REALIZING A LOOSELY-COUPLED STUDENTS PORTAL FRAMEWORK

[1]Rami Raba, [1]Dr. Marawan Al-Jemeli, [2]Mohammed Awad

[1]Information System Department,Limkokwing University
Cyberjaya, Malaysia

[2]Faculty of Applied Engineering and Urban Planning
University of Palestine Alzhra, Gaza, Palestine

Abstract

Most of the currently available students' portal frameworks are tightly-coupled frameworks. A recent research done by the authors of this paper has discussed how to distribute the concepts of the traditional students' portal framework and came out with a distributed interoperable framework. This paper realizes the distributed interoperable students' portal framework by developing a prototype. This prototype is based on Service Oriented Architecture (SOA). The prototype is tested using web service testing and compatibility testing.

Keywords

Tightly-coupled, Distribution, Interoperability, Loosely-Coupled & SOA

1. INTRODUCTION

As a conclusion of what has been explored in paper [1], figure 1 presents the distributed interoperable framework in which all academic and administrative departments of a university portal are distributed and encapsulated as components of a complete portal. These components are software packages or modules that encapsulate a set of related data. As shown in figure 1, the users that are connected to the internet/intranet can view and update their profiles and specific needs via registration and admission, students' affairs, financial affairs, graduate studies, library and/or other extended components. For loose coupling, the client does not have a direct access to the required component, while there is an Orchestration Point that works as a station to verify the validity of users and to identify his/her request. The new students' portal framework that is based on SOA has isolated the components of the framework in a way that these components are distributed and interoperable.

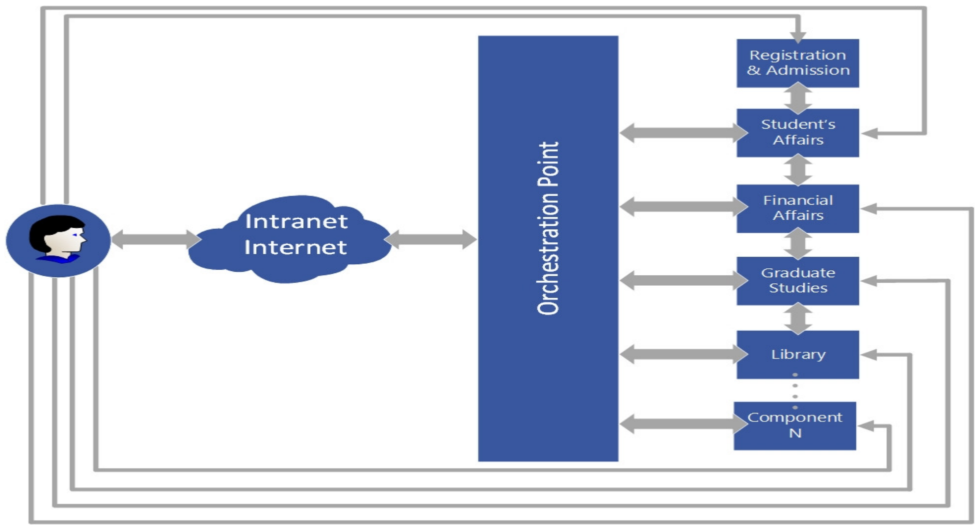


Figure1. The students' portal framework based on SOA

Figure 1 shows the students' portal framework based on SOA, which provides distribution and interoperability for the framework components. Section II discusses the requirements analysis of developing a prototype that proves the concepts of the framework shown in figure 1. The methodology followed to do the prototype is Scrum methodology [2-4] .

1.1.Analysis

Requirements analysis is the first activity that was performed in the life cycle of this prototype development to determine the whole backlog (requirements) of the prototype. The source considered for gathering the requirements of the prototype is the theoretical framework specifications and the meta-model explored in section III.

1.1.1.Requirements Determination

The whole prototype backlog (requirements) is the development of an SOA-based students' portal prototype that is based on the theoretical framework shown in figure 1.

1.1.2.Backlog Division

After determining the whole product backlog as the total product backlog is divided into items, the items of this product backlog are estimated or sized, because, knowing the size of the item is cost indicator. It helps prioritize the product backlog and facilitates planning the prototype. In addition, detailed task-level estimates are determined in the sprint planning. Furthermore, tasks and their estimates are captured in the sprint backlog.

The prototype backlog evolved, and its contents changed frequently. New items are discovered and added to the prototype backlog based on experts' feedback. Moreover, existing items are modified, reprioritized, refined, or removed. That is why the prototype product backlog was a dynamic artifact.

Then, all sprints of the prototype backlog were prioritized. The most important and highest-priority sprints were implemented first. These were found at the top of the prototype backlog. Once a sprint was done, it was removed from the prototype backlog. Some prioritization factors are considered to be useful, and these are the following:

- Sprint value: a sprint is considered valuable if it is necessary for achieving functional requirements of the prototype. If that is not the case, the sprint is irrelevant; and it is excluded from the top priority sprints. The second option keeps the prototype backlog simple and the researcher focused.

- Knowledge, uncertainty, and risk: Because risk and uncertainty influence prototype success, uncertain and risky sprints occupied high priorities. This speeds up the generation of new knowledge of the students' portal functional requirements of the prototype. In addition, it removes uncertainty, and reduces risk.

- Dependencies: Dependencies in the product backlog are an available fact. Functional requirements, for instance, often depend on other functional and even nonfunctional requirements.

Prioritization directed the team's work by focusing the team on the most important sprints of the prototype. It also enhances the management of the sprints.

Based on prioritization factors, the whole prototype backlog is divided into sprints in this order:

a. Database Sprint
b. Coding Students' Portal Components Sprint
c. Web Service Testing Sprint
d. Compatibility Testing Sprint

After dividing the prototype backlog into sprints, the requirements of every sprint are detailed in a meeting at the beginning of every sprint. In the same meeting, the general design of the sprint items is discussed among the research team and detailed design continued in the sprint. The team organized frequent discussion meetings while coding, and coding itself includes some design aspects. The design is considered done only when the sprint is done successfully. As mentioned in the section of backlog division, database sprint occupies the highest priority among other sprints.

1.1.3.The New Students' Portal Framework Design

The new student portal framework meta-model is designed in this section to support and represent the new students' portal framework and its concepts. A modeling language can be either a graphical or textual language [5-7].

