2023

Enterprise Systems Integration

ASSIGNMENT 3 (GROUP): SYSTEM INTEGRATION

Prepared By Group 17	
Group Members	**Student IDs**
Muhammad Waleed Qasim	48074853
Abdullah Shahid	45761000
Hammad Shaikh	48072184
Ryan Staskun	46489207

Dated: 31st October 2023

Contents

Project Brief

1. Group Meeting Schedule or Plan:

- Our team convenes for weekly meetings every Tuesday evening via Zoom to discuss the project's progress and address any concerns or questions.
- In addition to our virtual meetings, we capitalize on our shared practical class time for face-to-face interactions and project discussions.
- The flexibility in our meeting schedule allows us to make adjustments as required, pending unanimous approval from all group members.

2. Each Member's Role in the Project:

- **Muhammad Waleed Qasim**: As the lead for Task 1, Muhammad is delving into the intricacies of Collaborative Business Process and EAI, ensuring a thorough and detailed analysis and report of this critical project component.
- **Abdullah Shahid**: Abdullah has taken on the pivotal role of managing Task 2, where he is meticulously working on Web Service Design, a task that requires both technical acumen and creative problem-solving skills.
- **Hammad Shaikh**: Task 3 is in the capable hands of Hammad, who is tasked with the complex process of XML Schema Design, ensuring that the project meets the required specifications and standards.
- **Ryan Staskun**: Ryan is steering Task 4, where he is tasked with compiling an 800-word comprehensive analysis for the Discussion section of the project, bringing together all the elements of the project into a cohesive narrative.

  Each member was assigned their respective roles based on their strengths and expertise, ensuring a harmonious and synergistic team dynamic.

3. Identifying Risks and Ways to Manage Risks:

- **Risk 1: A team member not completing their task on time.**
  - **Management Strategy**: We conduct regular check-ins during our weekly meetings to gauge each member's progress. If any member is lagging, the team rallies to provide the necessary support and resources to get back on track.
- **Risk 2: Miscommunication or lack of communication among team members.**
  - **Management Strategy**: To foster clear and effective communication, we utilize Facebook Messenger group for regular, real-time updates and resource sharing, complemented by our weekly Zoom meetings for more detailed discussions.
- **Risk 3: Technical issues during online meetings.**
  - **Management Strategy**: Our face-to-face meetings during serve as a safety net, ensuring that any pressing matters or urgent discussions can be addressed in person should technical glitches occur during our online meetings.

4. Ways to Communicate Between Members:

- The team utilizes Facebook Messenger and Discord as our primary communication channels, ensuring real-time updates, resource sharing, and general project-related discussions.
- Our weekly Zoom meetings provide a platform for more in-depth and detailed discussions, allowing each member to voice their opinions, share findings, and address any questions or concerns.
- The face-to-face interactions during our practical class serve as an invaluable opportunity for real-time collaboration and discussion.

Meeting Log:

Week 1: Tuesday, 3rd October

- Discussion about the overall project and the tasks involved.
- Allocation of tasks based on individual strengths.
    - Muhammad Waleed Qasim: Task 1, Collaborative Business Process and EAI.
    - Abdullah Shahid: Task 2, Web Service Design.
    - Hammad Shaikh: Task 3, XML Schema Design.
    - Ryan Staskun: Task 4, Discussion (@ 800 words).
- Setting up the weekly meeting schedule, communication channel (Facebook Messenger), and the Google Docs file for logging.

Week 2: Tuesday, 10th October

- Gathered and reviewed initial research findings for each respective task.
- Engaged in a robust discussion regarding some of the project elements and how they would be covered in upcoming university classes.
- Mohammad and Abdullah shared valuable insights from their research on collaborative business processes and web service design, respectively.
- Hammad and Ryan provided updates on their progress and preliminary findings.
- The group collectively addressed any challenges faced in gathering information and shared resources to assist each other.
- Confirmed the support structure within the group, with each member understanding that they could seek help from other group members when faced with any difficulties.
- Discussed preliminary ideas on how to integrate the different tasks into a cohesive report.

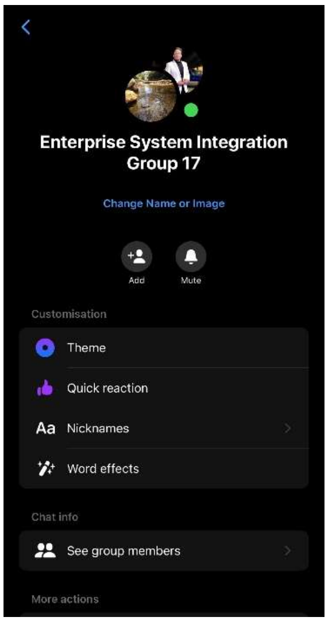
Week 3: Tuesday, 17th October

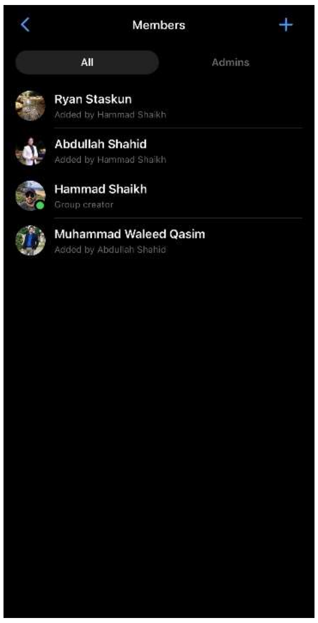
- Reviewed the updated progress of each task.
- Addressed and resolved any issues or challenges faced by group members in their tasks.
- Discussed the integration of the project elements covered in recent university classes.
- Mohammad and Abdullah shared new insights and findings from their ongoing research.
- Hammad and Ryan provided updates on the XML schema design and discussion section, respectively.
- The group brainstormed ideas on how to improve the overall quality of the project and ensure a coherent flow between different sections.
- Discussed and planned for the upcoming peer review and analytics.

Week 4: Tuesday, 24th October

- Conducted a comprehensive review of the work done so far.
- Each member presented their progress and received feedback from other group members.
- Engaged in a constructive peer review session, with each member providing valuable insights and suggestions for improvement.
- Discussed and addressed any last-minute questions or concerns before the upcoming analytics and peer review.
- Finalized the structure and flow of the project report.
- Reviewed the meeting logs and ensured all necessary details were accurately documented.
- Discussed and planned the final steps leading up to the completion of the project.

Facebook Messenger and Zoom Screenshots



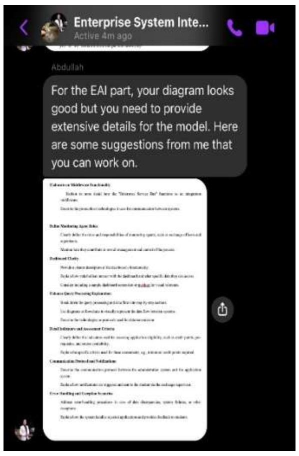


Ryan Staskun
Added by Hammad Shaikh

Abdullah Shahid
Added by Hammad Shaikh

Hammad Shaikh
Group creator

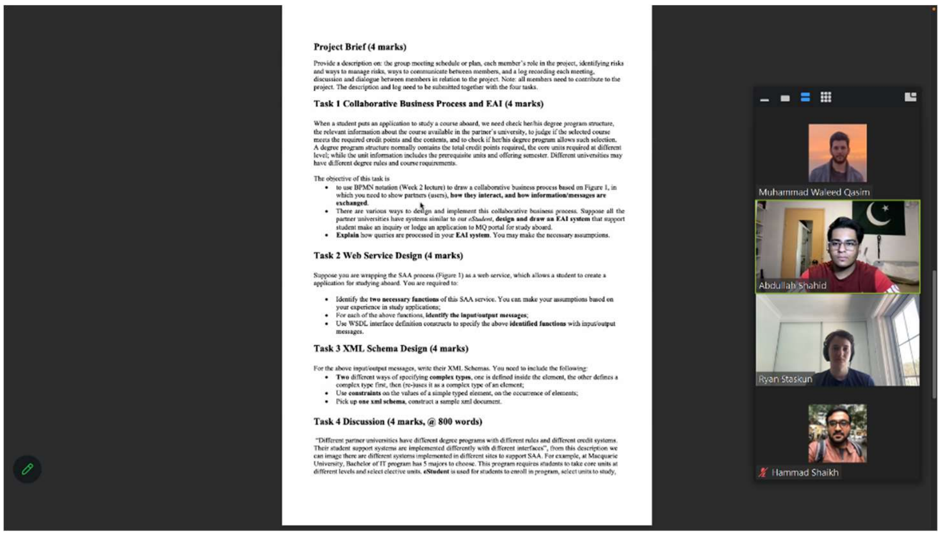
Muhammad Waleed Qasim
Added by Abdullah Shahid



Abdullah

For the EAI part, your diagram looks good but you need to provide extensive details for the model. Here are some suggestions from me that you can work on.

