**Odd Semester (2019/2020)**



**BINUS UNIVERSITY**



**BINUS INTERNATIONAL**



**Assignment Cover Letter**

**(Teamwork)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | | |  |  |  |  |
| **Student Information**: | **First Name** | | |  |  | **Last Name**  **Athallah Arief**  **Wardana**  **Trianindya Putra**  **A.** | **Student ID Number**  **2201798326**  **2201798345**  **2201798585**  **2201796636** |
| 1.  2.  3.  4. | **Fauzan**  **Winson**  **Rafli**  **Ricky** | |
|  |  |  |  |  |  |  |  |
| **Course Code** | **: COMP6340** |  |  |  |  | **Course Name** | **: Analysis of Algorithms** |
| **Class** | **: L3AC** |  |  |  |  | **Name of Lecturer(s)** | **:** **Maria Seraphina** |
|  |  |  |  |  |  |  |  |
| **Major** | **: CS** |  |  |  |  |  |  |
| **Title of Assignment**  (if any) | : **iPlay CMS** |  |  |  |  |  |  |
| **Type of Assignment**    **Submission Pattern** | **: Final Project** | |  |  |  |  |  |
| **Due Date** | **: 29 - 10 - 2019** | |  |  |  | **Submission Date** | **: 29 – 10 -2019** |

The assignment should meet the below requirements.

1. Assignment (hard copy) is required to be submitted on clean paper, and (soft copy) as per lecturer’s instructions.
2. Soft copy assignment also requires the signed (hardcopy) submission of this form, which automatically validates the softcopy submission.
3. The above information is complete and legible.
4. Compiled pages are firmly stapled.
5. Assignment has been copied (soft copy and hard copy) for each student ahead of the submission.

# Plagiarism/Cheating

BiNus International seriously regards all forms of plagiarism, cheating and collusion as academic offenses which may result in severe penalties, including loss/drop of marks, course/class discontinuity and other possible penalties executed by the university. Please refer to the related course syllabus for further information.

# Declaration of Originality

By signing this assignment, I understand, accept and consent to BiNus International terms and policy on plagiarism. Herewith I declare that the work contained in this assignment is my own work and has not been submitted for the use of assessment in another course or class, except where this has been notified and accepted in advance.

Signature of Student:

1. Fauzan Athallah Arief 2. Winson Wardana 3. Rafli Trianindya Putra 4. Ricky A.

**CONTENTS**

**1. Introduction 4  
 1.1. Background 4  
 1.2. Problem Description 4**

**2. Related Work 4**

**3. Proposed Alternative Solution 5**

**4. Implementation 5  
 4.1. Libraries Used 5  
 4.2. Implementation of Searching and Sorting Algorithm 6**

**5. Program Manual 8**

**6. Discussion 9  
6.1. Conclusion 9  
6.2. Recommendation 9  
 6.2.1. Offline Capabilities 9  
 6.2.2. Most Effective Sort and Search Method 9**

**7. References 10**

# Introduction

Our final project will revolve around using searching and sorting algorithm then combined it to create an online registration system program. This is a algorithm-oriented topic. We aim to create a program which solved the user tasks on a Windows computer by simply registering new data through user’s input and manage that data easily, so they don’t have to update it manually.

**1.1 Background**

## Fali had a family company that runs a different kind of business starting from property to a barbershop, and one of the business that runs in the company is a Montessori school name iPlay.

## iPlay already opened since July, and already started teaching since August. In registering some people find it difficult to get to the school, due to their parent schedule that are tight so they can't find a time to registering their child.

## So from there we like to make the system algorithm to make the system registration, so parents that are have some difficulties in registering their children can register their children from home, so they only need to come to the school to pay the tuition fee and signing some documents or just show the payment proof and sign documents. And their parent can easily see their kid class. So, we create this project name who want to apply and register to the Montessori.

## **1.2 Problem Description**

We needed a compatible algorithm for this project that will lets us do multiple tasks at once. While existing searching and sorting algorithm allows us to do many things, we saw a problem where sometimes the algorithm itself doesn’t match with its intended purposes. We wanted to make something that allows for quick and simple registration data that can be stored through efficiencies.

# Related Work

There are several related applications to ours. Microsoft Excel, and School Administration Software Platform (A.S.A.P.). While those other programs have enough data to do many different things with the help of context, we aim our program to be simpler but can do many things at once. This is because we implement our own algorithm methods and made them as simple as possible to understand and manage.

