

CAPSTONE PROJECT FINAL REPORT

Submitted to
Computer Science Department
College of Computing Sciences and Engineering
Kuwait University

Advisor

Dr. Mohamed Smaoui

Group Members

Talal Hamed - 2142128779 - Group Leader

El-Husseiny Zaghloul - 2151119762 - Member

25 November 2019

Acknowledgements

To our families and loved ones who supported us throughout life, and to our mentors and teachers who have inspired us and shaped us to whom we are today.

TABLE OF CONTENTS

1. <i>Introduction and Background</i>	8
2. <i>User and System Requirements</i>	9
3. <i>System Architecture</i>	11
4. <i>System Design Artifacts</i>	13
5. <i>Implementation Details</i>	35
5.1. Frameworks and Platforms.....	35
5.2. Component and Code Reuse	37
5.3. CASE Tools	37
6. <i>Modifications of the Original Plan (Optional)</i>	41
7. <i>Testing</i>	42
10. <i>LESSONS LEARNED</i>	88
11. <i>Conclusion</i>	89
<i>References</i>	90

Table of Figures

Figure 1: System overview architecture	11
Figure 2: ec2 Instance design/architecture	12
Figure 3: ER diagram	13
Figure 4: comment table	14
Figure 5: country table	14
Figure 6: discipline table	14
Figure 7: enrollment table	14
Figure 8: journal table	14
Figure 9: mention table	14
Figure 10: organization table	15
Figure 11: readingList table	15
Figure 12: role table	15
Figure 13: settings table	15
Figure 14: pecialty table	15
Figure 15: subspecialty table	15
Figure 16: user table	15
Figure 17: view table	16
Figure 18: login table	20
Figure 19: signup as member page	21
Figure 20: signup as organizer page	21
Figure 21: feed page	22
Figure 22: article page	22
Figure 23: article page (writing a comment)	23
Figure 24: article page (posting the comment from figure 23)	23
Figure 25: article page (mentioning another user in a comment)	24
Figure 26: feed page (notification label shows 1 for the comment that was mentioning this user from figure 25)	24
Figure 27: notification page	25
Figure 28: user profile page	25
Figure 29: user profile page (cont)	26
Figure 30: user profile page (highlighting dropdown menu on top right corner)	26
Figure 31: user edit profile page	27
Figure 32: user edit profile page (cont)	27
Figure 33: USER EDIT PROFILE PAGE (CONT)	28
Figure 34: organization profile page	28
Figure 35: organization profile page (cont)	29
Figure 36: organization profile page (cont)	29
Figure 37: organization profile page (cont)	30
Figure 38: administration dashboard on trello	31
Figure 39: overview/task dashboard on trello for monitering work between team members	32
Figure 40: database tables on phpmyadmin	32
Figure 41: instances deployed on elastic cloud	33
Figure 42: discriptive messures for instances deployed on the elastic cloud	33
Figure 43: bitbucket repository (highlighting different branches)	34
Figure 44: : bitbucket repository (highlighting commits from different team members)	34
Figure 45: : bitbucket repository	35

Figure 46: ETL data integration process	40
Figure 47	42
Figure 48	43
Figure 49	43
Figure 50	44
Figure 51	44
Figure 52	45
Figure 53	45
Figure 54	46
Figure 55	46
Figure 56	47
Figure 57	47
Figure 58	48
Figure 59	48
Figure 60	49
Figure 61	49
Figure 62	50
Figure 63	50
Figure 64	51
Figure 65	51
Figure 66	51
Figure 67	52
Figure 68	52
Figure 69	53
Figure 70	53
Figure 71	54
Figure 72	54
Figure 73	55
Figure 74	55
Figure 75	56
Figure 76	56
Figure 77	57
Figure 78	57
Figure 79	58
Figure 80	58
Figure 81	59
Figure 82	59
Figure 83	60
Figure 84	60
Figure 85	61
Figure 86	62
Figure 87	62
Figure 88	63
Figure 89	64
Figure 90	64
Figure 91	65
Figure 92	66
Figure 93	66

Figure 94	67
Figure 95	67
Figure 96	68
Figure 97	69
Figure 98	69
Figure 99	70
Figure 100	71
Figure 101	71
Figure 102	72
Figure 103	72
Figure 104	73
Figure 105	74
Figure 106	74
Figure 107	75
Figure 108	75
Figure 109	76
Figure 110	76
Figure 111	76
Figure 112	80
Figure 113	80
Figure 114	81
Figure 115	81
Figure 116	82
Figure 117	82
Figure 118	83
Figure 119	83
Figure 120	84
Figure 121	84
Figure 122	85
Figure 123	85
Figure 124	86
Figure 125	87
Figure 126	87

Abstract

One major problem that is currently facing the medical field is that the doctors after graduation stop renewing their medical knowledge with the latest discoveries. Our project tackles this problem by providing a platform that allows them to stay up to date in their field. Our system also provides an element of community to the users where they can share their opinions and input on a certain article. The system also provides motivational factors that motivate the users to read more and stay active on our platform.

1. INTRODUCTION AND BACKGROUND

In an age of rapid advances in medical discoveries and continuous development of therapeutic models and practices, it is more important than ever for physicians to keep their medical knowledge up to date. Medications that used to be prescribed a few years ago could be considered illegal today [1]. Recommended treatment plans for diagnosis across the medical spectrum keep changing as new clinical trials are conducted and new findings emerge [2, 3, 4]. Physicians are in a continuous race with time to keep their knowledge relevant and simultaneously successfully lead balanced family and professional lives.

Policy makers in countries with leading healthcare establishments have realized the danger of medical knowledge stagnation on society and health and have imposed many legal requirements on physicians to maintain their license to practice [5, 6, 7]. Of those requirements are the Continuing Professional Development yearly credits that various physicians have to accumulate to keep their licenses [8]. Physicians scout for educational venues such as accredited conferences, talks, and workshops to collect hour credits to maintain their licenses. These opportunities are designed in essence to take physicians out of their physical practice and expose them to new medical findings or refresh their knowledge on certain topics.

Geography plays a huge role in the type of workshops and continuing knowledge that a physician is exposed to. Physicians who live in a cosmopolitan city such as Boston have easier access to workshops at top Ivy-League institutions and leading hospitals compared to other physicians who practice in developing countries or remote areas [9, 10]. However, in an age conquered with extraordinary technology advances and the burst of social networks, the problem of physician “access” to knowledge could be mitigated. Social media has connected over 2 billion people worldwide and has become at many times the forefront of news! Through this work, we would like to explore how to use elements of social media to sort and organize medical “news” and findings.

Our project’s purpose is to make as easy as possible to access these latest articles and publications. We have created a software that grants doctors in the medical field the ability to access these publications seamlessly with the click of a button. Our system software system gathers all new findings and publications from top class publishers and writers and organizes it all in one place. the users will be able to access these files on the go. They will receive notifications regarding any new findings in their field. Each user will get his own personalized feed based on his/her professional field in the medical sciences. Our software also groups doctors that work in the same organization and gives their administrators the ability to keep track of their reading process, so They can tell whether they were up to date with the latest discoveries or not.

The tricky part about our system is that it will be fully automated. Which means that the fetching and sorting of the data will all be based on our AI algorithm which will gather the newest articles from the various medical sources, stores them into our database, sorts them for easy access, and then filter these findings and display them onto the users feed. this filtering algorithm will allow

the system to display an optimized, personalized feed based on each users personal profession in the various medical fields.

Surely, an application such as this, many challenges and risks shall arise in the building process. One of main challenges that concerns us as a team is handling the different APIs that the system will connect to in order to gather the millions of records and store them into our database. Another concern is finding the means to store all these records in a way that would allow for easy access later on. Another major concern would be finding the “perfect” design that would suit our audience just right so that they could use our system for as long as they need without any distress or discomfort. Finally, writing the algorithm that would filter each user’s feed to only show articles that relate to his field of interest could prove a bit challenging, but nothing that can’t be handled with the proper planning, dedication, and team cooperation.

2. USER AND SYSTEM REQUIREMENTS

2.1. FUNCTIONAL CABALITIES

This section describes the functional/user requirements which basically represent the capabilities that the user will be able to accomplish with our system.

2.1.1. GENERAL USER CAPABILITIES

- The user could either signup as an organizer/admin or a member of an organization.
- The user could join or administrate multiple organizations at the same time.
- The user could search for any medically related topic through all the articles in our database.
- The user will be presented with his own personalized feed page when he logs into our system. The feed page contains the latest published articles related to the user’s profession.
- The user could add any article to his reading list to access it easily later on.
- The user could comment on any article expressing his input or opinion on the topic.
- The user could read comments posted by other users.
- The user could mention any other user in a comment by mentioning the target user’s email.
- The user receives a notification when another user mentions them in the comment section.
- The user could view all his unread notifications from the notification page.
- The is presented with a link leading the user to the actual page of the published article.
- The user could edit all his personal info from his profile page at any time after registration.

- The user could view any other user's profile page which shows the public info and a biography of this user.
- The user could view statistics and charts that show the activities of the other users in the same organization.
- The organizer of any organization could assign other members of the same organization to become shared organizers of this organization.

2.1.2. SYSTEM ADMIN CAPABILITIES

- The admin of the system is capable of blocking an existing active system user.
- The admin has access to info regarding the systems specs (e.g. system users, organizations, articles, journal etc....)
- The admin has the ability to change his password
- The admin has access to the list of all system users and could browse through them freely.
- The admin has access to the list of all system organizations and browse through them freely
- The system's admin sets the interval between each opening of the document by the same user in which this opening would increase the number of views assigned to this article (for example if the interval is set to 15 mins then if the same user opens the same article more than once within the 15 mins, it only counts as though the user has only viewed it once and would only increase the views of this article by one. Although if the user re-opens the article after 15 mins from the previous opening, then the views of the article would increment by one.).
- The admin also sets the number of comments allowed per day for all users, to prevent DDOS attack and whatnot.

2.2. SUPPLEMENTAL REQUIREMENTS

This section describes the requirements that describe the system's performance and what we, as developers expect the system to perform as based on the functionalities and design decisions that we chose through the process of developing this system.

- The system is secure and follows the code of ethics regarding the user's private information and what not. The user's personal information will not be altered with or distributed or used in any way possible without the user's permission.
- The system is fast and handles high pressure efficiently based on multiple tests that we performed on the system. These tests will be explained thoroughly in a later section of this report.
- The system presents accurate and reliable data for the users. The data presented in the system was gathered from the databases of NCBI and PubMed which are both branches of the National Institutes of Health in united states.

- The system is easily extensible and scalable thanks to the use of CodeIgniter's PHP framework that follows the model-view-controller model. We deployed the system on Amazon Web Services to allow for a scalable architecture.

3. SYSTEM ARCHITECTURE

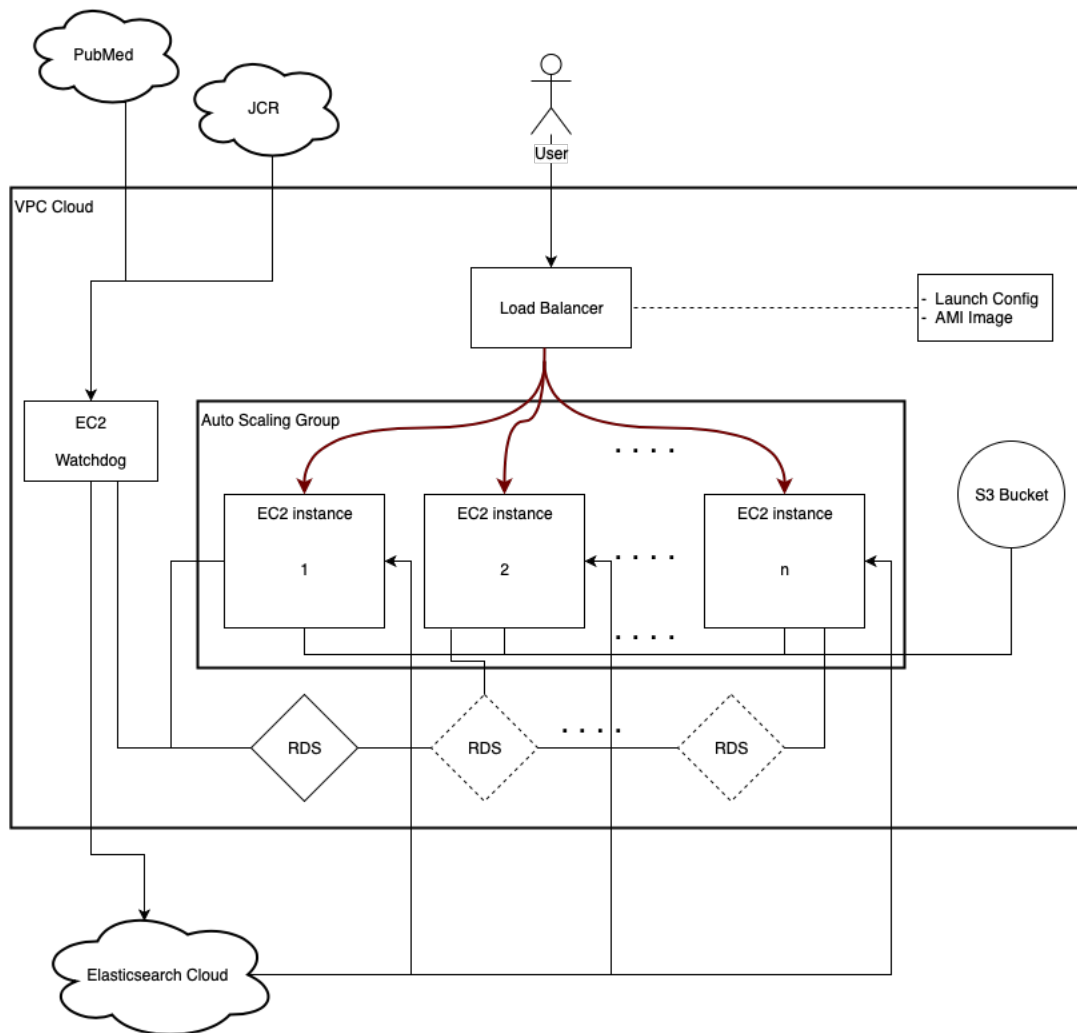


FIGURE 1: SYSTEM OVERVIEW ARCHITECTURE

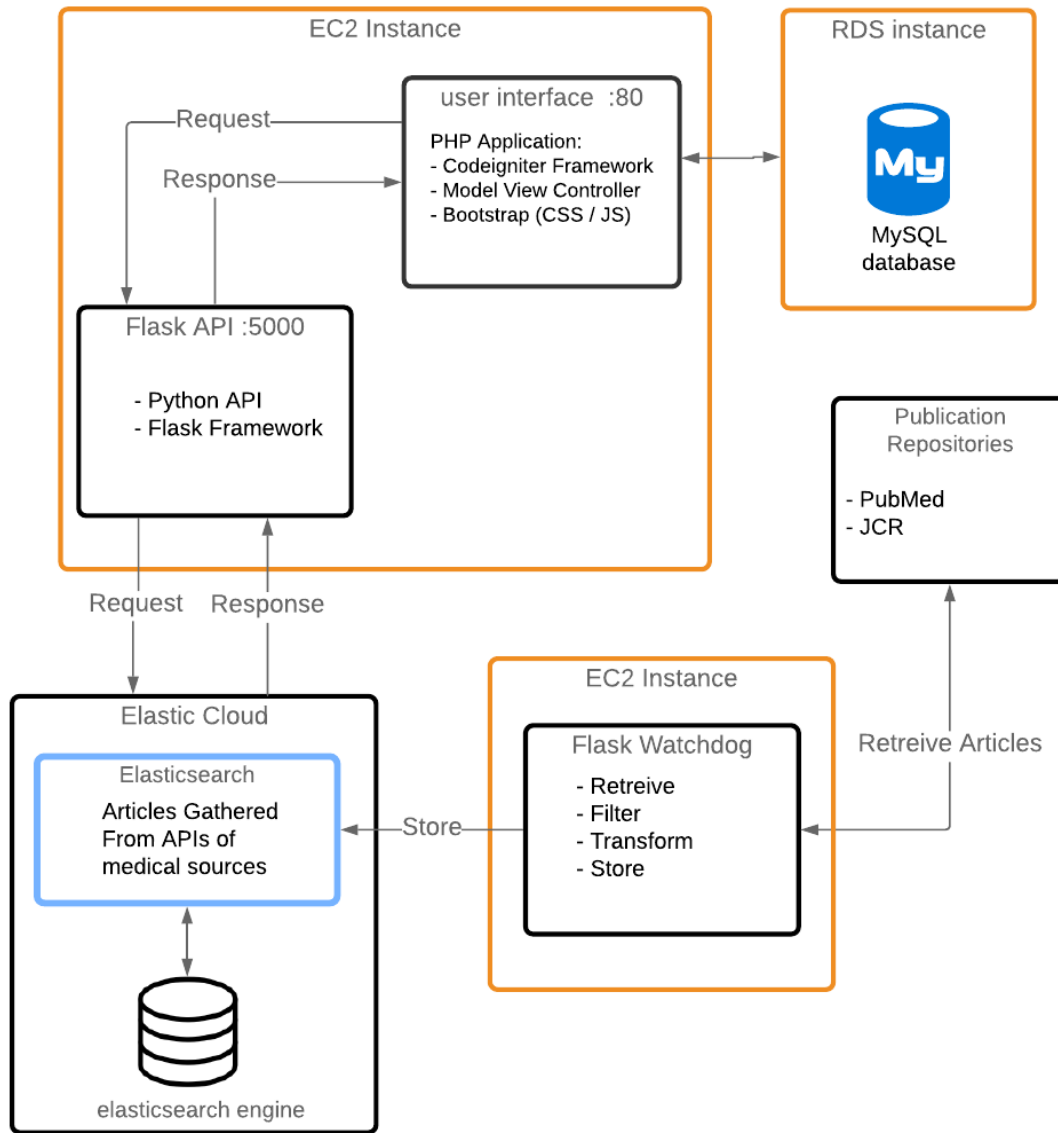


FIGURE 2: EC2 INSTANCE DESIGN/ARCHITECTURE

4. SYSTEM DESIGN ARTIFACTS

4.1. Database Design and Tables

4.1.1. ER Diagram

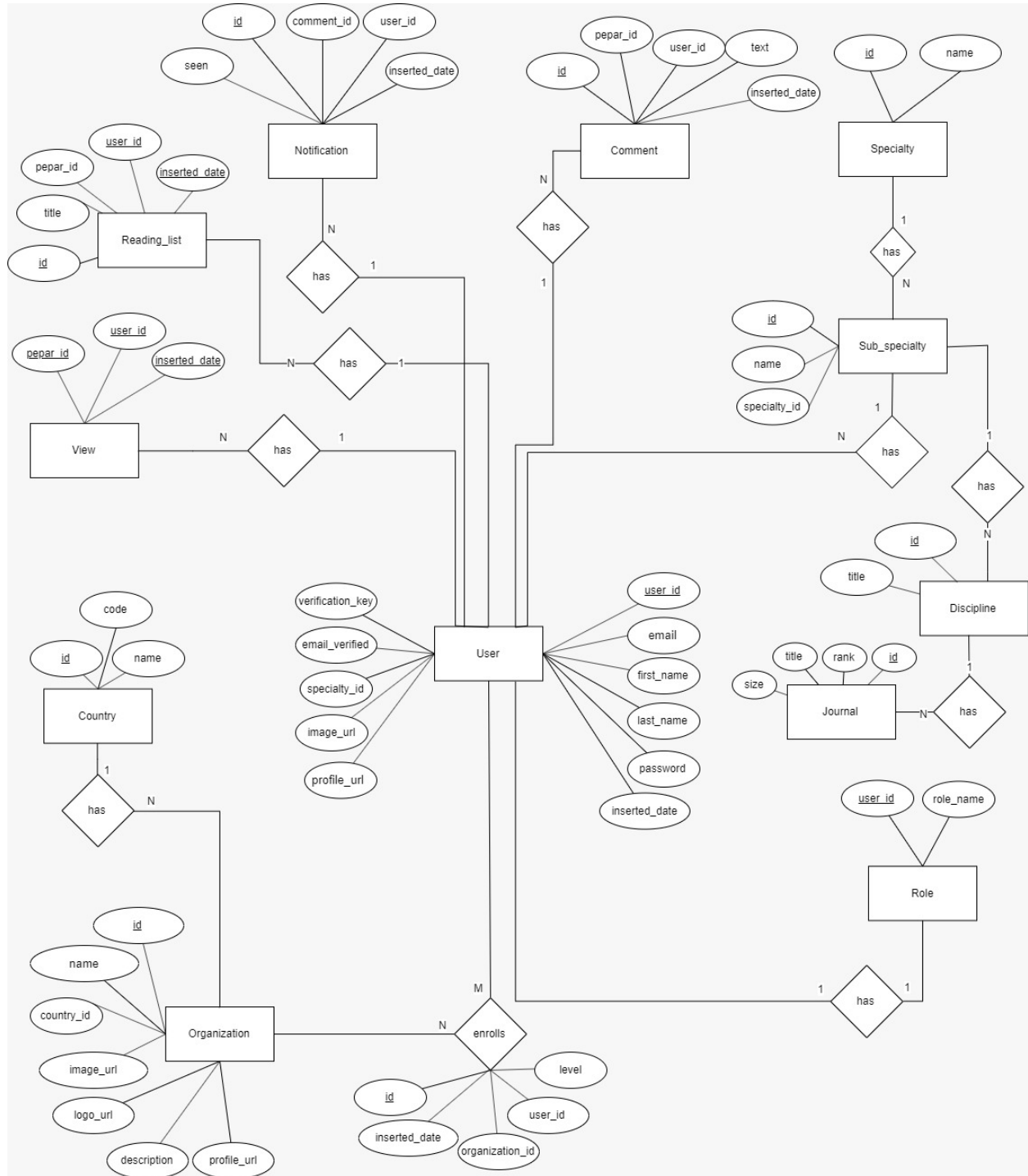


FIGURE 3: ER DIAGRAM

4.1.2. Database Tables

Field	Type	Null	Key	Default	Extra
comment_id	int(11)	NO	PRI	NULL	auto_increment
paper_id	varchar(50)	NO	MUL	NULL	
user_id	int(11)	NO	MUL	NULL	
text	text	NO		NULL	
inserted_date	datetime	NO		current_timestamp()	

FIGURE 4: COMMENT TABLE

Field	Type	Null	Key	Default	Extra
country_id	int(11)	NO	PRI	NULL	auto_increment
country_name	varchar(100)	NO		NULL	
country_code	varchar(3)	NO		NULL	

FIGURE 5: COUNTRY TABLE

Field	Type	Null	Key	Default	Extra
discipline_id	int(11)	NO	PRI	NULL	auto_increment
discipline_subspecialty_id	int(11)	NO	MUL	NULL	
discipline_title	varchar(250)	NO		NULL	

FIGURE 6: DISCIPLINE TABLE

Field	Type	Null	Key	Default	Extra
enrollment_id	int(11)	NO	PRI	NULL	auto_increment
organization_id	int(11)	NO	MUL	NULL	
user_id	int(11)	NO	MUL	NULL	
level	varchar(10)	NO		NULL	
enrollment_date	datetime	NO		current_timestamp()	
creator	int(1)	NO		0	

FIGURE 7: ENROLLMENT TABLE

Field	Type	Null	Key	Default	Extra
journal_id	int(11)	NO	PRI	NULL	auto_increment
journal_discipline_id	int(11)	NO	MUL	NULL	
journal_rank	int(11)	NO		NULL	
journal_title	varchar(250)	NO		NULL	
journal_groupSize	int(11)	NO		NULL	

FIGURE 8: JOURNAL TABLE

Field	Type	Null	Key	Default	Extra
mention_id	int(11)	NO	PRI	NULL	auto_increment
user_id	int(11)	NO	MUL	NULL	
is_seen	int(1)	NO		0	
inserted_date	datetime	NO		current_timestamp()	
comment_id	int(11)	NO		NULL	
sender_id	int(11)	NO	MUL	NULL	
article_title	text	NO		NULL	
article_id	varchar(50)	NO		NULL	

