**Enrolment System - Report Template**

|  |  |
| --- | --- |
| Student ID | Name |
| 001175001 | Jesse Hamilton-Young |

**Part I - Gathering and Documenting Requirements**

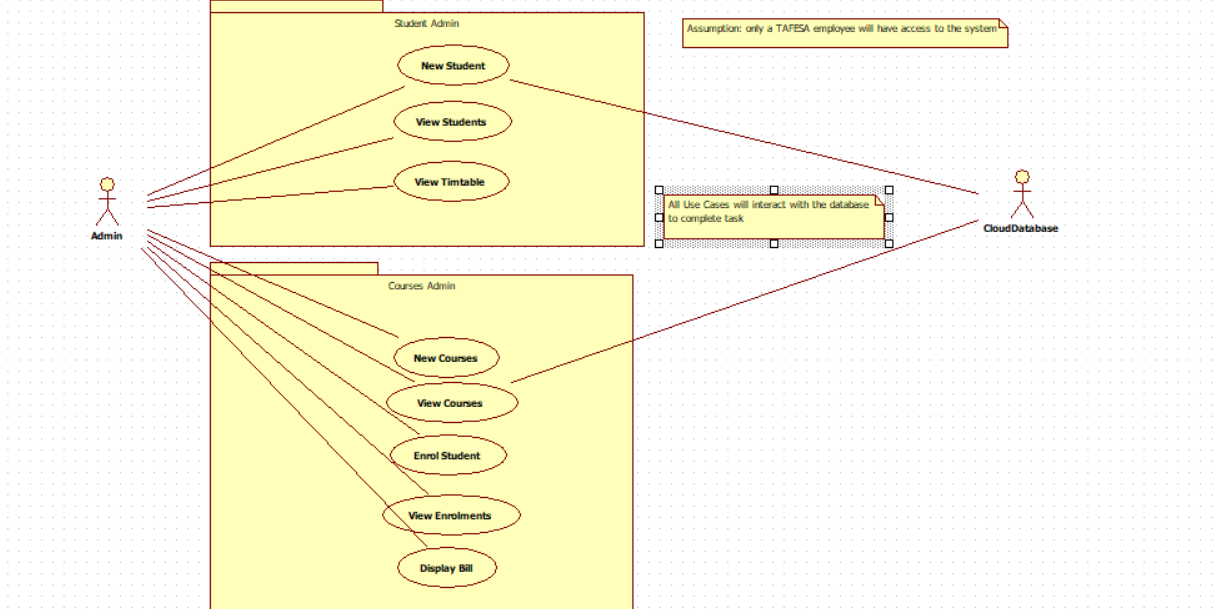
**Technical requirements:**

|  |  |
| --- | --- |
| **Requirement ID** | **Description** |
| ***RE01*** | ***Legacy UI must be maintained*** |
| RE02 | Database structure is to remain the same consisting of three tables Enrolment, Student and Course |
| RE03 | Must be a cloud multi-user application |
| RE04 | Must enable a single user to administer both Student and Courses |
| RE05 | Must be accessible by different users at various locations |

**Technical Considerations:**

|  |  |
| --- | --- |
| **Considerations** | **Response** |
| 1. Hosting | We will be using a cloud hosted solution:   * Highly scalable. * Maintenance and integration are as simple as subscribing to new features as they are released. * Cost effective - only way for storage and resources that you need when you need them. |
| 1. Cloud Model | SaaS is the most appropriate choice for the enrollment system.   * SaaS Services and updates are maintained by the cloud service provider. * SaaS services are High availability – cloud service providers tend to have many backups and redundancies limiting possible downtime. * SaaS services have a high level of security that is managed and maintained by the service provider. |
| 1. Service Provider (Max 100 words) | Azure:  As we are using the Microsoft eco system (C#, .NET, Visual Studio etc.) Azure cloud services are the obvious choice as the integration between services, language and IDE are almost seamless. This allows us to spend more time solving issues within our solution and help keep labour costs down through less messing around with connecting various services to our chosen IDE. |
| 1. Database (Max 100 words) | Azure SQL Server and SQL database would be the obvious choice for a database solution. These are fully managed services by Microsoft which allows for high availability, scalability, and security without the need for physical hardware. It has high compatibility with most SQL server features allowing for a smooth transition to a multiuser database solution.it is a modern and efficient replacement for our outdated SQL server solution. |
| 1. IDE | Visual Studio 2019:   * Azure DevOps Services – allowing us to publish our programs to various Azure services with ease. * Azure SDKS – allowing integration between Azure services and our application. |
| 1. Architecture and Framework | The cloud architecture will focus on the backend sever side handling of the data required to service the local frontend application the framework to be used is the .NET C# framework. |
| 1. Uptime strategy (Max 100 words) | Due to Azures SLA, Microsoft guarantees an uptime of 99.999% which leaves us with a potential downtime of 6 seconds per week, 25.9 seconds per month and 5.26 minutes per year. This is a high availability managed by Azure and requires no additional strategies on our part. |

