International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 13, Number 7 (2020), pp. 1795-1801

Secure and Robust Web Services for E-Payment of Tuition Fees

Feras H. Al-Hawari[1] and Mohammad S. Habahbeh[2]

[1] *Associate Professor, School of Electrical Engineering and IT, German Jordanian University, Jordan.*
[2] *Senior Java EE Programmer, Information Systems and Technology Center, German Jordanian University, Jordan.*

[1]*ORCID: 0000-0001-6948-3336*

Abstract

This paper proposes an e-payment processing architecture to permit students to transfer tuition fees payments to their university portal accounts using different payment forms such as e-banking, credit cards, and debit cards. The architecture is based on RESTful web services that can exchange financial transactions with all Jordanian banks via the eFAWATEERcom platform to process online tuition payments. Besides, various essential aspects to secure the proposed web services such as server security, network security, and data-integrity protection are addressed in this work to gain the trust of customers. Further, the web services are designed to be robust so they can appropriately handle input errors and software exceptions. In that regard, the deployment of the web services resulted in a high percentage of e-payments to overall payments, which asserts the popularity of the e-payment service amongst students.

Keywords: e-payment, accounting system, eFAWATEERcom, RESTful web services, application security, application robustness.

I. INTRODUCTION

Enabling students to register for courses online is considered one of the most important e-services that every university should offer. Nevertheless, this service is not deemed fully online when students cannot pay their required registration fees electronically. Because in the previous scenario, the students must be present on campus, or in the bank, and possibly stand in long lines to pay the required fees for online registration. Therefore, an e-payment service must be implemented along with online registration to allow students to pay for, and enroll in, courses from the comfort of their homes.

Based on [1], the online payment systems are divided into account-based and electronic-currency systems. Accordingly, the account-based systems allow payment via an existing personal account (e.g., bank account). On the other hand, the electronic-currency systems permit payment when the payer owns the electronic (digital) currency. Popular forms of account-based systems, according to [1-4], are credit cards, debit cards, mediating systems (e.g., PayPal [5]), and payments via online banking. While, examples of electronic-currency systems are smart cards and online cash (e.g., bitcoin [6]) systems.

In that regard, this paper explores the required methods to integrate web services with the eFAWATEERcom platform [7, 8] to allow the German Jordanian University (GJU) students to securely and instantly transfer money to their accounts in the university portal [9, 10] for online tuition fees payment. The eFAWATEERcom platform, established in partnership with the Central Bank of Jordan, is a real-time Electronic Bill Presentment and Payment (EBPP) solution that enables students to make payments seamlessly and securely, via computer or mobile. It has already processed over 39 million invoices valued at 28 billion US dollars [7]. It also enables students to make e-payments via credit cards, debit cards, and e-banking. Not to mention, in case a student does not have a bank account, the Jordanian Banks and Post Office all accept cash payments from the students to transfer e-payments on their behalf via the eFAWATEERcom platform.

The rest of the paper is organized as follows. In section II, a comparison to related works is presented. In section III, the e-payment processing architecture that is based on web services is discussed. In section IV, the security aspects related to the web services are explored. In section V, the robustness methods employed in the web services are explained. In section VI, the deployment results of the web services are presented. In section VII, a summary of this work is given.

II. COMPARISON TO RELATED WORKS

As far as related works, the student self-service system in [11] is used to permit students to pay registration fees electronically via PayPal [5]. PayPal is a mediating service for online transactions that requires providing credit card or bank account details as the source of payments [1]. Moreover, the MUK-OFPS system in [12] is introduced to allow paying university fees online using credit cards and debit cards. Besides, the application of online payment mode in university charging [13] is achieved by connecting with a third-party payment gateway and online banking. Also, the systems discussed in [14, 15] enable students to pay fees by interacting with a bank interface. Based on the feature comparison to previous related works shown in Table 1, the eFAWATEERcom platform is advantageous as it enables students to make e-payments using credit cards, debit cards, and e-banking. It also lets students transfer online tuition payments even if they do not have bank accounts.

Although the effect of securing e-payments on improving customer satisfaction is highlighted in [16-18], the related works in [11-15] did not elaborate much on this important aspect. Hence, unlike other works, this paper also focuses on data, server, and network security aspects related to web services that enable students to make online tuition payments via the eFAWATEERcom platform. Besides, it discusses several mechanisms to make the introduced web services robust by gracefully handling input errors and program exceptions.

