MINISTRY OF EDUCATION AND TRAINING

TRƯỜNG ĐẠI HỌC FPT

FPT UNIVERSITY

Capstone Project Document

Career Trend Suggestions

Group 1	
Group members	Thái Lý Anh Khuê – Team leader – SE62541 Tăng Hồ Duy Minh – Team member – SE62380 Đàm Phước Đức Duy – Team member – SE62412 Nguyễn Viết Hải – Team member – SE62107
Supervisor	Mr. Kiều Trọng Khánh
Ext. Supervisor	N/A
Capstone Project code	CTSA

-Ho Chi Minh City, **10/09/2018-**

This page is intentionally left blank

Table of Contents

List of Figures

List of Tables

This page is intentionally left blank

A. Introduction

1. Project Information

- Project name: **Career Trend Suggestions**
- Project Code: **CTSA**
- Product Type: **Web Services, Web Application**
- Start Date: **September 10th, 2018**
- End Date: **December 21st, 2018**

2. Introduction

In this document, we introduce a solution for suggesting trending careers in Vietnam. Specifically, this document focuses on IT positions and specific related skills for pupils and universities' freshmen in Vietnam.

Many pupils in Vietnam have some problems in deciding which specific majors should they study in college in order to apply for a job in the future; for freshmen, they find it confused choosing what career path to focus on. Based on our researches and analysis, we propose a solution for suggesting trending careers using the case study of IT careers or/and another one.

We build a system, which helps pupils and freshmen reduce their confusion of choosing from hundred majors and skills to learn, by which, helps them to have a clear look at their future career paths. In the process of analysis, we believe using predictive analytic is capable to resolve the problem by collecting historical data from recruitment websites to identify the likelihood of future trending careers. Besides of that, we also provide an information system as a third-party service for companies to post and manage their recruitments.

3. Current Situation

When searching for majors or things to learn in order to make preparation for future careers, pupils and freshman usually go to recruitment websites and look for positions with required skills that are attached in the posts. They may choose to focus on studying and preparing the skills that seem to be widely needed at the moment. When they are ready to apply for that specific positions, there are risks that those positions are no longer necessary since there were too much appliances before they apply.

4. Problem Definition

Below are disadvantages of current situation:

- **Freshmen' lack of experience:** In many universities in Vietnam, freshmen are not even taught subjects related to their majors yet, so they decide to search and study by themselves. Even they maybe lack of basic knowledges in their business fields, pupils and freshmen are young and too excited to study majors and skills which seem to be trending that the time without thinking about whether or not those are still needed when they are ready to apply for the jobs.
- **Information are too popular and diverse:** Recruitments are public in many recruitment websites. With a lot of new trending jobs and related skills are published almost daily basis, the information in the job market are getting too diverse and may confuse young people, especially pupils and freshmen, who are lack of knowledge and experience. They may choose to study majors and focus on skills which may be not related to others, without knowing that they are taking super risks.
- **Total of students, specifically in IT, increase every year:** The human resource market for IT positions are getting more and more concerned every year. With a lot of pupils signing up for IT majors in universities and thousands of IT students that are getting graduated each year, the competition is getting more and more acrimonious.

5. Proposed Solution

Our proposed solution is to build a career trend suggestions system named "CTSA system" to resolve the current situations by analyzing historical data and suggesting careers along with skills that may be trending in the future. This can help pupils and freshmen reduce their confusion while choosing what majors or skills to learn and focus on for their future career paths. We design the system to be open, flexible and scalable so as to deploy this system with career trends from multiple business fields in future plan.

5.1. Feature functions

- Web Services:
    - **Careers suggestions:** Pupils are able to receive suggestions about careers which they are suitable. Suitability depends on their academic results in comparison with samples. Samples are information collected from surveyed students in specific majors.

- **Career trend suggestions:** Career trend is updated whenever new related official information is published. Information will be collected and added manually for analyzing. Trend will be forecasted and produced based on analysis methods. Suggestions are made based on forecast trends and published as web services.
- **IT career trend prediction:** IT career trend is updated periodically. Data will be collected daily to analyze and trend suggestions will be produced and showed on the web application. Analyzed predictions are published as web services and be ready for other applications and devices to consume.
- **Companies recruitment information:** Companies that are defined as partners can update their recruiting positions and information. Their provided information will also be collected for analyzing.

– Web Applications:

- **CTSA Web Application:** Web application provides an easy-to-use interface for end users, such as pupils and students. These end users are able to receive suggestions for their future career paths through the application.
- **CTSA Content Management System (CMS):** CTSA CMS provides a clear user interface for authorized users to configure and make adjustments to contents presented on the Web Application.

5.2. Benefits and Trade-offs

– ***Benefits:***

- Users:
  - Users can have a clear view of trending careers with related skills so as to focus on improving specific skills which are suitable for the jobs.
  - Facebook users can quickly chat with the chatbot and get the suggestions that may helpful for their career paths.
- Partners:
  - Partners can help improving the accuracy of input data and also got opportunities for getting appliances from people who use the application for searching careers.

– ***Trade-offs:***

- System:

        - Collected data from recruitment websites are very diverse, headers of the posts have multiple lexical patterns which creates difficulties in cleaning the input data.
        - Information about students' academic results have to be collected through surveys, which may take up a lot of time.
        - Data are collected daily, in a short term – about 3 months, which may not be able to create good enough data granularity for analysis.
        - Prediction using historical data simply based on recruitment situations, meanwhile many other factors can also affect the career trend, such as: environment, politics, economy, etc.
- Users:
    - Suggestions may contain uncertainty.
    - Trends may be shown differently due to different periods, which can make users find it hard to focus on which career path.
- Partners:
    - Recruitments have to be handled manually and should follow the pre-defined template.

6. Functional Requirements

Function requirements of the system are listed as below:

- ***Authentication and Authorization:***
    - Authenticate and authorize users to use configuration and management functions.
- ***Data Scraper:***
    - Collect and clean data from chosen recruitment websites automatically.
- ***Data Analytic:***
    - Automatically analyze collected data by applying prediction algorithms.
- ***Guest:***
    - Get career suggestions.
    - Get career trends.
- ***Admin:***
    - Manage staff.
- ***Staff:***
    - Manage partners.

- Configure parameters for analyzing.

- ***Partner:***
  - Create recruitment.
  - Publish recruitment.
  - Expire recruitment.
  - Manage recruitment's information.

7. Role and Responsibility

No	Full Name	Role	Position	Contact
1	Kiều Trọng Khánh	Project Owner	Advisor	khanhkt@fpt.edu.vn
2	Thái Lý Anh Khuê	Scrum Master	Leader	khuetlase62541@fpt.edu.vn
3	Tăng Hồ Duy Minh	Developer	Member	minhthdse62380@fpt.edu.vn
4	Đàm Phước Đức Duy	Developer	Member	duydpdse62412@fpt.edu.vn
~~5~~	~~Nguyễn Viết Hải~~	~~Developer~~	~~Member~~	~~hainvse62107@fpt.edu.vn~~

Table 1: Roles and Responsibilities

This page is intentionally left blank

B. Software Project Management Plan

1. Problem Definition

1.1. Name of this Capstone Project

- Official name: Career Trend Suggestions
- Vietnamese name: Tư vấn chọn ngành nghề dựa trên phân tích số liệu
- Abbreviation: CTSA

1.2. Problem Abstract

We need to build this system to be flexible and scalable enough to cope with multiple business fields. Using the traditional monolithic approach for software development on this system can cause considerable problems when the number of business fields increase by time, with many other parameters to be analyzed.

Analyzing data and making suggestions require knowledge about Machine Learning, however, the whole team have never learned or had experience with this field. Besides, data about careers can be analyze referred to multiple different parameters, by different business fields, which is also needed to be highly concerned.

Historical data to be analyzed are not available, the team have to retrieve raw data from recruitment websites. Since the retrieved data are not always clean enough, it is the team's responsibility to develop different techniques to perform data cleansing.

1.3. Project Overview

1.3.1. Current Situation

Strengths:

- **Knowledge on software development**: The whole team have experience in software development. All members are familiar with the same server-side programming language, three out of four members have experience about the same framework.
- **Having experience working with distributed systems**: Two out of four members have worked with distributed systems and have more or less knowledge about improving the system's scalability. One member has experience in inter-communicating between modules in distributed systems.
- **Retrieving data**: All members have experience in retrieving data from websites and developed techniques to standardize data.

Weaknesses:

- **Data dependence**: Historical data are not officially public anywhere. Therefore, data are retrieved from many different sources, which can lead to inconsistency.
- **Lack of Machine Learning knowledge**: None of the team members have learned Machine Learning or any related techniques. Researches also showed that no current machine learning library can help dealing with the suggesting problem clearly.

Opportunities:

- **Learning Machine Learning**: The team have an opportunity Machine Learning and immediately apply knowledge on the system's specific use cases.
- **Improving data cleansing techniques**: Data are collected from different sources so that it requires many different cleansing techniques. The team have a great chance to improve, develop and perform different techniques for filtering data.

Challenges:

- **Distributed, flexible and scalable system**: The system needs to be open, flexible and scalable so as to integrate many other business fields in the future.
- **Multiple synonyms from different languages**: When retrieving data from recruitment websites, there are many synonyms from both English and Vietnamese, for example: "Developer", "Dev" or "Lập trình viên" can be supposed to be the same.

1.3.2. The Proposed System

According to previous researches, we choose to develop the system using the Microservice Architecture in replacement for the traditional monolithic approach. Applying Microservice Architecture can help us developing the system as a suite of small services, where each of them provides a firm module boundary. Using Microservices not only help breaking down the system into small modules for easier management but also help the service' s owner totally concentrates on the business capabilities. With Microservices, the system can

be elastic and always ready for expansion in case there are more business fields to be analyzed.

For analyzing and suggesting, through machine learning, we decide to use supervised learning, specifically, linear regression for making suggestions using historical data. This is the simplest solution so far for the team's current experience on machine learning.

In order to solve the multiple synonyms problem, we decide to develop a technique combining inverted index database and common relational database. All of the synonyms will be collected, refined and managed manually. Inverted index database is later used to extract all of the collected synonyms from a random text content.

Our system includes two subsystems:

- ***Web Services Provider*** works on collecting, cleaning and analyzing the data; later, information or suggestions will be provided to be consumed by web applications or mobile devices.
- ***Web Application*** for people to visit and see the career trend. Administrators and staff can use the Web Administration page to configure information or parameters for the system to retrieve data from sources.

1.3.2.1. Web Services Provider

Web services provider is the core of the system, which provides all of the necessary web services that will be consumed by web applications or any mobile device.

1.3.2.2. Web Application

Main web application is common place for users to visit and see how the career trend is suggested. This is also a place for administrators and staff to configure parameters for retrieving and analyzing data from sources.

- *For administrators:*
  - Manage accounts for staff.
- *For staff:*
  - Manage accounts for partners.
  - Define template for recruitment resources from partners.
  - Configure parameters for data analyzing.

        - Configure sources to retrieve data from websites.
        - Collect data from specific websites.
    - *For partners:*
        - Manage recruitment.
    - *For users:*
        - Get career trends.
        - Search for recruitment.
        - Get suggested recruitment and related skills.

1.3.3. Boundaries of the System

The team provides this system as a business product and we operate it as the owner. We play the role of the administrators and staff, where we can have access to all of the configurations for data analyzing and creating reports.

The system is intended for users such as pupils, freshmen need to know about the possibilities of how the career trends are going to be. For the time being, the system will focus on IT careers and related skills as the use case.

The language of the system that is provided for users is Vietnamese, meanwhile, the language for administration page is English.

The complete product includes:

- A web application for users to visit and get a possibility of career trends.
- A content management system for administrators and staff to configure parameters, as well as manage other official information, that is necessary for analyzing.

1.3.4. Future Plans

Current system only supports career that is IT-related. Processing synonyms for data cleansing is done manually. The chatbot's natural language processing training is also done manually. Therefore, we look forward to design the system to be more diverse in business fields and more intelligent.

- **Multiple business fields:** The system will cover much more business fields alongside IT.
- **Automatic learning:** Apply machine learning to make the data cleansing process can learn synonyms and filter synonyms automatically.

- **Applying payment method for setting priorities of partners' recruitment:** Partners who are eager to find appliances for their recruitment can pay for the system so that their recruitment will be highlighted and set on top of the suggested recruitment.

1.3.5. Development Environment

1.3.5.1. Hardware requirements

- For Server Web:

	Minimum Requirement	Recommended
Internet Connection	Cable, WiFi (4Mbps)	Cable, WiFi (8Mbps)
Computer Processor	Intel® Core™ i3 (2.50GHz)	Intel® Core™ i7 or faster
Operating System	Window 7	Ubuntu 18.04 LTS
Computer Memory	8GB Ram	16GB Ram

Table 2: Hardware Requirement for Server Web

1.3.5.2. Software requirements

Software	Name / Version	Description
Operating system	- Ubuntu 18.04 LTS - Windows 10 build 1803	Operating system and platform for development
Environment	- Java 8 - Spring Boot 2.0	Specification for developing web services and web application
IDE	IntelliJ IDEA 2018	Programming tool
Database Management System	MySQL 5.7 Elasticsearch 6.5	Tools used to create & manage the database for system
Source control	GitHub 2.14	Tools used for source control
Web browser	Chrome 70	Browser used in testing

Table 3: Software Requirements for Development

2. Project Organization

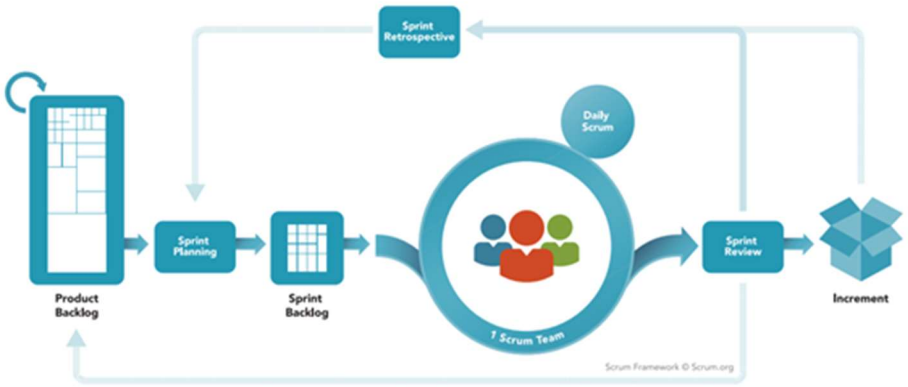
2.1. Software Process Model

As mentioned above, the team have to deal with the following situations:

- Historical data is supposed to has to be collected retrieved from many different sources, which can lead to inconsistency. Data need to be continuously collected and cleaned with respect to predefined parameters. Therefore, the team have to concentrate together on defining the necessary parameters for data cleansing.
- Machine learning and client-side technologies are new to the team, which makes it necessary or the whole team to stay together to learn and build the system effectively.
- Requirements from Product Owner can be changed base on the performance of the team. Therefore, the team must be prepared for any possible changes.

For the above-mentioned reasons, the team decided to develop this project following Scrum model, which will help us:

- Deal with possible changes in requirements.
- Have a specific duration to focus together on defining parameters and cleansing techniques.
- Continuously work on the cycle of Researching - Applying - Testing when facing machine learning and client-side technologies.
- Focus on code rather than design and documentation.



.

Figure 1: Scrum Model

https://www.scrum.org/resources/what-is-scrum

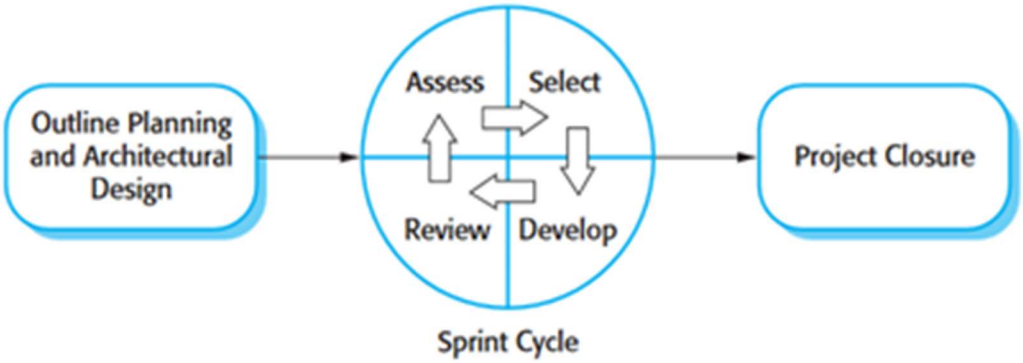


Figure 2: Scrum Process

Software Engineering 9th Edition, Ian Sommerville
(Chapter 3: Agile Software Development, page 63)

2.2. Roles and Responsibilities

No	Full Name	Role	Position	Responsibilities
1	Kiều Trọng Khánh	Project Owner	Advisor	– Specify user requirements – Control the development process – Support in technique and business logic
2	Thái Lý Anh Khuê	Scrum Master	Leader	– Manage process – Planning Scrum implementations – Create system architecture design – Support technique – Arrange daily meeting – Track backlog – Create product Increment – Create test plan – Configure development environment – Code – Test – Write documents and reports – Designing database
3	Tăng Hồ Duy Minh	Scrum Team Member	Member	– Clarify requirements – Commit individual process on time – Support other members to complete team work – Prepare the information for data cleansing – Code – Write documents and reports

4	Đàm Phước Đức Duy	Scrum Team Member	Member	– Clarify requirements – Commit individual process on time – Support other members to complete team work – Prepare the information for data cleansing – Code – Write documents and reports
~~5~~	~~Nguyễn Viết Hải~~	~~Scrum Team Member~~	~~Member~~	

Table 4: Roles and Responsibilities Details

2.3. Tools and Techniques

Tools		
Developing tool	Web Services Provider	IntelliJ IDEA 2018
	Web Application	IntelliJ IDEA 2018, Visual Studio Code 2018
Managing Database	MySQL Workbench 8.0, DataGrip 2018	
Source control	GitHub 2.14	

Table 5: Tools

Techniques		
Developing tool	Front-end	HTML 5, CSS 3, JavaScript ES6, jQuery 3.1, Bootstrap 4.1
	Back-end	JDK 1.8, Spring Boot 2.0, Spring Framework 5.0
Web Servers		Apache Tomcat 8, Netty 4
Database Management System		MySQL 5.7, Redis 5.0, Elasticsearch 6.4

Table 6: Techniques

3. Project Manage Plan

3.1. Product Backlog

ID	Feature	User story	Sprint
1	Collect data from recruitment website	As a Timer, it can collect data from recruitment websites for later analysis purpose.	1
2	Clean collected data	As a Timer, it can clean collected data into a predefined data structure which is suitable for analysis.	1
3	Create career suggestions	As a Timer, it can create suggestions about careers based on user's studying results inputs.	2
4	Create career trends prediction	As a Timer, it can create prediction about careers' trends based on historical data.	3
5	Synchronize data for extraction	As a Staff, i can synchronize new data to the extractor after verification, which allows the timer to do better in the cleansing process.	4
6	Define parameters for analyzing	As a Staff, i can define parameters for the analyzers to use in order to create suggestions or predictions.	4
7	Manage careers' historical data	As a Staff, i can manage historical data about careers in order to enrich data and help verify the sources of data.	5
8	Manage partner's recruitment	As a Partner, I can manage their recruitment, which helps verify and enrich the data for analysis.	5

Table 7: Product Backlog

3.3. All Meeting Minutes

Refer to “Meeting Minutes” folder.

4. Coding Convention

This project is applied some coding convention rules as listed below:

Java: Using to develop web application and web service.

- Naming Convention:
  - Follows camelcase syntax for naming the class, interface, method and variable.
  - If name is combined with two words, second word will start with uppercase letter always, e.g: actionPerformed(), firstName, ActionEvent, ActionListener etc.
  - Method names should be verbs.
  - Class names should be nouns, in mixed case with first letter of each internal word capitalized.

- Constant names should be all uppercase with words separated by underscore.
- Comment:
    - Using /* */ for block comments.
    - Using // for line comments.
- Follow Field Naming Conventions
- Other fields start with a lowercase letter.
- Public static final fields (constants) are ALL_CAPS_WITH_UNDERSCORES.
- Using Java coding convention from:
    - http://www.oracle.com/technetwork/java/codeconvtoc-136057.html

This page is intentionally left blank

C. Software Requirement Specification

1. User Requirement Specification

1.1. Guest Requirement

Guest is a person who can access to the system through web applications or mobile devices which consume the web services. Guest can use most of the functions in the system. Guest should have these following abilities:

- Get suggested career trends
- Search for trends of specific careers
- Get the recruitment

1.2. Administrator Requirement

Administrator can manage staff's accounts and policies of the system. Administrator should have the following functions:

- Provide accounts for staff

1.3. Staff Requirement

Staff is responsible for the partners' accounts, defining recruitment template and configuration for data analyzing. Staff should have the following functions:

- Provide accounts for partners
- Configure parameters for data analyzing
- Update new positions and skills for collecting data
- Update synonyms for data cleansing

1.4. Timer Requirement

System is also an actor, which is named Timer, runs in the background to keep the system working. Timer should have the following functions:

- Collect data from websites automatically
- Clean data for matching analyzing' requests
- Synchronize synonyms and keywords between database and Elasticsearch

2. System Requirement Specification

2.1. External user requirement

2.1.1. User Interface

The web application user interface will be in Vietnamese and provide the following components:

- Intuitive navigation functions.
- User registration and user login page.
- A line chart for presenting the suggested trends.
- Plain text field and button to search for recruitments.
- An area for presenting recruitments.
- Partner's information page.

The web administration user interface will be in English and contain the following components:

- Pages for partners to edit information and manage recruitments.
- Pages for staff to manage partners and their information.
- Intuitive data tables for staff to manage positions, skills and synonyms.
- Pages for administrator to manage staff.

2.1.2. Hardware Interface:

- N/A

2.1.3. Software Interface

- The web services are provided in JSON format.
- Web application works with Google Chrome (version 69) and Firefox (version 52).

2.1.4 Communication Protocol

- Use HTTP protocol (version 1.1) for communication between web services and web application or mobile device application that consumes the services.

2.2 System Overview User Case[1]

[1]OMG UML (August 2018), OMG Unified Modeling Language ™ (OMG UML) Superstructure

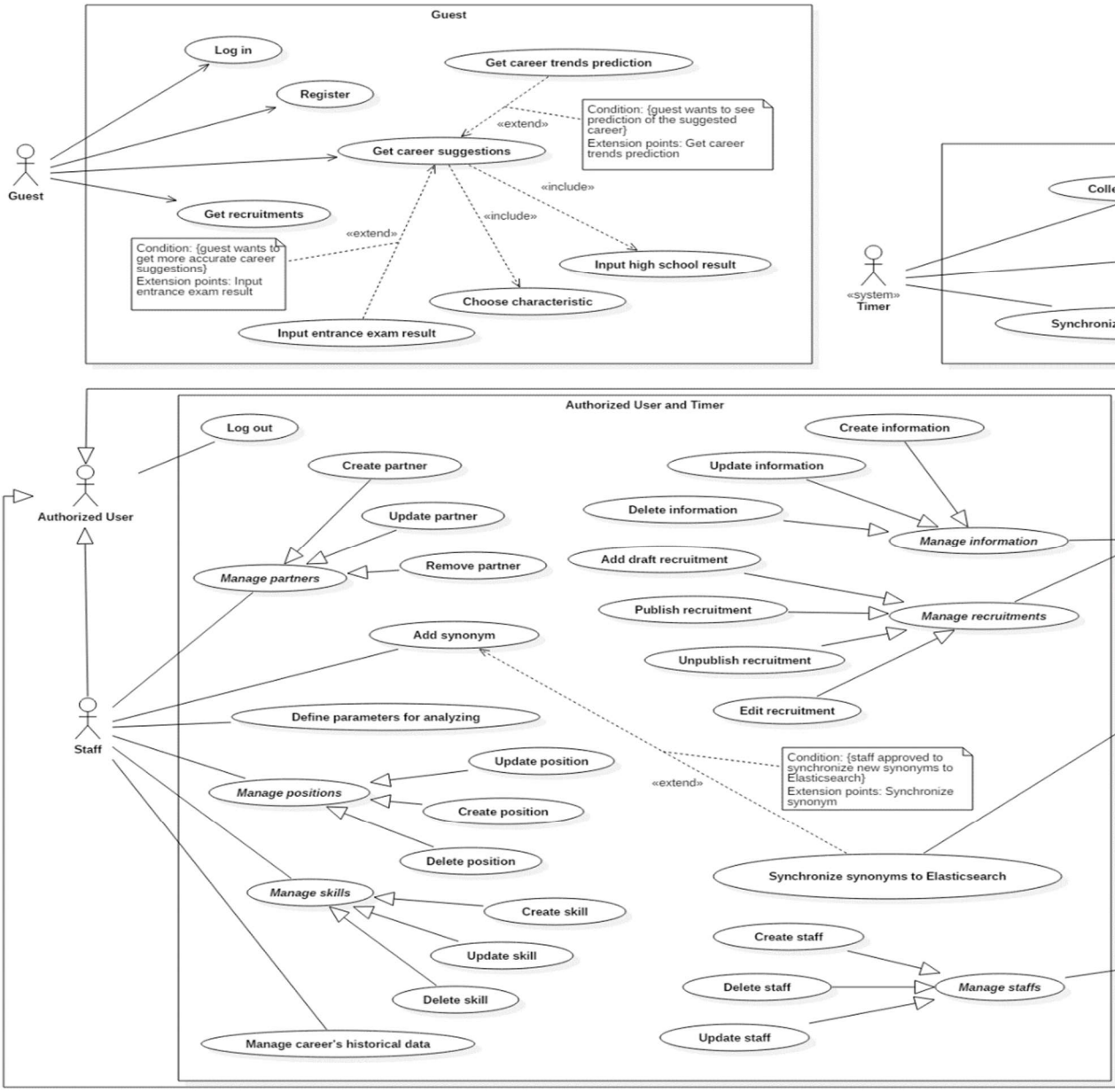


Figure 3: Overview Use Case

No	Actor	Use Case	Description	Note
1	Timer	Collect data from recruitment website	Timer collects data from recruitment websites for later analysis purpose.	
2	Timer	Clean collected data	Timer clean collected data into a predefine data structure which is suitable for analysis.	
3	Timer	Create career suggestions	Timer create suggestions about careers based on user's studying results inputs.	
4	Timer	Create career trends prediction	Timer create prediction about careers' trends based on historical data.	
5	Staff	Synchronize data for extraction	Staff synchronize new data to the extractor after verification, which allows the timer to do better in the cleansing process.	
6	Staff	Define parameters for analyzing	Staff can define parameters for the analyzers to use in order to create suggestions or predictions.	
7	Staff	Manage careers' historical data	Staff can manage historical data about careers in order to enrich data and help verify the sources of data.	
8	Partner	Manage partner's recruitment	Partner can manage their recruitment, which helps verify and enrich the data for analysis.	

Table 8: Description of use cases

No	Actor	Description	Note
1	Timer	The actor that is responsible for executing use cases that require automation.	
2	Staff	The actor that is responsible for executing use cases related to the system's configurations for analyzing.	
3	Guest	The actor that is responsible for executing most of the main use cases related to career trend suggestion and prediction.	
4	Administrator	The actor that is responsible for executing use cases related to managing the staff's accounts.	
5	Partner	The actor that is responsible for executing use cases related to managing accounts and recruitment of the company which registered to the system.	

Table 9: Description of Actors

2.3. List of Use Case

2.3.1. <Timer> Overview use case

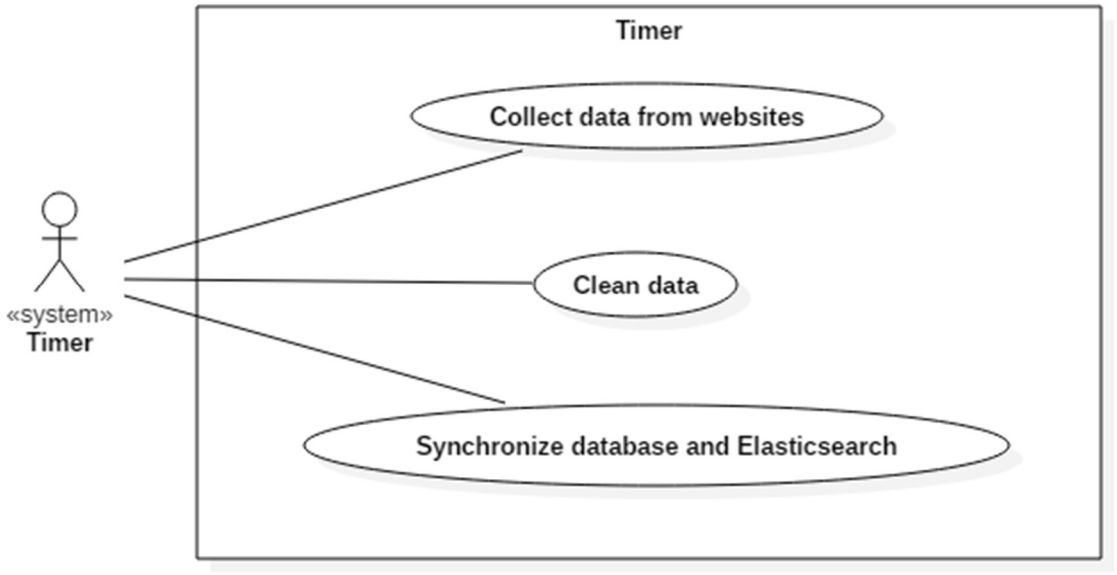


Figure 4: <Timer> Overview use case

2.3.1.1. <Timer> Collect data from websites use case (T01)

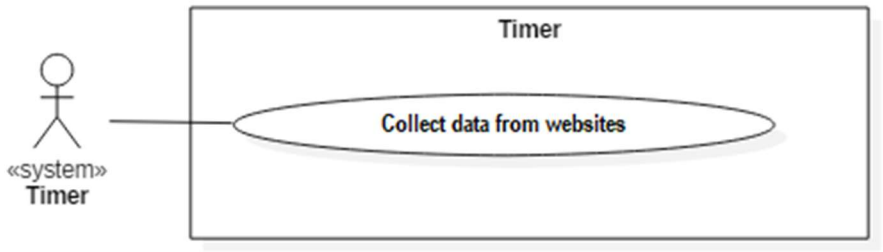


Figure 5: <Timer> Collect data from websites use case

USE CASE - T01			
Use Case No.	T01	**Use Case Version**	2.0
Use Case Name	Collect data from websites		
Author	Đàm Phước Đức Duy		
Date	23/09/2018	**Priority**	High

Actor:

Timer

Summary:

This use case describes how the system collect sources of data.

Goal:

To prepare sources of data for later collecting and cleaning purposes.

Triggers:

System runs timer task that trigger data scraping services.

Preconditions: N/A

Post Conditions:

- **Success:** Display status of each recruitment and complete message in system log.
- **Fail:** Error is tracked in a log file.

Main Success Scenario:

Step	Actor Action	System Response
1	System runs timer task to collect data from recruitment websites. [Alternative 1]	List of sources to the recruitment from which data need to be collected.

Alternative Scenarios:

No	Actor Action	System Response
1	Timer found existed recruitment's source in storage.	– Skip current source. – Display in system log: "Skipped: <source of the existed recruitment>".

Exceptions: N/A

Relationships: N/A

Business Rules:

- Timer automatically executes collecting task:
  - Periodically.
  - 24 hours after the last successful execution.
- Timer accesses to pre-selected recruitment websites and collect the web addresses of recruitment:
  - Addresses that existed in the system will be skipped.