**Use Case Model (Add your model here)**



**Classes and relationship (Add class diagrams here)**

A screenshot of a computer

Description automatically generated

**List of services to be implemented**

|  |
| --- |
| **Student Admin - New Students** |
| **Student Admin – view Courses with student enrolled** |
| **Student Admin – View Timetable** |
| **Course Admin – New Courses** |
| **Course Admin – View students enrolled in course** |
| **Course Admin – Enrol Student** |
| **Course Admin – View Grades** |
| **Course Admin – Display Bill** |

**Part III - Testing the Cloud Application**

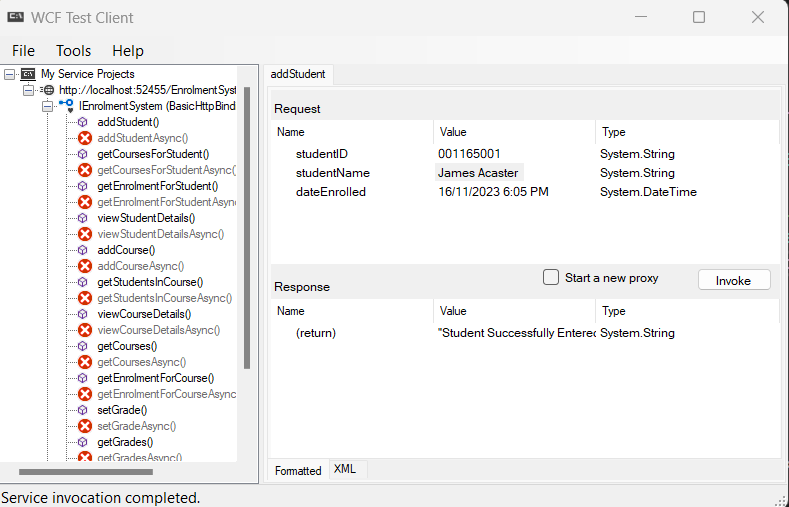
**Test Plan**

**Testing Services using WCF Client**

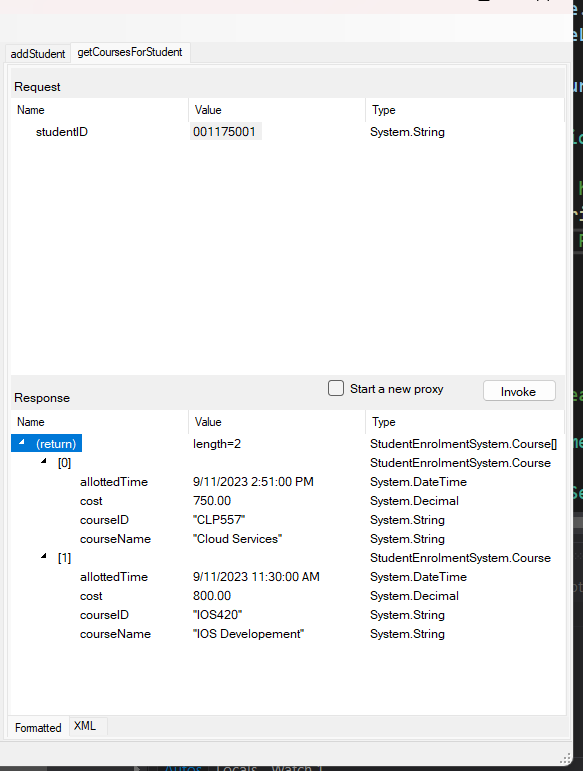
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Service** | **Input** | **Expected Output** | **Actual Output** | **Comments** |
| **1: addStudent(StudentID, StudentName, dateEnrolled)** | **StudentID: 001165001**  **StudentName: James Acaster**  **dateEnrolled: 16/11/2023 6:05pm** | **Student successfully Entered** | **Student successfully Entered** | **Works as expected** |
| **2: getCourseForStudent(StudentID)** | **StudentID: 001175001** | **A list of courses related to 001175001** | **A list of courses related to 001175001** | **Works as expected** |
| **3: getEnrolmentForStudent(StudentID)** | **StudentID: 001175001** | **A list of enrolments related to 001175001** | **A list of enrolments related to 001175001** | **Works as expected** |
| **4: viewStudentDetails(StudentID)** | **StudentID: 001175001** | **The student details related to 00175001 are returned** | **The student details related to 00175001 are returned** | **Works as expected** |
| **5: addCourse(courseID, courseName, cost, allottedTime)** | **courseID: EWD440**  **courseName: E-Commerce Website Design**  **cost: 750**  **allottedTime: 16/11/2023 6:19 PM** | **Course added Successfully** | **Course added Successfully** | **Works as expected** |
| **6: getStudentsInCourse(courseID)** | **courseID: IOS420** | **List of students related to IOS420 returned** | **List of students related to IOS420 returned** | **Works as expected** |
| **7: viewCourseDetails(courseID)** | **courseID: IOS420** | **Course details related to IOS420 returned** | **Course details related to IOS420 returned** | **Works as expected** |
| **8: getCourses()** | **null** | **A list of all courses in the database returned** | **A list of all courses in the database returned** | **Works as expected** |
| **9: getEnrolmentForCourses(courseID)** | **courseID: IOS420** | **A list of enrolments related to IOS420 returned** | **A list of enrolments related to IOS420 returned** | **Works as expected** |
| **10: setGrade(StudentID, courseID, grade)** | **StudentID: 00175001**  **courseID: CLP557**  **grade: PASS** | **The enrolment related to 001175001 and CLP557 has its grade updated to PASS** | **The enrolment related to 001175001 and CLP557 has its grade updated to PASS** | **Works as expected** |
| **11: getGrades(StudentID)** | **studentID: 00175001** | **Returns a list of strings containing pass/fail information related to 00175001** | **Returns a list of strings containing pass/fail information related to 00175001** | **Works as expected** |
| **12: enrolStudent(StudentID, courseID)** | **studentID: 001165001**  **courseID: IOS420** | **The enrolment information is added to the database** | **The enrolment information is added to the database** | **Works as expected** |
| **13: getStudents()** | **null** | **Returns a list of all students in the database** | **Returns a list of all students in the database** | **Works as expected** |