[illegible]



Project Brief (4 marks)

Provide a description on: the group meeting schedule or plan, each member's role in the project, identifying risks and ways to manage risks, ways to communicate between members, and a log recording each meeting, discussion and dialogue between members in relation to the project. Note: all members need to contribute to the project. The description and log need to be submitted together with the four tasks.

Task 1 Collaborative Business Process and EAI (4 marks)

When a student puts an application to study a course aboard, we need check her/his degree program structure, the relevant information about the course available in the partner's university, to judge if the selected course meets the required credit points and the contents, and to check if her/his degree program allows such selection. A degree program structure normally contains the total credit points required, the core units required at different level; while the unit information includes the prerequisite units and offering semester. Different universities may have different degree rules and course requirements.

The objective of this task is

- to use BPMN notation (Week 2 lecture) to draw a collaborative business process based on Figure 1, in which you need to show partners (users), **how they interact, and how information/messages are exchanged.**
- There are various ways to design and implement this collaborative business process. Suppose all the partner universities have systems similar to our *eStudent*, **design and draw an EAI system** that support student make an inquiry or lodge an application to MQ portal for study aboard.
- **Explain** how queries are processed in your **EAI system**. You may make the necessary assumptions.

Task 2 Web Service Design (4 marks)

Suppose you are wrapping the SAA process (Figure 1) as a web service, which allows a student to create a application for studying aboard. You are required to:

- Identify the **two necessary functions** of this SAA service. You can make your assumptions based on your experience in study applications;
- For each of the above functions, **identify the input/output messages**;
- Use WSDL interface definition constructs to specify the above **identified functions** with input/output messages.

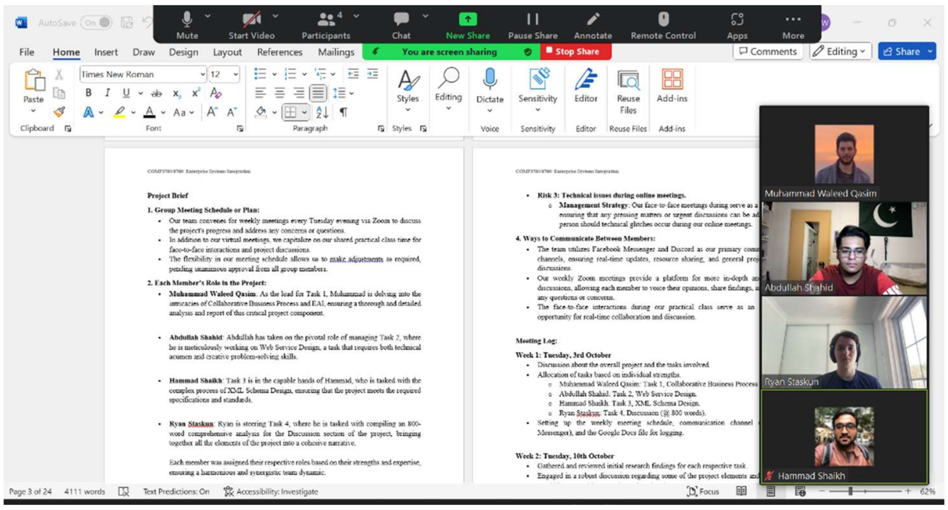
Task 3 XML Schema Design (4 marks)

For the above input/output messages, write their XML Schemas. You need to include the following:

- **Two** different ways of specifying **complex types**, one is defined inside the element, the other defines a complex type first, then (re-)uses it as a complex type of an element;
- Use **constraints** on the values of a simple typed element, on the occurrence of elements;
- Pick up **one xml schema**, construct a sample xml document.

Task 4 Discussion (4 marks, @ 800 words)

"Different partner universities have different degree programs with different rules and different credit systems. Their student support systems are implemented differently with different interfaces", from this description we can image there are different systems implemented in different sites to support SAA. For example, at Macquarie University, Bachelor of IT program has 5 majors to choose. This program requires students to take core units at different levels and select elective units. **eStudent** is used for students to enroll in program, select units to study,



COMP3760/6760: Enterprise Systems Integration

Project Brief

1. Group Meeting Schedule or Plan:

- Our team convenes for weekly meetings every Tuesday evening via Zoom to discuss the project's progress and address any concerns or questions.
- In addition to our virtual meetings, we capitalize on our shared practical class time for face-to-face interactions and project discussions.
- The flexibility in our meeting schedule allows us to make adjustments as required, pending unanimous approval from all group members.

2. Each Member's Role in the Project:

- **Muhammad Waleed Qasim**: As the lead for Task 1, Muhammad is delving into the intricacies of Collaborative Business Process and EAI, ensuring a thorough and detailed analysis and report of this critical project component.
- **Abdullah Shahid**: Abdullah has taken on the pivotal role of managing Task 2, where he is meticulously working on Web Service Design, a task that requires both technical acumen and creative problem-solving skills.
- **Hammad Shaikh**: Task 3 is in the capable hands of Hammad, who is tasked with the complex process of XML Schema Design, ensuring that the project meets the required specifications and standards.
- **Ryan Staskun**: Ryan is steering Task 4, where he is tasked with compiling an 800-word comprehensive analysis for the Discussion section of the project, bringing together all the elements of the project into a cohesive narrative.

Each member was assigned their respective roles based on their strengths and expertise, ensuring a harmonious and synergistic team dynamic.

COMP3760/6760: Enterprise Systems Integration

- **Risk 3: Technical issues during online meetings.**
  - **Management Strategy**: Our face-to-face meetings during serve as a
    ensuring that any pressing matters or urgent discussions can be ad
    person should technical glitches occur during our online meetings.

4. Ways to Communicate Between Members:

- The team utilizes Facebook Messenger and Discord as our primary comm
  channels, ensuring real-time updates, resource sharing, and general proj
  discussions.
- Our weekly Zoom meetings provide a platform for more in-depth an
  discussions, allowing each member to voice their opinions, share findings, a
  any questions or concerns.
- The face-to-face interactions during our practical class serve as an opportunity for real-time collaboration and discussion.

Meeting Log:

Week 1: Tuesday, 3rd October

- Discussion about the overall project and the tasks involved.
- Allocation of tasks based on individual strengths.
  - Muhammad Waleed Qasim: Task 1, Collaborative Business Process
  - Abdullah Shahid: Task 2, Web Service Design.
  - Hammad Shaikh: Task 3, XML Schema Design.
  - Ryan Staskun: Task 4, Discussion (@ 800 words).
- Setting up the weekly meeting schedule, communication channel
  Messenger), and the Google Docs file for logging.

Week 2: Tuesday, 10th October

- Gathered and reviewed initial research findings for each respective task.
- Engaged in a robust discussion regarding some of the project elements and