Table 7: <Timer> Collect data from websites use case specification table

2.3.1.2. <Timer> Clean data use case (T02)

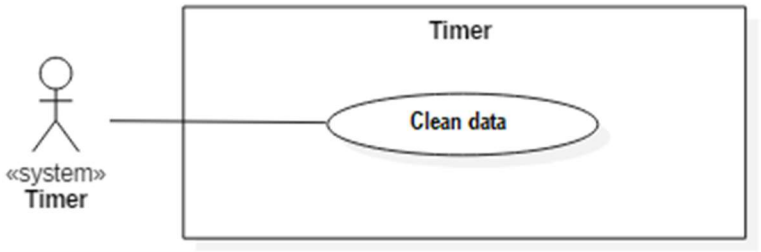


Figure 6: <Timer> Clean data use case

USE CASE - T02			
Use Case No.	T02	**Use Case Version**	2.0
Use Case Name	Clean data		
Author	Đàm Phước Đức Duy		
Date	23/09/2018	**Priority**	High

Actor:

Timer

Summary:

This use case describes how the system process data collected from recruitment.

Goal:

- To retrieve necessary information from recruitment.
- To prepare information for providing career suggestions.

Triggers:

System runs timer task that trigger data cleaning services.

Preconditions:

- List of unique recruitment sources are collected.
- Majors, positions and skills that needed to be collected are predefined.

Post Conditions:

- **Success:** Display success message in system log.
- **Fail:** Error is tracked in log file.

Main Success Scenario:

Step	Actor Action	System Response
1	Timer tries to retrieve list of unique recruitment sources that are not yet processed.	Display in log file: “Retrieving un-extracted sources.”. [Alternative 1]
2	Timer extracts necessary information from each recruitment.	Display extracted information in log file. [Exception 1]
3	Timer inserts recruitment’s data into database after cleaning.	– Cleaned data is inserted into database. – Display success message in system log.

Alternative Scenarios:

No	Actor Action	System Response
1	Timer tries to retrieve an empty list of unique recruitment sources that are not yet processed.	– Skip the cleaning process. – Display in system log: “All sources are processed.”.

Exceptions:

No	Cause	System Response
1	Extracted positions or skills are not available in the system’s storage.	– Skip current source. – Error details will be tracked in a log file.

Relationships: N/A

Business Rules:

- Timer accesses each source of recruitment from collected list in order to extract necessary information. Extracted information from a recruitment includes:
    - Recruited position
    - Start date
    - Due date
    - Required skills
    - Number of needed employees
- Timer extracts positions and skills based on data defined by staff.
- Collected recruitment must satisfy the following conditions:
    - Extracted position must not be empty.
    - At least 1 skill is available.

Table 7: <Timer> Clean data use case specification table

2.3.1.3. <Timer> Synchronize database and Elasticsearch (T03)

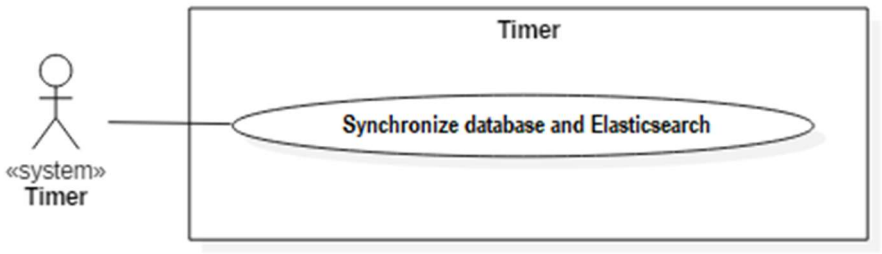


Figure 7: <Timer> Synchronize database and Elasticsearch use case

USE CASE - T03			
Use Case No.	T03	**Use Case Version**	2.0
Use Case Name	Synchronize database and Elasticsearch		
Author	Tăng Hồ Duy Minh		
Date	23/09/2018	**Priority**	High

Actor:

Timer

Summary:

This use case describes how the system insert lately added keywords to Elasticsearch's storage.

Goal:

- To synchronize data between system's storage and Elasticsearch's storage.
- To improve the accuracy of keyword extraction service.

Triggers:

System runs timer task that trigger keyword synchronization service.

Preconditions:

- Keywords are available in system.
- Keywords are set with status "Unpushed".

Post Conditions:

- **Success:** Display success message in system log.
- **Fail:** Error is tracked in log file.

Main Success Scenario:

Step	Actor Action	System Response
1	Timer tries to retrieve list of "Unpushed" keywords from system database.	List of "Unpushed" keywords. [Alternative 1]
2	Timer inserts the retrieved list of keywords to Elasticsearch's database.	Display information in system log and log file.
3	Timer set the processed keywords to status "Pushed".	Display success message in system log.

Alternative Scenarios:

No	Actor Action	System Response
1	Timer tries to retrieve an empty list of "Unpushed" keywords.	– Skip the process. – Display in system log: "No keywords to be pushed.".

Exceptions: N/A

Relationships: N/A

Business Rules:

- Timer automatically executes synchronizing task:
  - Periodically.
  - 24 hours after the last successful execution.
- Timer retrieves the keywords that haven't been pushed to Elasticsearch and insert them into the Elasticsearch's storage.

Table 8: <Timer> Synchronize database and Elasticsearch specification table

2.3.2. <Guest> Overview use case

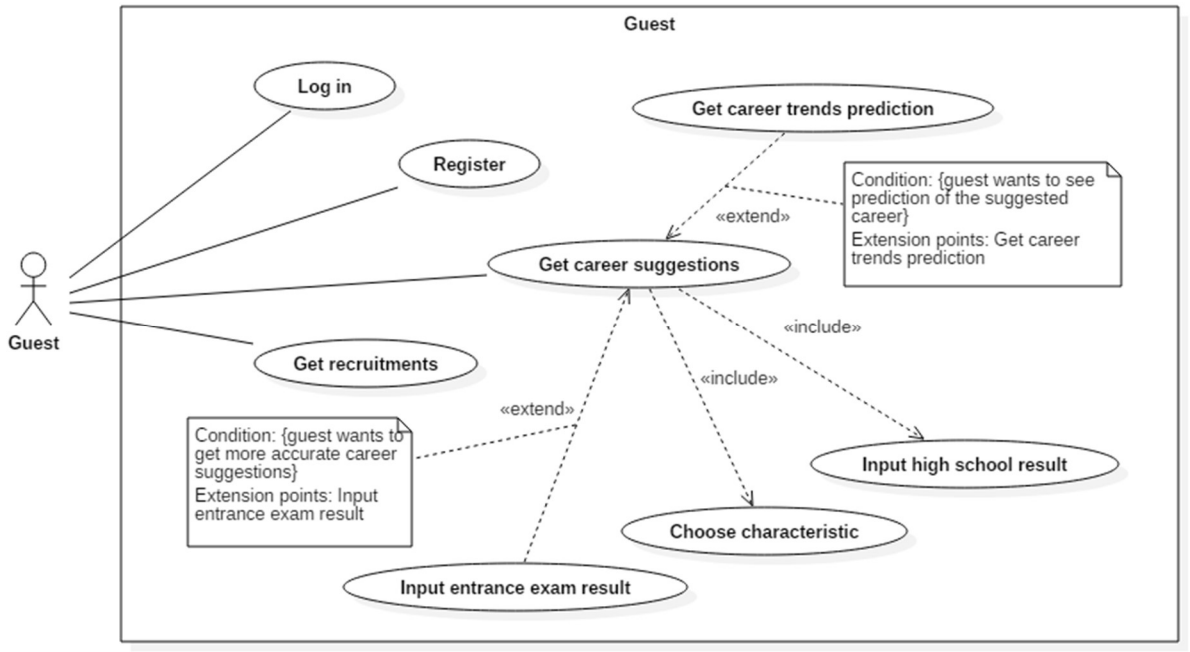


Figure 8: <Guest> Overview use case

2.3.2.1. <Guest> Get career suggestions (G01)

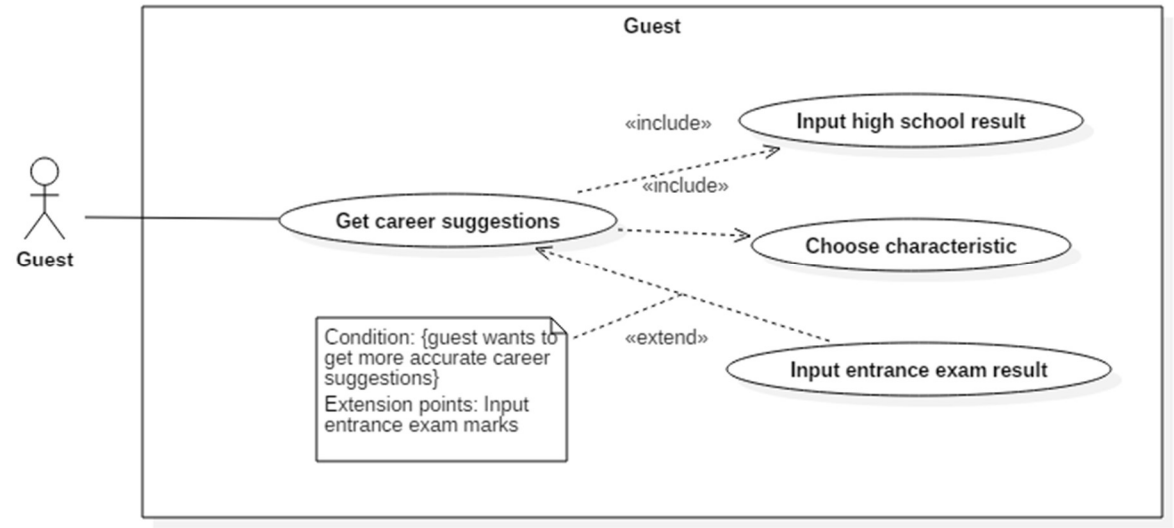


Figure 9: <Guest> Get career suggestions use case

USE CASE – G01			
Use Case No.	G01	**Use Case Version**	2.0
Use Case Name	Get career suggestions		
Author	Đàm Phước Đức Duy		

Date	25/09/2018	Priority	High

Actor:

Guest

Summary:

This use case describes how guest (high school pupil) receive career suggestions by providing his/her academic result.

Goal:

To provide career suggestions to guest.

Triggers:

Guest sends "Nhận gợi ý" request.

Preconditions:

Guest's academic result is valid.

Post Conditions:

- **Success:** Career suggestions are displayed.
- **Fail:** Error message is displayed.

Main Success Scenario:

Step	Actor Action	System Response
1	Guest sends "Học sinh cấp 3" request to receive suggestions for high school pupil. [Alternative 1]	System asks guest to provide the number of high school subjects that he/she is good at: – Number of high school subjects: number text input, required, value from 1 to 4.
2	Guest provides the number of high school subjects that he/she is good at.	System asks guest to specify the subjects that he/she is good at from the system's predefined list of high school subjects.

3	Guest specifies the subjects that he/she is good at.	System asks guest to provide the GPA for each high school subject that he/she is good at: – GPA of subject: number text input, required, value from 0 to 10. [Exception 1]
4	Guest provides the GPA for the high school subjects that he/she is good at.	System asks guest to provide his/her GPA of grade 12 (the senior year of high school): – GPA of grade 12: number text input, required, value from 0 to 10. [Exception 1]
5	Guest provides his/her GPA of grade 12.	System asks guest if he/she is an "Introvert" or "Extrovert".
6	Guest specifies his/her characteristic ("Introvert" or "Extrovert").	Guest may now send the "Nhận gợi ý" request for receiving the career suggestions.
7	Guest sends "Nhận gợi ý" request. [Alternative 2]	Provide the suggestions about careers in order of suitability to guest.

Alternative Scenarios:

No	Actor Action	System Response
1	Guest sends "Tân sinh viên ngành IT" request to receive the suggestions and	System asks guest to specifies an IT position from the system's predefined list of IT positions.

	information about an IT position.	
2	Guest sends "Bạn muốn nhận gợi ý chính xác hơn?" request for higher accuracy career suggestions.	System asks guests to provide his/her National Examination result: – Mark of subject: number text input, required, value from 0 to 10. For non-compulsory subjects, guest may not provide if he/she does not take part in: – Checkbox to disable (guest does not take part in this subject).

Exceptions:

No	Cause	System Response
1	Guest's input is not digits.	Display error message: "Vui lòng nhập chữ số".

Relationships: N/A

Business Rules:

- High school pupil wants to get career suggestions based on his/her academic result.
- In order to receive the career suggestions, pupil needs to provide the following information:
  - Number of high school subjects that pupil is good at (1 – 4).
    - Which are those subjects? (the number of selected subjects corresponds to the previous answer)
    - GPA for each subject (0 – 10).
  - GPA of grade 12 (0 – 10).
  - Pupil's characteristic (introvert or extrovert).
- Suggested careers will be ranked based on the pupil's suitability.
- For higher accuracy, pupil can provide his/her Vietnam's National High School Examination result, includes:
  - Taken subjects.
  - Mark of each subject (0 – 10).

– After combining the National Examination result with the previous information, suggestions will be ranked and provided again with better suitability.

Table 9: <Guest> Get career suggestions use case specification table

2.3.2.2. <Guest> Get career trends prediction (G02)

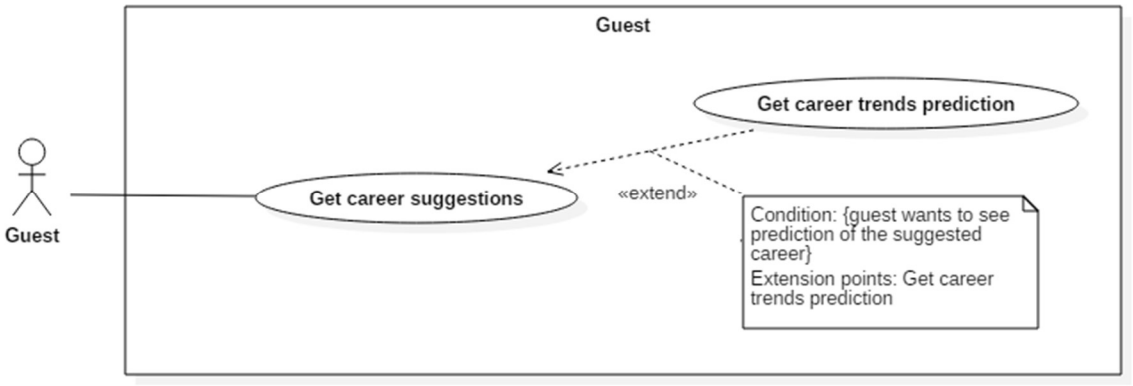


Figure 10: <Guest> Get career trends prediction use case

USE CASE – G02			
Use Case No.	G02	**Use Case Version**	2.0
Use Case Name	Get career trends prediction		
Author	Đàm Phước Đức Duy		
Date	23/09/2018	**Priority**	High

Actor:

Guest

Summary:

This use case describes how guest (high school pupil) receive career trends prediction of a major.

Goal:

To provide forecast for trend of a specific major to guest.

Triggers:

Guest sends request for receiving career trends prediction.

Preconditions:

Guest has received career suggestions.

Post Conditions:

- **Success:** Forecast for trend of the selected major are displayed.
- **Fail:** Error message is displayed.

Main Success Scenario:

Step	Actor Action	System Response
1	Guest sends request for receiving career trends prediction.	– Supported information of selected major in the past and its prediction in future. – Trend of the selected major in future.

Alternative Scenarios: N/A

Exceptions: N/A

Relationships: N/A

Business Rules:

- High school pupil wants to see the forecast for trend of a specific major from the year that historical data is collected to a certain year in future:
    - Historical data of selected major must be available in the system.
    - Historical data is collected from:
        - The White books of Information Technology in Vietnam
        - General Statistics Office of Vietnam
        - Google Public Data
    - The year until which the pupil wants to get forecast must satisfy the following conditions:
        - Larger than the current year.
        - No more than 5 years after the current year.

Table 10: <Guest> Get career trends prediction use case specification table

2.3.2.3. <Guest> Get recruitment use case (G03)

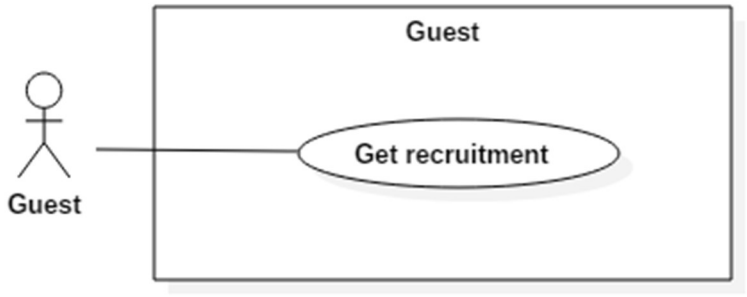


Figure 11: <Guest> Get career trends prediction use case

USE CASE – G03			
Use Case No.	G03	**Use Case Version**	2.0
Use Case Name	Get recruitment		
Author	Đàm Phước Đức Duy		
Date	23/09/2018	**Priority**	High

Actor:

Guest

Summary:

This use case allows guest to get all recruitment.

Goal:

To provide guest system's list of available recruitment.

Triggers:

Guest sends request for getting recruitment.

Preconditions: N/A

Post Conditions:

- **Success:** Forecast for trend of the selected major are displayed.
- **Fail:** Error message is displayed.

Main Success Scenario:

Step	Actor Action	System Response
1	Guest sends request for receiving list of available recruitment.	List of published recruitment. [Exception 1]

Alternative Scenarios: N/A

Exceptions:

Step	Actor Action	System Response
1	System tries to retrieve an empty list of available recruitment.	Display message: “Không tìm thấy tin tuyển dụng nào.”.

Relationships: N/A

Business Rules:

- Guest can see list of recruitment published by partners (companies approved by staff).
- Recruitment will be displayed as list with following information:
  - Company logo
  - Title of recruitment
  - Job description
  - Required skills
  - Start date
  - Due date
  - Location
  - Status (Hot, New)

Table 11: <Guest> Get recruitment use case specification table

2.3.3. <Staff> Overview use case

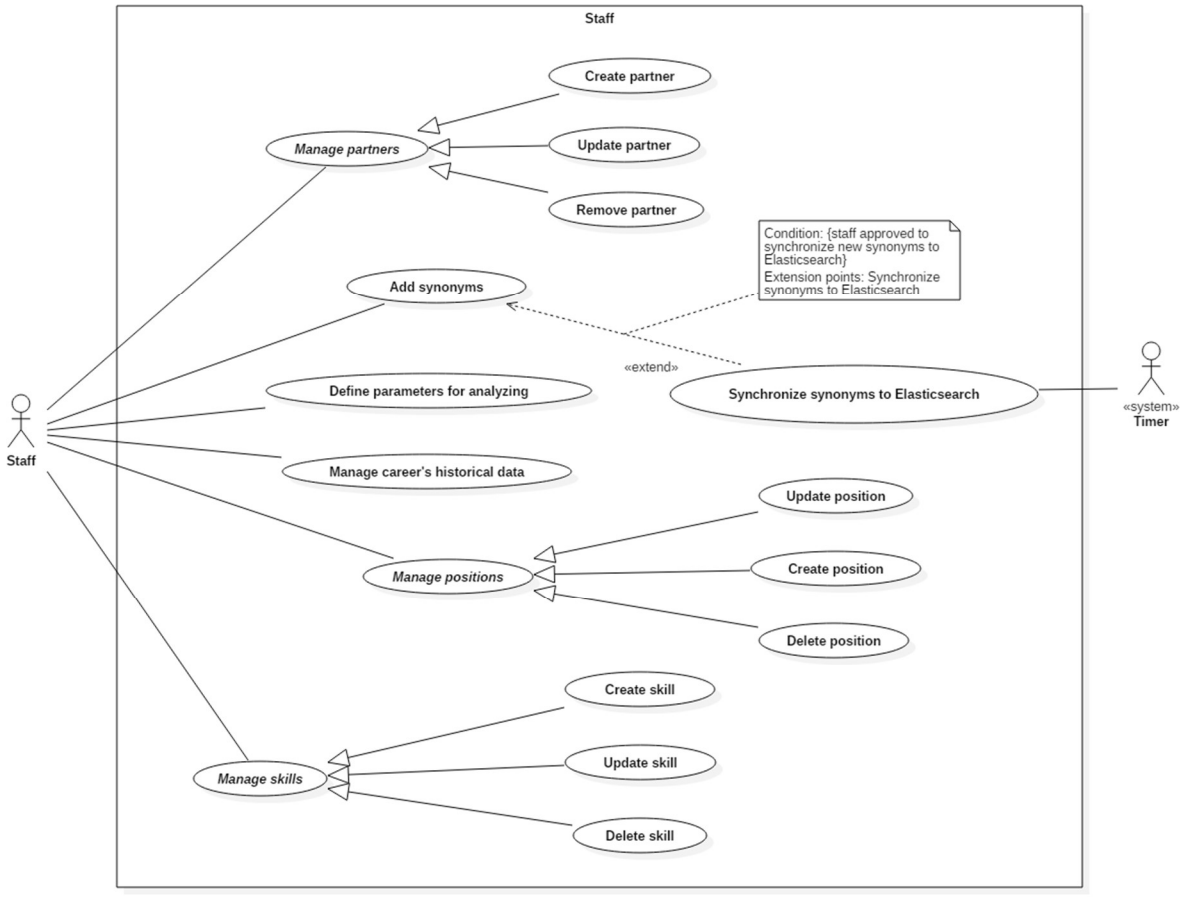


Figure 12: <Staff> Overview use case

2.3.3.1. <Staff> Add synonyms (S01)

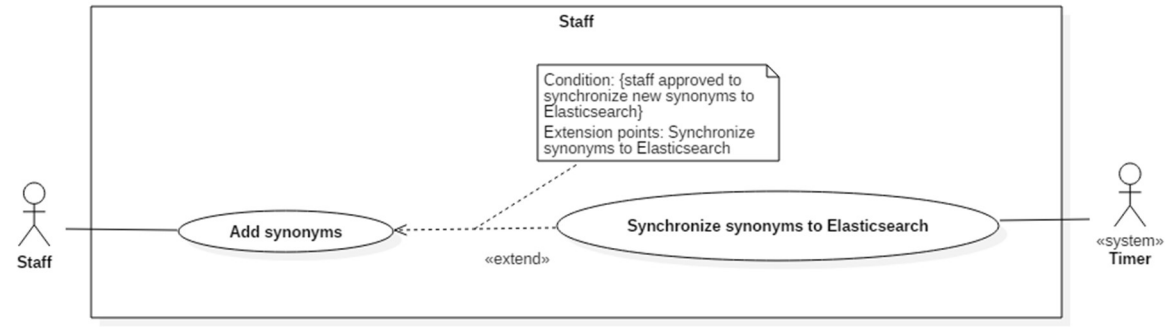


Figure 13: <Staff> Add synonyms use case

USE CASE – S01			
Use Case No.	S01	**Use Case Version**	2.0
Use Case Name	Add synonyms		
Author	Đàm Phước Đức Duy		

Date	23/09/2018	Priority	Normal

Actor:

Staff

Summary:

This use case describes how the staff add new synonym of position or skill into storage.

Goal:

To support the data cleaning process.

Triggers:

Staff sends "Add new synonym" request.

Preconditions:

Guest must login with role Staff.

Post Conditions:

- **Success:**
  - New synonym is inserted to storage.
  - Success message is displayed.
- **Fail:** Error message is displayed.

Main Success Scenario:

Step	Actor Action	System Response
1	Staff sends "Configure parameters" request to receive list of available keywords.	Display list of keywords and fields to input new synonym: – Synonym: blank text input, required, length [1, 255]. [Exception 1]
2	Staff inputs synonym and sends "Add new synonym" request.	Insert new synonym of selected word into database. Display success message. Display inserted synonym in the selected word's tab.

		[Exception 2, 3]

Alternative Scenarios: N/A

Exceptions:

No	Actor Action	System Response
1	Staff cancels request.	Input form disappear and nothing change.
2	Length of new synonym is not in range [1, 255].	Display error message: "Please input new synonym to add.".
3	Input synonym has already existed.	Display error message: "Synonym existed. Please try another one…".

Relationships: N/A

Business Rules:

- For better keyword extraction while collecting and cleaning data from recruitment websites, staff may add synonym for the available keywords.
- New synonym must satisfy following conditions:
  - Haven't existed in database before.
  - Main keyword of synonym is available.

Table 12: <Staff> Add synonyms use case specification table

2.3.3.2. <Staff> Synchronize synonyms to Elasticsearch (S02)

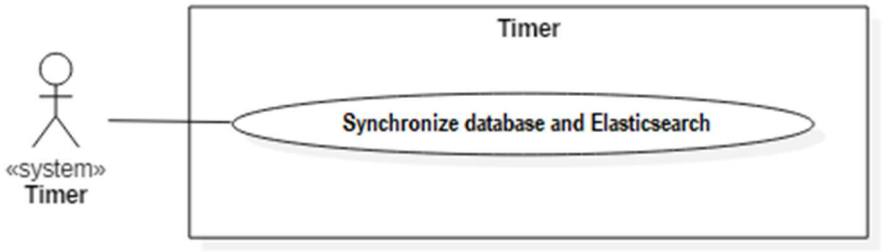


Figure 14: <Timer> Synchronize database and Elasticsearch use case

USE CASE - S02			
Use Case No.	S02	**Use Case Version**	2.0
Use Case Name	Synchronize database and Elasticsearch		
Author	Tăng Hồ Duy Minh		
Date	23/09/2018	**Priority**	High

Actor:

Staff

Summary:

This use case allows staff to manually insert lately added keywords into Elasticsearch's storage.

Goal:

- To synchronize data between system's storage and Elasticsearch's storage.
- To improve the accuracy of keyword extraction service.
- To manually insert keywords into Elasticsearch's storage besides the synchronizing timer task.

Triggers:

Staff sends request for synchronizing keywords to Elasticsearch.

Preconditions:

- Keywords are available in system.
- Keywords are set with status "Unpushed".

Post Conditions:

- **Success:** Display success message.
- **Fail:**
  - Display error message.
  - Error is tracked in log file.

Main Success Scenario:

Step	Actor Action	System Response
1	Staff sends request for synchronizing list of "Unpushed" keywords to Elasticsearch.	Display success message. [Alternative 1]

Alternative Scenarios:

No	Actor Action	System Response
1	Staff sends request for synchronizing empty list of "Unpushed" keywords to Elasticsearch.	– Skip the process. – Display message: "No keywords to be pushed.".

Exceptions: N/A

Relationships: N/A

Business Rules:

- Besides the synchronizing timer task that automatically inserts "Unpushed" keywords into Elasticsearch.
- Staff can manually insert keywords into Elasticsearch without waiting for the timer task.

Table 13: <Staff> Synchronize database and Elasticsearch specification table

2.3.3.2. <Staff> Define parameters for analyzing (S03)

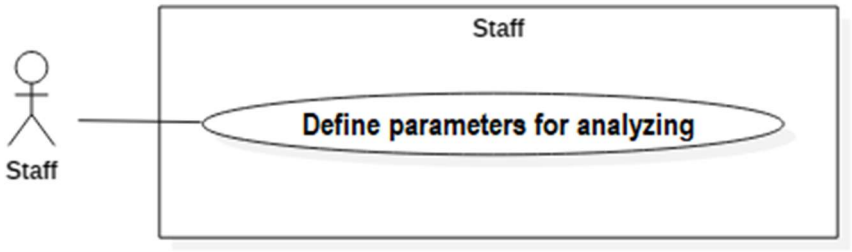


Figure 15: <Staff> Define parameters for analyzing use case

USE CASE – S03			
Use Case No.	S03	**Use Case Version**	2.0
Use Case Name	Define parameters for analyzing		
Author	Đàm Phước Đức Duy		
Date	23/09/2018	**Priority**	Normal

Actor:

Staff

Summary:

This use case allows staff to define the parameters for the analyzing processes.

Goal:

To define parameters for the data analyzing process.

Triggers:

Staff sends “Configure parameters” request.

Preconditions:

Guest must login with role Staff.

Post Conditions:

- **Success:**
  - Modified parameters are updated.
  - Success message is displayed.

- **Fail:** Error message is displayed.

Main Success Scenario:

Step	Actor Action	System Response
1	Staff sends "Configure parameters" request. [Alternative 1]	Display list of parameters and fields to input parameter's value: – Value: number text input, required.
2	Staff configures parameters and send "Save" request.	– Update modified parameters. – Display success message. [Exception 1, 2]

Alternative Scenarios:

No	Actor Action	System Response
1	Staff cancels request.	Input form disappear and nothing change.

Exceptions:

No	Actor Action	System Response
1	Input value of parameter is empty.	Display error message: "Parameter's value can't be empty!".
2	Input value of parameter is not digits.	Display error message: "Parameter's value must be digits. Please try again…".

Relationships: N/A

Business Rules:

- Staff can edit parameters of the analyzing process for better suggestions.

Table 14: <Staff> Define parameters for analyzing use case specification table

2.3.3.3. <Staff> Manage career's historical data (S04)

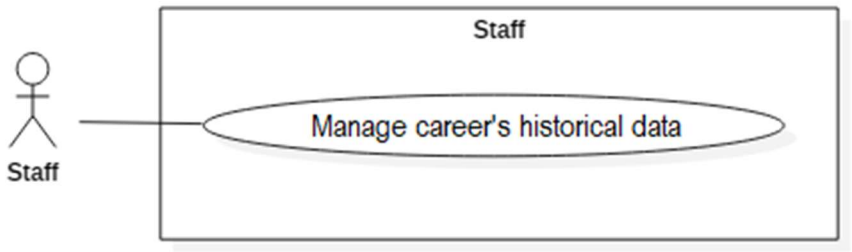


Figure 16: <Staff> Manage career's historical data use case

USE CASE – S04			
Use Case No.	S03	**Use Case Version**	2.0
Use Case Name	Manage career's historical data		
Author	Đàm Phước Đức Duy		
Date	23/09/2018	**Priority**	Normal

Actor:

Staff

Summary:

This use case allows staff to define and edit the supported information of a major which help the system provide the trends prediction.

Goal:

- To define historical data of a major.
- To provide better trends prediction to guest.

Triggers:

Staff sends request to manage historical data.

Preconditions:

Guest must login with role Staff.

Post Conditions:

- **Success:**

- Historical data is inserted to database.
- Success message is displayed.

– **Fail:** Error message is displayed.

Main Success Scenario:

Step	Actor Action	System Response
1	Staff sends request to manage the historical data of a major. [Alternative 1]	List of editable historical data fields by year: – Value: number text input, required.
2	Staff edit the historical data and sends "Save" request.	– Update modified values. – Display success message. [Exception 1]

Alternative Scenarios:

No	Actor Action	System Response
1	Staff cancels request.	Input form disappear and nothing change.

Exceptions:

No	Actor Action	System Response
1	Input value is not digits.	Display error message: "Historical value must be digits. Please try again…".

Relationships: N/A

Business Rules:

– Staff can define supported information collected from websites so that the system can provide better trends prediction for a major.