Table 1. A feature comparison to related works.

	Credit Cards	Debit Cards	PayPal	E-Banking	Online without a Bank Account
This Work	Yes	Yes	No	Yes	Yes
[11]	No	No	Yes	No	No
[12]	Yes	Yes	No	No	No
[13-15]	No	No	No	Yes	No

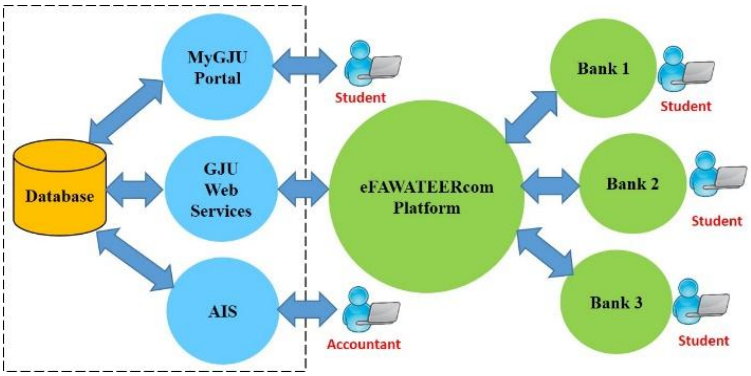


Fig. 1. The adopted e-payment processing archirecture.

III. E-PAYMENT PROCESSING ARCHITECTURE

The adopted e-payment processing architecture is shown in Fig. 1 and is based on web services. The web services exchange financial requests with banks via the eFAWATEERcom platform [7, 8] to process online tuition payments. Once an e-payment notification for a student is received, the web service will promptly record a payment transaction in the Accounting Information System (AIS) database [19]. Consequently, the payment transaction gets immediately reflected on the student's statement of account in the MyGJU portal [9]. Accordingly, the student will be able to register online [9] for as many courses as the current balance amount will cover.

The types, implementation, communication protocol, and messages of the web services that are used to process the online tuition payments at the GJU side are discussed in the following subsections.

III.I. Web Service Types

The supported web service types are:

- **StudentValidation**: This service is required to enable a bank to validate that the customer (i.e., student) has a correct MyGJU portal account and thus avoid sending the desired payment to a wrong destination. The service performs the customer validation based on a student ID included in the request message. The validation is deemed successful when a student record for the student ID is found in the MyGJU portal database. In case of success, the bank sends another request message to deposit the payment. Otherwise, the bank cancels the payment and reports the error to the customer.
- **PaymentNotification**: When the student validation succeeds, this service can be used to record a tuition payment transaction in the AIS database, given that the payment amount and student ID are both found in the request message of this service.

III.II. Web Service Implementation

The Java API for RESTful Web Services (JAX-RS) is utilized to develop the e-payment web services based on the Representational State Transfer (REST) architecture [20]. The JAX-RS uses runtime annotations to facilitate the definition and deployment of web services. Accordingly, the related Java class files can be decorated with JAX-RS annotations to define the resource classes and the actions that the clients may carry out on those resources. The annotated resources are later configured and exposed to clients when the web application archive (i.e., war file) containing the resource classes is deployed to a Java EE [21] server.

In that regard, the resource classes for the StudentValidation and PaymentNotification web services are shown in Fig. 2 and Fig. 3, respectively. The @Path annotation at line 1 in the aforementioned figures is used to define the relative Uniform Resource Identifier (URI) of the resource classes. While, the @Path annotations at line 6 in Fig. 2 and Fig. 3 specify the relative URI of the processStudentValidation method and processPaymentNotification method, respectively. The @POST annotation at line 7 in both figures indicates that the two methods process HTTP POST requests. Both methods produce content of the MIME media type APPLICATION_XML as identified by the @Produces annotation at line 8 in each figure.

Additionally, the resource class files must be packaged as part of a web application to deploy the web services. In that respect, the class shown in Fig. 4 is required to define the web service resources to be deployed. The @ApplicationPath annotation at line 1 specifies the base URI of the application servlet and all resource URIs. Whereas, the getClasses method that starts at line 3 is needed to return a list of RESTful web service resources exposed by the web application.