## **Proposed Alternative Solution**

From the problem discussion, we would like to make a Python application, based on algorithm that can help the user to do their tasks on a Windows computer by creating a content management system (CMS) for admin in Montessori school (iPlay). We chose to create our own algorithm by combining existing searching and sorting algorithm then customize them, so it matches the intended function but more efficient and improved.

We will then implement our own method of content management system by using MySql as database for storing the data.

1. **Implementation**

**4.1. Libraries Used**

* **Matplotlib** - plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+.

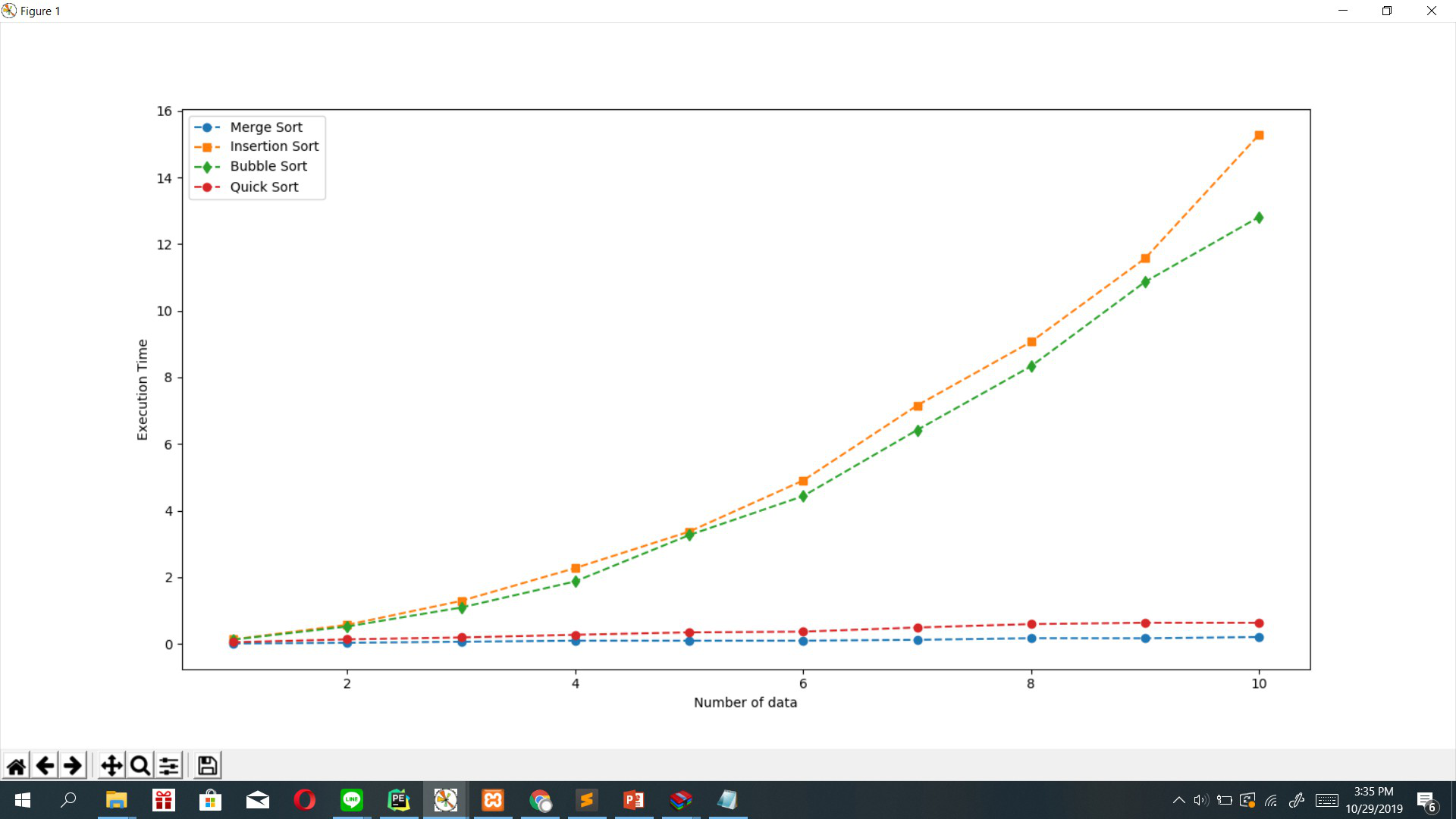
# 4.2. Implementation of Searching and Sorting Algorithm

Our project focuses on how we can search and sort data simultaneously through our own algorithm that can be executed by python. The approach that we chose is by scanning the text for keywords and objects or nouns that can be operated on. This approach is inspired by bottom-up parsing.