– Supported information includes:

  - Human resource

- Population
- Salary
- Number of students that are currently studying the major
- Number of jobs provided

Table 15: <Staff> Manage career's historical data use case specification table

2.3.4. <Partner> Overview use case

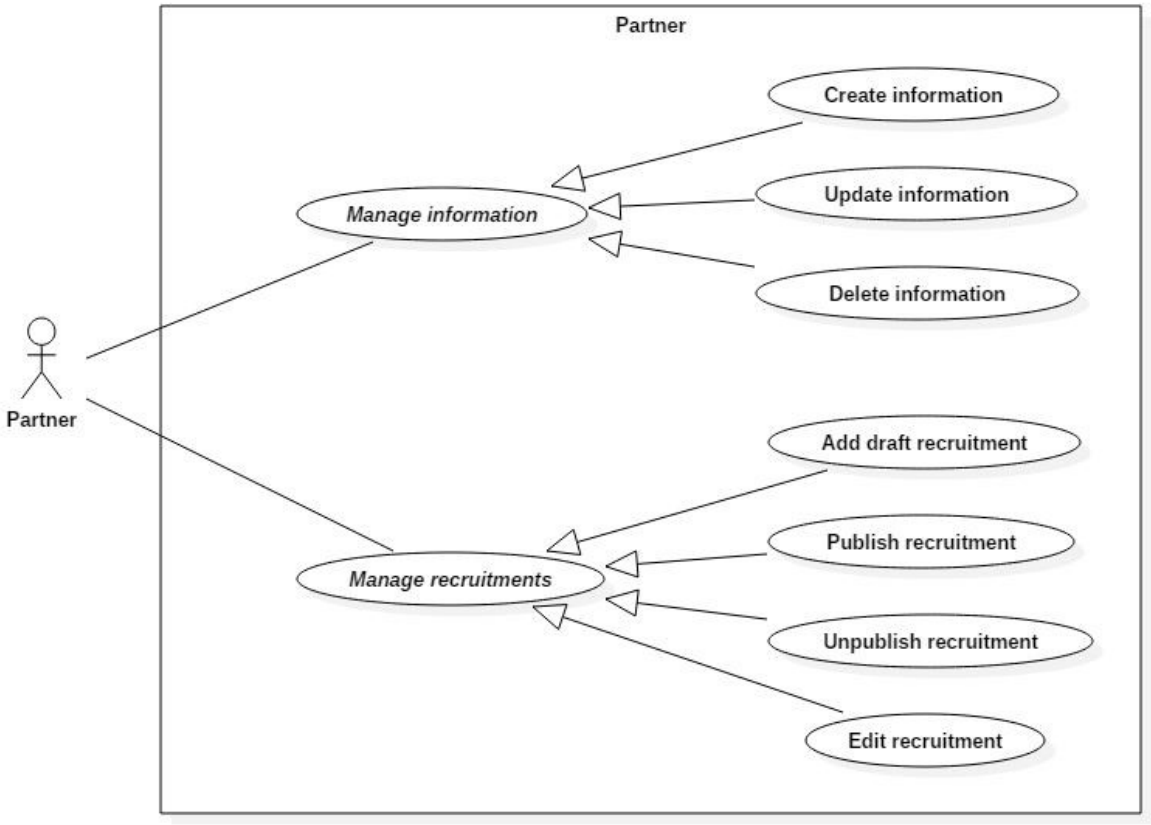


Figure 17: <Partner> Overview use case

2.3.4.1. <Partner> Manage partner's recruitment (P01)

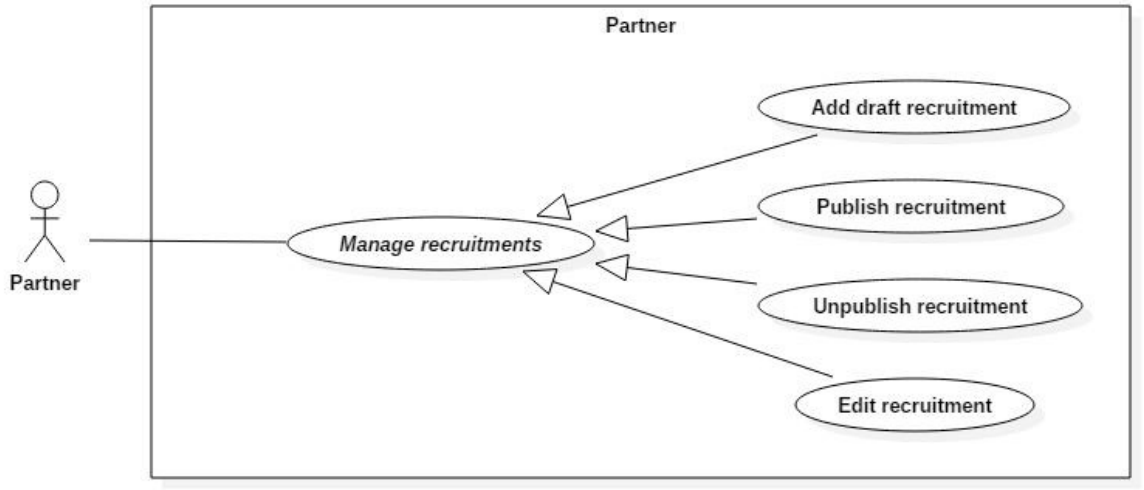


Figure 18: <Partner> Manage partner's recruitment use case

2.3.4.1.1. <Partner> Add draft recruitment

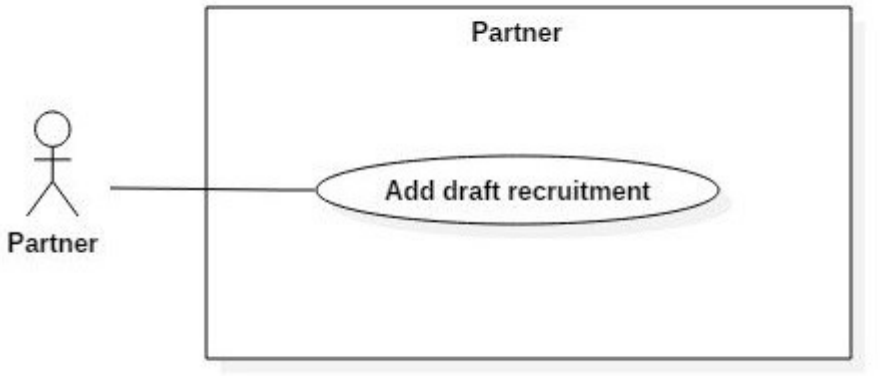


Figure 19: <Partner> Add draft recruitment use case

USE CASE – P01A			
Use Case No.	P01A	**Use Case Version**	2.0
Use Case Name	Add draft recruitment		
Author	Đàm Phước Đức Duy		

Date	24/09/2018	Priority	Normal

Actor:

Partner

Summary:

This use case allows partner to create a draft recruitment.

Goal:

- To help partners recruit employees with needed skills.
- To help users find suitable job.
- To enrich information in system.
- To provide trends of programming skills.

Triggers:

Partner sends request to add draft recruitment.

Preconditions:

- Guest must login with role Partner.
- Partner has been approved by Staff.
- Draft recruitment required contents is valid.

Post Conditions:

- **Success:**
  - New recruitment was created with status “Unpublished”.
  - Success message is displayed.
- **Fail:** Error message is displayed.

Main Success Scenario:

Step	Actor Action	System Response
1	Partner sends request for creating draft recruitment.	System asks partner to provide draft recruitment’s required information, includes: – Recruitment title: blank text input, required.

	[Alternative 1]	– Due date: datetime picker, required. – Number of needed employees: number text input, required. – Job description: blank text input, required. – Job requirement: blank text input, required. – Email: blank text input, required. – Phone: blank text input, required. – Address: blank text input, required. – Additional information (optional, maximum 10 fields): • Field title: blank text input • Field information: blank text input
2	Partner inputs required information and sends request to add draft recruitment.	Display success message. [Exception 1]

Alternative Scenarios:

No	Actor Action	System Response
1	Partner cancels request.	Return to previous page.

Exceptions:

No	Cause	System Response
1	Required field is not specified.	Display error message “<Field name> is required.”.

Relationships: N/A

Business Rules:

– Partner can create recruitment for recruiting employees.

– Created draft recruitment has not been published yet.

Table 16: <Partner> Add draft recruitment use case specification table

2.3.4.1.2. <Partner> Publish recruitment

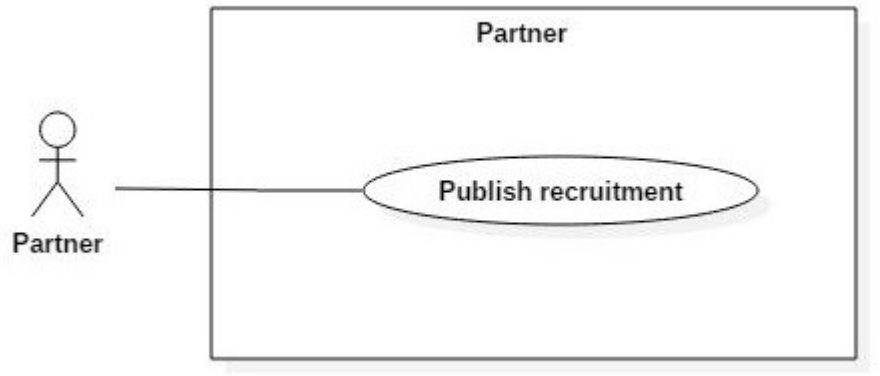


Figure 20: <Partner> Publish recruitment use case

USE CASE – P01B			
Use Case No.	P01B	**Use Case Version**	2.0
Use Case Name	Publish recruitment		
Author	Đàm Phước Đức Duy		
Date	24/09/2018	**Priority**	Normal

Actor:

Partner

Summary:

This use case allows partner to post their draft recruitment.

Goal:

To publish partner's draft recruitment.

Triggers:

Partner sends request to publish a recruitment.

Preconditions:

- Guest must login with role Partner.
- Partner has been approved by Staff.
- Partner's recruitment is draft or has been unpublished.

Post Conditions:

- **Success:**
  - Recruitment is posted on the system's list recruitment.
  - Recruitment is set with status "Published".
  - Success message is displayed.
- **Fail:** Error message is displayed.

Main Success Scenario:

Step	Actor Action	System Response
1	Partner sends request to publish a specific draft recruitment.	– Recruitment is moved to published recruitment table. – Display success message dialog: "Đăng tin thành công!".

Alternative Scenarios: N/A

Exceptions:

No	Cause	System Response
1	Could not load data.	Display error message dialog: "Đã có lỗi xảy ra. Xin vui lòng thử lại sau…".

Relationships: N/A

Business Rules:

- Partner wants to post their draft recruitment so that guests can see them.
- Recruitment's start date is set to the system's current time once it is published for the first time.

Table 17: <Partner> Publish recruitment use case specification table

2.3.4.1.3. <Partner> Unpublish recruitment

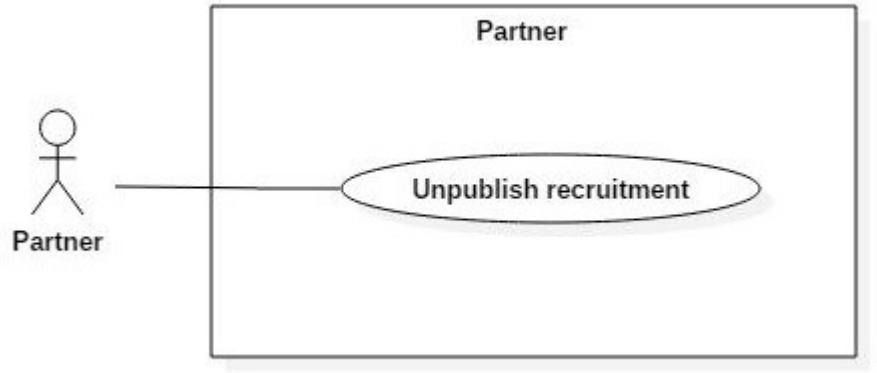


Figure 21: <Partner> Unpublish recruitment use case

USE CASE – P01C			
Use Case No.	P01C	**Use Case Version**	2.0
Use Case Name	Unpublish recruitment		
Author	Đàm Phước Đức Duy		
Date	24/09/2018	**Priority**	Normal

Actor:

Partner

Summary:

This use case allows partner to remove their recruitment from system's list of published recruitment.

Goal:

- To remove recruitment from system's list of published recruitment.
- To make recruitment editable.

Triggers:

Partner sends request to unpublish recruitment.

Preconditions:

- Guest must login with role Partner.
- Partner has been approved by Staff.

– Partner's recruitment has been published.

Post Conditions:

– **Success:**
  - Recruitment is removed from the system's list recruitment.
  - Recruitment is set with status "Unpublished".
  - Success message is displayed.

– **Fail:** Error message is displayed.

Main Success Scenario:

Step	Actor Action	System Response
1	Partner sends request to unpublish a specific recruitment.	– Recruitment is moved to draft recruitment table. – Display success message dialog: "Gỡ tin thành công!". – Published recruitment is removed from the system's list recruitment.

Alternative Scenarios: N/A

Exceptions:

No	Cause	System Response
1	Could not load data.	Display error message dialog: "Đã có lỗi xảy ra. Xin vui lòng thử lại sau…".

Relationships: N/A

Business Rules:

– Partner can remove a published recruitment for editing or unpublishing.

Table 18: <Partner> Unpublish recruitment use case specification table

2.3.3.1.3. <Partner> Edit recruitment (P04)

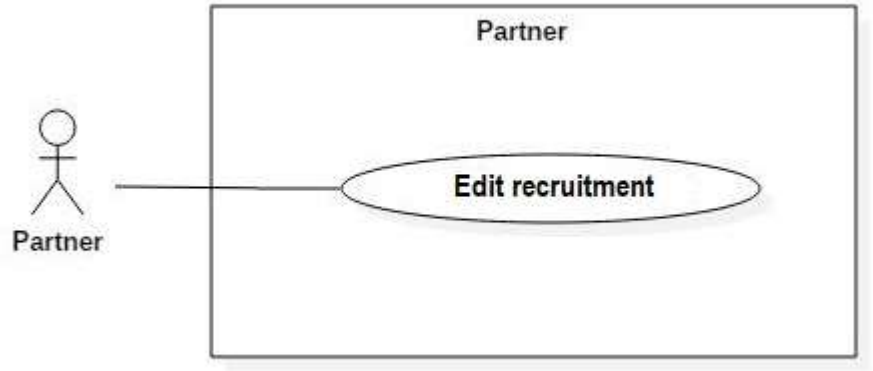


Figure 22: <Partner> Edit recruitment use case

USE CASE – P01D			
Use Case No.	P01D	**Use Case Version**	2.0
Use Case Name	Edit recruitment		
Author	Đàm Phước Đức Duy		
Date	24/09/2018	**Priority**	Normal

Actor:

Partner

Summary:

This use case allows partner to edit an unpublished recruitment.

Goal:

To edit recruitment's content.

Triggers:

Partner sends request to edit an unpublished recruitment.

Preconditions:

- Guest must login with role Partner.
- Partner has been approved by Staff.
- Partner's recruitment is draft or has been unpublished.

Post Conditions:

- **Success:**
  - Recruitment is updated.
  - Success message is displayed.
- **Fail:** Error message is displayed.

Main Success Scenario:

Step	Actor Action	System Response
1	Partner sends request to edit an unpublished recruitment.	– Recruitment is moved to draft recruitment table. – Display success message dialog: "Gỡ tin thành công!". – Published recruitment is removed from the system's list recruitment.

Alternative Scenarios: N/A

Exceptions:

No	Cause	System Response
1	Could not load data.	Display error message dialog: "Đã có lỗi xảy ra. Xin vui lòng thử lại sau…".

Relationships: N/A

Business Rules:

- Partner can edit an unpublished recruitment.

Table 19: <Partner> Edit recruitment use case specification table

3. Software System Attribute

3.1 Usability

- Guest can learn to use the website within 5 minutes.
- Staff can learn to use functions in website application within less than 1 day of training.

3.2 Availability

- Server running 24/7

3.3 Maintainability

- Web services are separated into modules for easy maintenance.

4. Conceptual Diagram2

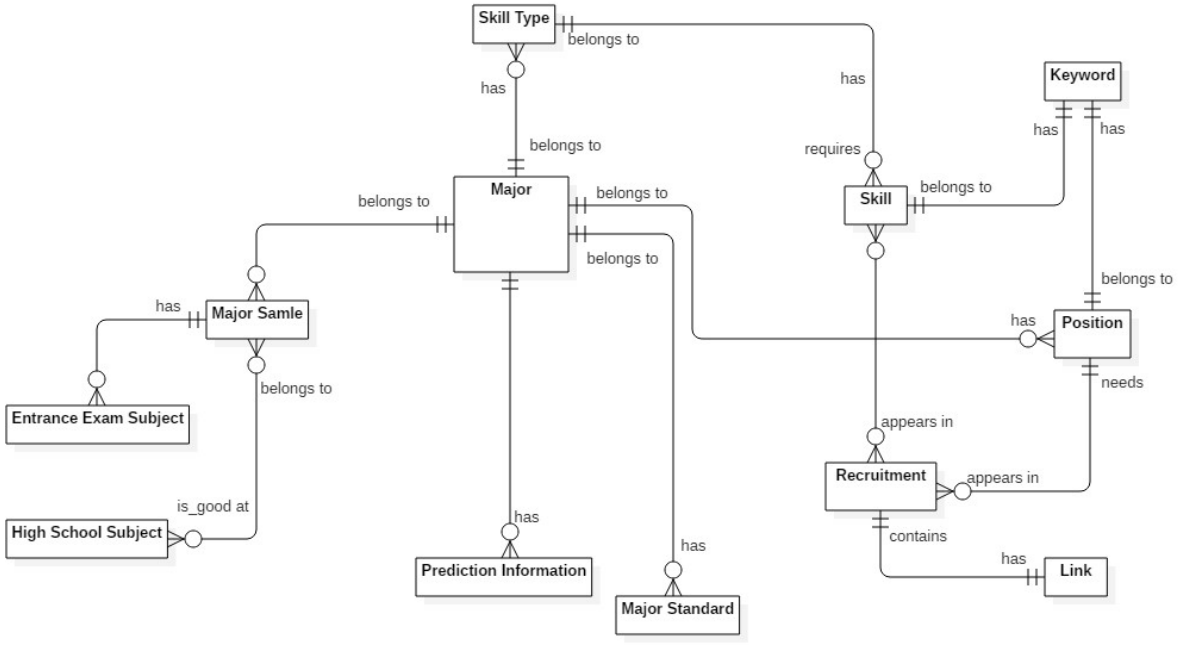


Figure 23: Conceptual Diagram

Data Dictionary

Entity Data dictionary: describe content of all entities	
Entity Name	**Description**
Major	Describe all majors in the system
Major Sample	Describe all sample data of majors in the system
Major Standard	Describe all standards of majors in the system
High School Subject	Describe all high school subjects in the system
Entrance Exam Subject	Describe all National Examination subjects in the system
Position	Describe all positions in the system
Skill Type	Describe all types of skills in the system
Skill	Describe all skills in the system
Keyword	Describe all keywords and their synonyms in the system
Prediction Information	Describe all prediction of majors in the system
Recruitment	Describe all recruitment information in the system
Link	Describe all sources of recruitment in the system

Table 20: Conceptual Data Dictionary

[2] OMG UML (August 2018), OMG Unified Modeling Language ™ (OMG UML) Superstructure

This page is intentionally left blank

D. Software Design Description

I. Design Overview

This document describes the technical and user interface design of CTSA System. It includes the architectural design, the component design, the user interface design and the design of database model:

- The architectural design describes the overall architecture of the system and the architecture of each main component and subsystem.
- The component design describes all system's component (class, interface…) together with relationships between them. It also explains clearly their purposes, methods within each component and detailed algorithm, pseudo code to implement them.
- User interface design which describes all screens' interface; each screen includes which types of input, output element and used by what action.
- The database design describes the relationships between entities and details of each entity.

II. System Architectural Design

The CTSA system is developed applying the Microservice Architecture.

Instead of building a single monstrous, monolithic application, the idea is to split the application into set of smaller, interconnected services. Based on the afore-mentioned requirements of CTSA system, Microservice Architecture is taken into consideration for the following reasons:

- **Efficient modularity:**
  - Better testability: Services are smaller and faster to test.
  - Better deployability: Services can be deployed independently.
- **Improved fault isolation**:
  - If there is a defect in one service, only that service will be affected. The other services will continue to handle requests. In comparison, one misbehaving component of a monolithic architecture can bring down the entire system.
- **Flexible choices of technology stacks:**
  - When developing a new service, developer (or team) can pick a new technology stack.
  - Similarly, when making major changes to an existing service, it is possible to rewrite it using a new technology stack.

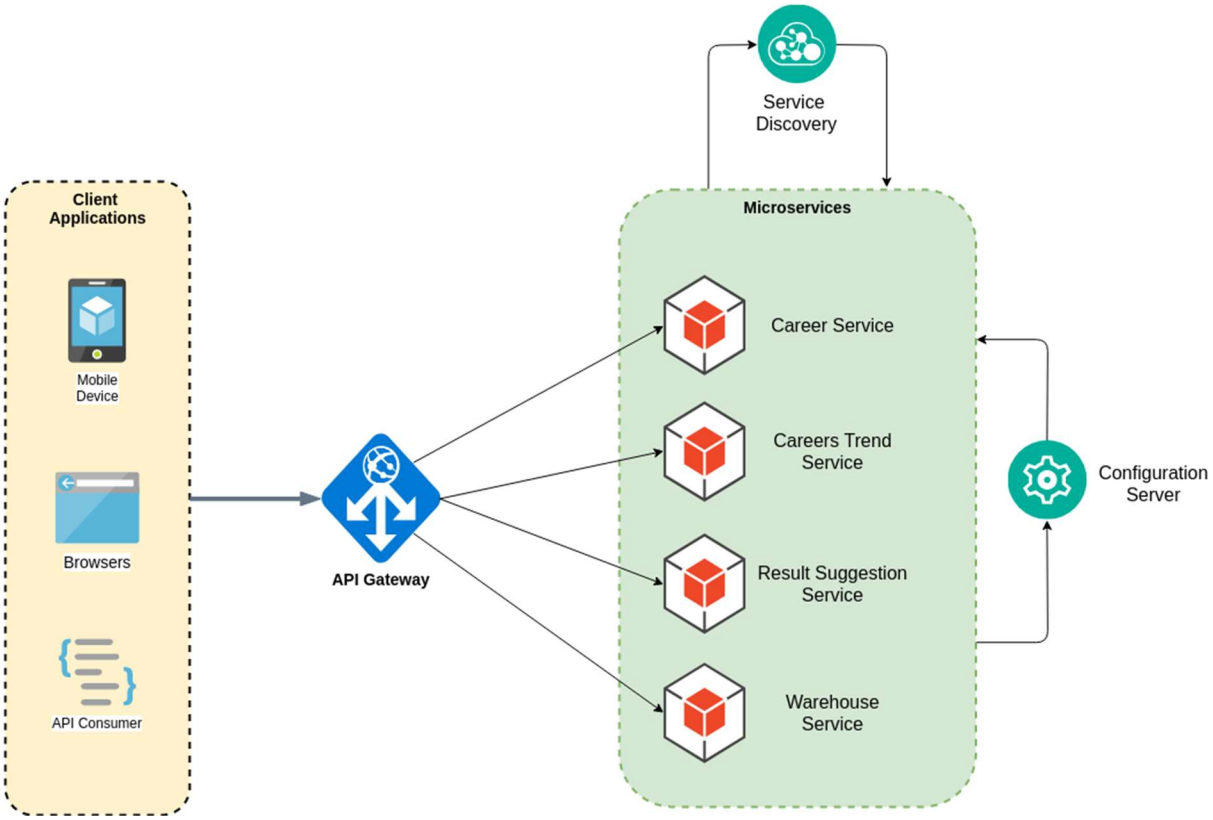


Figure 24: Microservice Architecture[3,4]

This application applies Microservices Architecture with following components:

- **Microservices**: Services which each handles a single business capability, following *database per service* pattern. Each service provides APIs relating to a single boundary context.
- **API Gateway**: Single entry point for API consumers (browsers, devices, other APIs).
- **Service Discovery**: A service which enables client-side load-balancing and decouples service providers from consumers without the need for DNS.
- **Configuration Server**: Dynamic, centralized configuration management for decentralized services. Microservices are built consuming the corresponding configurations on startup and then refreshes for update-to-date configurations without restarting.
- **Client Applications**: Web applications, mobile devices or any APIs consumers, which consume APIs directly from the API Gateway and presents information to the users and how users interact with the system.

[3] Wasson, M. and Celarier, S. (November 13, 2018), *Microservices architecture style*. Retrieved from https://docs.microsoft.com/en-us/azure/architecture/guide/architecture-styles/microservices
[4] Spring Team, *Spring Cloud,* Retrieved from https://spring.io/

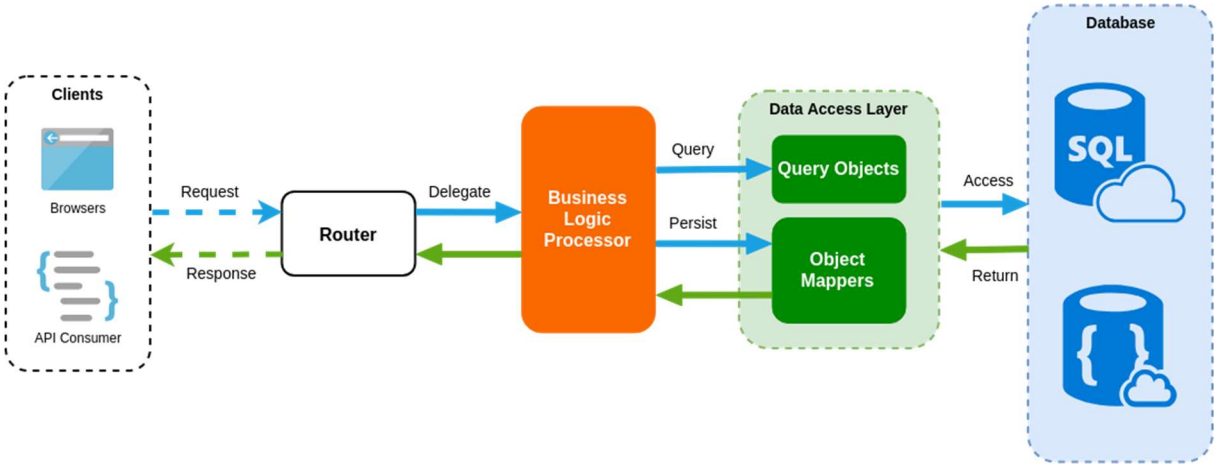


Figure 25: Service Architecture

Microservices inside the CTSA system share the same architecture design, each contains the following components:

- **Database**: The service's data storage, which can be structure or non-structure databases, and are either local or cloud-based. Following the database per service pattern, each microservice can only have access to one specific database.
- **Data Access Layer:** The only component which has direct access to the database, and consists of two components: Query Object and Objects Mapper.
- **Query Objects:** Components that are sent from the business logic processor and later transferred into queries of the corresponding database.
- **Object Mappers:** Components that represent the structure of the corresponding tables or documents; which are used for persisting data to database or receiving results returned from database.
- **Business Logic Processor:** The processor which is responsible for processing the primary business logic of the microservice.
- **Router:** The component which handle requests from clients and send responses back to the clients. Requests are not directly processed but delegated to corresponding business logic processor, results are returned back to router for generating responses.
- **Clients**: Browsers or any APIs consumers, which consume APIs. In the CTSA system, clients are other microservices and the API Gateway.

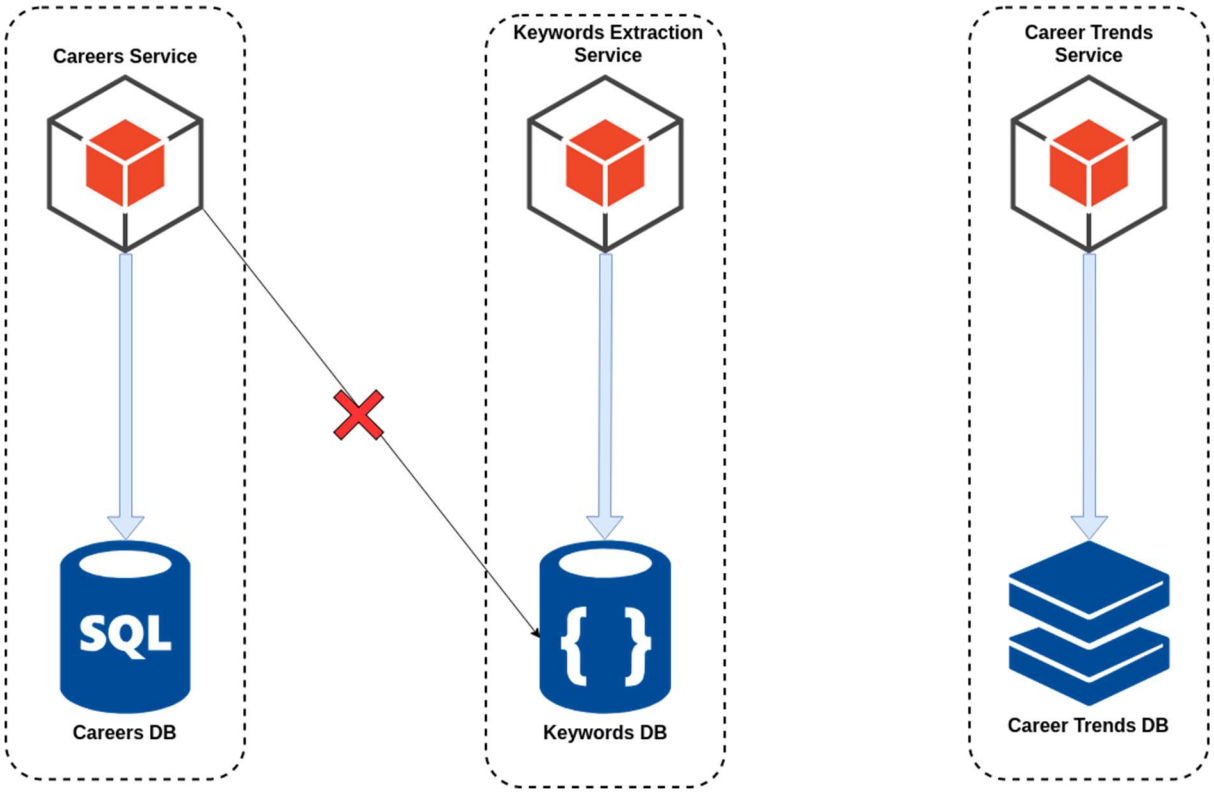


Figure 26: Database per service pattern[5]

As mentioned above, services must be loosely coupled so that they can be developed, deployed and scaled independently. Moreover, different services have different data storage requirements.

Applying the *database per service* pattern, the database of the CTSA system is broken down into smaller databases which is called ***sub-database*** in this document. Following this pattern, each service's database is effectively part of the implementation of that service, and cannot be accessed directly by other services.

Specifically, the CTSA system contains the followings sub-databases, divided into different groups, based on the needs of use:

- **Structure Databases:**
- *Includes*: Careers Database, Warehouse Database, Recruitment Database.
- *Reasons*: The above-mentioned databases contain information about careers and recruitment with relative positions and skills. Data from these databases are later used to run machine learning algorithms for providing suggestions. Such data need to be well-organized and optimized for data loading instead of data retrieving, as well as require high consistency.
- *Trade-offs*: Using structure databases sacrifices the system's performance when receiving requests to retrieve data from clients. On the contrary, the main goal of the CTSA system is to analyzing and providing suggestions, which requires optimization of data for machine learning algorithms; therefore, the trade-off in

[5] Richardson, C. (December 4, 2015), *Event-Driven Data Management for Microservices*. Retrieved from https://www.nginx.com/blog/event-driven-data-management-microservices/

this circumstance is acceptable. *The system's speed is later enhanced with the use of non-structure databases.*

- **Non-structure Databases:**
- With the low performance due to the use of structure databases, non-structure databases are taken into consideration for applying in storing data whose consistency is not critical.
- **Inverted-index database:**
- *Includes:* Keyword Database
- *Reasons:* The service connecting to the mentioned database is responsible for data extraction, which is a critical part of the data cleansing process of the CTSA system. Such function can be optimized using full-text search techniques. The use of inverted-index structure in data storing enables faster full-text searches.
- *Trade-offs:* Data stored in such database is non-structured, which brings considerable difficulties in handling and processing. The team do not completely understand the inverted-index as well as full-text search techniques. Therefore, the use of inverted-index database is minimized down to only one service.
- **In-memory database:**
- *Includes:* Career Trends Database, Result Suggestions Database
- *Reasons:* Services connecting to above-mentioned databases are responsible for running machine learning algorithms and providing suggestions, temporary data is generated during the process of calculating need to be stored and are later used for comparison. Since only small piece of generated data are later chosen to perform suggestions, using structure databases in this circumstance is not unnecessary. Traditional structure databases also store data on disks, which leads to low performance on querying among thousands or even millions records for comparison. On the contrary, in-memory databases store data on memories, which is able to perform faster data retrieving for comparison. Since storage of in-memory databases is small, unnecessary generated data can be automatically evicted by default based on the configurations of the databases.
- *Trade-offs:* The storage of in-memory databases is small, which can lead to unexpected data loss. The eviction of data protocols of such databases needs to be configured using Least Recently Used algorithm.

