UNIVERSITI TEKNOLOGI PETRONAS

OBJECT ORIENTED PROGRAMMING TFB1033/TEB1043

HR WEBSITE

Bug Fix

GROUP NAME: ALGEBROS

During our presentation on Thursday, 21 November 2024, our group encountered an bug in the code. Upon investigation, we identified that the issue stemmed from the functions ReadAllEmployee and ReadAllApplication, which were originally designed for console applications. The code before works perfectly fine for console. However, the current implementation was intended for a Web Service API.

We have since resolved the issue by modifying the code to ensure compatibility with the Web Service API. The updated functions now align with the requirements of the API, addressing the error and improving functionality.

Bug in Service Provider

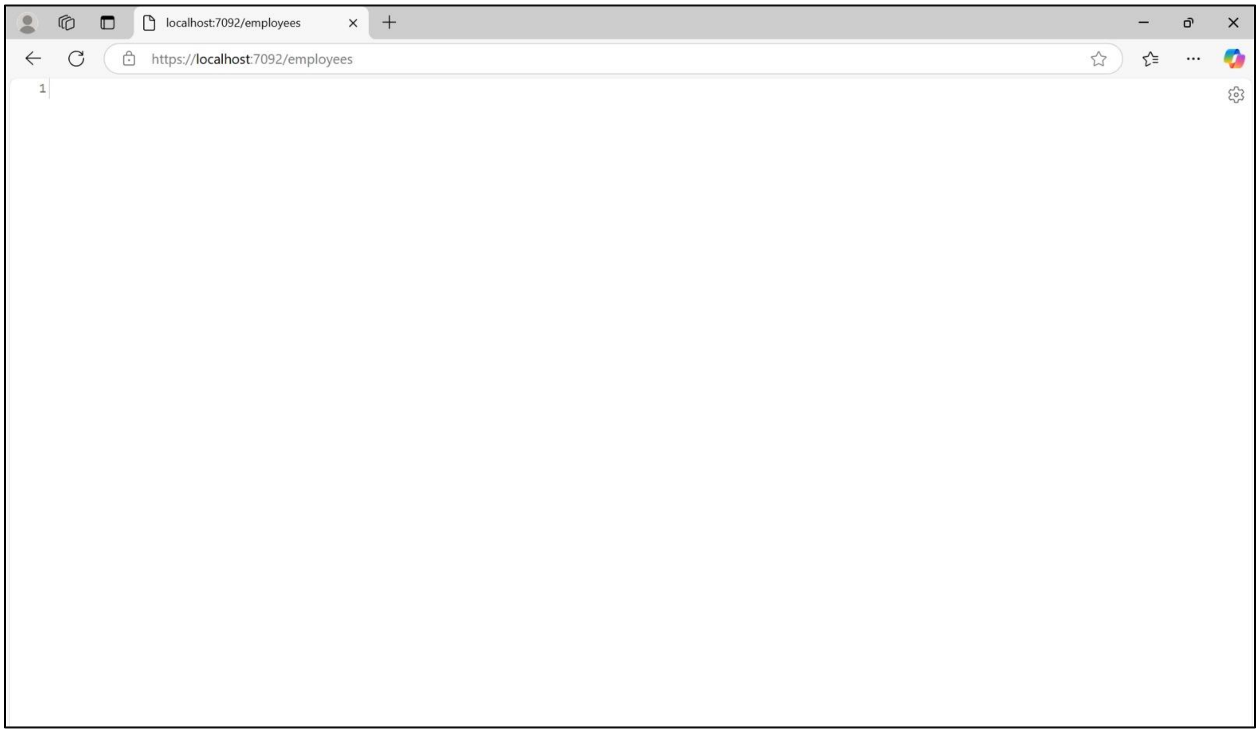


Figure 1.0 Output on Service Provider for Employees Path

```
public async Task<List<Employee>> ReadAllEmployees()

    {

        List<Employee> readEmployees = new List<Employee>();
        //Collection Reference - Points to the "employee" collection

        CollectionReference collectionRef = db.Collection("employee");

        Console.WriteLine("-----------------------------------------------------");

        Console.WriteLine("Retrieving all employees...");

        Console.WriteLine("-----------------------------------------------------");


        //Fetch all documents in the collection

        QuerySnapshot snapshot = await collectionRef.GetSnapshotAsync();


        //Process each document in the snapshot

        foreach (DocumentSnapshot doc in snapshot.Documents)

        {

          if (doc.Exists)

          {

            //Safely retrieve each field and handle missing or null values

            string employeeName = doc.ContainsField("EmployeeName") ?
doc.GetValue<string>("EmployeeName") : "Unknown";

            string employeeIDStr = doc.ContainsField("EmployeeID") ?
doc.GetValue<string>("EmployeeID") : "0";


            //Convert EmployeeID from string to int

            int employeeID = int.TryParse(employeeIDStr, out int idValue) ? idValue : 0;
```

```
                string position = doc.ContainsField("Position") ?
doc.GetValue<string>("Position") : "Unknown";

                string contactNumber = doc.ContainsField("ContactNumber") ?
doc.GetValue<string>("ContactNumber") : "N/A";

                string status = doc.ContainsField("Status") ? doc.GetValue<string>("Status") :
"Inactive";



                //Create and add the Employee object to the list

                Employee readEmployee = new Employee(employeeName, employeeID,
position, contactNumber, status);

                readEmployees.Add(readEmployee);

                Console.WriteLine("Employee Name: " + readEmployee.EmpName);

                Console.WriteLine("Employee ID: " + readEmployee.EmpID);

                Console.WriteLine("--------------------------------------------------");

             }

          }

          Console.WriteLine("All employees retrieved successfully. Total count: " +
readEmployees.Count);

          return readEmployees;

       }
```