Consequently, a client can access a resource using a resource URI constructed as shown in Fig. 5. Where the serverName is

the DNS name mapped to the host machine. The contextPath is the name of the Java EE application. The baseURI is the value of @ApplicationPath annotation (see Fig. 4). The resourceURI is the @Path value of the resource or method (see Fig. 3).

```
1. @Path("StudentValidation")
2. public class StudentValidation {
3.   @Context
4.    private UriInfo context;

5.    public StudentValidation () { }

6.    @Path("ProcessStudentValidation")
7.    @POST
8.    @Produces(MediaType.APPLICATION_XML)
9.    public String processStudentValidation(InputStream xmlString) {
10.      // method body
       }
   }
```

Fig. 2. The StudentValidation resource class.

```
1. @Path("PaymentNotification")
2. public class PaymentNotification {
3.     @Context
4.     private UriInfo context;

5.     public PaymentNotification() { }

6.    @Path("ProcessPaymentNotification")
7.    @POST
8.    @Produces(MediaType.APPLICATION_XML)
9.    public String processPaymentNotification(InputStream xmlString) {
10.      // method body
11.   }
    }
```

Fig. 3. The PaymentNotification resource class.

```
1.  @javax.ws.rs.ApplicationPath("Services")
2.  public class ApplicationConfig extends Application {
3.      @Override
4.      public Set<Class<?>> getClasses() {
5.          Set<Class<?>> resources = new java.util.HashSet<>();
6.          resources.add(Services.StudentValidation.class);
7.          resources.add(Services.PaymentNotification.class);
8.          return resources;
        }
    }
```

Fig. 4. The ApplicationConfig class.

- **A resource URI is constructed as follows:**
https://serverName/contextPath/baseURI/resourceURI

Accordingly:
- **The URI for the *StudentValidation* resource is:**
https://epayment.gju.edu.jo/EPaySrvcs/Services/StudentValidation
- **The URI for the *processStudentValidation* method is:**
https://epayment.gju.edu.jo/EPaySrvcs/Services/StudentValidation/ProcessStudentValidation
- **The URI for the *PaymentNotification* resource is:**
https://epayment.gju.edu.jo/EPaySrvcs/Services/PaymentNotification
- **The URI for the *processPaymentNotification* method is:**
https://epayment.gju.edu.jo/EPaySrvcs/Services/PaymentNotification/ProcessPaymentNotification

Fig. 5. The web services resources and methods URIs.

III.III. Web Service Communication and Messages

The communication between clients and RESTful web services is done via HTTP requests and responses. Therefore, a client sends a request to carry out an action (i.e., invoke a method) on a resource (i.e., a class), and the web service returns a response including the produced result. Accordingly, the input and output messages of the service method are placed in the HTTP request and response bodies, respectively. Besides, the XML data format is used to represent the included input and output messages.

Each input and output message consists of a header, body, and footer sections analogous to the input message for the processStudentValidation method shown in Fig. 6. The header section contains the message timestamp, biller code (e.g., 450), and message type (i.e., STDVALREQ, STDVALRES, PMTNTFREQ, or PMTNTFRES). The body section includes the message details. The footer section encloses a signature used to ensure that the message body is not modified on the way (i.e., for data integrity protection).

The essential information in the message body for the four different message types are discussed next:

- Request message (with type STDVALREQ) for the processStudentValidation method (see Fig. 6): This message is used to validate the student (i.e., customer) status before sending the payment. Hence, it includes the needed information to do so such as the BillingNo (i.e., student ID) of the student to be validated, ServiceType (i.e., payment type), and DueAmount (i.e., payment amount).
- Response message (with type STDVALRES) for the processStudentValidation method (see Fig. 7): The Result section in this message contains the student validation result details. In case the student is successfully validated, the ErrorInfo section value will be Success. Otherwise, the validation failure reason (e.g., student Id not found, student inactive, exception encountered) will be shown in the ErrorInfo section. Besides, the student ID and payment information used to validate the student are included in this message.
- Request message (with type PMTNTFREQ) for the processPaymentNotification method (see Fig. 8): This message is sent to deposit money in the student MyGJU portal account. Therefore, it includes payment information such as eFAWATEERcom transaction Id (i.e., the value of JOEBPPSTrans), recipient (e.g., a student with BillingNo 20191209029), date, bank code, amount (e.g., 2320.000 JOD), access channel (e.g., Mobile), service type (e.g., Tuition), etc.
- Response message (with type PMTNTFRES) for the processPaymentNotification method (see Fig. 9): This message contains the payment notification Result section, eFAWATEERcom transaction Id (i.e., JOEBPPSTrans), processing date, and statement date.