For the purpose of designing the new student portal framework, Unified Modeling Language (UML) is used as the graphical modeling language [5, 8-9],UML is formally defined by a meta-model (or semantic model) and it is used to represent software design since it is widely used for modeling both research and industry works [5, 10-12]. UML provides notations for specifying the packaging of a logical design into components that represent a distributed computing architecture and this can be modeled using a UML component diagram [5, 8, 12]. Furthermore, according to [8], several approaches have been proposed to model different aspects of a student portal, such as the conceptual model of the student portal, and the design of the Student Portal processes. However, few efforts have been dedicated to the modeling of the components' physical design of a student portal from the early stages of a student portal project.

Component diagram supported by other UML notations is used to create a meta-model for the new framework. The components of the meta-model and the relationships among these components are explored in this section. Figure 2 represents a meta-model for the distributed interoperable students' portal framework.

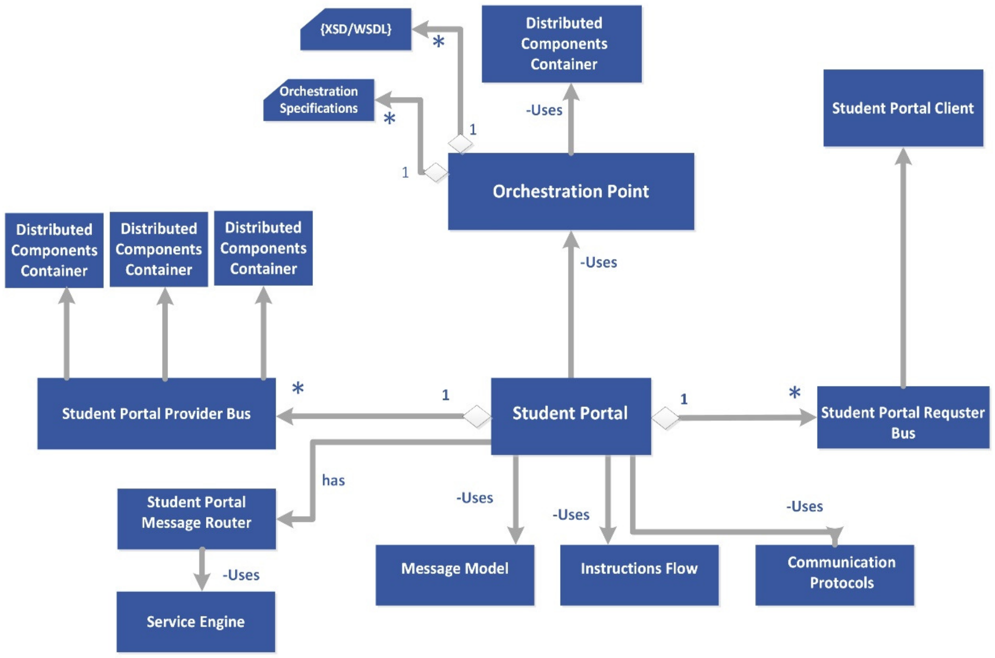
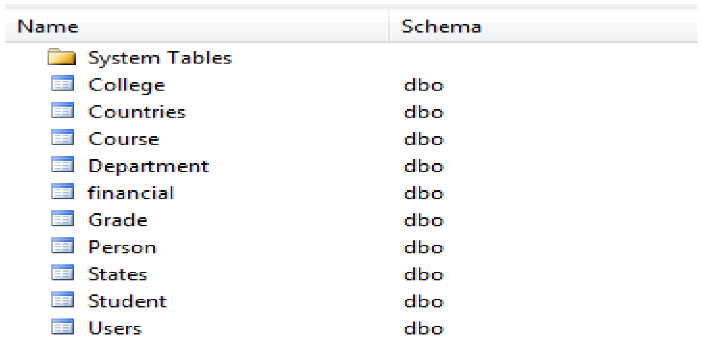

{XSD/WSDL}
Distributed Components Container
Orchestration Specifications
*
*
1
1
-Uses
Student Portal Client
Orchestration Point
Distributed Components Container
Distributed Components Container
Distributed Components Container
-Uses
Student Portal Provider Bus
*
1
Student Portal
1
*
Student Portal Requster Bus
has
Student Portal Message Router
-Uses
-Uses
-Uses
-Uses
Message Model
Instructions Flow
Communication Protocols
Service Engine

Figure 2: Meta-Model for Interoperable and Distributed Students' Portal Framework Components Based on SOA

2.Database Sprint

2.1.Database Table Structure

The component of Registration and Admission was chosen as a sample component for the prototype. The prototype used a students' database schema. The database was used as a sample for developing, deploying and testing the students' portal components of the new students' portal framework. In addition, The student database consists a number of tables such as(College, Student, Grade, financial, Department ,Course, Countries and Users). These tables stores data for students and other related data.

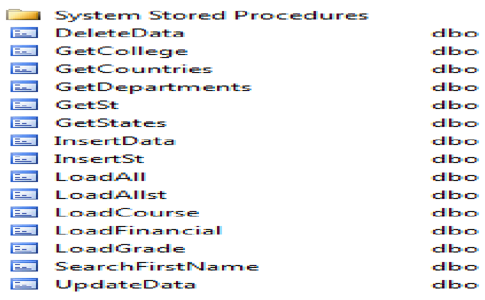


Name	Schema
System Tables	
College	dbo
Countries	dbo
Course	dbo
Department	dbo
financial	dbo
Grade	dbo
Person	dbo
States	dbo
Student	dbo
Users	dbo

Figure 3: Students' Portal Database Tables

2.2.Stored Procedures Creation

Among the processes execution of the students' portal, necessary database is generated. Therefore; the databases could be prepared before the execution. In this prototype, the database generation is automated to make the testing process more simple and clear. The generated database used stored procedures, and some of these procedures used to insert data to SQL database (InsertData, InsertSt), some of them were used to get data from the database (GetCountries ,GetCollege ,...etc.) and some were used for extracting data from the database (LoadCourse , LoadFinancial, LoadGrade, LoadAllst, LoadAll). In addition, there are procedures used to update and delete (DeleteData, UpdateData).