TASK 1 Collaborative Business Process and EAI

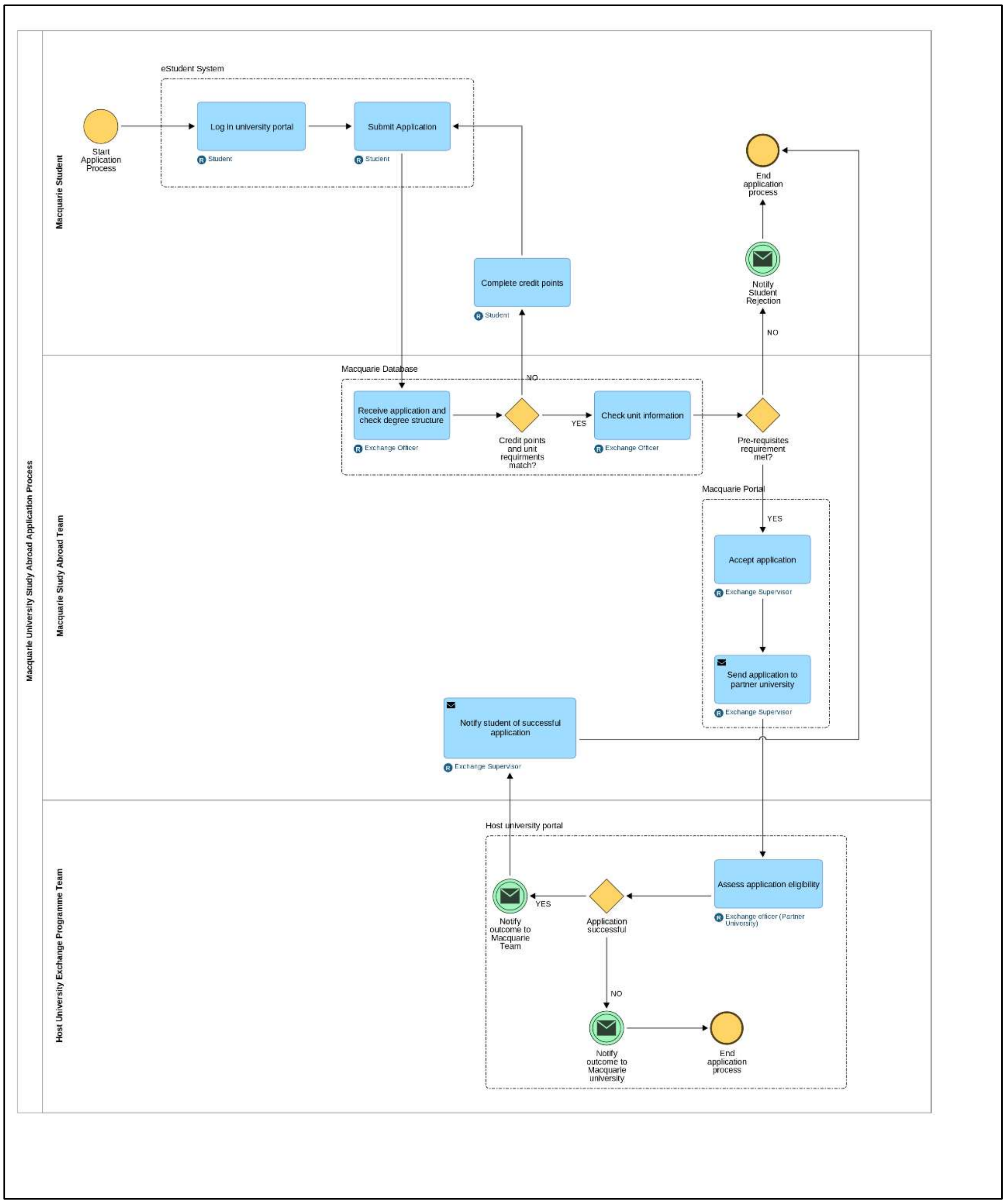

Macquarie University Study Abroad Application Process
Macquarie Student
Macquarie Study Abroad Team
Host University Exchange Programme Team
eStudent System
Start Application Process
Log in university portal
Student
Submit Application
Student
Complete credit points
Student
End application process
Notify Student Rejection
NO
Macquarie Database
Receive application and check degree structure
Exchange Officer
NO
Credit points and unit requirments match?
YES
Check unit information
Exchange Officer
Pre-requisites requirement met?
Macquarie Portal
YES
Accept application
Exchange Supervisor
Send application to partner university
Exchange Supervisor
Notify student of successful application
Exchange Supervisor
Host university portal
Assess application eligibility
Exchange officer (Partner University)
Application successful
YES
Notify outcome to Macquarie Team
NO
Notify outcome to Macquarie university
End application process

Figure 1 Collaborative Study Abroad Application Process (BPMN)

The diagram above is a business process flow using BPMN notations showing a step-by-step sequence of tasks that take place in the high-level study abroad application process and what users are responsible for execution of those tasks.

Assumptions of the BPMN model

- Both study abroad teams at Macquarie and host university are connected via an integrated communication system.
- It is also assumed that the student portal system is uniform among Macquarie and other host universities.
- Macquarie university has access to host university database to streamline the application process and increase the efficiency of overall process. This is further discussed in our Enterprise Application Integration system below.

Highlights of the Process

- The student logs in the eStudent portal and submits an application of interest to study abroad for a chosen course.
- Macquarie university study abroad team receives application, and an exchange officer is assigned to it. The officer is responsible for checking the degree structure student is currently enrolled in.
- If completed credit points of the student matches with the requirements of study abroad course the application is moved to the next stage.
- Next the officer matches the pre-requisite unit requirements of student's degree with study abroad course. If the degree units align with the road map of study abroad course, then the application is accepted by Macquarie university and student is notified of the outcome.
- All relevant degree details are packaged, and exchange supervisor of Macquarie university shares the application with the host university.
- The host university exchange officer of study abroad team reviews the application and performs verification checks. If the assessment is successful a notification is sent to Macquarie university. Otherwise, the host university ends the process and notifies the same to exchange supervisor of the application at Macquarie university.

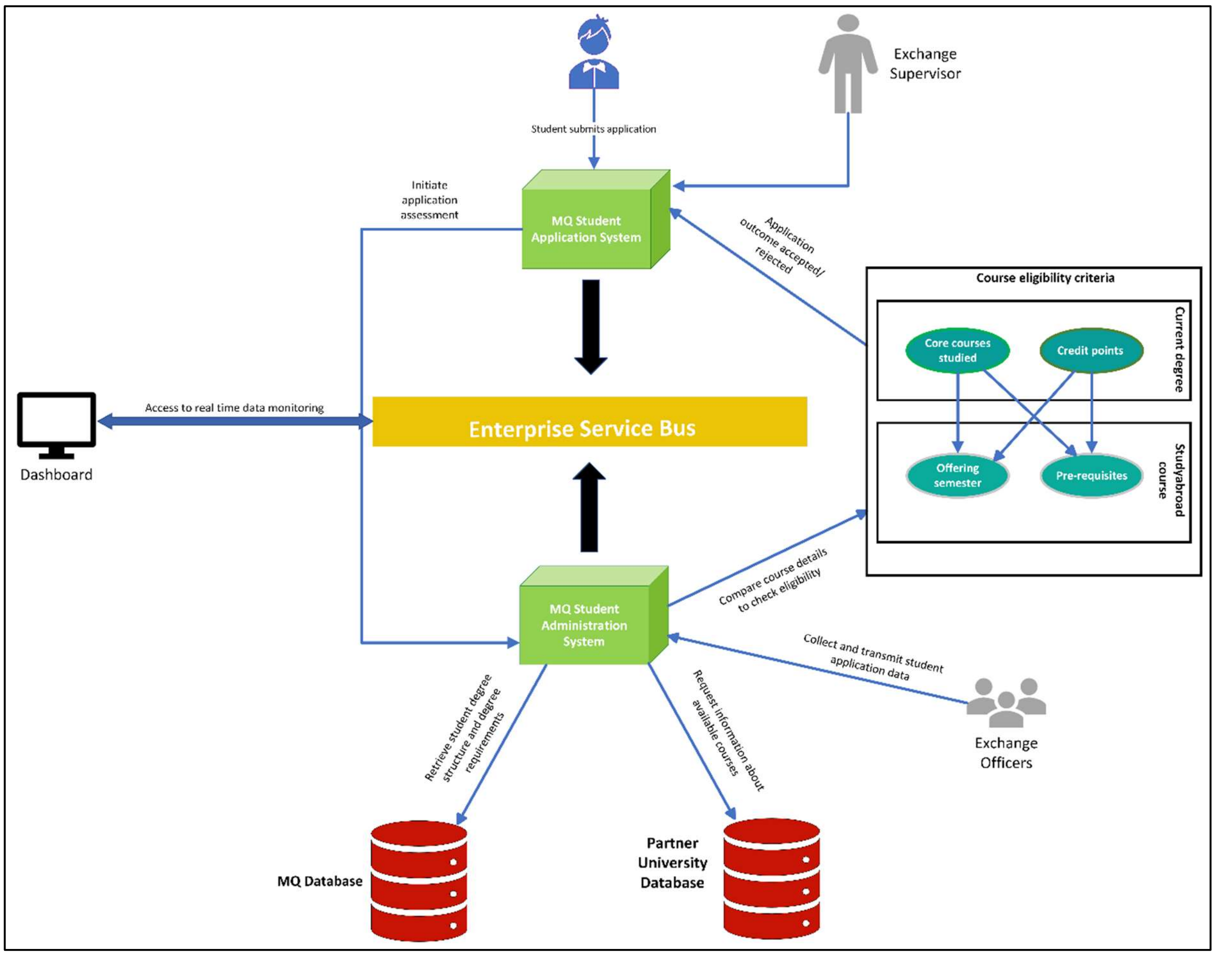

Exchange Supervisor
Student submits application
Initiate application assessment
MQ Student Application System
Application outcome accepted/ rejected
Course eligibility criteria
Core courses studied
Credit points
Current degree
Offering semester
Pre-requisites
Studyabroad course
Access to real time data monitoring
Enterprise Service Bus
Dashboard
Compare course details to check eligibility
MQ Student Administration System
Collect and transmit student application data
Exchange Officers
Retrieve student degree structure and degree requirements
Request information about available courses
MQ Database
Partner University Database

Figure 2 Enterprise Application Integration (Study Abroad Application Process)

The diagram above is an ‘Enterprise Application Integration’ (EAI) system of the business process discussed above. This system caters to the students at Macquarie University when they lodge a study abroad application in the MQ portal.

Assumptions and data flow of EAI system

This EAI system is designed with the following assumptions:

1. Student application system and the student administration system are two distinct systems in Macquarie University to manage this process.
2. There are two databases that keeps the record of all relevant information regarding the degree programs.
3. An ‘Enterprise Service Bus’ plays the role of integration middleware to enable communication and exchange of information between the systems.
4. There are monitoring agents such as the exchange officers and supervisors to provide overall management and control of the process.
5. A dashboard acts as the user interface which sends/receives all data from enterprise service bus for stakeholders involved, so that they can access real time data for performance management and application tracking to main the efficiency of the overall system.

<u>Query processing in EAI system</u>

When a student lodges an application in the student portal, the student application system triggers the administration system by sending a request to review the application. The exchange officer part of the administration system queries the Macquarie database for required degree details including core units and elective units, credit points, degree structure and course outline.

