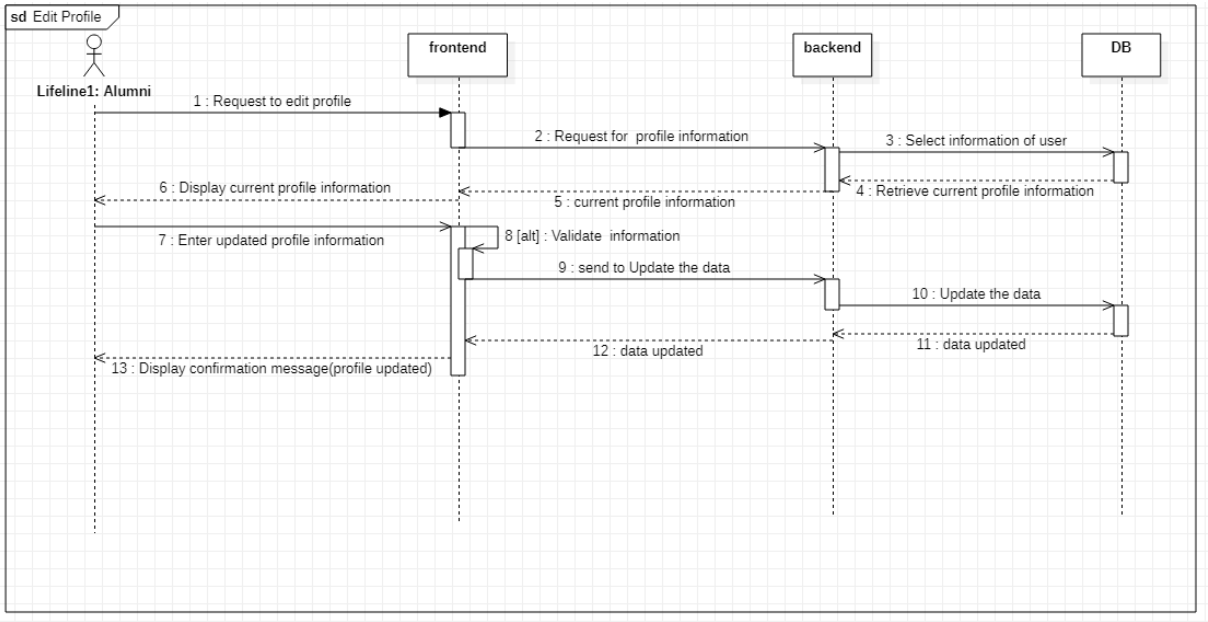

Figure 3.5: Edit Profile Sequence Diagram

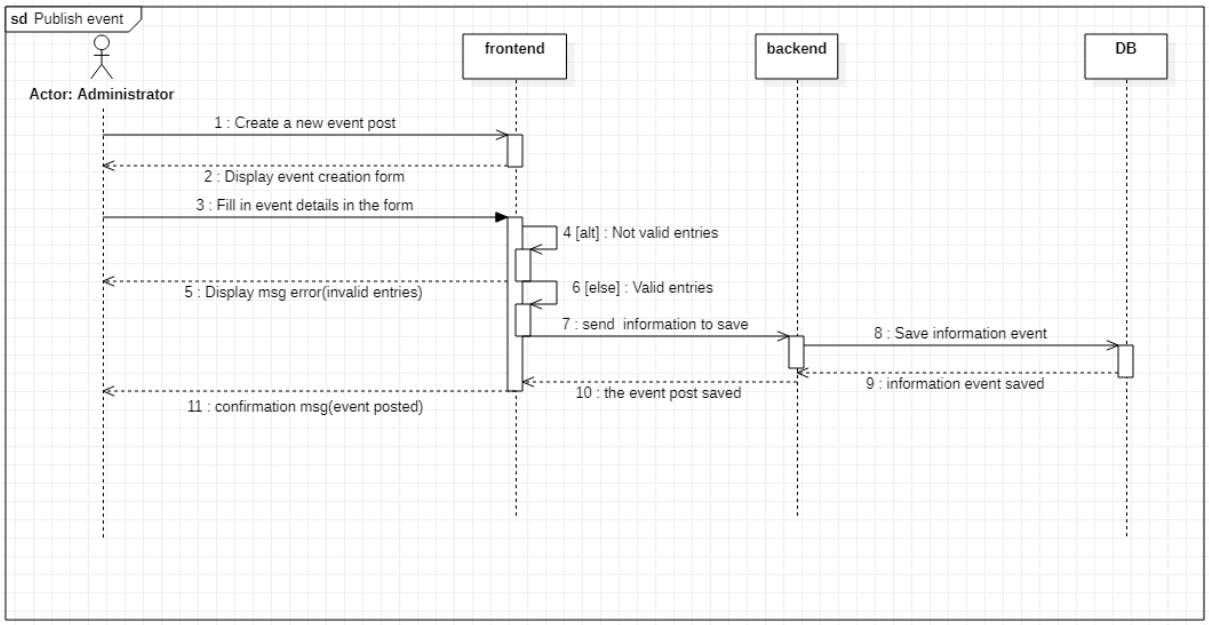

Figure 3.6: Publish Event Sequence Diagram

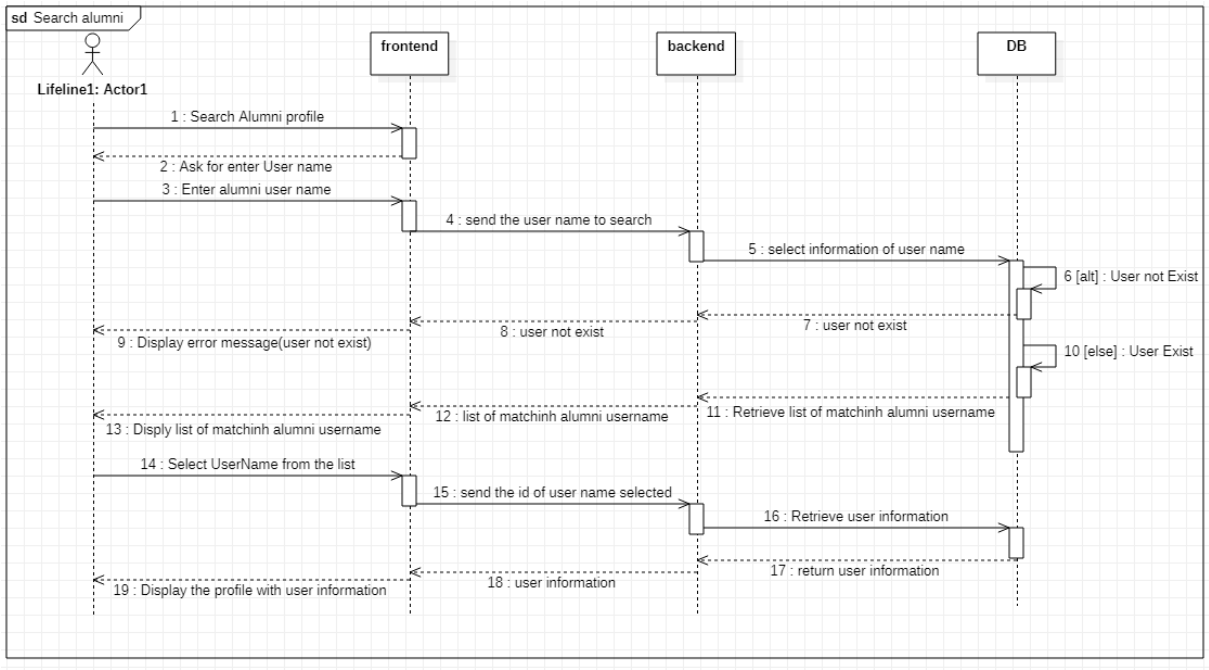

Figure 3.7: Search Alumni Sequence Diagram

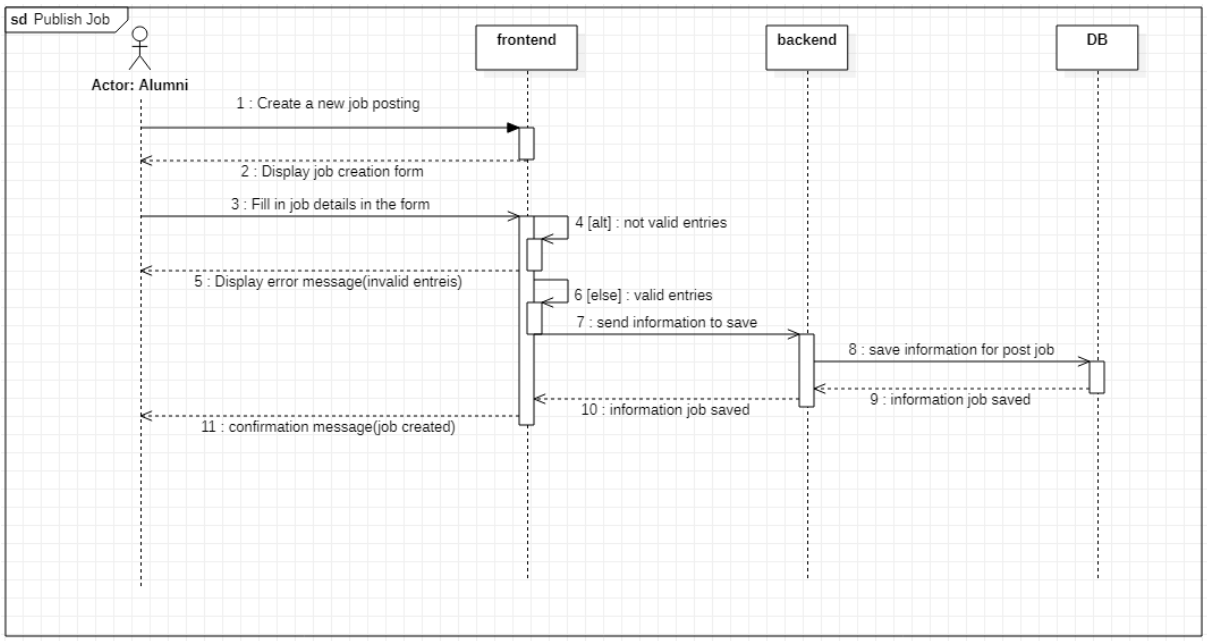

Figure 3.8: Publish Job Sequence Diagram

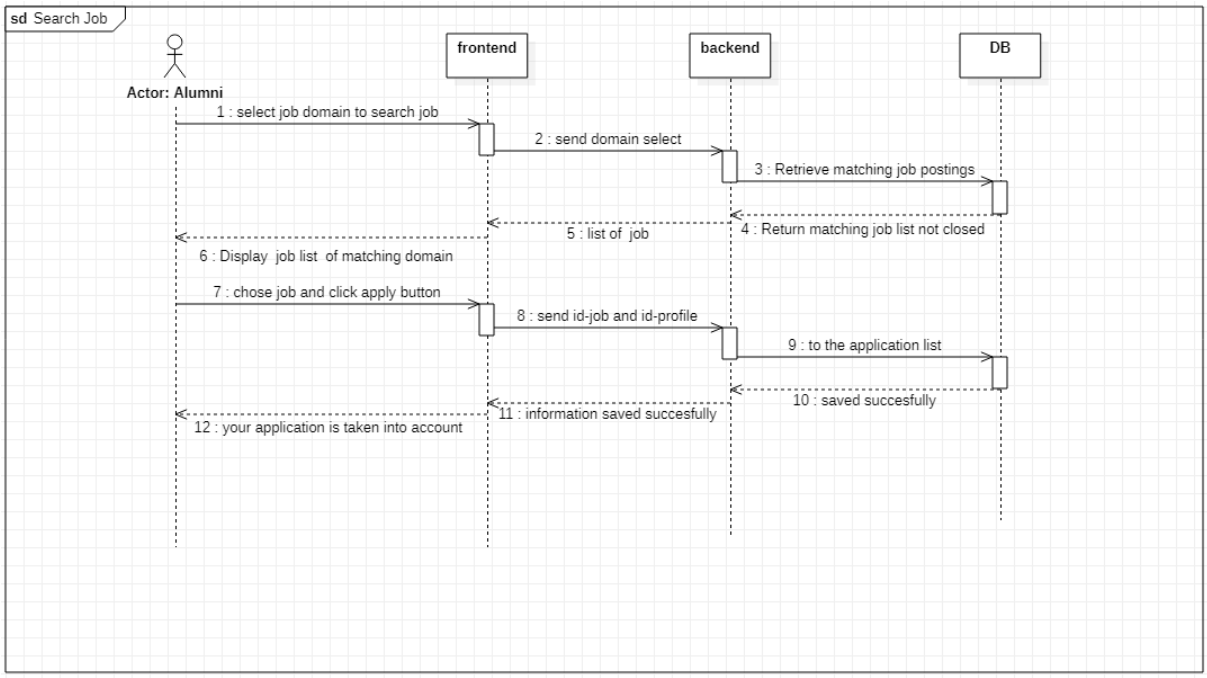

Figure 3.9: search Job Sequence Diagram

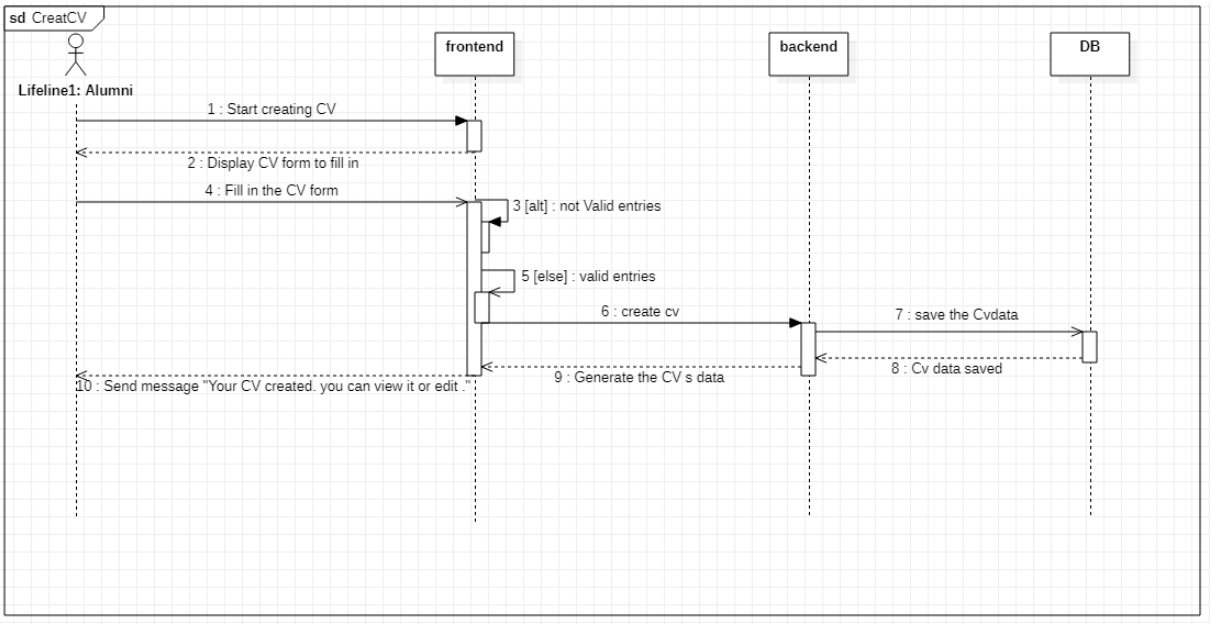

Figure 3.10: Create CV Sequence Diagram

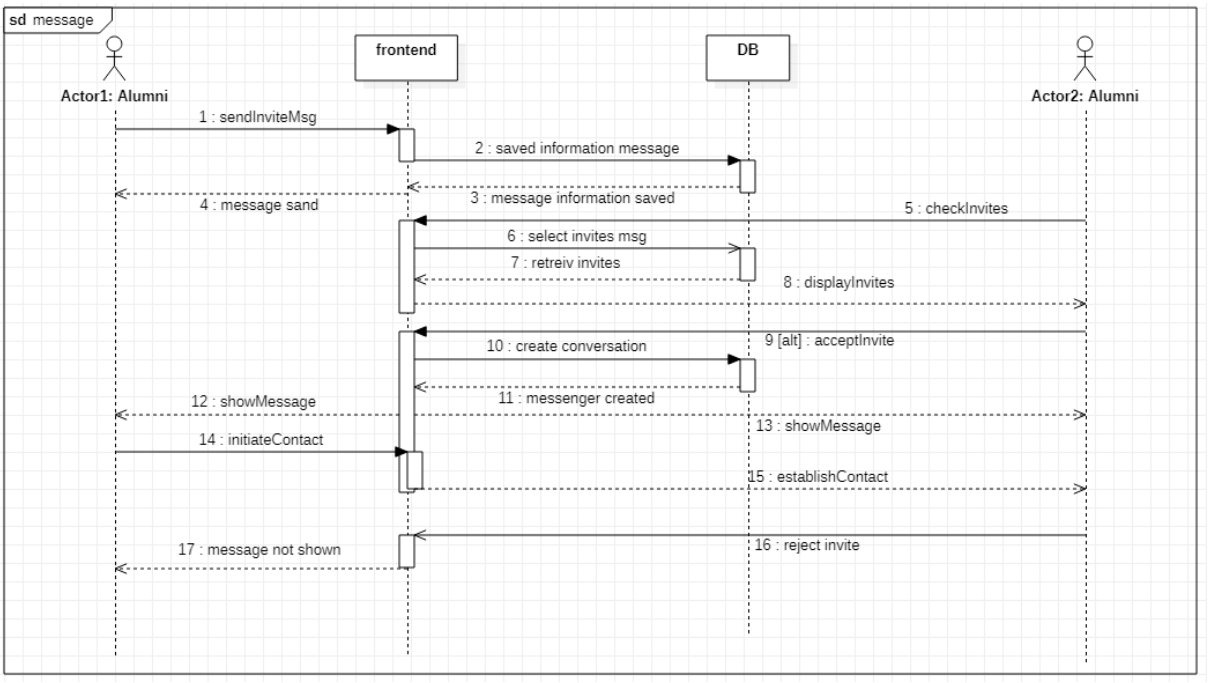

Figure 3.11: Messaging Sequence Diagram

3.4 Conclusion

In this chapter we first dissected the needs in terms of requirements of the application which are functional and non-functional. Then we modeled the application using UML diagrams. In the next chapter, we will move to the development of the application and delve in the detailed description of the architectural aspect of the application.

CHAPTER 4

DEVELOPMENT OF THE ALUMNI APPLICATION

4.1 Introduction

The purpose of this chapter is to display the complete work of Alumni application. First, we present the environment and the different development tools used. Secondly, we will see the security measures for Alumni application.Lastly, we will illustrate the realization of our work through screenshots of the most important interfaces of our application.

4.2 Environment and Development Tools

MySQL

MySQL, the most popular Open Source SQL database management system, is a software or service used to create and manage databases based on a relational model , is developed and distributed by Oracle Corporation[27].

PhpMyAdmin

PhpMyAdmin is a free software tool written in PHP, The main purpose of phpMyAdmin is to handle the administration of MySQL over the Web. It also provides a wide range of operations on MySQL and MariaDB[28].

Postman

Postman is an API platform thet offers tools for building, testing and using APIs. It provides features to create requests, define parameters, and specify headers and authentication methods, that allows developers to design and document APIs[29].

Angular

Angular is an open-source javaScript framework written in TypeScript, supported and developed by Google, it is for developing Single-page Web Applications (SPAs) using HTML and TypeScript[30].

HTML

HTML stands for hypertext markup language. Html made of keywords and commands that web designers use for creating websites, it's provides the structure or the way text, pictures, and so on will appear on the website [33].

Css

CSS, It is a style sheet language which is used to describe the look and formatting of a document written in markup language to change the style of web pages and user interfaces[31].

Typescript

TypeScript is a object-oriented, compiled programming language that builds on JavaScript, that allows to implement complex features on web pages[32].

Bootstrap 5

Bootstrap is a open source and free CSS framework that was designed for creating responsive websites. This framework was developed using HTML, CSS, and JavaScript, and consists of templates for creating various elements[33].

Angular Material