**Screenshots:**

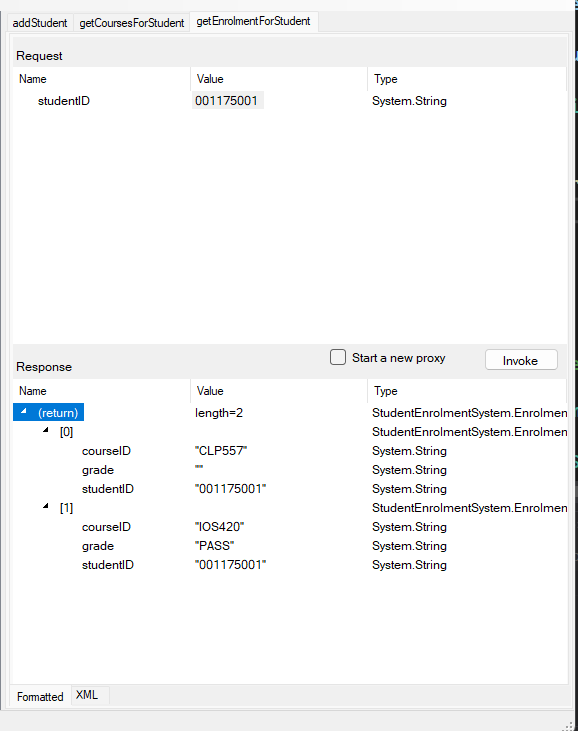
**1:**

****

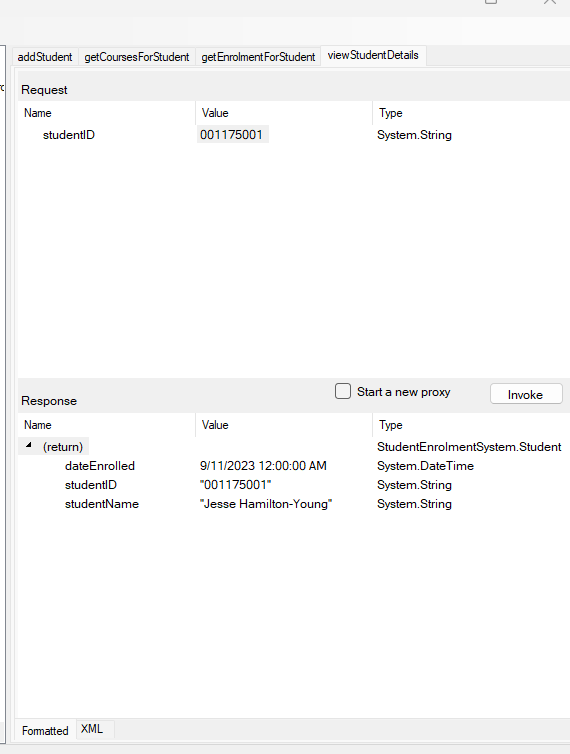
**2:**

****

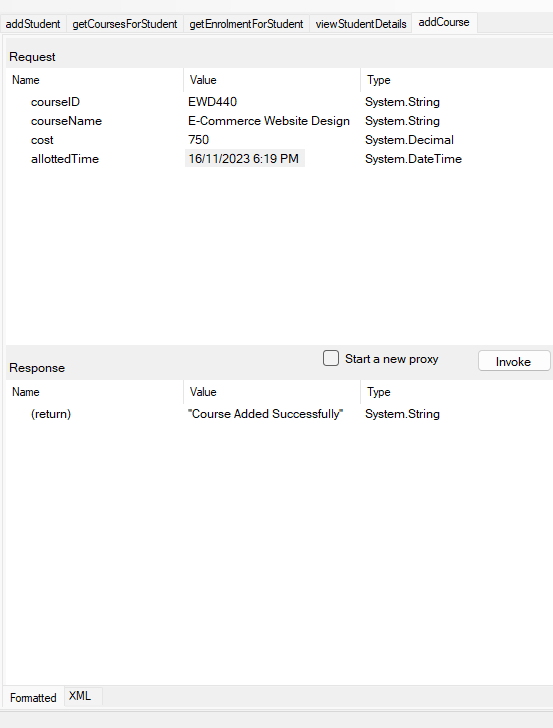
**3:**

****

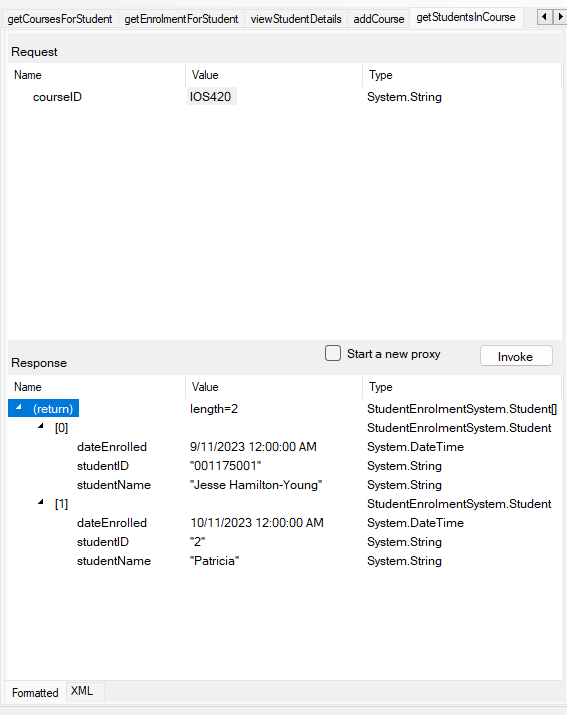
**4:**

****

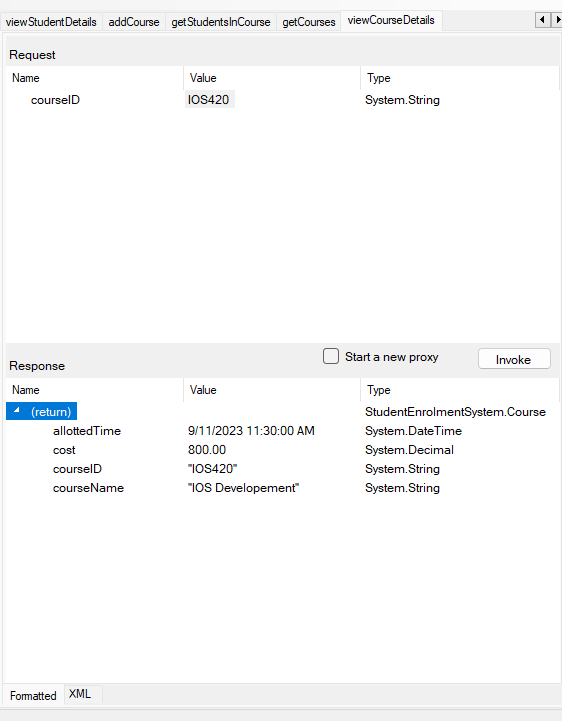
**5:**

****

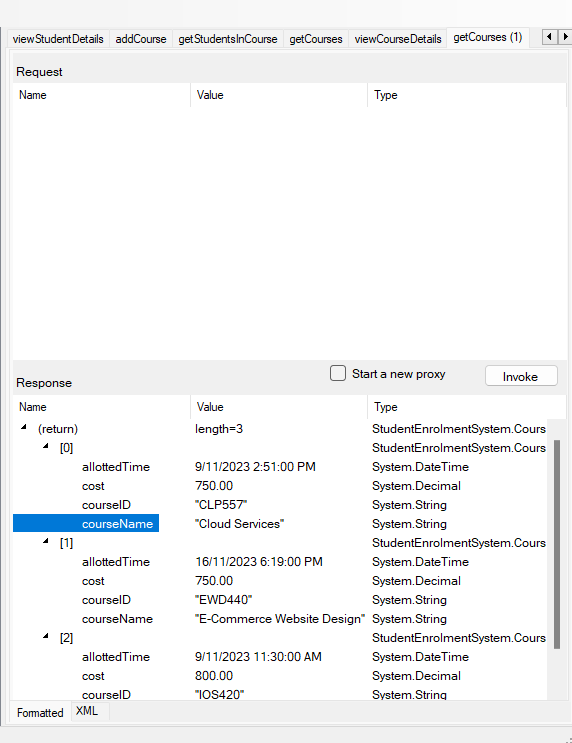
**6:**

****

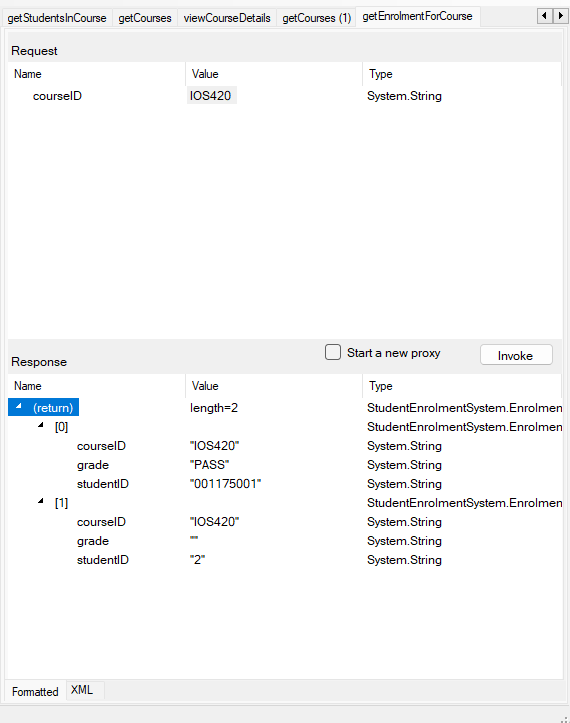
**7:**

****

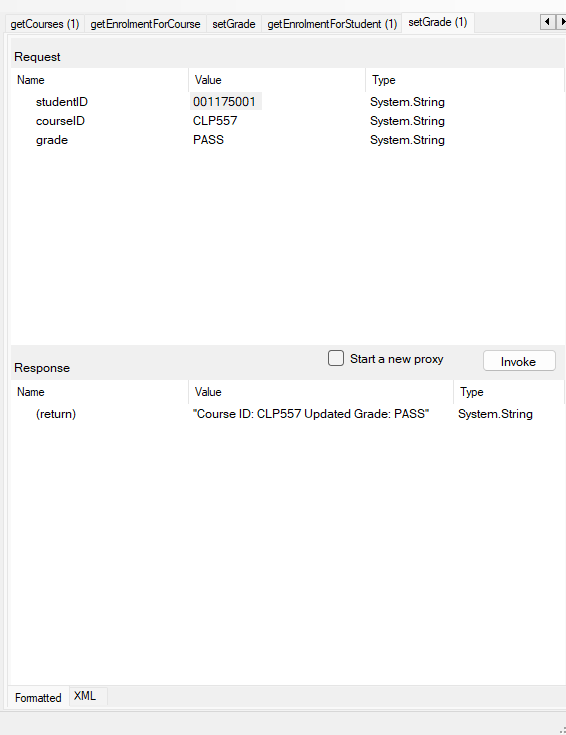
**8:**

****

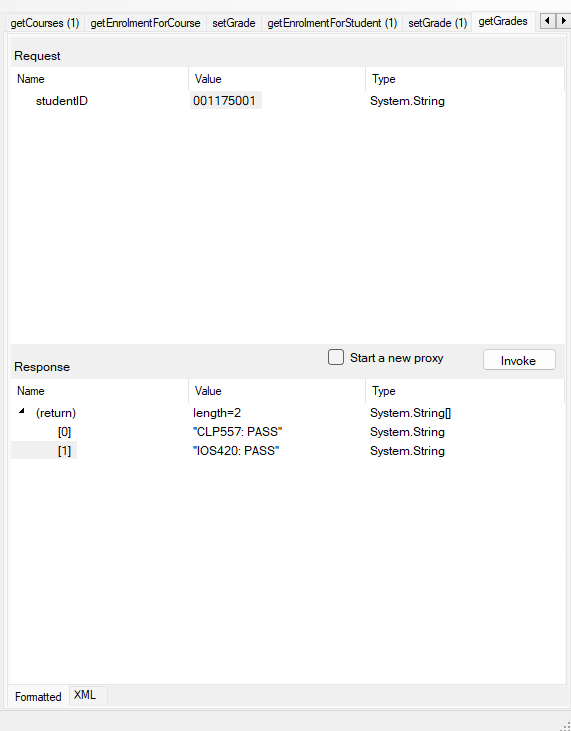
**9:**

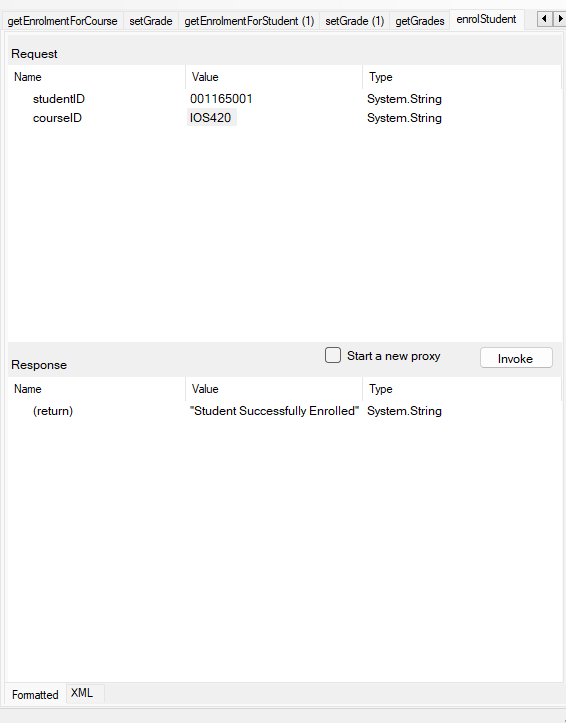