FIGURE 9: MENTION TABLE

Field	Type	Null	Key	Default	Extra
organization_id	int(11)	NO	PRI	NULL	auto_increment
organization_name	varchar(255)	NO		NULL	
country_id	int(11)	NO	MUL	NULL	
image_url	text	NO		NULL	
organization_profile_url	varchar(255)	NO		NULL	
logo_url	text	NO		NULL	
organization_desc	varchar(500)	NO		NULL	

FIGURE 10: ORGANIZATION TABLE

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
paper_title	text	NO		NULL	
paper_id	varchar(50)	NO		NULL	
user_id	int(11)	NO	MUL	NULL	
inserted_date	datetime	NO		current_timestamp()	

FIGURE 11: READINGLIST TABLE

Field	Type	Null	Key	Default	Extra
user_id	int(11)	NO	PRI	NULL	
user_role	varchar(25)	NO		NULL	

FIGURE 12: ROLE TABLE

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
article_views	int(11)	NO		15	
comments_per_day	int(11)	NO		100	

FIGURE 13: SETTINGS TABLE

Field	Type	Null	Key	Default	Extra
specialty_id	int(11)	NO	PRI	NULL	auto_increment
specialty_name	varchar(100)	YES		NULL	

FIGURE 14: PECIALTY TABLE

Field	Type	Null	Key	Default	Extra
subspecialty_id	int(11)	NO	PRI	NULL	auto_increment
subspecialty_name	varchar(100)	YES		NULL	
subspecialty_specialty_id	int(11)	YES	MUL	NULL	

FIGURE 15: SUBSPECIALTY TABLE

Field	Type	Null	Key	Default	Extra
user_id	int(11)	NO	PRI	NULL	auto_increment
user_email	varchar(255)	NO	UNI	NULL	
user_password	varchar(255)	NO		NULL	
user_first_name	varchar(50)	NO		NULL	
user_last_name	varchar(50)	NO		NULL	
user_submitted_date	datetime	NO		current_timestamp()	
user_profile_url	varchar(255)	NO		NULL	
user_img_url	text	NO		NULL	
subspecialty_id	int(11)	YES	MUL	NULL	
is_email_verified	int(1)	NO		0	
verification_key	text	NO		NULL	
brief	text	YES		NULL	
brief_last_updated_date	datetime	YES		NULL	
last_visit_date	datetime	YES		NULL	
isblocked	int(1)	NO		0	

FIGURE 16: USER TABLE

Field	Type	Null	Key	Default	Extra
paper_id	varchar(50)	NO	PRI	NULL	
user_id	int(11)	NO	PRI	NULL	
inserted_date	datetime	NO	PRI	current_timestamp()	

FIGURE 17: VIEW TABLE

4.2.TECHNICAL DESCRIPTION

The first interaction between the user and the system will be through the signup page. Once the user opens up our website, he is presented with the signup page. In the signup page the user registers his information on our database so he can reenter our website again and start from where he previously left off. The user has two ways of registering into our system. The user could either signup as a member of an existing organization, or as an organizer of a new organization. The first step of either ways is the same. Although each method has its own form, but the first few fields of the two forms are the same. All of the new users are asked to enter their credentials and personal info which consist of the user's first name, last name, email, password, confirmation of the password, a profession, and a sub-profession. After that depending on the user's choice, he is either asked to enter the code of the organization he would like to join, which in this case he will be a member of this organization. Or the user could choose to enter the name and country of the organization he would like to create. Which in this case he will be the organizer of this organization. More on the capabilities of the member and organizer will be cleared later on. After the user enters his data, the data is stored in our database along with a verification flag that shows whether the user verified his account or not. When the user enters his data and submits the form, he is then redirected to a page that tells him to has a message that states "in order to login, please verify your account by clicking on the link we sent to your email." and a verification link is sent to his email. Once the user clicks on that link, he is presented with a page which confirms that the process of verifying his account was successful. After that, the verification flag will turn into 1 and the user will be able to log into our system.

The login page is where the user logs into the system after he creates his account and verifies it via the verification link sent to his email after signup completion. The login page has a form with two fields. An email and a password field. The user enters his email and the password associated with that

email which he chose when he first signed up into our system. Our system takes the user's email and password to confirm the user's existence in our database. The user's email is compared as is. As for the password it is decrypted and compared with the decrypted password in our database. If the two fields are confirmed and deemed correct by our system, the user is redirected to his feed page. Otherwise he is shown a message that tells the user that the info he entered is incorrect and that he should try again.

the feed page is the page that the user is redirected to after he successfully signs in, although there is a case where the user would be redirected to the organization's profile page, but this case will be discussed later on. The feed page is divided into four main parts. The upper side of the page holds a navigation bar where the user could access his notifications, profile page, organizations he is involved with, or a link that leads him back to the feed page which he is currently in. this navigation bar exists throughout the rest of the website's pages. Below the navigation bar is where the user has quick access to the users he's following, his personal feed, the organizations he's currently involved with, and a list of the articles he most recently read. The list of users the user has followed are displayed on the right side of the page. Next to that at the very center of the page exists the user's personalized list of articles that are suggested by our system. This list will show the most recently published articles that are related to the user's profession. The user could scroll down the feed page to view more articles. If the list ends the user could choose to load more articles down below. We chose to only display the 10 most recently published articles so the user wouldn't be overwhelmed by the number articles shown. If user wishes to view more article, he could scroll down to the last article which has a "load" button right below it for the user to load more articles from Elasticsearch. Now on the right side of the page next to the user's feed there exist two quick access boxes that are sorted over one over the other. The upper box contains a list of the organizations that the user is involved with. The lower box contains a list of the most recent articles that the user has read. Only the titles of the articles are shown in the list of articles in the lower box. As for the articles displayed in the user's feed, the articles title, snippet of the abstract, keywords related to article, publisher, journal, and date of the articles publishing are displayed. If the user wishes to view the entire abstract of the article, he could click on the article which will take him to the full article's page.

The article's page holds more information about the article than the info displayed in the feed page. This page displays the entire article for the user, including the title, keywords, author, journal, and date of the publication. Another main part of the article's page is the comment section. This section is where the users could express their insights and thoughts about the article. They could hold discussions in this section. The comments written can be viewed by any user that opens the article's page. The comment section has a box that allows the user to write his own comment and below that is the list of comments associated to this article which were written by other users. The comments are sorted by their date starting by the most recent and going down. Each comment box shows the name of the user that wrote this comment and the user click on that name to open the other user's profile. Another great use of the comment section is that the user could mention each other in any comment. So, for example, the user could suggest an article for his colleague by mentioning him in the comment.

Next on the list is the user's profile page. The user's profile page is where the user has access to all his personal info that he provided on registration. The user could also give his profile a bit of a glow by choosing a personal profile picture representing him and a cover picture that brightens up his/her profile page. The information stored in the user's profile page is up for changing at any time by the himself, including the email and password assuming of course that no other user is using the email that the user is trying to change to. In the profile page the user could also post a small biography that gives the other users that open his account an idea about himself. The user is the only one that is able to change his personal info. When other users choose to open the user's profile page they are only able to view his public info. The user's public info mainly consists of the user's full name, email, profile and cover pics, and of course the user's biography.

Ok, now that we're done with the user's profile page, let's move on to the organizations profile page. the organization's profile page is a little different from the user's profile page in terms of privacy. The organization's page is completely viewable by any user. Since the organization's profile page doesn't have any sensitive info, the users can view everything about the organization. any organization only has two fields that identify it, these two fields are the organization's name and country. These combination of these two fields is what separates an organization from other organizations. The admin of the

organization could also write a small biography about the organization in the profile page. the admin of the organization could also show the organizations logo and give it a cover image that both show in the organization's profile page. only the organizations admin is able to edit the organization's data, but as I've mentioned before, the information is viewable by any user in the organization's profiles. One thing to point out is that when a member or admin of an organization opens the organizations page, they get different views than the users that are not involved with the organization. When a member or admin of an organization open the organization's page, they get something that no one else has access to. This difference is what makes our platform special. The extra thing that the members of an admin of an organization can view is the statistics of the members of the organization. These statistics will play a huge factor in the motivating the users to read more and stay up to date with the latest discoveries. The statistics will show how active each user is and what they most recently read and so on.

Now that we're done with the organization part, let's talk about a very feature in our system. This feature is the notification system. Our system will send the users notifications when new publications in their field have been published and will also notify them if any user mentions them in an article. Now let me explain how this feature works behind the scenes. First when the user enters the system, he opens a socket that connects him to the notification API. The notification API (pusher) provides each user with their unique id. So, when the user for example mentions another user in a comment, this comment is sent to the notification API and will have a destination that is set to the mentioned user. Once that user enters the system, he also opens a portal with the notification API and gets notified of any notifications that were addressed to him.

Now that the main parts of the system are explained, below are screenshots of our system with some extra details highlighted in each page.

4.3. SYSTEM SCREENSHOTS



FIGURE 18: LOGIN TABLE



MedicalPlus

DISCIPLINES
68

JOURNALS
5652

ARTICLES
2 Million

[Sign in](#) [Sign up](#)

As Member **As Organization**

Email Address

Password

Repeat Password

First Name

Last Name

-- Select Specialty --


-- Select Sub Specialty --

Organization ID

Get Started



FIGURE 19: SIGNUP AS MEMBER PAGE



MedicalPlus

DISCIPLINES
68

JOURNALS
5652

ARTICLES
2 Million

[Sign in](#) [Sign up](#)

As Member **As Organization**

Email Address

Password

Repeat Password

First Name

Last Name

-- Select Specialty --

-- Select Sub Specialty --

Organization Name

-- Select Country --

Get Started

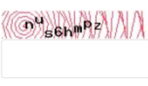


FIGURE 20: SIGNUP AS ORGANIZER PAGE

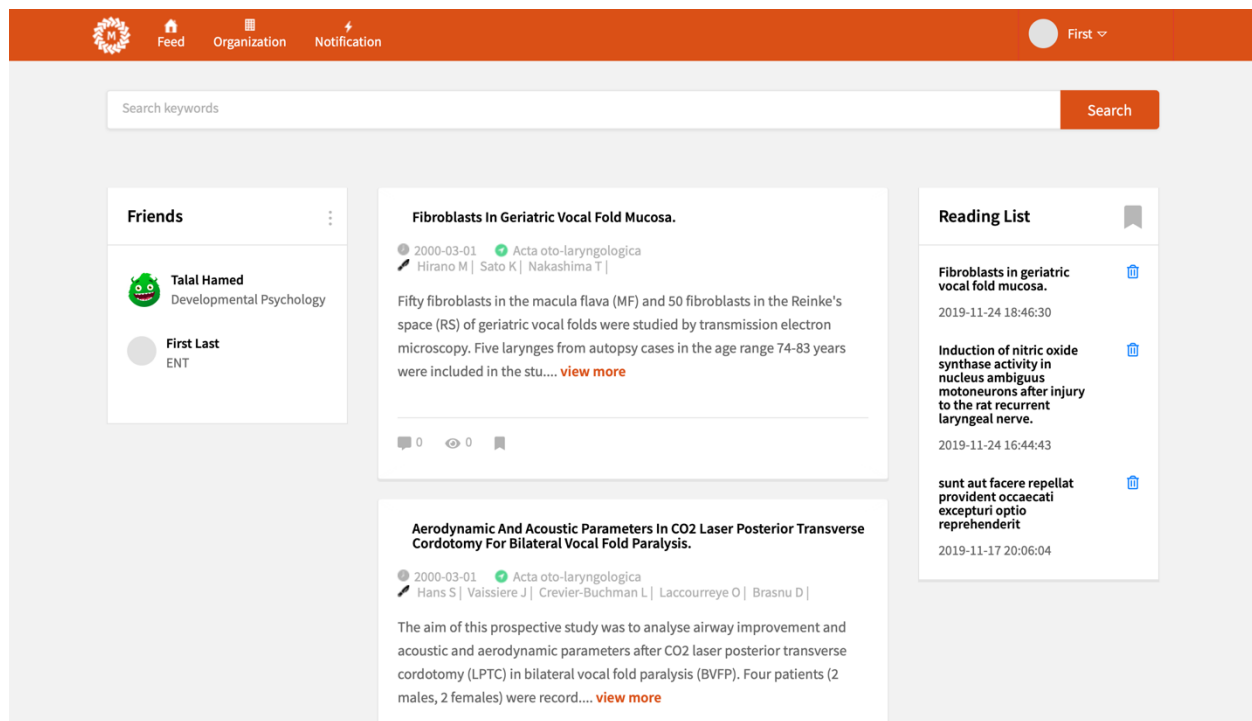


FIGURE 21: FEED PAGE

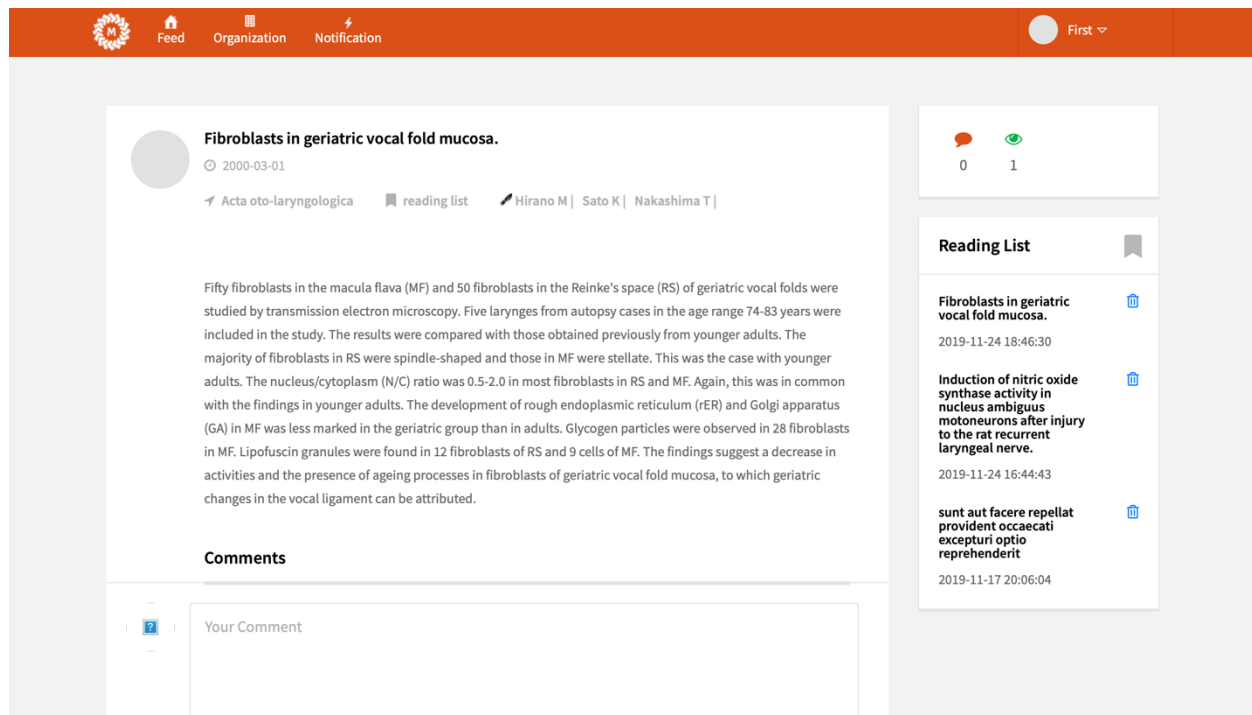


FIGURE 22: ARTICLE PAGE

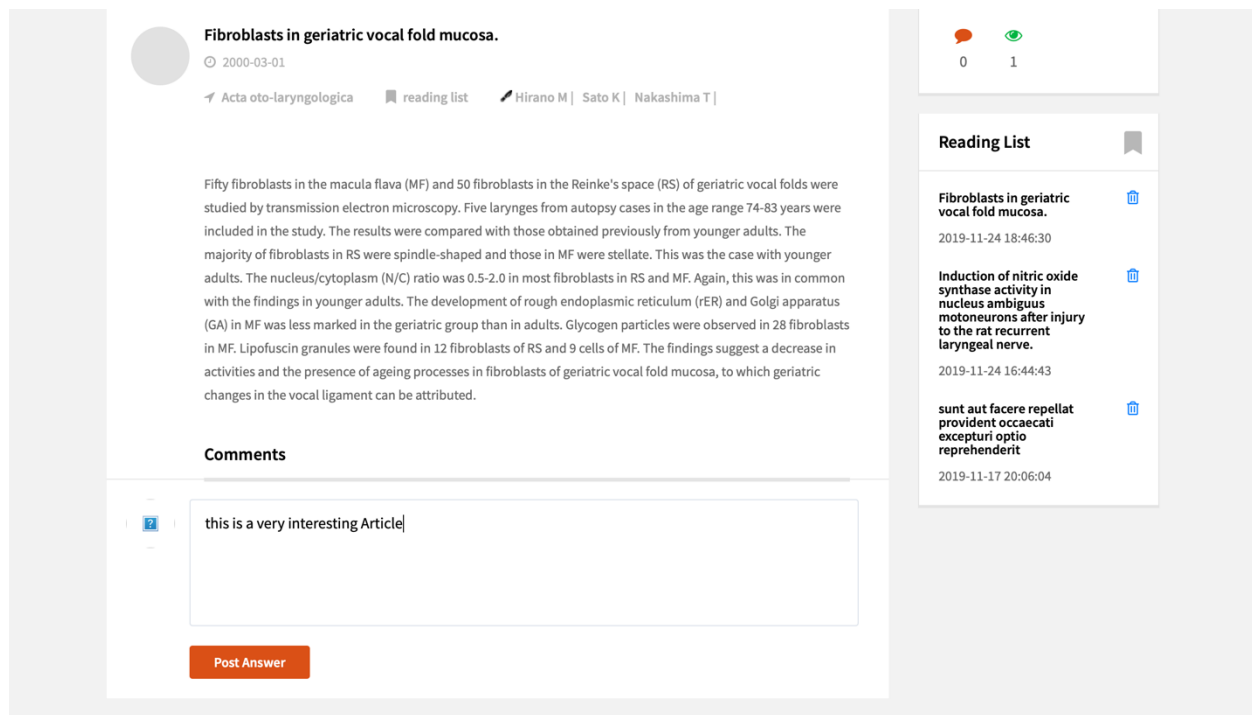


FIGURE 23: ARTICLE PAGE (WRITING A COMMENT)

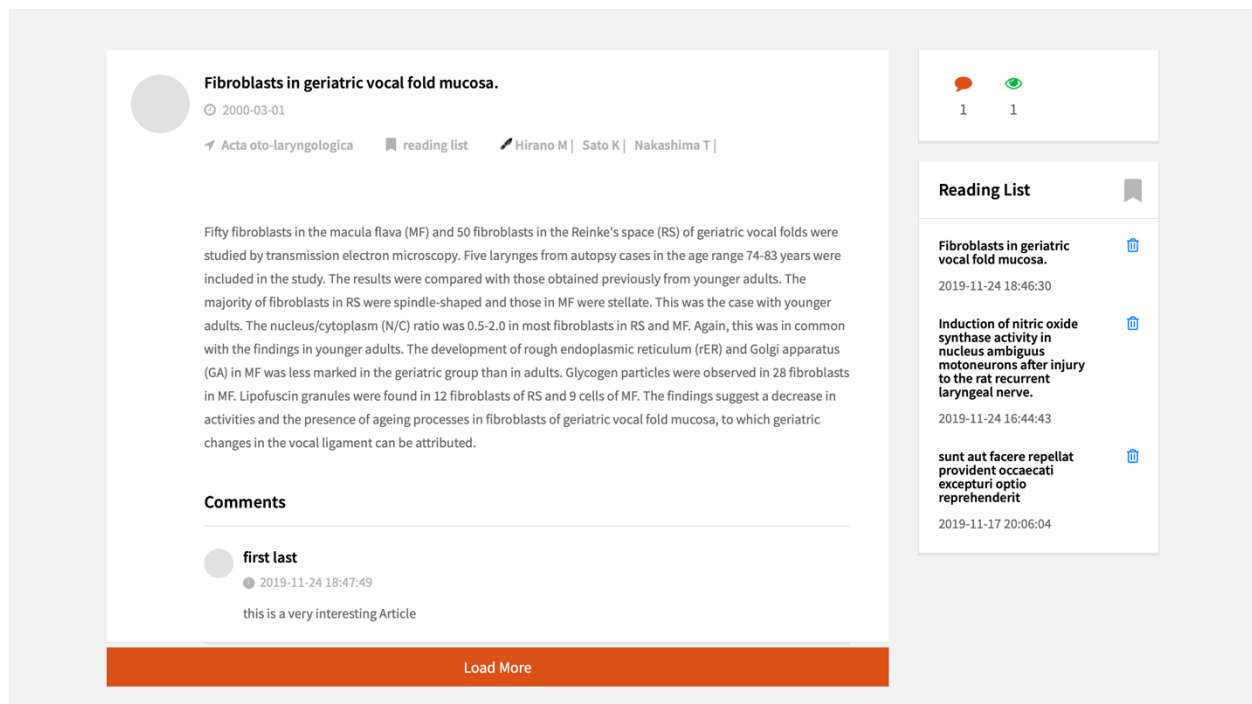


FIGURE 24: ARTICLE PAGE (POSTING THE COMMENT FROM FIGURE 23)

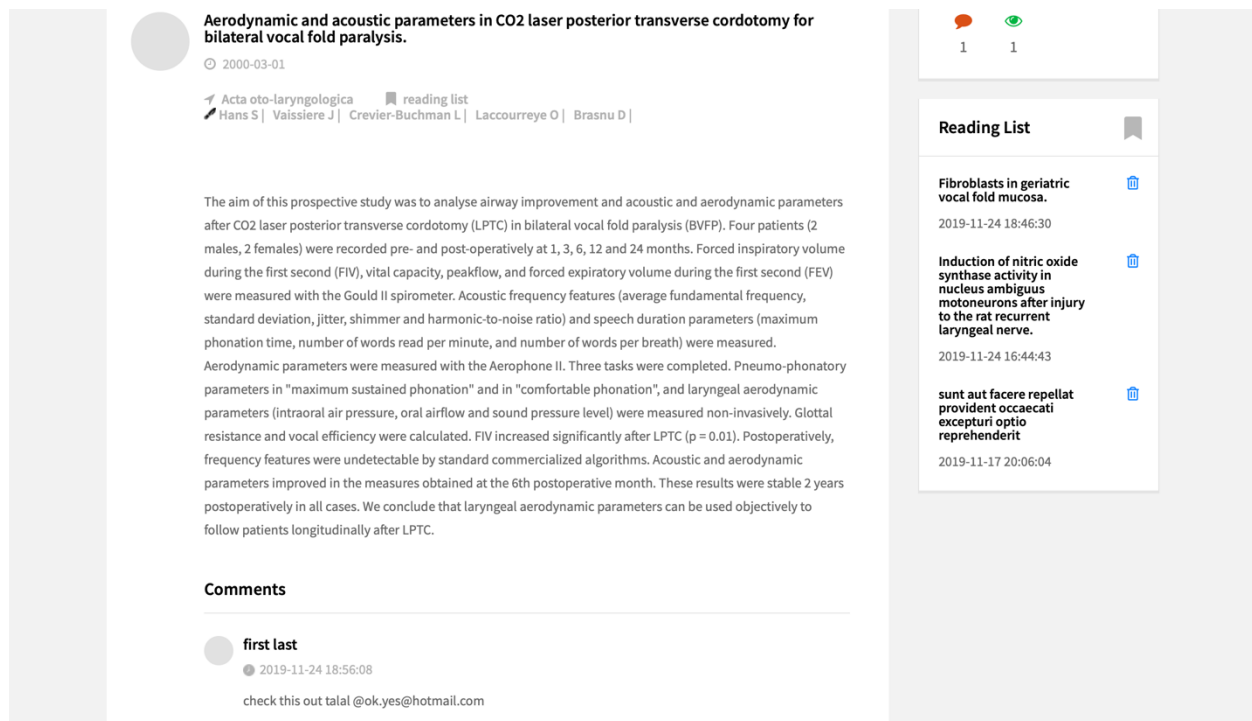


FIGURE 25: ARTICLE PAGE (MENTIONING ANOTHER USER IN A COMMENT)