As the system is integrated using middleware the administration system has access to query partner university database to retrieve details of the course student wishes to study abroad. Application eligibility is assessed using indicators like completed credit points, pre-requisites, and semester in which the course is offered to see if the student's current degree aligns with the course of interest, he/she wishes to study abroad.

When the application is assessed against the indicators the administration system communicates with the application system to trigger an outcome to the student via exchange supervisor.

TASK 2 Web Service Design

We explore the workings of the Study Abroad Application (SAA) system in this section, viewing it as a web application that streamlines the application process for students who want to study overseas. We will outline the two key functions that are essential to this service, along with the input and output messages that are required for each. After that, we will use the Web Services Description Language (WSDL) interface definition constructs to precisely define these functions, which will guarantee an application process that is both efficient and user-friendly.

Functions Identified

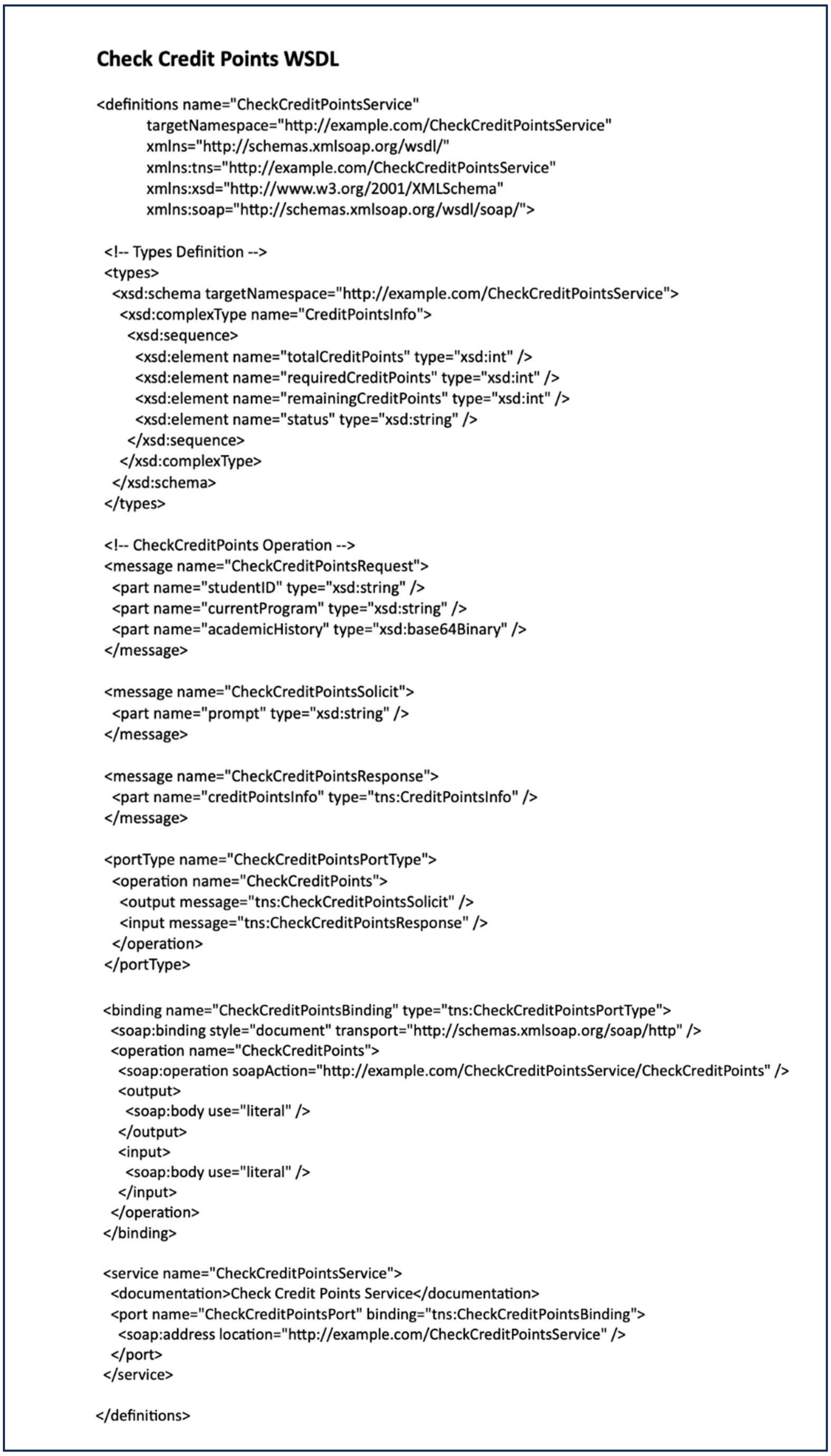
Function 1: CheckCreditPoints

- **Function Name:** CheckCreditPoints
- **Assumptions:**
    - All required fields must be completed before the credit points can be checked.
    - The student's academic history must be available and accessible.
    - The student must be currently enrolled in a study program.
- **Input Messages:**
    - **studentID**: String. The student's unique identification number.
    - **currentProgram**: String. The program in which the student is currently enrolled.
    - **academicHistory**: File. Transcripts and records of the student's academic history.
- **Output Messages:**
    - **totalCreditPoints**: Integer. The total number of credit points the student has earned.
    - **requiredCreditPoints**: Integer. The total number of credit points required for the student's current program.

  - **remainingCreditPoints**: Integer. The number of credit points the student needs to complete their program.
  - **status**: String. A message indicating whether the student meets the credit points requirements for their program.

Function 2: CheckApplicationStatus

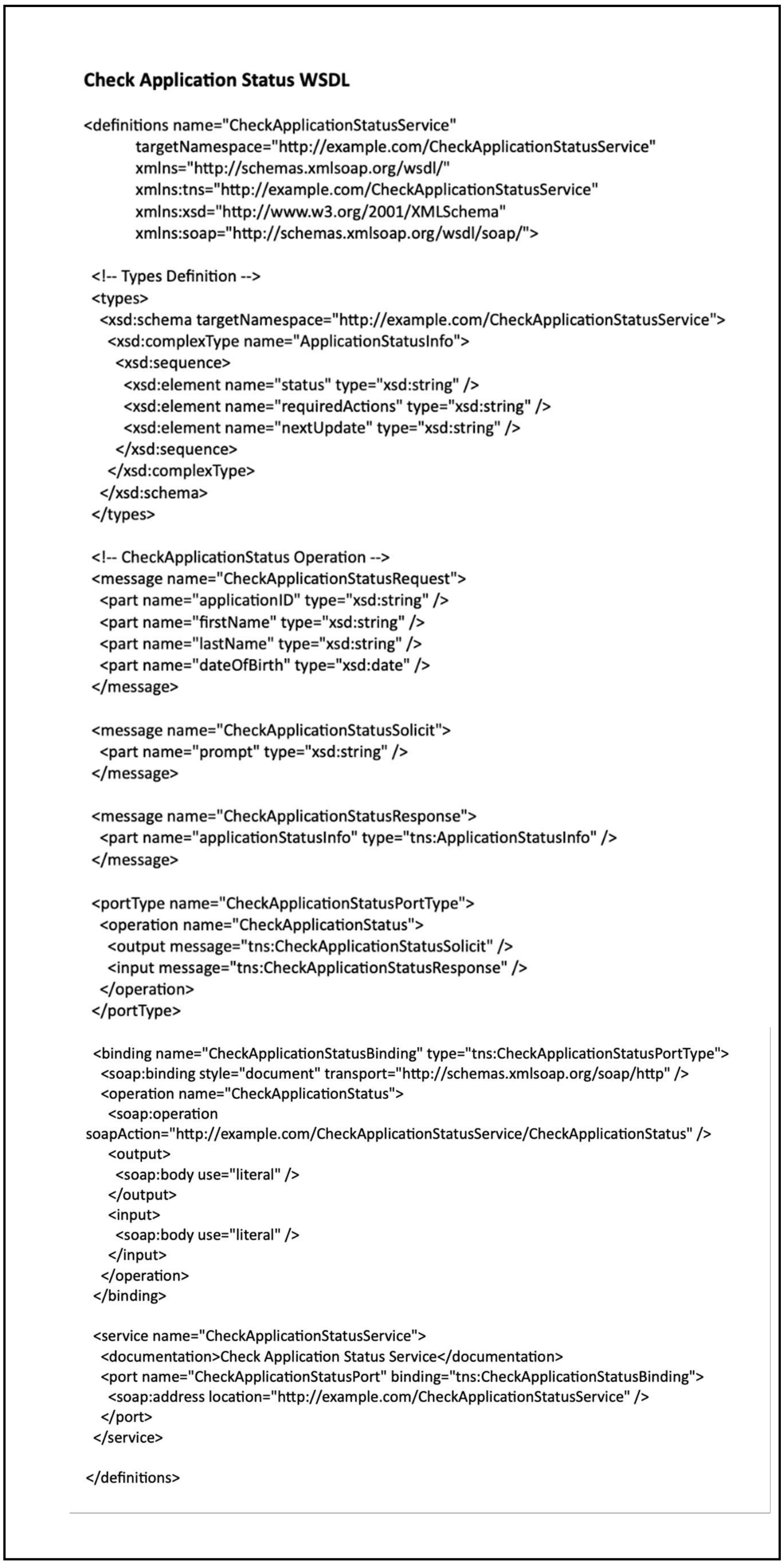
- **Function Name:** CheckApplicationStatus
- **Assumptions:**
  - The student must provide a valid application ID to retrieve the status.
  - The student's personal information must match the information provided during application submission.
- **Input Messages:**
  - **applicationID**: String. The unique identifier for the submitted application.
  - **firstName**: String. The student's first name for verification.
  - **lastName**: String. The student's last name for verification.
  - **dateOfBirth**: Date. The student's date of birth for verification.
- **Output Messages:**
  - **status**: String. The current status of the application.
  - **requiredActions**: String. Any required actions or additional information needed from the student.
  - **nextUpdate**: String. The estimated time until the next status update.