System Stored Procedures	
DeleteData	dbo
GetCollege	dbo
GetCountries	dbo
GetDepartments	dbo
GetSt	dbo
GetStates	dbo
InsertData	dbo
InsertSt	dbo
LoadAll	dbo
LoadAllst	dbo
LoadCourse	dbo
LoadFinancial	dbo
LoadGrade	dbo
SearchFirstName	dbo
UpdateData	dbo

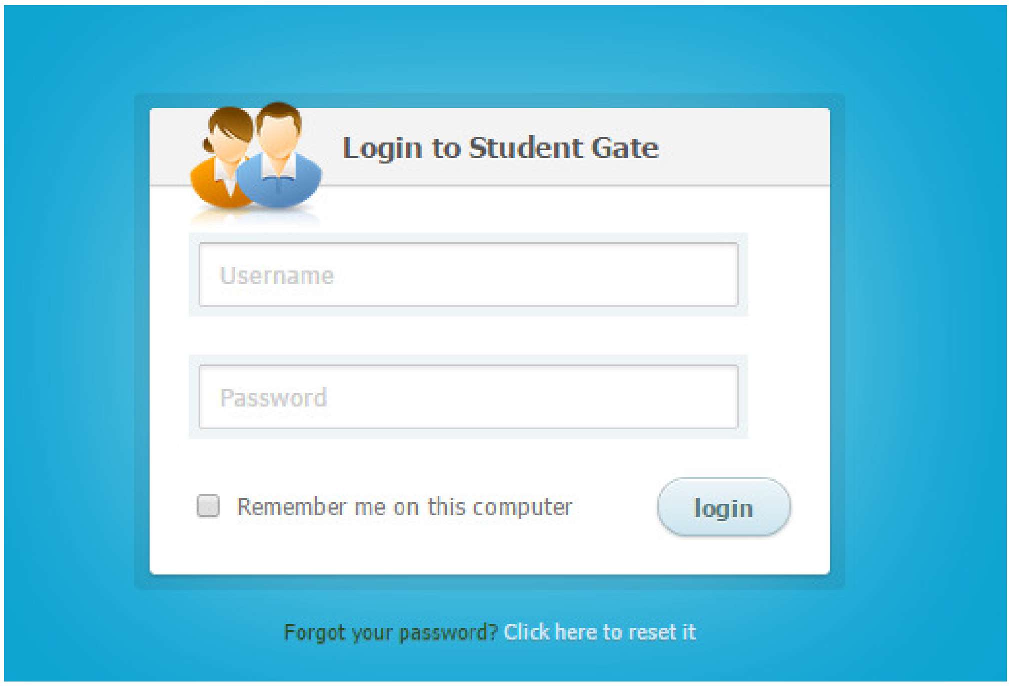
Figure 4: Database Stored Procedures

2.3.Coding Students' Portal Components Sprint

After completing the database sprint successfully, this sprint is done to accomplish the business logic coding of the students' portal components. C# programming language is chosen as the core language to implement this prototype, because it has some advantages over other languages. In conducting this sprint, there are two options to meet the theoretical framework specifications: The first is to do coding using Visual Studio. Therefore, the following tools are used to code the business logic of the Students' Portal components:

- Visual Studio 2012.
- SQL Server 2008 R2.
- IIS web Server.
- Web browsers: Internet Explorer, Mozilla Firefox and Google Chrome.
-

Coding portal gate (SOA Client) passed through three stages, the first stage was creating Students' Portal Gate Layout using HTML and CSS , students' portal gate is shown in figure 5.


Login to Student Gate
Username
Password
Remember me on this computer
login
Forgot your password? Click here to reset it

After that, the business object (BO) was created, BO stores and passes data which comes from business object layer (BOL) Class to display it on the students' portal, and figure 6 shows the business object layer created for the students' portal.

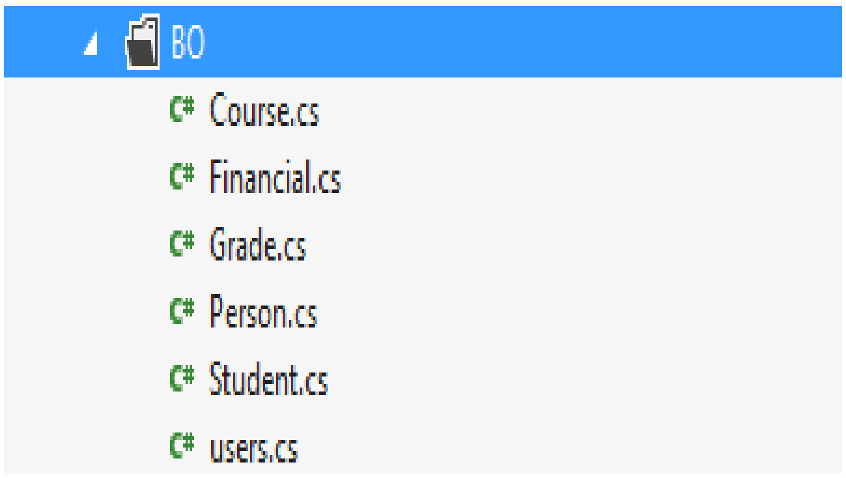

BO
Course.cs
Financial.cs
Grade.cs
Person.cs
Student.cs
users.cs

Figure 6: Business Object Classes

Then business object layer (BOL) classes were created. These classes will get and pass data from SOA Client to data access layer (DAL). Figure 7 shows some of the classes created in BOL.

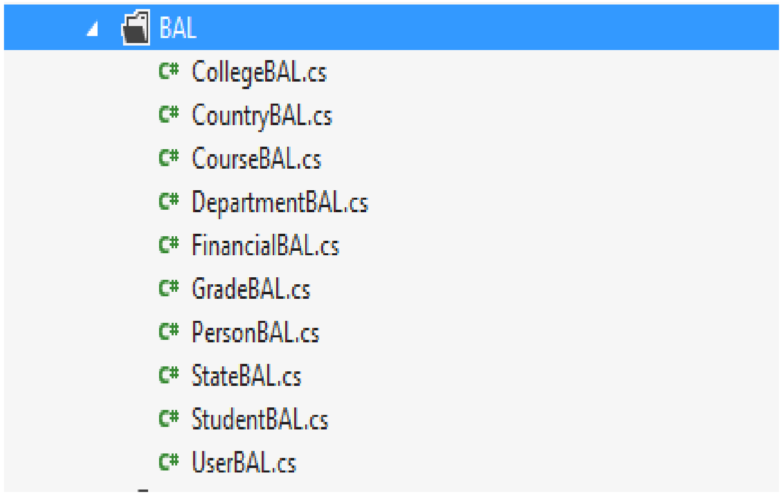

BAL
CollegeBAL.cs
CountryBAL.cs
CourseBAL.cs
DepartmentBAL.cs
FinancialBAL.cs
GradeBAL.cs
PersonBAL.cs
StateBAL.cs
StudentBAL.cs
UserBAL.cs

Figure 7: Business Object Layer Classes

Figure 8 shows the created data access layer (DAL) classes, which takes data from business object layer and passes it to the database using web services, and loads data from database using web services to business object layer to display data on students' portal gate.