Angular Material is a UI library component, it is specially designed and developed for AngularJS developers in 2014. It helps to design the application in a structured manner. Its components help to construct attractive, consistent, and functional web pages and web applications [34].

VS code

Visual Studio Code is a lightweight but powerful source code editor which runs on desktop. It comes with built in support for JavaScript, TypeScript and Node.js. It's has a rich ecosystem of extensions for other languages and runtimes such as C++, C, Java, Python and PHP[35].

4.2.1 Security Measures for Alumni application

The alumni application has security measures in place to improve data safety, with a focus on authentication with JSON Web Tokens (JWT), password hashing and preventing SQL injection attacks.

1. **Authentication with JSON Web Tokens (JWT)** JSON Web Token (JWT) is an open standard (RFC 7519) that determine a compact and self-contained method for securely transmitting information between parties as a JSON object[46]. **Implementation** In successful login of the user, the server creates a JWT containing encrypted user information and a signature to ensure its integrity.Then the token sent to the client and stored and included in following requests. With each demand, the server verifies the authenticity of the JWT to grant access to protected resources and ensure the authenticated.

2. **Hashing Passwords**
   Password hashing is an essential security feature used to safeguard user credentials saved in the database of the alumni site. In order to make it harder for bad actors to recover the original passwords even if the database is stolen, it entails turning passwords into irreversible cryptographic representations[37].

   Implementation
   To store user passwords hashing. The alumni application use an industry standard and strong hashing algorithm. The original password is never preserved, only the hashed password is kept in the database.

3. **Guarding Against SQL Injection**
   A common security flaw known as SQL injection happens when an attacker modifies input data to run malicious SQL queries, potentially obtaining access to the database or impairing its integrity. Protecting the data on the alumni site requires preventing SQL injection[38].

   Implementation
   Alumni application has included strong input validation methods To strengthen security safeguards against SQL injection threats by carefully review user inputs and look for the presence of potentially dangerous characters, particularly single quotes(').

4.3 Presentation of Alumni application

Sign in

This page allows users of Alumni laghouat to log in by entering their credentials (username and email).

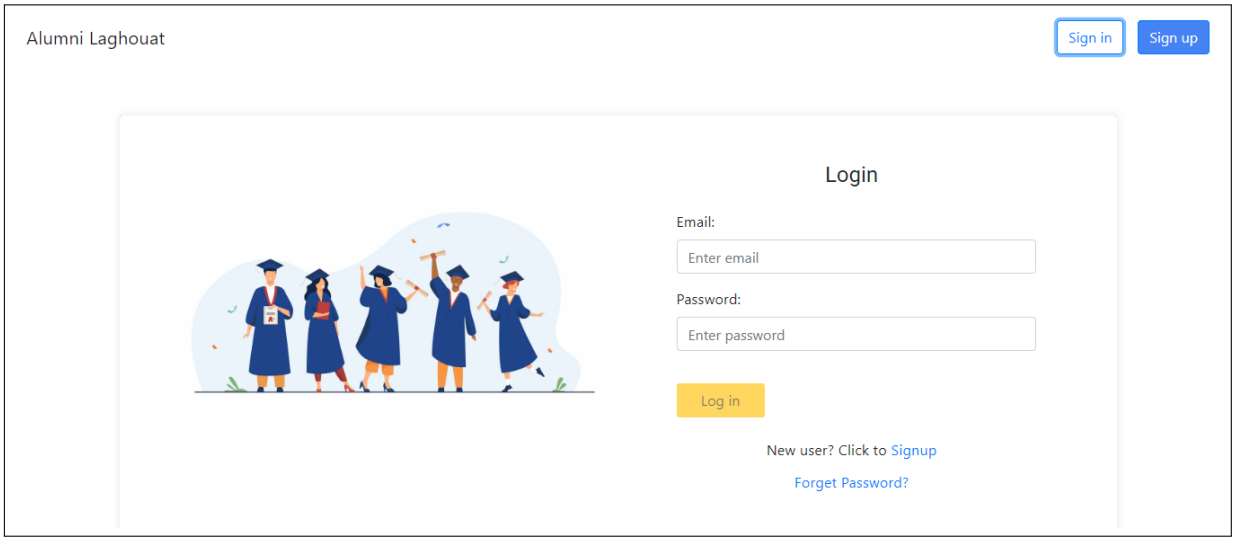

Figure 4.1: Sign In Page

Sing Up

This page enables a new users of alumni laghouat to create an account by providing the email of the university and user name.

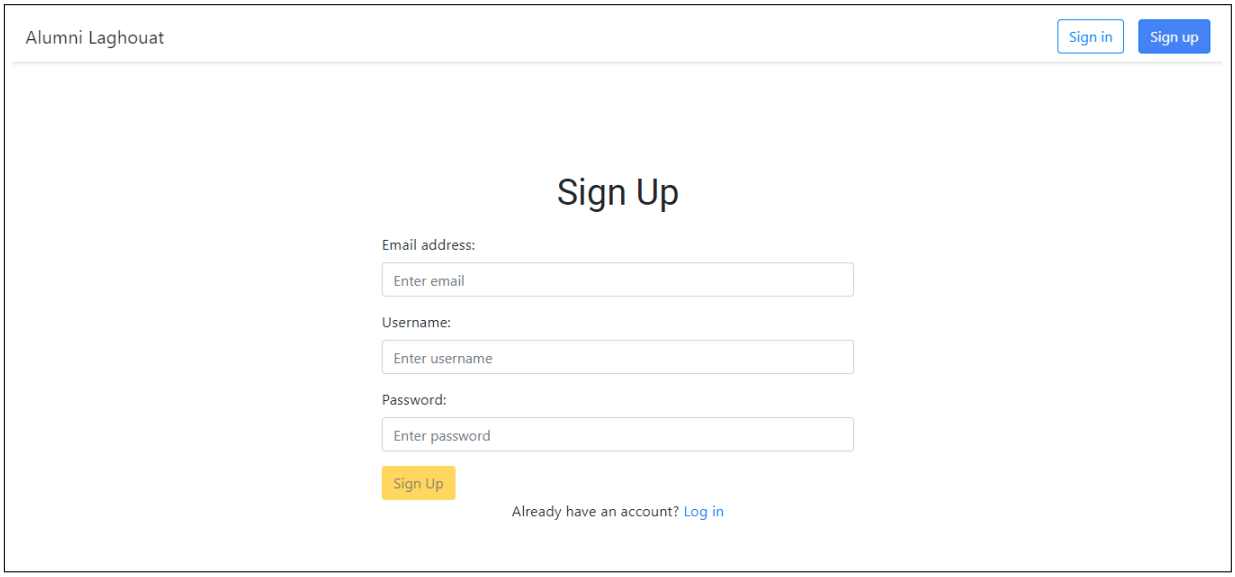

Figure 4.2: Sign Up Page

Home Page

This home page is the central hub that provides an overview of all the pages available in the alumni project, including event information, job listings, alumni success stories, and more.