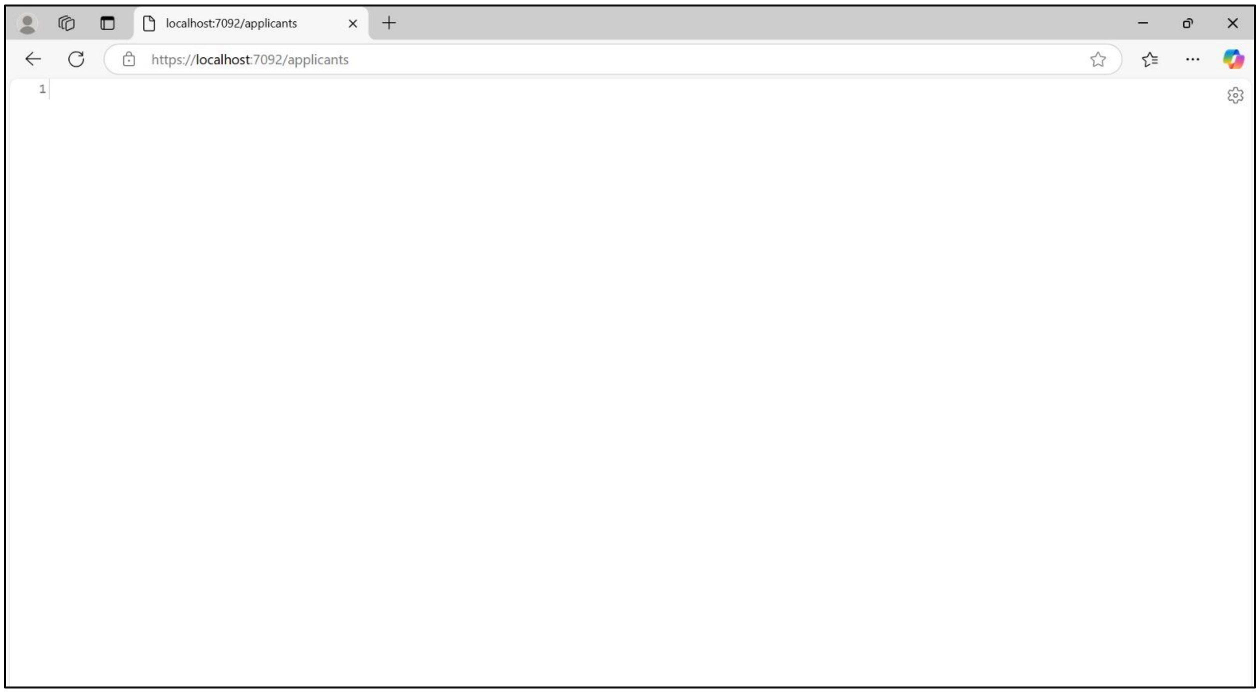


Figure 1.1 Output on Service Provider for Applicants Path

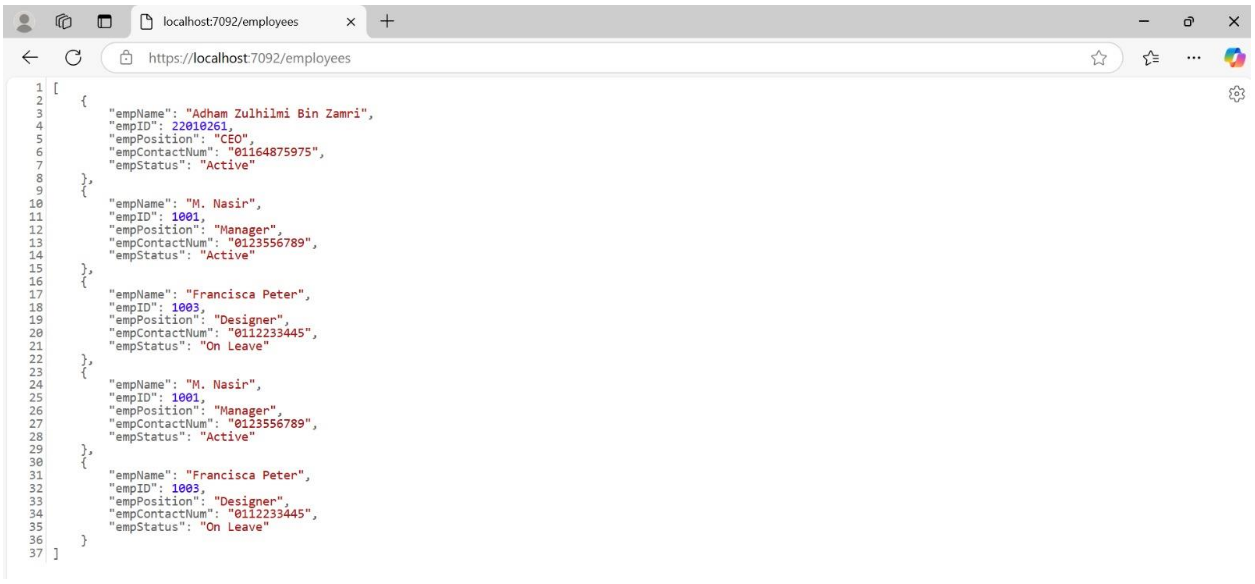
```
public async Task<List<Applicant>> ReadAllApplicants()
    {
        List<Applicant> readApplicants = new List<Applicant>();


        //Collection Reference - Points to the "applicant" collection
        CollectionReference collectionRef = db.Collection("applicant");
        Console.WriteLine("-------------------------------------------------------");
        Console.WriteLine("Retrieving all applicants...");
        Console.WriteLine("-------------------------------------------------------");


        //Fetch all documents in the collection
```

```
      QuerySnapshot snapshot = await collectionRef.GetSnapshotAsync();


      foreach (DocumentSnapshot doc in snapshot.Documents)

      {

        if (doc.Exists)

        {

          //Safely retrieve each field and handle missing or null values

          string applicantName = doc.ContainsField("ApplicantName") ?
doc.GetValue<string>("ApplicantName") : "Unknown";

          string applicantContactNum = doc.ContainsField("ApplicantContactNum") ?
doc.GetValue<string>("ApplicantContactNum") : "N/A";


          //Convert ApplicantID from string to int (if applicable)

          int applicantTempID = doc.ContainsField("ApplicantID") &&
int.TryParse(doc.GetValue<string>("ApplicantID"), out int tempID) ? tempID : 0;


          string role = doc.ContainsField("Role") ? doc.GetValue<string>("Role") :
"Unknown";


          //Handle InterviewDate (stored as string in Firestore)

          string interviewDateStr = doc.ContainsField("InterviewDate") ?
doc.GetValue<string>("InterviewDate") : "N/A";