Using a database per service has the following benefits:

- Ensure that the services are loosely coupled. Changes to one service's database does not impact any other services.
- Each service can use the type of database that is best suited to its needs.

III. Component Diagram[6]

[6] OMG UML (August 2018), OMG Unified Modeling Language ™ (OMG UML) Superstructure

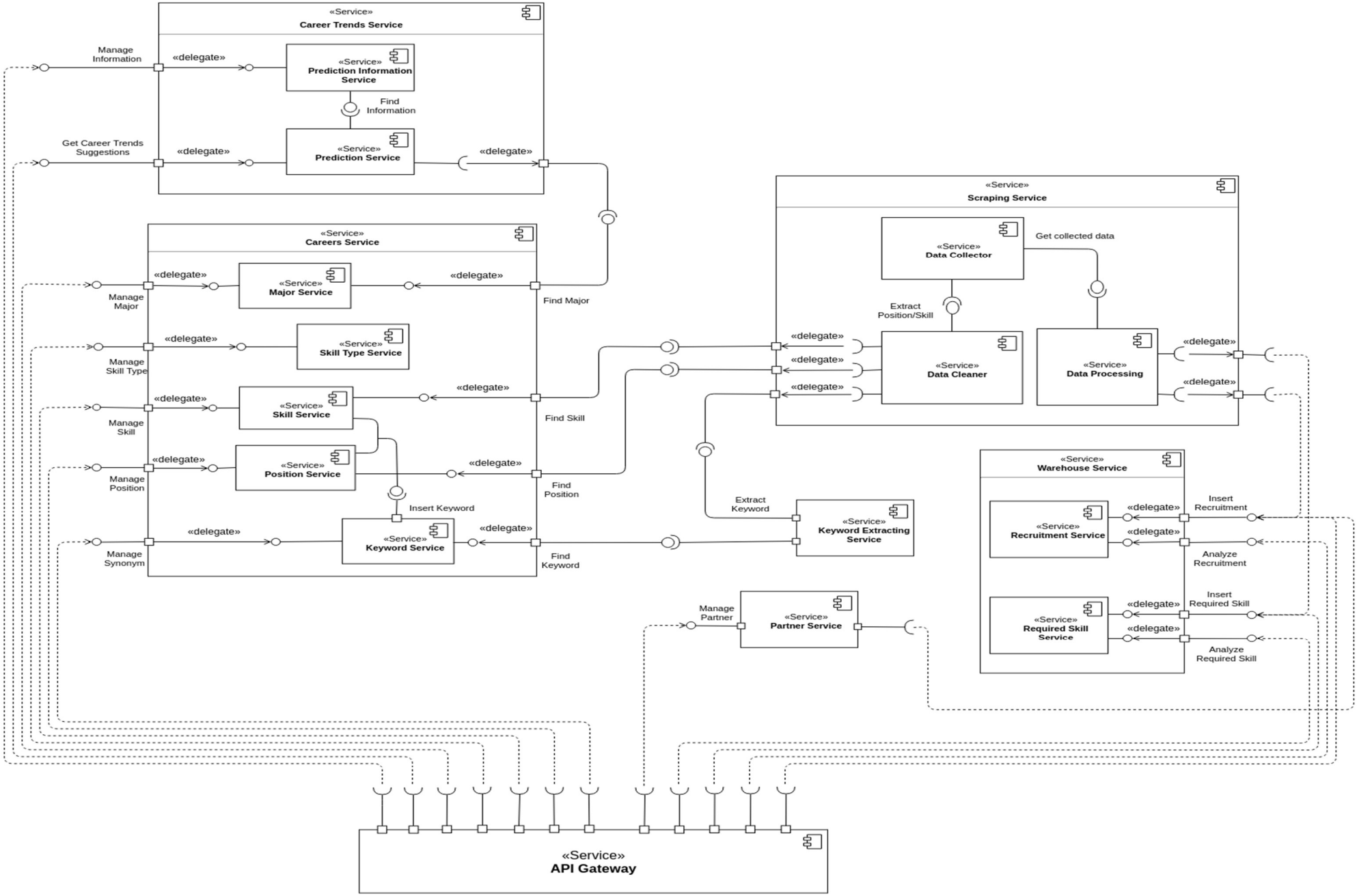


Figure 27: Component Diagram

Component dictionary	
Component name	**Description**
Career Trends Service	A service that is responsible for managing information about careers and using that information to provide predictions about upcoming trends of careers. Incoming requests or outcome responses are delegated to its inner services, which include: Prediction Information Service and Prediction Service.
Prediction Information Service	A service that is responsible for managing predefined types of information that is necessary when making predictions about career trends.
Prediction Service	A service that is responsible for using provided information to make predictions about career trends. Results are ported through the Career Trends Service so that other services in the CTSA system are able to obtain.
Careers Service	A service that is responsible for managing majors, positions, skill types, skills and keywords. This service is also responsible for providing interfaces for finding those elements. Incoming requests or outcome responses are delegated to its inner services, which include: Majors Service, Position Service, Skill Type Service, Skill Service and Keyword Service.
Major Service	A service that is responsible for managing majors and providing interfaces for finding majors. *Majors in the CTSA system are careers' fields such as Information Technology, Medical, Economy, etc.*
Position Service	A service that is responsible for managing positions and providing interfaces for finding positions. *Positions in the CTSA system are a majors' titles. For example: positions in Information Technology are: Developer, Tester, Project Manager, etc.*
Skill Type Service	A service that is responsible for managing skill types and providing interfaces for finding skill types. *Skill Types in the CTSA system are majors' groups of skills. For example: positions in Information Technology are: Programming Languages, Testing Skills, Deployment Methods, etc.*

Skill Service	A service that is responsible for managing skills and providing interfaces for finding skills. *Skills in the CTSA system are majors' skills. For example: positions in Information Technology are: Java, C#, JavaScript, Automation Testing, etc.*
Keyword Service	A service that is responsible for managing keywords and their synonyms, as well as providing interfaces for finding keywords. Synonyms are also identified as keywords. But, keywords that contains synonyms are called *root keywords*.
Scraping Service	A service that is responsible for collecting, cleaning data from recruitment websites, and storing cleaned data to the CTSA system's storages. Incoming requests or outcome responses are delegated to its inner services, which include: Data Collector, Data Cleaner and Data Processor.
Data Collector	A component that is responsible for collecting data from recruitment websites, receiving and producing processed data for storing.
Data Cleaner	A component that is responsible for receiving and cleaning raw data.
Data Processor	A component that is responsible for transforming cleaned data into predefined data structure that is possible for storing, then saves the qualified data to the CTSA system's storage.

Table 21: Component Data Dictionary

IV. Detailed Description

4.1. Class Diagram[7]

[7] OMG UML (August 2018), OMG Unified Modeling Language ™ (OMG UML) Superstructure

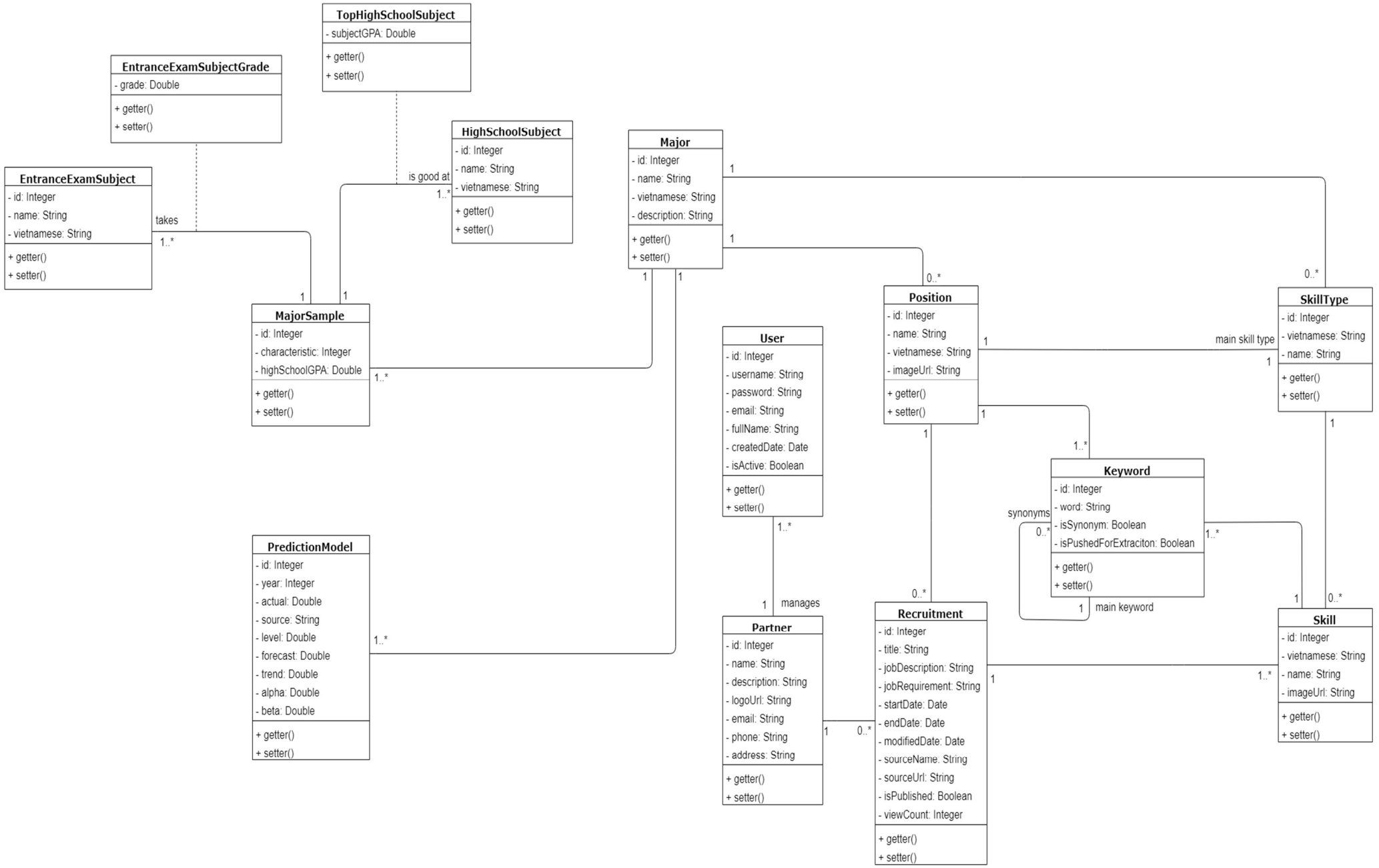


Figure 28: Class Diagram

Class Name	Mapping column with Conceptual Diagram	Description
Major	Major	Contain the major information.
MajorSample	MajorSample	Contain the sample data of major.
HighSchoolSubject	HighSchoolSubject	Contain the high school subject information.
TopHighSchoolSubject	N/A	Not exist in conceptual diagram. But needed in class diagram to contain the result of high school subject.
EntranceExamSubject	EntranceExamSubject	Contain the entrance exam subject information.
EntranceExamSubjectGrade	N/A	Not exist in conceptual diagram. But needed in class diagram to contain the result of entrance exam subject.
Position	Position	Contain the position information.
SkillType	SkillType	Contain the skill type information.
Skill	Skill	Contain the skill information.
Keyword	Keyword	Contain the keyword information.
PredictionModel	PredictionModel	Contain the prediction information.
Recruitment	Recruitment	Contain the recruitment information.
User	User	Contain the user information.
Partner	Partner	Contain the partner information.

Table 22: Class dictionary

4.2 Class Diagram Explanation

4.2.1. Major

Attribute

Attribute	Type	Visibility	Description
MajorID	Integer	Private	Unique identifier of a major
Name	String	Private	Name of major
Vietnamese	String	Private	Name of major in Vietnamese
Description	String	Private	Description of major

Method

Method	Return Type	Visibility	Description
Getter	Major type	Public	Get attribute value
Setter	Void	Public	Set value of attribute

4.2.2. MajorSample

Attribute

Attribute	Type	Visibility	Description
MajorSampleID	Integer	Private	Unique identifier of a major sample
Characteristic	Integer	Private	Characteristic of pupil

HighSchoolGPA	Double	Private	GPA of high school

Method

Method	Return Type	Visibility	Description
Getter	MajorSample type	Public	Get attribute value
Setter	Void	Public	Set value of attribute

4.2.3. HighSchoolSubject

Attribute

Attribute	Type	Visibility	Description
HighSchoolSubjectID	Integer	Private	Unique identifier of a high school subject
Name	String	Private	Name of high school subject
Vietnamese	String	Private	Name of high school subject in Vietnamese

Method

Method	Return Type	Visibility	Description
Getter	HighSchoolSubject type	Public	Get attribute value
Setter	Void	Public	Set value of attribute

4.2.4. TopHighSchoolSubject

Attribute

Attribute	Type	Visibility	Description
SubjectGPA	Double	Private	GPA of high school subject

Method

Method	Return Type	Visibility	Description
Getter	TopHighSchoolSubject type	Public	Get attribute value
Setter	Void	Public	Set value of attribute

4.2.5. EntranceExamSubject

Attribute

Attribute	Type	Visibility	Description
EntranceExamSubjectID	Integer	Private	Unique identifier of an entrance exam subject
Name	String	Private	Name of entrance exam subject
Vietnamese	String	Private	Name of entrance exam subject in Vietnamese

Method

Method	Return Type	Visibility	Description
Getter	EntranceExamSubject type	Public	Get attribute value
Setter	Void	Public	Set value of attribute

4.2.6. EntranceExamSubjectGrade

Attribute

Attribute	Type	Visibility	Description
Grade	Double	Private	Grade of entrance exam subject

Method

Method	Return Type	Visibility	Description
Getter	EntranceExamSubjectGrade type	Public	Get attribute value
Setter	Void	Public	Set value of attribute

4.2.7. Position

Attribute

Attribute	Type	Visibility	Description
PositionID	Integer	Private	Unique identifier of a position
Name	String	Private	Name of position
Vietnamese	String	Private	Name of position in Vietnamese
ImageURL	String	Private	Image link

Method

Method	Return Type	Visibility	Description
Getter	Position type	Public	Get attribute value
Setter	Void	Public	Set value of attribute

4.2.8. SkillType

Attribute

Attribute	Type	Visibility	Description
SkillTypeID	Integer	Private	Unique identifier of a skill type
Name	String	Private	Name of skill type
Vietnamese	String	Private	Name of skill type in Vietnamese

Method

Method	Return Type	Visibility	Description
Getter	SkillType type	Public	Get attribute value
Setter	Void	Public	Set value of attribute

4.2.9. Skill

Attribute

Attribute	Type	Visibility	Description
SkillID	Integer	Private	Unique identifier of a skill
Name	String	Private	Name of skill
Vietnamese	String	Private	Name of skill in Vietnamese
ImageURL	String	Private	Image link

Method

Method	Return Type	Visibility	Description
Getter	Skill type	Public	Get attribute value
Setter	Void	Public	Set value of attribute

4.2.10. Keyword

Attribute

Attribute	Type	Visibility	Description
KeywordID	Integer	Private	Unique identifier of a keyword
Word	String	Private	Content of keyword
IsSynonym	Boolean	Private	Tell that keyword is a synonym of another root keyword
IsPushedForExtraction	Boolean	Private	Tell that keyword is synchronized to Elasticsearch

Method

Method	Return Type	Visibility	Description
Getter	Keyword type	Public	Get attribute value
Setter	Void	Public	Set value of attribute

4.2.11. PredictionModel

Attribute

Attribute	Type	Visibility	Description
PredictionModelID	Integer	Private	Unique identifier of a prediction model
Year	Integer	Private	Year of actual data
Actual	Double	Private	Value of actual data
Source	String	Private	Source of actual data
Level	Double	Private	Level of prediction
Forecast	Double	Private	Prediction for the next year
Trend	Double	Private	Trend of prediction
Alpha	Double	Private	Coefficient
Beta	Double	Private	Coefficient

Method

Method	Return Type	Visibility	Description
Getter	PredictionModel type	Public	Get attribute value
Setter	Void	Public	Set value of attribute

4.2.12. Recruitment

Attribute

Attribute	Type	Visibility	Description
RecruitmentID	Integer	Private	Unique identifier of a recruitment
Title	Integer	Private	Title of recruitment
JobDescription	String	Private	Job description of recruitment
JobRequirement	String	Private	Job requirement of recruitment
StartDate	Date	Private	Start date of recruitment
EndDate	Date	Private	End date of recruitment
ModifiedDate	Date	Private	Modified date of recruitment
SourceName	Double	Private	Name of website
SourceURL	Double	Private	Link to website
IsPublished	Boolean	Private	Tell that recruitment is published

ViewCount	Integer	Private	View count of recruitment

Method

Method	Return Type	Visibility	Description
Getter	Recruitment type	Public	Get attribute value
Setter	Void	Public	Set value of attribute

4.2.13. User

Attribute

Attribute	Type	Visibility	Description
UserID	Integer	Private	Unique identifier of a user
Username	String	Private	Unique account username of user
Password	String	Private	Account password of user
Email	String	Private	Email of user
FullName	String	Private	Full name of user
CreatedDate	Date	Private	Created date of user account
IsActive	Boolean	Private	Tell that user account is activated

Method

Method	Return Type	Visibility	Description
Getter	User type	Public	Get attribute value
Setter	Void	Public	Set value of attribute

4.2.14. Partner

Attribute

Attribute	Type	Visibility	Description
PartnerID	Integer	Private	Unique identifier of a partner
Name	String	Private	Name of partner
Description	String	Private	Description of partner
LogoURL	String	Private	Logo link of partner
Email	String	Private	Email of partner
Phone	String	Private	Phone number of partner
Address	String	Private	Address of partner

Method

Method	Return Type	Visibility	Description
Getter	Partner type	Public	Get attribute value
Setter	Void	Public	Set value of attribute

4.3 Interaction Diagram[8]

4.3.1. Get Suggestion

Summary:

- *This diagram shows process of guest gets suggestion after inputting all the necessary information.*

[8] OMG UML (August 2018), OMG Unified Modeling Language ™ (OMG UML) Superstructure

- *Described in use case G01.*

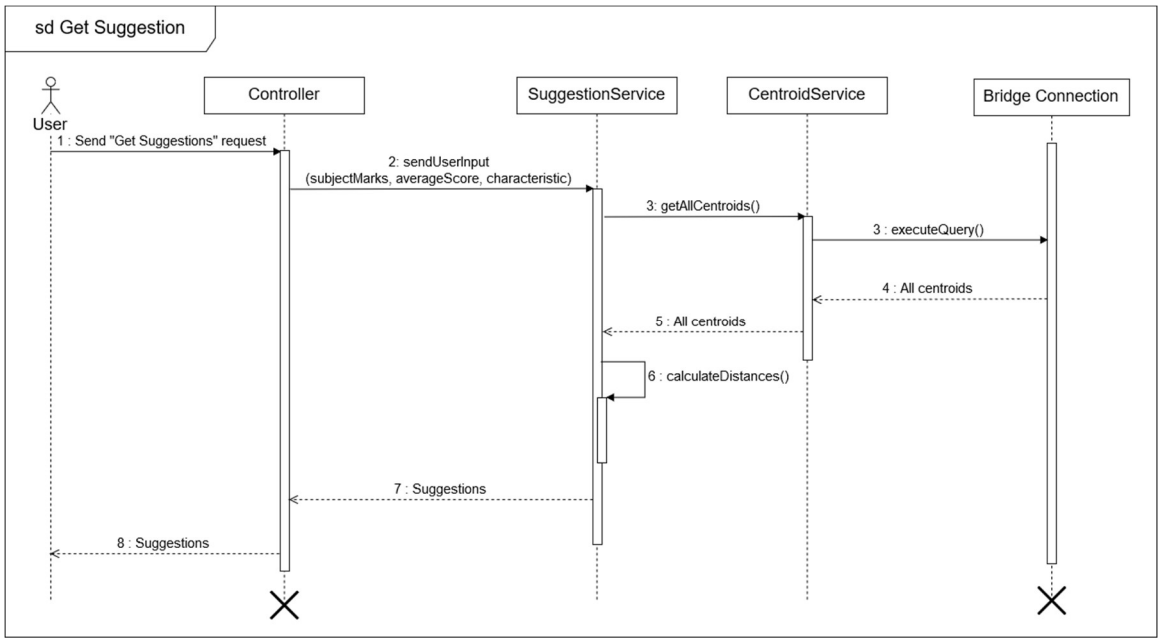


Figure 29: Sequence Diagram Get Suggestion

4.3.2. Scrapping Service

Summary:

- *This diagram shows how the system scraping websites to collect data.*
- *Described in use case T01*

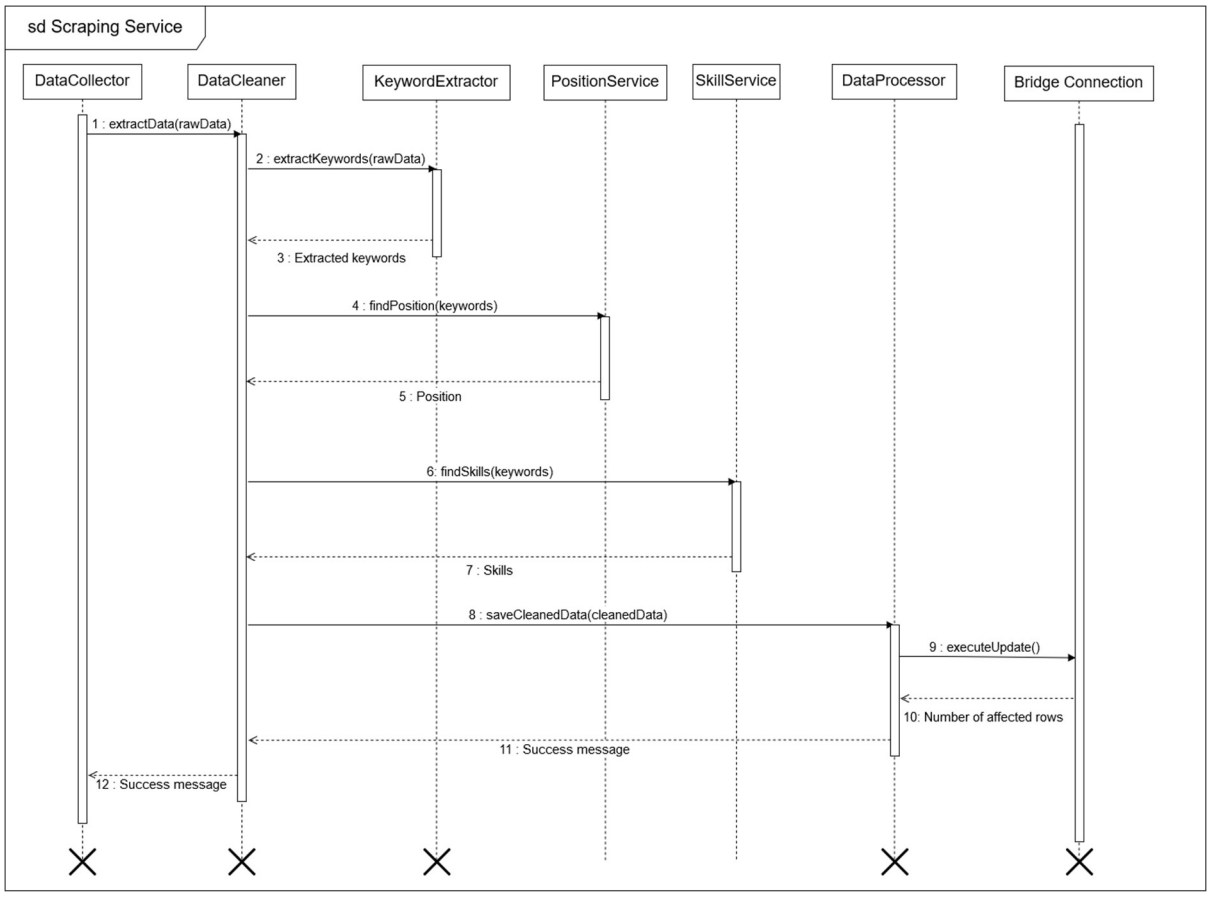


Figure 30: Sequence Diagram Scraping Service

4.3.3 Prediction Service

Summary:

- *This diagram shows process of guest gets prediction for a major.*
- *Described in use case G02.*

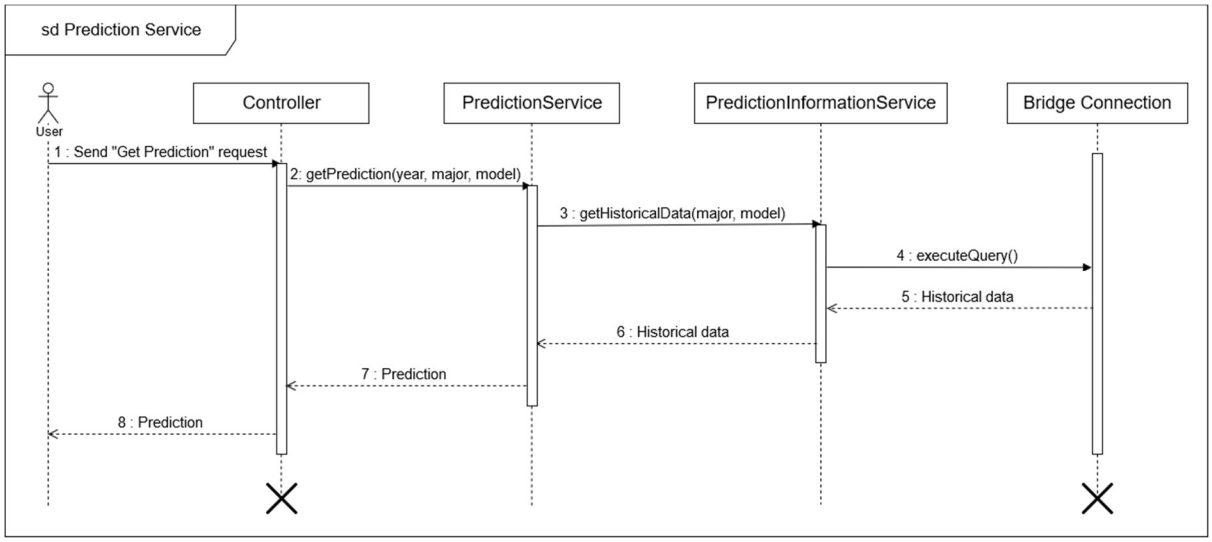


Figure 31: Sequence Diagram Prediction Service

4.3.4. Add Recruitment

Summary:

- *This diagram shows process of partner adds new recruitment*
- *Described in use case P01A*

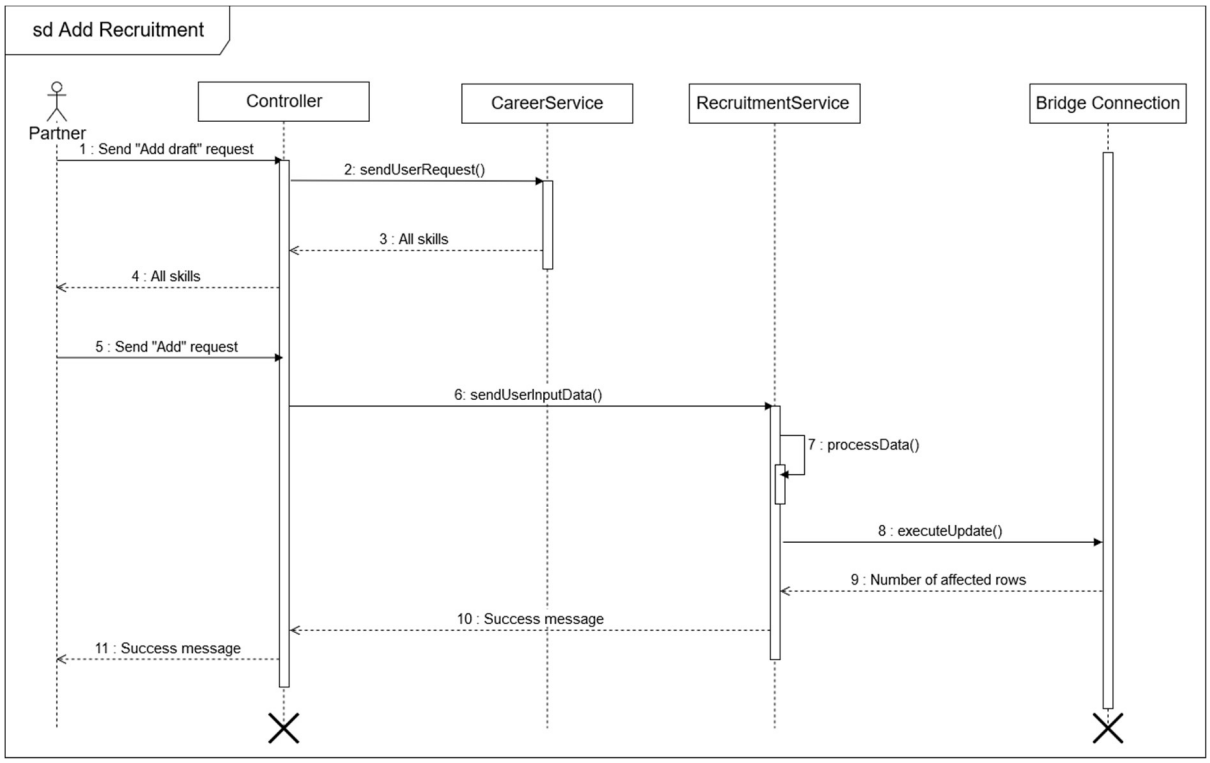


Figure 32: Sequence Diagram Add Recruitment

4.3.5. Edit Recruitment

Summary:

- *This diagram shows process of partner adds new recruitment*
- *Described in use case P01A*