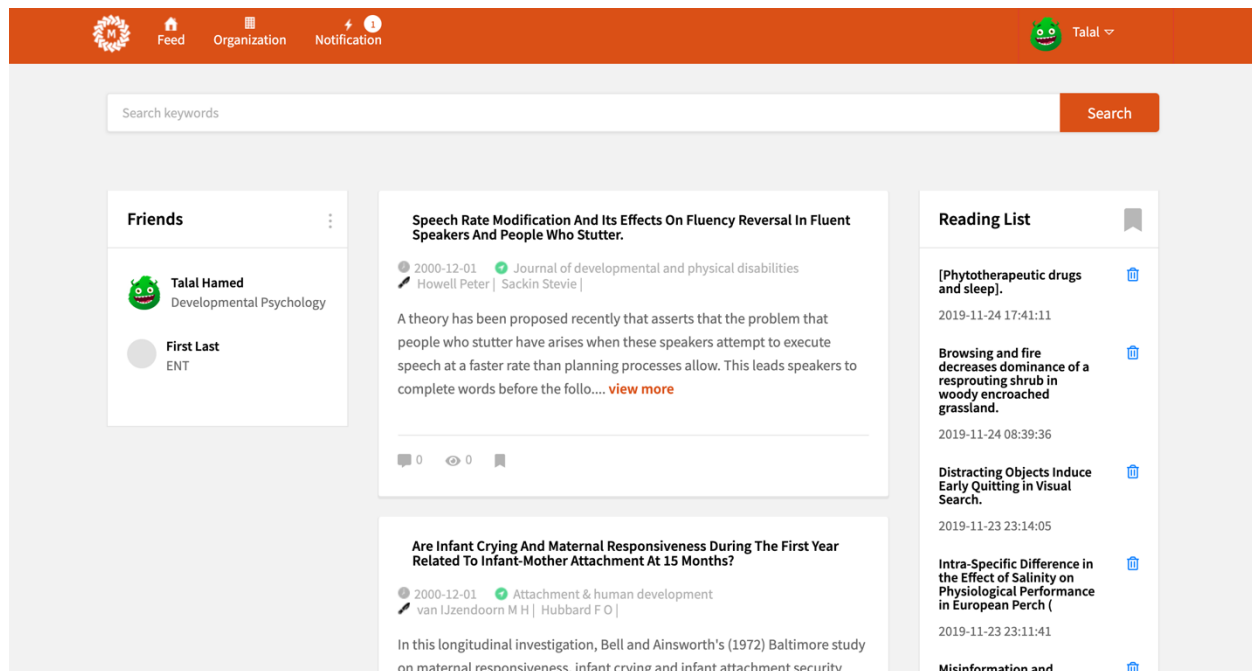


FIGURE 26: FEED PAGE (NOTIFICATION LABEL SHOWS 1 FOR THE COMMENT THAT WAS MENTIONING THIS USER FROM FIGURE 25)

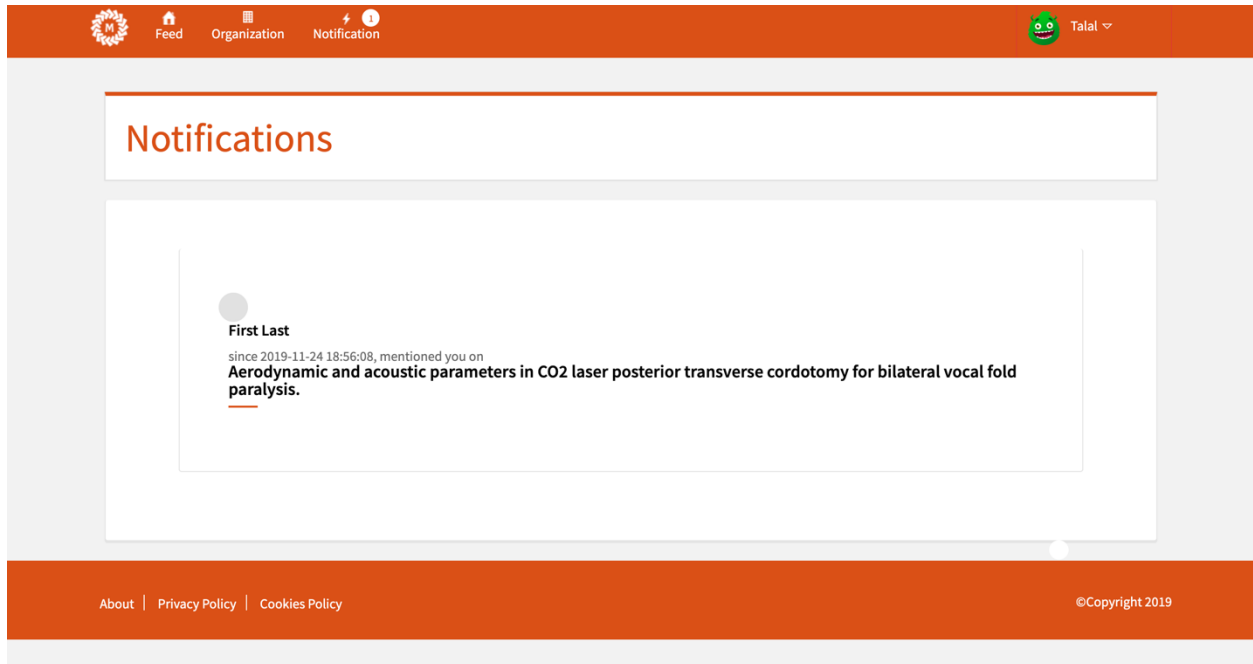


FIGURE 27: NOTIFICATION PAGE

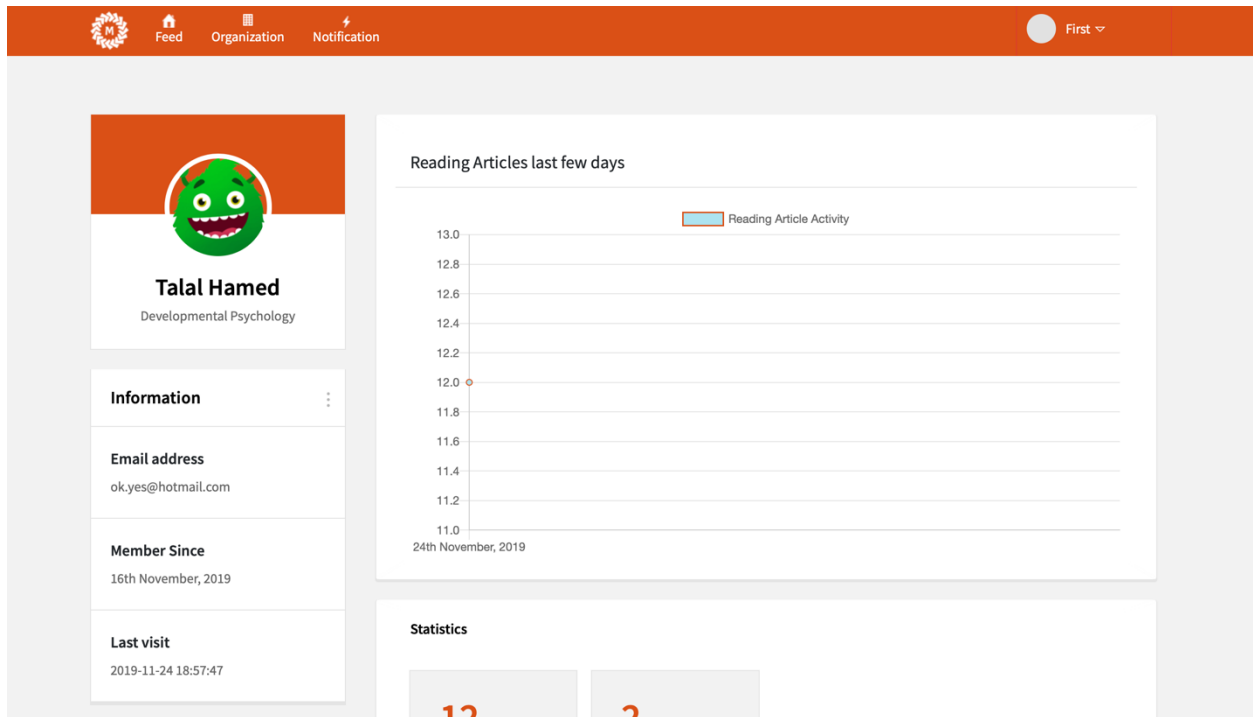


FIGURE 28: USER PROFILE PAGE

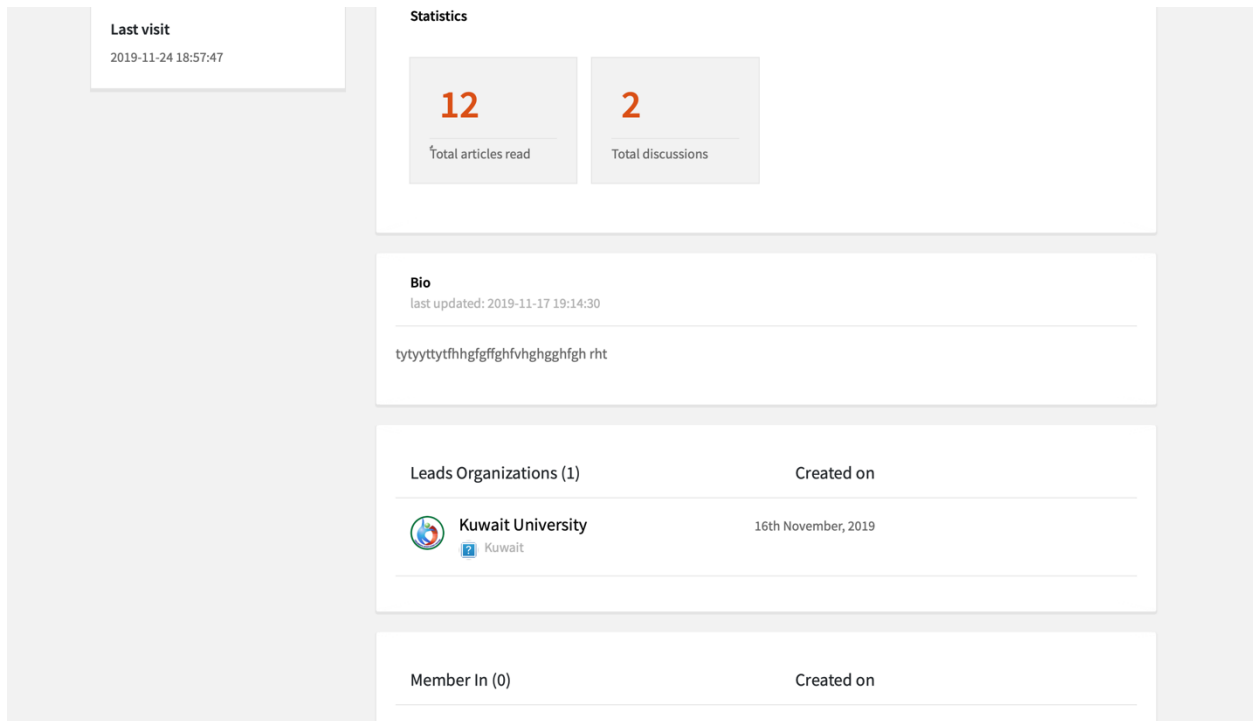


FIGURE 29: USER PROFILE PAGE (CONT)

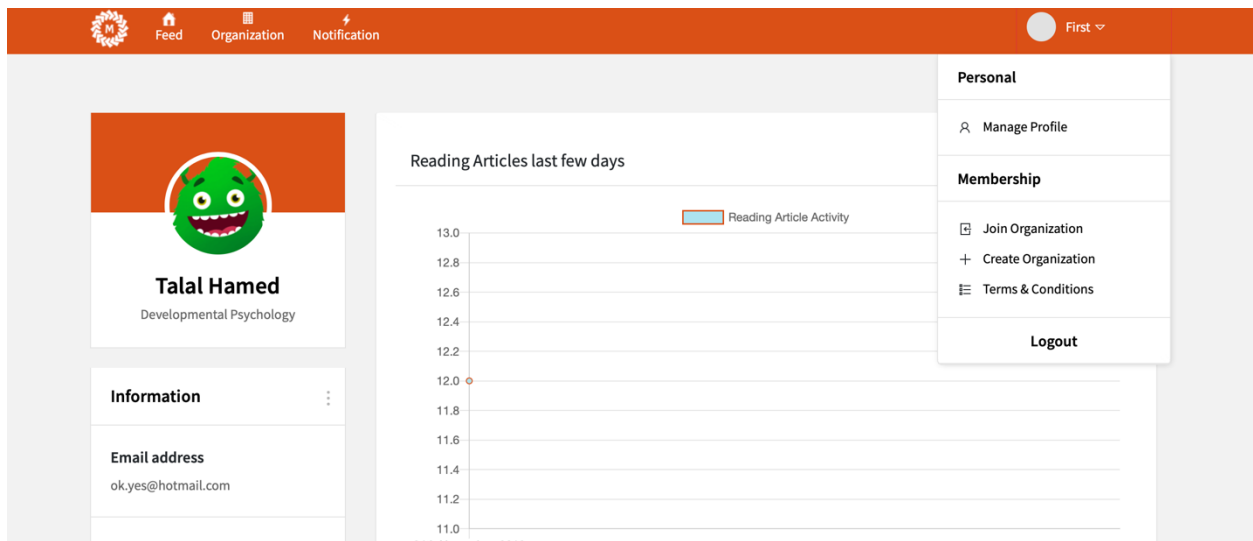





FIGURE 30: USER PROFILE PAGE (HIGHLIGHTING DROPDOWN MENU ON TOP RIGHT CORNER)




Feed



Organization



Notification



First ▾

First Last

ENT

Personal Photo

Choose File

no file selected

Upload

Specialty

Surgery

ENT

Save

FIGURE 31: USER EDIT PROFILE PAGE

Bio

Describe yourself in a couple of lines.

5000 character left

ABC

Save

Personal Information

first

last

Save

FIGURE 32: USER EDIT PROFILE PAGE (CONT)

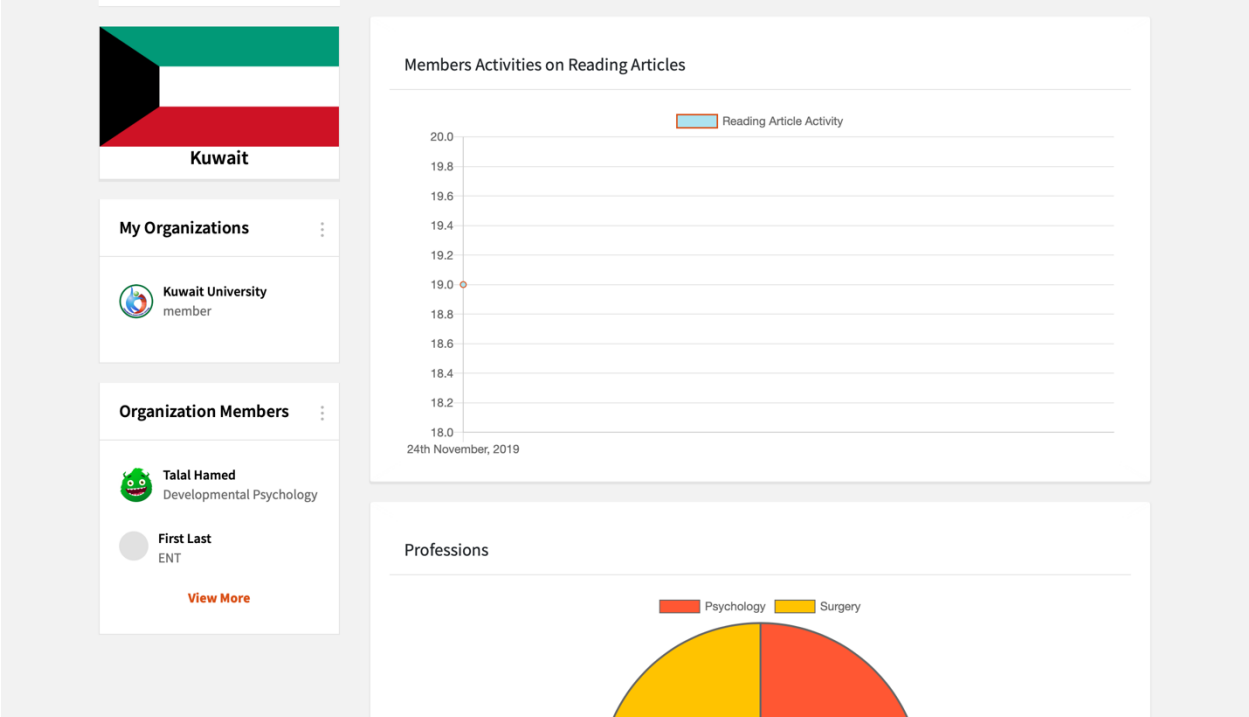


FIGURE 35: ORGANIZATION PROFILE PAGE (CONT)

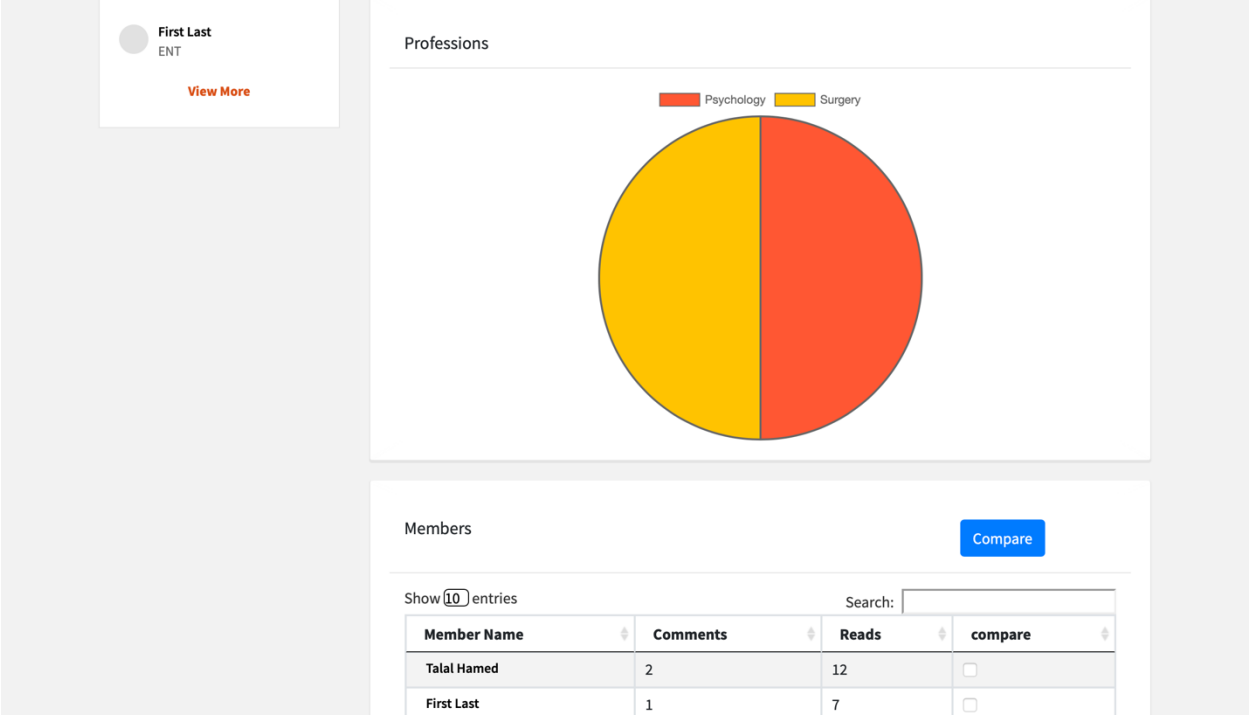


FIGURE 36: ORGANIZATION PROFILE PAGE (CONT)

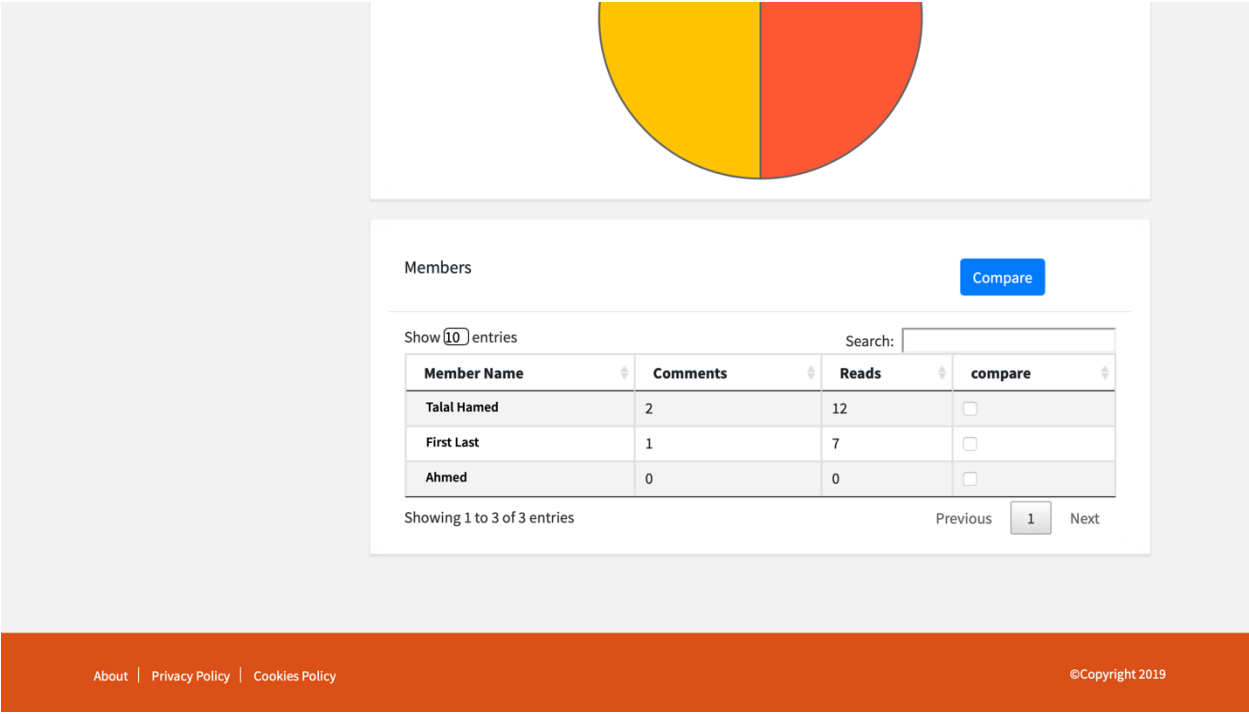


FIGURE 37: ORGANIZATION PROFILE PAGE (CONT)