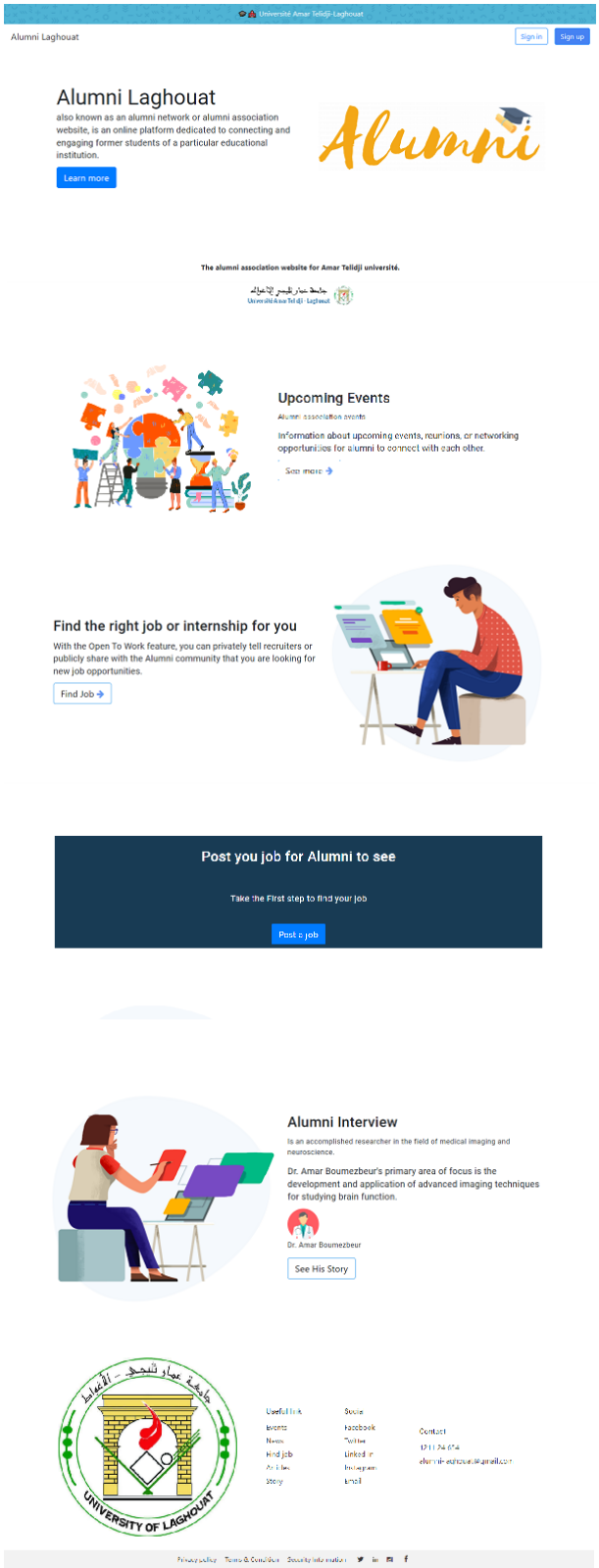

Figure 4.3: Home Page

Edit Profile page

This page enables alumni to personalize and update their information, including profile picture and university details.

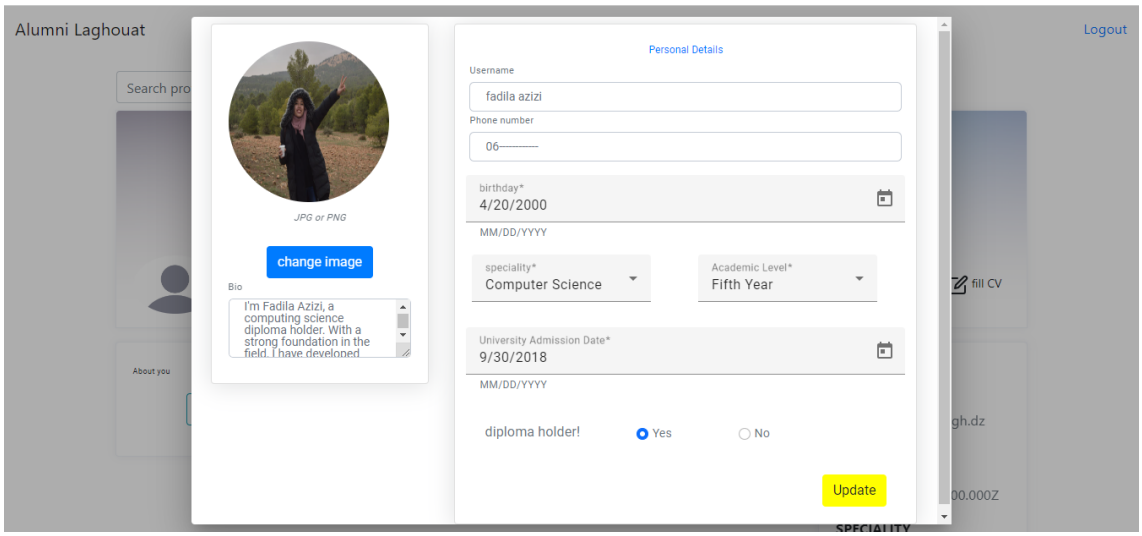

Figure 4.4: Edit Profile page

Alumni Profile

This page show individual profiles of alumni with relevant information of university.

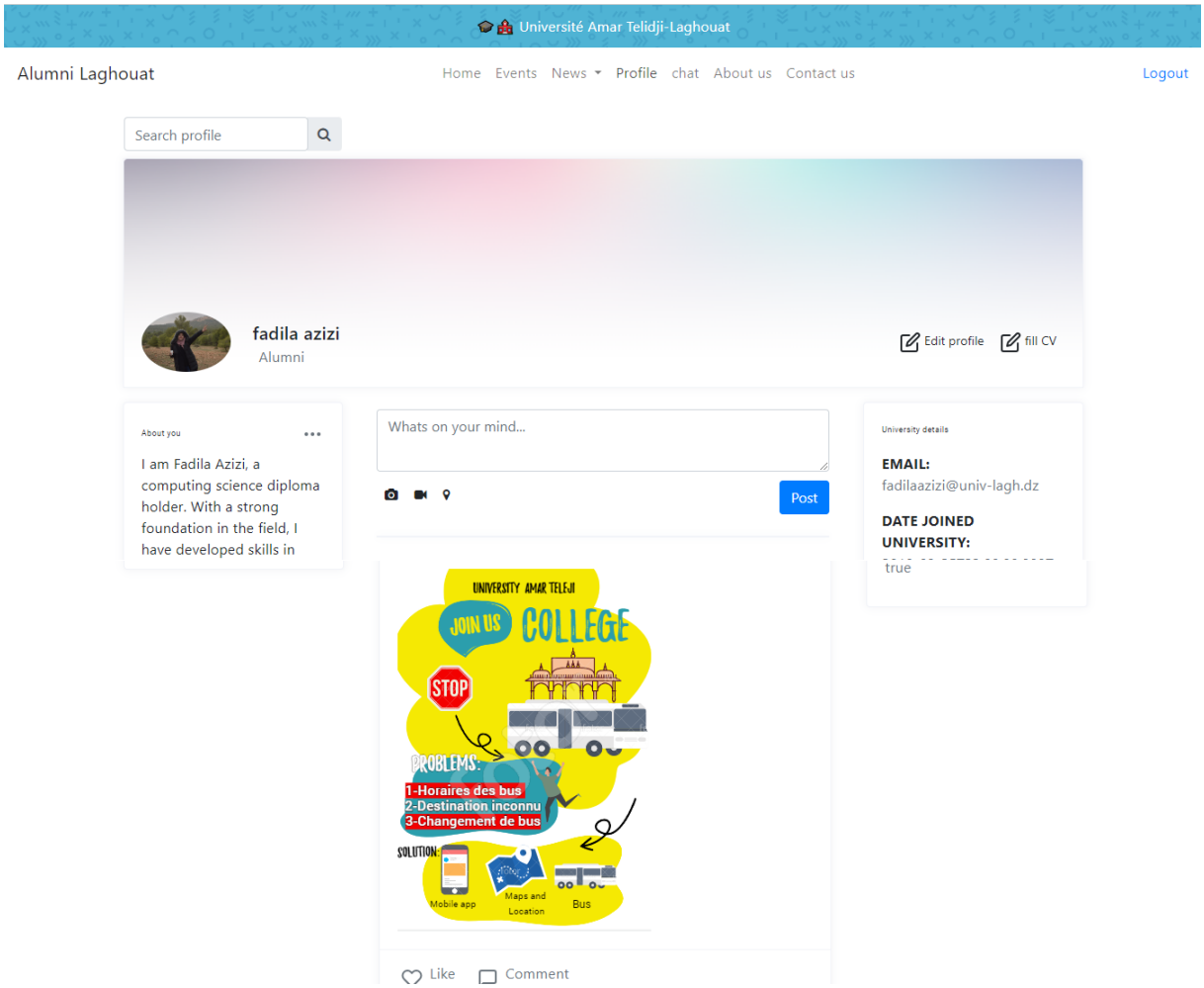

Figure 4.5: Alumni Profile Page

Events interface

This page allows the adminstrators to publish events and displays information about past or upcoming events about the university of laghouat.

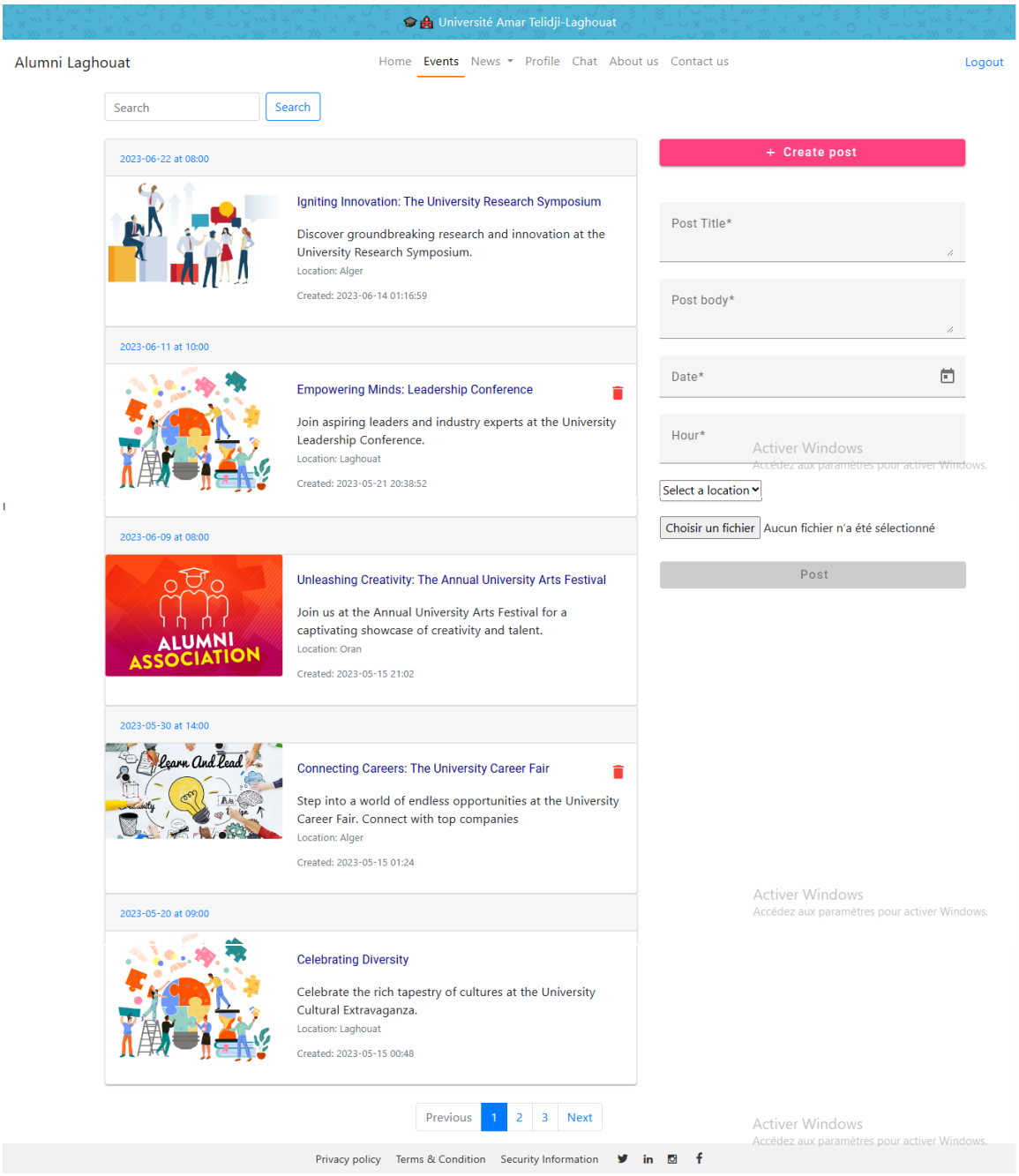

Figure 4.6: Event Page

Jobs interface

This page provides a platform for alumni to publish, search for job opportunities and appky for job.

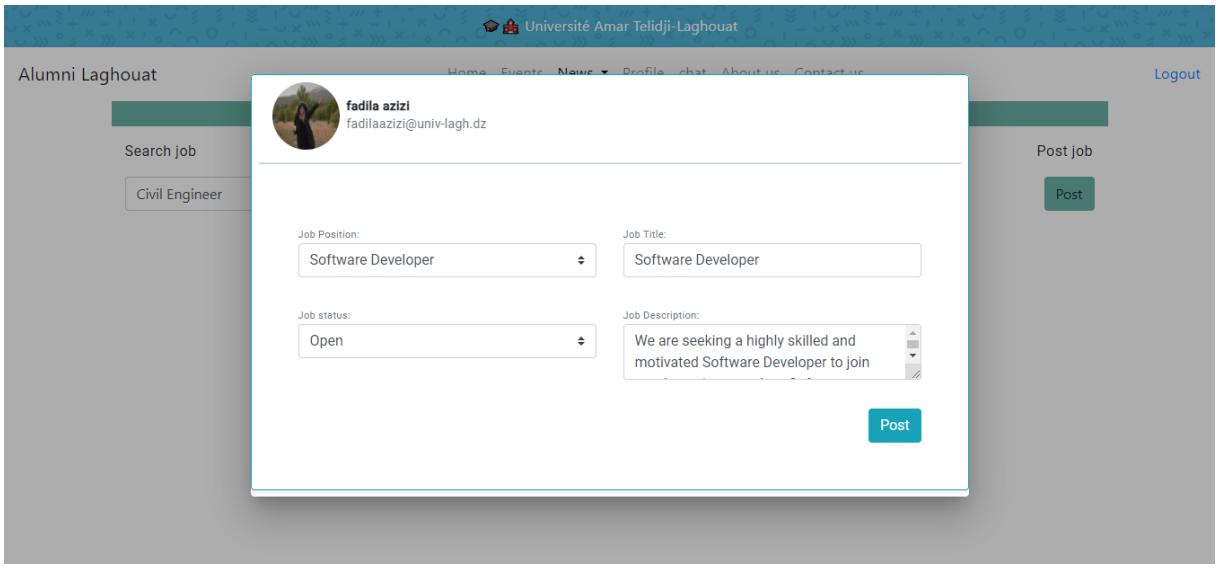

Figure 4.7: Publish Jobs Page

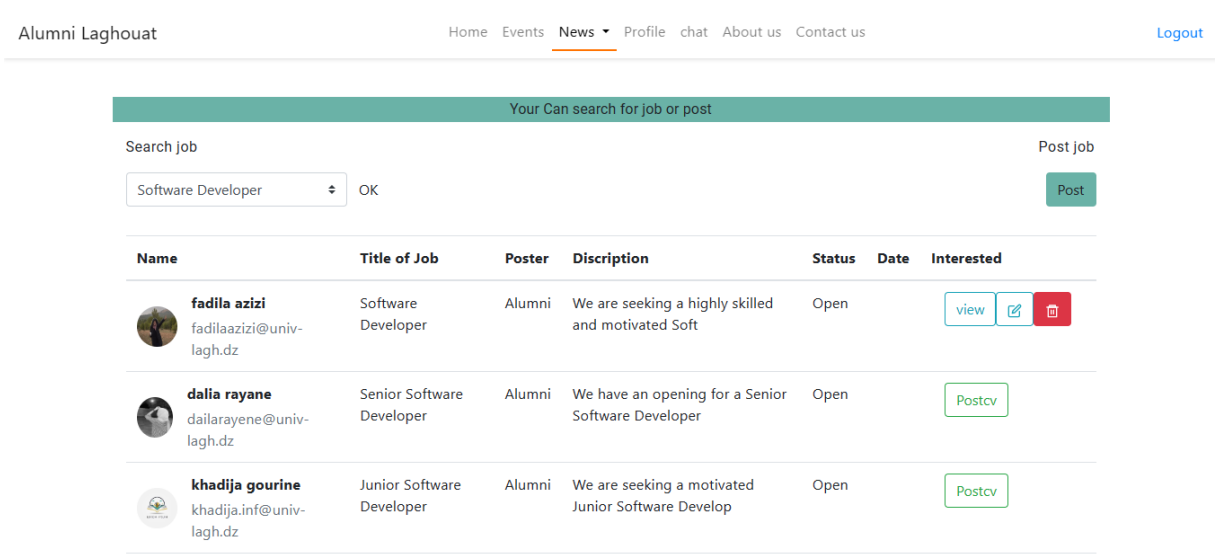

Figure 4.8: List Job Page

Application jobs

This page provides alumni who post a job with the ability to see users who have applied for their jobs.