****

**10:**

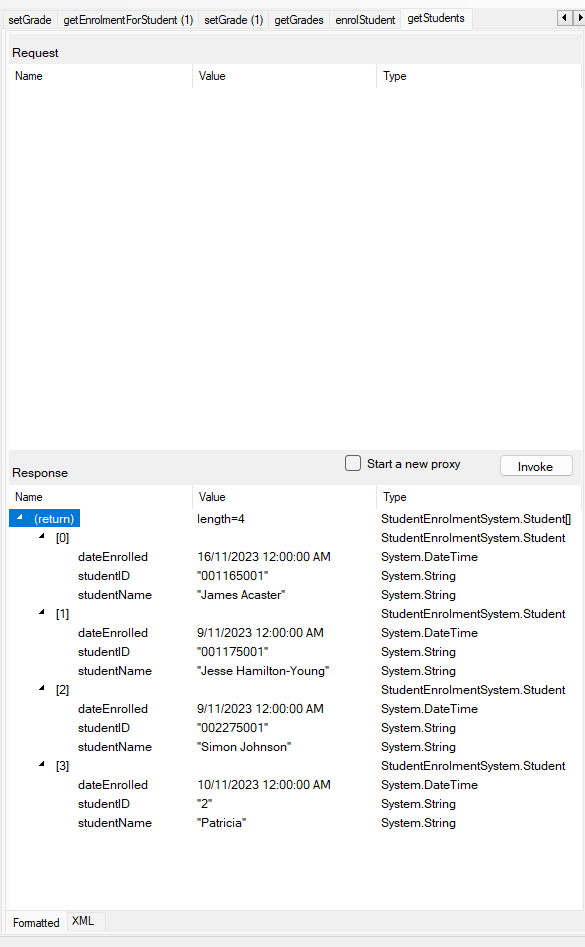
****

**11:**

**  
12:**

****

**13:**

****

**Testing Services using Windows Client**

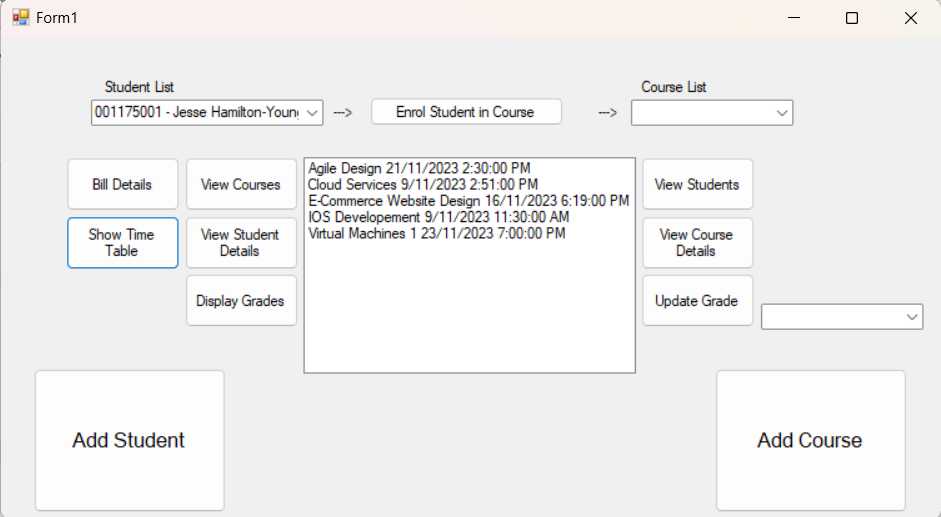
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Service** | **Input** | **Expected Output** | **Actual Output** | **Comments** |
| 1. **Student Admin - New Students** | **StudentID:**  **123456789**  **Student Name:**  **TestStudent1** | **The test student is added successfully** |  |  |