4.4.PROJECT PLANNING AND MANAGEMENT

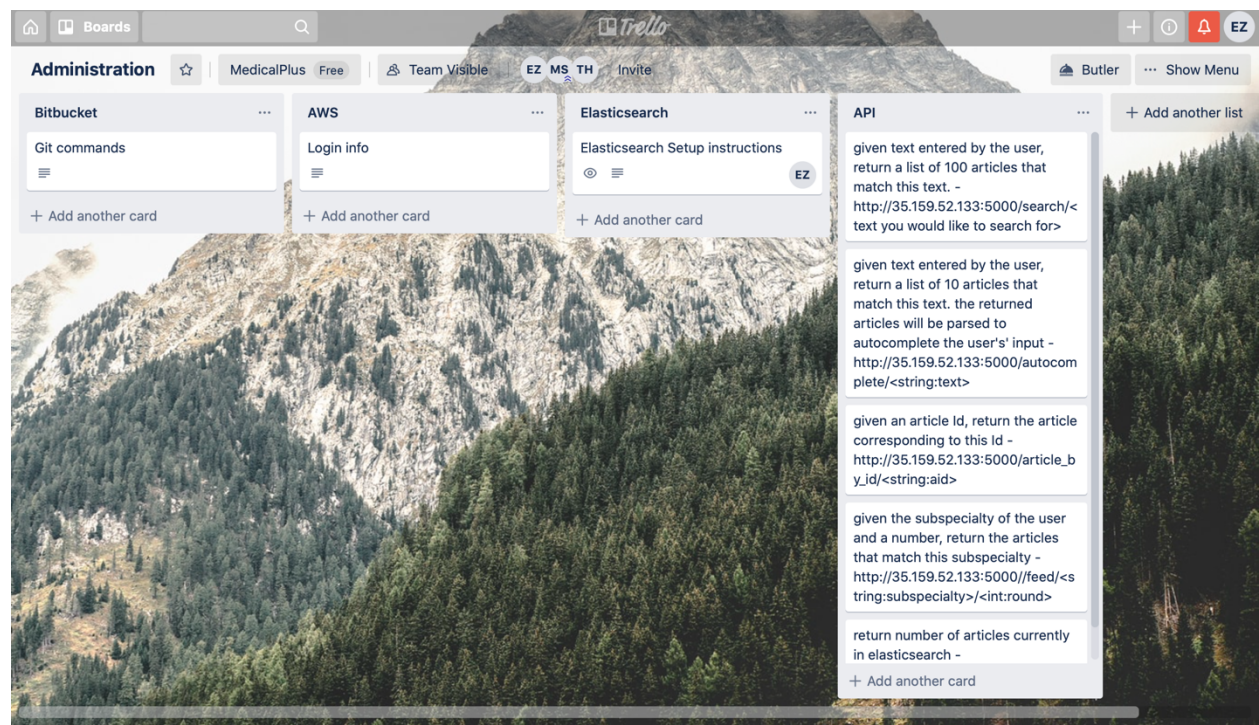


FIGURE 38: ADMINISTRATION DASHBOARD ON TRELLO

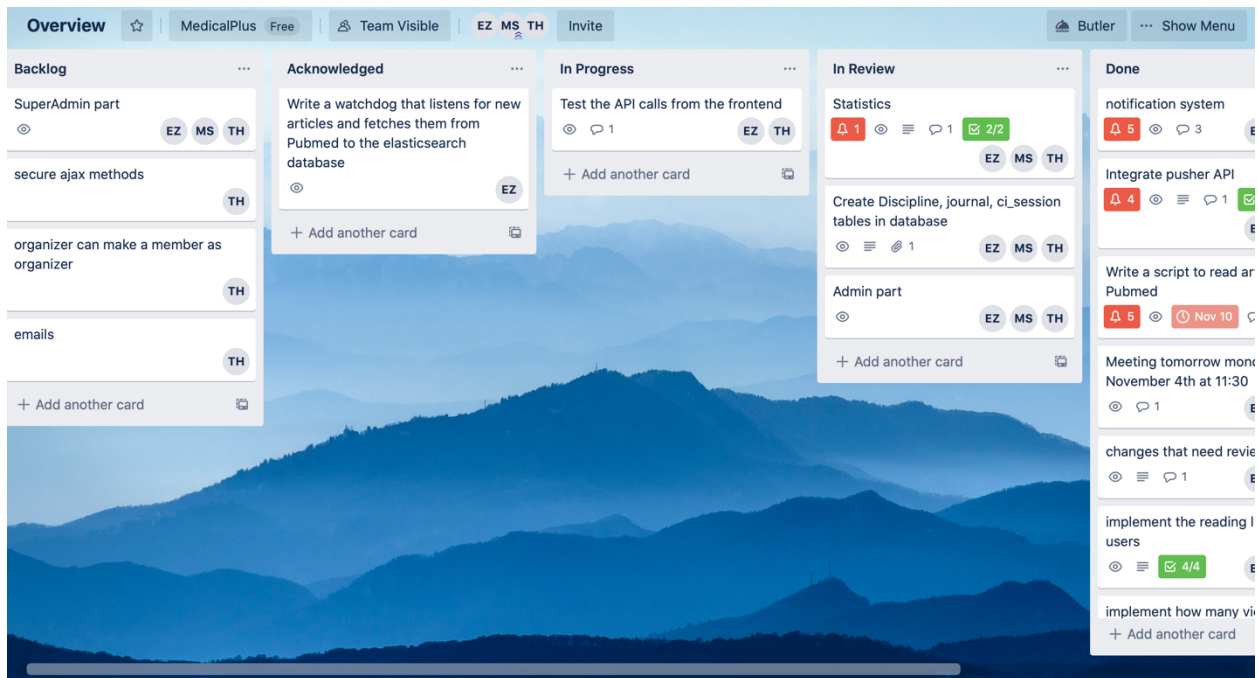


FIGURE 39: OVERVIEW/TASK DASHBOARD ON TRELLO FOR MONITERING WORK BETWEEN TEAM MEMBERS

Containing the word: <input type="text"/>									
Table	Action	Rows	Type	Collation	Size	Overhead			
<input type="checkbox"/> ci_sessions	★ Browse Structure Search Insert Empty Drop	81	InnoDB	utf8_general_ci	80 KiB	-			
<input type="checkbox"/> comment	★ Browse Structure Search Insert Empty Drop	4	InnoDB	utf8_general_ci	48 KiB	-			
<input type="checkbox"/> country	★ Browse Structure Search Insert Empty Drop	250	InnoDB	utf8_general_ci	16 KiB	-			
<input type="checkbox"/> discipline	★ Browse Structure Search Insert Empty Drop	68	InnoDB	utf8_general_ci	32 KiB	-			
<input type="checkbox"/> journal	★ Browse Structure Search Insert Empty Drop	5,652	InnoDB	utf8_general_ci	544 KiB	-			
<input type="checkbox"/> mention	★ Browse Structure Search Insert Empty Drop	1	InnoDB	utf8_general_ci	48 KiB	-			
<input type="checkbox"/> organization	★ Browse Structure Search Insert Empty Drop	2	InnoDB	utf8_general_ci	32 KiB	-			
<input type="checkbox"/> organization_user	★ Browse Structure Search Insert Empty Drop	3	InnoDB	utf8_general_ci	48 KiB	-			
<input type="checkbox"/> reading_list	★ Browse Structure Search Insert Empty Drop	9	InnoDB	utf8_general_ci	32 KiB	-			
<input type="checkbox"/> role	★ Browse Structure Search Insert Empty Drop	1	InnoDB	utf8_general_ci	16 KiB	-			
<input type="checkbox"/> settings	★ Browse Structure Search Insert Empty Drop	1	InnoDB	utf8_general_ci	16 KiB	-			
<input type="checkbox"/> specialty	★ Browse Structure Search Insert Empty Drop	18	InnoDB	utf8_general_ci	16 KiB	-			
<input type="checkbox"/> sub_specialty	★ Browse Structure Search Insert Empty Drop	50	InnoDB	utf8_general_ci	32 KiB	-			
<input type="checkbox"/> user	★ Browse Structure Search Insert Empty Drop	4	InnoDB	utf8_general_ci	48 KiB	-			
<input type="checkbox"/> view	★ Browse Structure Search Insert Empty Drop	23	InnoDB	utf8_general_ci	64 KiB	-			
15 tables	Sum	6,167	InnoDB	utf8_general_ci	1 MiB	0 B			

FIGURE 40: DATABASE TABLES ON PHPMYADMIN

Instances

[All instances](#) 3 [aws.master.r4](#) 1 [aws.data.highcpu.m5](#) 2

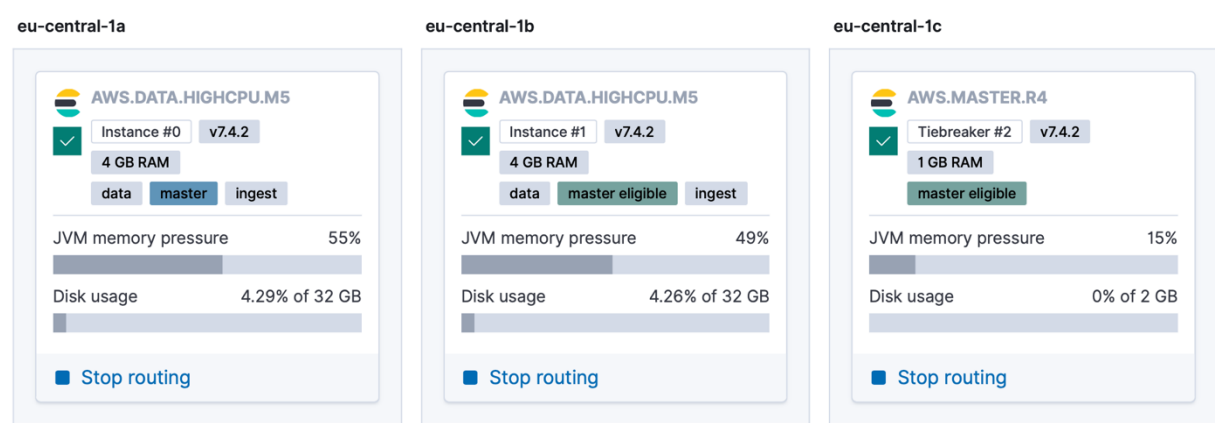


FIGURE 41: INSTANCES DEPLOYED ON ELASTIC CLOUD

[mpes](#)

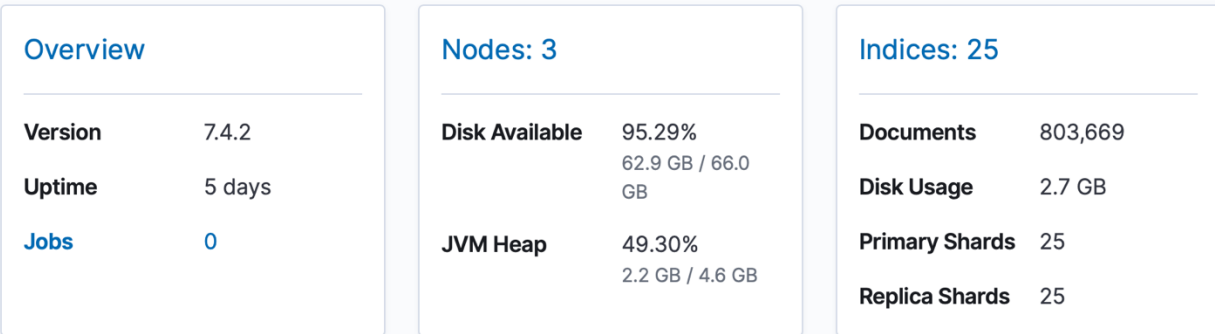


FIGURE 42: DISCRIPTIVE MESSURES FOR INSTANCES DEPLOYED ON THE ELASTIC CLOUD

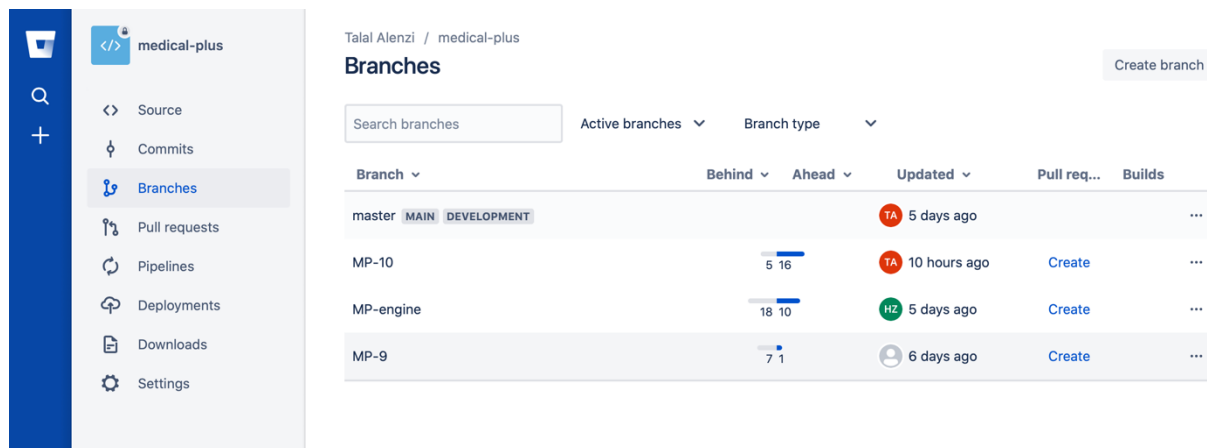


FIGURE 43: BITBUCKET REPOSITORY (HIGHLIGHTING DIFFERENT BRANCHES)

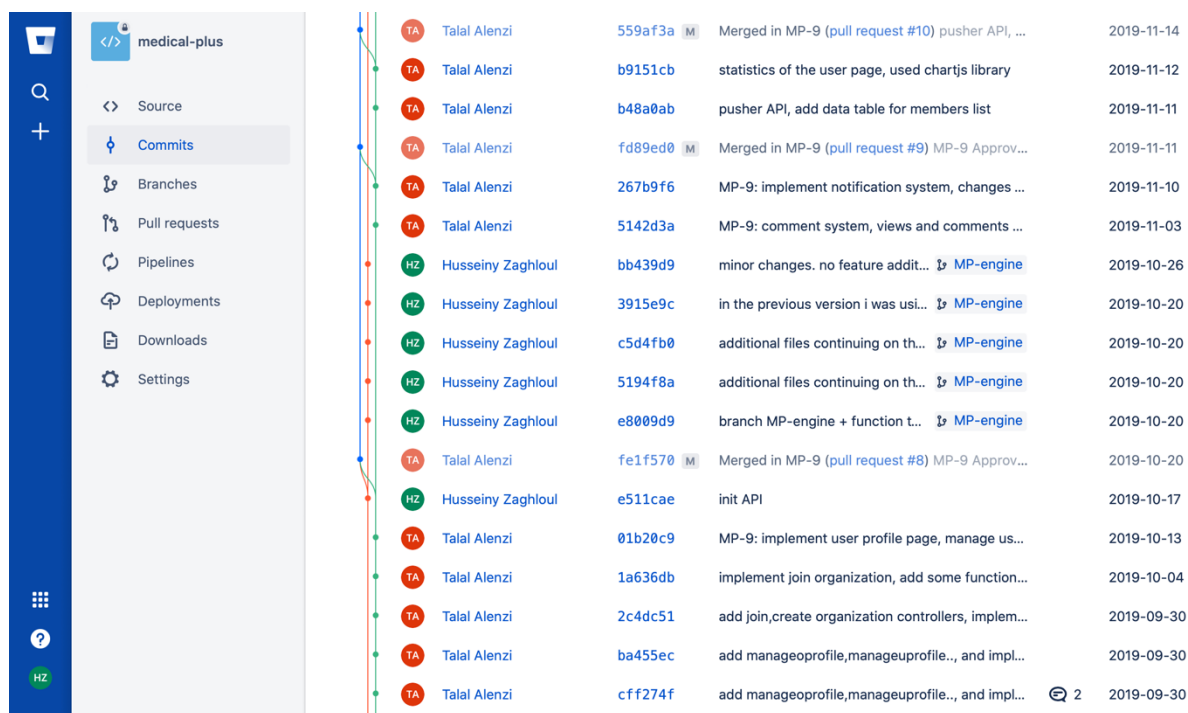


FIGURE 44: : BITBUCKET REPOSITORY (HIGHLIGHTING COMMITS FROM DIFFERENT TEAM MEMBERS)

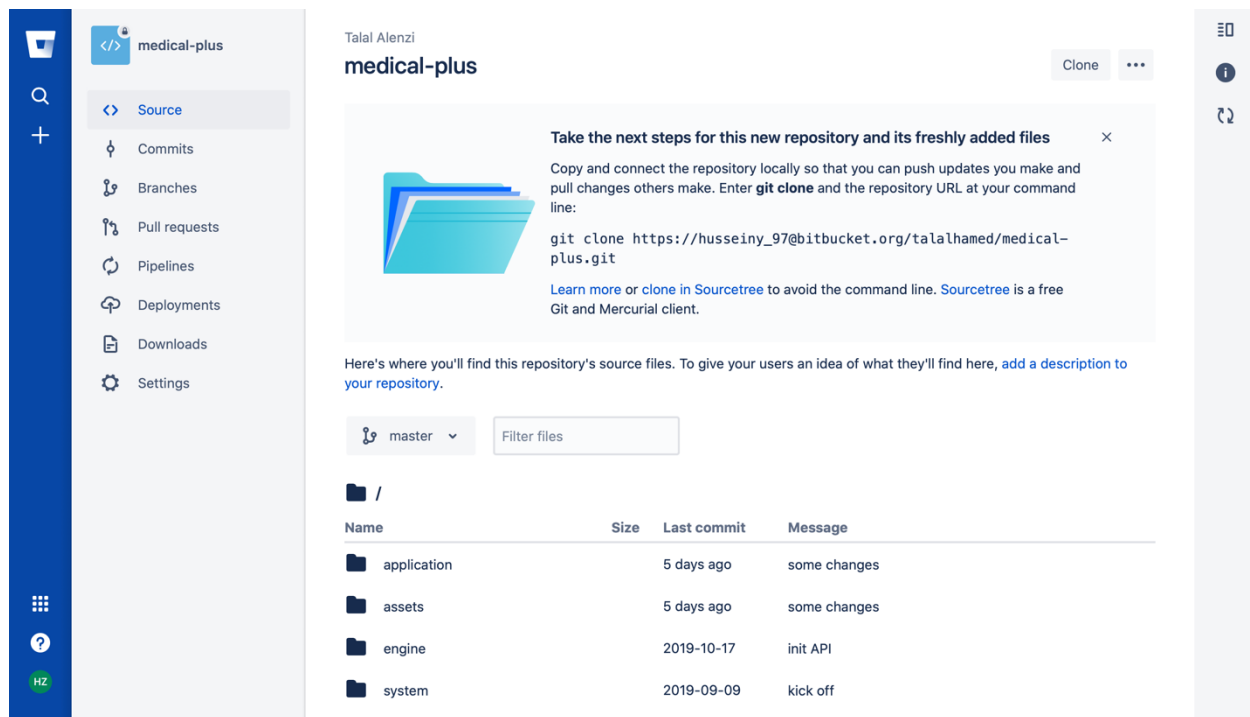


FIGURE 45: : BITBUCKET REPOSITORY

5. IMPLEMENTATION DETAILS

5.1. FRAMEWORKS AND PLATFORMS

This section describes the frameworks that we used to develop our system, along with a simple explanation of the mentioned framework and the reason for choosing this framework over other frameworks. After that we'll discuss the target platform for our system and why we chose this specific platform over other platforms.

5.1.1. USED FRAMEWORKS

- **Flask:** Flask is a relatively young framework, only in use since 2010. Flask is considered more “[Pythonic](#)” than Django. That is simply because Flask web application code is, in most cases, more explicit. Flask is the choice of most beginners due to the lack of roadblocks to getting a simple app up and running.[11] Which makes Flask perfect for our use since we are only using it to build our API which will

gather and store data to Elasticsearch and will serve user requests. Below are also some the pros that pulled us towards using flask.

- Extremely flexible
 - Minimalist without sacrificing power
 - Simple to learn and use
 - Routing URLs is easy
 - Small core and easily extensible
- **CodeIgniter:** CodeIgniter is a powerful PHP framework with a very small footprint, built for developers who need a simple and elegant toolkit to create full-featured web applications. We are currently using CodeIgniter to build our web application. And here are some of the pros that pulled us towards using this framework.
 - Built-in libraries and helpers
 - Most active online community
 - Clear & thorough documentation
 - Has a smaller footprint
 - Offers modern separation of concerns (MVC)
 - Accompanies built-in security features
 - Simple to extend and customize
 - Easy error handling
- **Bootstrap:** Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains CSS- and JavaScript-based design templates for typography, forms, buttons, navigation and other interface components

5.1.2. TARGET PLATFORM

Our system will be distributed as a web application platform. The reason behind making this decision was based on the type of users that will be getting the best out of our system. Adding to that the type of content that we are providing for our users. The target crowd of our system will be medical physicians and administrators. And based on our research, we concluded that they mostly use their personal computers for reading and doing research. Plus, it is safer to read for the readers to use bigger screens when reading for a long period of time. So, based on these conclusions we decided that a web application would be most suited for our system.

5.2. COMPONENT AND CODE REUSE

This section lists and describes all the reusable libraries that we used in our code.

- **Elasticsearch:** python library that is the official low-level client for Elasticsearch. Its goal is to provide common ground for all Elasticsearch-related code in Python; because of this it tries to be opinion-free and very extendable.
- **render_template:** Flask library that simplifies the process of rendering HTML templates. All it needs is the path to your HTML document and it will render it without a problem.
- **Flask:** Flask library that provides an extensible list of python functions that simplifies a lot of tasks that would otherwise be complex.
- **Session:** CodeIgniter library that provides a huge variety of functions that allow retrieving session data and manipulating them easily.
- **Database:** CodeIgniter library that provide various means of connecting to a databases and querying data from these databases.
- **Form_validation:** CodeIgniter library that saves a lot of effort in attempting to validate HTML forms and such.
- **CodeIgniter:** Official CodeIgniter library that provides an extensible list of PHP functions that simplifies a lot of tasks that would otherwise be complex
- Pymysql:
- Json:
- xml.etree.ElementTree:
- os:

5.3. CASE TOOLS

This section discusses the tools that we used for in developing our system. The tools vary from documentation tools, to coding tools, and even communication tools. Below is the list of tools along with their role or the benefit we got from it. After that is the licensing associated with some of the tools.

- **Microsoft Word:** mainly used for writing the report to document the progress of the project.
- **Microsoft PowerPoint:** used for creating the slides for the presentations that we held in front of the committee
- **Sublime text:** this tool is an IDE that was used for writing the code of our system.
- **VS code:** this tool is also an IDE that was used for writing code by the other member of the team.
- **Draw.io:** is a nice tool that allowed us to draw descriptive diagrams of our system in order to our system clear to the stakeholders and whatnot.
- **Trello:** this tool was used for managing the progress of the system. It had cards that each describes a task of some sort and that task was assigned to some team member. It was like a dashboard that had the tasks that were to be done and the current state of these tasks and so on. All team members had access to this dashboard.

- **Bitbucket:** was used as a repository to our code. It also made reviewing the code pretty easy. It's basically a web platform that uses git. Where one team member could push the code to the repository, and another would review it for instance. It also allowed us to move between different versions of the code in case we any changes that needed to be redacted for some reason. It made that very easy for us.
 - **Slack:** this tool was a communication tool that we used to communicate with each other.
 - **XAMPP:** is a simple, free, open-source, lightweight Apache distribution that makes it extremely easy for us to create a local web server for testing and deployment purposes
 - **Postman:** postman is the tool that we used to test our system's API.
 - **JMeter:** is the tool that we used to stress test our system.
-
- Sublime text has a Proprietary license.
 - The Microsoft products we are using have both Full Packaged Product (FPP) licenses and OEM licenses.
 - XAMPP has a GNU general public license.
 - Flask has a BSD license.
 - CodeIgniter has an MIT license.

5.4.MODELS AND ALGORITHMS

This section describes some of the main models that we are using to develop our system. Some of these models were adopted by our method of programming and some are just being used by the framework or service that we are using. Either way these models are mentioned here.

- **MVC:** CodeIgniter is based on the Model-View-Controller development pattern. MVC is a software approach that separates application logic from presentation. In practice, it permits your web pages to contain minimal scripting since the presentation is separate from the PHP scripting.
 - The **Model** represents your data structures. Typically, your model classes will contain functions that help you retrieve, insert, and update information in your database.
 - The **View** is the information that is being presented to a user. A View will normally be a web page, but in CodeIgniter, a view can also be a page fragment like a header or footer. It can also be an RSS page, or any other type of "page".
 - The **Controller** serves as an *intermediary* between the Model, the View, and any other resources needed to process the HTTP request and generate a web page.
- **NoSQL:** When compared to relational databases, NoSQL databases are more scalable and provide superior performance, and their data model addresses several issues that the relational model is not designed to address:

- Large volumes of rapidly changing structured, semi-structured, and unstructured data
 - Agile sprints, quick schema iteration, and frequent code pushes
 - Object-oriented programming that is easy to use and flexible
 - Geographically distributed scale-out architecture instead of expensive, monolithic architecture.
- **OOP** (Object-Oriented Programming): Object-oriented programming (OOP) is a programming language model in which programs are organized around data, or objects, rather than functions and logic. An object can be defined as a data field that has unique attributes and behavior.

5.5. DATA INTEGRATION APPROACH

The data integration approach we implemented into our system is the ETL (Extract, Transform, Load) approach. This approach is widely known when extracting data from one database and storing it into another. Keep in mind that this entire process is fully automated.