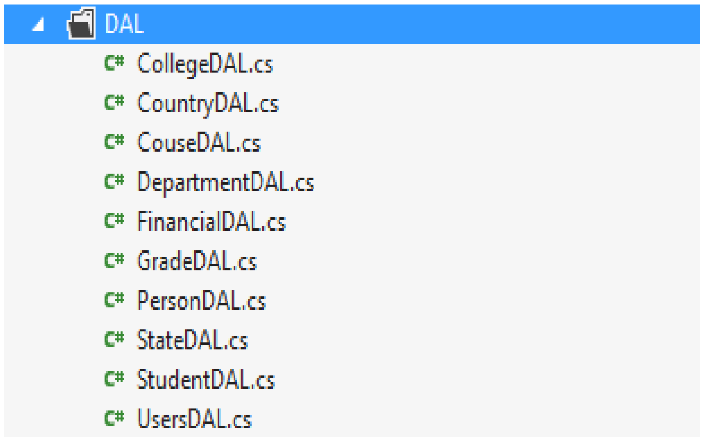
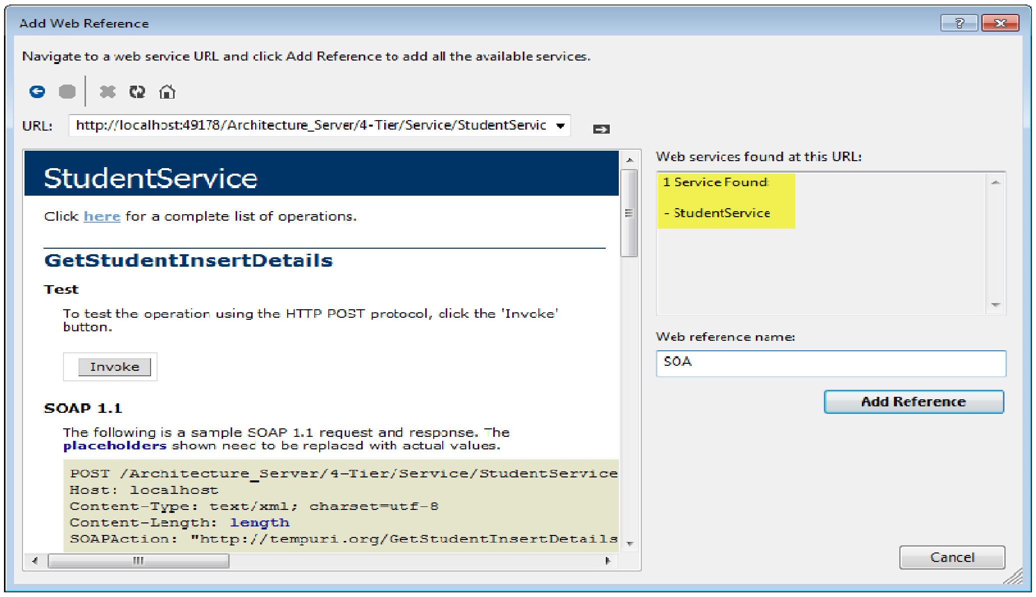

DAL
CollegeDAL.cs
CountryDAL.cs
CouseDAL.cs
DepartmentDAL.cs
FinancialDAL.cs
GradeDAL.cs
PersonDAL.cs
StateDAL.cs
StudentDAL.cs
UsersDAL.cs

Figure 8: Data Access Layer Classes

2.4. Web Service Testing Sprint

The testing sprints are done for the purpose of validating and verifying that the prototype meets the requirements that are listed in section II to validate that the prototype works as expected and can be implemented with the same required characteristics. Figures 9 to 12 show the tested students' portal web services (StudentService, GradeService, CourseService, FinicialService).


Add Web Reference

Navigate to a web service URL and click Add Reference to add all the available services.

URL: http://localhost:49178/Architecture_Server/4-Tier/Service/StudentServic

StudentService

Click here for a complete list of operations.

GetStudentInsertDetails

Test

To test the operation using the HTTP POST protocol, click the 'Invoke' button.

Invoke

SOAP 1.1

The following is a sample SOAP 1.1 request and response. The placeholders shown need to be replaced with actual values.

```
POST /Architecture_Server/4-Tier/Service/StudentService
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/GetStudentInsertDetails
```

Web services found at this URL:

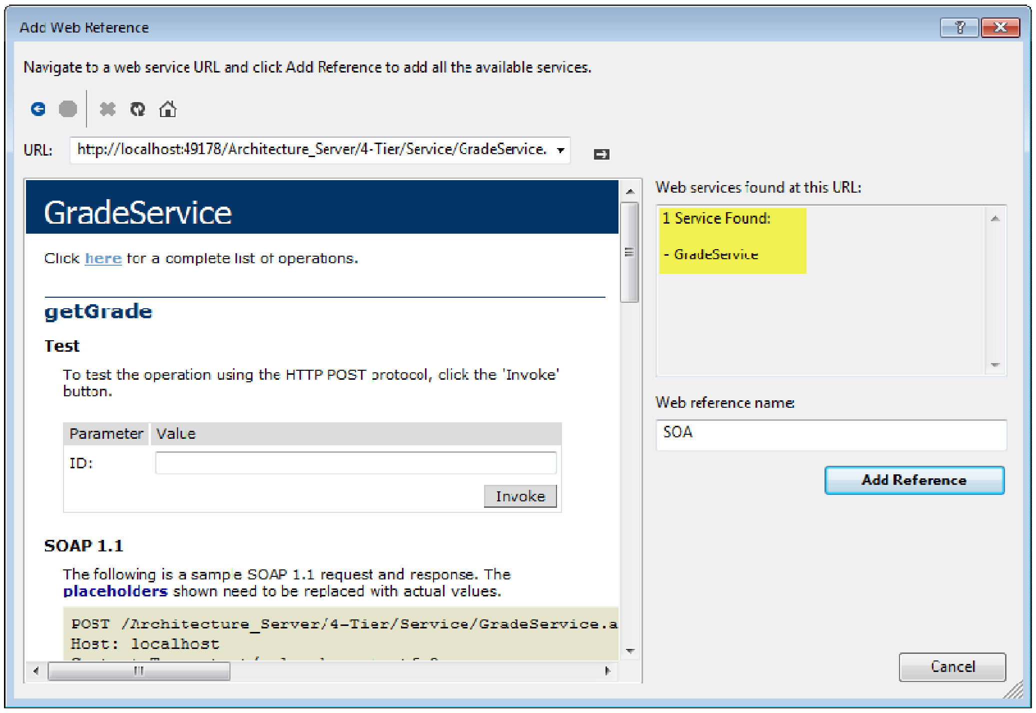
1 Service Found:
- StudentService

Web reference name: SOA

Add Reference

Cancel

Figure 9: Students' Portal Service Testing (StudentService)



Add Web Reference

Navigate to a web service URL and click Add Reference to add all the available services.