| 1. **Admin – view courses student is enrolled in** | **Select a student from dropdown and**  **View Courses button is pressed** | **Courses the student is enrolled in are displayed** |  |  |
| 1. **Student Admin – View Timetable** | **Select a student from dropdown and view timetable button is pressed** | **The allotted times for all enrolled courses for that student is shown** |  |  |
| 1. **Course Admin – New Courses** | **Course ID: NEW156**  **Course Name:**  **NewTestCourse**  **Cost: 200.00**  **Date: 23/11/2023 2:30PM** | **Course is successfully added to the DB** |  |  |
| 1. **Course Admin – View Students Enrolled in course** | **Select a course from dropdown and**  **View students button is pressed** | **Students enrolled in the course are displayed** |  |  |
| 1. **Course Admin – Enrol Student** | **Student selected from the drop down and course selected from the drop-down enrolment button is pressed** | **Student Enrolment is added to the DB** |  |  |
| 1. **student Admin – View Grades** | **Select a student from dropdown and view Grades button is pressed** | **Courses and their corresponding grades are shown for the student selected** |  |  |
| 1. **Student Admin – Display Bill** | **Select a student from the dropdown**  **Display bill button pressed** | **The courses and corresponding costs are displayed for the selected student as well as a total payable** |  |  |
| 1. **Student Admin Update Grade** | **Student selected from the drop down and course selected from the drop-down**  **A grade is selected from the dropdown update Grade button is pressed** | **The grade for the selected student and course is updated to the grade selected** |  |  |