IV. WEB SERVICE SECURITY

The various security aspects that are related to the used web services are explored in the following subsections.

IV.I. Server Security

The server hosting the web services is protected as follows: the secure Linux platform is used to operate the host, all unwanted

services are disabled, strong password rules are imposed on all users, all applications are regularly updated/upgraded to close any security holes, and anti-virus software is constantly running in the background.

```
<MFEP>
  <MsgHeader>
    <TimeStmp>2019-04-12T23:15:38</TimeStmp>
    <BillerCode>450</BillerCode>
    <MsgTyp>STDVALREQ</MsgTyp>
  </MsgHeader>
  <MsgBody>
      <AccountInfo>
          <BillingNo>20191209029</BillingNo>
      </AccountInfo>
      <DueAmount>2320.000</DueAmount>
      <ServiceType>Tuition</ServiceType>
  </MsgBody>
  <MsgFooter>
      <BodySig>I75Ro...lUlwQRz</BodySig>
  </MsgFooter>
</MFEP>
```

Fig. 6. Input message for processStudentValidaion.

```
<MsgBody>
      <Result>
        <ErrorInfo>Success</ErrorInfo>
        <ErrorType>Info</ErrorType>
      </Result>
      <AccountInfo>
        <BillingNo>20191209029</BillingNo>
      </AccountInfo>
      <DueAmount>2320.0</DueAmount>
      <ServiceType>Tuition</ServiceType>
</MsgBody>
```

Fig. 7. Output message body for processStudentValidaion.

```
<MsgBody>
    <Transaction>
        <AccountInfo>
            <BillingNo>20191209029</BillingNo>
        </AccountInfo>
        <JOEBPPSTrans>20190412213051864</JOEBPPSTrans>
        <BankTransID>97547834</BankTransID>
        <BankCode>250</BankCode>
        <DueAmount>2320</DueAmount>
        <ProcessDate>2019-04-12T23:02:21</ProcessDate>
        <STMTDate>2019-04-13</STMTDate>
        <AccessChannel>Mobile</AccessChannel>
        <ServiceType>Tuition</ServiceType>
    </Transaction>
</MsgBody>
```

Fig. 8. Input message body for processPaymentNotification.

```
<MsgBody>
    <Transaction>
        <JOEBPPSTrans>20190412213051864</JOEBPPSTrans>
        <ProcessDate>2019-04-12T23:02:21</ProcessDate>
        <STMTDate>2019-04-13</STMTDate>
        <Result>
            <ErrorInfo>Success</ErrorInfo>
            <ErrorType>Info</ErrorType>
        </Result>
    </Transaction>
</MsgBody>
```

Fig. 9. Output message body for processPaymentNotification.

IV.II. Network Security

The HTTPS protocol is enabled in the Java EE application server to secure communication between the eFAWATEERcom client and the e-payment web services. Furthermore, the security firewall is configured (see Fig. 10) to block any unauthorized access to the MyGJU database and web services servers from the internet or intranet. Accordingly, only administrators are granted access to both servers from certain static IP addresses and via secure Virtual Private Network (VPN) connections. Besides, the eFAWATEERcom client can only connect to the web services HTTPS port via secure VPN channels from an authorized set of static IP addresses. Yet, any client may access the MyGJU portal via a dedicated HTTPS port.