The Extraction phase consists of extracting the data from the PubMed database and filtering the data based on our systems requirements. To preserve the trustiness and worthiness of the articles that we promised the users, we didn't just extract every article from PubMed and provide it to the users, no, what we do is we filter each and every article based on certain rules that would guarantee the article's worth and relevance it adds to our system. To do that, we collected a group of over 5000 of the most trusted and highly ranked journals on JCR. After that we mapped each of those journals with a discipline from the medical field. And we mapped the disciplines with a specialty and those specialties were mapped to sub-specialties in the medical field. So, we start with the most general and narrowed it down to the bare minimum. After that we started fetching the Id's of the articles 100000 at a time for each year. We took those 100000 Id's and started requesting the actual data corresponding to these Id's 400 at a time due to limitations and guidelines demanded by the NCBI API for security reasons. After fetching the 400 articles we started filtering each article one by one. The way we filtered the data was first to determine if the article contains all the information we needed to guarantee its authenticity (e.g. journal, author, title, publish date...). after that we checked if the article's journal corresponds to any of the journals we gathered from PubMed. If the journal corresponds, we map it with a discipline, and from discipline to specialty, and from specialty to subspecialty, and finally assigned this article the corresponding subspecialty. If the journal does not correspond to our list of journals, we discard the article and move on to the next.

The transformation phase is where we transform the articles into the form that we require to store them into our database. After extracting the articles and filtering them to fit our standards we then started to extract the fields one by one to only extract what we need from each article. The articles were returned from the PubMed database in XML form and we needed to store them onto Elasticsearch in JSON form. So, we parsed the XML of each article and collected the data we need from it to insert it into the JSON form we demand.

The Load phase consists of loading the data into our own database. So, after parsing the entire list of 100000 articles and parsing them, we stored them into an array. Once the batch of 100000 articles is done parsing and transforming, we started to insert them into our Elasticsearch database one by one.

By going through this thorough process, we guarantee the users the accuracy and authenticity of the articles we promised.

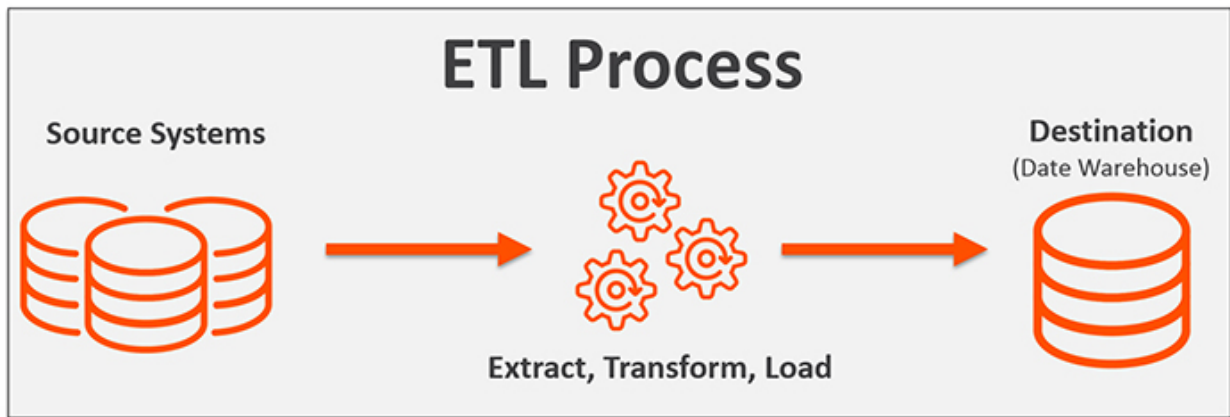


FIGURE 46: ETL DATA INTEGRATION PROCESS

5.5.1. DATA INTEGRATION PSUEDO CODE

```
Loop over years starting from 2019 going down:
    Fetch number of articles in "current year"
    loop over list of article Ids in current year:
        fetch 100000 at a time
        call data Extractor Function given (100000 Ids)

data Extractor (Id List):
    Loop over List of Ids:
        Fetch articles corresponding to 400 Ids at a time
        Loop over fetched articles one by one:
            Check if contains all data:
                Map article journal with JCR list of journals
                If matches with one of them:
                    Map article journal with corresponding discipline
                    Map discipline with Specialty
                    Map Specialty with subspecialty
                    Assign subspecialty to the current article
                Else:
                    Discard article
            Extract required data from article
            Transform data into Json form
            Add article in json form to processed list
        Else:
            Discard current article
    Loop over list of processed articles:
        Insert article into Elasticsearch.
```

6. MODIFICATIONS OF THE ORIGINAL PLAN (OPTIONAL)

- Changing the architecture: in the beginning the architecture didn't support the scalability of the system. But now, thanks to AWS our system is scalable up and down based on our needs and traffic. With our current configurations on AWS our number of instances on AWS will increase and decrease automatically based on the current CPU usage caused by traffic on our system. Once the number of instances increases or decreases, the load balancer will automatically balance the requests among the running instances.
- Another main change was changing the application from B2C to B2B. the reason we did this change in the system design was to motivate the users to read more.

7. TESTING

7.1. Testing Plan

Our plan was to first do unit testing to make sure that each small unit works correctly. After that we did integration testing to see if the components work together correctly. We used selenium to perform integration testing. Adding to that, we as developers tested each component alone and the whole system as a whole. We tried to fail the system by giving invalid input. And finally, we performed Deployment testing.

7.2. Deployment Testing

One of the main problems was that MVC was programmed on a windows environment which was not case sensitive when dealing with controller names. On the other hand, AWS runs a linux environment which is in fact case sensitive. So, we had to go back and change all the controller file names to deal with this problem.

7.3. Unit Test Cases

7.3.1. Comment Model Test

Test Name	get_comment_count test function
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_comment_model.php
Line Number	24
Notes	expected 18 comments for article id = 1

TABLE 1

```
18 public function test_get_comment_count()
19 {
20     $article_id = 1;
21     $expected_value = 18;
22     $test = $this->comment_model->get_comment_count($article_id);
23     $test_name = 'get_comment_count test function';
24     echo $this->unit->run($test,$expected_value, $test_name, 'expected 18 comments for article id = 1');
25 }
```

FIGURE 47

Test Name	get_comment_count_by_user test function
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	C:\xampp\htdocs\medical-

	plus\application\controllers\test_models\Test_comment_model.php
Line Number	33
Notes	expected 15 comments for user_id = 85

TABLE 2

```

27 public function test_get_comment_count_by_user()
28 {
29     $user_id = 85;
30     $expected_value = 15;
31     $test = $this->comment_model->get_comment_count_by_user($user_id);
32     $test_name = 'get_comment_count_by_user test function';
33     echo $this->unit->run($test,$expected_value, $test_name, 'expected 15 comments for user_id = 85');
34 }

```

FIGURE 48

Test Name	get_comments test function
Test Datatype	Array
Expected Datatype	Array
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_comment_model.php
Line Number	42
Notes	expected an array

TABLE 3TABLE 4

```

36 public function test_get_comments()
37 {
38     $article_id = 1;
39     $expected_value = 'is_array';
40     $test = $this->comment_model->get_comments($article_id);
41     $test_name = 'get_comments test function';
42     echo $this->unit->run($test,$expected_value, $test_name, 'expected an array');
43 }

```

FIGURE 49

Test Name	get_comments_more test function
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_comment_model.php
Line Number	51
Notes	expected 2 items

TABLE 5

```

45     public function test_get_comments_more()
46     {
47         $article_id = 1;
48         $expected_value = 2;
49         $test = $this->comment_model->get_comments_more($article_id,2,1);
50         $test_name = 'get_comments_more test function';
51         echo $this->unit->run(count($test),$expected_value, $test_name, 'expected 2 items');
52     }

```

FIGURE 50

Test Name	addComment test function
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_comment_model.php
Line Number	63
Notes	expected count_after = count_before

TABLE 6

```

54     public function test_addComment()
55     {
56         $user_id = 85;
57         $article_id = 1;
58         $comment_text = 'text';
59         $comments_count_before = $this->comment_model->get_comment_count_by_user($user_id);
60         $test = $this->comment_model->addComment($user_id,$article_id,$comment_text);
61         $comments_count_after = $this->comment_model->get_comment_count_by_user($user_id);
62         $test_name = 'addComment test function';
63         echo $this->unit->run($comments_count_after,$comments_count_before+1, $test_name, 'expected count_after = count_before');
64     }
65

```

FIGURE 51

7.3.2. Country Model Test

Test Name	get_countries test function
Test Datatype	Array
Expected Datatype	Array
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_country_model.php
Line Number	22
Notes	expected an array of countries

TABLE 7


```

18     public function test_get_countries()
19     {
20         $test = $this->country_model->get_countries();
21         $test_name = 'get_countries test function';
22         echo $this->unit->run($test,'is_array', $test_name, 'expected an array of countries');
23     }

```

FIGURE 52

7.3.3. Enrollment Model Test

Test Name	insert_new_enrollment test function
Test Datatype	Boolean
Expected Datatype	Boolean
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_enrollment_model.php
Line Number	29
Notes	expected TRUE

TABLE 8

```

18     public function test_insert_new_enrollment()
19     {
20         $enrollment_data = [
21             'organization_id' => 78,
22             'user_id' => 111,
23             'level' => 'member',
24             'enrollment_date' => $date=date('Y-m-d H:i:s')
25         ];
26         $enrollmentID = $this->enrollment_model->insert_new_enrollment($enrollment_data);
27         $test_name = 'insert_new_enrollment test function';
28         $expected_result = ($enrollmentID > 0);
29         echo $this->unit->run(TRUE,$expected_result, $test_name, 'expected TRUE ');
30     }

```

FIGURE 53

Test Name	getOrganizersCountOfOrganization test function
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_enrollment_model.php
Line Number	38
Notes	expected 1 organizer

TABLE 9

```

33     public function test_getOrganizersCountOfOrganization()
34     {
35         $org_id = '78';
36         $test = $this->enrollment_model->getOrganizersCountOfOrganization($org_id);
37         $test_name = 'getOrganizersCountOfOrganization test function';
38         echo $this->unit->run($test,1, $test_name, 'expected 1 organizer');
39     }

```

FIGURE 54

Test Name	getMembersCountOfOrganization test function
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_enrollment_model.php
Line Number	46
Notes	expected 2 members

TABLE 10

```

42     public function test_getMembersCountOfOrganization(){
43         $org_id = '78';
44         $test = $this->enrollment_model->getMembersCountOfOrganization($org_id);
45         $test_name = 'getMembersCountOfOrganization test function';
46         echo $this->unit->run($test,2, $test_name, 'expected 2 members');
47     }

```

FIGURE 55

7.3.4. Feed Model Test

Test Name	getArticles test function
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_feed_model.php
Line Number	26
Notes	expected 10 articles

TABLE 11

Test Name	getArticles test function
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	C:\xampp\htdocs\medical-

	plus\application\controllers\test_models\Test_feed_model.php
Line Number	30
Notes	expected empty array because no data for specialty xxxx

TABLE 12

```

20 public function test_getArticles()
21 {
22     $user_specialty_name = 'Cardiology';
23     $test = $this->feed_model->getArticles($user_specialty_name,0);
24     $test_name = 'getArticles test function';
25     $expected_result = 10;
26     echo $this->unit->run(count($test),$expected_result, $test_name, 'expected 10 articles');
27     $user_specialty_name = 'xxxx';
28     $test = $this->feed_model->getArticles($user_specialty_name,0);
29     $expected_result = 0;
30     echo $this->unit->run(count($test),$expected_result, $test_name, 'expected empty array because no data for specialty xxxx')
31 }

```

FIGURE 56

Test Name	getArticlesById test function
Test Datatype	Array
Expected Datatype	Array
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_feed_model.php
Line Number	39
Notes	expected an array

TABLE 13

Test Name	getArticlesById test function
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_feed_model.php
Line Number	40
Notes	expected 1 article

TABLE 14

```

34 public function test_getArticlesById()
35 {
36     $article_id = 'i2mpl24BExqqpYTsorVu';
37     $test = $this->feed_model->getArticlesById($article_id);
38     $test_name = 'getArticlesById test function';
39     echo $this->unit->run($test,'is_array', $test_name, 'expected an array');
40     echo $this->unit->run(count($test),1, $test_name, 'expected 1 article');
41 }

```

FIGURE 57

7.3.5. Mention Model Test

Test Name	add_mention test function
Test Datatype	String
Expected Datatype	String
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_mention_model.php
Line Number	28
Notes	expected array of items

TABLE 15

```
18     public function test_add_mention()
19     {
20         $user_id = 85;
21         $comment_id = 389; // exists in database
22         $article_id = 1;
23         $sender_id = 86;
24         $article_title = 'some title';
25         $return_id = $this->mention_model->add_mention($user_id,$comment_id,$article_id,$sender_id,$article_title);
26         $expected_result = $this->mention_model->get_mention_by_id($return_id)[0]->article_title;
27         $test_name = 'add_mention test function';
28         echo $this->unit->run($article_title,$expected_result, $test_name, 'expected array of items');
29     }
```

FIGURE 58

Test Name	get_mentions_ids test function
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_mention_model.php
Line Number	38
Notes	expected 2 mentions

TABLE 16

```
32     public function test_get_mentions_ids()
33     {
34         $user_id = 85;
35         $test = $this->mention_model->get_mentions_ids($user_id);
36         $expected_result = 2;
37         $test_name = 'get_mentions_ids test function';
38         echo $this->unit->run(count($test),$expected_result, $test_name, 'expected 2 mentions');
39     }
```

FIGURE 59

Test Name	get_mentions test function
Test Datatype	Integer

Expected Datatype	Integer
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_mention_model.php
Line Number	48
Notes	expected 2 items of mention

TABLE 17

```

42     public function test_get_mentions()
43     {
44         $user_id = 85;
45         $mentions_ids = $this->mention_model->get_mentions_ids($user_id);
46         $mentions_data = $this->mention_model->get_mentions($mentions_ids,$user_id);
47         $test_name = 'get_mentions test function';
48         echo $this->unit->run(count($mentions_data), 2, $test_name, 'expected 2 items of mention');
49     }

```

FIGURE 60

Test Name	get_mention_by_id test function
Test Datatype	String
Expected Datatype	Integer
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_mention_model.php
Line Number	69
Notes	expected mentionID that retrieved from DB = 230

TABLE 18

```

64     public function test_get_mention_by_id()
65     {
66         $mentionID_in_DB = 230;
67         $test = $this->mention_model->get_mention_by_id($mentionID_in_DB);
68         $test_name = 'get_mention_by_id test function';
69         echo $this->unit->run($test[0]->mention_id, $mentionID_in_DB, $test_name, 'expected mentionID that retrieved from DB = 230');
70     }

```

FIGURE 61

Test Name	set_mention_as_seen test function
Test Datatype	Boolean
Expected Datatype	Boolean
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_mention_model.php
Line Number	76
Notes	expected true

TABLE 19

```

72     public function test_set_mention_as_seen(){
73         $mentionID_in_DB = 230;
74         $test = $this->mention_model->set_mention_as_seen($mentionID_in_DB);
75         $test_name = 'set_mention_as_seen test function';
76         echo $this->unit->run($test, true, $test_name, 'expected true');
77     }

```

FIGURE 62

Test Name	set_all_mentions_as_seen test function
Test Datatype	Boolean
Expected Datatype	Boolean
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_mention_model.php
Line Number	83
Notes	expected true

TABLE 20

```

79     public function test_set_all_mentions_as_seen(){
80         $userID = 85;
81         $test = $this->mention_model->set_all_mentions_as_seen($userID);
82         $test_name = 'set_all_mentions_as_seen test function';
83         echo $this->unit->run($test, true, $test_name, 'expected true');
84     }

```

FIGURE 63

Test Name	get_count test function
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_mention_model.php
Line Number	90
Notes	expected 2 items

TABLE 21

```

86     public function test_get_count(){
87         $userID = 85;
88         $test = $this->mention_model->get_count($userID);
89         $test_name = 'get_count test function';
90         echo $this->unit->run($test, 2, $test_name, 'expected 2 items');
91     }

```

FIGURE 64

Test Name	get_next test function
Test Datatype	String
Expected Datatype	Integer
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_mention_model.php
Line Number	97
Notes	expected mention id = 230

TABLE 22

```

93     public function test_get_next(){
94         $userID = 85;
95         $test = $this->mention_model->get_next(1,1,$userID);
96         $test_name = 'get_next test function';
97         echo $this->unit->run($test[0]->mention_id, 230, $test_name, 'expected mention id = 230');
98     }

```

FIGURE 65

Test Name	clear_all_mentions test function
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_mention_model.php
Line Number	57
Notes	expected 2 will be deleted

TABLE 23

```

52     public function test_clear_all_mentions()
53     {
54         $user_id = 85;
55         $test = $this->mention_model->clear_all_mentions($user_id);
56         $test_name = 'clear_all_mentions test function';
57         echo $this->unit->run($test, 2, $test_name, 'expected 2 will be deleted');
58     }

```

FIGURE 66

7.3.6. Organization Model Test

Test Name	insert_new_organization test function
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_organization_model.php
Line Number	29
Notes	after insert a user, the expected return array of organizations (1 item)

TABLE 24

```
18     public function test_insert_new_organization()
19     {
20         $data = [
21             'organization_name' => 'General Clinic',
22             'organization_profile_url' => md5('General Clinic'),
23             'country_id' => 747
24         ];
25         $this->organization_model->insert_new_organization($data);
26         $array_of_org_data = $this->organization_model->getByProfileURL(md5('General Clinic'));
27         $expected_result = 1;
28         $test_name = 'insert_new_organization test function';
29         echo $this->unit->run(count($array_of_org_data),$expected_result, $test_name, 'expected array of organizations (1 item)');
30     }
```

FIGURE 67

Test Name	updateDescription test function
Test Datatype	Boolean
Expected Datatype	Boolean
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_organization_model.php
Line Number	39
Notes	expected true

TABLE 25

```
33     public function test_updateDescription()
34     {
35         $desc = "some words";
36         $org_id = 84;
37         $test = $this->organization_model->updateDescription($org_id,$desc);
38         $test_name = 'updateDescription test function';
39         echo $this->unit->run($test, true, $test_name, 'expected true');
40     }
```

FIGURE 68

Test Name	setCover test function
Test Datatype	Boolean

Expected Datatype	Boolean
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_organization_model.php
Line Number	49
Notes	expected true

TABLE 26

```

43     public function test_setCover()
44     {
45         $coverURL = "cover url";
46         $org_id = 85;
47         $test = $this->organization_model->setCover($org_id,$coverURL);
48         $test_name = 'setCover test function';
49         echo $this->unit->run($test, true, $test_name, 'expected true');
50     }

```

FIGURE 69

Test Name	setLogo test function
Test Datatype	Boolean
Expected Datatype	Boolean
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_organization_model.php
Line Number	59
Notes	expected true

TABLE 27

```

53     public function test_setLogo()
54     {
55         $LogoURL = "logo url";
56         $org_id = 85;
57         $test = $this->organization_model->setCover($org_id,$LogoURL);
58         $test_name = 'setLogo test function';
59         echo $this->unit->run($test, true, $test_name, 'expected true');
60     }

```

FIGURE 70

Test Name	getAllByUserEmail test function
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_organization_model.php
Line Number	69
Notes	expected 1, so this user has joined only one organization

TABLE 28

```

63 public function test_getAllByUserEmail()
64 {
65     $userEmail = 'ok.yes@live.org';
66     $test = $this->organization_model->getAllByUserEmail($userEmail);
67     $test_name = 'getAllByUserEmail test function';
68     $expected_value = 1;
69     echo $this->unit->run(count($test), $expected_value, $test_name, 'expected 1, so this user has joined only one organization');
70 }

```

FIGURE 71

Test Name	getAllIdsByUserEmail test function
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_organization_model.php
Line Number	79
Notes	expected 1, joined only one organization

TABLE 29

```

73 public function test_getAllIdsByUserEmail()
74 {
75     $userEmail = 'ok.yes@hotmail.com';
76     $test = $this->organization_model->getAllIdsByUserEmail($userEmail);
77     $test_name = 'getAllIdsByUserEmail test function';
78     $expected_value = 1;
79     echo $this->unit->run(count($test), $expected_value, $test_name, 'expected 1, joined only one organization');
80 }

```

FIGURE 72

Test Name	getOnlyManagableByUserEmail test function
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_organization_model.php
Line Number	89
Notes	expected 1, create one organization

TABLE 30

Test Name	getOnlyManagableByUserEmail test function
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_organization_model.php
Line Number	93

Notes	expected 0, not an organizer
-------	------------------------------

TABLE 31

```

83 public function test_getOnlyManagableByEmail()
84 {
85     $userEmail = 'ok.yes@hotmail.com';
86     $test = $this->organization_model->getOnlyManagableByEmail($userEmail);
87     $test_name = 'getOnlyManagableByEmail test function';
88     $expected_value = 1;
89     echo $this->unit->run(count($test), $expected_value, $test_name, 'expected 1, create one organization');
90     $userEmail = 'ok.yes@live.org';
91     $test = $this->organization_model->getOnlyManagableByEmail($userEmail);
92     $expected_value = 0;
93     echo $this->unit->run(count($test), $expected_value, $test_name, 'expected 0, not an organizer');
94 }

```

FIGURE 73

Test Name	getOnlyUserMemberInByUserEmail test function
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_organization_model.php
Line Number	103
Notes	expected 0, not a memeber in any organization

TABLE 32

Test Name	getOnlyUserMemberInByUserEmail test function
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_organization_model.php
Line Number	107
Notes	expected 1, only one organization assign with it

TABLE 33

```

97 public function test_getOnlyUserMemberInByUserEmail()
98 {
99     $userEmail = 'ok.yes@hotmail.com';
100     $test = $this->organization_model->getOnlyUserMemberInByUserEmail($userEmail);
101     $test_name = 'getOnlyUserMemberInByUserEmail test function';
102     $expected_value = 0;
103     echo $this->unit->run(count($test), $expected_value, $test_name, 'expected 0, not a memeber in any organization');
104     $userEmail = 'ok.yes@live.org';
105     $test = $this->organization_model->getOnlyUserMemberInByUserEmail($userEmail);
106     $expected_value = 1;
107     echo $this->unit->run(count($test), $expected_value, $test_name, 'expected 1, only one organization assign with it');
108 }

```

FIGURE 74

Test Name	getByProfileURL test function
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	C:\xampp\htdocs\medical-

	plus\application\controllers\test_models\Test_organization_model.php
Line Number	117
Notes	expected 1 organization

TABLE 34

```

111     public function test_getByProfileURL()
112     {
113         $org_profile = md5('General Clinic');
114         $test = $this->organization_model->getByProfileURL($org_profile);
115         $test_name = 'getByProfileURL test function';
116         $expected_value = 1;
117         echo $this->unit->run(count($test), $expected_value, $test_name, 'expected 1 organization');
118     }

```

FIGURE 75

Test Name	isExist test function
Test Datatype	Boolean
Expected Datatype	Boolean
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_organization_model.php
Line Number	126
Notes	expected true

TABLE 35

```

121     public function test_isExist()
122     {
123         $org_id = 78;
124         $test = $this->organization_model->isExist($org_id);
125         $test_name = 'isExist test function';
126         echo $this->unit->run($test, true, $test_name, 'expected true');
127     }

```

FIGURE 76

Test Name	get_all_organizations test function
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_organization_model.php
Line Number	152
Notes	expected 2 members

TABLE 36

```

149     public function test_get_all_organizations(){
150         $test = $this->organization_model->get_all_organizations();
151         $test_name = 'get_all_organizations test function';
152         echo $this->unit->run(count($test), 2, $test_name, 'expected 2 members');
153     }

```

FIGURE 77

Test Name	getAllMembers test function
Test Datatype	Integer
Expected Datatype	Integer
Result	Failed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_organization_model.php
Line Number	142
Notes	Message: Trying to get property 'organization_id' of non-object Filename: organization/Organization_model.php Line Number: 220 I have use index[0] to get the object in the array.