A screenshot of a computer

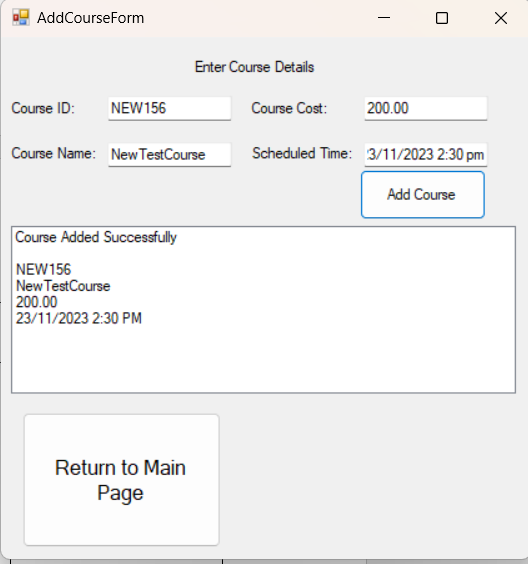
Description automatically generated

**A screenshot of a computer

Description automatically generated**

****



****

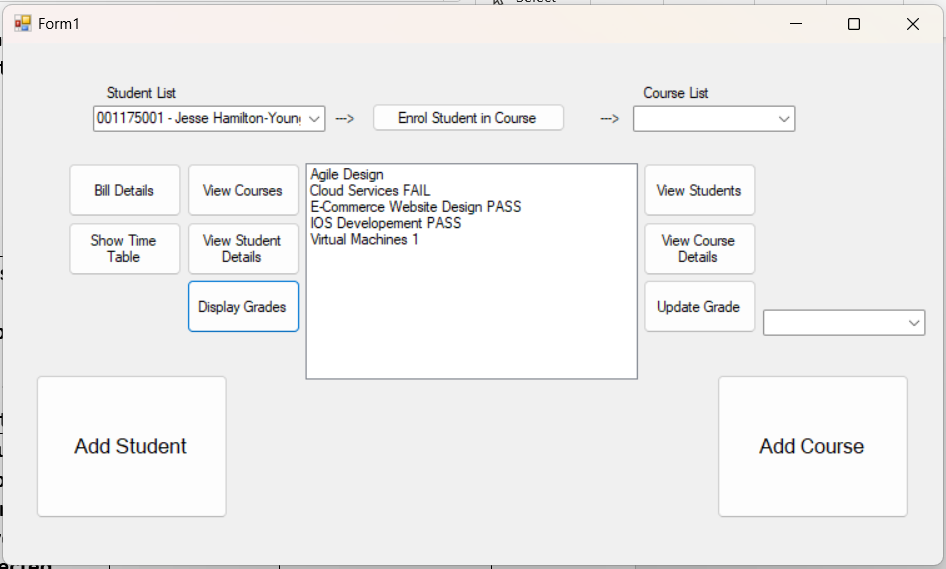
A screenshot of a computer

Description automatically generated

**A screenshot of a computer

Description automatically generated**A screenshot of a computer

Description automatically generated

****

**A screenshot of a computer

Description automatically generated**

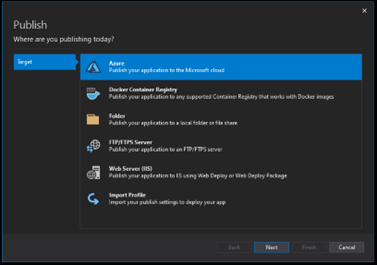
**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Part IV - Deploying /Publishing Services to Azure**

****

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

|  |  |
| --- | --- |
| **URL** | **https://studentenrolmentsystem20231110133131.azurewebsites.net/** |