IV.III. Data Integrity Protection

As an extra security measure, public key cryptography [22] is utilized to ensure that the body of any exchanged message is not altered in transit (i.e., to protect the integrity of the financial transaction). Therefore, eFAWATEERcom and GJU exchange public keys to enable a receiving side to decrypt messages signed by the private key of the sending side. Accordingly, a web service (or eFAWATEERcom) method produces a message body verification signature as follows:

- The GJU (or eFAWATEERcom) private key is used (see Fig. 11) to encrypt (i.e., sign) the body text of a response (or request) message to be sent. In that context, the Java security API [23] is used to encrypt the message body.
- The signed message body (i.e., signature) is then placed in the BodySig section (see Fig. 6) within the response (or request) message footer. Hence, the receiving side can use the public key of the sending side in addition to the body and signature in a received message to authenticate and verify the integrity of the received message body

On the other hand, a web service (or eFAWATEERcom) method can verify a received signature as follows:

- The body section (see Fig. 6) from a request (or a response) message is extracted.
- The signature text (see Fig. 6) in a BodySig section is extracted.
- The Java security API methods (see Fig. 12) are used to verify the integrity of the message body by using the eFAWATEERcom (or GJU) public key to decrypt the extracted signature and then match (i.e., verify) the result with the received message body.

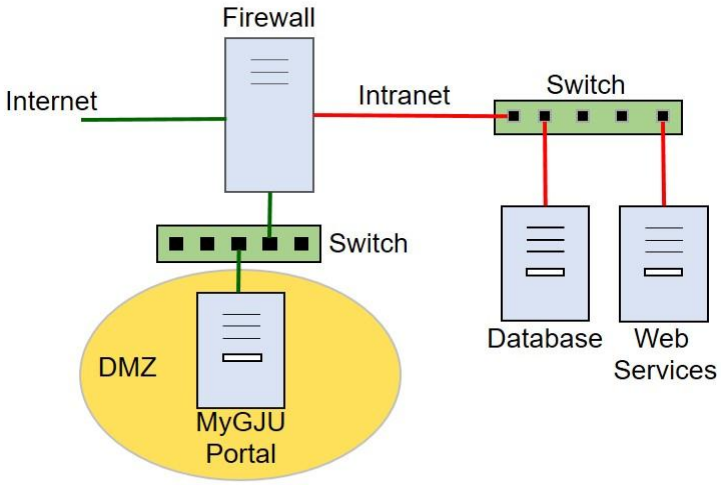


Fig. 10. The secure network configuration.

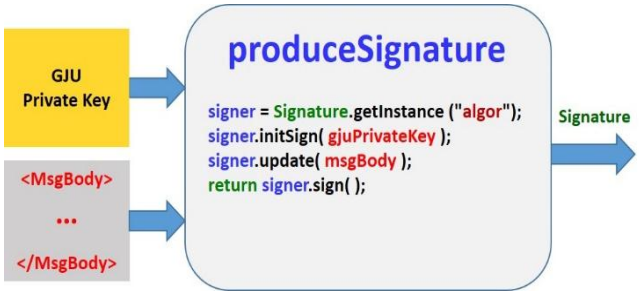


Fig. 11. The produceSignature software module.

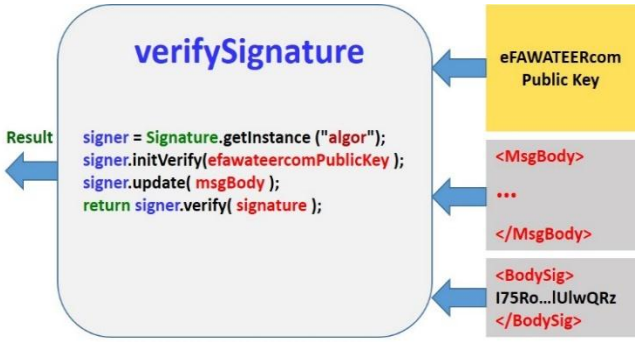


Fig. 12. The verifySignature software module.

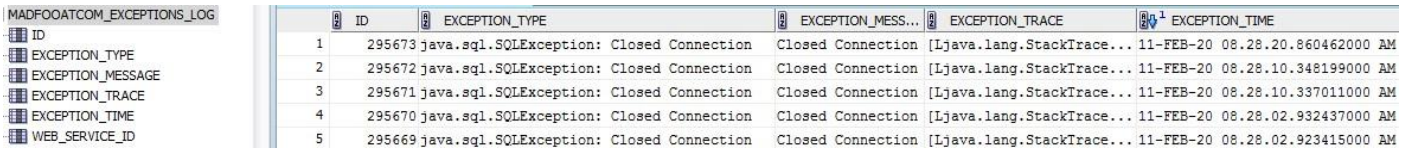
V. WEB SERVICE ROBUSTNESS

The processing methods of the web services are designed to be robust by handling input errors as well as software exceptions. Accordingly, users can investigate what caused an error for possible solutions. The main mechanisms to achieve web service robustness are:

- The BillerCode value (see Fig. 6) of the message recipient is first extracted and verified to make sure not to process the message in case the intended destination is incorrect.
- The message signature is also extracted and verified, as explained in subsection IV.III, to cancel the transaction and report an error in case the message signature and body do not match (i.e., the validation fails).
- The XML input syntax errors (e.g., unterminated element, wrong element name, invalid element content, etc.) are detected and reported while parsing the XML request message.
- Software exceptions (e.g., XML parser, I/O, database connection, and NULL pointer exceptions) are caught, logged (as shown in Fig. 13), and reported for further debugging and investigation.
- Every received request message and its corresponding response message are saved in a related database table for archiving and debugging purposes.

VI. VALIDATION AND RESULTS

The e-payment service was launched on the first 2016/2017 semester. Since then, it gained wide popularity amongst students because it is very practical and comfortable. According to Fig. 14, the percentage of the electronic payments amount with respect to total payments amount (i.e., the amount of e-payments and in-person payments) in the first 2016/2017 semester was about 23%. Then when most students were informed about the service and its benefits, the percentage of the e-payments amount jumped to almost 96% in the second 2016/2017 semester and maintained similar or higher values in the following semesters.



	ID	EXCEPTION_TYPE	EXCEPTION_MESS...	EXCEPTION_TRACE	EXCEPTION_TIME
1	295673	java.sql.SQLException: Closed Connection	Closed Connection	[Ljava.lang.StackTrace...	11-FEB-20 08.28.20.860462000 AM
2	295672	java.sql.SQLException: Closed Connection	Closed Connection	[Ljava.lang.StackTrace...	11-FEB-20 08.28.10.348199000 AM
3	295671	java.sql.SQLException: Closed Connection	Closed Connection	[Ljava.lang.StackTrace...	11-FEB-20 08.28.10.337011000 AM
4	295670	java.sql.SQLException: Closed Connection	Closed Connection	[Ljava.lang.StackTrace...	11-FEB-20 08.28.02.932437000 AM
5	295669	java.sql.SQLException: Closed Connection	Closed Connection	[Ljava.lang.StackTrace...	11-FEB-20 08.28.02.923415000 AM

Fig. 13. The exceptions log database table and data.

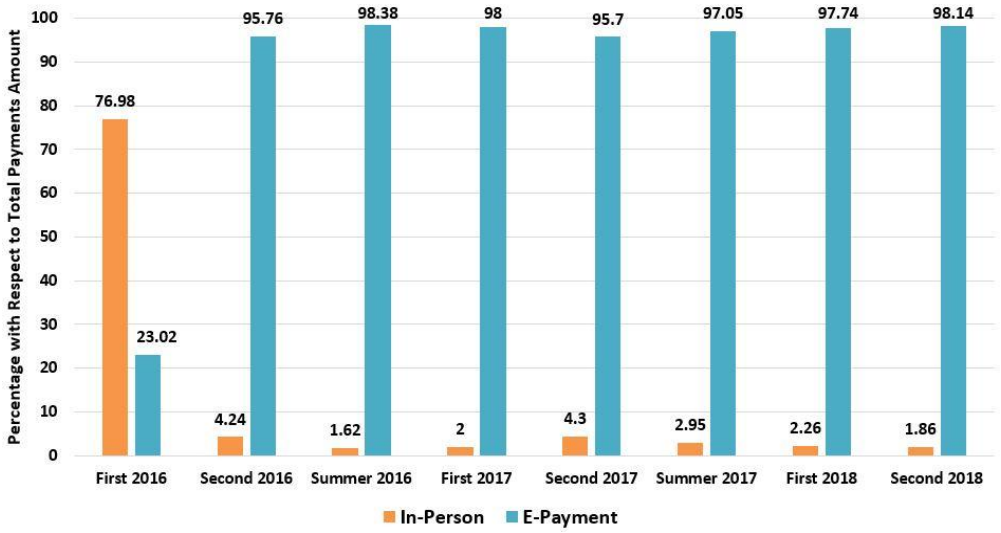


Fig. 14. The percentage of in-person payments versus e-payments amounts with respect to the total payments amount for eight academic semesters.