First, our program splits the input by the user (either it’s from speech to text, or directly typing it on the input box) by looking for the words “then” and “and.” This first step is crucial in chaining the commands and avoid the program from activating the wrong command from a long sentence.

Second, our program looks for important keywords that signals it to look for another set of object keywords. The following picture shows our current list of keywords. Note that we designed this program with modularity in mind, thus more keywords can be added in the future without much hassle with the rest of the code.

|  |
| --- |
| def searchName(x): |
|  | p = 0 |
|  | unsorted=[] |
|  | temp=[] |
|  | sql\_select\_Query = "select \* from customer" |
|  | cursor = connection.cursor() |
|  | cursor.execute(sql\_select\_Query) |
|  | records = cursor.fetchall() |
|  | for row in records: |
|  | unsorted.append([row[1],row[0]]) |
|  | found = [] |
|  | temp = [] |
|  | insertionSort(unsorted) |
|  | for name in unsorted: |
|  | if(len(name[0]) >= len(x)): |
|  | test = '' |
|  | z = name[0] |
|  | for i in range(len(x)): |
|  | test+=z[i] |
|  | found.append([test,name[1]]) |
|  |  |
|  |  |
|  | masihAda = True |
|  | while masihAda: |
|  | result = binarySearch(found, x) |
|  | if result != None: |
|  | temp.append(found[result]) |
|  | del found[result] |
|  | p = 1 |
|  | else: |
|  | masihAda = False |
|  | fullName = [] |
|  | for i in range (len(temp)): |
|  | getNameById(fullName,temp[i][1]) |
|  | temp=[] |
|  |  |
|  |  |
|  | for name in fullName: |
|  | if(len(x) < len(name[0])): |
|  | temp.append([name[0][len(x)],name[1]]) |
|  | elif(len(x) == len(name[0])): |
|  | temp.append([" ",name[1]]) |
|  |  |
|  |  |
|  | insertionSort(temp) |
|  | fullName = [] |
|  | for i in range (len(temp)): |
|  | getNameById(fullName,temp[i][1]) |
|  | for name in fullName: |
|  | sql\_select\_Query = "select \* from customer where id = '{}' ".format(name[1]) |
|  | cursor = connection.cursor() |
|  | cursor.execute(sql\_select\_Query) |
|  | records = cursor.fetchall() |
|  | for row in records: |
|  | table.append\_row([row[1],row[2],row[3],row[4],row[5],row[6],row[7]]) |
|  | print(table) |
|  | table.clear() |
|  | return p |



So after we fix the comparison of the sorting algorithm we know that from the data sets from the graph that the fastest algorithm is the merge sort, so in our case we use the insertion sort because the merge sort code that we use from geekforgeeks that the merge sort code is not optimize. So it messes the time run and complexity and make its slow

Program Manual

A screenshot of a social media post

Description automatically generated

* When the page loaded, you will be needed to input your child’s personal data such as : Full Name, Birthdate, Age, Address, etc.
* Once all of the data is filled, then you can press submit button below.
* After then your data will be stored in the database, then can be accessed by the admin.
* Then the admin can access the database to search or even sort the children’s data.

# Discussion

## **6.1. Conclusion**

From what we have programmed, we feel that we’ve achieved our goals to help the Montessori school. Although we were satisfied with what we have come up with, we could still improve on some aspect of the program; for instance, we designed the program with simple steps in mind but we think the program can be more simplified.

**6.2. Recommendation**

### **6.2.1. Offline Capabilities**

One of the drawbacks from today’s online registration system is that the need of online connection, because they store the data outside not local. Our program doesn’t have this kind of problem.

### **6.2.2. Most Effective Sort and Search Method**

With machine learning, the process of parsing the commands would be completely different. We will no longer be simply using a bottom-up model which means that the program would be able to understand a wider set of sentence structures and correct misspoken words without even adding them to the list of possible misspoken words

# References

* http://zetcode.com/python/prettytable/
* https://www.geeksforgeeks.org/binary-search-bisect-in-python/
* <https://www.geeksforgeeks.org/merge-sort/>
* Asoka Wotulo
* Fernandha Dzaky

**Github Link : https://github.com/winsonwardana/finalprojectAOA**

**Link Video : (Belom)**