          DateTime interviewDate = DateTime.MinValue;
```

```
            //Parse InterviewDate if it exists

            if (!string.IsNullOrEmpty(interviewDateStr)) //Check if the string is not null or
empty

            {

              if (DateTime.TryParse(interviewDateStr, out DateTime parsedDate))

              {

                interviewDate = parsedDate; //Parsed successfully

              }

              else

              {

                Console.WriteLine($"Invalid date format for InterviewDate:
{interviewDateStr}");

              }

            }

            /* DateTime Watergate Issue!

             * out: data type, variable to be passed by reference, where the result will be
stored

             * parsedDate: The parsed DateTime value

             * changed interviewDate to parsedDate for update

             */



            //Create and add the Applicant object to the list

            Applicant readApplicant = new Applicant(applicantName,
applicantContactNum, role, interviewDate, applicantTempID);
```

```
                readApplicants.Add(readApplicant);



                //Log the Applicant details

                Console.WriteLine("Applicant Name: " + readApplicant.AppName);

                Console.WriteLine("Applicant Contact Number: " +
readApplicant.AppContactNum);

                Console.WriteLine("Role: " + readApplicant.AppRole);

                Console.WriteLine("Interview Date: " + (interviewDate != DateTime.MinValue ?
interviewDate.ToString("yyyy-MM-dd HH:mm:ss") : "N/A"));

            }

        }

        Console.WriteLine("All applicants retrieved successfully. Total count: " +
readApplicants.Count);

        return readApplicants;

    }
```