Further, the number of recorded transactions since the launch of the service is about 24972 transactions without any security issues. Only 78 transactions (i.e., about 0.3% of the overall transactions) had to be repeated due to database connection timeouts.

Besides, based on the student satisfaction survey results, shown in Fig. 15, most of the GJU students (1972 out of 2230 students; i.e., 88.3% of the students who participated in the survey) strongly agreed or agreed that the online tuition payment service is very convenient and useful.

Not to mention, the cost of an online tuition payment when a student uses e-banking as a payment channel varies as shown in Table 2. On the other hand, when a student uses a credit card as a method of payment the cost is 2.5% the paid amount. Accordingly, the offered service proved to be secure, robust, popular, and inexpensive.

VII. SUMMARY

An e-payment processing architecture that is based on RESTful web services is introduced in this paper. The web services can exchange financial transactions with all Jordanian banks via the eFAWATEERcom platform to process online tuition payments. Two types of web services are discussed in this work. The StudentValidation web service is used to validate the identity of a student and thus prevent a bank from sending a payment to a wrong student account. The PaymentNotification web service is utilized to record a payment transaction in the related student account.

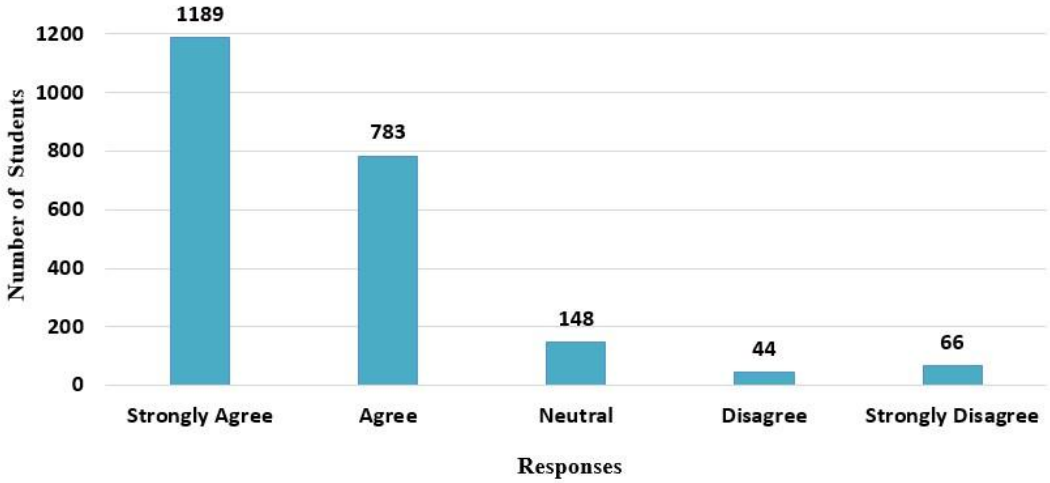


Fig. 15. The student satisfaction survey results.

Table 2. The cost of a tuition payment via e-banking.

Payment Amount (in Jordanian Dinar)	Payment Cost (in Jordanian Dinar)
1-500	0.5
501-1000	1
More than 1000	2

The Java API for RESTful Web Services (JAX-RS) is used to implement the e-payment web services. Accordingly, annotations are used to define the resource classes and the actions that the clients may perform on those resources. The HTTPS protocol is used to exchange financial transactions between banks and web services via the eFAWATEERcom platform. The security of the server and network, as well as the integrity of the data, are addressed to secure the web services. The input errors and software exceptions are also handled to make the web services robust.

The e-payment service has been heavily used by the GJU students since its deployment because it allows them to pay and then register for courses from anywhere. Nevertheless, it will be expanded in the future to allow students to pay the fees of the admission application, student proof, official transcript, graduation, parking, and transportation online.

REFERENCES

[1] OECD, "E-commerce, electronic payments," *OECD Digital Economy Papers*, no. 117, pp. 1-57, 2006.