Check Credit Points WSDL

```
<definitions name="CheckCreditPointsService"
        targetNamespace="http://example.com/CheckCreditPointsService"
        xmlns="http://schemas.xmlsoap.org/wsdl/"
        xmlns:tns="http://example.com/CheckCreditPointsService"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">

 <!-- Types Definition -->
 <types>
  <xsd:schema targetNamespace="http://example.com/CheckCreditPointsService">
   <xsd:complexType name="CreditPointsInfo">
    <xsd:sequence>
     <xsd:element name="totalCreditPoints" type="xsd:int" />
     <xsd:element name="requiredCreditPoints" type="xsd:int" />
     <xsd:element name="remainingCreditPoints" type="xsd:int" />
     <xsd:element name="status" type="xsd:string" />
    </xsd:sequence>
   </xsd:complexType>
  </xsd:schema>
 </types>

 <!-- CheckCreditPoints Operation -->
 <message name="CheckCreditPointsRequest">
  <part name="studentID" type="xsd:string" />
  <part name="currentProgram" type="xsd:string" />
  <part name="academicHistory" type="xsd:base64Binary" />
 </message>

 <message name="CheckCreditPointsSolicit">
  <part name="prompt" type="xsd:string" />
 </message>

 <message name="CheckCreditPointsResponse">
  <part name="creditPointsInfo" type="tns:CreditPointsInfo" />
 </message>

 <portType name="CheckCreditPointsPortType">
  <operation name="CheckCreditPoints">
   <output message="tns:CheckCreditPointsSolicit" />
   <input message="tns:CheckCreditPointsResponse" />
  </operation>
 </portType>

 <binding name="CheckCreditPointsBinding" type="tns:CheckCreditPointsPortType">
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" />
  <operation name="CheckCreditPoints">
   <soap:operation soapAction="http://example.com/CheckCreditPointsService/CheckCreditPoints" />
   <output>
    <soap:body use="literal" />
   </output>
   <input>
    <soap:body use="literal" />
   </input>
  </operation>
 </binding>

 <service name="CheckCreditPointsService">
  <documentation>Check Credit Points Service</documentation>
  <port name="CheckCreditPointsPort" binding="tns:CheckCreditPointsBinding">
   <soap:address location="http://example.com/CheckCreditPointsService" />
  </port>
 </service>

</definitions>
```

Figure 3 Check Credit Points WSDL



Check Application Status WSDL

```
<definitions name="CheckApplicationStatusService"
        targetNamespace="http://example.com/CheckApplicationStatusService"
        xmlns="http://schemas.xmlsoap.org/wsdl/"
        xmlns:tns="http://example.com/CheckApplicationStatusService"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">

 <!-- Types Definition -->
 <types>
  <xsd:schema targetNamespace="http://example.com/CheckApplicationStatusService">
   <xsd:complexType name="ApplicationStatusInfo">
    <xsd:sequence>
     <xsd:element name="status" type="xsd:string" />
     <xsd:element name="requiredActions" type="xsd:string" />
     <xsd:element name="nextUpdate" type="xsd:string" />
    </xsd:sequence>
   </xsd:complexType>
  </xsd:schema>
 </types>

 <!-- CheckApplicationStatus Operation -->
 <message name="CheckApplicationStatusRequest">
  <part name="applicationID" type="xsd:string" />
  <part name="firstName" type="xsd:string" />
  <part name="lastName" type="xsd:string" />
  <part name="dateOfBirth" type="xsd:date" />
 </message>

 <message name="CheckApplicationStatusSolicit">
  <part name="prompt" type="xsd:string" />
 </message>

 <message name="CheckApplicationStatusResponse">
  <part name="applicationStatusInfo" type="tns:ApplicationStatusInfo" />
 </message>

 <portType name="CheckApplicationStatusPortType">
  <operation name="CheckApplicationStatus">
   <output message="tns:CheckApplicationStatusSolicit" />
   <input message="tns:CheckApplicationStatusResponse" />
  </operation>
 </portType>

 <binding name="CheckApplicationStatusBinding" type="tns:CheckApplicationStatusPortType">
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" />
  <operation name="CheckApplicationStatus">
   <soap:operation
soapAction="http://example.com/CheckApplicationStatusService/CheckApplicationStatus" />
   <output>
    <soap:body use="literal" />
   </output>
   <input>
    <soap:body use="literal" />
   </input>
  </operation>
 </binding>

 <service name="CheckApplicationStatusService">
  <documentation>Check Application Status Service</documentation>
  <port name="CheckApplicationStatusPort" binding="tns:CheckApplicationStatusBinding">
   <soap:address location="http://example.com/CheckApplicationStatusService" />
  </port>
 </service>

</definitions>
```

Figure 4 Check Application Status WSDL

WSDL Description

In my WSDL document, I have meticulously defined the Study Abroad Application (SAA) service, utilizing the Web Services Description Language (WSDL) to meticulously map out the structure and functionalities of the web service. Through abstract and concrete definitions, my WSDL document thoroughly illustrates the service's operations, messages, and data types, thereby facilitating a seamless and comprehensible interaction for clients with the service.

The WSDL's <types>, <message>, <operation>, and <portType> components all have abstract definitions. The data types used by the service, like the AddressType complex type that represents a student's address, are defined in the <types> element. The data transferred between the client and the service is defined by the <message> element, which also specifies the input and output messages for each action. The activities carried out by the service are described in the <operation> element, with each reference to the input and output messages that were previously defined. A concise and abstract explanation of the service's capabilities may be found in the <portType> element, which is a collection of operations that define the service's interface.

However, the WSDL contains concrete definitions for the <binding>, <port>, and <service> elements. The protocol and data format for each operation are defined by the <binding> element, which also defines how the abstract operations will be utilised in practise. The location where the service is accessible is determined by the <port> element, which binds a particular network address or endpoint to a binding. Everything is combined in the <service> element, which also provides a full description of the web service and links a port to it.

Furthermore, the <portType> parts for the CheckCreditPoints and CheckApplicationStatus actions clearly show that I have made my WSDL compatible with the solicit-response operation paradigm. The service may now proactively request certain information or activities from the client thanks to this design decision, which improves the application process's user-friendliness and engagement for students hoping to study abroad.