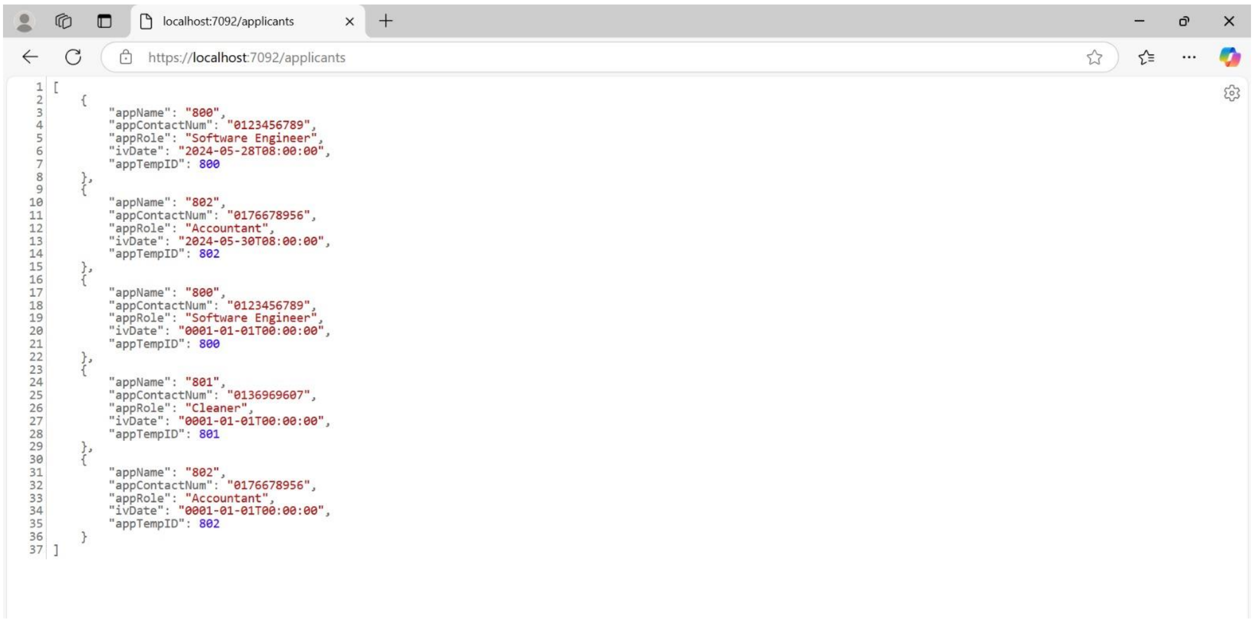
Bug Fixing on Service Provider



```
[
    {
        "empName": "Adham Zulhilmi Bin Zamri",
        "empID": 22010261,
        "empPosition": "CEO",
        "empContactNum": "01164875975",
        "empStatus": "Active"
    },
    {
        "empName": "M. Nasir",
        "empID": 1001,
        "empPosition": "Manager",
        "empContactNum": "0123556789",
        "empStatus": "Active"
    },
    {
        "empName": "Francisca Peter",
        "empID": 1003,
        "empPosition": "Designer",
        "empContactNum": "0112233445",
        "empStatus": "On Leave"
    },
    {
        "empName": "M. Nasir",
        "empID": 1001,
        "empPosition": "Manager",
        "empContactNum": "0123556789",
        "empStatus": "Active"
    },
    {
        "empName": "Francisca Peter",
        "empID": 1003,
        "empPosition": "Designer",
        "empContactNum": "0112233445",
        "empStatus": "On Leave"
    }
]
```