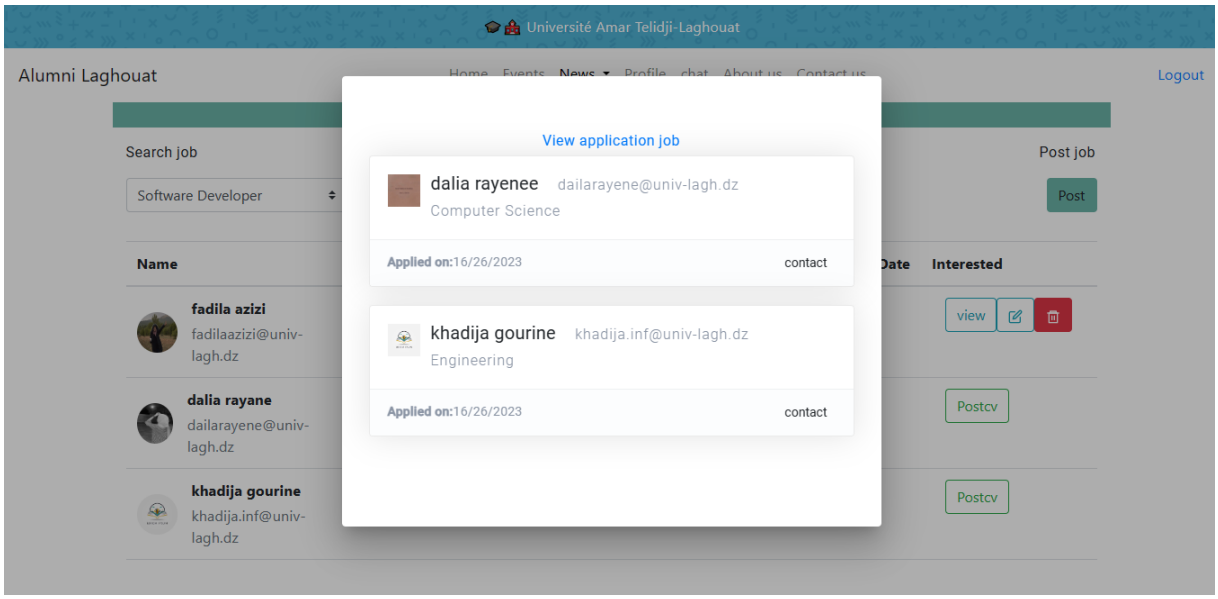

Figure 4.9: Application Jobs Page

Page Careers

This page allows alumnis to publish the thier success stories and display them, this page inspiring current and future alumni.

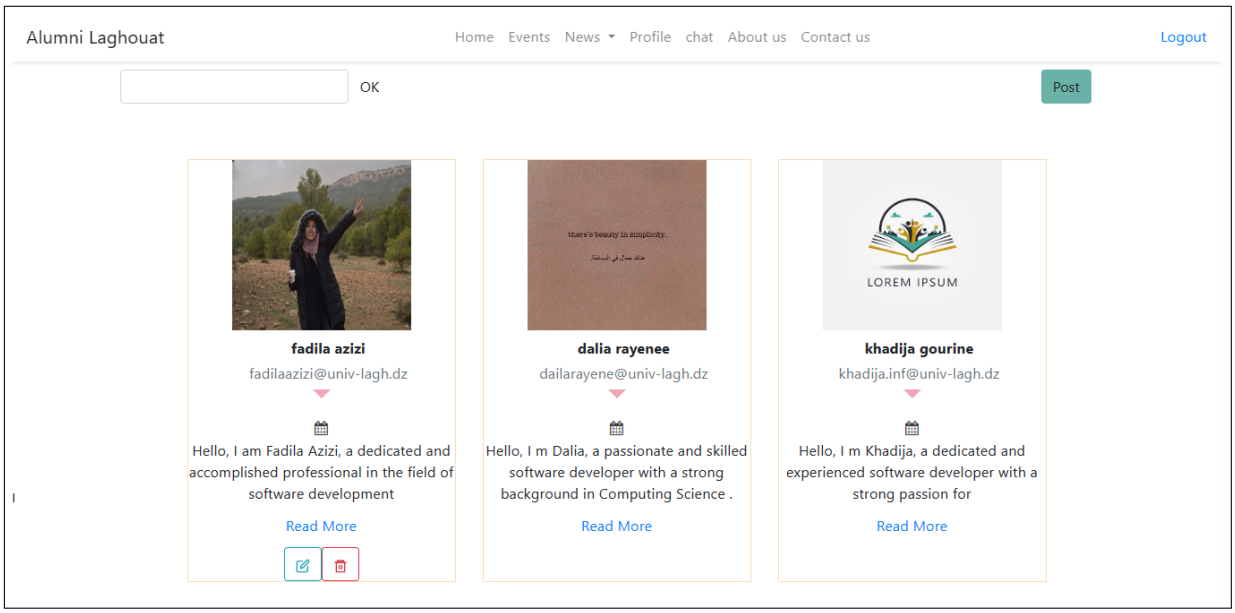

Figure 4.10: page Careers

create CV

This page allows users of alumni laghouat to create their curriculum vitae (CV).

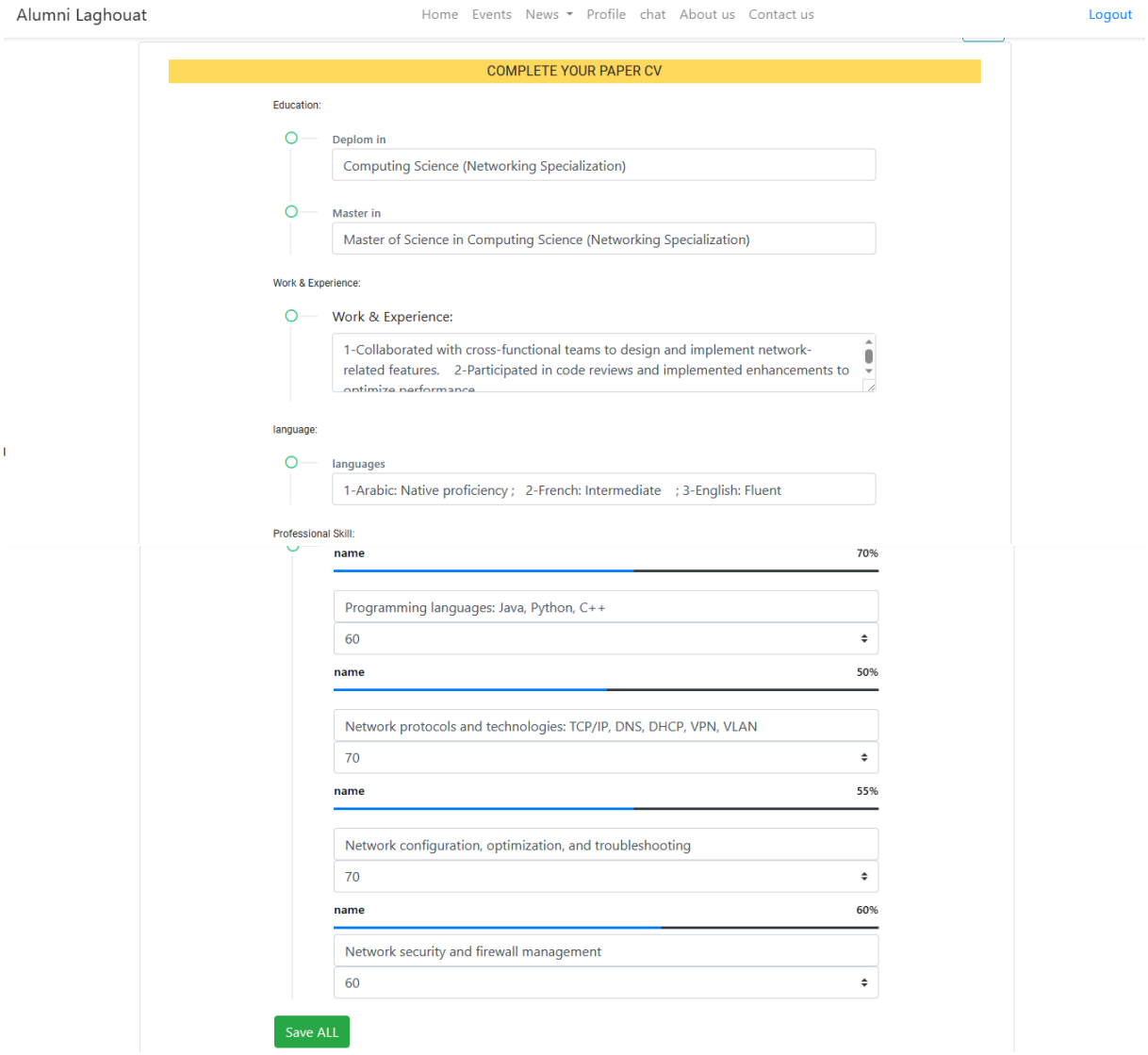

Figure 4.11: Create CV Page

Message page

This page provides a communication platform for alumni to connect and send messages to each other.

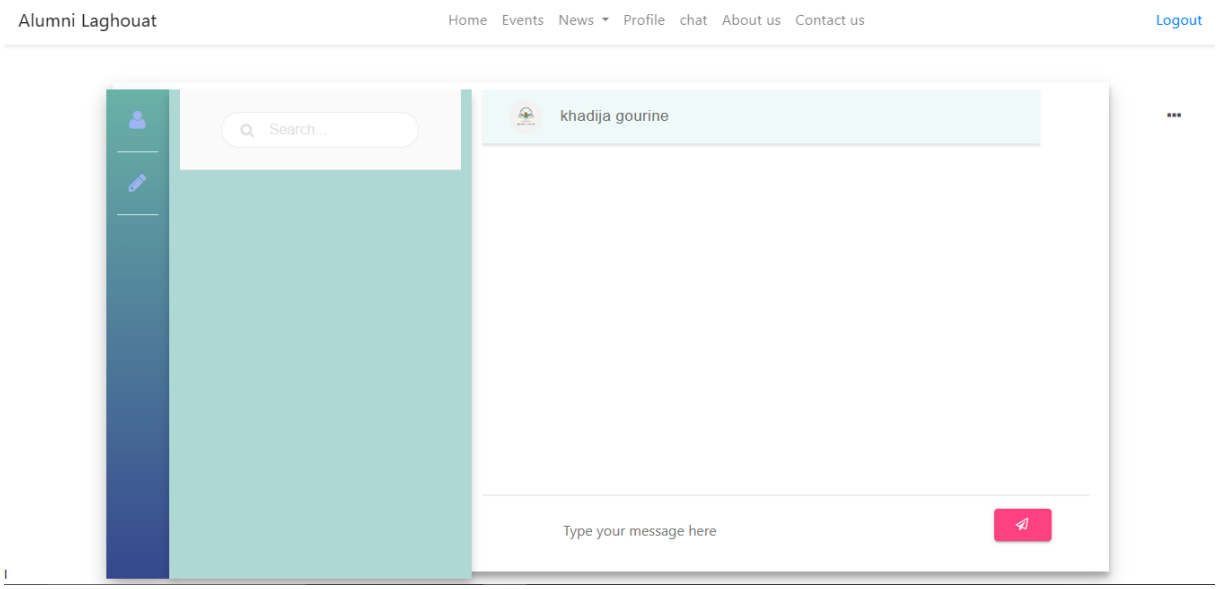

Figure 4.12: Message Page

4.3.1 Conclusion

In this chapter, we presented the environment and the different development tools. Secondly, we took look at the security measures for Alumni application. Lastly, we exposed the final product of the Alumni application by describing the most important pages of our application. In the next chapter we will test the workload of Alumni application and dissscus the results using Jmeter as test loading tool, then we compare between the results of load-balanced and a non-load-balanced application.

CHAPTER 5

LOAD TESTING WITH JMETER

5.1 Introduction

Performance testing is an extremely important aspect of the software and application development life cycle. If your web applications, sites, or APIs are going to be used by large numbers of users, you need to know how they are going to stand up to peak traffic or extended periods of sustained high traffic[39]. Early identification of software load limitations helps to configure the system appropriately to avoid unexpected crashes[40]. The objective of performance testing is not to discover bugs but to eliminate performance bottlenecks. The Software testing is done to ensure the quality characteristics of an application like functionality, performance, usability, scalability. Performance testing is mainly categorised as Stress and Load testing[41]. However, Load testing is just subset, or type, of the performance tests within the performance testing family. Other types of performance tests include Stress testing, spike testing, endurance testing, scalability testing, and volume testing.