URL: http://localhost:49178/Architecture_Server/4-Tier/Service/GradeService.

GradeService

Click here for a complete list of operations.

getGrade

Test

To test the operation using the HTTP POST protocol, click the 'Invoke' button.

Parameter	Value
ID:	

SOAP 1.1

The following is a sample SOAP 1.1 request and response. The **placeholders** shown need to be replaced with actual values.

```
POST /Architecture_Server/4-Tier/Service/GradeService.a
Host: localhost
```

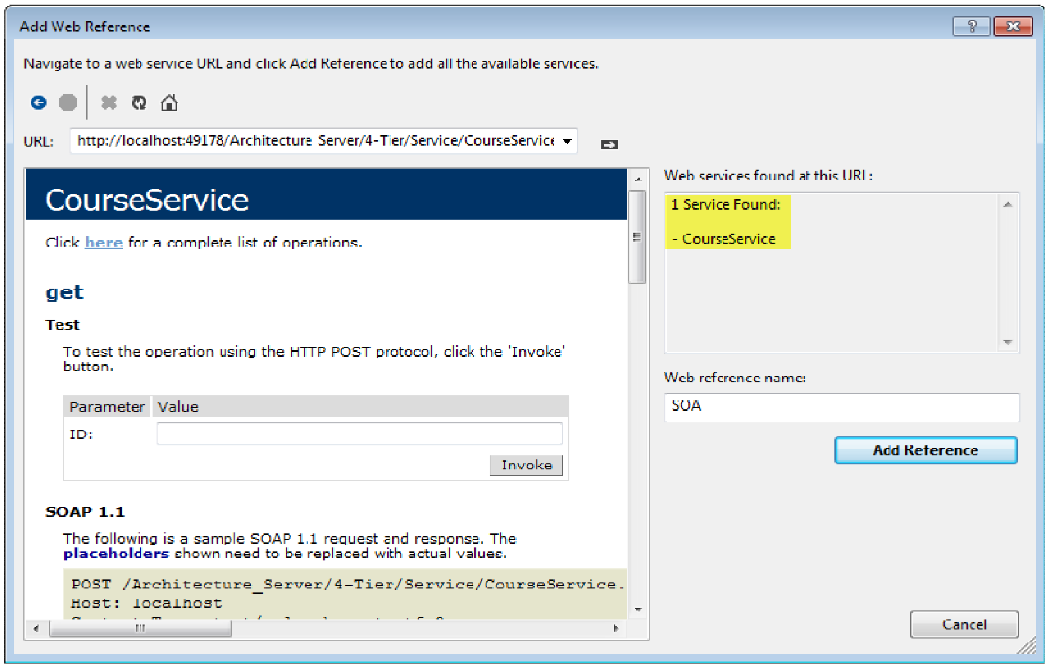
Web services found at this URL:

1 Service Found:

- GradeService

Web reference name: SOA

Figure 10: Students' Grade Service Testing(GradeService)



Add Web Reference

Navigate to a web service URL and click Add Reference to add all the available services.

URL: http://localhost:49178/Architecture_Server/4-Tier/Service/CourseService

CourseService

Click here for a complete list of operations.

get

Test

To test the operation using the HTTP POST protocol, click the 'Invoke' button.

Parameter	Value
ID:	

SOAP 1.1

The following is a sample SOAP 1.1 request and response. The **placeholders** shown need to be replaced with actual values.

```
POST /Architecture_Server/4-Tier/Service/CourseService.
Host: localhost
```

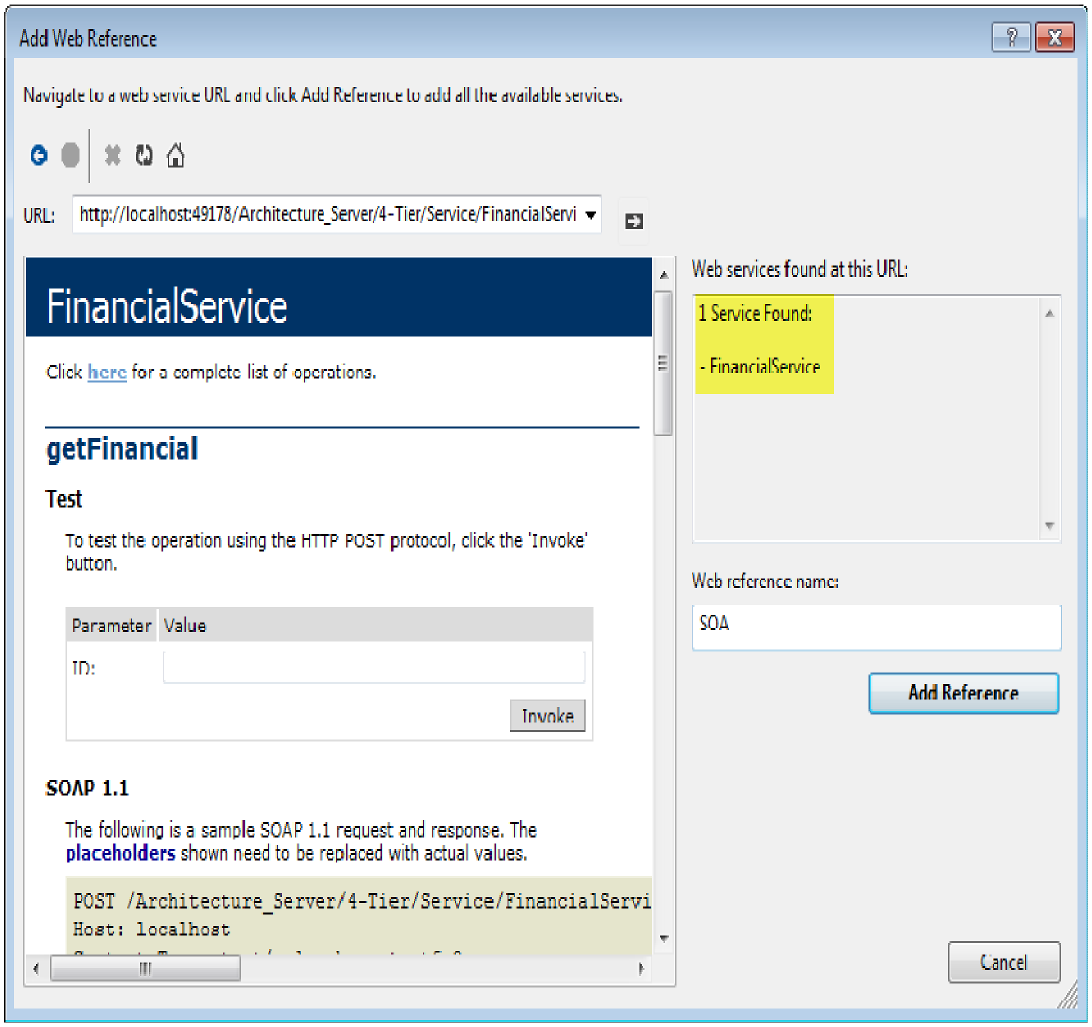
Web services found at this URL:

1 Service Found:

- CourseService

Web reference name: SOA

Figure 11: Students' Course Service Testing (CourseService)


Add Web Reference
Navigate to a web service URL and click Add Reference to add all the available services.
URL:
Web services found at this URL:
Web reference name:
Add Reference
Cancel
Parameter
Value
Invoke

http://localhost:49178/Architecture_Server/4-Tier/Service/FinancialServi

FinancialService

Click here for a complete list of operations.

getFinancial

Test

To test the operation using the HTTP POST protocol, click the 'Invoke' button.

ID:

SOAP 1.1

The following is a sample SOAP 1.1 request and response. The placeholders shown need to be replaced with actual values.

```
POST /Architecture_Server/4-Tier/Service/FinancialServi
Host: localhost
```

1 Service Found:

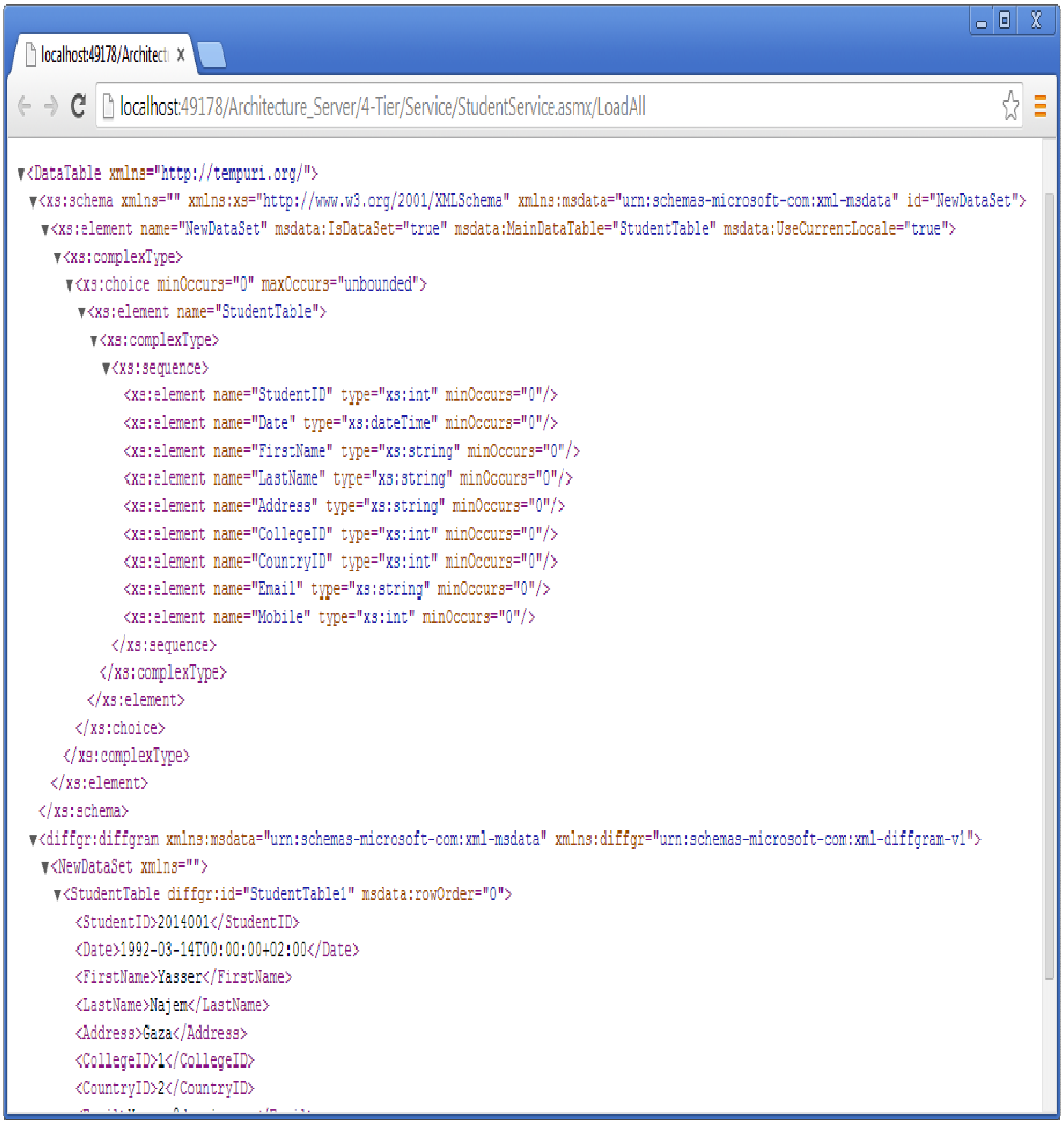
- FinancialService

SOA

Figure 12: Students' Course Service Testing (FinancialService)