Figure 2.0 Output on Service Provider for Employees Path

```
public async Task ReadAllEmployees()
{
    Query collectionQuery = db.Collection("employee");
    QuerySnapshot allQuerySnapshot = await collectionQuery.GetSnapshotAsync();
    foreach (DocumentSnapshot documentSnapshot in allQuerySnapshot.Documents)
    {
        Dictionary<string, object> data = documentSnapshot.ToDictionary();
        int EmpID = int.Parse(data["EmployeeID"].ToString());
        Employee TempList = new Employee(data["EmployeeName"].ToString(),
                                         EmpID,
                                         data["Position"].ToString(),
                                         data["ContactNumber"].ToString(),
                                         data["Status"].ToString());
        employeeList.AddEmployee(TempList);
     }
 }
```



```
[
    {
        "appName": "800",
        "appContactNum": "0123456789",
        "appRole": "Software Engineer",
        "ivDate": "2024-05-28T08:00:00",
        "appTempID": 800
    },
    {
        "appName": "802",
        "appContactNum": "0176678956",
        "appRole": "Accountant",
        "ivDate": "2024-05-30T08:00:00",
        "appTempID": 802
    },
    {
        "appName": "800",
        "appContactNum": "0123456789",
        "appRole": "Software Engineer",
        "ivDate": "0001-01-01T00:00:00",
        "appTempID": 800
    },
    {
        "appName": "801",
        "appContactNum": "0136969607",
        "appRole": "Cleaner",
        "ivDate": "0001-01-01T00:00:00",
        "appTempID": 801
    },
    {
        "appName": "802",
        "appContactNum": "0176678956",
        "appRole": "Accountant",
        "ivDate": "0001-01-01T00:00:00",
        "appTempID": 802
    }
]
```

Figure 2.1 Output on Service Provider for Applicants Path

```
public async Task ReadAllApplicants()
{
    Query collectionQuery = db.Collection("applicant");
    QuerySnapshot allQuerySnapshot = await collectionQuery.GetSnapshotAsync();
    foreach (DocumentSnapshot documentSnapshot in allQuerySnapshot.Documents)
    {
        Dictionary<string, object> data = documentSnapshot.ToDictionary();
        int ApptemID = int.Parse(data["ApplicantID"].ToString());
        DateTime IvDate = DateTime.MinValue;
        if (data.TryGetValue("InterviewDate", out var interviewDateValue))
        {
            string dateFormat = "M/d/yyyy h:mm:ss tt"; // Exact format of the date string
            if (!DateTime.TryParseExact(
                    interviewDateValue?.ToString(),
                    dateFormat,
                    CultureInfo.InvariantCulture,
                DateTimeStyles.None,
                out IvDate))
            {
                Console.WriteLine($"Failed to parse InterviewDate: {interviewDateValue}");
            }
            else
            {
                Console.WriteLine($"Successfully parsed InterviewDate: {IvDate}");
            }
        }
        Applicant TempData = new Applicant(data["ApplicantID"].ToString(),
                                           data["ApplicantContactNum"].ToString(),
                                           data["Role"].ToString(),
                                           IvDate,
                                           ApptemID);
        applicantList.AddApplicant(TempData);
    }
}
```