TABLE 37

Test Name	getAllMembers test function
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_organization_model.php
Line Number	142
Notes	expected 2 users

TABLE 38

```

135     public function test_getAllMembers(){
136         $data = [
137             $this->organization_model->getByProfileURL('c58fbc9e6d988f4ba484f546e02193ba')[0],
138             $this->organization_model->getByProfileURL('8cbd4e3e6f785fdc7190d1b1ece66e82')[0]
139         ];
140         $test = $this->organization_model->getAllMembers($data);
141         $test_name = 'getAllMembers test function';
142         echo $this->unit->run(count($test), 2, $test_name, 'expected 2 users');
143     }

```

FIGURE 78

Test Name	getMembers test function
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_organization_model.php

Line Number	134
Notes	expected 2 members

TABLE 39

7.3.7. Reading List Model Test

Test Name	getList test function
Test Datatype	Array
Expected Datatype	Array
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_readinglist_model.php
Line Number	24
Notes	expected array of items

TABLE 40

```

18     public function test_getList()
19     {
20         $user_id = 85;
21         $test = $this->readinglist_model->getList($user_id);
22         $expected_result = 'is_array';
23         $test_name = 'getList test function';
24         echo $this->unit->run($test,$expected_result, $test_name, 'expected array of items');
25     }

```

FIGURE 79

Test Name	getAllList test function
Test Datatype	Array
Expected Datatype	Array
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_readinglist_model.php
Line Number	34
Notes	expected array of items

TABLE 41

```

28     public function test_getAllList()
29     {
30         $user_id = 85;
31         $test = $this->readinglist_model->getAllList($user_id);
32         $expected_result = 'is_array';
33         $test_name = 'getAllList test function';
34         echo $this->unit->run($test,$expected_result, $test_name, 'expected array of items');
35     }

```

FIGURE 80

Test Name	addItem test function
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed

File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_readinglist_model.php
Line Number	47
Notes	expected \$item_after = \$item_before+1

TABLE 42

Test Name	addItem test function
Test Datatype	Null
Expected Datatype	Null
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_readinglist_model.php
Line Number	49
Notes	expected null, no duplicate item

TABLE 43

```

38     public function test_addItem()
39     {
40         $user_id = 85;
41         $items_before = count($this->readinglist_model->getList($user_id));
42         $article_title = 'title';
43         $article_id = 675;
44         $this->readinglist_model->addItem($user_id,$article_title,$article_id);
45         $items_after = count($this->readinglist_model->getList($user_id));
46         $test_name = 'addItem test function';
47         echo $this->unit->run($items_after, $items_before + 1, $test_name, 'expected $item_after = $item_before+1');
48         $test = $this->readinglist_model->addItem($user_id,$article_title,$article_id);
49         echo $this->unit->run($test, 'is_null', $test_name, 'expected null, no duplicate item');
50     }

```

FIGURE 81

Test Name	deleteItem test function
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_readinglist_model.php
Line Number	61
Notes	expected after_deletion=\$before_deletion

TABLE 44

```

53     public function test_deleteItem()
54     {
55         $reading_list_item_id = 260;
56         $user_id = 85;
57         $items_before_deletion = count($this->readinglist_model->getList($user_id));
58         $test = $this->readinglist_model->deleteItem($user_id,$reading_list_item_id);
59         $items_after_deletion = count($this->readinglist_model->getList($user_id));
60         $test_name = 'deleteItem test function';
61         echo $this->unit->run($items_after_deletion, $items_before_deletion-1, $test_name, 'expected after_deletion=$before_deletion');
62     }

```

FIGURE 82

Test Name	check_duplicate_item test function
Test Datatype	Boolean
Expected Datatype	Boolean

Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_readinglist_model.php
Line Number	73
Notes	expected true, duplicate item

TABLE 45

Test Name	check_duplicate_item test function
Test Datatype	Boolean
Expected Datatype	Boolean
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_readinglist_model.php
Line Number	74
Notes	expected false, no duplication

TABLE 46

```

65     public function test_check_duplicate_item()
66     {
67         $reading_list_item_id = 1;
68         $user_id = 85;
69         $test1 = $this->readinglist_model->check_duplicate_item($user_id,$reading_list_item_id);
70         $reading_list_item_id = 999999;
71         $test2 = $this->readinglist_model->check_duplicate_item($user_id,$reading_list_item_id);
72         $test_name = 'check_duplicate_item test function';
73         echo $this->unit->run($test1, true, $test_name, 'expected true, duplicate item');
74         echo $this->unit->run($test2, false, $test_name, 'expected false, no duplication');
75     }

```

FIGURE 83

7.3.8. Settings Model Test

Test Name	getArticleViewsCounter test function
Test Datatype	String
Expected Datatype	Integer
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_settings_model.php
Line Number	23
Notes	expected 5 mins for interval

TABLE 47

```

18     public function test_getArticleViewsCounter()
19     {
20         $test = $this->settings_model->getArticleViewsCounter();
21         $test_name = 'getArticleViewsCounter test function';
22         $expected_result = 5;
23         echo $this->unit->run($test,$expected_result, $test_name, 'expected 5 mins for interval');
24     }

```

FIGURE 84

Test Name	get_allow_comments_per_day test function
Test Datatype	String
Expected Datatype	Integer
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_settings_model.php
Line Number	31
Notes	expected 3 comments per day

TABLE 48

```

26     public function test_get_allow_comments_per_day()
27     {
28         $test = $this->settings_model->get_allow_comments_per_day();
29         $test_name = 'get_allow_comments_per_day test function';
30         $expected_result = 3;
31         echo $this->unit->run($test,$expected_result, $test_name, 'expected 3 comments per day');
32     }

```

FIGURE 85

Test Name	update_settings test function
Test Datatype	Boolean
Expected Datatype	Boolean
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_settings_model.php
Line Number	41
Notes	expected true

TABLE 49

Test Name	update_settings test function
Test Datatype	String
Expected Datatype	Integer
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_settings_model.php
Line Number	44
Notes	expected 10

TABLE 50

Test Name	update_settings test function
Test Datatype	String
Expected Datatype	Integer
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_settings_model.php
Line Number	45
Notes	expected 15

TABLE 51TABLE 52

```

34     public function test_update_settings()
35     {
36         $view_conter = 10;
37         $comment_per_day = 15;
38         $test = $this->settings_model->update_settings($view_conter,$comment_per_day);
39         $test_name = 'update_settings test function';
40         $expected_result = TRUE;
41         echo $this->unit->run($test,$expected_result, $test_name, 'expected true');
42         $view_conter_in_database = $test = $this->settings_model->getArticleViewsCounter();
43         $comments_per_day_in_database = $this->settings_model->get_allow_comments_per_day();
44         echo $this->unit->run($view_conter_in_database,10, $test_name, 'expected 10');
45         echo $this->unit->run($comments_per_day_in_database,15, $test_name, 'expected 15');
46     }
47 }

```

FIGURE 86

7.3.9. Specialty Model Test

Test Name	get_stat_count_each_specialty_by_organization test function
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_specialty_model.php
Line Number	24
Notes	expected array of specialties

```

18     public function test_get_stat_count_each_specialty_by_organization()
19     {
20         $org_id = 78;
21         $test = $this->specialty_model->get_stat_count_each_specialty_by_organization($org_id);
22         $test_name = 'get_stat_count_each_specialty_by_organization test function';
23         $expected_result = 2;
24         echo $this->unit->run(count(json_decode($test)->data),$expected_result, $test_name, 'expected 2 specialties');
25     }

```

FIGURE 87

7.3.10. Sub_Specialty Model Test

Test Name	get_sub_specialties test function
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	C:\xampp\htdocs\medical-

	plus\application\controllers\test_models\Test_subspecialty_model.php
Line Number	24
Notes	expected 6 sub-specialties for specialtyid 18 that stored in database

TABLE 53

```

18 public function test_get_sub_specialties()
19 {
20     $specialty_id = 18;
21     $test = $this->subspecialty_model->get_sub_specialties($specialty_id);
22     $expected_result = 6;
23     $test_name = 'get_sub_specialties test function';
24     echo $this->unit->run(count($test),$expected_result, $test_name, 'expected 6 sub-specialties for
25     | specialtyid 18 that stored in database');
26 }

```

FIGURE 88

7.3.11. User Model Test

Test Name	inser_new_user test function
Test Datatype	Null
Expected Datatype	String
Result	Failed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\tests\Test_user_model.php
Line Number	37
Notes	after insert a user, the expected return value is the identifier of the user, by using the another function to get the name of the user to check if desired name
Error	Message: Trying to get property 'user_first_name' of non-object Filename: tests/Test_user_model.php Line Number: 32 getUser function return an array of user data , should use the zero index to get the user first name

TABLE 54

Test Name	inser_new_user test function
Test Datatype	String
Expected Datatype	String
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\tests\Test_user_model.php
Line Number	37
Notes	after insert a user, the expected return value is the identifier of the user, by using the another function to get the name of the user to check if desired name

TABLE 55

```

24 public function test_insert_new_user(){
25
26     $data = [
27         'user_first_name' => 'ahmed',
28         'user_email' => 'email@test.com'
29     ];
30
31     $user_id = $this->user_model->insert_new_user($data);
32     $test = $this->user_model->getUser($user_id)[0]->user_first_name;
33     $expected_result = 'ahmed';
34     $test_name = 'insert_new_user test function';
35     echo $this->unit->run($test,$expected_result,$test_name,'after insert a user, the expected return value' .
36                                     'is the identifier of the user, by using the another'.
37                                     ' function to get the name of the user to check if desired name');

```

FIGURE 89

Test Name	verify_user test function
Test Datatype	Boolean
Expected Datatype	Boolean
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\tests\Test_user_model.php
Line Number	51
Notes	verify the user email after registration

TABLE 56

```

41 public function test_verify_user(){
42     $data = [
43         'user_first_name' => 'ahmed',
44         'user_email' => 'email@test.com'
45     ];
46     $user_id = $this->user_model->insert_new_user($data);
47     $verification_key = $this->user_model->getUser($user_id)[0]->verification_key;
48     $test = $this->user_model->verify_user($verification_key);
49     $expected_result = TRUE;
50     $test_name = 'verify_user test function';
51     echo $this->unit->run($test,$expected_result,$test_name,'verify the user email after registration');
52 }

```

FIGURE 90

Test Name	can_login test function
Test Datatype	Array
Expected Datatype	Array
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\tests\Test_user_model.php
Line Number	68
Notes	Expected an array of data

TABLE 57

Test Name	can_login test function
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\tests\Test_user_model.php

Line Number	69
Notes	Expected just one item in the returned array

TABLE 58

Test Name	can_login test function
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\tests\Test_user_model.php
Line Number	72
Notes	Expected empty array

TABLE 59

```

55 public function test_can_login(){
56     $data = [
57         'user_first_name' => 'ahmed',
58         'user_email' => 'email@test.com',
59         'user_password' => '1'
60     ];
61     $user_id = $this->user_model->insert_new_user($data);
62     $username = 'email@test.com';
63     $correct_password = '1';
64     $wrong_password = '123';
65     $test = $this->user_model->can_login($username,$correct_password);
66     $expected_array_count = 1;
67     $test_name = 'can_login test function';
68     echo $this->unit->run($test,'is_array',$test_name,'Expected an array of data');
69     echo $this->unit->run(count($test),$expected_array_count,$test_name,'Expected just one item in the returned array');
70     $test = $this->user_model->can_login($username,$wrong_password);
71     $expected_array_count = 0;
72     echo $this->unit->run(count($test),$expected_array_count,$test_name,'Expected empty array');
73 }

```

FIGURE 91

Test Name	is_user_manager test function
Test Datatype	Boolean
Expected Datatype	Boolean
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\tests\Test_user_model.php
Line Number	87
Notes	Expected TRUE because the user is an organizer

TABLE 60

Test Name	is_user_manager test function
Test Datatype	Boolean
Expected Datatype	Boolean
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\tests\Test_user_model.php
Line Number	88
Notes	Expected FALSE because the user is a member

TABLE 61

```

76     public function test_is_user_manager(){
77         /**
78          * in database I have 2 stored users:
79          * USER(1) USER EMAIL: ok.yes@hotmail.com AS organizer
80          * USER(2) USER EMAIL: ok.yes@live.com AS member
81          */
82         $test1 = $this->user_model->is_user_manager('ok.yes@hotmail.com');
83         $test2 = $this->user_model->is_user_manager('ok.yes@live.com');
84         $expected_for_test1 = TRUE;
85         $expected_for_test2 = FALSE;
86         $test_name = 'is_user_manager test function';
87         echo $this->unit->run($test1,$expected_for_test1,$test_name,'Expected TRUE because the user is an organizer');
88         echo $this->unit->run($test2,$expected_for_test2,$test_name,'Expected FALSE because the user is a member');
89     }

```

FIGURE 92

Test Name	getUser test function
Test Datatype	String
Expected Datatype	String
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\tests\Test_user_model.php
Line Number	99
Notes	expected return value "ahmed"

TABLE 62

```

90     public function test_getUser()
91     {
92         /**
93          * in database I have 2 stored users:
94          * email@test.com has identifier 106 and name ahmed
95          */
96         $test = $this->user_model->getUser(106)[0]->user_first_name;
97         $expected_result = 'ahmed';
98         $test_name = 'getUser test function';
99         echo $this->unit->run($test, $expected_result, $test_name, 'expected return value "ahmed"');
100     }

```

FIGURE 93

Test Name	getUserByProfileURL test function
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\tests\Test_user_model.php
Line Number	115
Notes	expected return an array of one item

TABLE 63

Test Name	getUserByProfileURL test function
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\tests\Test_user_model.php
Line Number	119

Notes	expected return an empty array
-------	--------------------------------

TABLE 64

```

103     public function test_getUserByProfileURL(){
104         $userurl = md5('email@test.com');
105         $data = [
106             'user_first_name' => 'ahmed',
107             'user_email' => 'email@test.com',
108             'user_password' => '1',
109             'user_profile_url' => $userurl
110         ];
111         $this->user_model->insert_new_user($data);
112         $array = $this->user_model->getUserByProfileURL($userurl);
113         $expected_result = 1;
114         $test_name = 'getUserByProfileURL test function';
115         echo $this->unit->run(count($array), $expected_result, $test_name, 'expected return an array of one item');
116         $expected_result = 0;
117         $userurl = md5('email@test.com');
118         $array = $this->user_model->getUserByProfileURL($userurl);
119         echo $this->unit->run(count($array), $expected_result, $test_name, 'expected return an empty array');
120     }

```

FIGURE 94

Test Name	updateBio test function
Test Datatype	Boolean
Expected Datatype	bool
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\tests\Test_user_model.php
Line Number	128
Notes	expected Boolean

TABLE 65

Test Name	updateBio test function
Test Datatype	String
Expected Datatype	String
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\tests\Test_user_model.php
Line Number	130
Notes	expected return value "ABC"

TABLE 66

```

123     public function test_updateBio(){
124         $Bio = "ABC";
125         $user_id = 86;
126         $test = $this->user_model->updateBio($user_id,$Bio);
127         $test_name = 'updateBio test function';
128         echo $this->unit->run($test, 'is_bool', $test_name, 'expected boolean');
129         $expected_result = $this->user_model->getUser($user_id)[0]->brief;
130         echo $this->unit->run("ABC", $expected_result, $test_name, 'expected return value "ABC"');
131     }

```

FIGURE 95

Test Name	updateUserName test function
-----------	------------------------------

Test Datatype	Boolean
Expected Datatype	bool
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\tests\Test_user_model.php
Line Number	140
Notes	expected Boolean

TABLE 67

Test Name	updateUserName test function
Test Datatype	String
Expected Datatype	String
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\tests\Test_user_model.php
Line Number	142
Notes	expected return value "last"

TABLE 68

```

134     public function test_updateUserName(){
135         $f_name = "first";
136         $l_name = "last";
137         $user_id = 86;
138         $test = $this->user_model->updateUserName($user_id,$f_name,$l_name);
139         $test_name = 'updateUserName test function';
140         echo $this->unit->run($test, 'is_bool', $test_name, 'expected boolean');
141         $expected_result = $this->user_model->getUser($user_id)[0]->user_last_name;
142         echo $this->unit->run("last", $expected_result, $test_name, 'expected return value "last"');
143     }

```

FIGURE 96

Test Name	updateSpecialty test function
Test Datatype	Boolean
Expected Datatype	bool
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\tests\Test_user_model.php
Line Number	151
Notes	expected Boolean

TABLE 69

Test Name	updateSpecialty test function
Test Datatype	String
Expected Datatype	String
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\tests\Test_user_model.php
Line Number	153
Notes	expected return value "46"

TABLE 70


```

146     public function test_updateSpecialty(){
147         $specialty_id = 46;
148         $user_id = 86;
149         $test = $this->user_model->updateSpecialty($user_id,$specialty_id);
150         $test_name = 'updateSpecialty test function';
151         echo $this->unit->run($test, 'is_bool', $test_name, 'expected boolean');
152         $expected_result = $this->user_model->getUser($user_id)[0]->subspecialty_id;
153         echo $this->unit->run("46", $expected_result, $test_name, 'expected return value "46"');
154     }

```

FIGURE 97

Test Name	updateImageURL test function
Test Datatype	Boolean
Expected Datatype	bool
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\tests\Test_user_model.php
Line Number	162
Notes	expected Boolean

TABLE 71

Test Name	updateImageURL test function
Test Datatype	String
Expected Datatype	String
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\tests\Test_user_model.php
Line Number	164
Notes	expected imageurl

TABLE 72

```

157     public function test_updateImageURL(){
158         $new_image_url = 'imageurl';
159         $user_id = 86;
160         $test = $this->user_model->updateImageURL($user_id,$new_image_url);
161         $test_name = 'updateImageURL test function';
162         echo $this->unit->run($test, 'is_bool', $test_name, 'expected boolean');
163         $expected_result = $this->user_model->getUser($user_id)[0]->user_img_url;
164         echo $this->unit->run('imageurl', $expected_result, $test_name, 'expected imageurl');
165     }

```

FIGURE 98

Test Name	updateUserEmail test function
Test Datatype	Boolean
Expected Datatype	bool
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\tests\Test_user_model.php
Line Number	173
Notes	expected Boolean

TABLE 73

Test Name	updateUserEmail test function
-----------	-------------------------------

Test Datatype	String
Expected Datatype	String
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\tests\Test_user_model.php
Line Number	175
Notes	expected ok.yes@live.org

TABLE 74

```

168  public function test_updateUserEmail(){
169      $new_email = 'ok.yes@live.org';
170      $user_id = 86;
171      $test = $this->user_model->updateUserEmail($user_id,$new_email);
172      $test_name = 'updateUserEmail test function';
173      echo $this->unit->run($test, 'is_bool', $test_name, 'expected boolean');
174      $expected_result = $this->user_model->getUser($user_id)[0]->user_email;
175      echo $this->unit->run('ok.yes@live.org', $expected_result, $test_name, 'expected ok.yes@live.org');
176  }

```

FIGURE 99

Test Name	updateUserPassword test function
Test Datatype	Boolean
Expected Datatype	bool
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\tests\Test_user_model.php
Line Number	184
Notes	expected Boolean

TABLE 75

Test Name	updateUserPassword test function
Test Datatype	String
Expected Datatype	String
Result	Failed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\tests\Test_user_model.php
Line Number	186
Notes	Has to compare with MD5 function, expected 123

TABLE 76

Test Name	updateUserPassword test function
Test Datatype	String
Expected Datatype	String
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\tests\Test_user_model.php
Line Number	186
Notes	expected md5(123)

TABLE 77

```

179     public function test_updateUserPassword(){
180         $new_pass = '123';
181         $user_id = 86;
182         $test = $this->user_model->updateUserPassword($user_id,$new_pass);
183         $test_name = 'updateUserPassword test function';
184         echo $this->unit->run($test, 'is_bool', $test_name, 'expected boolean');
185         $expected_result = $this->user_model->getUser($user_id)[0]->user_password;
186         echo $this->unit->run(md5('123'), $expected_result, $test_name, 'expected 123');
187     }

```

FIGURE 100

Test Name	is_current_password_correct test function
Test Datatype	Boolean
Expected Datatype	bool
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\tests\Test_user_model.php
Line Number	195
Notes	expected Boolean

TABLE 78

Test Name	is_current_password_correct test function
Test Datatype	Boolean
Expected Datatype	Boolean
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\tests\Test_user_model.php
Line Number	197
Notes	expected true

TABLE 79

```

190     public function test_is_current_password_correct(){
191         $current_pass = '123';
192         $user_id = 86;
193         $test = $this->user_model->is_current_password_correct($user_id,$current_pass);
194         $test_name = 'is_current_password_correct test function';
195         echo $this->unit->run($test, 'is_bool', $test_name, 'expected boolean');
196         $expected_result = TRUE;
197         echo $this->unit->run($test, $expected_result, $test_name, 'expected true');
198     }

```

FIGURE 101

Test Name	setLastVisit test function
Test Datatype	Boolean
Expected Datatype	bool
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\tests\Test_user_model.php
Line Number	205

Notes	expected Boolean
-------	------------------

TABLE 80

Test Name	setLastVisit test function
Test Datatype	Boolean
Expected Datatype	Boolean
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\tests\Test_user_model.php
Line Number	207
Notes	expected true

TABLE 81

```

201     public function test_setLastVisit(){
202         $user_id = 86;
203         $test = $this->user_model->setLastVisit($user_id);
204         $test_name = 'setLastVisit test function';
205         echo $this->unit->run($test, 'is_bool', $test_name, 'expected boolean');
206         $expected_result = TRUE;
207         echo $this->unit->run($test, $expected_result, $test_name, 'expected true');
208     }

```

FIGURE 102

Test Name	get_stat_users_comments_and_reads_by_org test function
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\tests\Test_user_model.php
Line Number	217
Notes	expected return an array of one item

TABLE 82

Test Name	get_stat_users_comments_and_reads_by_org test function
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\tests\Test_user_model.php
Line Number	220
Notes	expected return an empty array

TABLE 83

```

211     public function test_get_stat_users_comments_and_reads_by_org(){
212         $exists_org_id = 78;
213         $not_exists_org_id = 123; // no such organization with this id
214         $test1 = $this->user_model->get_stat_users_comments_and_reads_by_org($exists_org_id);
215         $expected_result = 2; // 2 members in the organization
216         $test_name = 'get_stat_users_comments_and_reads_by_org test function';
217         echo $this->unit->run(count($test1), $expected_result, $test_name, 'expected return an array of one item');
218         $expected_result = 0;
219         $test2 = $this->user_model->get_stat_users_comments_and_reads_by_org($not_exists_org_id);
220         echo $this->unit->run(count($test2), $expected_result, $test_name, 'expected return an empty array');
221     }

```

FIGURE 103

Test Name	get_all_users test function
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\tests\Test_user_model.php
Line Number	228
Notes	expected true

TABLE 84

```

224     public function test_get_all_users(){
225         $test = $this->user_model->get_all_users();
226         $test_name = 'get_all_users test function';
227         $expected_result = 3; // all users in the system
228         echo $this->unit->run(count($test), $expected_result, $test_name, 'array of user');
229     }

```

FIGURE 104

Test Name	isEmailVerified test function
Test Datatype	Integer
Expected Datatype	String
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\tests\Test_user_model.php
Line Number	259
Notes	expected 1 because ok.yes@live.org is verified

TABLE 85

Test Name	changeBlockState test function
Test Datatype	String
Expected Datatype	String
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\tests\Test_user_model.php
Line Number	237
Notes	expect 1

TABLE 86

Test Name	changeBlockState test function
Test Datatype	String
Expected Datatype	String
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\tests\Test_user_model.php
Line Number	240
Notes	expect 0

TABLE 87

```

232     public function test_chengeBlockState(){
233         $user_id = 86; // state 0
234         $this->user_model->chengeBlockState($user_id);
235         $test_name = 'chengeBlockState test function';
236         $expected_result = $this->user_model->getUser($user_id)[0]->isblocked;
237         echo $this->unit->run($expected_result, '1', $test_name, 'expect 1');
238         $this->user_model->chengeBlockState($user_id);
239         $expected_result = $this->user_model->getUser($user_id)[0]->isblocked;
240         echo $this->unit->run($expected_result, '0', $test_name, 'expect 0');
241     }

```

FIGURE 105

Test Name	is_registered_before test function
Test Datatype	Integer
Expected Datatype	String
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\tests\Test_user_model.php
Line Number	248
Notes	expected 1 because ok.yes@live.org exists in database

TABLE 88

Test Name	is_registered_before test function
Test Datatype	Integer
Expected Datatype	String
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\tests\Test_user_model.php
Line Number	251
Notes	expected 0 because ok.yes@live.orgggg does not exist in database

TABLE 89

```

244     public function test_is_registered_before(){
245         $email = 'ok.yes@live.org';
246         $test = $this->user_model->is_registered_before($email);
247         $test_name = 'is_registered_before test function';
248         echo $this->unit->run($test, '1', $test_name, 'expected 1 because ok.yes@live.org exists in database');
249         $email = 'ok.yes@live.orgggg';
250         $test = $this->user_model->is_registered_before($email);
251         echo $this->unit->run($test, '0', $test_name, 'expected 0 because ok.yes@live.orgggg does not exist in database');
252     }

```

FIGURE 106

7.3.12. View Model Test

Test Name	getCountViews test function
Test Datatype	Integer
Expected Datatype	Integer

Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_view_model.php
Line Number	24
Notes	expected 22 views that stored in database for article_id 1

TABLE 90

```

18     public function test_getCountViews()
19     {
20         $article_id = 1;
21         $test = $this->view_model->getCountViews($article_id);
22         $expected_result = 22;
23         $test_name = 'getCountViews test function';
24         echo $this->unit->run($test,$expected_result, $test_name, 'expected 22 views that stored in database for article_id 1');
25     }

```

FIGURE 107

Test Name	setViewArticle test function
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_view_model.php
Line Number	39
Notes	Expected 23 view, this function stores users views on the articles, to calculate how many views for each article. an INTERVAL will set for each opened article by the user, first time a user opens an article, will save one row in the table, during the INTERVAL, if the user opened the article second time, will not add a new row in the table, unless the INTERVAL finishes.