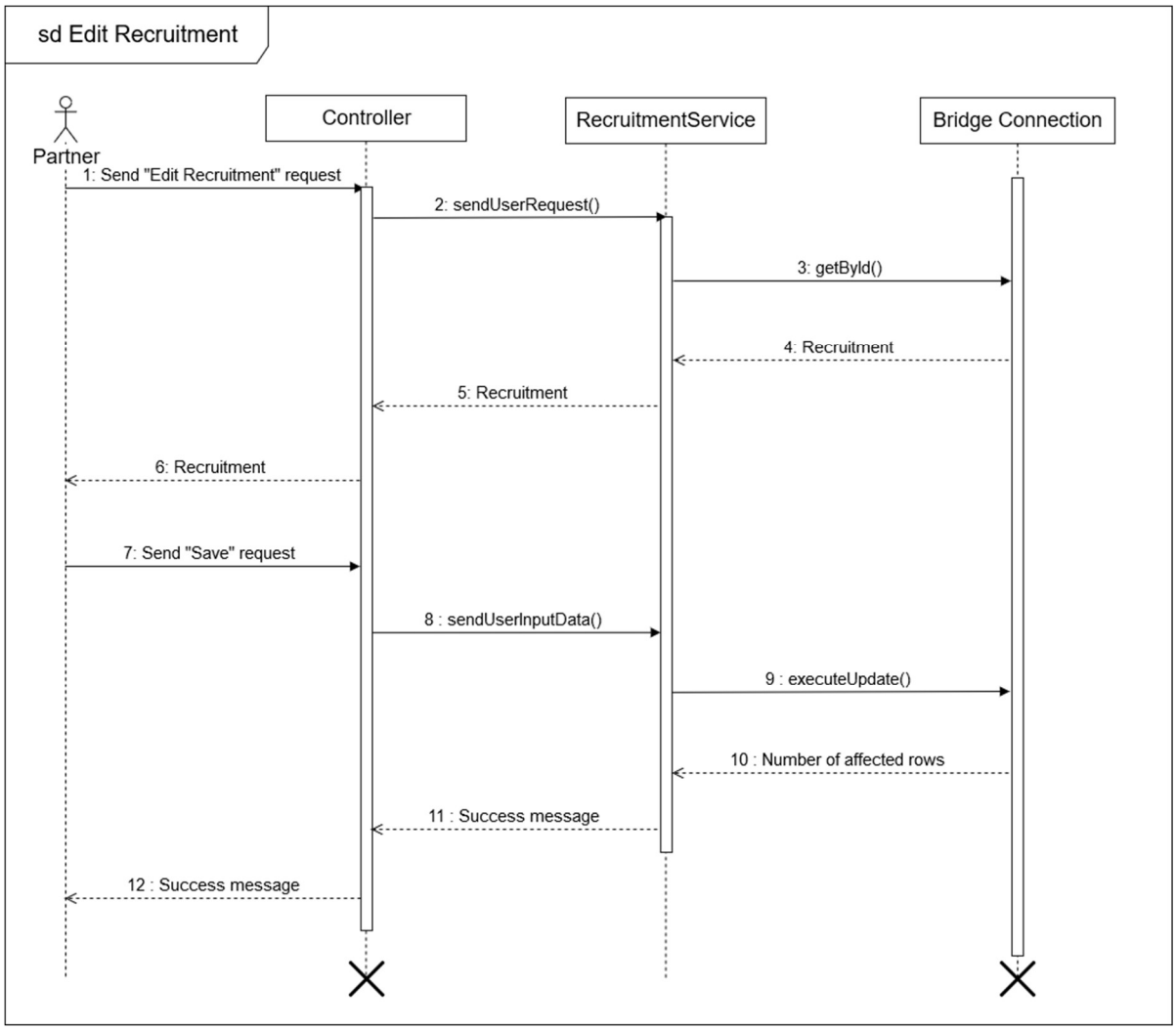


Figure 33: Sequence Diagram Edit Recruitment

4.3.6. Delete Recruitment

Summary: This diagram shows process of partner deletes recruitment

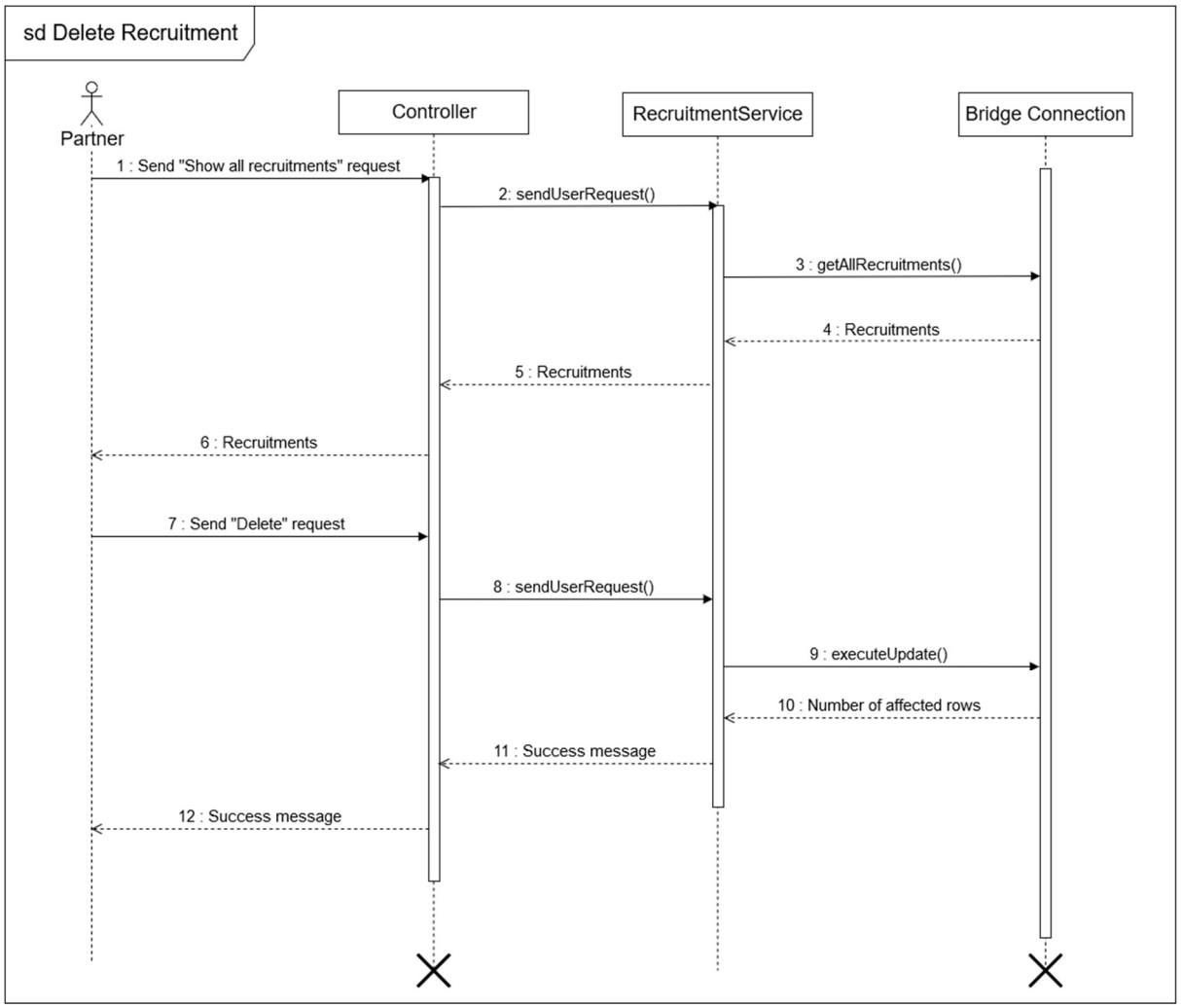


Figure 34: Sequence Diagram Delete Recruitment

4.3.7. Add Historical Data

Summary:

- *This diagram shows process of Staff creates new historical data.*
- *Described in use case* S03.

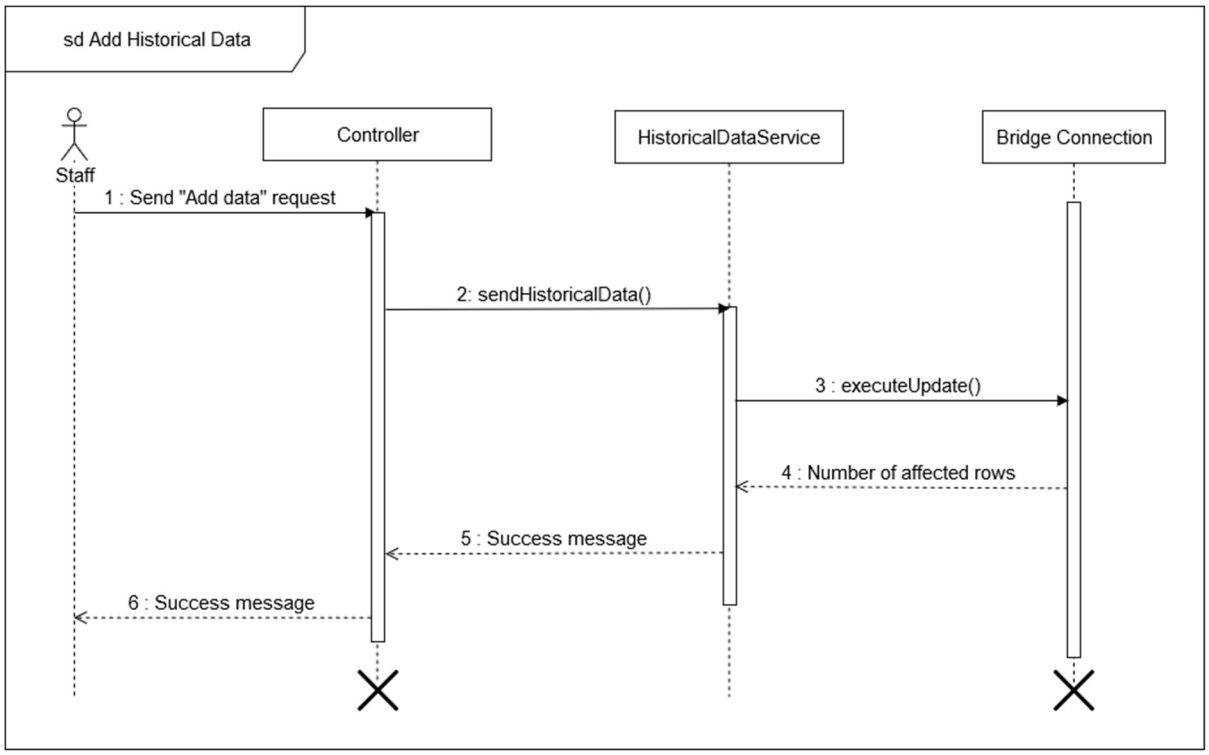


Figure 35: Sequence Diagram Add Historical Data

4.3.8. Edit Historical Data

Summary: This diagram shows process of Staff edits historical data

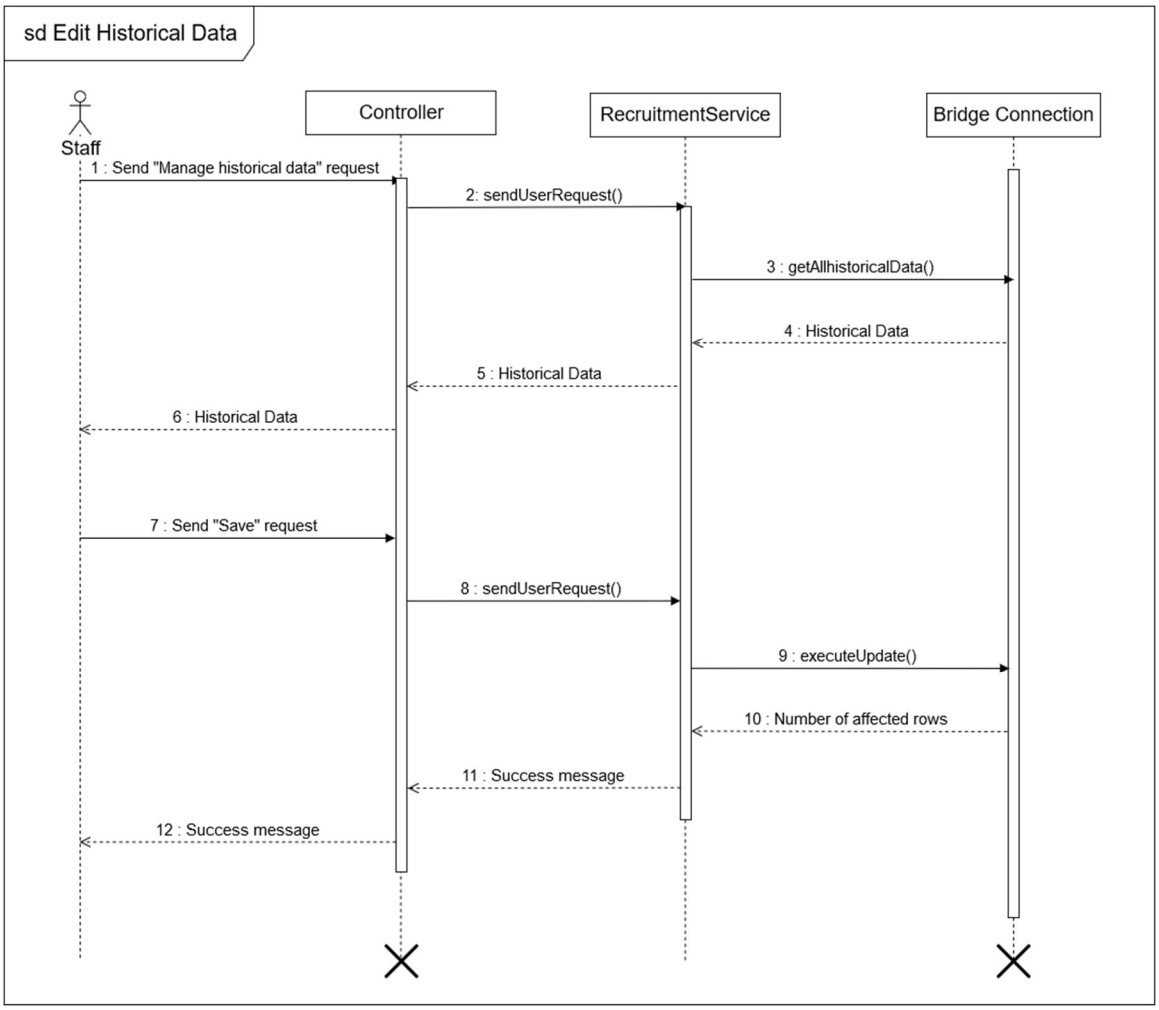


Figure 36: Sequence Diagram Edit Historical Data

4.3.9. Delete Historical Data

Summary: This diagram shows process of Staff deletes historical data

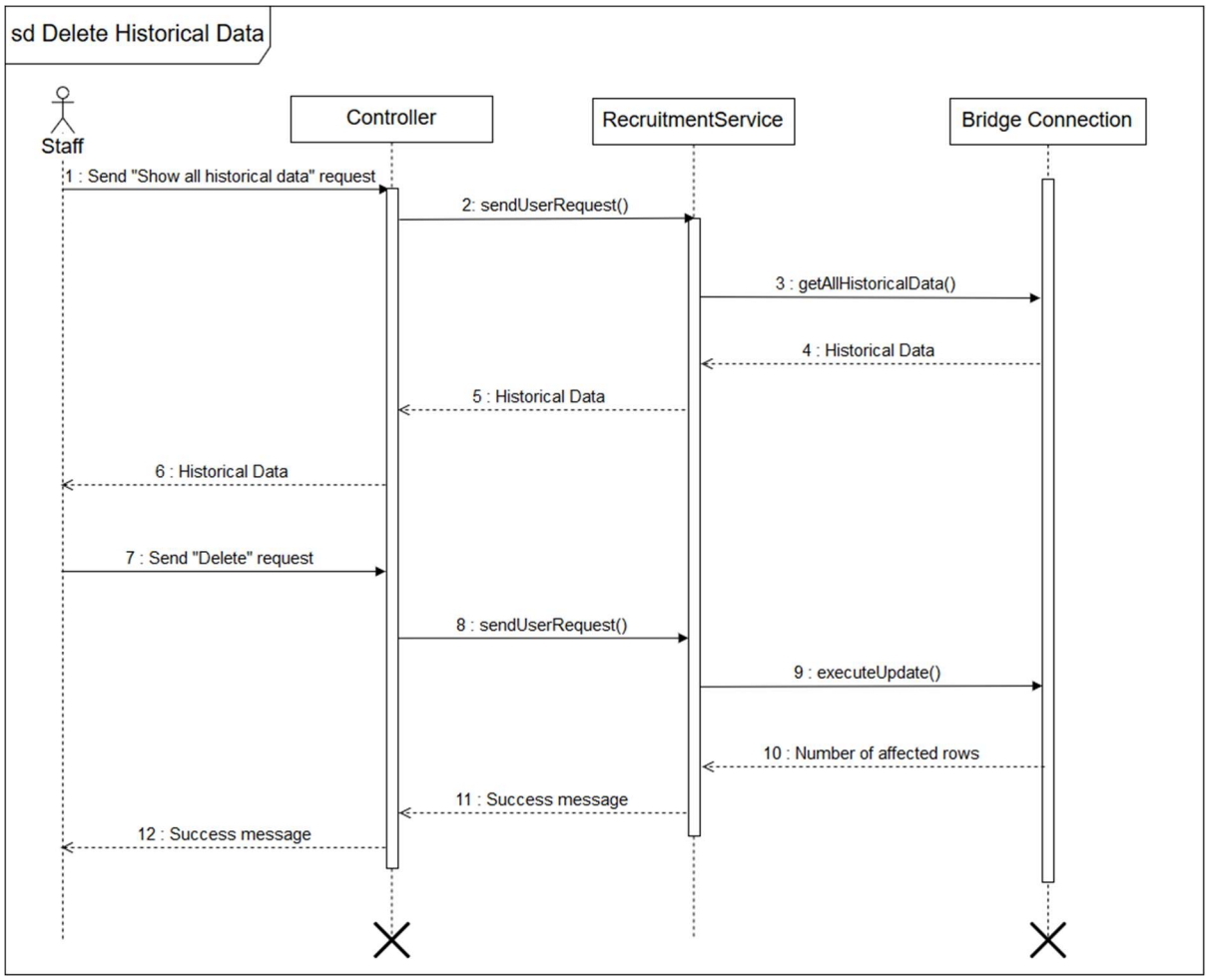


Figure 37: Sequence Diagram Delete Historical Data

4.3.10 Synchronize Data

- *This diagram shows how the timer synchronizes data between two databases on a daily basis*
- *Described in use case T03*

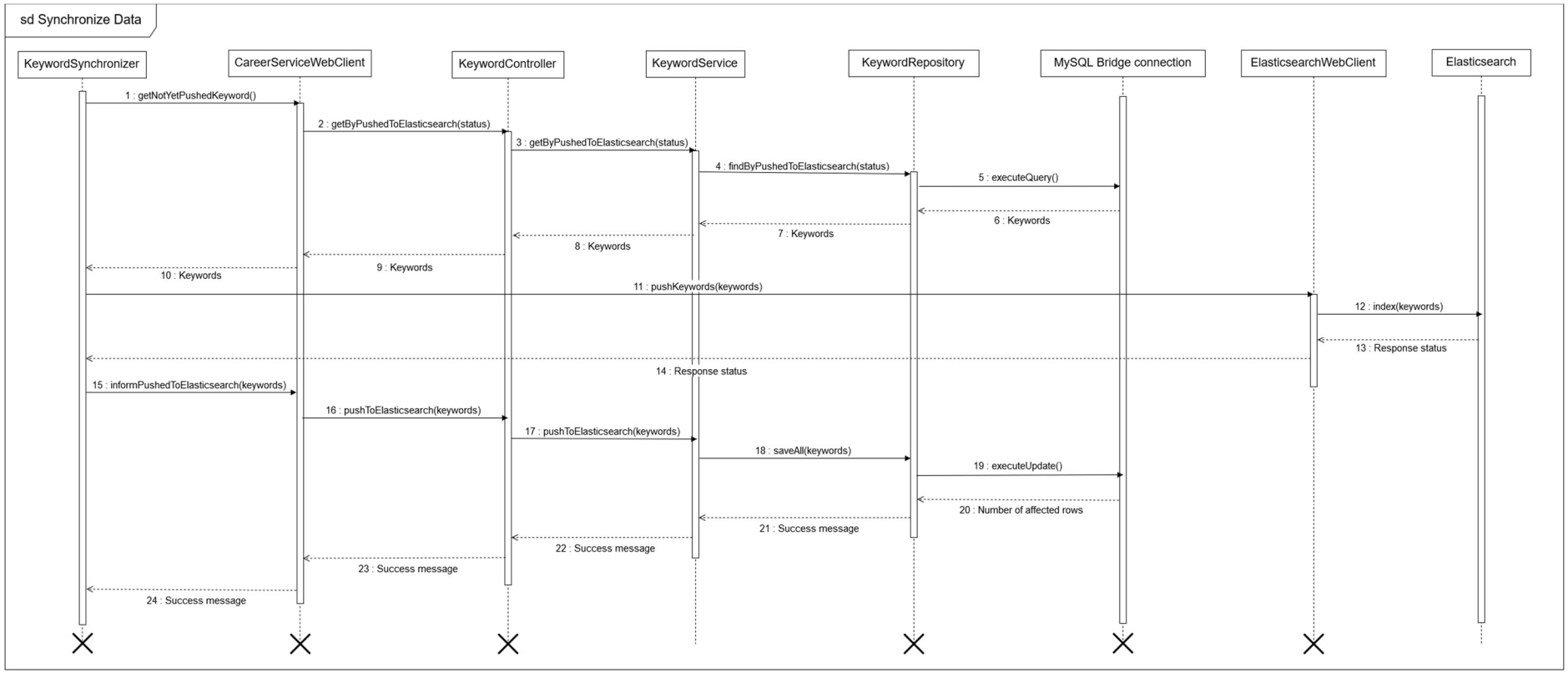


Figure 38: Sequence Diagram Synchronize Data

V. User Interface Design

Figure 39: Choose occupation index page

Buttons/Hyperlinks:

No	Function	Description	Validation	Outcome
1	Home Page	Home page link	N/A	Redirect to home page
2	High School Student	High School Student option	N/A	Display question form for high school student
3	College Pupil	College pupil option	N/A	Display question form for college pupil

Table 23: <Buttons/Hyperlinks> Choose occupation index page

Figure 40: Choose subjects index page

Buttons/Hyperlinks:

No	Function	Description	Validation	Outcome
1	High school subjects	High school subject buttons	N/A	Highlight the chosen buttons

Table 24: <Buttons/Hyperlinks> Choose subjects index page

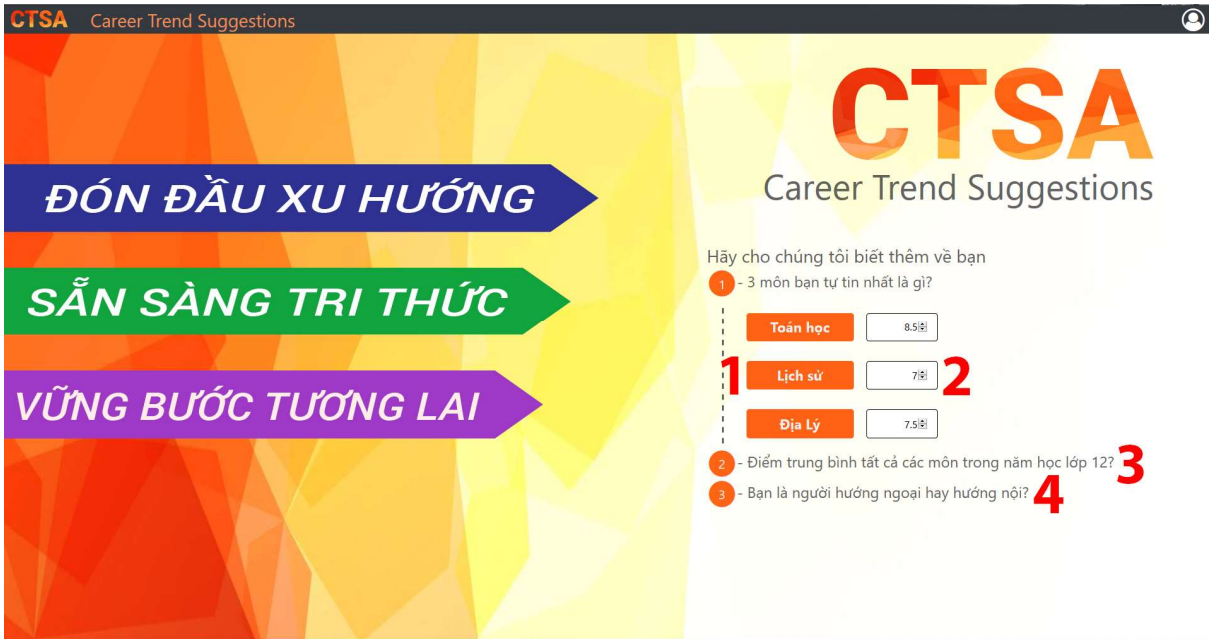


Figure 41: Input subject score index page

Fields

No	Field Name	Description	Read only	Mandatory	Control Type	Data Type	Length
1	Chosen Subjects	Subjects that are chosen by guest	Yes	Yes	Textbox	String	N/A
2	Subject score	Number text input for the subject score	No	Yes	Textbox	String	N/A

Table 25: <Fields> Input subject score index page

Buttons/Hyperlinks:

No	Function	Description	Validation	Outcome
3	2nd Question	2nd question option	N/A	Open the 2nd question of the form input
4	3rd Question	3rd question option	N/A	Open the 3rd question of the form input

Table 26: <Buttons/Hyperlink> Input subject score index page

Figure 42: Input average score index page

Fields

No	Field Name	Description	Read only	Mandatory	Control Type	Data Type	Length
2	Average subjects score	Number text input for the subject score	No	Yes	Textbox	String	N/A

Table 27: <Fields> Input average score index page

Buttons/Hyperlinks:

No	Function	Description	Validation	Outcome
1	1st Question	1st question option	N/A	Open the 1st question of the form input
3	3rd Question	3rd question option	N/A	Open the 3rd question of the form input

Table 28: <Buttons/Hyperlink> Input average score index page

Figure 43: Choose characteristic and get suggestions index page

Fields

No	Field Name	Description	Read only	Mandatory	Control Type	Data Type	Length
3	Characteristic Description	The description of the hovered characteristic button	Yes	Yes	Textbox	String	N/A

Table 29: <Fields> Choose characteristic and get suggestions index page

Buttons/Hyperlinks:

No	Function	Description	Validation	Outcome
1	Introvert	Choose characteristic Introvert	N/A	Highlight the clicked button display Get Suggestion button
2	Extrovert	Choose characteristic Extrovert	N/A	Highlight the clicked button, display Get Suggestion button
3	Get Suggestion	Take all the guest's input and send to the server	All textboxes in the form have been filled in	Redirect to result page

Table 30: <Buttons/Hyperlink> Choose characteristic and get suggestions index page

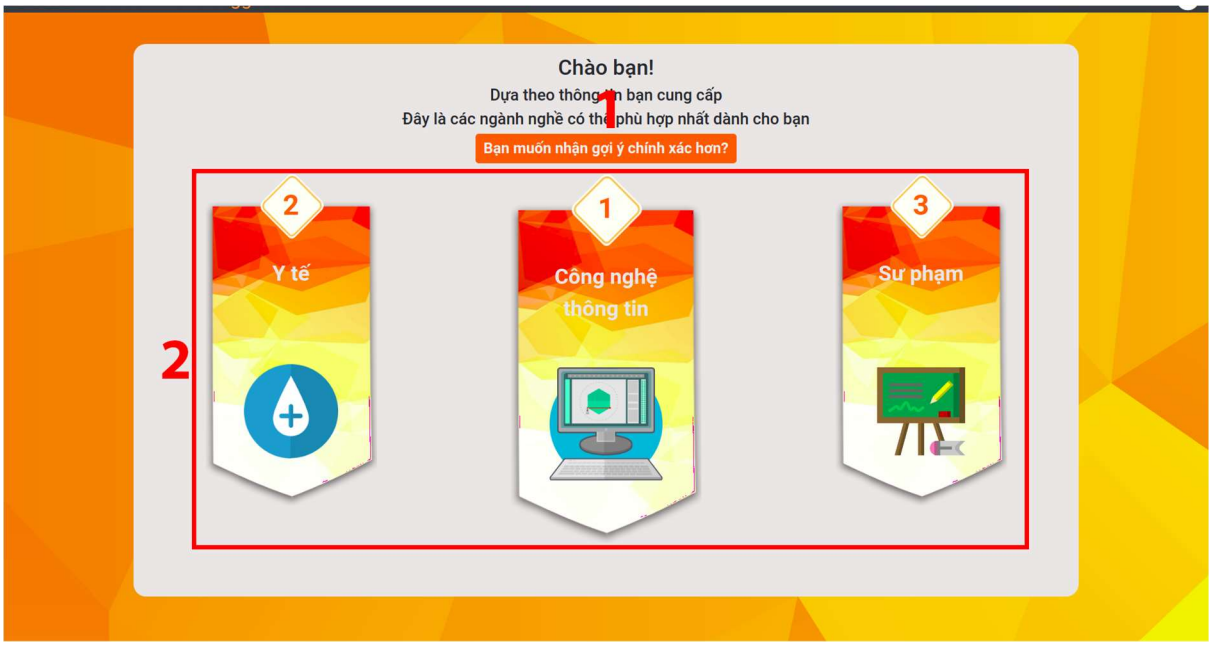


Figure 44: Result page

Buttons/Hyperlinks:

No	Function	Description	Validation	Outcome
1	Get more suggestion	Allow guest to input their college score for more suggestions	N/A	Open college score modal
2	Major Information	Get more information about the selected major	N/A	Open additional information modal

Table 31: <Buttons/Hyperlink> Result page

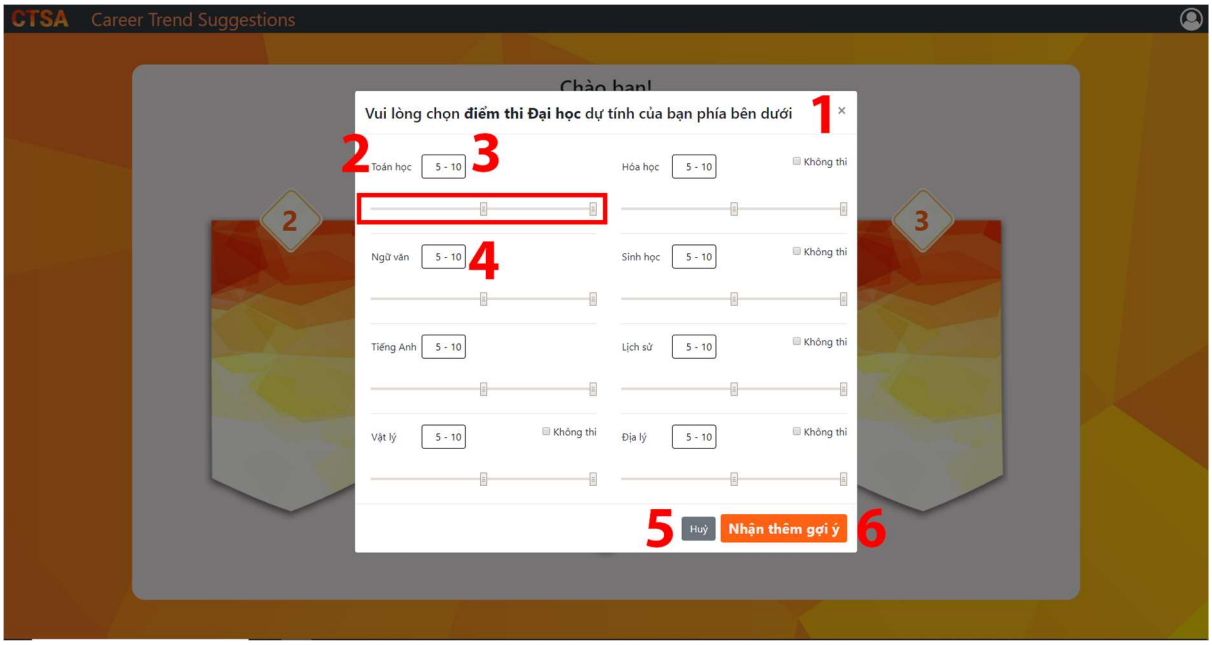


Figure 45: College Score page

Fields

No	Field Name	Description	Read only	Mandatory	Control Type	Data Type	Length
2	Subject Name	College Subject name	Yes	Yes	Text	String	N/A
3	College Score	Guest's college score for that subject	No	Yes	Textbox	String	N/A
4	Slider Score	Guest use slider to specify the range of the college score	No	Yes	Range	String	N/A

Table 32: <Fields> College Score page

Buttons/Hyperlinks:

No	Function	Description	Validation	Outcome
1	Close	Close the modal	N/A	Return to result page
5	Cancel	Close the modal	N/A	Return to result page
6	Get More Suggestion	Take all the guest's input and send to the server	All textboxes in the form have been filled in	Result page shows new result

Table 33: <Buttons/Hyperlink> College Score page

Figure 46: Statistic tab

Buttons/Hyperlinks:

No	Function	Description	Validation	Outcome
1	Prediction	Prediction tab	N/A	Open prediction tab
2	Recruitment	Recruitment tab	N/A	Open recruitment tab
3	Detail Ranking	Show detail ranking of selected top banner	N/A	Open detail ranking modal