The Apache JMeter application an open source software, a pure Java application designed to load test functional behavior and measure performance. It was originally designed for testing Web Applications but has since expanded to other test functions[42]. The software features includes FTP and HTTP requests and extensible custom scripting features[41]. This chapter of this thesis focus on:

Seeing all the steps taken to execute and configure the load testing scenarios using the Apache JMeter. Also, presents the results and analyzes obtained from the load testing experiments, under different workload distributions and scenarios.

5.2 Load Testing with JMeter

5.2.1 Configuring JMeter for Load testing Simulation

To simulate and evaluate the load balancing mechanism in the alumni website, Apache JMeter was utilized as the load testing tool. The following steps provide an overview of the configuration process for JMeter in order to achieve optimal load balancing simulation:

1. **Verify System Requirements:**
   Just like any other desktop application, you need to make sure your system meets the necessary basic requirements to be able to run JMeter. JMeter is a Java-based application, so you need to ensure you have Java installed, as well as the correct version. JMeter is compatible with versions of Java 8 or higher. JMeter can run on different OS[39].

2. **Installation and Setup:**
   You can download the latest version of Apache JMeter, depending on your specific environment or requirements, to serve as the load testing environment.

3. **Create a Load Test Plan:**

   - Create a new JMeter test plan to define the load testing scenario for load balancing evaluation.
   - The test plan included thread groups to simulate concurrent users, each with their own set of actions and requests.
   - Create a realistic workload distribution by defining particularly properties that influence the load test such as number of threads (Users), Ramp-up period, and Loop Count where:
     (a) **Threads (Users):** Threads represent the virtual users or concurrent users that JMeter will attempt to simulate.
     (b) **Ramp-up Period:** The ramp-up period specifies the time it takes to reach the total number of threads. It is measured in seconds.
     (c) **Loop Count:** The number of times to execute the test.

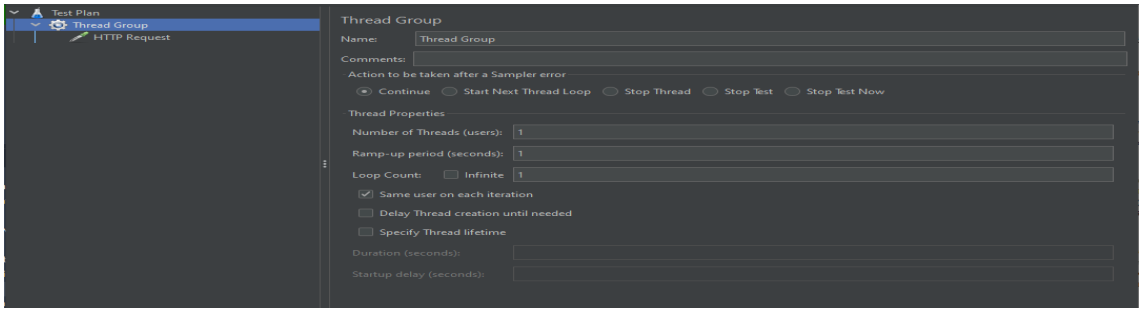

Figure 5.1: Apache Jmeter test plan

4. **Add and Configure Sampler:**

   In JMeter, Samplers are what allow JMeter to send the different types of requests. For example, these can be an HTTP, FTP, SMTP and TCP request, as well as many others. From here you are displayed with entering additional details such as the Protocol (HTTP/S), Server Name or IP, URL Path, and what type of request, such as GET, POST, HEAD, PUT, etc[39].

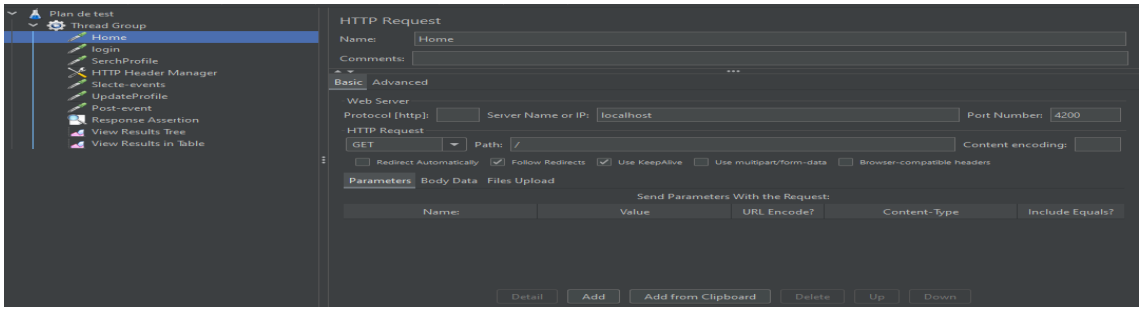

Figure 5.2: Samples

5. **JMeter HeadLess Tests:**

   To run JMeter in headless (non-GUI) mode, which means without any UI, to run load tests use the following command:

   jmeter -n -t [Location for jmeter test script] -l [Location for the result file]

   jmeter -n -t Alumnisite.jmx -l Loadtest.csv

   The command line has the following parameters:

   - -n: run in non-GUI mode.
   - -t: specifies the path to source .jmx script to run.
   - -l: specifies the path to the CSV file which will contain the raw results.

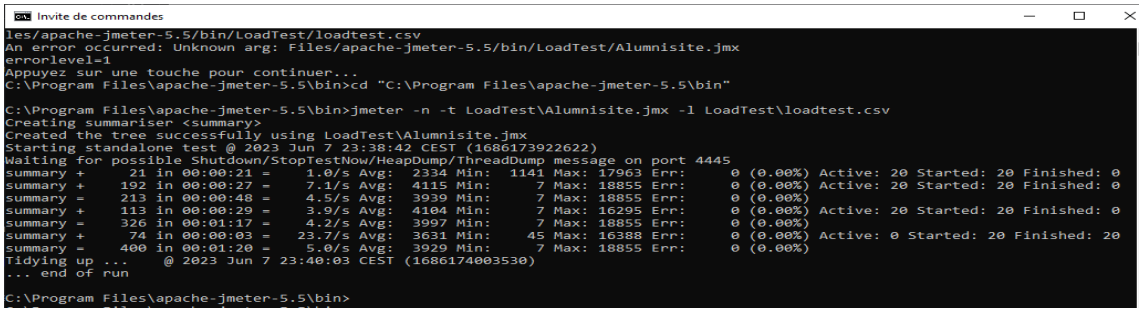

Figure 5.3: JMeter run in non-GUI mode

These options specify that JMeter should generate an HTML report of the test results: **jmeter -n -t [Location for jmeter test script] -l [Location for the result file] -e -o [Location for the HTMLreport folder]**

- -e: flag enables the generation of the report.
- -o: flag specifies the output directory for the HTML report.

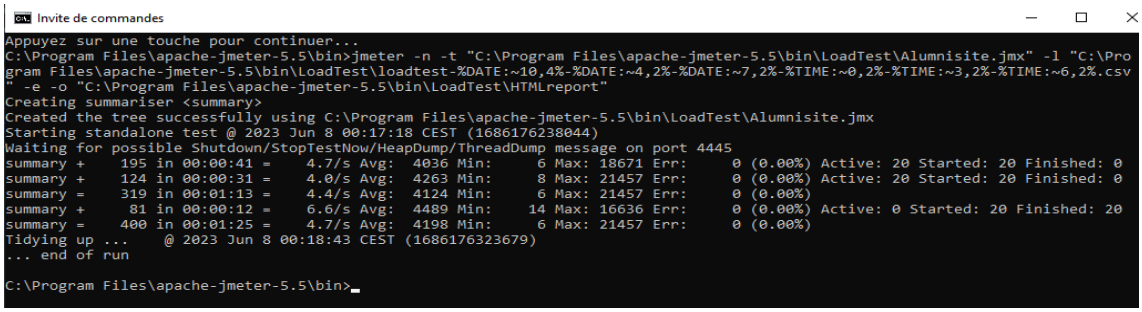

Figure 5.4: Generate an HTML report

6. **UI Listeners:**

   JMeter has a number of UI Listeners which can be used to view results directly in JMeter UI such as Summary Report, Aggregate Graph, View Results Tree, View Results in Table, plus many other options, to view and analyze your test results.

   As a general rule of thumb, avoid using UI Listeners. They consume a lot of memory. They aren't suitable for real load tests. Some may even trigger and Out Of Memory error with just a few concurrent threads groups running.[43].

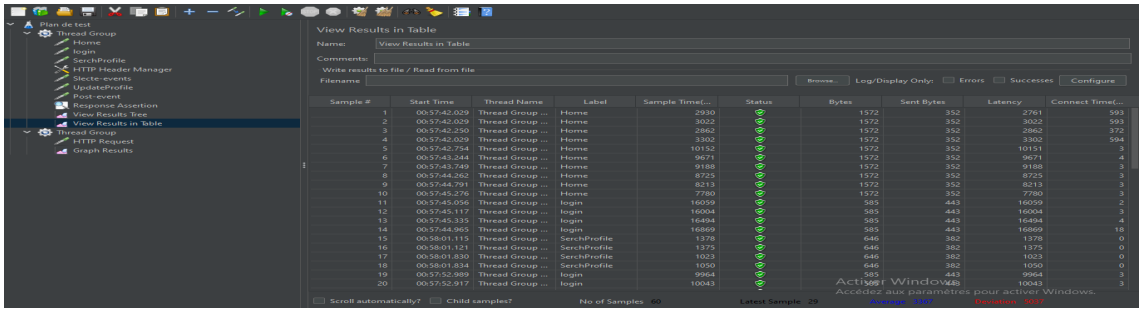

Figure 5.5: Apache Jmeter Listeners

7. **Execution and Load Testing:**

   You can view the results as the test runs. From these results you can see where any errors occurred or where there may be slow load times.

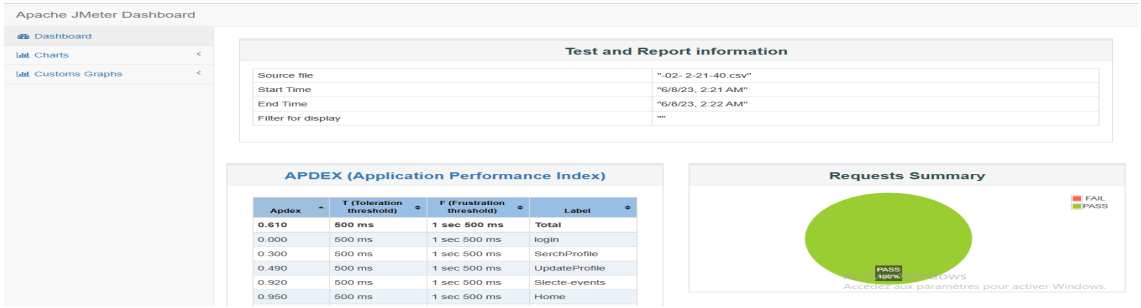

Figure 5.6: Apache JMeter Dashboard

5.2.2 Test Scenarios and Workload Distribution

Multiple test scenarios and workload distributions were designed and executed using Apache JMeter. for the purpose of evaluating the performance and effectiveness of the load balancing mechanism in the alumni website. The following test scenarios were defined to simulate different user loads and usage patterns:

1. **Scenario 1: Normal Load**

   - **Description:** This scenario simulates normal usage of the alumni website with an average number of concurrent users.
   - **Workload Distribution:** The workload is evenly distributed across multiple virtual users, each simulating typical user actions such as browsing profiles, searching for alumni, and accessing events.

     Number of Threads (Users): 50

     Ramp-up Period (Secondes): 5

     Loop Count: 1

2. **Scenario 2: High Traffic Load**

   - **Description:** This scenario simulates a high traffic situation where a significant number of users access the website simultaneously.
   - **Workload Distribution:** The workload is distributed among a large number of virtual users, with a concurrent logins, profile updates, and event registrations.
     Number of Threads: 500
     Ramp-up Period: 10
     Loop Count: 3

3. **Scenario 3: Spike Load**

   - **Description:** This scenario tests the website's response to sudden spikes in user traffic, simulating different situations.
   - **Workload Distribution:** The workload is concentrated on a specific timeframe, where a burst of virtual users performs actions like event registrations.
     Number of Threads: 500
     Ramp-up Period: 1
     Loop Count: 3

4. **Scenario 4: Gradual Load Increase**

   - **Description:** This scenario evaluates the website's scalability by gradually increasing the number of concurrent users over a specified time period.
   - **Workload Distribution:** The workload starts with a low number of virtual users and gradually increases at predetermined intervals, reflecting the growth in user activity and load on the website. Number of Threads: 50
     Ramp-up Period: such as 30 second
     Loop Count: 5 or use a Forever option to continue the load indefinitely.

5.3 Results and Analysis

5.3.1 Understanding JMeter Metrics:

The load testing experiments conducted using JMeter provided valuable results in the alumni website. Some of the commonly used Performance Metrics are described below:

- **Response time:** Response Time is the time elapsed between a request and the completion of request by the system. Response time is critical in Real Time Applications[44]. It indicates the overall performance of your application from the user's perspective.
- **Throughput:** Throughput is the number of requests processed by the server it is calculated as requests/unit of time. The time is calculated from the start of the first sample to the end of the last sample. This includes any intervals between samples, as it is supposed to represent the load on the server. The formula is:
  Throughput = (number of requests) / (total time)[45]. It represents the system's capacity to handle concurrent requests and is measured in requests per second (RPS) or transactions per second (TPS).
- **Error Rate:** Error rate is percentage of requests resulting in errors as compared to total number of requests. Rate of errors may increase when application starts reaching its threshold limit or goes beyond[44].
- **Scalability:** Scalability refers to the system's ability to handle increasing user loads without a significant degradation in performance.
- **Elapsed time:** Measures the elapsed time from just before sending the request to just after the last chunk of the response has been received[44].
- **Hits Per Second:** Measures the number of requests processed by the server per second, to see the capability of the server in handling the load.
- **Connect Time:** JMeter measures the time it took to establish the connection, including SSL handshake. Note that connect time is not automatically subtracted from latency. In case of connection error, the metric will be equal to the time it took to face the error, for example in case of Timeout, it should be equal to connection timeout[45].