TABLE 91

```

28     public function test_setViewArticle()
29     {
30         // the interval is 5 mins
31         $article_id = 1;
32         $user_id = 85;
33         $views_count = $this->view_model->getCountViews($article_id);
34         $this->view_model->setViewArticle($article_id,$user_id);
35         $this->view_model->setViewArticle($article_id,$user_id);
36         $this->view_model->setViewArticle($article_id,$user_id);
37         $expected_value = $views_count + 1;
38         $test_name = 'setViewArticle test function';
39         echo $this->unit->run(23, $expected_value, $test_name, 'Expected 23 view, this function stores users views on the articles,
40         to calculate how many views for each article.
41         an INTERVAL will set for each opened article by the user,
42         first time a user opens an article,
43         will save one row in the table, during the INTERVAL,
44         if the user opened the article second time,
45         will not add a new row in the table, unless the INTERVAL finishes.');
```

FIGURE 108

Test Name	get_stat_article_read_by_user test function
Test Datatype	Array
Expected Datatype	Array
Result	Passed

File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_view_model.php
Line Number	55
Notes	expected array of user statistics about reading articles

TABLE 92

```

49     public function test_get_stat_article_read_by_user()
50     {
51         $user_id = 85;
52         $test = $this->view_model->get_stat_article_read_by_user($user_id,10);
53         $test_name = 'get_stat_article_read_by_user test function';
54         $array = json_decode($test);
55         echo $this->unit->run($array->data, 'is_array', $test_name, 'expected array of user statistics about reading articles');
56     }

```

FIGURE 109

Test Name	get_stat_article_read_by_users_for_organization test function
Test Datatype	Array
Expected Datatype	Array
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_view_model.php
Line Number	65
Notes	expected array of org statistics about reading articles

TABLE 93

```

59     public function test_get_stat_article_read_by_users_for_organization()
60     {
61         $org_id = 85;
62         $test = $this->view_model->get_stat_article_read_by_users_for_organization($org_id,10);
63         $test_name = 'get_stat_article_read_by_users_for_organization test function';
64         $array = json_decode($test);
65         echo $this->unit->run($array->data, 'is_array', $test_name, 'expected array of org statistics about reading articles');
66     }

```

FIGURE 110

Test Name	get_count_articles_read_by_user test function
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	C:\xampp\htdocs\medical-plus\application\controllers\test_models\Test_view_model.php
Line Number	75
Notes	expected 19 reads that stored in database for the user

TABLE 94

```

69     public function test_get_count_articles_read_by_user()
70     {
71         $user_id = 85;
72         $test = $this->view_model->get_count_articles_read_by_user($user_id);
73         $expected_result = 19;
74         $test_name = 'get_count_articles_read_by_user test function';
75         echo $this->unit->run($test,$expected_result, $test_name, 'expected 19 reads that stored in database for the user');
76     }

```

FIGURE 111

7.4. Integration Tests

These following are what resulted from running Selenium on performing the specified actions

- **Running 'Test Login Page failed'** 16:51:01
 - 1.open on <http://localhost/medical-plus/authentication> OK 16:51:02
 - 2.setWindowSize on 750x800 OK 16:51:03
 - 3.click on name=login_email OK 16:51:04
 - 4.type on name=login_email with value ok.yes@hotmail.com OK 16:51:06
 - 5.click on name=login_password OK 16:51:07
 - 6.type on name=login_password with value 123 OK 16:51:09
 - 7.click on css=.col-lg-6:nth-child(4) > button OK 16:51:10
 - 8.click on css=.text-danger OK 16:51:11
 - 9.verifyText on css=.text-danger with value Invalid username or password. OK 16:51:13
 - **'Test Login Page' completed successfully**
-
- **Running 'Test Login Page Success'** 17:26:26
 - 1.open on <http://localhost/medical-plus/authentication> OK 17:26:26
 - 2.setWindowSize on 754x800 OK 17:26:28
 - 3.click on name=login_email OK 17:26:29
 - 4.type on name=login_email with value ok.yes@hotmail.com OK 17:26:30
 - 5.click on name=login_password OK 17:26:32
 - 6.type on name=login_password with value 1 OK 17:26:33
 - 7.click on css=.col-lg-6:nth-child(4) > button OK 17:26:35
 - 8.click on css=.suggestions h3 OK 17:26:36
 - 9.click on css=.suggestions-list OK 17:26:38
 - 10.storeWindowHandle on root OK 17:26:39
 - 11.selectWindow on handle=\${root} OK 17:26:40
 - 12.click on css=.suggestions h3 OK 17:26:41
 - 13.click on css=.suggestions h3 OK 17:26:43
 - 14.click on css=.suggestions h3 OK 17:26:44
 - 15.doubleClick on css=.suggestions h3 OK 17:26:46
 - 16.verifyTitle on Medical Plus - Feed Page OK 17:26:47
 - **'Test Login Page Success' completed successfully** 17:26:47
-
- **Running 'Test Show user Profile'** 17:38:50
 - 1.open on <http://localhost/medical-plus/oprofile> OK 17:38:51
 - 2.setWindowSize on 756x803 OK 17:38:52
 - 3.click on css=.suggestions:nth-child(4) .suggestion-usd:nth-child(1) h4 OK 17:38:53
 - 4.click on css=.sd-title:nth-child(2) > p OK 17:38:55
 - 5.verifyTitle on Medical Plus - user profile Page OK 17:38:56
 - **'Test Show user Profile' completed successfully** 17:38:57

- **Running 'Test Upload Cover Page of Organization'** 17:48:24
- 1.open on <http://localhost/medical-plus/oprofile> OK 17:48:25
- 2.setWindowSize on 1552x840 OK 17:48:26
- 3.click on id=nav_firstname OK 17:48:27
- 4.click on linkText=Manage Organization OK 17:48:29
- 5.click on css=li:nth-child(2) > a > span:nth-child(2) OK 17:48:31
- 6.click on id=saved-tab OK 17:48:33
- 7.click on id=image_file_cover OK 17:48:34
- 8.type on id=image_file_cover with value C:\fakepath\Movie-WB-1600x400.png **Failed:** 17:48:35

{"code":-32000,"message":"Not allowed"}

- **'Test Upload Cover Page of Organization' ended with 1 error(s)**

- **Running 'Change specialty of a user'** 17:58:55
- 1.open on <http://localhost/medical-plus/oprofile> OK 17:58:56
- 2.setWindowSize on 766x804 OK 17:58:57
- 3.click on id=nav_firstname OK 17:58:58
- 4.click on linkText=Manage Profile OK 17:59:00
- 5.click on id=specialty OK 17:59:01
- 6.click on id=specialty OK 17:59:03
- 7.click on id=sub_specialty OK 17:59:04
- 8.select on id=sub_specialty with value label=Transplantation OK 17:59:06
- 9.click on id=sub_specialty OK 17:59:07
- 10.click on id=btn_update_specialty OK 17:59:08
- **'Change specialty of a user' completed successfully** 17:59:09

- **Running 'delete an article from reading list'** 18:10:44
- 1.open on <http://localhost/medical-plus/oprofile> OK 18:10:45
- 2.setWindowSize on 1552x840 OK 18:10:46
- 3.click on linkText=Feed OK 18:10:47
- 4.Trying to find css=#\32 87 > .far... OK 18:10:55
- **'delete an article from reading list' completed successfully** 18:10:56

- **Running 'Test Search In Members List'** 18:25:35
- 1.open on <http://localhost/medical-plus/oprofile> OK 18:25:35
- 2.setWindowSize on 1552x840 OK 18:25:37
- 3.click on linkText=View More OK 18:25:38
- 4.click on css=input OK 18:25:40
- 5.type on css=input with value Talal OK 18:25:42
- 6.click on css=h4 OK 18:25:43
- 7.click on id=fullname OK 18:25:45
- 8.assertText on id=fullname with value Talal Hamed OK 18:25:47
- **'Test Search In Members List' completed successfully** 18:25:47

- **Running 'Test register new member, with exists email address'** 18:53:15
- 1.open on <http://localhost/medical-plus/authentication> OK 18:53:16

- 2.setWindowSize on 1552x840 OK 18:53:17
- 3.click on linkText=Sign up OK 18:53:19
- 4.click on name=member_email OK 18:53:20
- 5.type on name=member_email with value user4@hotmail.com OK 18:53:22
- 6.type on name=member_password with value 123 OK 18:53:23
- 7.type on name=member_repeat_password with value 123 OK 18:53:25
- 8.type on name=member_first_name with value Kaled OK 18:53:26
- 9.type on name=member_last_name with value Ahmed OK 18:53:28
- 10.click on id=specialty_member_form OK 18:53:29
- 11.click on id=specialty_member_form OK 18:53:31
- 12.click on id=sub_specialty_member_form OK 18:53:33
- 13.click on id=sub_specialty_member_form OK 18:53:34
- 14.click on name=organizationID OK 18:53:36
- 15.type on name=organizationID with value 78 OK 18:53:37
- 16.click on css=.col-lg-6:nth-child(9) > button OK 18:53:39
- 17.click on css=#member_form .col-lg-12:nth-child(1) OK 18:53:40
- 18.assertText on css=.col-lg-12 > .text-danger with value This Email already exists please enter another email address OK18:53:42
- **'Test register new member, with exists email address' completed successfully** 18:53:43

- **Running 'Test register new member, successful registration'** 18:59:57
- 1.open on <http://localhost/medical-plus/authentication> OK 18:59:57
- 2.setWindowSize on 1552x840 OK 18:59:59
- 3.click on linkText=Sign up OK 19:00:00
- 4.click on name=member_email OK 19:00:02
- 5.type on name=member_email with value user5@hotmail.com OK 19:00:03
- 6.type on name=member_password with value 1 OK 19:00:05
- 7.type on name=member_repeat_password with value 1 OK 19:00:06
- 8.type on name=member_first_name with value Reem OK 19:00:08
- 9.type on name=member_last_name with value Talal OK 19:00:09
- 10.type on name=organizationID with value 78 OK 19:00:11
- 11.click on css=.col-lg-6:nth-child(9) > button OK 19:00:12
- 12.assertText on css=.col-lg-12 > .text-success with value check your email address OK 19:00:14
- **'Test register new member, with exists email address' completed successfully** 19:00:15

7.5. Stress and Performance Tests

- Testing with 10 virtual users:

Quick Summary

Maximum virtual users:	10
Duration:	10 min, 54 s
Total hits:	9,554
Average hits per second:	14.61
Total errors:	0

FIGURE 112

Requests Summary

Label	#Samples	Average [ms]	Median [ms]	90% line [ms]	95% line [ms]	99% line [ms]	Minimum [ms]	Maximum [ms]	Error Rate	Rate / sec	Avg. Bandwidth [KB/s]	Std. Deviation
001_Medical Plus - Login Page	281	8863	8408	10719	11555	13766	7293	14396	0.00%	0.4	161.6	1379.74
TOTAL	281	8863	8408	10719	11555	13766	7293	14396	0.00%	0.4	161.6	1379.74

FIGURE 113

HitsPerSecond

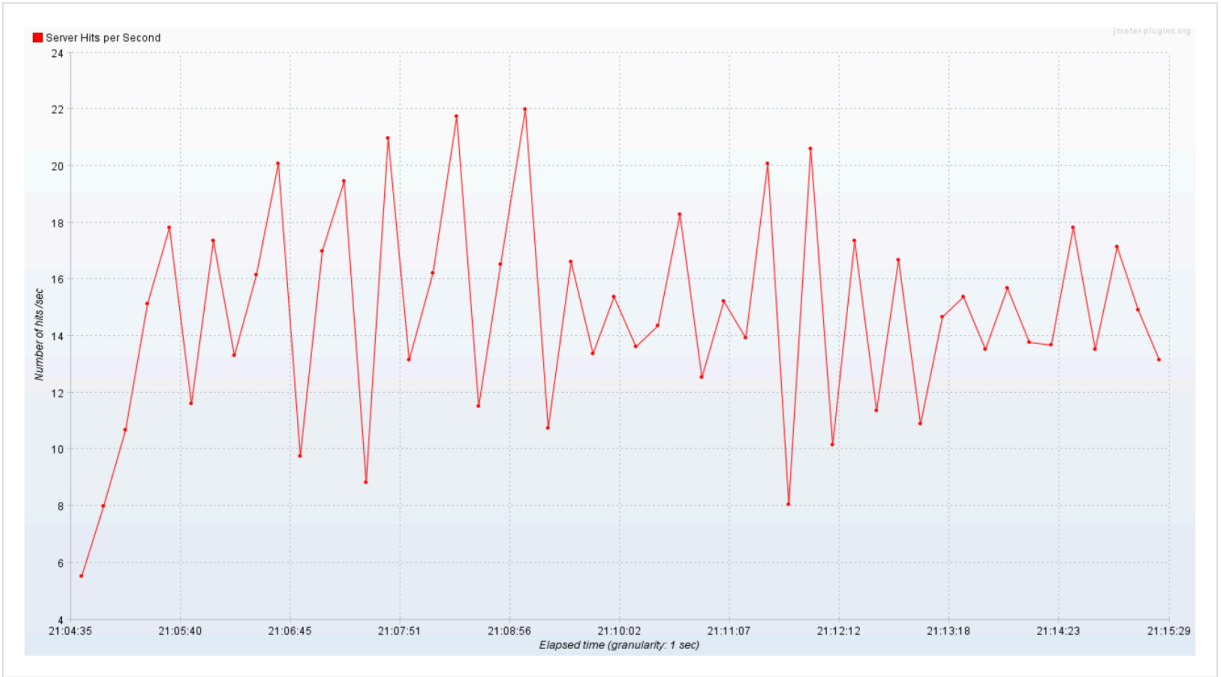


FIGURE 114

ResponseCodesPerSecond

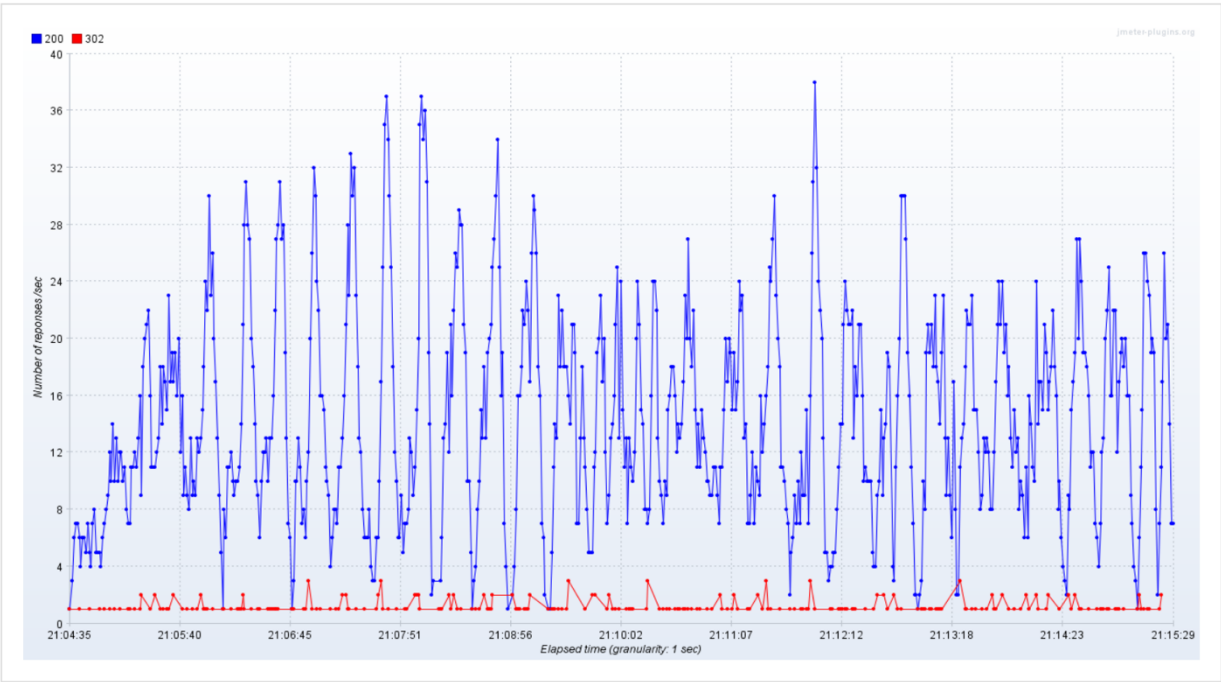


FIGURE 115

ResponseTimesOverTime

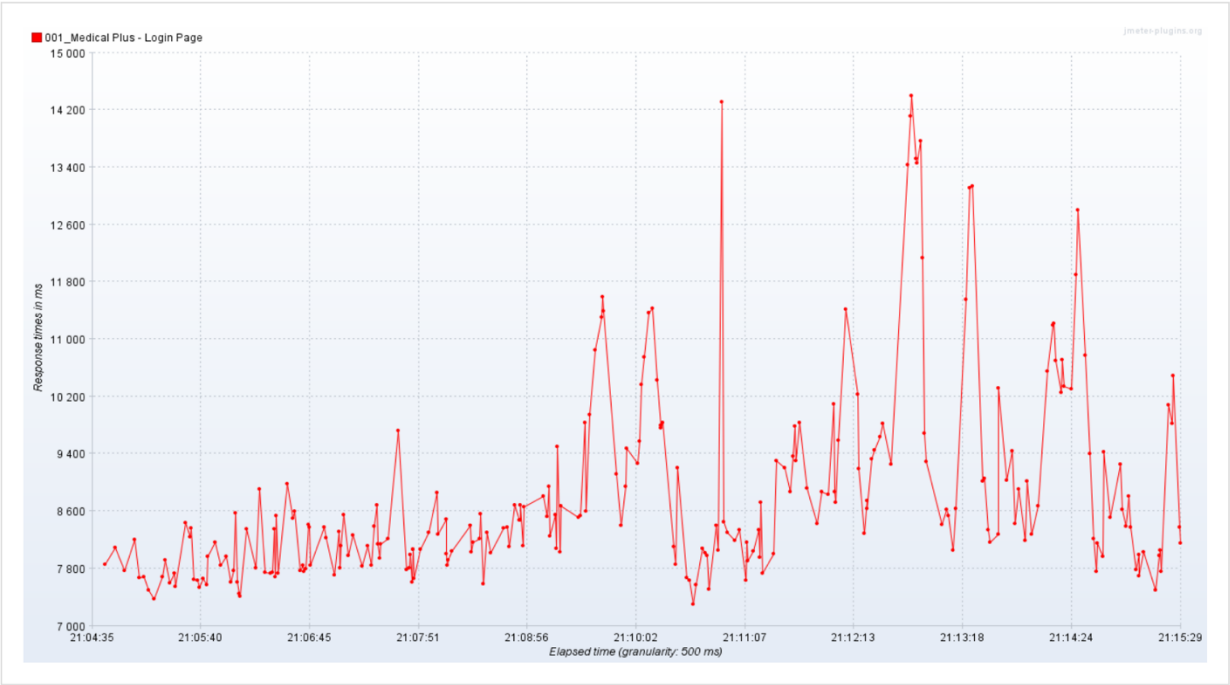


FIGURE 116

ResponseTimesOverTimeAggregate

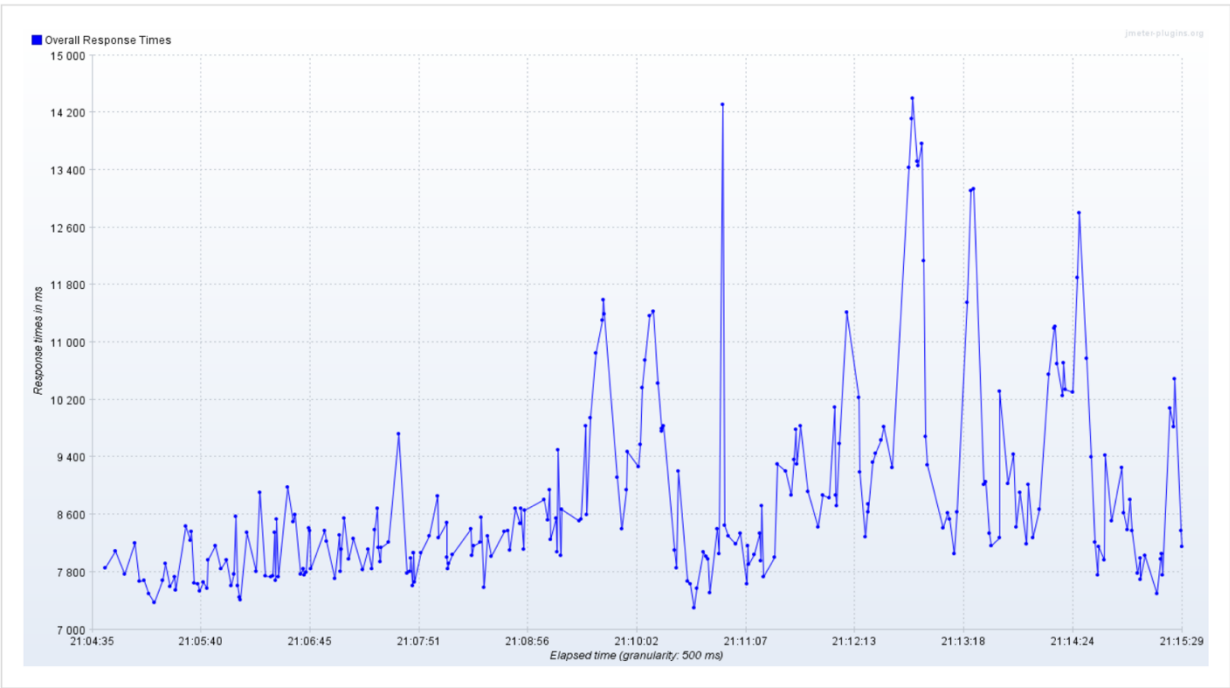


FIGURE 117

- Testing with 150 virtual users:

Quick Summary

Maximum virtual users:	150
Duration:	14 min, 58 s
Total hits:	20,842
Average hits per second:	23.21
Total errors:	1,915

FIGURE 118

Requests Summary

Label	#Samples	Average [ms]	Median [ms]	90% line [ms]	95% line [ms]	99% line [ms]	Minimum [ms]	Maximum [ms]	Error Rate	Rate / sec	Avg. Bandwidth [KB/s]	Std. Deviation
001_Medical Plus - Login Page	613	145277	139571	284201	329895	394376	9080	631272	100.00%	0.7	242.8	100129.69
TOTAL	613	145277	139571	284201	329895	394376	9080	631272	100.00%	0.7	242.8	100129.69