Table 34: <Buttons/Hyperlink> Statistic tab

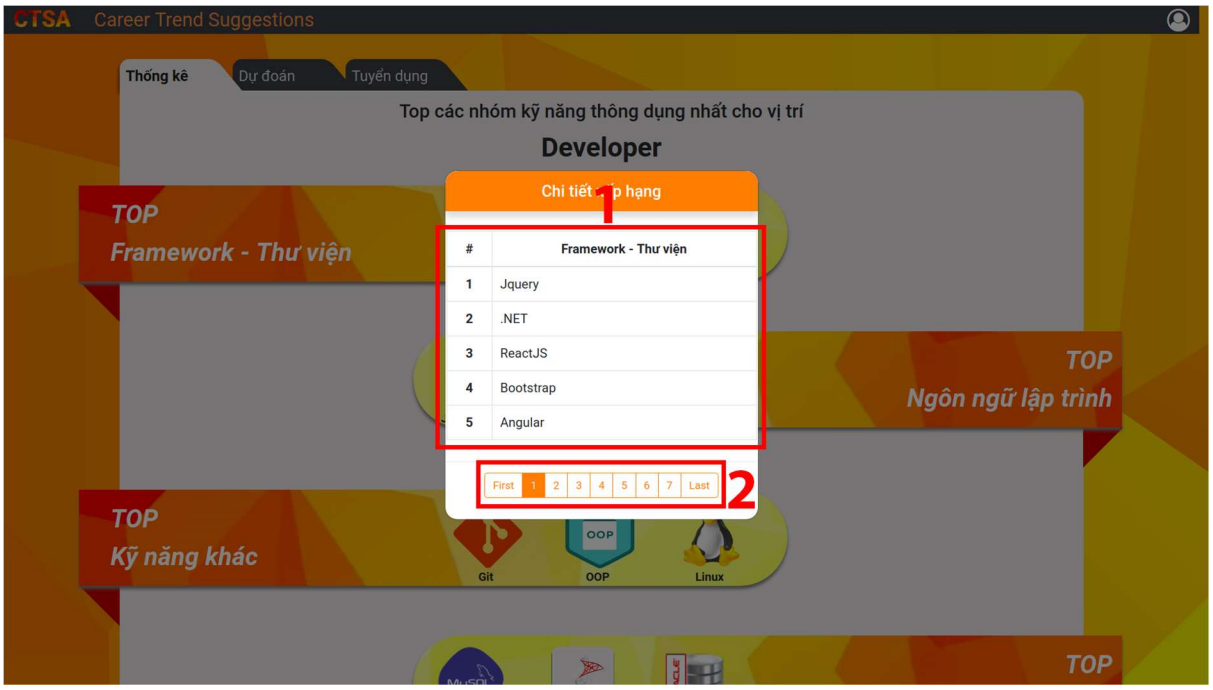


Figure 47: Detail ranking

Fields

No	Field Name	Description	Read only	Mandatory	Control Type	Data Type	Length
1	Detail Ranking	Table shows detail ranking based on the top banner user clicked	Yes	Yes	Table	N/A	N/A

Table 35: <Fields> Detail ranking

Buttons/Hyperlinks:

No	Function	Description	Validation	Outcome
2	Pagination	Pagination for detail ranking	N/A	Switch to respective page based on user clicked

Table 36: <Buttons/Hyperlink> Detail ranking

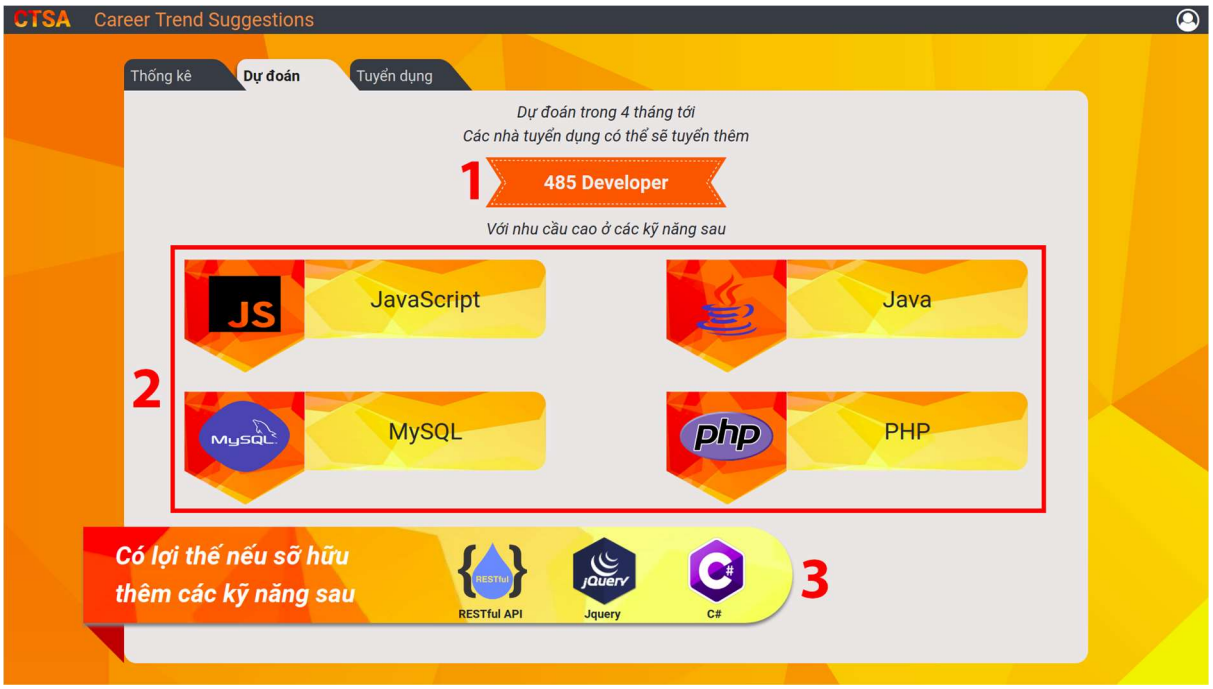


Figure 48: Prediction tab

Fields

No	Field Name	Description	Read only	Mandatory	Control Type	Data Type	Length
1	Predicted Job Number	Show predicted number for a job in the future	Yes	Yes	Text	String	N/A
2	Predicted Skill	Show predicted skill for a job in the future	Yes	Yes	Text	String	N/A
3	Additional skill	Show addition skill needed for a job in the future	Yes	Yes	Text	String	N/A

Table 37: <Fields> Prediction tab

VI. Database Design

6.1. Entity Relationship Diagrams[9]

Based on the business of the CTSA system, database is split into two modules:

- Database for High School Pupils suggestions
- Database for Career Trend Predictions

6.1.1. ERD for High School Pupils Suggestions

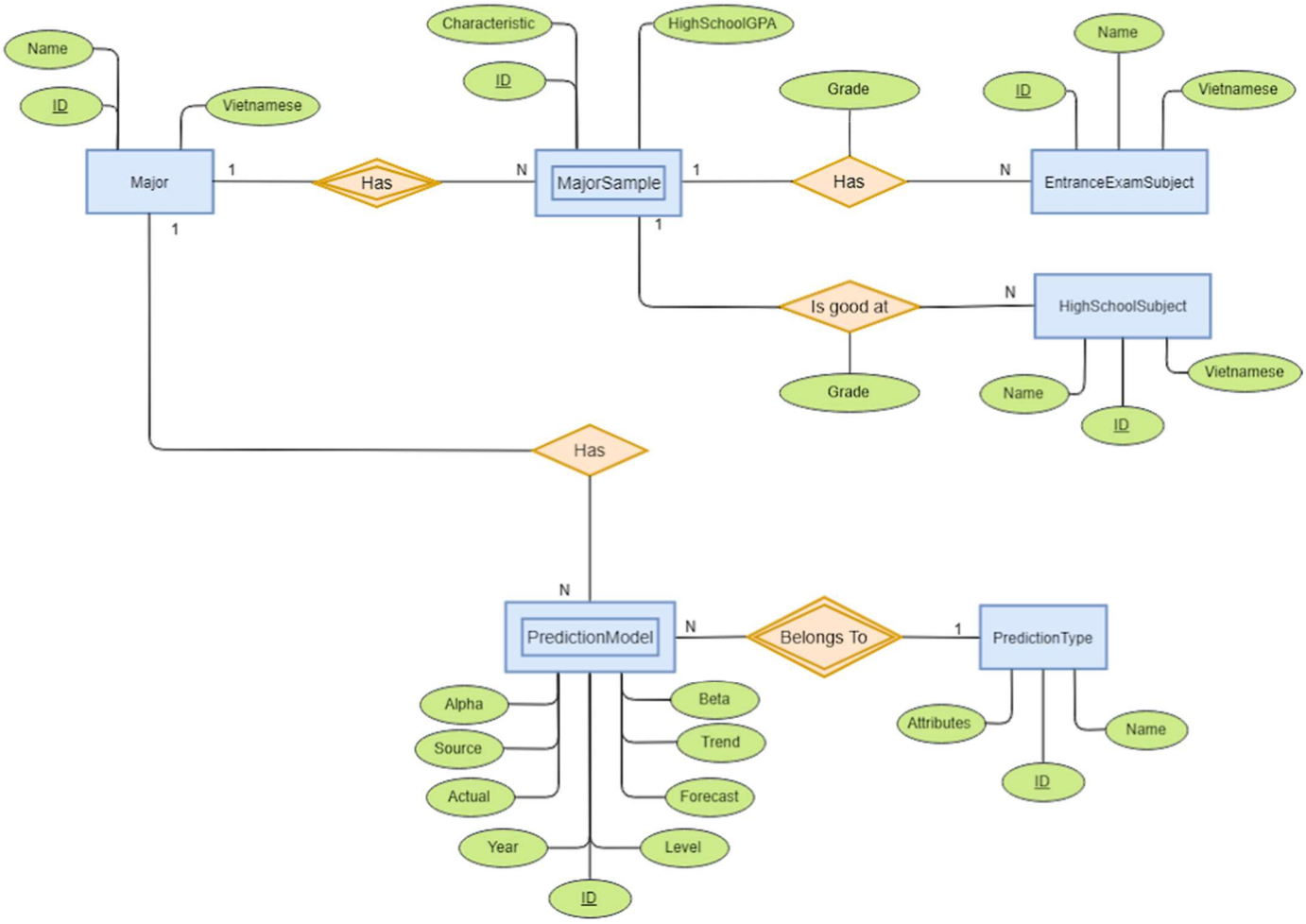


Figure 49: ERD for High School Pupils Suggestions

6.1.2. ERD for Career Trend Predictions

[9] Peter Chen (March 1976), The entity-relationship model - toward a unified view of data, ACM Transactions on Database Systems (TODS) - Special issue: papers from the international conference on very large data bases: September 22–24, 1975, Framingham, MA

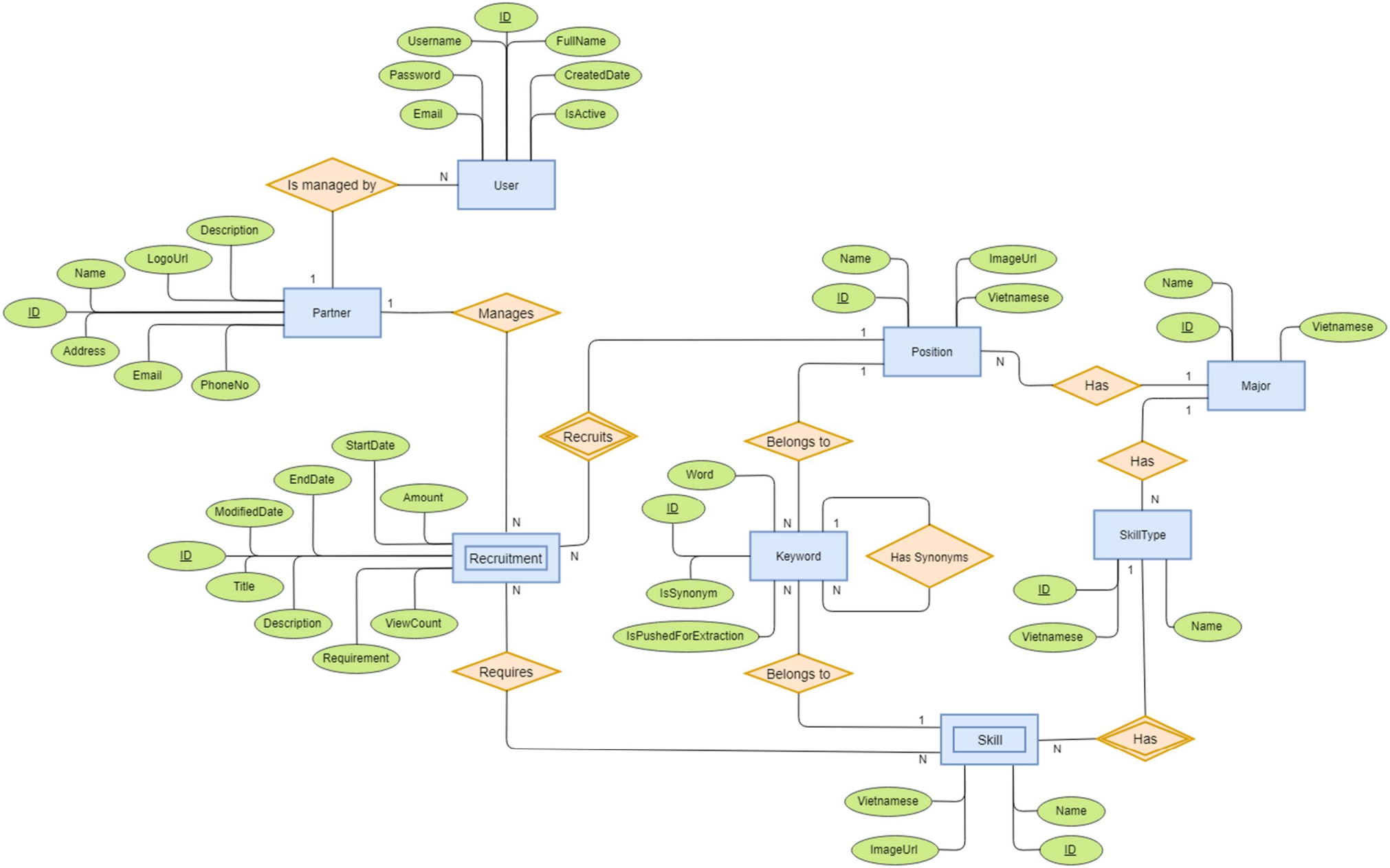


Figure 50: ERD for Career Trend Prediction

6.2. Data Dictionary

6.2.1. ERD for High School Pupils Suggestions

Entity Data dictionary	
Entity Name	**Description**
Major	An entity which represents for a major. *Majors in the CTSA system are careers' fields such as Information Technology, Medical, Economy, etc.*
MajorSample	An entity which represents for information collected from surveyed students. Information from this entity is used when it comes to statistics and analysis purposes.
HighSchoolSubject	An entity which represents for a subject in high school. *High school subjects in the CTSA system are input by staff, using subjects that appear in the general education program of Vietnam's Ministry of Education and Training.* *For examples: Mathematics, Literature, English, Chemistry, etc.*
EntranceExamSubject	An entity which represents for a subject that can be taken in the Vietnam's National High School Examination. *These subjects in the CTSA system are input by staff, using subjects that appear in the list of subjects which is announced by the Vietnam's Ministry of Education and Training.* *For examples: Mathematics, Literature, English, Physics, Chemistry, etc.*
PredictionType	An entity which contains information of unique attributes for a prediction model.
PredictionModel	An entity which represents for a criterion that needs to be analyzed for creating predictions. *For examples: Salary, Human Resources, etc.*

Table 38: High School Pupils Suggestions Entity Dictionary

Entity name	Attributes	Description	Domain	Null
Major	ID	ID of the major	int	No
	Name	Name of the major	nvarchar(255)	Yes
	Vietnamese	Vietnamese name of the major	nvarchar(255)	Yes
MajorSample	ID	ID of the major sample	int	No
	HighSchoolGPA	The number represents the student's high school GPA	double	Yes
	Characteristic	Characteristic of the student	int	Yes
EntranceExamSubject	ID	ID of the entrance exam subject	int	No
	Name	Name of the entrance exam subject	nvarchar(255)	Yes
	Vietnamese	Vietnamese name of the entrance exam subject	nvarchar(255)	Yes
HighSchoolSubject	ID	ID of the high school subject	int	No
	Name	Name of the high school subject	nvarchar(255)	Yes
	Vietnamese	Vietnamese name of the high school subject	nvarchar(255)	Yes
PredictionModel	ID	Id of the recruitment	int	No
	Alpha	The number represents the smoothing parameter for the level	double	Yes
	Beta	The number represents the smoothing parameter for the trend	double	Yes

	Source	Source contains actual value	Nvarchar(255)	Yes
	Trend	The number represents the estimate of the trend of the series at a specific time	double	Yes
	Actual	The number represents the real value	double	Yes
	Forecast	The number represents the predicted value	double	Yes
	Level	The number represents the estimate of the level of the series at a specific time	double	Yes
	Year	The number represents the current predicting year	int	Yes
PredictionType	ID	Id of the prediction type	int	No
	Name	Name of the prediction type	nvarchar(255)	Yes
	Attributes	Unique attribute of a prediction model	nvarchar(255)	Yes

Table 39: High School Pupils Suggestions Entities' details

6.2.2. ERD for Career Trend Predictions

Entity Data dictionary	
Entity Name	**Description**
Major	An entity which represents for a major. *Majors in the CTSA system are careers' fields such as Information Technology, Medical, Economy, etc.*
Position	An entity which represents for a position. *Positions in the CTSA system are a majors' titles. For example: positions in Information Technology are: Developer, Tester, Project Manager, etc.*
SkillType	An entity which represents for a group of skills. *Skill Types in the CTSA system are majors' groups of skills. For example: positions in Information Technology are: Programming Languages, Testing Skills, Deployment Methods, etc.*
Skill	An entity which represents for a skill. *Skills in the CTSA system are majors' skills. For example: positions in Information Technology are: Java, C#, JavaScript, Automation Testing, etc.*
Keyword	An entity which represents for a keyword. *A keyword in the CTSA system is a name or a synonym of a position or a skill.*
Recruitment	An entity which represents information of a single recruitment. For example: a recruitment can contain position, skills required that the company needs.
User	An entity which represents for an authenticated user.
Partner	An entity which represents for a company or an organization which is registered in CTSA system.

Table 40: Career Trend Predictions Data Dictionary

Entity name	Attributes	Description	Domain	Null
Major	ID	Id of the major	int	No
	Name	Name of the major	nvarchar(255)	Yes
	Vietnamese	Vietnamese name of the major	nvarchar(255)	Yes
SkillType	ID	Id of the skill type	int	No
	Name	Name of the skill type	nvarchar(255)	Yes
	Vietnamese	Vietnamese name of the skill type	nvarchar(255)	Yes
Skill	ID	Id of the skill type	int	No
	Name	Name of the skill type	nvarchar(255)	Yes
	Vietnamese	Vietnamese name of the skill type	nvarchar(255)	Yes
	ImageUrl	Image link of the skill	nvarchar(255)	Yes
Keyword	ID	Id of the keyword	int	No
	Word	The keyword	nvarchar(255)	Yes
	IsSynonym	Check if the word is a synonym or not	boolean	Yes
	IsPushedForExtraction	Check if the word is already pushed for extraction	boolean	Yes
Position	ID	Id of the position	int	No
	Name	Name of the position	nvarchar(255)	Yes
	Vietnamese	Vietnamese name of the position	nvarchar(255)	Yes
	ImageUrl	Image link of the position	nvarchar(255)	Yes
Recruitment	ID	Id of the recruitment	int	No

	Title	Title of the position that the company or organization is recruiting	nvarchar(255)	Yes
	Description	Description in the recruitment	nvarchar(255)	Yes
	Recruitment	Image link of the position	nvarchar(255)	Yes
	ViewCount	Number represents the total view count each recruitment has	int	Yes
	StartDate	Date represents the time when the recruitment is posted	date	Yes
	EndDate	Date represents the time when the recruitment is closed	date	Yes
	ModifiedDate	Date represents the time when the recruitment is modified	date	Yes
	Amount	The number represents how many employees which the company or organization is recruiting	int	Yes
Partner	ID	Id of the partner registered to CTSA	int	No
	Name	Name of the company or organization that is registered to CTSA	nvarchar(255)	Yes
	Description	Description of the company or organization that is registered to CTSA	nvarchar(255)	Yes
	Address	Address of the company or organization that is registered to CTSA	nvarchar(255)	Yes
	Email	Email of the company or organization that is registered to CTSA	nvarchar(100)	Yes
	PhoneNo	Phone number of the company or organization that is registered to CTSA	nvarchar(30)	Yes
	LogoUrl	Logo image link of the company or organization that is registered to CTSA	nvarchar(255)	Yes

User	ID	Id of the authenticated user	int	No
	Username	Username set up by user to login	nvarchar(255)	Yes
	Password	Password set up by user to login	nvarchar(255)	Yes
	Email	Email set up by user	nvarchar(100)	Yes
	FullName	The first and last name of the user	nvarchar(100)	Yes
	CreatedDate	The date represents the time when the account was created	date	Yes
	IsActive	Check if the account is currently active or disabled	boolean	Yes

Table 41: Career Trend Predictions Entities' details

VII. Algorithms

7.1. Keywords Extraction

7.1.1. Definition

Keyword extraction is the way to retrieve the predefined keywords if they exist in a text content.

7.1.2. Define Problem

The process of collecting data from recruitment websites is difficult due to the fact that job descriptions or requirements contain information about specific skills that are required, which is needed to be collected for analysis, along with a lot of unnecessary text. It is the CTSA system's responsibility to extract the information that is really necessary for analysis.

7.1.3. Requirement

Input a job description, the keyword extraction service of the CTSA system have to be able to identify which position is mentioned, and what skills are needed for that position, which is also mentioned in the job requirement.

7.1.4. Solution

To solve the problem, we apply the inverted index structure, which is designed to allow very fast full-text searches. Fortunately, there is an available open source service called Elasticsearch, which is already implemented the very basic forms and structure for inverted index. Using Elasticsearch is more preferable for us in doing this research since it helps saving a lot of implementation time. However, the original structure of inverted index cannot

completely satisfy our purposes; therefore, we made some minors modifications to the Elasticsearch's inverted index structure to meet our goal.

7.1.4.1. Original inverted index

Originally, an inverted index consists of a list of all the unique words that appear in any document. For each unique word, inverted index establishes a list of the documents in which it appears. For example: Given two documents containing the following:

1. *This job requires developers with experience in Java, MySQL, and Spring Framework.*
2. *Java and MongoDB are required for this job. Devs with experience in Spring Framework have lots of advantages.*

To create an inverted index, we split the document into separate and unique words. Later, these words are transformed into lowercase. Words which are pretty similar, or have the same root word, are transformed into their root word for simplicity. For example, "*requires*" and "*required*" are pretty similar since they have the same root word of "*require*"; therefore, both of them are index as "*require*" in inverted index structure. ***Table 16*** shows the result of the above example after processed by inverted index.

Term	Doc 1	Doc 2
this	**X**	**X**
job	**X**	**X**
require	**X**	**X**
developer	**X**	
with	**X**	**X**
experience	**X**	**X**
in	**X**	**X**
java	**X**	**X**
mysql	**X**	
and	**X**	**X**
spring	**X**	**X**
framework	**X**	**X**

mongodb		**X**
are		**X**
for		**X**
devs		**X**
have		**X**
lots		**X**
of		**X**
advantages		**X**

Table 42: Original inverted indexes

Now, if we search for "Java Developer", the result looks something like this:

Term	Doc 1	Doc 2
java	**X**	**X**
developer	**X**	

Table 43: Original inverted index result

7.1.4.2. CTSA inverted index

a. Index keywords, not documents

Originally, Elasticsearch indexes the documents and produces unique terms to apply full-text searches. On the contrary, we make Elasticsearch to index the keywords that is predefined, not the whole document. To be simple, we are trying to reverse the default way that Elasticsearch works on.

b. Index on n-grams terms

Instead of indexing each word independently, we index the whole term, then we could retain more of the context in which the words were used. For example, originally, if we input the term "Project Manager", it is inverted indexed to be {"project", "manager"}, instead of that, our inverted index structure now indexes the term "Project Manager" as "project manager", since we need to find "project manager" in the job description, not just "project" or "manager".

c. Documents are input:

Originally, inverted index accepts some words and returns documents that contains those terms. Our inverted index structure, on the contrary, requires input to be document and extracts keywords from the provided document.

7.1.5. Example

Given two different collections of predefined keywords as follow:

+ IT positions: **P** = {*Developer, Tester, Project Manager*}
+ IT skills: **S** = {*Java, MySQL, MongoDB*}

And the following document:

+ *This job requires developers with experience in Java, MySQL, and Spring Framework.*

CTSA inverted index search result:

["Developer", "Java", "MySQL", "Spring Framework"]

7.1.6. Complexity:

With 1.000 keywords and an input of 10.000 characters, it took:

– 125ms to extract keywords.

7.2. Majors Clustering

7.2.1. Definition

Majors Clustering is the way that CTSA system uses in order to identify and group collected samples from students into different clusters. Career suggestions for High Schools' students are later provided based on these clusters.

7.2.2. Define Problem

Given a collection of data about students from different majors. A single student' data includes:

1. What major is the student currently studying?
2. Subjects that the student feels most confident with in high school, with their marks.
3. Their results on the Vietnamese National High School Exam.

The CTSA system is responsible to put data into groups of majors based on their similarities.

7.2.3. Solution

a. Find groups

Take an example in which we only consider the marks of Mathematics and English among 30 surveyed students. With x-axis represents the mark of Mathematics and the y-axis for English, we end up having the graph in ***Figure 51.***

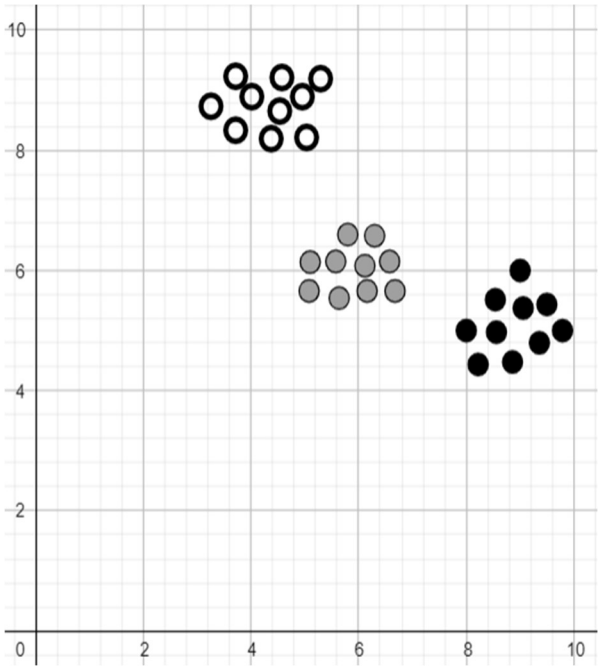


Figure 51

In ***Figure 51***, it is obvious that we are able to group 30 samples into 3 different groups, which we called a *cluster.* Each cluster represents a major in the CTSA system, to be more specific, in ***Figure 51***:

- Black: Represents students whose major is Medical.
- Grey: Represents students whose major is IT.
- White: Represents students whose major is Pedagogy.

With these identified clusters, for any random input of $\boldsymbol{K}(x_K, y_K)$, with:

- x_K: Represents the Mathematics' mark
- y_K: Represents the English's mark

The CTSA system is expected to identify which cluster that $\boldsymbol{K}(x_K, y_K)$ belongs to, therefore, is able to give a suggestion about which major should the pupil decide to study in college. The situation is represented in ***Figure 52***.

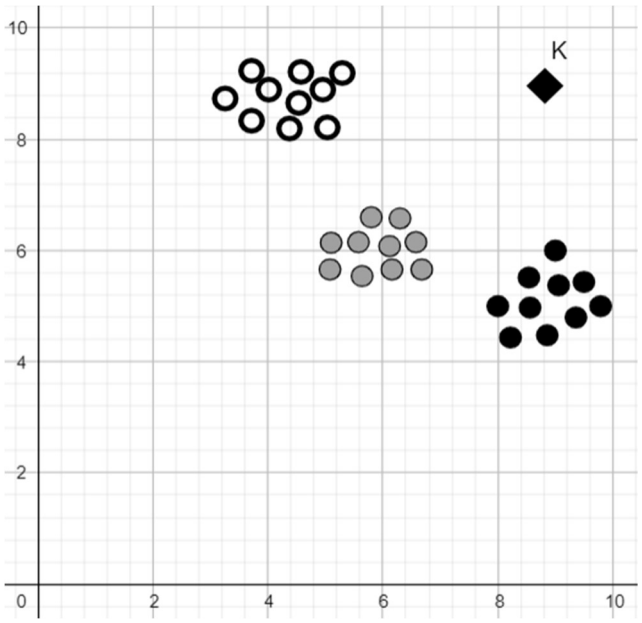


Figure 52

b. Find centroids

In order to identify which cluster ***K*** belongs to, we apply a naive method of finding distances between ***K*** and clusters. The shorter the distance, the more suitable for ***K*** to join the corresponding cluster. The idea is represented in ***Figure 53***.

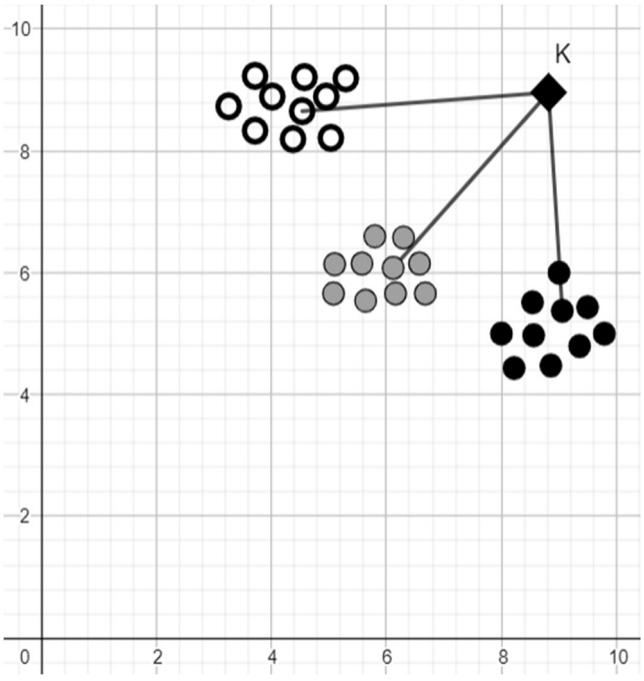


Figure 53

For simplicity, we find the central points of every cluster, which we call *centroids.* In ***Figure 54***, centroids are represented as diamonds.

These centroids are identified using a naive average calculation method. For example, in ***Figure 54***, $\boldsymbol{A}(x_A, y_A)$ represents the centroid for Medical cluster, in which:

- x_A : Average of all x-axises in Medical cluster.
- y_A : Average of all y-axises in Medical cluster.

Similarly, $\boldsymbol{B}(x_B, y_B)$ and $\boldsymbol{C}(x_C, y_C)$ represent centroids for IT and Pedagogy clusters, respectively.

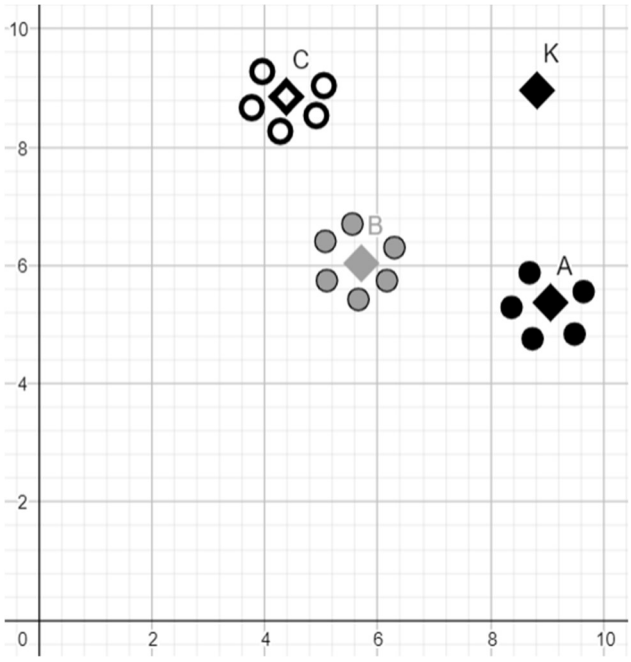


Figure 54

Now we only need to calculate the distances between $\boldsymbol{K}$ and the centroids for finding the most suitable cluster for $\boldsymbol{K}$.

c. Calculate distances

Following the previous example, as shown in ***Figure 55***, distances between $\boldsymbol{K}$ and $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ are annotated as dA, dB, and dC, respectively.

