ASHESI

ASHESI UNIVERSITY COLLEGE

QR CODE VERIFICATION SYSTEM FOR ASHESI TRANSCRIPT

Applied Project

B.Sc. Computer Science

Ibrahim Abdullah

2018

ASHESI UNIVERSITY COLLEGE

QR Code Verification System for Ashesi Transcript

Applied Project

Applied project submitted to the Department of Computer Science, Ashesi University College in partial fulfilment of the requirements for the award of Bachelor of Science degree in Computer Science

Ibrahim Abdullah

April 2018

DECLARATION

I hereby declare that this Applied Project is the result of my own original work and that no part of it has been presented for another degree in this university or elsewhere.

Candidate's Signature:

………………………………………………………………………………………

Candidate's Name:

………………………………………………………………………………………

Date:

………………………………………………………………………………………

I hereby declare that preparation and presentation of this Applied Project were supervised in accordance with the guidelines on supervision of Applied Project laid down by Ashesi University College.

Supervisor's Signature:

………………………………………………………………………………………

Supervisor's Name:

………………………………………………………………………………………

Date:

………………………………………………………………………………………

ACKNOWLEDGEMENT

I give thanks to Almighty Allah for the strength to complete this capstone project.

To my supervisor, Dr. Suzanne Buchele and the computer science department of Ashesi University College, I am thankful for the continuous support and assistance over the course of my studies. I am also grateful to the academic registry team at Ashesi University for their assistance.

To the MasterCard Foundation and their representative team at Ashesi University College, I am very grateful for the scholarship to pursue my undergraduate studies.

My sincere gratitude goes to my parents; Mr. Abddulai Ibrahim and Mrs. Rahinatu Abdulai for their unconditional love and support.

ABSTRACT

The current approach adopted by universities in Ghana to verify transcripts is time-consuming and error-prone. The verification process involves personnel comparing students' grades on a hard copy transcript to grades retrieved from the academic institution's database. This approach increases the possibility of academic institutions mistakenly approving incorrect or falsified transcripts as genuine. In this project, a software system is developed to automate the transcript verification process at Ashesi University College. The system is used to generate transcripts with QR code embedded on them. The information stored in the QR code is used to automate the transcript verification process.

Table of Contents

Chapter 1: Introduction

1.1 Introduction

Most tertiary institutions in Ghana have digitized the academic records of its students. This means student academic records are stored in a localized or internet database and retrieved whenever needed. This has made student academic records accessible from multiple locations and reduced time spent in searching for specific records as compared to paper record keeping. However, digitization of student academic records means academic institutions must deal with the challenging issue of ensuring data authenticity and correctness. The digitized student academic records must be protected from any form of unauthorized access. Otherwise, there will exist the possibility of academic institutions issuing incorrect academic transcript when the digital academic records are maliciously modified. In cases where transcripts are issued before the digital data is compromised, the paper transcripts can also be changed to have the same information as the corresponding digital record due to advancement in printing, scanning, and copying technologies (Yahya et al., 2017). As a result, academic institutions must implement unbreakable protection technologies to protect their digital student academic records.

Unfortunately, the task of academic institutions implementing unbreakable protection technologies to protect their digital student academic records is difficult to achieve. This is because even the leading internet companies in the world continues to struggle with full protection of their digital data from unauthorised access. For example, Yahoo, one the leading internet companies in the United State had billions of its client account breached in 2013(Mullen & Fiegerman, 2017). Therefore, complete protection of digital student records is not a guarantee. Academic institutions must, therefore, implement effective measures and technologies to detect incorrect or falsified transcripts that may result from compromised digital student records or transcripts during verification. This will serve as a complement to already implemented security measures to prevent unauthorised access to students' academic records. Should an unauthorised user succeed in gaining access to modify students' records or generate falsified transcripts, academic institutions should be able to detect such incorrect information when submitted for verification.

1.2 Problem Description

Currently, the approaches adopted by universities in Ghana to verify the integrity of transcripts are ineffective. With reference to Ashesi University College, transcript verification is done manually by personnel comparing student grades retrieved from the internet database with that on a scanned, electronic, or hard copy of the transcript. This approach is time-consuming and does not scale with the increase in student and alumni population. Ashesi University College currently has student and alumni population of less than two thousand (Ashesi University College, 2018). Hence, few transcripts are received at the academic registry office for verification. However, when the student and alumni population grow to about ten thousand, the current approach will become inefficient and unscalable. More transcripts will be received at the academic registry for verification due to the increase in student and alumni population. Unless more personnel are hired to keep up with the volume of transcripts that will have to be verified, the current approach will be time-consuming. Nevertheless, hiring more personnel is not a scalable approach for making the transcript verification process efficient. Also, with an increase in the population of student and alumni, the possibility of the academic registry approving falsified transcript will be on the increase. This can be attributed to mistakes by the academic registry personnel due to distractions and tiredness. In such situations, the current approach of verifying transcripts will be ineffective, less efficient and error-prone.