5.4 Load Testing Results and Metrics

The load testing results presented for each scenario , including the normal load, high traffic load, spike load, and gradual load increase, provides generale understanding of the system's performance under different load conditions. Including relevant tables and graphs for better understanding and interpreting the results, where:

- Label: is the name of the request.
- #Samples: is the total number of executions.
- Average: Average Response Time in milliseconds.
- Min: Minimum Response Time.
- Max: Maximum Response Time.
- Errors %: Percentage of errors (errors / (errors + samples) * 100).
- Throughput: Number of samples per second.

5.4.1 Load Testing Results: Normal Load Scenario

- Response Time: In a normal load scenario, The average response time was 8002 milliseconds, with a minimum response time of 6 milliseconds and a maximum response time of 46 second, indicating that the system responds to all user requests under normal load conditions.
- Throughput: typically an average throughput of 5 transactions per second was achieved, with peak values reaching 26 requests per second, indicating that the system efficiently processed the incoming requests.
- Error Rate: In normal load conditions, the error rate was 0.00%, indicating that the system performed reliably without errors or failures.
- Concurrent Users: The system successfully handled 50 concurrent users without performance degradation indicating stability and optimal performance.

Tables and graphs depicting the performance metrics are included below:

Table 5.1: Load Testing Results For Normal Load

Requests	**Executions**			**Response Time (ms)**			**Throughput**
Label	**Sample**	**Fail**	**Error %**	**Average**	**Min**	**Max**	**Transaction/s**
Total	**300**	**0**	**0.00%**	**8002.11**	**6**	**46556**	**5.94**
Home	50	0	0.00%	125.08	6	788	11.31
Login	50	0	0.00%	43782.84	29683	46556	1.04
Post event	50	0	0.00%	170.00	58	265	26.97
Search Profile	50	0	0.00%	2504.14	873	9603	2.56
Select Event	50	0	0.00%	605.92	84	7107	4.96
Update Profile	50	0	0.00%	824.68	618	2015	13.81

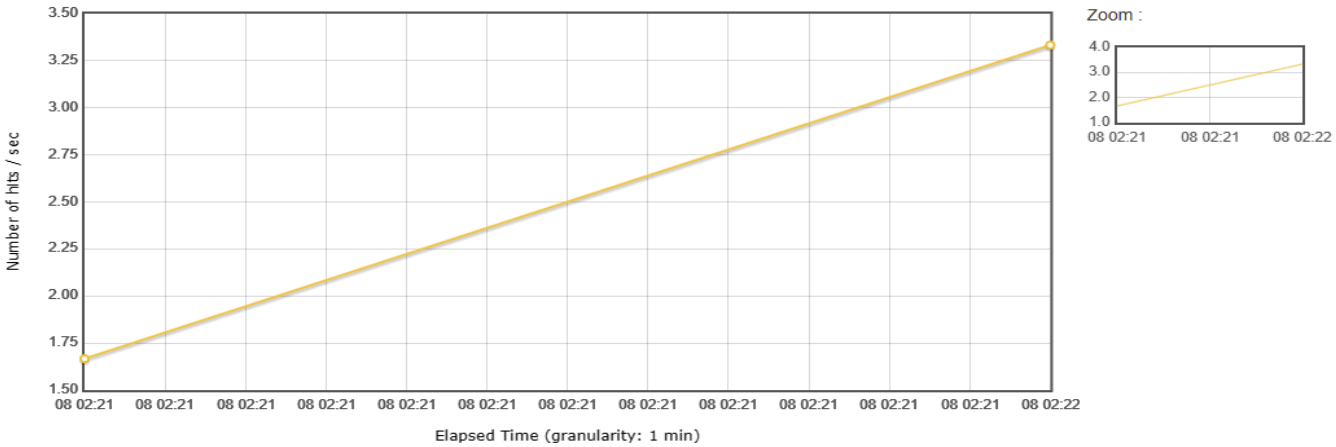

Figure 5.7: Hits Per Second in Normal Load scenario

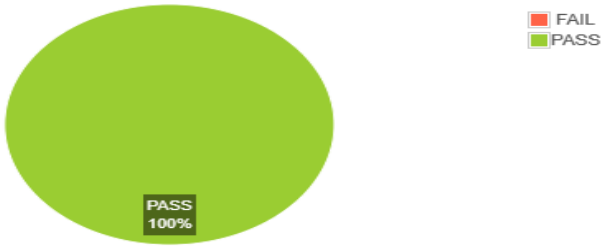

Figure 5.8: Requests Summary in Normal Load Scenario

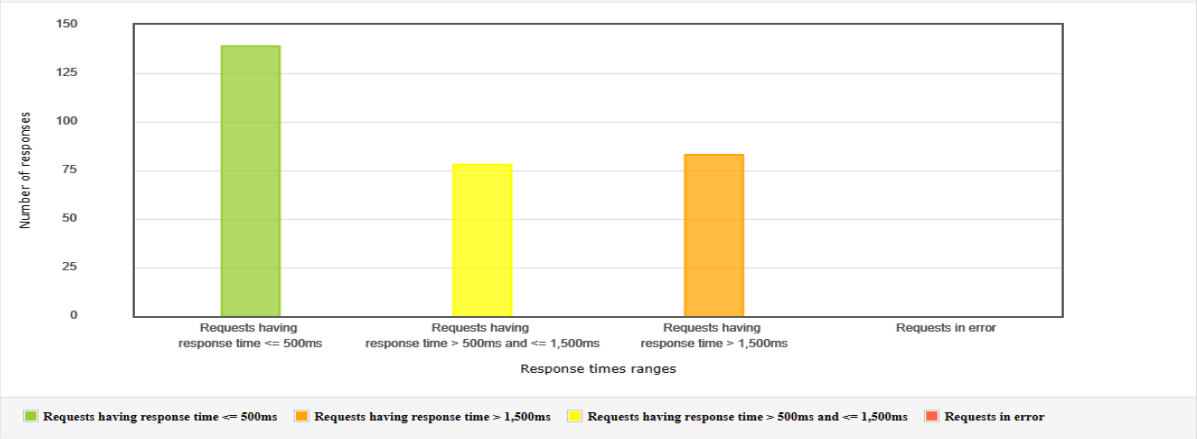

Figure 5.9: Response Time Overview in Normal Load scenario

5.4.2 Load Testing Results: High Load Scenario

- Response Time : In a high load scenario, The average response time was 28 second. Indicating that the system takes longer to process and respond to user requests.
- Throughput : Incrising in the average throughput reaching 10 transactions per second during the high load. This indicates that the system was unable to handle a high volume of requests and struggled to keep optimal response times.
- Error Rate : The error rate was 33.88%, this indicates that the system was unable to handle the increase in traffic effectively and reached its capacity limits.
- Concurrent Users : The system couldn't handle all the 500 concurrent users, leading to increase the response times, higher error rates, and potential performance degradation.

Table 5.2: Load Testing Results For High Load Scenario

Requests	**Executions**			**Response Time (ms)**			**Throughput**
Label	**Sample**	**Fail**	**Error %**	**Average**	**Min**	**Max**	**Transaction/s**
Total	**9000**	**3049**	**33.88%**	**28776.96**	**1**	**294573**	**10.37**
Home	1500	0	0.00%	1122.75	4	21644	1.78
Login	1500	633	42.20%	66696.42	1	294573	1.74
Post event	1500	600	40.00%	2608.582	1	14400	1.74
Search Profile	1500	6001	40.07%	53076.57	1	231107	1.75
Select Event	1500	607	40.47%	1451.53	1	31723	1.75
Update Profile	1500	608	40.53%	47705.92	1	233653	1.74

Table 5.3: Errors in the High Load Scenario

Types of error	Number of error	% in error	% in all samples
Non HTTP response code: java.net.SocketException/Non HTTP response message: Connection reset	49	1.61%	0.54%
Non HTTP response code: org.apache.http.conn.HttpHostConnectException/Non HTTP response message: Connect to localhost:3000 [localhost/127.0.0.1, localhost/0:0:0:0:0:0:0:1] failed: Connection refused: connect	3000	98.39%	33.33%

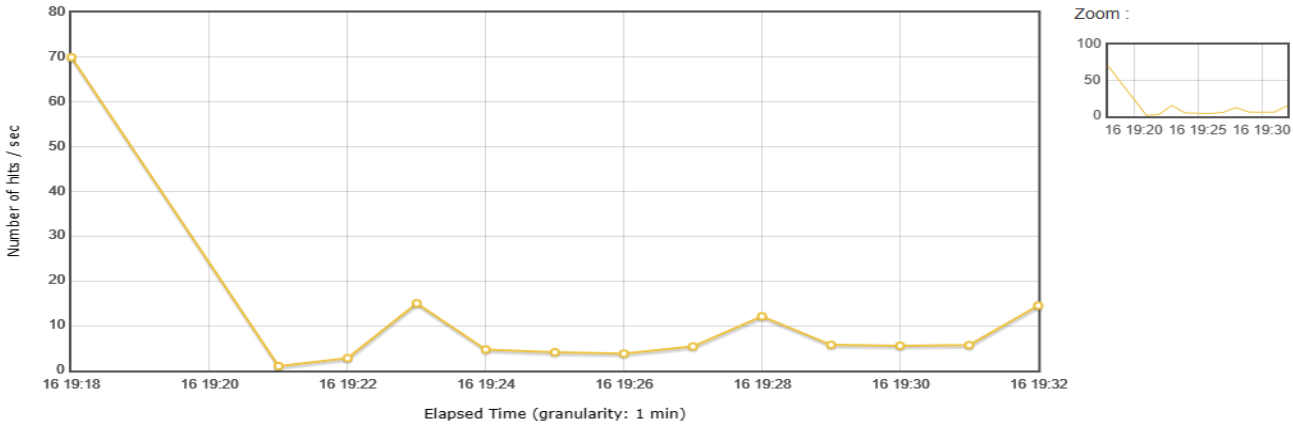

Figure 5.10: Hits Per Second in high Load scenario

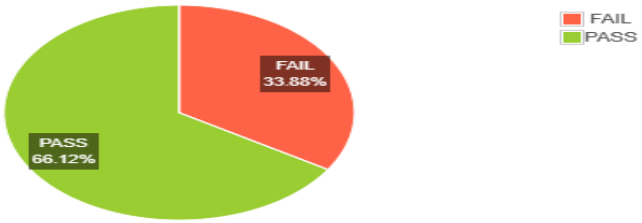

Figure 5.11: Requests Summary in high Load scenario

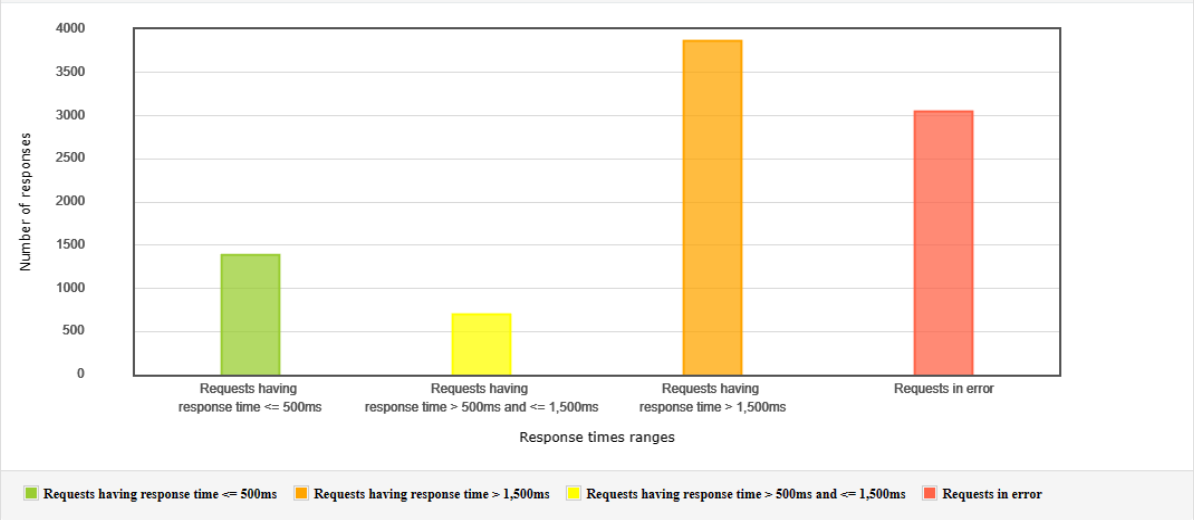

Figure 5.12: Response Time Overview in high Load scenario

5.4.3 Load Testing Results: Spike Load Scenario

- Response Time: The average response times was 29 seconds. Indicating that the system struggled to handle the sudden surge in traffic, leading to slower response times and potential delays for users.
- Throughput: Throughput reaches 10.45 request per second during the spicke load.
- Error Rate: The error rate increased to 35.64% during the spike load, indicating that some request failures or timeouts. Also indicates that the system reached its capacity limits.
- Concurrent Users: The system couldn't handle the sudden load spike, leading to increase the response times, lower performance and errors.