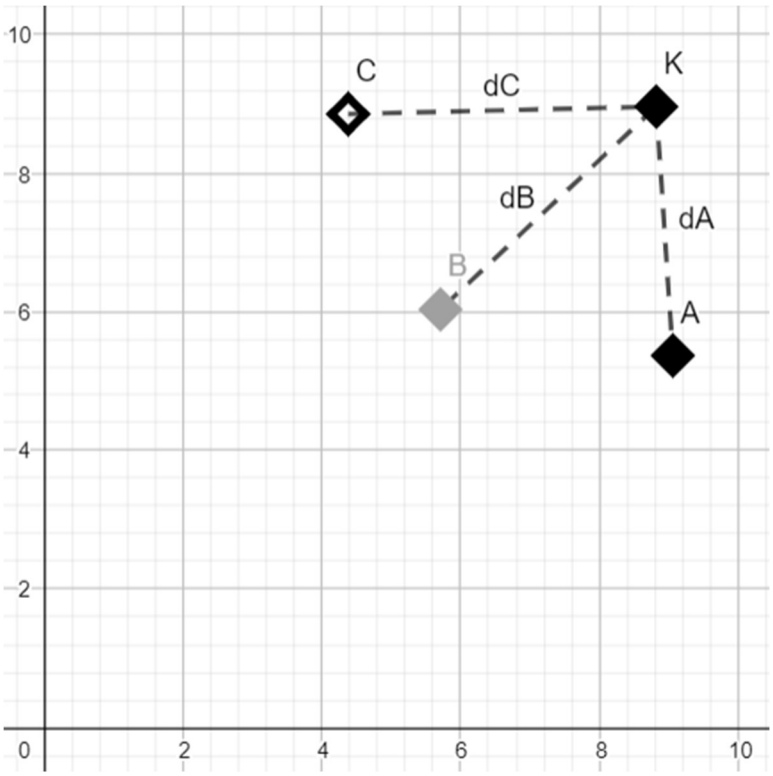


Figure 55

In order to calculate the distances, we take dA as an example.

With $\boldsymbol{K}(x_K, y_K)$ and $\boldsymbol{A}(x_A, y_A)$ are two points in two-dimensional space, then the distance (dA) from $\boldsymbol{K}$ to $\boldsymbol{A}$ is given by the Pythagorean formula:

$$dA = \sqrt{(x_K - x_A)^2 + (y_K - y_A)^2}$$

Applying the same formula to $\boldsymbol{B}$ and $\boldsymbol{C}$, we are able to acquire distances between $\boldsymbol{K}$ to every centroid. The comparison between those distances are not taken into consideration in this algorithm.

d. Clusters in n-dimensional space

Applying the above-mentioned solution to the CTSA system's situation, in which, each point consists of multiple parameters.

We denote a sample of a student as $\boldsymbol{S} = (S_1, S_2, S_3, \dots S_n)$. So that, with *m* samples of students who study major A, we have:

$$\boldsymbol{S_A} = \{S.A1, S.A2, S.A3, \dots, S.Am\}$$

Where:

$$S.A1 = (S.A1_1, S.A1_2, S.A1_3, \dots, S.A1_n),$$
$$S.A2 = (S.A2_1, S.A2_2, S.A2_3, \dots, S.A2_n),$$
$$S.A3 = (S.A3_1, S.A3_2, S.A3_3, \dots, S.A3_n),$$
$$\dots,$$
$$S.Am = (S.Am_1, S.Am_2, S.Am_3, \dots, S.Am_n)$$

The centroid of major A is denoted as $\boldsymbol{A} = (A_1, A_2, A_3, \dots A_n)$, where:

$$A_n = \frac{\sum_{i=1}^{m} S.Am_n}{m}$$

Now, the input **K** from previous example now can be denoted as

$$\boldsymbol{K} = (K_1, K_2, K_3, \dots K_n).$$

Finally, applying the Euclidean distance in n-dimensional space, the distance from $\boldsymbol{K}$ to $\boldsymbol{A}$ is given by the following formula:

$$d(K, A) = \sqrt{(K_1 - A_1)^2 + (K_2 - A_2)^2 + \cdots + (K_n - A_n)^2}$$

7.2.4. Complexity:

With 200 samples in 3 different majors. It took:

- 125ms to identify the centroids.
- 50ms to calculate the distance between a random point to a centroid.

7.3. Time Series Analysis

7.3.1. Definition

Time series analysis is a method to analyze and forecast the trends of majors based on their historical data.

7.3.2. Define Problem

As mentioned, the CTSA system is expected to provide the forecast of career trends based on historical data. It is necessary that the forecasts are made using a reliable method of analysis, and reliable sources of historical data.

7.3.3. Requirement

Historical data are input by staff, using reliable sources. The applied method is required provide a properly forecast about the trend of careers, based on analysis of those data from the past.

7.3.4. Solution

After researches, there are two methods that are taken into consideration: Linear Regression and Time Series Analysis.

a. Linear Regression

Linear Regression is "finding the best-fitting straight line for a set of data", based mostly on Statistics. This method's outcome is a regression line, which represents the linear equation that has the least amount of error (distance) between the line and the actual data. The tighter that the data points are around the regression line, the more reliable the linear equation is. ***Figure 56*** shows an example of the regression line with its actual data points.

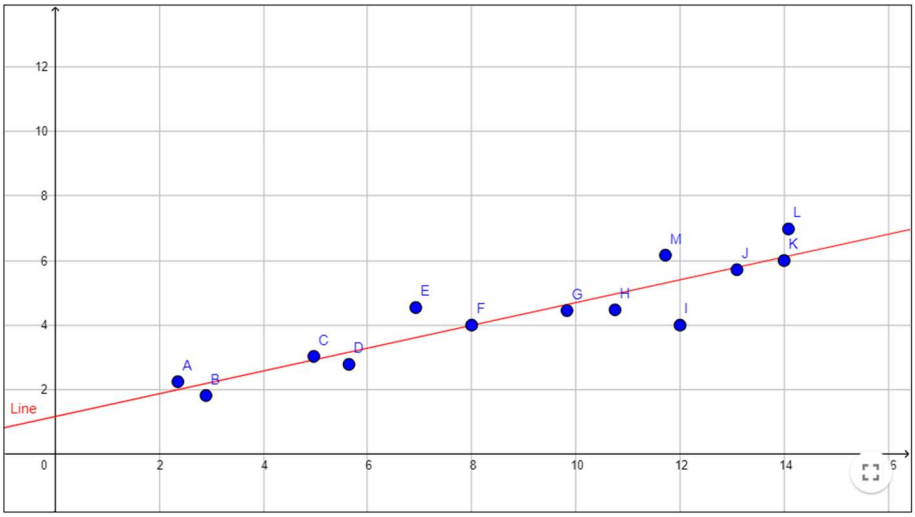


Figure 56

Once having this line, it is possible to make predictions about future outcomes if we only have data for one of the variables. However, linear regression analysis is only good for simple relationships like height and age, or time studying and GPA. Meanwhile, actual data of careers over time may consist of very specific characteristics related to trends. In order to identify and analyze these trends, it is necessary to use a time series analysis.

b. Time Series Analysis

Unlike Linear Regression, even Time Series Analysis is also based on Statistics' background, it is mostly use and more widely known as an Econometrics' method. In economics, this method is used to help identify the trend of business models through time, then give out predictions about the upcoming trend. These predictions help business owners to provide more suitable business strategies. ***Figure 57*** presents a typical example of a time series collection of data.

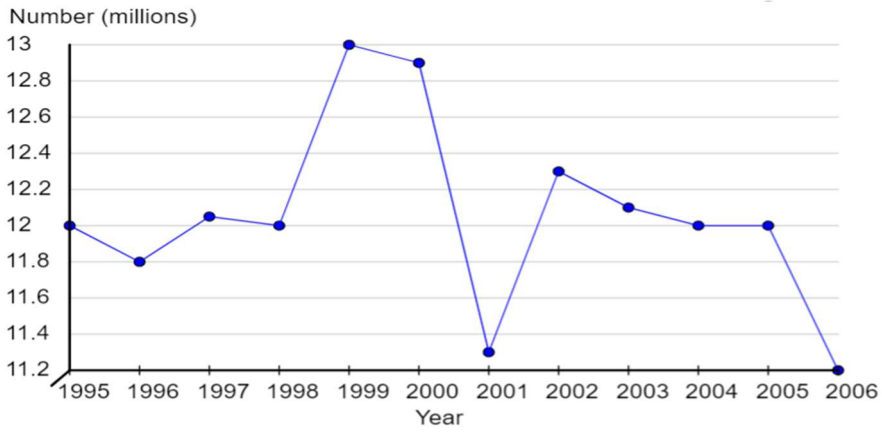


Figure 57

The basic objective of this method usually is to determine the pattern of the data in time series. Time series analysis is able to help analyze the trend of data, which is one of the most important characteristics of data that matches the need of the CTSA system.

c. Holt's Linear Trend Method

Holt (1957) used exponential smoothing to allow the forecasting of data with a trend. This method is considered as a Time Series Analysis method. It involves 1 forecast equation and 2 smoothing equations (one for the level and one for the trend):

- Forecast equation: $\boldsymbol{F_{t+h}} = L_t + hT_t$

- Level equation: $\boldsymbol{L_t} = \alpha A_t + (1-\alpha)(L_{t-1} + T_{t-1})$

- Trend equation: $\boldsymbol{T_t} = \beta(L_t - L_{t-1}) + (1-\beta)T_{t-1}$

Where:

- L_t denotes an estimate of the level of the series at time *t.*

- T_t denotes an estimate of the trend of the series at time *t.*

- α is the smoothing parameter for the level, $0 \leq \alpha \leq 1$.

- β is the smoothing parameter for the trend, $0 \leq \beta \leq 1$.

In comparison to the Linear Regression method, the forecast function is no longer flat but trending.

Figure 58 presents a visualization of applying Holt's Linear Trend method for analyzing trend of total pupils in Vietnam through years; the straight line represents the actual data, and the dotted line represents the forecasted data.

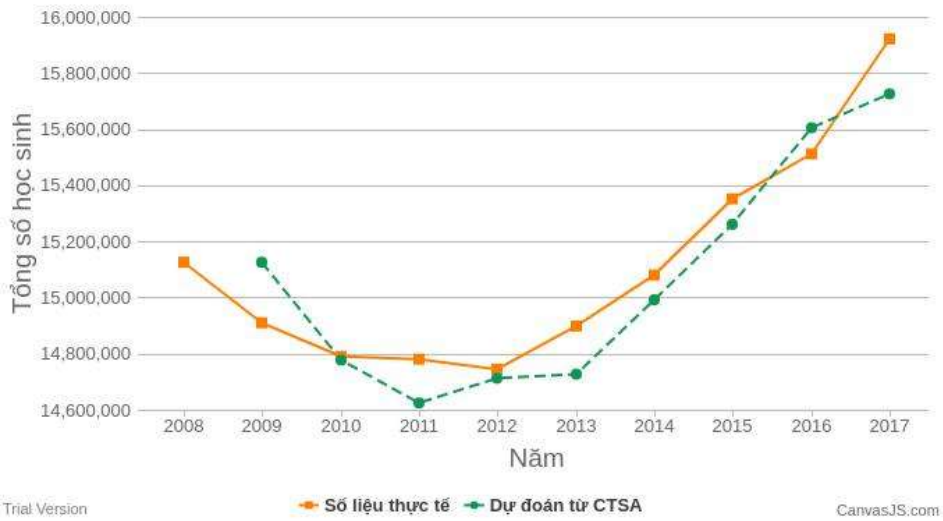


Figure 58

The critical part of the method, which is estimating the 2 smoothing parameters, α and β, however, is not mentioned in the formula. The Hill Climbing Algorithm in section 7.4 is taken into consideration so as to solve the problem of estimating the most properly values for these parameters.

7.3.5. Complexity:

With data from 2008 to 2016, it took:

- 1s to predict the data in 2017.
- 2s to predict the data in 2018.
- 2s to predict the data in 2019.

7.4. Hill Climbing Algorithm

7.4.1. Definition

Hill Climbing a method of Heuristic Search applied for mathematical optimization problems in the field of Artificial Intelligence.

7.4.2. Define Problem

As mentioned in the section 7.3, both smoothing parameters of the Time Series Analysis need to be estimated for providing the most properly trends prediction. With values of these parameters can be real numbers between 0 and 1, it is the CTSA system's responsibility to identify what values can be set for those parameters to achieve the most properly forecast equation.

7.4.3. Requirement

With historical data provided by the system's staff, the CTSA system is expected to provide a pair values of α and β , which is later used in the Time Series Analysis. This pair should be optimized so that the forecast equation of the Time Series Analysis is able to results in the more precise predictions the better.

7.4.4. Solution

After researches, there are two methods that are taken into consideration: Linear Regression and Time Series Analysis.

a. Heuristic Search

Heuristic Search is an AI search technique that applies heuristic for its moves. This method is uninformed (or blind) since it does not take the goal into account until it finds a path that leads to a node (or point) that satisfies the goal. Heuristic Search methods generally generate tests and pick the best solution from those tests. Steps of a general heuristic search method can be described as follow:

1. Generate a possible solution which can either be a point in the problem space or a path from the initial state.
2. Test to see if this possible solution is a real solution by comparing the state reached with the set of goal states.

3. If it is a real solution, return the result. Otherwise, repeat from step 1.

Generally, a heuristic search method provides a heuristic function, which will rank all the possible alternatives at any branching step in search algorithm based on the available information. It helps the algorithm to select the best route out of possible routes.

b. Hill Climbing Algorithm

Hill Climbing is one of Heuristic Search methods to solve certain optimization problems. This method starts with a sub-optimal solution and the solution is improved repeatedly until some condition is maximized.

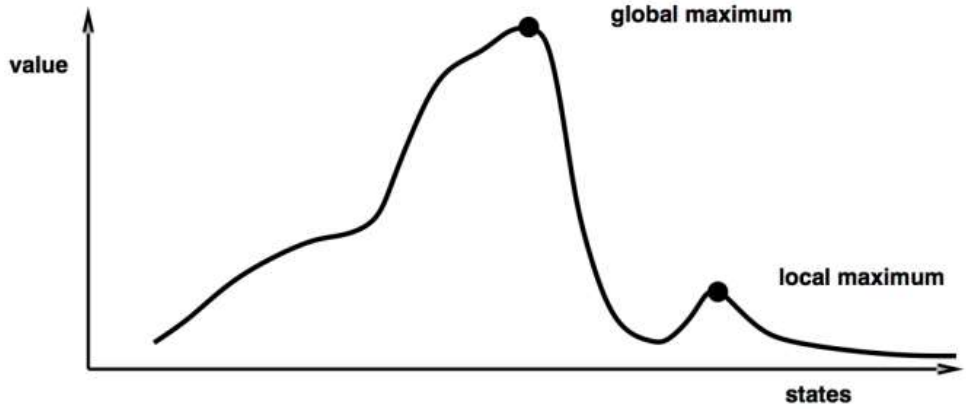


Figure 59

In Hill-Climbing technique, starting at the base of a hill, we walk upwards until we reach the top of the hill. In other words, we start with initial state and we keep improving the solution until its optimal.

This is a variation of a generate-and-test algorithm which discards all states which do not look promising or seem unlikely to lead us to the goal state. To take such decisions, it uses heuristics which indicates how close the current state is to the goal state.

Simple Hill climbing algorithm:

1. Define the ***current state*** as an *initial state*
2. Loop until the goal state is achieved or no more operators can be applied on the current state:
   - 2.1. Apply an operation to current state and get ***a new state***
   - 2.2. Compare the new state with the goal
   - ***2.3. Quit if the goal state is achieved***
   - 2.4. Evaluate new state with heuristic function and compare it with the current state
   - ***2.5. If the newer state is closer to the goal compared to current state, update the current state***

Obviously, it reaches the goal state with iterative improvements. In Hill-Climbing algorithm, finding goal is equivalent to reaching the top of the hill.

c. Estimating smoothing parameters based on MSE

A typical criterion that is widely used to prove the quality of the smoothing parameters is the Mean Squared Error (MSE).

For example, given collection of time series data with n samples, applying the Holt's Linear Trend method, we can create a test with $\alpha = \alpha_1$ and $\beta = \beta_1$, results are shown in ***Table 44***.

Time	Actual	Forecast	Level	Trend
t_1	A_1	F_1	L_1	T_1
t_2	A_2	F_2	L_2	T_2
t_3	A_3	F_3	L_3	T_3
...	...	...	...	...
t_n	A_n	F_n	L_n	T_n

Table 44

Therefore, MSE of the test is calculated using the following formula:

$$MSE = \frac{\sum_1^n (A_n - F_n)^2}{n}$$

Obviously, MSE of the Time Series Analysis method changes based on values of α and β. The smaller the MSE is, the more optimized these smoothing parameters are, and helps the Time Series Analysis method to provide more precise result.

The CTSA system applied Hill Climbing Algorithm in order to identify the best pair of α and β, based on the MSE that they provide.

The CTSA system's pseudocode for estimating each parameter:

```
initiate currentState
currentMSE ← MSE(currentState)
tryStepBackward ← MSE(currentState-step)
tryStepForward ← MSE(currentState+step)

if (tryStepBackward < currentMSE) then
       return moveBackward(currentState)
else if (tryStepForward < currentMSE) then
       return moveForward(currentState)
else
       return currentState
end if
```

```
function moveBackward(currentState)
       currentMSE ← MSE(currentState)
       nextMSE ← MSE(currentState – step)

              if (currentState – step > 0 & currentMSE > nextMSE)
       then
            return moveBackward(currentState – step)
       end if

       return currentState
end function
```

```
function moveForward(currentState)
       currentMSE ← MSE(currentState)
       nextMSE ← MSE(currentState + step)

              if (currentState + step < 1 & currentMSE > nextMSE)
       then
            return moveForward(currentState + step)
       end if

       return currentState
end function
```

7.3.5. Complexity:

With data from 2008 to 2016, it took:

- 2s calculate the smoothing parameters.

This page is intentionally left blank.

E. Software Test Documentation

1. Introduction

1.1. System Overview

- This section provides in detail all necessary information about test plans, test cases, test results, test environments, pass/fail criteria and risks estimation as well as a checklist to cover all possible cases of CTSA.

1.2. Test Approach

- Goal: To determine and discover all remain defects of CTSA.
- Method: Black-box testing.
- Size: System components.

2. Database Relationship Diagram

2.1. Physical Diagram

2.1.1. High School Pupils Suggestions

a) Physical Diagram

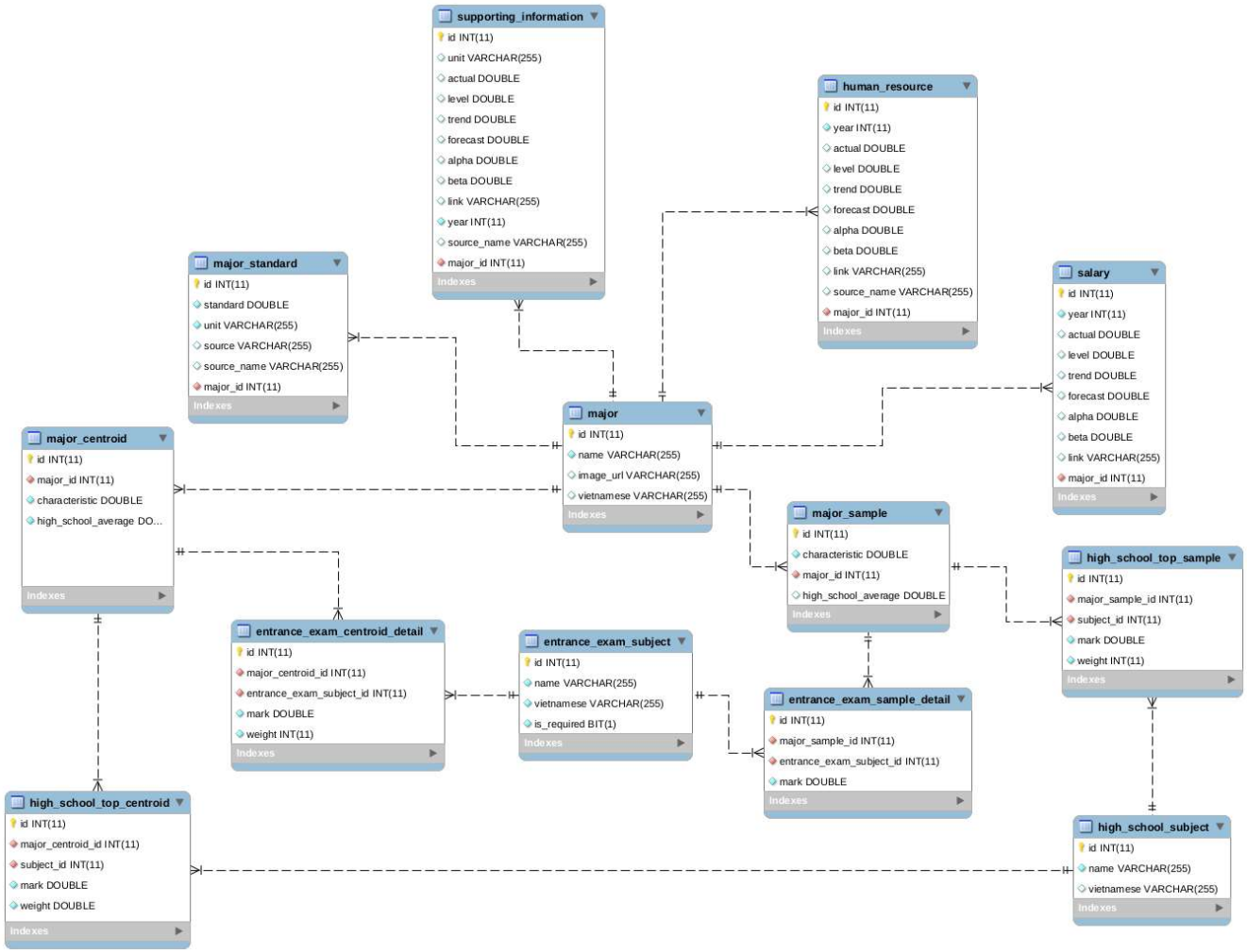

supporting_information
id INT(11)
unit VARCHAR(255)
actual DOUBLE
level DOUBLE
trend DOUBLE
forecast DOUBLE
alpha DOUBLE
beta DOUBLE
link VARCHAR(255)
year INT(11)
source_name VARCHAR(255)
major_id INT(11)
Indexes
human_resource
id INT(11)
year INT(11)
actual DOUBLE
level DOUBLE
trend DOUBLE
forecast DOUBLE
alpha DOUBLE
beta DOUBLE
link VARCHAR(255)
source_name VARCHAR(255)
major_id INT(11)
Indexes
major_standard
id INT(11)
standard DOUBLE
unit VARCHAR(255)
source VARCHAR(255)
source_name VARCHAR(255)
major_id INT(11)
Indexes
salary
id INT(11)
year INT(11)
actual DOUBLE
level DOUBLE
trend DOUBLE
forecast DOUBLE
alpha DOUBLE
beta DOUBLE
link VARCHAR(255)
major_id INT(11)
Indexes
major
id INT(11)
name VARCHAR(255)
image_url VARCHAR(255)
vietnamese VARCHAR(255)
Indexes
major_centroid
id INT(11)
major_id INT(11)
characteristic DOUBLE
high_school_average DO...
Indexes
major_sample
id INT(11)
characteristic DOUBLE
major_id INT(11)
high_school_average DOUBLE
Indexes
high_school_top_sample
id INT(11)
major_sample_id INT(11)
subject_id INT(11)
mark DOUBLE
weight INT(11)
Indexes
entrance_exam_centroid_detail
id INT(11)
major_centroid_id INT(11)
entrance_exam_subject_id INT(11)
mark DOUBLE
weight INT(11)
Indexes
entrance_exam_subject
id INT(11)
name VARCHAR(255)
vietnamese VARCHAR(255)
is_required BIT(1)
Indexes
entrance_exam_sample_detail
id INT(11)
major_sample_id INT(11)
entrance_exam_subject_id INT(11)
mark DOUBLE
Indexes
high_school_top_centroid
id INT(11)
major_centroid_id INT(11)
subject_id INT(11)
mark DOUBLE
weight DOUBLE
Indexes
high_school_subject
id INT(11)
name VARCHAR(255)
vietnamese VARCHAR(255)
Indexes

Figure 60

b) Data Dictionary

Data dictionary: describe content of all tables	
Table name	**Description**
major_centroid	Contains the information of a major's centroid
high_school_top_centroid	Contains the information of a high school top centroid
entrance_exam_centroid_detail	Contains the information of an entrance exam centroid detail
major_standard	Contains the information of a major standard
supporting_information	Contains the information of a supporting information for predicting trend of a major
major	Contains the information of a major
entrance_exam_subject	Contains the information of an entrance exam subject
human_resource	Contains the information about human resource of each major in the system
major_sample	Contains the information about a major sample
entrance_exam_sample_detail	Contains the information about an entrance exam sample detail
salary	Contains the information about salary of each major in the system
high_school_top_sample	Contains the information about a high school top sample
high_school_subject	Contains the information about a high school subject

Table 45: Data Dictionary of High School Pupils Suggestion Physical Diagram

Table name	Attributes	Description	Domain	Null
major_centroid	id{PK}	Unique identifier of major centroid	int	No
	major_id	Major's id	int	Yes
	characteristic	Characteristic of the major centroid	double	Yes
	high_school_average	High school average score of the respective major centroid	double	Yes
high_school_top_centroid	id{PK}	Unique identifier of high school top centroid	int	No
	major_centroid_id	Major centroid's id	int	Yes
	subject_id	Subject's id	int	Yes
	mark	Mark of the high school top centroid	double	Yes
	weight	Numerical advantage when calculating a grade point average	double	Yes

major_standard	id{PK}	Unique identifier of major standard	int	No
	standard	Mean mark of the major standard	double	Yes
	unit	Unit of the major standard	varchar(255)	Yes
	source	Link URL of the source	varchar(255)	Yes
	source_name	Name of the source	varchar(255)	Yes
	major_id	Major's id	int	Yes
entrance_exam_centroid_detail	id{PK}	Unique identifier of entrance exam centroid detail	int	No
	major_centroid_id	Major centroid's id	int	Yes
	entrance_exam_subject_id	Entrance exam subject's id	int	Yes
	mark	Mark of the respective entrance exam centroid	double	Yes
	weight	Numerical advantage when calculating a grade point average	int	Yes
supporting_information	id{PK}	Unique identifier of supporting information	Int	No
	unit	Words represent the unit of the supporting information	varchar(255)	Yes
	actual	The number represents the real value	double	Yes
	level	The number represents the estimate of the level of the series at a specific time	double	Yes
	trend	The number represents the estimate of the trend of the series at a specific time	double	Yes
	forecast	The number represents the predicted value	double	Yes
	alpha	The number represents the smoothing parameter for the level	double	Yes
	beta	The number represents the smoothing parameter for the trend	double	Yes
	link	Source contains actual value	varchar(255)	Yes
	year	The number represents the current predicting year	int	Yes
	source_name	Name of the source	varchar(255)	Yes

	major_id	Major's id	int	Yes
major	id{PK}	Unique identifier of major	int	No
	name	Name of the major	varchar(255)	Yes
	Image_url	Image link of the major	varchar(255)	Yes
	vietnamese	The Vietnamese name of the major	varchar(255)	Yes
entrance_exam_subject	id{PK}	Unique identifier of entrance exam subject	int	No
	name	Name of the entrance exam subject	varchar(255)	Yes
	vietnamese	The Vietnamese name of the entrance exam subject	varchar(255)	Yes
	is_required	Check if the subject is a compulsory subject in the entrance exam	bit	Yes
human_resource	id{PK}	Unique identifier of human resource	int	No
	year	The number represents the current predicting year	int	Yes
	actual	The number represents the real value	double	Yes
	level	The number represents the estimate of the level of the series at a specific time	double	Yes
	trend	The number represents the estimate of the trend of the series at a specific time	double	Yes
	forecast	The number represents the predicted value	double	Yes
	alpha	The number represents the smoothing parameter for the level	double	Yes
	beta	The number represents the smoothing parameter for the trend	double	Yes
	link	Source contains actual value	varchar(255)	Yes
	source_name	Name of the source	varchar(255)	Yes
	major_id	Major's id	int	Yes
major_sample	id{PK}	Unique identifier of major sample	int	No
	characteristic	Characteristic of the major sample	double	Yes
	major_id	Major's id	int	Yes

	high_school_average	High school average score of the respective major sample	double	Yes
entrance_exam_sample_detail	id{PK}	Unique identifier of entrance exam sample detail	int	No
	major_sample_id	Major sample's id	int	Yes
	entrance_exam_subject	Name of the entrance exam subject in the sample	int	Yes
	mark	Mark of the entrance exam subject in the sample	double	Yes

Table 46: Table Detail of High School Pupils Suggestion Physical Diagram

2.1.2. Career Trend Prediction

a) Physical Diagram

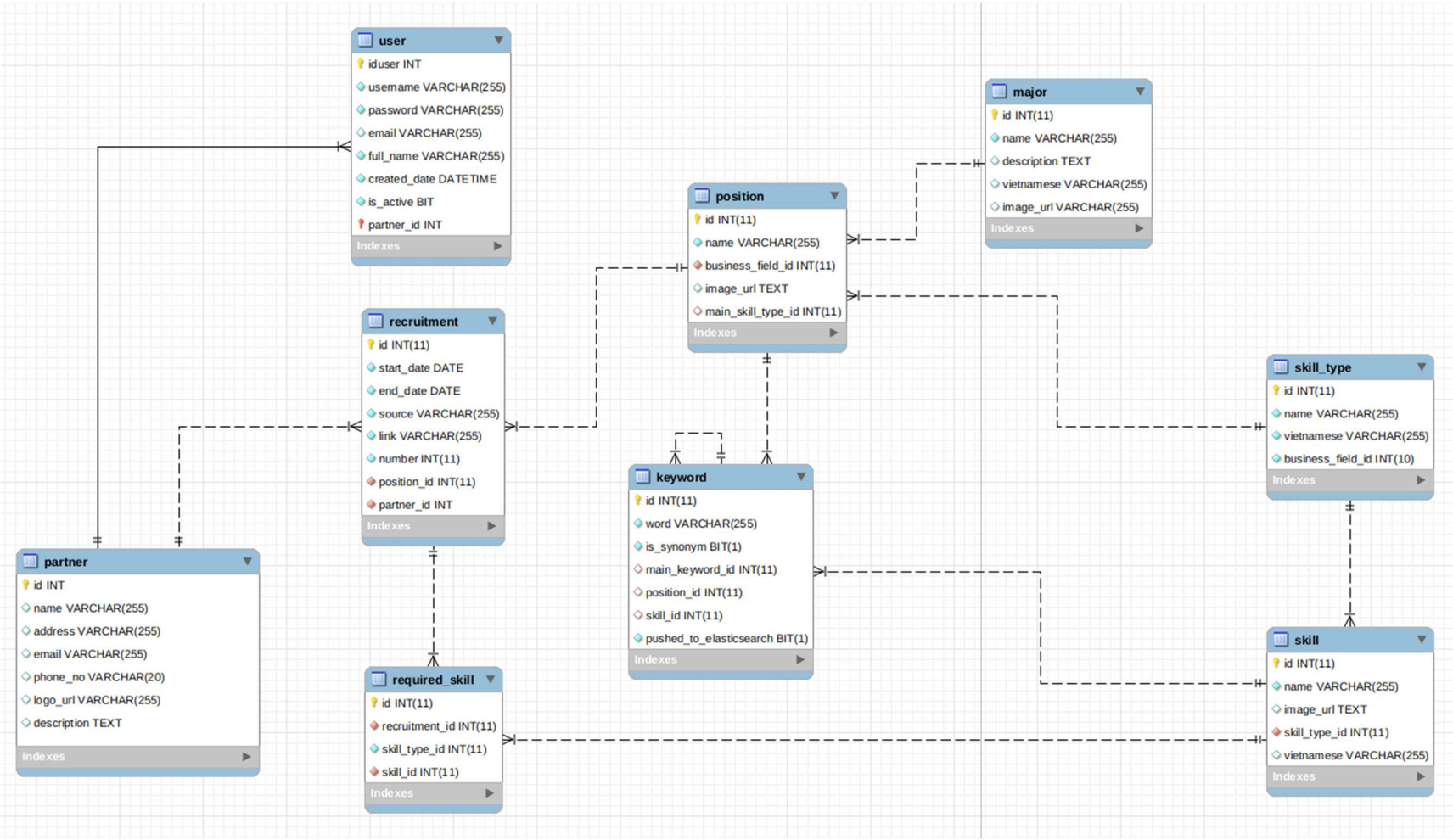

user
iduser INT
username VARCHAR(255)
password VARCHAR(255)
email VARCHAR(255)
full_name VARCHAR(255)
created_date DATETIME
is_active BIT
partner_id INT
Indexes
major
id INT(11)
name VARCHAR(255)
description TEXT
vietnamese VARCHAR(255)
image_url VARCHAR(255)
Indexes
position
id INT(11)
name VARCHAR(255)
business_field_id INT(11)
image_url TEXT
main_skill_type_id INT(11)
Indexes
recruitment
id INT(11)
start_date DATE
end_date DATE
source VARCHAR(255)
link VARCHAR(255)
number INT(11)
position_id INT(11)
partner_id INT
Indexes
skill_type
id INT(11)
name VARCHAR(255)
vietnamese VARCHAR(255)
business_field_id INT(10)
Indexes
keyword
id INT(11)
word VARCHAR(255)
is_synonym BIT(1)
main_keyword_id INT(11)
position_id INT(11)
skill_id INT(11)
pushed_to_elasticsearch BIT(1)
Indexes
partner
id INT
name VARCHAR(255)
address VARCHAR(255)
email VARCHAR(255)
phone_no VARCHAR(20)
logo_url VARCHAR(255)
description TEXT
Indexes
required_skill
id INT(11)
recruitment_id INT(11)
skill_type_id INT(11)
skill_id INT(11)
Indexes
skill
id INT(11)
name VARCHAR(255)
image_url TEXT
skill_type_id INT(11)
vietnamese VARCHAR(255)
Indexes