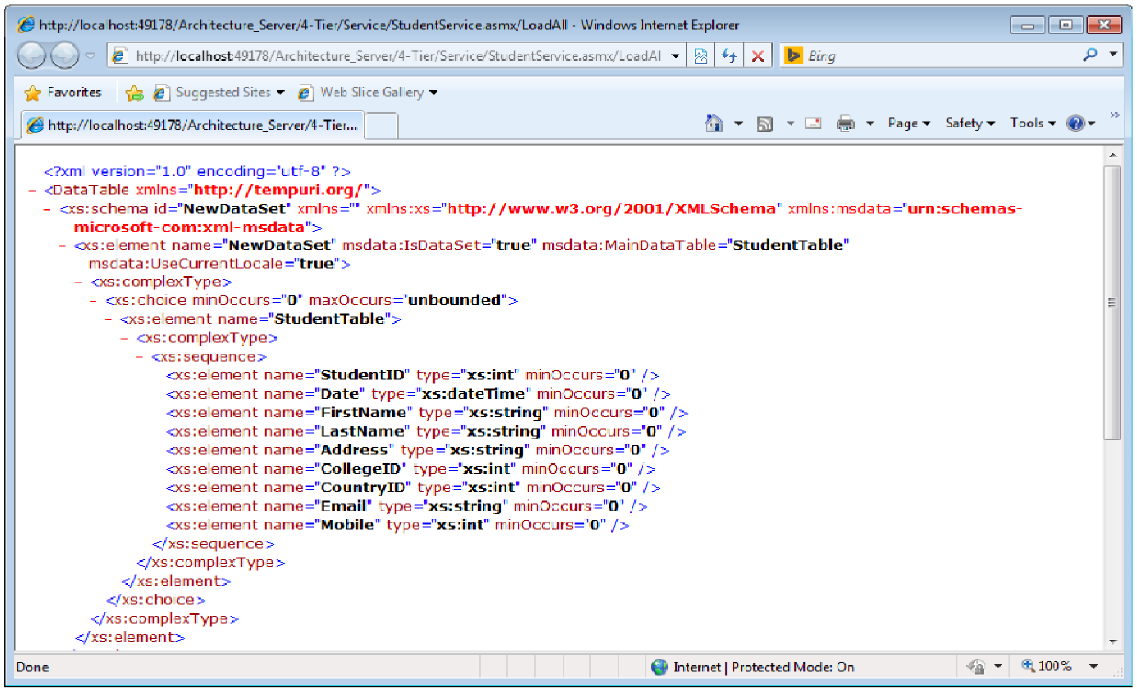
Compatibility Testing Sprint

The compatibility testing was done to test the compatibility of the prototype with different browsers. Microsoft Internet Explorer, Mozilla Firefox and Google Chrome web browsers were used to test the browser compatibility, and the results showed that it works exactly the same with the three browsers as XML data.


localhost:49178/Architect
localhost:49178/Architecture_Server/4-Tier/Service/StudentService.asmx/LoadAll

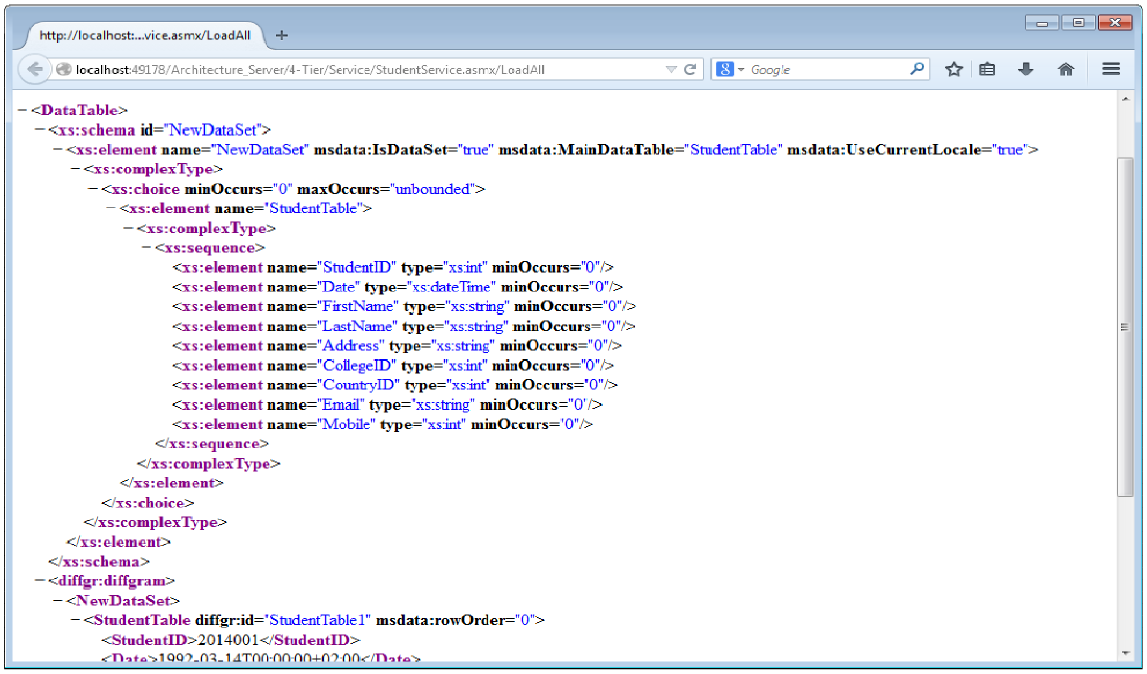
```
▼<DataTable xmlns="http://tempuri.org/">
  ▼<xs:schema xmlns="" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:msdata="urn:schemas-microsoft-com:xml-msdata" id="NewDataSet">
    ▼<xs:element name="NewDataSet" msdata:IsDataSet="true" msdata:MainDataTable="StudentTable" msdata:UseCurrentLocale="true">
      ▼<xs:complexType>
        ▼<xs:choice minOccurs="0" maxOccurs="unbounded">
          ▼<xs:element name="StudentTable">
            ▼<xs:complexType>
              ▼<xs:sequence>
                  <xs:element name="StudentID" type="xs:int" minOccurs="0"/>
                  <xs:element name="Date" type="xs:dateTime" minOccurs="0"/>
                  <xs:element name="FirstName" type="xs:string" minOccurs="0"/>
                  <xs:element name="LastName" type="xs:string" minOccurs="0"/>
                  <xs:element name="Address" type="xs:string" minOccurs="0"/>
                  <xs:element name="CollegeID" type="xs:int" minOccurs="0"/>
                  <xs:element name="CountryID" type="xs:int" minOccurs="0"/>
                  <xs:element name="Email" type="xs:string" minOccurs="0"/>
                  <xs:element name="Mobile" type="xs:int" minOccurs="0"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:choice>
        </xs:complexType>
      </xs:element>
    </xs:schema>
  ▼<diffgr:diffgram xmlns:msdata="urn:schemas-microsoft-com:xml-msdata" xmlns:diffgr="urn:schemas-microsoft-com:xml-diffgram-v1">
    ▼<NewDataSet xmlns="">
      ▼<StudentTable diffgr:id="StudentTable1" msdata:rowOrder="0">
          <StudentID>2014001</StudentID>
          <Date>1992-03-14T00:00:00+02:00</Date>
          <FirstName>Yasser</FirstName>
          <LastName>Najem</LastName>
          <Address>Gaza</Address>
          <CollegeID>1</CollegeID>
          <CountryID>2</CountryID>
```