TASK 3 XML Schema Design

<u>XML Schema for CheckCreditPoints Function Input/Ouput Messages</u>

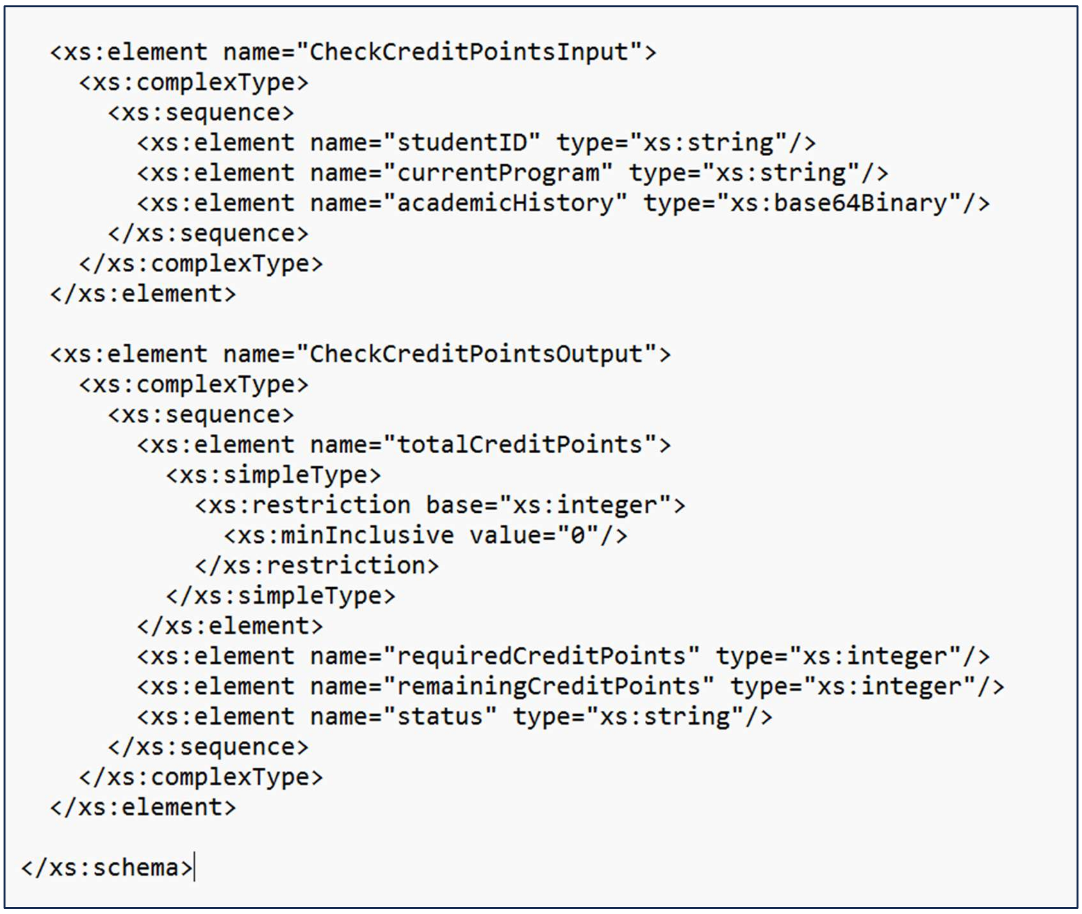
Function Name: CheckCreditPoints

Input Messages:

studentID: String. The student's unique identification number.
currentProgram: String. The program in which the student is currently enrolled.
academicHistory: File. Transcripts and records of the student's academic history.

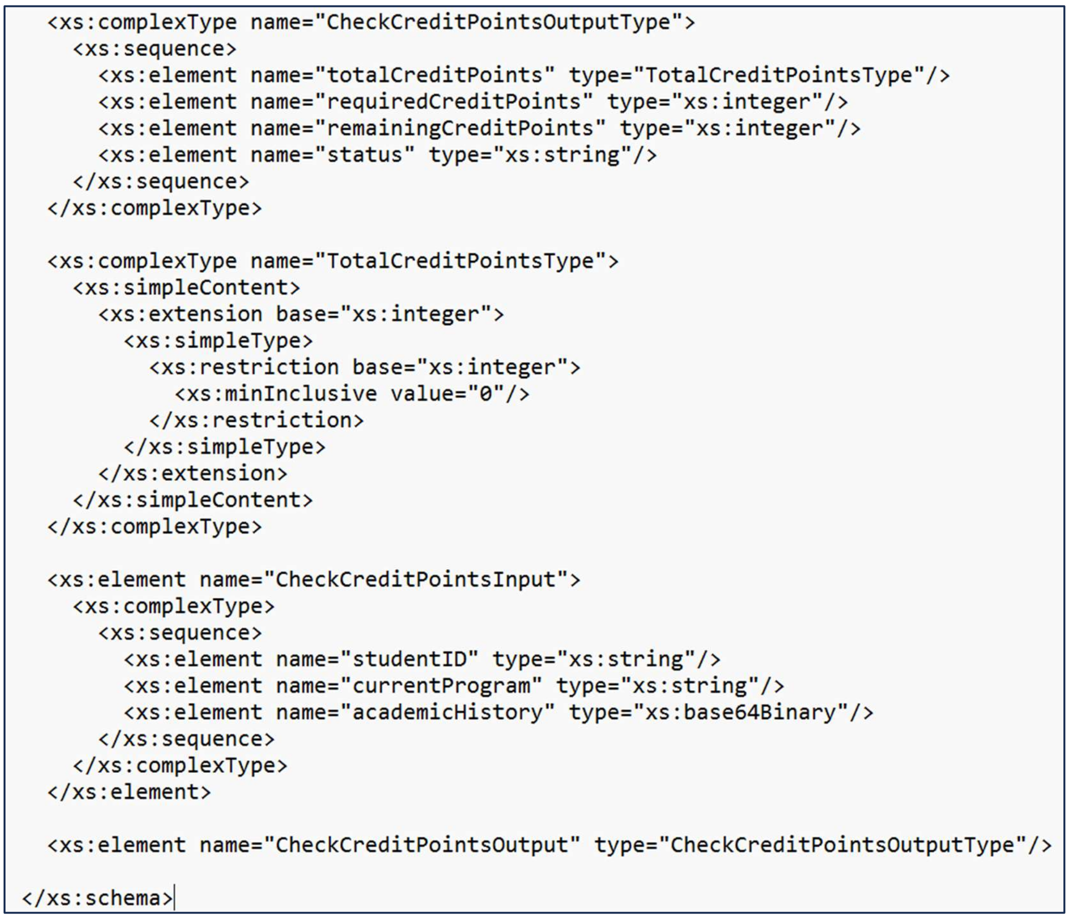
Output Messages:

totalCreditPoints: Integer. The total number of credit points the student has earned.
requiredCreditPoints: Integer. The total number of credit points required for the student's current program.
remainingCreditPoints: Integer. The number of credit points the student needs to complete their program.
status: String. A message indicating whether the student meets the credit points requirements for their program.



```
  <xs:element name="CheckCreditPointsInput">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="studentID" type="xs:string"/>
        <xs:element name="currentProgram" type="xs:string"/>
        <xs:element name="academicHistory" type="xs:base64Binary"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="CheckCreditPointsOutput">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="totalCreditPoints">
          <xs:simpleType>
            <xs:restriction base="xs:integer">
              <xs:minInclusive value="0"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
        <xs:element name="requiredCreditPoints" type="xs:integer"/>
        <xs:element name="remainingCreditPoints" type="xs:integer"/>
        <xs:element name="status" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

Figure 5 XML Schema for Check Credit Points Function Input/Output Messages



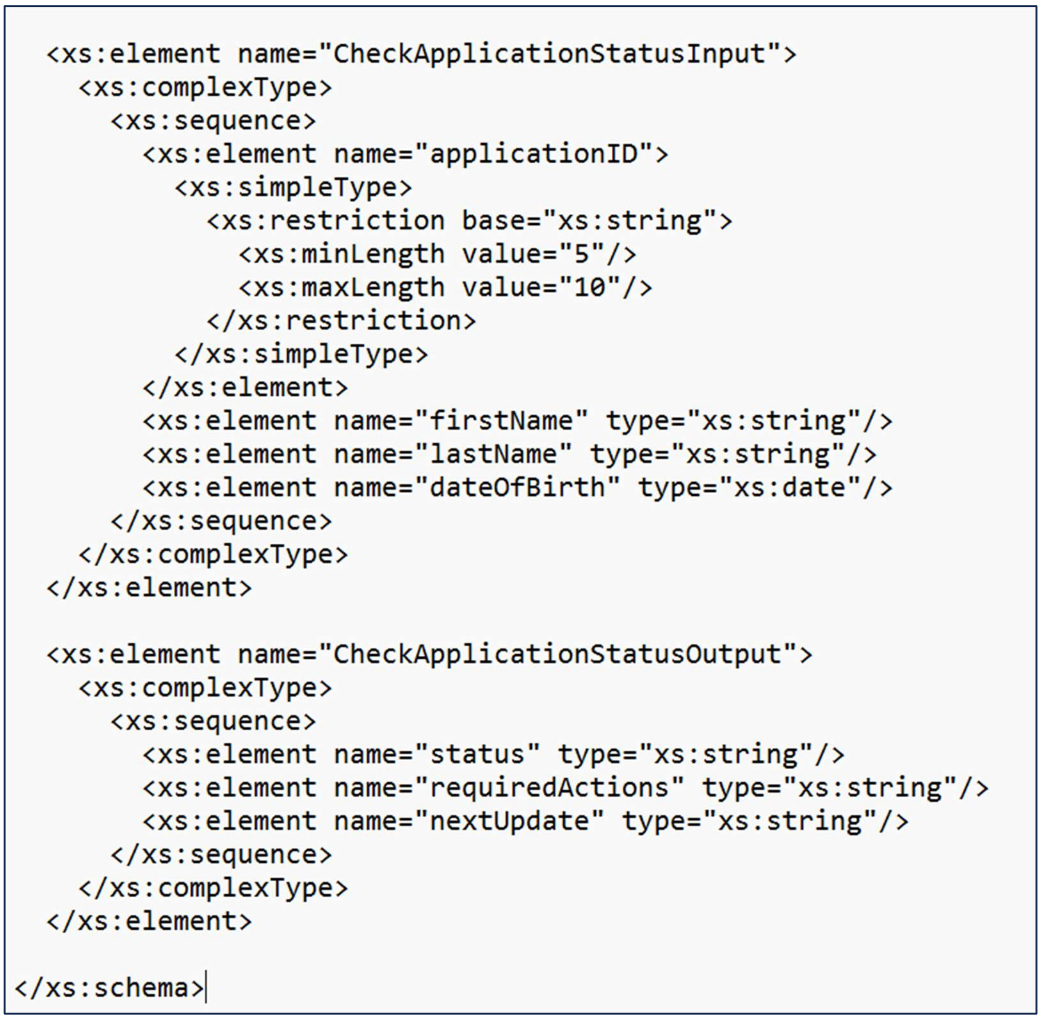
```
  <xs:complexType name="CheckCreditPointsOutputType">
    <xs:sequence>
      <xs:element name="totalCreditPoints" type="TotalCreditPointsType"/>
      <xs:element name="requiredCreditPoints" type="xs:integer"/>
      <xs:element name="remainingCreditPoints" type="xs:integer"/>
      <xs:element name="status" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="TotalCreditPointsType">
    <xs:simpleContent>
      <xs:extension base="xs:integer">
        <xs:simpleType>
          <xs:restriction base="xs:integer">
            <xs:minInclusive value="0"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:element name="CheckCreditPointsInput">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="studentID" type="xs:string"/>
        <xs:element name="currentProgram" type="xs:string"/>
        <xs:element name="academicHistory" type="xs:base64Binary"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="CheckCreditPointsOutput" type="CheckCreditPointsOutputType"/>