Table 5.4: Load Testing Results For Spike Load

Requests	**Executions**			**Response Time (ms)**			**Throughput**
Label	**Sample**	**Fail**	**Error %**	**Average**	**Min**	**Max**	**Transaction/s**
Total	**9000**	**3208**	**35.64%**	**29165.66**	**1**	**285974**	**10.45**
Home	1500	218	14.53%	1978.64	3	16947	1.80
Login	1500	606	40.40%	66898.71	1	285974	1.76
Post event	591	1000	39.40%	2811.44	1	10817	1.75
Search Profile	591	208	39.40%	54352.16	1	219859	1.76
Select Event	608	1000	40.53%	1832.15	1	20042	1.76
Update Profile	1500	594	39.60%	4712084.84	1	221576	1.75

Table 5.5: Errors in the Spike Load Scenario

Types of error	Number of error	% in error	% in all samples
Non HTTP response code: java.net.SocketException/Non HTTP response message: Connection reset	35	1.09%	0.39%
Non HTTP response code: org.apache.http.conn.HttpHostConnectException/Non HTTP response message: Connect to localhost:4200 [localhost/127.0.0.1, localhost/0:0:0:0:0:0:0:1] failed:Connection refused: connect	2955	92.11%	32.83%
Non HTTP response code: org.apache.http.conn.HttpHostConnectException/Non HTTP response message: Connect to localhost:3000 [localhost/127.0.0.1, localhost/0:0:0:0:0:0:0:1] failed: Connection refused: connect	218	6.80%	2.42%

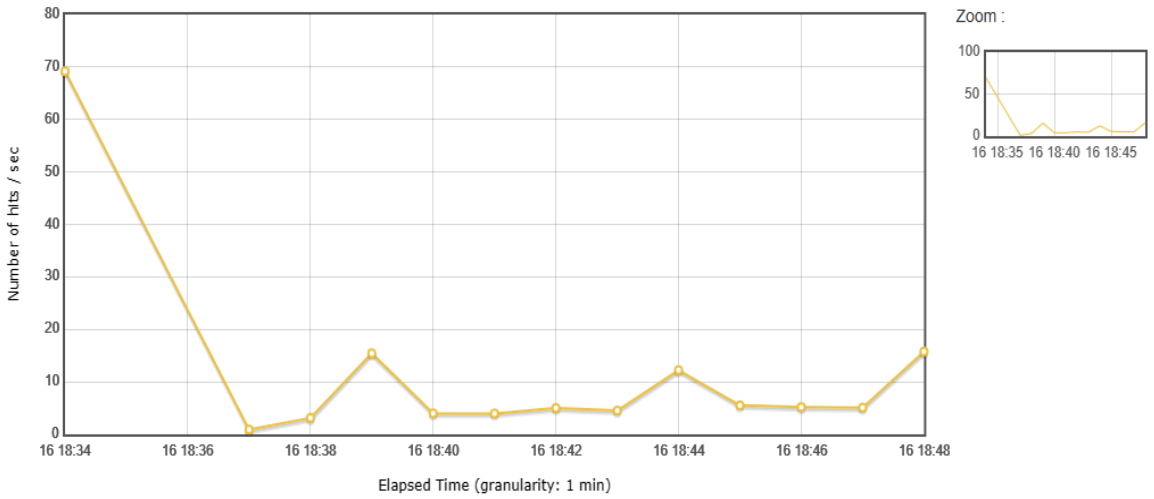

Figure 5.13: Hits Per Second in Spike Load Scenario

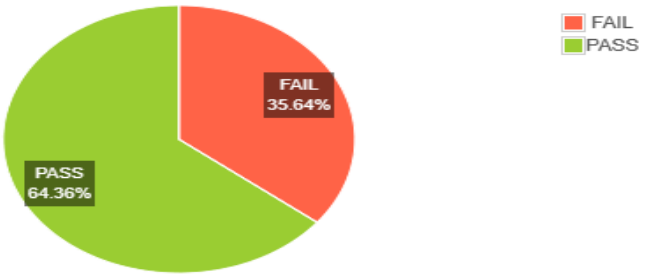

Figure 5.14: Requests Summary in Spike Load Scenario

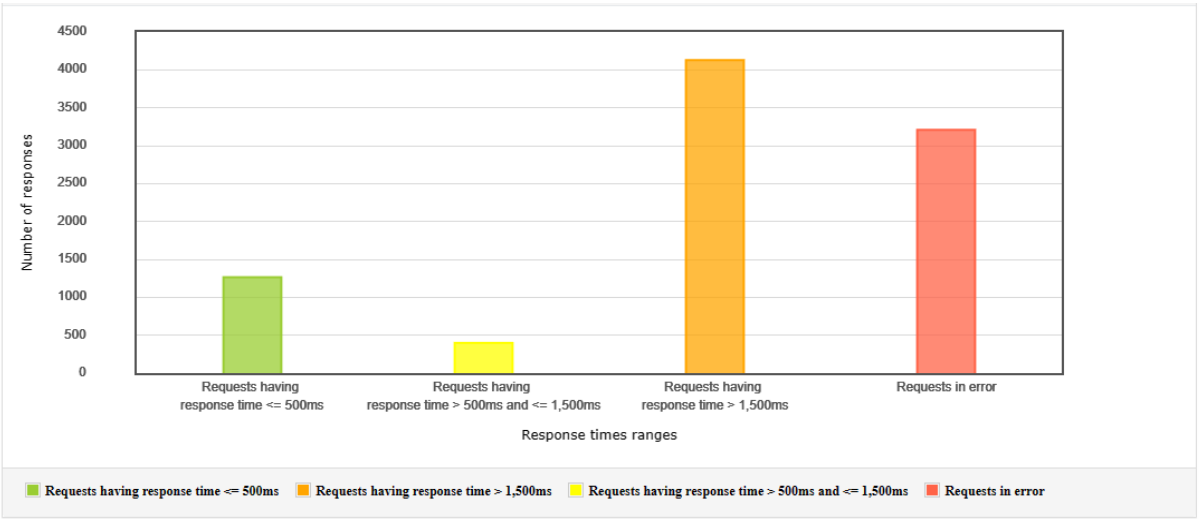

Figure 5.15: Response Time Overview in Spike Load Scenario

5.4.4 Load Testing Results: Gradual Load Scenario

- Response Time: The response times gradually increased, reaching an average of 5 seconds. Indicating that the system could handle the increased load without performance degradation with good scalability.
- Throughput: The throughput also increased, with an average of 6 requests per second. This indicates that the system efficiently processed the incoming requests.
- Error Rate: The error rate was 0.00%. This indicate that the system successfully handled the increased load without any failures or errors.
- Concurrent Users: The system handled the load with The gradual increase in concurrent users without any performance issues.

Table 5.6: Load Testing Results For Gradual Load

Requests	**Executions**			**Response Time (ms)**			**Throughput**
Label	**Sample**	**Fail**	**Error %**	**Average**	**Min**	**Max**	**Transaction/s**
Total	**1500**	**0**	**0.00%**	**7336.49**	**5**	**35486**	**6.22**
Home	250	0	0.00%	21.02	6	359	1.09
Login	250	0	0.00%	25373.54	3212	35486	1.04
Post event	250	0	0.00%	3802.70	11	8029	1.07
Search Profile	250	0	0.00%	5689.08	131	12680	1.05
Select Event	250	0	0.00%	3898.58	96	7532	1.06
Update Profile	250	0	0.00%	5234.01	5	12760	1.06

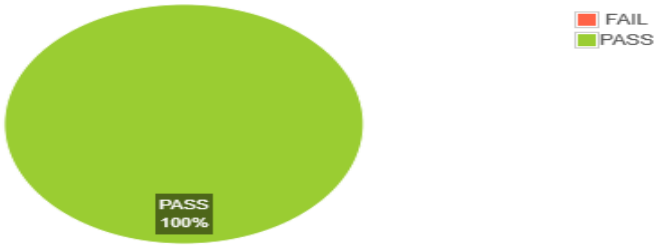

Figure 5.16: Requests Summary in Gradual Load Scenario

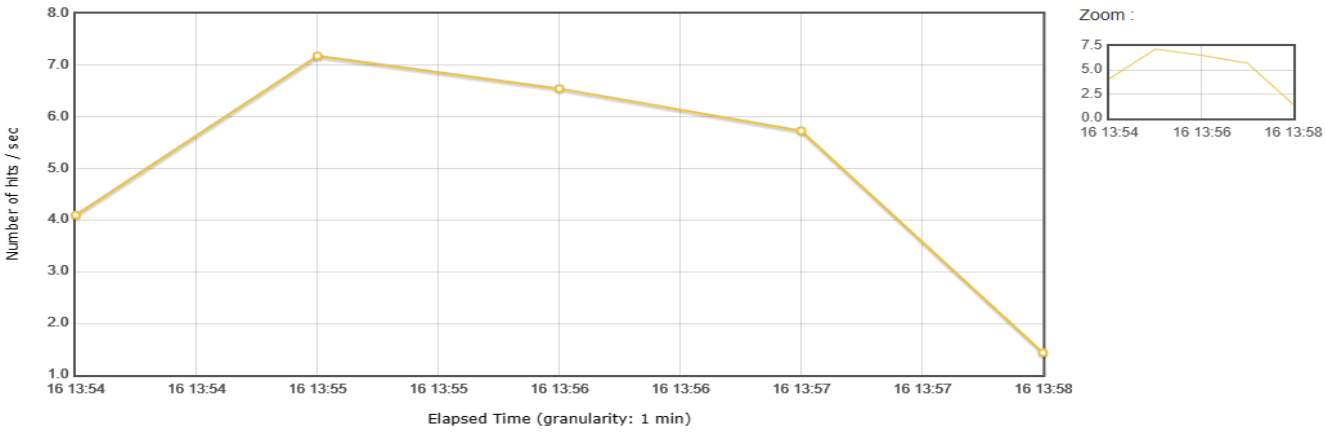

Figure 5.17: Hits Per Second in Gradual Load Scenario

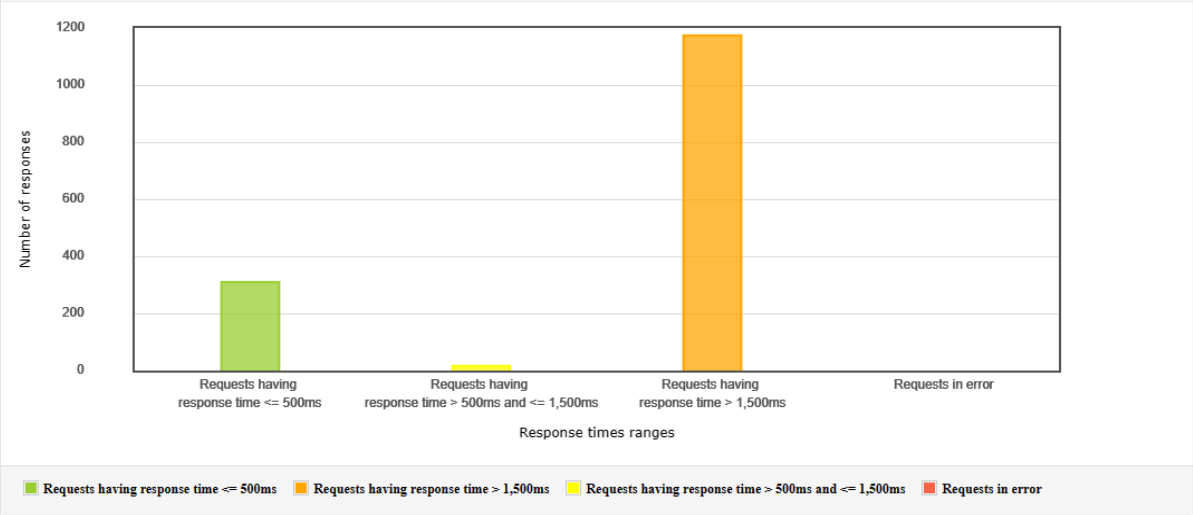

Figure 5.18: Response Time Overview in Gradual Load Scenario

5.5 Configuration of the Load-Balanced System

In this case, we are using two PCs as master and slave machines for simulating the load balancing mechanism using the Apache JMeter tool.

1. **Connecte the PCs:**
   - Connect both PCs to the same local network, such as a Wi-Fi network.
   - Ensure that the PCs are having IP addresses on the same subnet. For example, in our case we assign IP addresses like 192.168.1.45 and 192.168.1.35 to the PCs.
   - Ensure that the necessary ports used by JMeter for communication between the master and slave PCs are not blocked by a firewall.
2. **Configure Apache JMeter:**

   1. Install JMeter: Download JMeter both on the master PC and salve PC, both PCs should have the same version of JMeter.

   2. Create the Test Plan:

   • Create and configure the Test Plan on the master and slave PC in GUI mode. Save the Test Plan as a .jmx file.

   • Generate the rmi-keystore.jks file on the master PC, The rmi-keystore.jks file is used for RMI (Remote Method Invocation) communication in JMeter's distributed testing mode.

   • Execute the following command to generate the rmi-keystore.jks file:

```
C:\Users\HP>cd C:\Program Files\apache-jmeter-5.5\bin

C:\Program Files\apache-jmeter-5.5\bin>keytool -genkey -alias rmi_key -keyalg RSA -keystore rmi_keystore.jks -validity 7 -keysize 2048
Enter keystore password:
```

Figure 5.19: Generating the rmi-keystore file

• Transfer the generated rmi-keystore.jks file to the slave PC as well. This will ensure that both the master and slave PCs have the necessary rmi-keystore.jks file for RMI communication.

• Make the necessary changes within "server.properties" file on both PCs.

```
# Keystore file that contains private key
server.rmi.ssl.keystore.file=rmi_keystore.jks
#
# Password of keystore
server.rmi.ssl.keystore.password=password
#
# Key alias
server.rmi.ssl.keystore.alias=rmi
#
```

Figure 5.20: Jmeter-properties File

- **On the master PC:**
  - Make the necessary changes within "jmeter.properties" file by Specifying Remote Engine IP of the slave PC.
  - Add "Port" field as the default value (default is 1099).

```
# Remote Hosts - comma delimited
remote_hosts=0.0.0.0,192.168.1.35:1099
#remote_hosts=localhost:1099,localhost:2010

# RMI port to be used by the server (must start rmiregistry with same port)
server_port=1099

# To change the port to (say) 1234:
# On the server(s)
# - set server_port=1234
# - start rmiregistry with port 1234
# On Windows this can be done by:
# SET SERVER_PORT=1234
# JMETER-SERVER
#
```

Figure 5.21: Remote Engine IPs in Jmeter-properties File

• Start JMeter server on the master machine.