Figure 61

b) Data Dictionary

Data dictionary: describe content of all tables	
Table name	**Description**
user	Contains the information of a user
partner	Contains the information of a partner
recruitment	Contains the information of a recruitment
required_skill	The required skill for a position
position	Contains the information of a position
keyword	Contains the information of a keyword
major	Contains the information of a major
skill_type	Name of the category consists of a group of skills
skill	Contains the information of a skill

Table 47: Data Dictionary of Career Trend Prediction Physical Diagram

Table name	Attributes	Description	Domain	Null
partner	id{PK}	Unique identifier of partner company or organization	int	No
	name	Name of the partner company or organization	varchar(255)	Yes
	address	Address of the partner company or organization	varchar(255)	Yes
	email	Email of the partner company or organization	varchar(255)	Yes
	phone	Phone of the partner company or organization	varchar(20)	Yes
	logo_url	Image Link of the partner company or organization	varchar(255)	Yes
	description	Description about the company	text	Yes
user	id{PK}	Unique identifier of the user	int	No
	username	Username of the user used to login the system	varchar(255)	Yes
	password	Password of the user used to login the system	varchar(255)	Yes
	email	Email of the user	varchar(255)	Yes
	full_name	Full name of the user	varchar(255)	Yes

	created_date	The date when the account was created	datetime	Yes
	is_active	Check if the account is active or disabled	bit	Yes
	partner_id	Partner's id	int	Yes
required_skill	id{PK}	Unique identifier of required skill	int(11)	No
	recruitment_id	Recruitment's id	int(11)	Yes
	skill_type_id	Skill type's id	int(11)	Yes
	skill_id	Skill's id	int(11)	Yes
position	id{PK}	Unique identifier of position	int(11)	No
	name	Name of the position	varchar(255)	Yes
	business_field_id	Business field's id	iInt(11)	Yes
	image_url	Image link of the position	text	Yes
	main_skill_type_id	Main skill type's id	int(11)	Yes
recruitment	id{PK}	Unique identifier of recruitment	int(11)	No
	start_date	The date when the recruitment starts	date	Yes
	end_date	The date when the recruitment is ended	date	Yes
	source	Source of the recruitment	varchar(255)	Yes
	link	Link URL of the recruitment	varchar(255)	Yes
	number	Number represents how many employees that the company or organization is recruiting	int(11)	Yes
	position_id	Position's id	int(11)	Yes
	partner_id	Partner's id	int	Yes
keyword	id{PK}	Unique identifier of keyword	int(11)	No
	word	The keyword	varchar(255)	Yes
	is_synonym	Check if the word is a synonym of another word or not	bit(1)	Yes
	main_keyword_id	Main keyword's id	int(11)	Yes
	position_id	Position's id	int(11)	Yes
	skill_id	Skill's id	int(11)	Yes
	pushed_to_elasticsearch	Check if the keyword is pushed to elasticsearch database for cleaning data	bit(1)	Yes
major	id{PK}	Unique identifier of major	int(11)	No

	name	Name of the major	varchar(255)	Yes
	description	Description about the major	Text	Yes
	vietnamese	Vietnamese name of the major	varchar(255)	Yes
	image_url	Image link of the major	varchar(255)	Yes
skill_type	id{PK}	Unique identifier of the skill type	int(11)	No
	name	Name of the skill type	varchar(255)	Yes
	vietnamese	Vietnamese name of the skill type	varchar(255)	Yes
	business_field_id	Business field's id	int(10)	Yes
skill	id{PK}	Unique identifier of the skill	int(11)	No
	name	Name of the skill	varchar(255)	Yes
	image_url	Link image of the skill	text	Yes
	skill_type_id	Skill type's id	int(11)	Yes
	vietnamese	Vietnamese name of the skill	varchar(255)	Yes

Table 48: Table Detail of Career Trend Prediction Physical Diagram

III. Test Plan

All following features will be tested, which can include one or more functions. These features will be focused and tested thoroughly during the test phrase:

- Public page:
  - Input High School score to get Suggestions (High School Senior)
  - Input college score to get more precise suggestions
  - Suggestions for IT college pupils
  - Get prediction for a major
  - Manage careers' historical data
  - Manage partner's recruitment
- Web admin:
  - Keyword Management

III. Test Case

1. Public page

1.1. Input High School score to get Suggestions (High School Senior)

ID	Test Case Description	Test Case Procedure	Expected Output	Result	Test Date	Note
HS001	Get Suggestions	At index page, after	Redirect to the result page with	10 Fails 3 Passes	25/9/2018 to	

	for High School Pupil	choosing "High school senior", selecting 3 subjects, input 3 subject score and high school average score, selecting "Hướng ngoại" or "Hướng nội". Select **Nhận gợi ý**	suggestions based on user's input		1/10/2018	
HS002	Choose 3 subjects Toán, Lý, Anh to get suggestions	At index page, before selecting **Nhận gợi ý** choose 3 subjects Toán, Lý, Anh from 8 subjects provided	Display "Công nghệ thông tin" as top 1 on result page	5 Passes	01/10/2018 to 04/10/2018	
HS003	Choose 3 subjects **Văn, Sử, Địa** to get suggestions	At index page, before selecting **Nhận gợi ý** choose 3 subjects **Văn, Sử, Địa** from 8 subjects provided and input marks from 0 to 10 for those 3 subjects.	Display "Sư Phạm" as top 1 on result page	5 Fails 10 Passes	15/10/2018	
HS004	Choose 3 subjects **Toán, Hóa, Sinh** to get suggestions	At index page, before selecting **Nhận gợi ý** choose 3 subjects **Toán, Hóa,**	Display "Y Dược" as top 1 on result page	2 Fails 5 Passes	16/10/2018	

		Sinh from 8 subjects provided and input marks from 0 to 10 for those 3 subjects.				
HS005	Choose 3 subjects **Toán, Hóa, Sinh** with high score to get suggestions	At index page, before selecting **Nhận gợi ý** choose 3 subjects **Toán, Hóa, Sinh** from 8 subjects provided, input marks from 8-10 marks for all 3 subjects.	Display "Y Dược" as top 1 on result page	5 Passes	17/10/2018	

Table 49: Test Case Input High School score to get Suggestions (High School Senior)

1.2. Input college score to get more precise suggestions

ID	Test Case Description	Test Case Procedure	Expected Output	Result	Test Date	Note
CS001	Open college score input modal	At the result page, clicks **Bạn muốn nhận gợi ý chính xác hơn?**	Open college score input modal and dim the background	5 Fails 3 Passes	01/10/2018	
CS002	Check slider input	In college score input modal, drag the slider of a subject to specify the predicted score range	The number text input of the subject changes at real time based on user sliding the slider input	2 Fails 3 Passes	01/10/2018	
CS003	Disable subject checkbox	In college score input modal, click	The subject with checked disable checkbox is	3 Fails 10 Passes	01/10/2018	

		on the disable checkbox within the desired subject that the user wants to disable	greyed out and cannot be edited			
CS004	Receive more suggestions	In college score input modal, after inputting all score from 0 to 10 or using the slider for all the subjects that user desired. User clicks **Nhận thêm gợi ý**	Close the modal, the result page is updated based on the user's input	10 Fails 20 Passes	02/10/2018 to 05/10/2018	
CS005	Close the modal	In college score input modal, User clicks on the close button on the top right of the modal	Close the modal, the result page returns to normal.	1 Fails 3 Passes	01/10/2018	
CS005 a	Close the modal - alternative	In college score input modal, User clicks **Hủy**	Close the modal, the result page returns to normal.	3 Passes	01/10/2018	
CS005 b	Close the modal - alternative	In college score input modal, User clicks outside of the modal	Close the modal, the result page returns to normal.	3 Passes	01/10/2018	

Table 50: Test Case Input college score to get more precise suggestions

1.3. Suggestions for IT college pupils

ID	Test Case Description	Test Case Procedure	Expected Output	Result	Test Date	Note

KM001	Load already added keywords	At the index page, clicks **keyword link page**	Redirect to keyword page and load all added keywords in the database	10 Fails 5 Passes	01/10/2018 to 02/10/2018	
KM002	Add new keyword	At keyword page, type "Javac" in the keyword text input, clicks **Update**	The keyword text input is cleared, "Javac" appeared in already added keyword field	12 Fails 20 Passes	02/10/2018 to 4/10/2018	

Table 51: Test Case Suggestions for IT college pupils

1.4. Get prediction for a major

ID	Test Case Description	Test Case Procedure	Expected Output	Result	Test Date	Note
GP001	Get additional information about the major	On the result page, user clicks on the major banner	The additional major information modal is shown and highlighted, dim the result page	15 Fails 30 Passes	01/10/2018 to 07/10/2018	
GP002	Get detailed additional information about the major	On the additional major information modal, user clicks **Xem thêm chi tiết**	Redirect to detailed major information page	10 Fails 20 Passes	07/10/2018 to 14/10/2018	

Table 52: Test Case Get prediction for a major

1.5. Manage careers' historical data

ID	Test Case Description	Test Case Procedure	Expected Output	Result	Test Date	Note
MC001	Add historical data	On the historical page, staff input new historical data to the	Success or fail message will appear. If it is a success, a newly added historical data will appear	5 Fails 8 Passes	01/10/2018 to 02/10/2018	

		form. User clicks **Add**	on the historical page			
MC002	Edit historical data	On the historical page, from the list of available historical data, staff edits the data and then clicks **Save**	Success or fail message will appear. If it is a success, the historical page will be updated	8 Fails 8 Passes	02/10/2018	
MC002	Delete historical data	On the historical page, from the list of available historical data, staff clicks Delete	Success or fail message will appear. If it is a success, chosen historical data will be deleted	8 Fails 10 Passes	03/10/2018	

Table 53: Test Case Manage careers' historical data

1.6. Manage partner's recruitment

ID	Test Case Description	Test Case Procedure	Expected Output	Result	Test Date	Note
MP001	Add new partner's recruiting information	On the partner's recruitment page, staff input new partner's recruitment to the form. User clicks **Add**	Success or fail message will appear. If it is a success, a newly added partner's recruiting information will appear on the partner's recruitment page	10 Fails 5 Passes	01/10/2018 to 02/10/2018	
MP002	Edit partner's recruiting information	On the partner's recruitment page, from the list of available	Success or fail message will appear. If it is a success, the partner's recruitment	5 Fails 8 Passes	02/10/2018	

		partner's recruitment, staff edits the data and then clicks **Save**	page will be updated			
MP002	Delete partner's recruiting information	On the partner's recruitment page, from the list of available partner's recruitment, staff clicks Delete	Success or fail message will appear. If it is a success, chosen partner's recruitment will be deleted	5 Fails 5 Passes	03/10/2018	

Table 54: Test Case Manage partner's recruitment

2. CTSA CMS

2.1. Keyword Management

ID	Test Case Description	Test Case Procedure	Expected Output	Result	Test Date	Note
KM001	Load already added keywords	At the index page, clicks **keyword link page**	Redirect to keyword page and load all added keywords in the database	10 Fails 5 Passes	01/10/2018 to 02/10/2018	
KM002	Add new keyword	At keyword page, type "Javac" in the keyword text input, clicks **Update**	The keyword text input is cleared, "Javac" appeared in already added keyword field	12 Fails 20 Passes	02/10/2018 to 4/10/2018	

Table 55: Test Case Keyword Management

F. Software User's Manual

I. Installing Guide

1. Setting up environment at server side

The following software must be installed into the server machine:

1.1. Hardware requirement

– For Server Web:

	Minimum Requirement	Recommended
Internet Connection	Cable, WiFi (4Mbps)	Cable, WiFi (8Mbps)
Computer Processor	Intel® Core™ i3 (2.50GHz)	Intel® Core™ i7 or faster
Operating System	Window 7	Ubuntu 18.04 LTS
Computer Memory	8GB Ram	16GB Ram

Table 56: Hardware Requirement for Server Web

1.2. Software requirements

Software	Name / Version	Description
Operating system	- Ubuntu 18.04 LTS - Windows 10 build 1803	Operating system and platform for development
Environment	- Java 8 - Spring Boot 2.0 - Java EE 8.0	Specification for developing web services and web application
IDE	IntelliJ IDEA 2018	Programming tool
Database Management System	MySQL 5.7 Elasticsearch 6.5 Redis 5.0	Tools used to create & manage the database for system
Source control	GitHub 2.14	Tools used for source control

Web Server	Apache Tomcat 8.0 Netty 4.0	Deployment Environment
Web browser	Chrome 70	Browser used in testing

Table 57: Software Requirements for Development

2. Deployment at server side

2.1 Prepare deployment package

Step 1: Download and install Java 8
Link download:
Step 2: Download and install Maven 3.5
Link download:
Step 3: Download and install Docker
Link download:
Step 3: Download and install MySQL 5.7 on Docker
Link tutorial:
Step 4: Download and install Elasticsearch 6.5 on Docker
Link tutorial:
Step 5: Configurate Elasticsearch 6.5 on Docker
Link tutorial:
Step 6: Download and install Redis 5.0 on Docker
Link tutorial:

2.2 Configure Server before deploy

- Make sure Java 8 is in the environment path:
  - Using the command line or terminal, type:
    - `java -version`
    - The result should look like:

```
java version "1.8.0_102"
Java(TM) SE Runtime Environment (build 1.8.0_102-b14)
Java HotSpot(TM) 64-Bit Server VM (build 25.102-b14, mixed mode)
```

- Make sure Maven is in the environment path:
  - Using the command line or terminal, type:
    - `mvn -v` or `mvn –version`
    - The result should look like:

```
Apache Maven 3.5.4 (1edded0938998edf8bf061f1ceb3cfdeccf443fe; 2018-06-17T14:33:14-04:00)
Maven home: /usr/local/Cellar/maven/3.3.9/libexec
Java version: 1.8.0_102, vendor: Oracle Corporation
```

- Make sure Docker is running.
  - Using the command line or terminal, type:
    - `docker`

- The result should look like:

– Make sure MySQL, Elasticsearch and Redis is running.
  - Using the command line or terminal, type:
    - `docker ps`
    - The result should look like:

2.3 Deploy web application on server

Start web services by the following order:

1. Eureka Service
2. Config Service
3. Careers Service
4. Careers Suggestions Service
5. Careers Trend Service
6. Warehouse Service

Then, deploy the web application.

Services and web application can be deployed using the following steps:

Step 1: Go to the service or web application folder

Step 2: Open command line or terminal

Step 3: Type the following command

– `mvn spring-boot:run`

– The result should look like:

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::  (v2.1.1.RELEASE)
....... . . .
....... . . . (log output here)
....... . . .
........ Started Example in 2.222 seconds (JVM running for 6.514)
```

3 Setting up the environment at client side

3.1 Setting up for computer

The client devices should have one of the following browsers to access the website:

- Google Chrome (version 69)
  - Link download:
- Firefox (version 52)
  - Link download:

II. User Guide

1. Getting user's suggestion for high school students

Figure 62: Getting user's suggestion for high school students guide 1

Figure 63: Getting user's suggestion for high school students guide 2

Figure 64: Getting user's suggestion for high school students guide 3

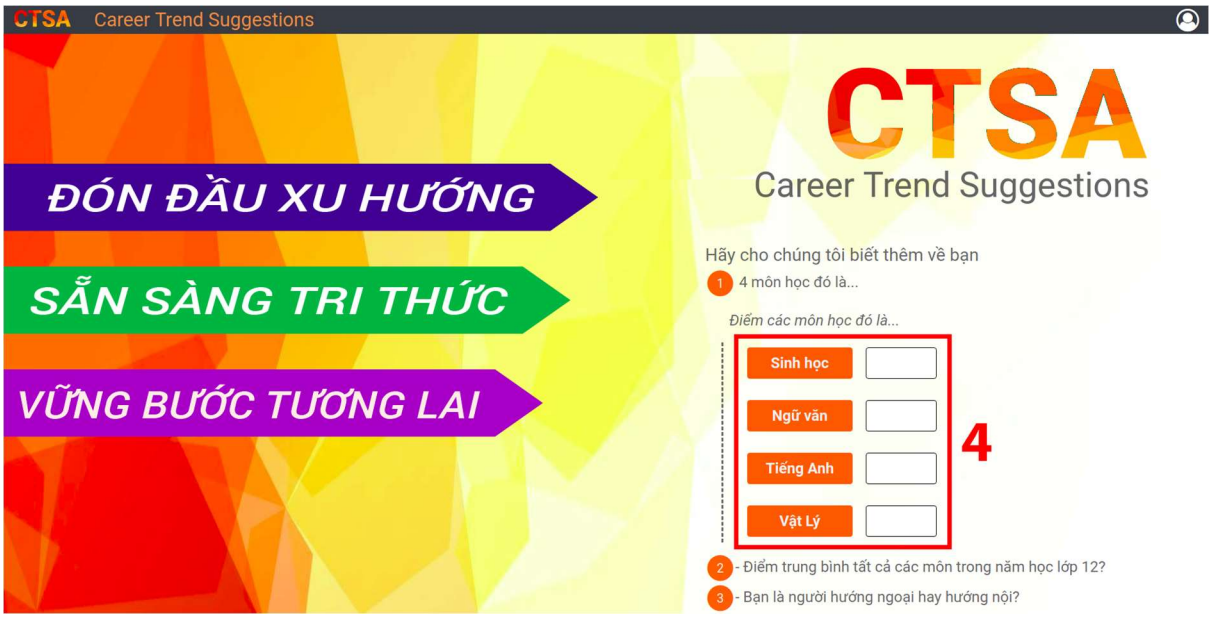


Figure 65: Getting user's suggestion for high school students guide 4

Figure 66: Getting user's suggestion for high school students guide 5

Figure 67: Getting user's suggestion for high school students guide 6

Step	Description
1	Click "Học sinh cấp 3"
2	Fill the field "Number of subjects"
3	Choose the subjects based on user's desire
4	Fill the field "Subject's mark"
5	Fill the field "Average Score"
6	Choose the guest's characteristic
7	Click "Nhận gợi ý"

Table 58: Getting user's suggestion for high school students guide

2. Getting user's suggestion for IT college pupils

Figure 68: Getting user's suggestion for IT college pupils guide 1

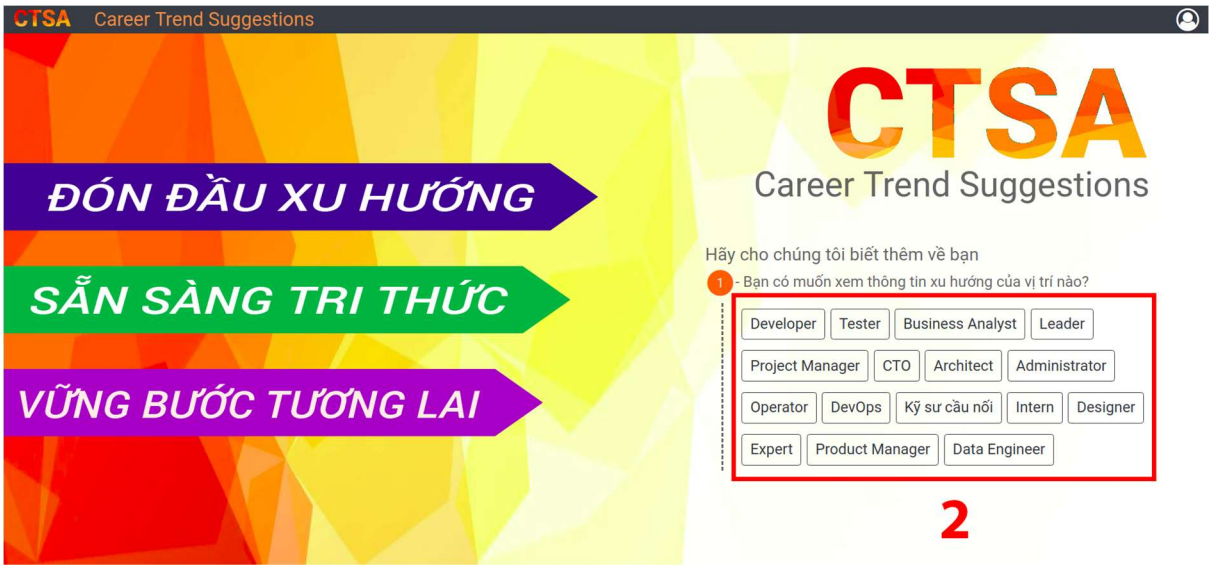


Figure 69: Getting user's suggestion for IT college pupils guide 2

Step	Description
1	Click "Tân sinh viên ngành IT"
2	Choose the position which the user wants to see suggesting information

Table 59: Getting user's suggestion for IT college pupils guide

3. Getting more user's suggestion with college subject's mark

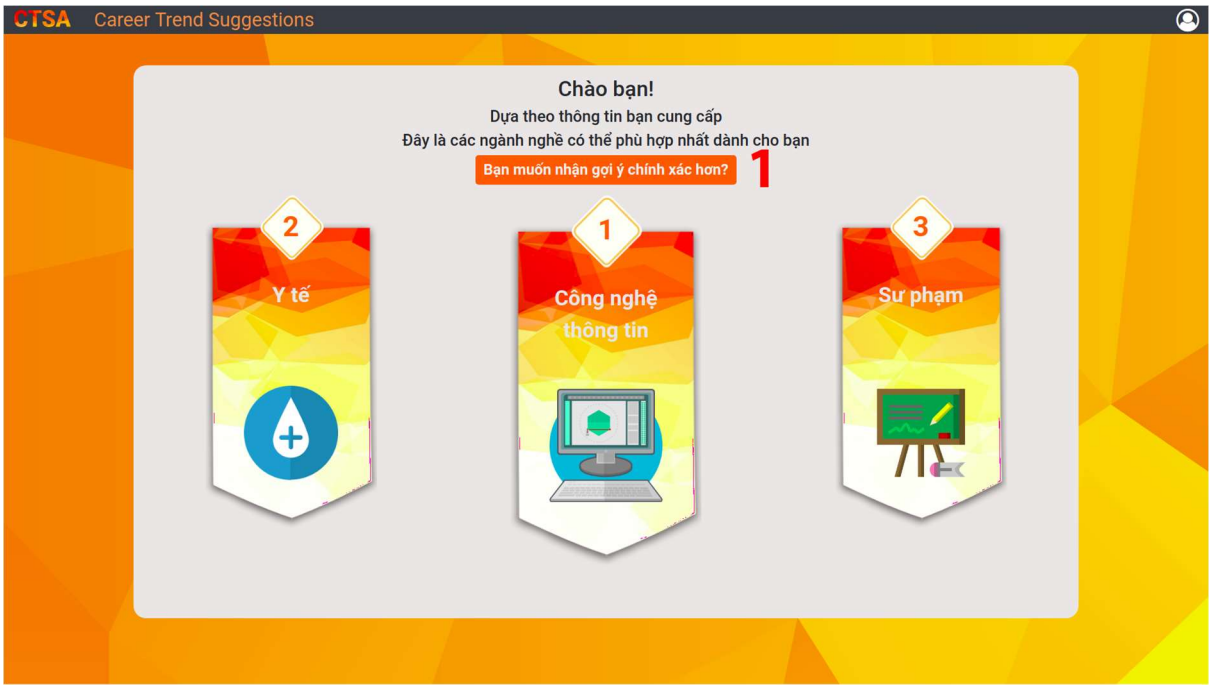


Figure 70: Getting more user's suggestion with college subject's mark guide 1

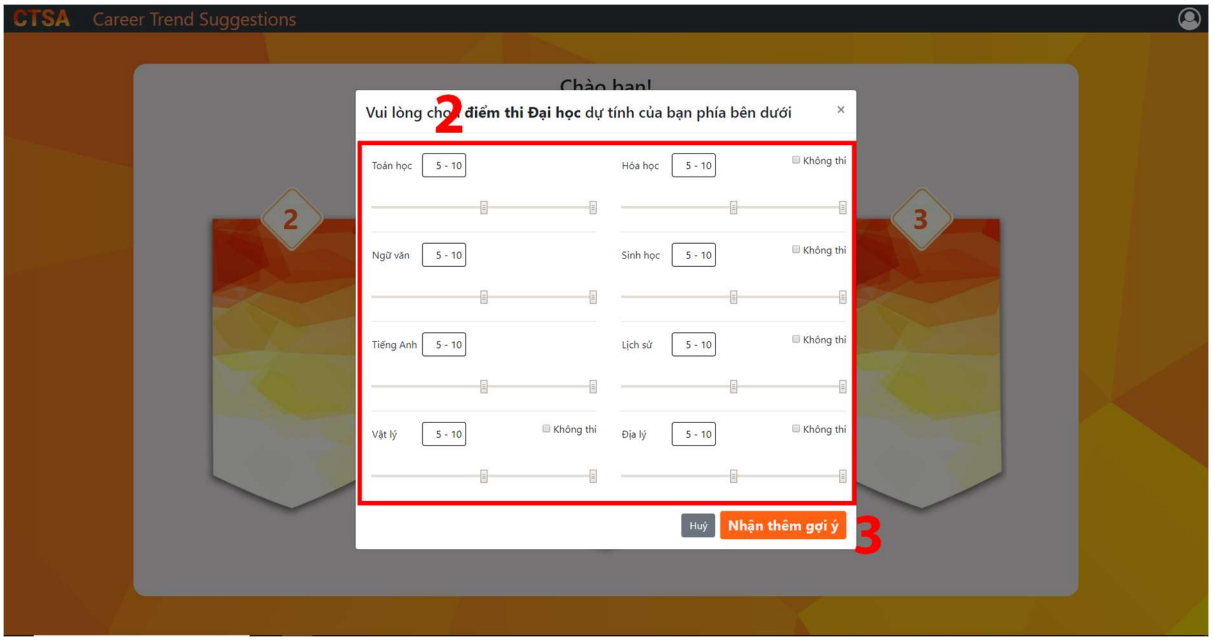


Figure 71: Getting more user's suggestion with college subject's mark guide 2

Step	Description
1	Click "Bạn muốn nhận gợi ý chính xác hơn?"
2	Fill all the field score user desired
3	Click "Nhận thêm gợi ý"

Table 60: Getting more user's suggestion with college subject's mark guide

4. Getting prediction about major

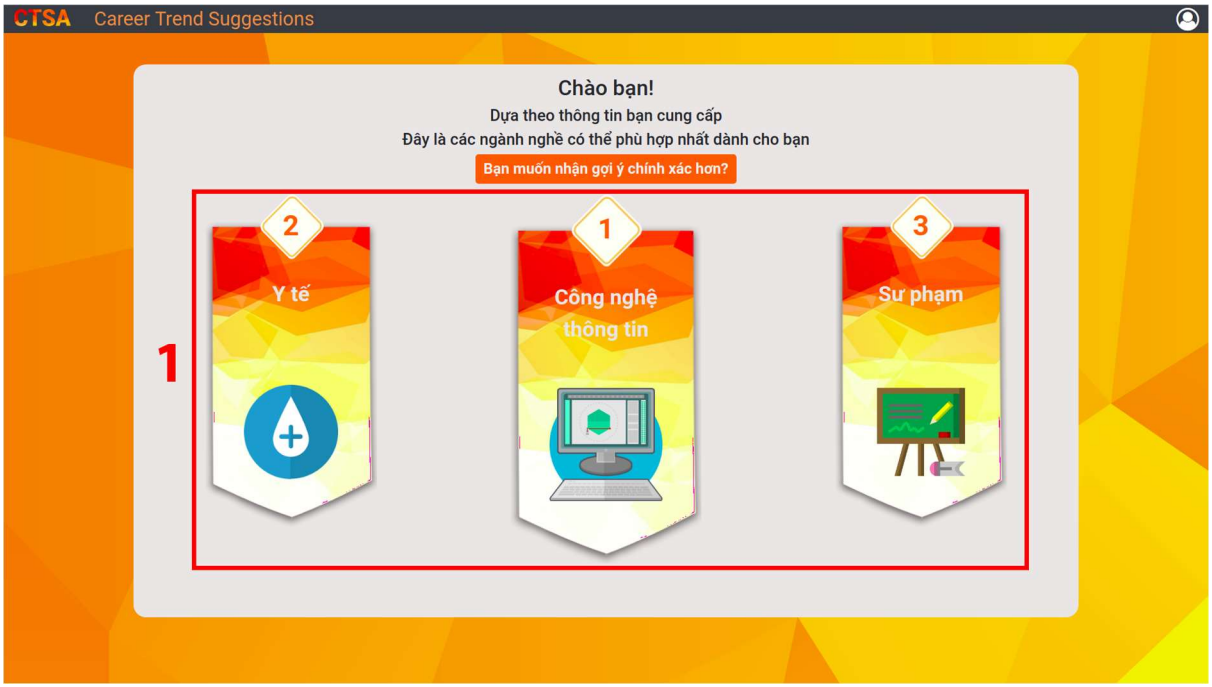


Figure 72: Getting prediction about major guide 1

Figure 73: Getting prediction about major guide 2

Figure 74: Getting prediction about major guide 3

Step	Description
1	Click major banner
2	Click "Chi tiết ngành"
3	Edit year number to specify which year the user wants to analyze

Table 61: Getting prediction about major guide 3

References

[1] OMG UML (August 2018), OMG Unified Modeling Language ™ (OMG UML) Superstructure

[2] Wasson, M. and Celarier, S. (November 13, 2018), *Microservices architecture style*. Retrieved from https://docs.microsoft.com/en-us/azure/architecture/guide/architecture-styles/microservices

[3] Spring Team, *Spring Cloud,* Retrieved from https://spring.io/

[4] Richardson, C. (December 4, 2015), *Event-Driven Data Management for Microservices*. Retrieved from https://www.nginx.com/blog/event-driven-data-management-microservices/

[9] Peter Chen (March 1976), The entity-relationship model - toward a unified view of data, ACM Transactions on Database Systems (TODS) - Special issue: papers from the international conference on very large data bases: September 22–24, 1975, Framingham, MA