</xs:schema>
```

In the first Xml schema above the CheckCreditPointInput element represents the input message and includes three child elements: studentID, current program and academic history. There is also a constraint on totalCreditPoints element where the minimum value can be ‘0’. The checkCreditPointsOutput element represents the output message along with its child elements.

In the second XMl schema above the complex type is defined separately and then reused as the type of the "CheckCreditPointsOutput" element.

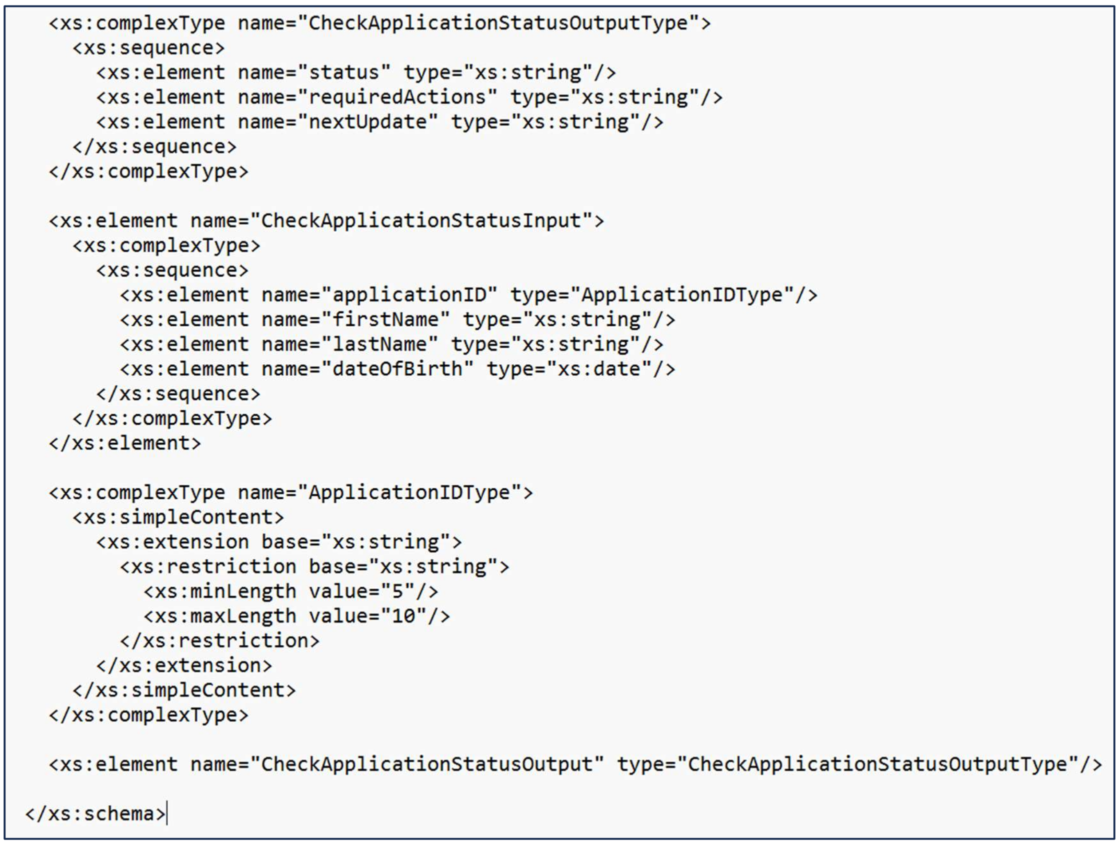
XML Schema for CheckCreditPoints Function Input/Ouput Messages

Function Name: CheckApplicationStatus
Input Messages:
applicationID: String. The unique identifier for the submitted application.
firstName: String. The student's first name for verification.
lastName: String. The student's last name for verification.
dateOfBirth: Date. The student's date of birth for verification.
Output Messages:
status: String. The current status of the application.
requiredActions: String. Any required actions or additional information needed from the student.
nextUpdate: String. The estimated time until the next status update.



```
  <xs:element name="CheckApplicationStatusInput">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="applicationID">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:minLength value="5"/>
              <xs:maxLength value="10"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
        <xs:element name="firstName" type="xs:string"/>
        <xs:element name="lastName" type="xs:string"/>
        <xs:element name="dateOfBirth" type="xs:date"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="CheckApplicationStatusOutput">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="status" type="xs:string"/>
        <xs:element name="requiredActions" type="xs:string"/>
        <xs:element name="nextUpdate" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

Figure 6 Figure 5 XML Schema for Check Application Status Function Input/Output Messages

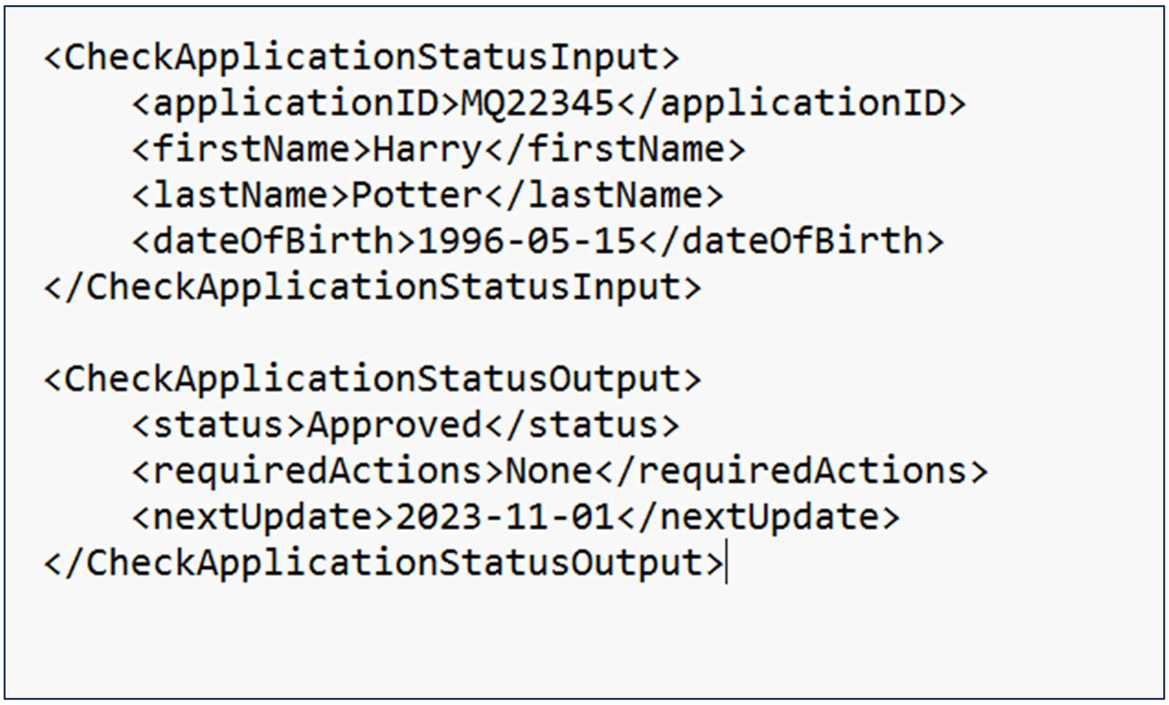


```
  <xs:complexType name="CheckApplicationStatusOutputType">
    <xs:sequence>
      <xs:element name="status" type="xs:string"/>
      <xs:element name="requiredActions" type="xs:string"/>
      <xs:element name="nextUpdate" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>

  <xs:element name="CheckApplicationStatusInput">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="applicationID" type="ApplicationIDType"/>
        <xs:element name="firstName" type="xs:string"/>
        <xs:element name="lastName" type="xs:string"/>
        <xs:element name="dateOfBirth" type="xs:date"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:complexType name="ApplicationIDType">
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:restriction base="xs:string">
          <xs:minLength value="5"/>
          <xs:maxLength value="10"/>
        </xs:restriction>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:element name="CheckApplicationStatusOutput" type="CheckApplicationStatusOutputType"/>

</xs:schema>
```

In the First XMl schema above CheckApplicationStatusInput is defined with a complex type including its child elements: applicationID, first name, last name and date of birth. A constraint is added to application ID where minimum value can be 5 integers and maximum value can be 10 integers. The CheckApplicationStatus output element represents the output message and is defined as a complex type within the element.