FIGURE 119

HitsPerSecond

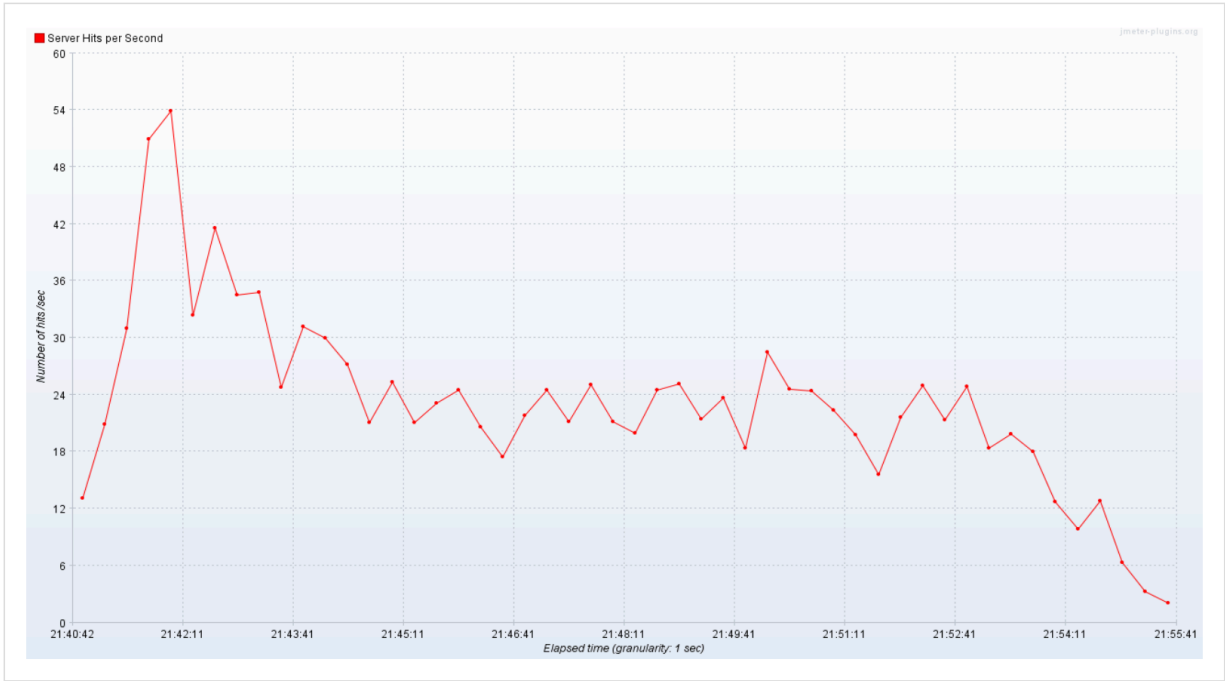


FIGURE 120

ResponseCodesPerSecond

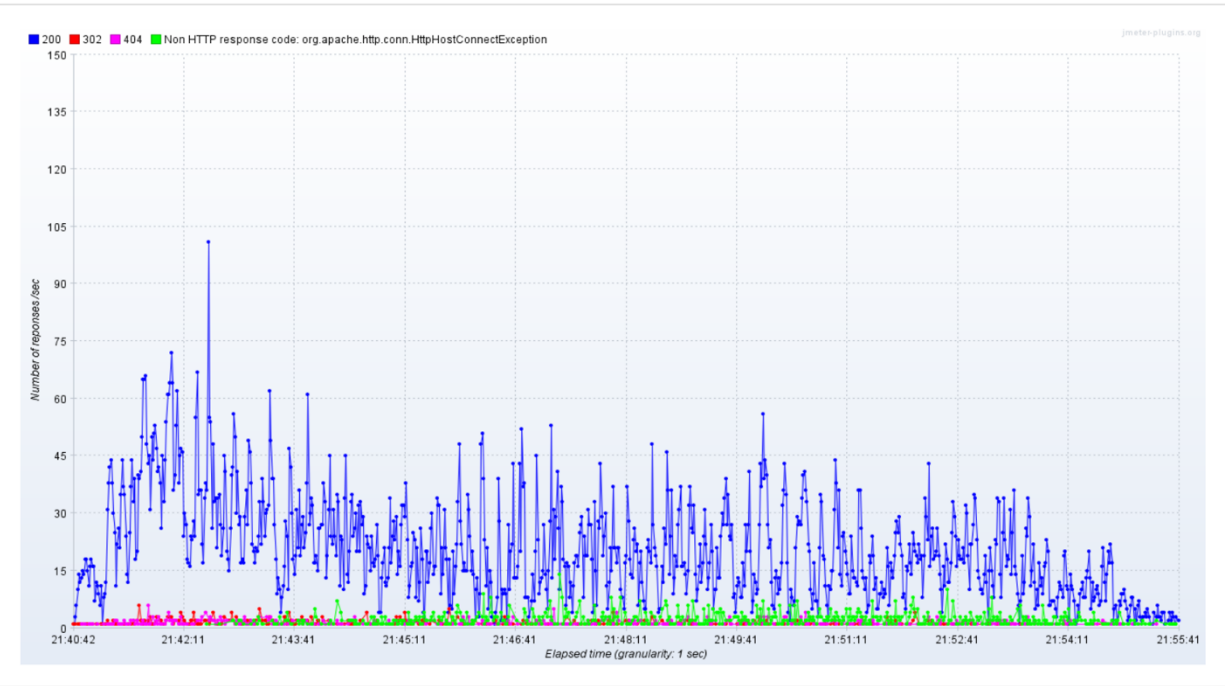


FIGURE 121

ResponseTimesOverTime

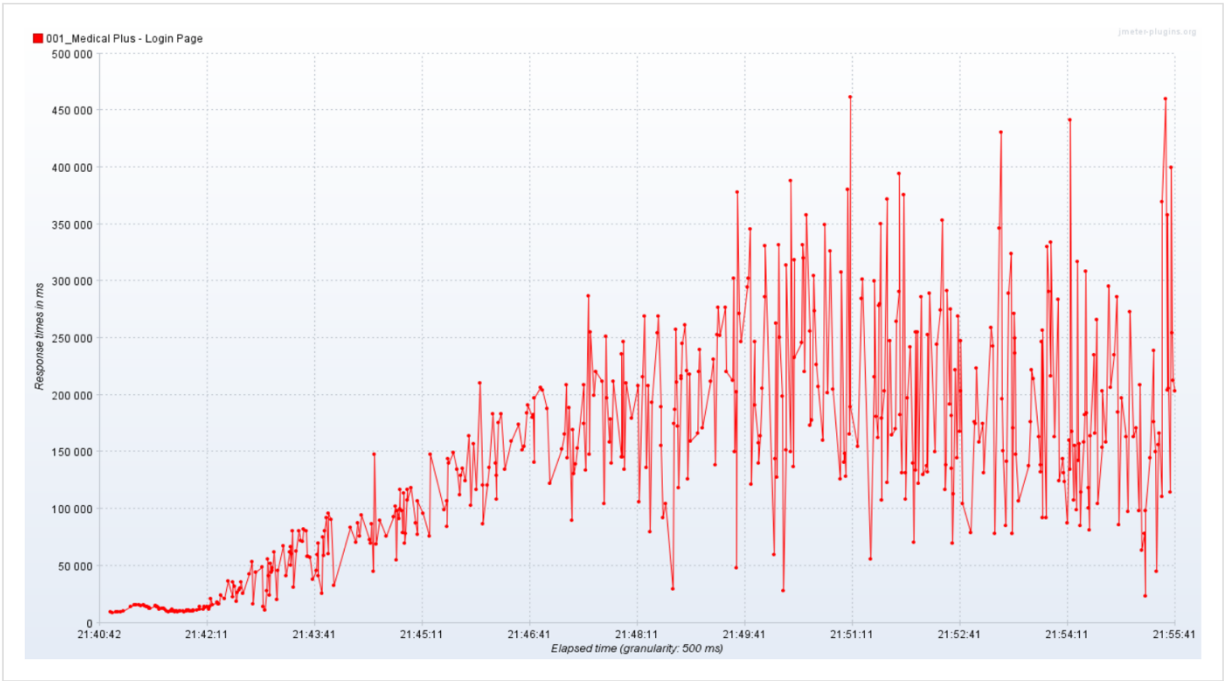


FIGURE 122

TransactionsPerSecond

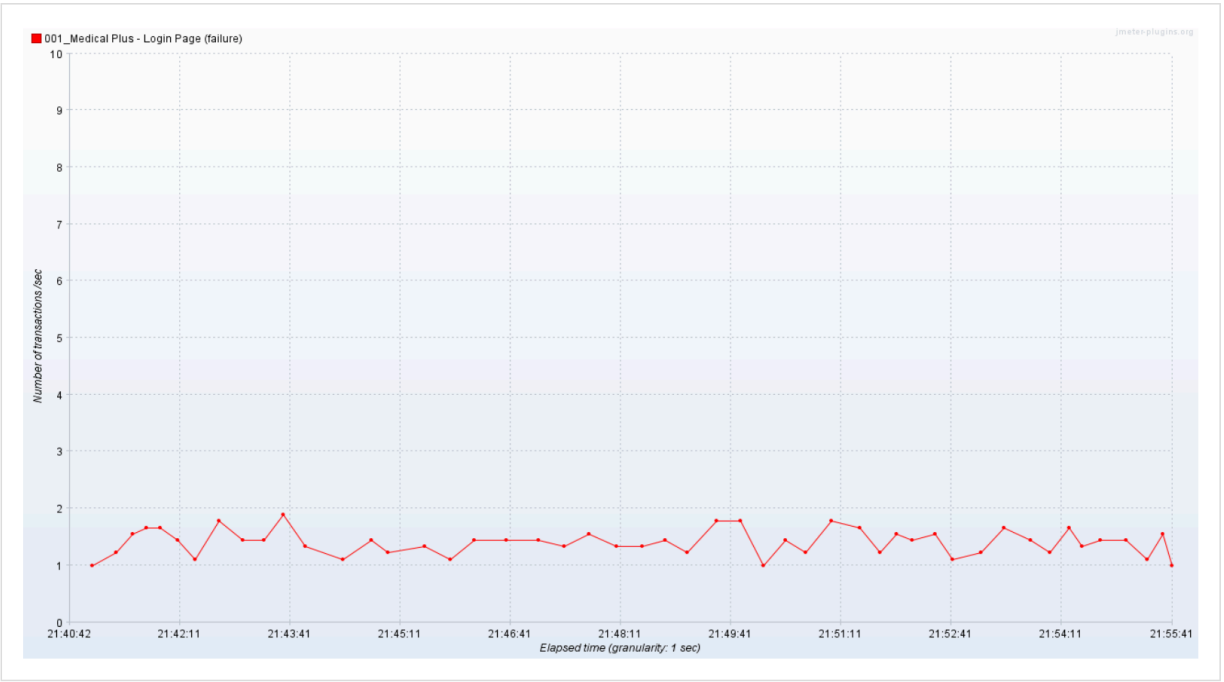


FIGURE 123

ResponseTimesOverTimeAggregate

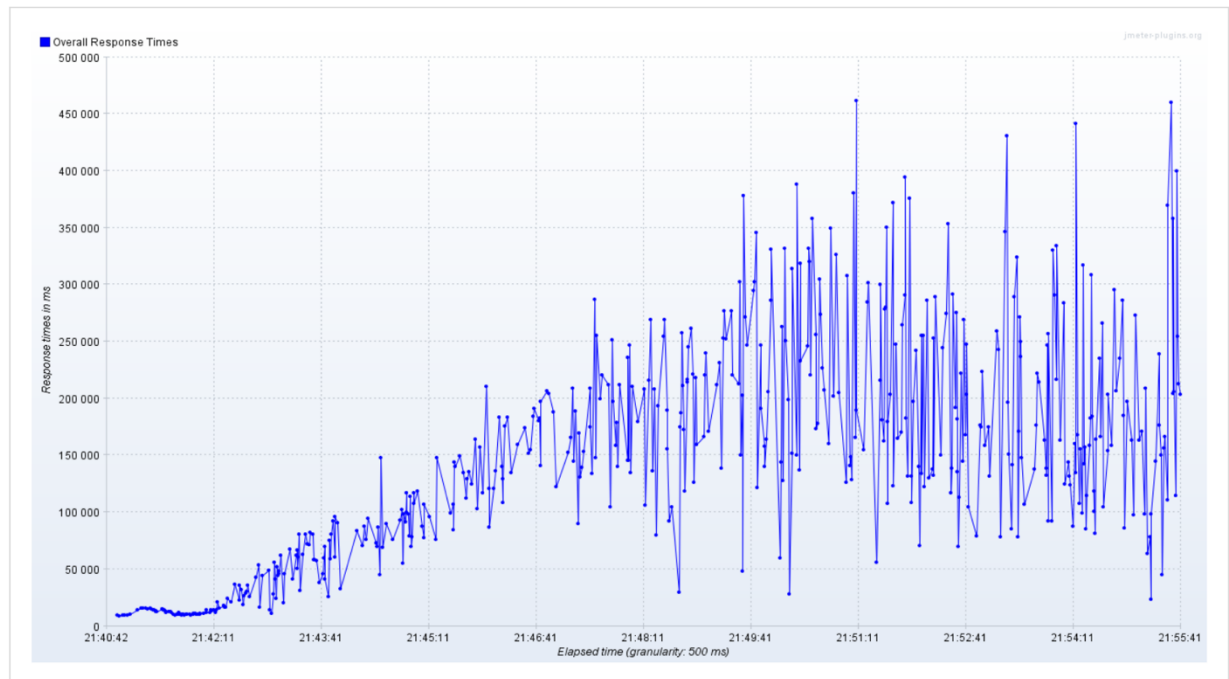


FIGURE 124

9. RISK MANAGEMENT AND HANDLING

- Dependency on other APIs: this risk was a great deal and would have had a massive effect on our system in case any of the APIs we used were terminated or not working for some reason. What we did to deal with this risk was switching the dependency into our hands by storing all the data we need into our own storage. So now we only fetch the data once from the PubMed API and after that we store into Elasticsearch on the Elasticsearch Cloud. So, now, whenever a user requests a specific article, we don't request the data from PubMed, no, we fetch it directly from our own database. By doing that we eliminate the risk of depending on third parties and at the same time delivering top notch articles that were published on highly ranked journals thanks to our filtering algorithm we developed. This method also improves the fetching time drastically.
- Scalability of millions of medical findings and documents: this risk was also a major concern when we were thinking about our project. The reason is because we needed to find a way that save us from losing everything we had in case something went wrong and a disk crashed or something. We also wanted to find a way that would be able to fetch data as fast as possible with accurate results. Since our dataset was ever increasing and stores millions of articles, this risk was a major concern. Thanks to the age we live in,

there is something called Elasticsearch that was built specifically to solve problems like our. Elasticsearch is database engine that is based on apache Lucene. By using Elasticsearch we were able to solve both problems in one go. The way we configured Elasticsearch was by having a load balancer, this load balancer directs requests to two shards that contain our data. One is the primary shard and the other is a replica shard. The primary shard serves both read and write requests while the replica shard serves only read requests. Using this architecture we are guaranteed a backup plan and a quick response to data requests. All of this is hosted on the Elastic Cloud which in return is based on AWS. Elasticsearch also offers a lot of features that help us visualize and manage our data in a neat and easy way.

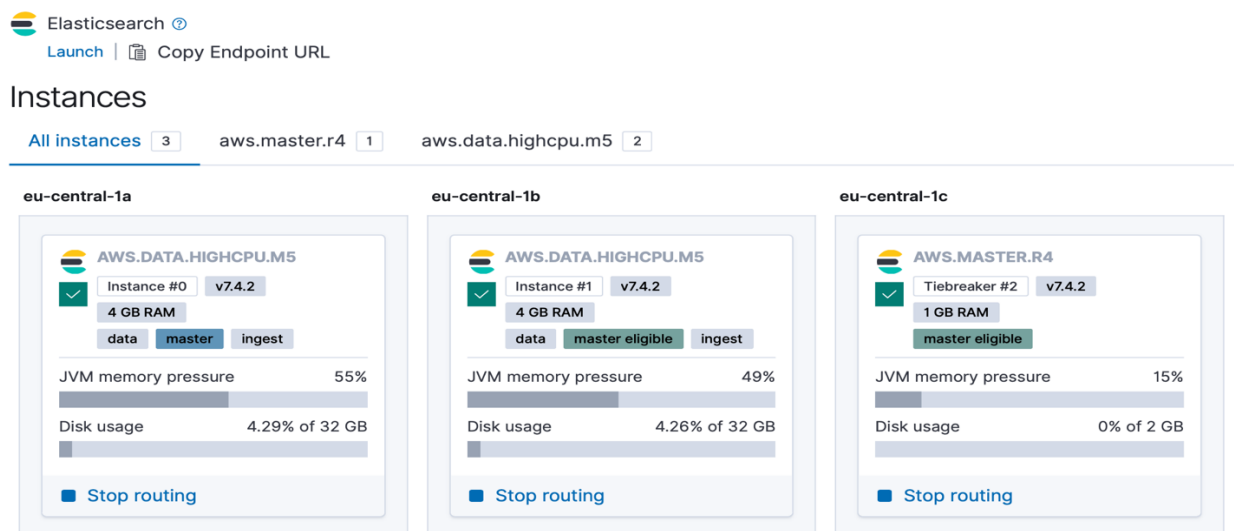


FIGURE 125

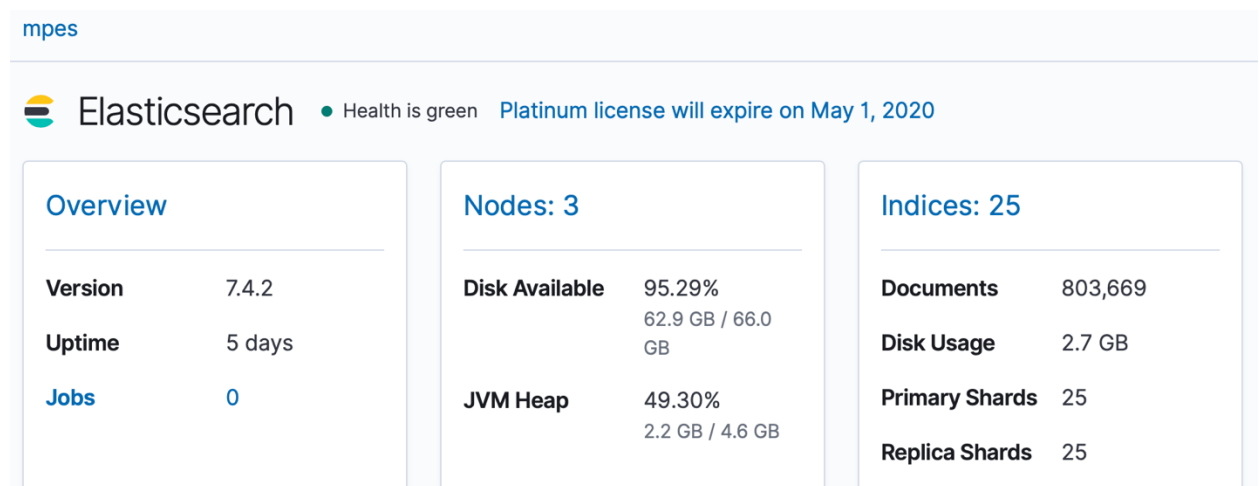


FIGURE 126

- Reliability of medical findings and discoveries: to tackle this risk we had to go through two steps. First, choose a reliable source to get our data from. Second, make sure that this

data is actually reliable and worth keeping. So, the source we used to gather our data was PubMed. PubMed is a medical library that is a branch of the National Institution of health which resides in the USA. Although PubMed is a medical database, but it also contains data from sciences that are partially related to medicine but not quite fit to fit our standards. It also contained articles from many journals that weren't very highly ranked so we had to find a way to differentiate between the good and the bad to only deliver quality content to our users. In order to accomplish this goal, we retrieved a list of over 5000 of the highest ranked journals from JCR and we started filtering our articles by comparing the journal that published the paper with the list of journals that we collected from JCR. We also made sure that each article contains all the data that would prove its authenticity like for example a "doi" that would map it to its original publication site. The filtration process is explained extensively in the Data Integration Approach.

10. LESSONS LEARNED

This section lists the lessons learned through the process of completing this project and going through this long, beneficial journey.

- **Git:** using git greatly increased our work performance and saved us a lot of effort and time. None of our team members have used git before, but surely after this project, no other future work will be done without it.
- **project management:** managing such a huge project was definitely one of the main lessons that we learned during this period.
- **teamwork:** having to work as a team was not as easy as it looked before we started this project. Surely when working with a team, you will sometimes reach some rough patches and conflicts along the road, but at the end you get up and keep working. You put in the time and effort needed to complete your task.
- **AWS:** AWS was definitely one of the most useful tools used in this project. It gave us functionalities that we didn't know were possible. Even though it was hard to learn and took a respectable amount of time, but it was a lesson worth learning.
- **Elasticsearch:** Elasticsearch and dealing with a NoSQL database was also a very important skill learned in this course.
- **data integration:** data integration was of course a necessity and had to be learned in the highest fashion in order to eliminate our risks and provide the services and qualities we promised.
- **following the code of ethics:** following the code of ethics was also an extremely important lesson learned through this course. Knowing your bounds and following the law and order.
- **time management:** this project is proof that with the right time management, anything is possible.
- **working under pressure:** to accomplish such a huge project in such little time is the definition of working under pressure and that's definitely what we did.

- **dealing with new technologies:** technology advances everyday, and so we need to advance with it. Most of the tools and technologies used in this project were initially new at the beginning of the project and harnessing this skill is definitely a must with the fast paced world we live in now.

11. CONCLUSION

In conclusion, our system strives to solve a major problem in today's medical society. The medical field is and will always be a very sensitive field. one mistake there is a hundred mistakes in any other field. That's why our system is trying to make something different. There are other websites and platforms out there that publish articles and papers in the medical field, but for some reason the problem still exists in an unacceptable quantity. That's why we tried to think in a different way. We try to think of our system as a community for the doctors in the medical field. We built this platform specifically for them in an attempt to reduce medical mistakes due to the lack of knowledge. We tried to give them the freedom to express their opinion and input without any complication. We offered them the motivation that would help them get ahead of this problem and would lead them to the right path. Technology advances everyday, and so they should advance with it. With the current technology, the discoveries never stop. At Medical-Plus, we try to share the most relevant and accurate data to each doctor's interest and profession. After this long journey in completing this project, we hope that it accomplishes the goal and change we desire, and would lead to a better and safer tomorrow.

REFERENCES

- [1] https://en.wikipedia.org/wiki/List_of_withdrawn_drugs
- [2] Kazdin, A. E., & Kendall, P. C. (1998). Current progress and future plans for developing effective treatments: Comments and perspectives. *Journal of clinical child psychology*, 27(2), 217-226.
- [3] Lessenger, J. E., & Feinberg, S. D. (2008). Abuse of prescription and over-the-counter medications. *J Am Board Fam Med*, 21(1), 45-54.
- [4] Hammes, B. J., Rooney, B. L., & Gundrum, J. D. (2010). A comparative, retrospective, observational study of the prevalence, availability, and specificity of advance care plans in a county that implemented a n advance care planning microsystem. *Journal of the American Geriatric s Society*, 58 (7) , 1 2 4 9 - 1 2 5 5 .
- [5] Lineberry, T. W., & Bostwick, J. M. (2004, August). Taking the physician out of "physician shopping": a case series of clinical problems associated with Internet purchases of medication. In *Mayo Clinic Proceedings* (Vol. 79, No. 8, pp. 1031-1034). Elsevier.
- [6] Shiyanbola, O. O., Farris, K. B., & Chrischilles, E. (2013). Concern beliefs in medications: Changes over time and medication use factors related to a change in beliefs. *Research in Social and Administrative Pharmacy*, 9(4), 446-457.
- [7] Johnson, D. A., Austin, D. L., & Thompson, J. N. (2005). Role of state medical boards in continuing medical education. *Journal of Continuing Education in the Health Professions*, 25(3), 183-189.
- [8] Davis, D. A., Thomson, M. A., Oxman, A. D., & Haynes, R. B. (1995). Changing physician performance: a systematic review of the effect of continuing medical education strategies. *Jama*, 274(9), 700-705.
- [9] Lehmann, U., Dieleman, M., & Martineau, T. (2008). Staffing remote rural areas in middle-and low-income countries: a literature review of attraction and retention. *BMC health services research*, 8(1), 19.
- [10] Ryu, S., Ho, S. H., & Han, I. (2003). Knowledge sharing behavior of physicians in hospitals. *Expert Systems with applications*, 25(1), 113-122.
- [11] <https://www.codingdojo.com/blog/choosing-python-web-frameworks>