Figure 13:Compatibility of Students' Portal on Google Chrome Browser


http://localhost:49178/Architecture_Server/4-Tier/Service/StudentService.asmx/LoadAll - Windows Internet Explorer
Favorites
Suggested Sites
Web Slice Gallery
Page
Safety
Tools
Done
Internet | Protected Mode: On
100%

```
<?xml version="1.0" encoding="utf-8" ?>
- <DataTable xmlns="http://tempuri.org/">
  - <xs:schema id="NewDataSet" xmlns="" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:msdata="urn:schemas-
      microsoft-com:xml-msdata">
    - <xs:element name="NewDataSet" msdata:IsDataSet="true" msdata:MainDataTable="StudentTable"
        msdata:UseCurrentLocale="true">
      - <xs:complexType>
        - <xs:choice minOccurs="0" maxOccurs="unbounded">
          - <xs:element name="StudentTable">
            - <xs:complexType>
              - <xs:sequence>
                  <xs:element name="StudentID" type="xs:int" minOccurs="0" />
                  <xs:element name="Date" type="xs:dateTime" minOccurs="0" />
                  <xs:element name="FirstName" type="xs:string" minOccurs="0" />
                  <xs:element name="LastName" type="xs:string" minOccurs="0" />
                  <xs:element name="Address" type="xs:string" minOccurs="0" />
                  <xs:element name="CollegeID" type="xs:int" minOccurs="0" />
                  <xs:element name="CountryID" type="xs:int" minOccurs="0" />
                  <xs:element name="Email" type="xs:string" minOccurs="0" />
                  <xs:element name="Mobile" type="xs:int" minOccurs="0" />
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:choice>
        </xs:complexType>
      </xs:element>
```

Figure 14: Compatibility of Students' Portal on Internet Explorer Browser


http://localhost:...vice.asmx/LoadAll
localhost:49178/Architecture_Server/4-Tier/Service/StudentService.asmx/LoadAll
Google

```
- <DataTable>
  - <xs:schema id="NewDataSet">
    - <xs:element name="NewDataSet" msdata:IsDataSet="true" msdata:MainDataTable="StudentTable" msdata:UseCurrentLocale="true">
      - <xs:complexType>
        - <xs:choice minOccurs="0" maxOccurs="unbounded">
          - <xs:element name="StudentTable">
            - <xs:complexType>
              - <xs:sequence>
                  <xs:element name="StudentID" type="xs:int" minOccurs="0"/>
                  <xs:element name="Date" type="xs:dateTime" minOccurs="0"/>
                  <xs:element name="FirstName" type="xs:string" minOccurs="0"/>
                  <xs:element name="LastName" type="xs:string" minOccurs="0"/>
                  <xs:element name="Address" type="xs:string" minOccurs="0"/>
                  <xs:element name="CollegeID" type="xs:int" minOccurs="0"/>
                  <xs:element name="CountryID" type="xs:int" minOccurs="0"/>
                  <xs:element name="Email" type="xs:string" minOccurs="0"/>
                  <xs:element name="Mobile" type="xs:int" minOccurs="0"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:choice>
        </xs:complexType>
      </xs:element>
    </xs:schema>
  - <diffgr:diffgram>
    - <NewDataSet>
      - <StudentTable diffgr:id="StudentTable1" msdata:rowOrder="0">
          <StudentID>2014001</StudentID>
          <Date>1992-03-14T00:00:00+02:00</Date>
```

Figure 15: Compatibility of Students' Portal on Mozilla Firefox Browser

3.CONCLUSION AND FUTURE WORK

This paper has proved the concept of loose coupling for students’ portal framework. It has explored the analysis, design, development and testing of the SOA-based students' portal prototype, which was based on the specifications and recommendations of the theoretical framework for distributed students' portal components. This research work did not discuss the hosting characteristics of the resulted portal to get the maximum advantages of the SOA features, especially the cloud computing hosting, which is considered as a future research work.

References

1. Rami A.Y Raba , M.M.I.A., Students' portal architecture based on SOA. IEEE, 2015(Control, Engineering & Information Technology (CEIT), 2015 3rd International Conference): p. 1 - 4.
2. Sutherland, J., Viktorov, A., Blount, J., & Puntikov, N. , Distributed Scrum: Agile Project Management with Outsourced Development Teams. Proceedings of 40th Hawaii International Conference on System Sciences, 2007.
3. Rayhan, S.H., & Haque, N., Incremental Adoption of Scrum for Successful Delivery of an IT Project in a Remote Setup. 2008(Proceedings of Agile 2008 Conference).
4. Cardozo, E., Neto, J. B. F. A., Barza, A., França, A., & da Silva, F. , SCRUM and productivity in software projects: a systematic literature review. EASE, 2010(In 14th International Conference on Evaluation and Assessment in Software Engineering).
5. Kobryn, c., Modeling Components and Framework with UML. Communications of the ACM, 2000: p. 43 (10).
6. Yong Xia, M.G., Rigorous EBNF-based Definition for a Graphic Modeling Language. Winterthurerstr, 190,CH-8057 Zurich, Switzerland, 2002.
7. OMG, Common Warehouse Metamodel (CWM) Specification. USA : OMG Headquarters., 2003. 1st ed. Vol.1.
8. Lujanmora, J.T., Physical Modeling of Data Werehouses Using UML. ACM Journal., 2004.
9. Koch, N., & Kraus, A. , The expressive power of uml-based web engineering. IWWOST02, 2002. 16(In Second International Workshop on Web-oriented Software Technology).
10. Kruchten, W.K., Bran, & Slice, Describing Software Architecture with UML. Relational Software, 2001.
11. Atkinson, T.K.H., Rearchitecting the UML Infrastructure. ACM Transactions on Modeling and Simulation, 2002. 12 (4): p. 290-321.
12. OMG, Unified Modeling Language. from http://www.uml.org/, 2011.