In the second XML schema the complex type is defined separately and reused as the type of the "CheckApplicationStatusOutput" element.

<u>XML Instance for CheckApplicationStatus Function Input/Ouput Messages</u>



```xml
<CheckApplicationStatusInput>
    <applicationID>MQ22345</applicationID>
    <firstName>Harry</firstName>
    <lastName>Potter</lastName>
    <dateOfBirth>1996-05-15</dateOfBirth>
</CheckApplicationStatusInput>

<CheckApplicationStatusOutput>
    <status>Approved</status>
    <requiredActions>None</requiredActions>
    <nextUpdate>2023-11-01</nextUpdate>
</CheckApplicationStatusOutput>
```

Figure 7 XML Instance

TASK 4 Discussion

- **How does robotic process automation (RPA) assist in the entire process?**

Robotic Process Automation (RPA) is the process of automating repetitive and routine tasks using software robots (Casey, 2020). When referring to RPA for the Study Abroad Applications (SAA), consider tasks within the process that are repetitive and routine, this could be checking for prerequisites for courses, verifying eligibility for each student or sending certain emails such as acknowledgement emails. Instead of having an employee manually complete all these repetitive tasks they can be automated using RPA. The use of RPA can help increase efficiency, reduce human errors, have significantly faster response time, and save money (Madakam et al., 2019).

A useful application of RPA within the SAA process could be data transfer and synchronisation. With there being so many different partnered universities around the world that students are applying for, it is almost guaranteed that there will be many different databases and systems used. This will create repetitive and time-consuming tasks in transferring each student's data from one university to another, this is where the implementation of an RPA would be massively beneficial. An example of how this RPA would work could be, that once a student has been accepted into a partner university, the RPA software robot can collect that student data from Macquarie's eStudent system and automatically input it into the partner university system.

Another use case for RPA within the SAA process is within the notification systems. Software robots can be used to automatically send notifications to students regarding their application status, course enrolments, special considerations, and any other routine email they might receive throughout the SAA process.
These use cases of RPA in the context of the study abroad process would be very beneficial in increasing efficiency throughout the process and having significantly faster response times with fewer errors (Olavsrud & Boulton, 2022).

- **The possible technologies potentially used to support system integration at various levels.**

There are several different technologies that could be used to support system integration at various levels, these include middleware solutions, Application programming interfaces (APIs) and Web Services (Editor, 2021).

Middleware Solutions is a software that is used to connect different software applications or systems together (Raychoudhury et al., 2013). In the context of the SAA process, middleware solutions could allow for the student portal to connect with the university databases or other platforms the university might use. Middleware helps to ensure that data is successfully transferred and at the same time is translated into the correct format for the receiving system to be able to understand.

APIs are responsible for communication between two different systems (Lutkevich & Nolle, 2022). For the SAA process, an API could be useful to get course information from one of the partnered universities or communicate with Macquarie's eStudent system to view a student's academic records.

Web Services are an interaction between two devices over a network, some examples of web services that are commonly used are XML-RPC (Remote Procedure Call) and SOAP (Simple Object Access Protocol) (Walker, 2023). Web services could be useful when looking at the SAA process by using it for checking student details or viewing course availability in real-time (Lewis, 2021).

These technologies could be vital in the integration support for the system by opening many different possibilities to create a more efficient and reliable system that allows students and staff to have more ease of access to information and resources (Leonard-Barton & A. Kraus, 2014).

- **The options and considerations of the technologies along with their advantages and disadvantages.**

There are many considerations that need to be made before any of these technologies can be used with the SAA process as each one of them could have great benefits but also has potential downsides (da Silveira et al., 2021).

Middleware Solutions:

- Advantages: Efficiently able to connect different applications together and can handle large amounts of data transfer which would be very useful in the SAA process are there are a large number of students that need their data transferred from one university's system to another (Raychoudhury et al., 2013). Middleware is also flexible as it is between both systems, this means that if one system goes down it will be able to support the other system till both are running again.
- Disadvantages: Middle solutions can be difficult and complex technology to set up and can potentially cause latency in data transfer. Another downside to middleware is compatibility issues as both systems need to be set up in a similar way.

APIs:

- Advantages: Allows for real-time data exchange, is the most common method of communication within the system which can be very useful in the set-up process, APIs are also flexible and secure (Lutkevich & Nolle, 2022).
- Disadvantages: Both systems are required to be API-compatible and rely on external systems to function.

Web Services:

- Advantages: Web services are both versatile and scalable which is very important for a system that is constantly growing such as the SAA process which is continually having more students applying. Web services are also always available from anywhere with an internet connection, which can be potentially a big asset in saving money in the long run (Lewis, 2021).
- Disadvantages: Needs a reliable internet connection to work, potential security issues as it can be accessed from anywhere and can be slower than when using APIs.

The Integration of different systems across different universities requires well-thought-out use of technologies and strategies. RPAs can help with the efficiency and automation of routine tasks, while system integration technologies help to ensure that different systems can communicate with each other. The choice of technology to use for system integration should be based on the specific requirements that a system needs, existing infrastructure, and what level of integration and automation is wanted within the system (Liu et al., 2009).

References

1. Anagnoste, S. (1970) *Setting up a Robotic Process Automation Center of Excellence*, *Management Dynamics in the Knowledge Economy*. Available at: https://www.ceeol.com/search/article-detail?id=767450 (Accessed: 28 October 2023).
2. Casey, K. (2020) *How to explain robotic process automation (RPA) in plain english*, *The Enterprisers Project*. Available at: https://enterprisersproject.com/article/2019/5/rpa-robotic-process-automation-how-explain (Accessed: 25 October 2023).
3. da Silveira, F., Lermen, F.H. and Amaral, F.G. (2021) 'An overview of Agriculture 4.0 development: Systematic review of descriptions, technologies, barriers, advantages, and disadvantages', *Computers and Electronics in Agriculture*, 189, p. 106405. doi:10.1016/j.compag.2021.106405.
4. Editor (2021) *System integration: Types, approaches, and implementation steps*, *AltexSoft*. Available at: https://www.altexsoft.com/blog/system-integration/ (Accessed: 26 October 2023).
5. Hofmann, P., Samp, C. and Urbach, N. (2019) 'Robotic Process Automation', *Electronic Markets*, 30(1), pp. 99–106. doi:10.1007/s12525-019-00365-8.
6. Leonard-Barton, D. and A. Kraus, W. (2014) *Implementing New Technology*, *Harvard Business Review*. Available at: https://hbr.org/1985/11/implementing-new-technology (Accessed: 28 October 2023).
7. Lewis, S. (2021) *What are web services? definition from searchapparchitecture*, *App Architecture*. Available at: https://www.techtarget.com/searchapparchitecture/definition/Web-services (Accessed: 28 October 2023).
8. Liu, S. *et al.* (2009) 'Integration of decision support systems to improve decision support performance', *Knowledge and Information Systems*, 22(3), pp. 261–286. doi:10.1007/s10115-009-0192-4.
9. Lutkevich, B. and Nolle, T. (2022) *What is an API (application programming interface)?*, *App Architecture*. Available at: https://www.techtarget.com/searchapparchitecture/definition/application-program-interface-API (Accessed: 28 October 2023).
10. Madakam, S., Holmukhe, R.M. and Kumar Jaiswal, D. (2019) 'The Future Digital Work Force: Robotic Process Automation (RPA)', *Journal of Information Systems and Technology Management*, 16, pp. 1–17. doi:10.4301/s1807-1775201916001.
11. O'Shaughnessy, S.A. *et al.* (1970) *Identifying advantages and disadvantages of Variable Rate Irrigation: An updated review*, *Applied Engineering in Agriculture*. Available at: https://elibrary.asabe.org/abstract.asp?aid=51008 (Accessed: 28 October 2023).
12. Olavsrud, T. and Boulton, C. (2022) *What is RPA? A revolution in business process automation*, *CIO*. Available at: https://www.cio.com/article/227908/what-is-rpa-robotic-process-automation-explained.html (Accessed: 25 October 2023).

13. Partner, F.M.D. (2023) *System integration: Types, methods, and approaches - A complete guide*, *Folio3 Dynamics Blog*. Available at: https://dynamics.folio3.com/blog/system-integration/ (Accessed: 26 October 2023).
14. Pellegrino, J.W. and Quellmalz, E.S. (2014) 'Perspectives on the integration of technology and assessment', *Journal of Research on Technology in Education*, 43(2), pp. 119–134. doi:10.1080/15391523.2010.10782565.
15. Raychoudhury, V. *et al.* (2013) 'Middleware for Pervasive Computing: A survey', *Pervasive and Mobile Computing*, 9(2), pp. 177–200. doi:10.1016/j.pmcj.2012.08.006.
16. Walker, A. (2023) *What are web services? architecture, types, example*, *Guru99*. Available at: https://www.guru99.com/web-service-architecture.html (Accessed: 26 October 2023).