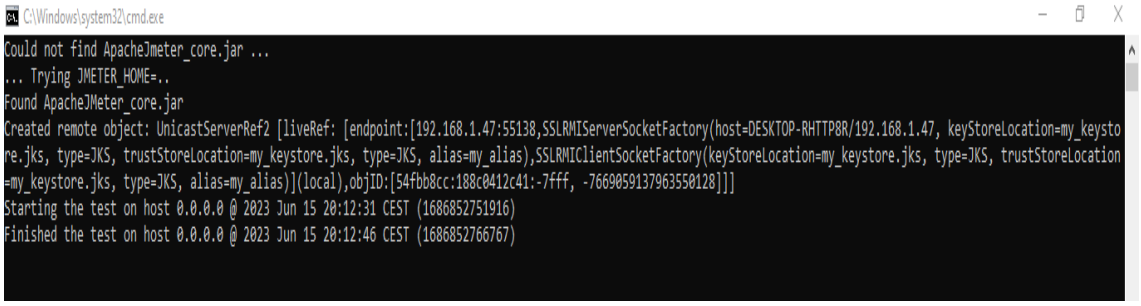

Figure 5.22: Jmeter Server On The Master PC

- **On the slave PC:**

  Start JMeter server on the slave machine.

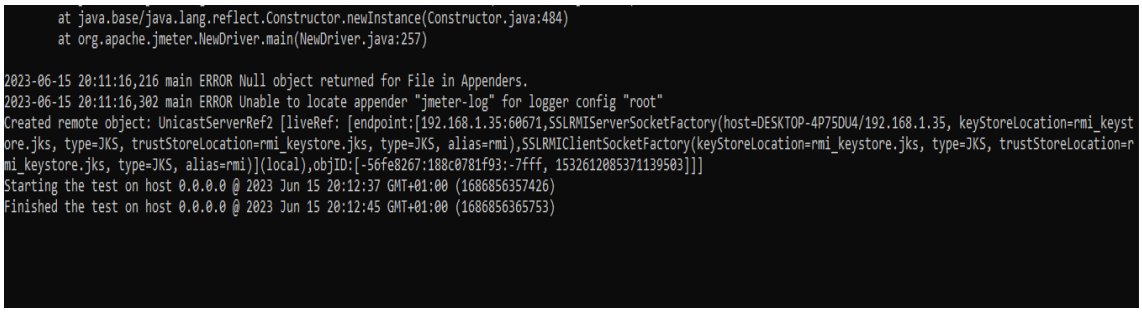

Figure 5.23: Jmeter Server On The Slave PC

3. **Run the Test Plan:**

   On the master PC run the load test, click on "Options" then "Remote Start all" (0.0.0.0 and 192.168.1.35). Or run JMeter in (non-GUI) mode using this command:

   jmeter -n -t [/path/to/testplan.jmx] -R [ip address of the slave machine:1099],[ip address of the local machine]

   -R: used to specify the IP addresses of remote (slave) machines.

   JMeter will distribute the load across the master and slave PCs using the round robin algorithm by distributing the load evenly on both servers.

5.6 Performance Analysis of the Load-Balanced System

In this section, we will analyze the performance of the load-balanced system in the alumni website comparing to the results obtained from the high load scenario, with the same workload distribution where:

- Number of Threads: 500
- Ramp-up Period: 10
- Loop Count: 3

Metrics Analysis

- Response Time: We observe a significant improvement in response time, with an average response time of 18 seconds.
- Throughput : The load balancing mechanism have a positive impact on the system's throughput. With achieving an average throughput of 12 requests per second.
- Error Rate: The error rate was 0.4% during this experiment, with negligible percentage of errors.
- Concurrent Users: The system successfully handled 500 concurrent users without performance degradation, by evenly distributing the load.

Table 5.7: Load Balancing Statics

Requests	**Executions**			**Response Time (ms)**			**Throughput**
Label	**Sample**	**Fail**	**Error %**	**Average**	**Min**	**Max**	**Transaction/s**
Total	**4500**	**18**	**0.4%**	**24419.34**	**5**	**228155**	**12.45**
Home	750	0	0.00%	46.97	5	859	1.11
Login	750	0	0.00%	92668.75	2259	228155	1.08
Post event	750	0	0.00%	4191.84	67	11564	1.63
Search Profile	750	0	0.00%	70581.62	2681	196959	1.39
Select Event	750	18	2.40%	2214.53	30	15368	1.50
Update Profile	750	0	0.00%	60812.35	1368	193441	1.54

5.6.1 Comparison with Non-Load-Balanced System

To evaluate the load balancing mechanism's performance against a non-load-balanced system in order to determine how effective it is for the alumni application.Both systems were put through the same load testing scenarios and setups in order to make a comparison. The comparison's main conclusions are as follows:

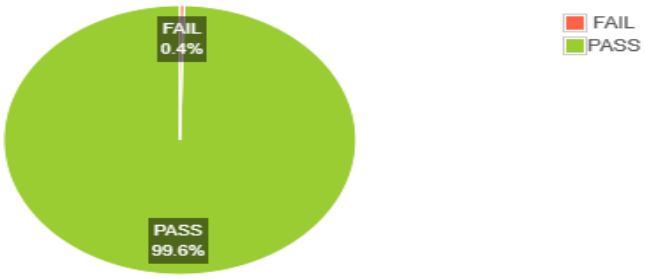

Figure 5.24: Requests Summary For the load Balanced system

1. **Response Time:** Compared to the non-load-balanced system, the load-balanced system showed faster response times. The load-balanced preserve constant response times under rising user demands, whereas the non-load-balanced system saw a increase in response times. In the load testing result, the load-balanced system achieved an average response time of 24 seconds, compared to an average response time of 28 seconds for the non-load-balanced system.

2. **Throughput:** Comparing the load-balanced and non-load-balanced systems, the load-balanced system had a major throughput. The system's average throughput with load balancing was 12 requests per second, compared to the non-balanced system's average throughput of 10 requests per second. The load balancing technique effectively divided the load between the two servers

3. **Reliability:** Load balancing positively impacted the reliability of the system. The load-balanced system exhibited a lower error rate compared to the non-load-balanced system. In the load testing result, the load-balanced system has an error rate of 00.4%, compared to an error rate of 35.88%for the non-load-balanced system.

It is clear from the comparison that the load balancing technique greatly boosts the alumni application's performance, scalability, and reliability when compared to a non-load-balanced system. Under different user loads, the load-balanced system maintained faster response times, greater throughput, and enhanced dependability. These results shows how crucial load balancing is to system performance optimization.

5.7 Conclusion

In this chapter we focused on load balancing and load testing in the context of the alumni website. In order to simulate actual user behavior, we looked at several test scenarios and workload distribution strategies. We were able to detect bottlenecks to manage system resources

and provide a flawless user experience by analyzing important performance metrics such as response times, throughput, and scalability.

The necessity of load testing and load balancing in establishing a high-performing website was highlighted throughout this chapter. by introducing Apache JMeter as a strong load testing tool. In order to manage heavy traffic and distribute workload across numerous server instances, load balancing has become essential. By analyzing the outcomes of these scenarios, we could determine the effect of load balancing on system behavior. These tests confirmed the load balancing mechanism's efficiency in enhancing the alumni website's performance and scalability.

GENERAL CONCLUSION

This work is mainly composed by two parts: Back-end and Front-end. For the Back-end: MySQL was our chosen database management system. MySQL offered reliable data storage and retrieval capabilities, allowing us to manage user profiles, alumni information, and other relevant data. We designed the database schema, created tables, and implemented necessary relationships to support the functionality of the alumni website. For the Front-end we created a platform called Alumni in order to provide a valuable space for alumni in purpose of fostering connections, collaboration, and engagement among former students, share professional opportunities, and contribute to the growth and development of their respective communities, using the Angular framework. Angular provided us with a robust framework to create interactive web pages, handling user interactions, and connecting to the back-end services. We utilized Angular's components, services, and routing capabilities to build a seamless user experience.

For the load testing and load balancing simulation the Apache JMeter was the chosen, in order to validate the performance and scalability of the alumni website, JMeter allowed us to simulate different user loads, measure response times, and analyze the system's behavior under stress. By configuring different HTTP requests, defining test data, and implementing assertions, we colected valuable insights into the website's performance characteristics.

By exploring the concept of load balancing and its role in distributing incoming traffic across multiple servers. Where the load balancing plays a critical role in maintaining high availability, improving performance, and ensuring scalability. we could evaluate how workload distribution affected resource utilization and overall system performance.

The purpose of this study is to develop a Alumni website for Amar Telidji University of Laghouat that can handle a significant number of concurrent users. The knowledge obtained

from this thesis can be applied to other web development projects. By integrating load testing and load balancing techniques, developers can create robust and scalable web applications that provide a an exceptional user experience.

Future perspectives

First, from a future perspective we would like to apply and explore the sophisticated load balancing techniques and methods such as predictive load balancing, dynamic load balancing algorithms and scpecialy machine learning based load balancing. These techniques have more capability to adapt to changing traffic patterns and to make intelligent load-balancing decisions, which improved a hight performance and scalability comparing to the static techniques.

Another method we would like to attempt in the future, is to use cloud based Load Balancing methods. These methods offer sophisticated load balancing features, scalability and flexibility. Trying options such as Amazon Elastic Load Balancer or Google Cloud Load Balancer it will be further more enhance the load balancing capabilities of the Alumni application.

Next step, is implementing advanced security tests, the security testing play a crucial role in Alumni web application, because it's dealing with sensitive user data, to identify vulnerabilities, and protect against common web application security threats, such as cross-site scripting (XSS) and SQL injection.

In side of Alumni application scalability, the implementing of performance testing strategy ensures that the alumni website's performance is monitored, for making improvements to the Alumni website's design by implement new features based on users expectations . and continuously ameliorate the load balancing strategy based on real world usage.

BIBLIOGRAPHY

[1] Omid H. Jader, Subhi R. M. Zeebaree, Rizgar R. Zebari, A State Of Art Survey For Web Server Performance Measurement And Load Balancing Mechanisms, December 2019, ISSN 2277-8616.

[11] Shantanu Shukla,Raghuraj Singh Suryavanshi,Survey on Load Balancing Techniques,ResearchGate, February 2019,

[40] Dmitri Nevedrov, Using JMeter to Performance Test Web Services, International Journal of Advanced Research in Computer Science, 08/02/2006.

[41] Jha, Nisha; Popli, Rashmi, Comparative Analysis of Web Applications using JMeter, International Journal of Advanced Research in Computer Science, Mar/Apr2017, Vol. 8 Issue 3, p774-777. 4p.

[2] https://www.nginx.com/resources/glossary/load-balancing/, [consulted on 30/05/2023].

[3] https://www.cloudflare.com/learning/performance/what-is-load-balancing/, [consulted on 30/05/2023].

[4] https://www.ibm.com/topics/load-balancing , [consulted on 01/06/2023].

[5] https://avinetworks.com/what-is-load-balancing/,[consulted on 01/06/2023].

[6] https://www.g2.com/articles/load-balancer, [consulted on 01/06/2023].

[7] https://www.f5.com/glossary/load-balancer, [consulted on 01/06/2023].

[8] https://www.enjoyalgorithms.com/blog/types-of-load-balancing-algorithms ,[consulted on 02/06/2023].

[9] https://kemptechnologies.com/what-is-load-balancing , [consulted on 02/06/2023].

[10] https://www.appviewx.com/education-center/load-balancer-and-types/,[consulted on 02/06/2023].

[12] https://www.educative.io/answers/what-is-the-weighted-round-robin-load-balancing-tech [consulted on 03/06/2023].

[13] https://kemptechnologies.com/resources/glossary/source-ip-hash-load-balancing, [consulted on 03/06/2023].

[14] https://www.cloudflare.com/learning/performance/types-of-load-balancing-algorithms/, [consulted on 04/06/2023].

[15] https://kemptechnologies.com/load-balancer/load-balancing-algorithms-techniques ,[consulted on 04/06/2023].

[16] https://www.devgraph.com/resource/why-is-it-important-to-use-load-balancing\-for-your-web-application/, [consulted on 04/06/2023].

[17] https://www.cloudflare.com/learning/performance/cloud-load-balancing-lbaas/,[consulted on 04/06/2023].

[18] https://avinetworks.com/glossary/load-balancing-as-a-service/, [consulted on 04/06/2023].

[19] https://www.webwerks.in/blogs/what-cloud-load-balancing-and-what-are-its-benefits, [consulted on 17/06/2023].

[20] https://www.cloudflare.com/learning/performance/cloud-load-balancing-lbaas/, [consulted on 04/06/2023].

[21] https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/,[consulted on 07/06/2023].

[22] https://creately.com/blog/diagrams/uml-diagram-types-examples/, [consulted on 07/06/2023].

[23] https://creately.com/blog/diagrams/uml-diagram-types-examples/#UseCaseDiagram,[consulted on 07/06/2023].

[24] https://www.ibm.com/docs/en/rsm/7.5.0?topic=uml-sequence-diagrams, [consulted on 07/06/2023].

[25] https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/, [consulted on 30/05/2023].

[26] https://www.methodsandtools.com/tools/staruml.php , [consulted on 16/06/2023].

[27] https://www.hostinger.com/tutorials/what-is-mysql, [consulted on 07/06/2023].

[28] https://www.phpmyadmin.net/ ,[consulted on 07/06/2023].

[29] https://www.postman.com/api-platform/, [consulted on 07/06/2023].

[30] https://www.knowledgehut.com/tutorials/angular/getting-started, [consulted on 07/06/2023].

[31] https://influencermarketinghub.com/glossary/html/, [consulted on 07/06/2023].

[32] https://www.javatpoint.com/what-is-css, [consulted on 07/06/2023].

[33] https://thenewstack.io/what-is-typescript/, [consulted on 07/06/2023].

[34] https://linuxhint.com/bootstrap-5/ ,[consulted on 07/06/2023].

[35] https://upstackhq.com/blog/software-development/what-is-angular-material,[consulted on 07/06/2023].

[36] https://code.visualstudio.com/docs, [consulted on 07/06/2023].

[37] https://stytch.com/blog/what-is-password-hashing/, [consulted on 16/06/2023].

[38] https://portswigger.net/web-security/sql-injection, [consulted on 16/06/2023].

[39] https://www.loadview-testing.com/jmeter-load-performance-testing-tutorial/, [consulted on 30/05/2023].

[42] https://jmeter.apache.org/, [consulted on 30/05/2023].

[43] https://octoperf.com/blog/2017/10/19/how-to-analyze-jmeter-results/#pre-requisites, [consulted on 08/06/2023].

[44] https://www.webomates.com/blog/software-testing/performance-testing/,[consulted on 09/06/2023].

[45] https://jmeter.apache.org/usermanual/glossary.html,[consulted on 09/06/2023].

[46] https://jwt.io/introduction,[consulted on 18/06/2023].