[2] B. U. I. Khan, R. F. Olanrewaju, A. M. Baba, A. A. Langoo, and S. Assad, "A compendious study of online payment systems: past developments, present impact, and future considerations," *International journal of advanced computer science applications*, vol. 8, no. 5, pp. 256-271, 2017.

[3] M. Masihuddin, B. U. I. Khan, M. Mattoo, and R. F. Olanrewaju, "A survey on e-payment systems: elements, adoption, architecture, challenges and security concepts," *Indian Journal of Science Technology*, vol. 10, no. 20, pp. 1-19, 2017.

[4] M. Niranjanamurthy, N. Kavyashree, S. Jagannath, and D. Chahar, "Analysis of e-commerce and m-commerce: advantages, limitations and security issues," *International Journal of Advanced Research in Computer Communication Engineering*, vol. 2, no. 6, pp. 2360-2370, 2013.

[5] PayPal. (2020). *PayPal*. Available: https://www.paypal.com

[6] bitcoin. (2020). *bitcoin*. Available: https://bitcoin.org/en/

[7] MadfooatCom. (2020). *MadfooatCom*. Available: http://madfooat.com

[8] A. Al-Ashqar, "The impact of electronic bills on customer satisfaction: a field study on Efawateercom users," M.S. Thesis, Middle East University, Amman, Jordan, 2018.

[9] F. Al-Hawari, "MyGJU student view and its online and preventive registration flow," *International Journal of Applied Engineering Research*, vol. 12, no. 1, pp. 119-133, 2017.

[10] F. Al-Hawari, A. Alufeishat, M. Alshawabkeh, H. Barham, and M. Habahbeh, "The software engineering of a three - tier web - based student information system (MyGJU)," *Computer Applications in Engineering Education*, vol. 25, no. 2, pp. 242-263, 2017.

[11] T. Sheeba, S. Begum, and I. AlHarthy, "Student self service system " *Journal of student research*, vol. 1, pp. 1-5, 2017.

[12] G. Kobusinge, "Online fees payment system for Makerere University (MUK-OFPS)," Project Report, Department of Information Systems, Makerere University, Kampala, Uganda, 2013.

[13] H. Zhu, Y. Xie, M. Hou, K. Yan, and T. Li, "Application of online payment mode in university charging management," in *International Conference on Mechatronics and Intelligent Robotics*, 2018, pp. 225-230: Springer.

[14] M. Nyondo and N. Lameck, "Design and development of a secondary school payment system," *The International Journal of Multi-Disciplinary Research*, pp. 1-23, 2020, Art. no. CFP/1517/2020.

[15] A. S. Misal, S. R. Misal, A. R. Chavhanke, and P. A. Ambatkar, "Online fee payment system," *International Journal for Research in Applied Science & Engineering Technology*, vol. 4, no. 3, pp. 806-809, 2016.

[16] H. Nasereddin and S. Khazneh, "An empirical study of factors affecting the acceptance of mobile payments in Jordan," *International Journal of Recent Research and Applied Studies*, vol. 29, no. 3, pp. 110-121, 2016.

[17] A. San Martino and X. Perramon, "Defending e-banking services: antiphishing approach," in *Second International Conference on Emerging Security Information, Systems and Technologies*, 2008, pp. 93-98: IEEE.

[18] Y. Prihastomo, A. N. Hidayanto, and H. Prabowo, "The key success factors in e-marketplace implementation: a systematic literature review," in *International Conference on Information Management and Technology (ICIMTech)*, 2018, pp. 443-448: IEEE.

[19] F. Al-Hawari, "Analysis and design of an accounting information system," *International Research Journal of Electronics and Computer Engineering*, vol. 3, no. 2, pp. 16-21, 2017.

[20] R. T. Fielding and R. N. Taylor, "Architectural styles and the design of network-based software architectures," Doctoral dissertation, University of California, Irvine Irvine, 2000.

[21] J. Juneau, *Introducing Java EE 7: a look at what's new*. New York, NY, USA: Apress, 2013.

[22] J. Katz and Y. Lindell, *Introduction to modern cryptography*. Boca Raton, FL, USA: CRC Press, 2007.

[23] S. Oaks, *Java security: writing and deploying secure applications*, 2nd ed. Newton, MA, USA: O'Reilly Media, 2001.