Moreover, it is possible to have digitized student academic records maliciously changed and transcript, either hard or electronic copy, modified to have the same information as the corresponding digital record. In such a case, the current approach of verifying transcripts might not detect the changes in the students' records. Hence, the current process of verifying transcripts becomes ineffective. This might result in Ashesi University College approving falsified transcripts which do not reflect the academic capabilities of its students.

An implication of academic institutions approving incorrect or falsified transcripts is a loss of brand integrity to effectively treating student unequally (Guhr, 2012). A university's core business is to provide quality education and award degrees that correspond to the capabilities of its students. In so doing, academic institutions must always keep track of who achieved what and when effectively. Hence, when academic institutions continue to approve falsified academic information or transcripts as authentic, they will lose the trust of organizations, individuals, and all stakeholders as an institution capable of serving its core mandate. It is therefore imperative that effective and efficient ways of verifying transcripts are implemented by academic institutions to avoid the approval of incorrect or falsified transcripts.

The aim of this project is to develop a software system that will address problems pertaining to the current approach used by the academic registry at Ashesi University College to verify transcripts. The proposed solution is further discussed in section 1.4 of this report.

1.3 Technical background information

This section provides background information to the technical concepts used to describe the proposed solution in section 1.5. The concepts include encryption, Quick Response (QR) code, and hashing.

1.3.1 Encryption

Encryption is the process of transforming data or information into another form such that only the people it was intended for can read and understand (Singh & Garg, 2013). An encryption algorithm and a key are used to transform the information into the unintelligible text called ciphertext. A decryption algorithm and the key that was used to encrypt the information are required to convert the ciphertext back to a readable plain text format. Encryption and decryption, therefore, solve two kinds of security problems: privacy and authentication (Diffie & Hellman, 1976). The key is a secret information that the encryption algorithm uses to lock the information and it is required by the decryption algorithm to unlock the ciphertext. There are two types of encryption, namely, public or asymmetric key encryption and private or symmetric key encryption (Singh & Garg, 2013).

1.3.2 Quick Response (QR) code

Quick Response code is a two-dimensional barcode used to store text. Unlike one-dimensional barcodes, QR code stores text with different encoding in vertical and horizontal directions (Kapsalis, 2013). The text stored in the QR code can be retrieved using a QR code reader software. Another advantage of QR code over one-dimensional barcode is that information stored in QR codes can be retrieved successfully if the QR code is partially damaged (Kapsalis, 2013). There are several versions of QR codes with different capacities. The largest versions can store up 4296 alphanumeric characters (Kapsalis, 2013).

Figure 1.1: QR code image

1.3.3 Hashing

Hashing is the use of a mathematical or hash function to compute a value or values called hash codes from a text (Adamchik, 2009). A good hash function always generates different hash codes for different text. Additionally, the resulting hash code is usually shorter in length than the original text. Also, given a hash code, the original text that resulted in the given hash code after applying the hash function on it cannot be reproduced (Pornin, 2013). This feature of hash code makes it a reliable approach for detecting changes made to messages. To detect changes made to a message using hashing, a hash of the intended message is computed. The receiver uses the same hash function to compute the hash of the message. When the two hash codes match, then the message has not been tampered with. Hashing is the industry standard method for verifying the integrity of messages.

1.4 Related Works and Existing Solution

A research team at St. Xavier's College (Autonomous), Kolkata has devised an approach to secure their digital student records and automate their mark-sheets verification process. Mark-sheet is a paper showing marks awarded to a student. The student marks are encrypted using the Trisha, Tamodeep, Joyshree, Shayan and Nath (TTJSA) algorithm before they are stored in the internet database. TTJSA is a symmetric key algorithm which uses three cryptographic methods namely, (i) Generalized Modified Vernam Cipher method, (ii) Meheboob, Saima and Asoke (MSA) method, and (iii) Neeraj, Joel, Joyshree, Amlan, Asoke (NJJSA) method (Verma & Gedam, 2014). The encrypted student's marks are stored in a QR code and embedded on the mark-sheet. During verification, the system reads the QR code embedded on the mark-sheet to retrieve the encrypted marks which are then decrypted using TTJSA decryption algorithm to retrieve the actual marks. Some personnel then compare the decrypted marks with the marks on the mark-sheet to verify the credibility of the student's grades. This approach detects changes made to mark-sheets after they have been issued. This is because the encrypted marks stored in the QR code cannot be decrypted without the key used for encryption. The ciphertext produced by the TTJSA algorithm is proven to be unbreakable (Chatterjee et al., 2011). However, verification of mark-sheets is done manually where some personnel compare the marks on the mark-sheets with decrypted marks retrieved from the QR code. Hence, mistakes made by the personnel can result in the approval of incorrect mark-sheets.

Yahya et al. (2017) propose a system for securing paper certificates. Multiple academic institutions can use the proposed system to secure their students' certificates. This approach uses a mobile application to verify the authenticity of certificates. When generating certificates, the system retrieves and encrypt student's marks from the relevant academic institutions' database. The encrypted student record (marks) is stored on a different database server with a unique record identification (ID). The record ID is encrypted, and the ciphertext stored into a QR code. The generated QR code is embedded on the student certificate, which is later used for verification.

A mobile application is used to scan the QR code embedded on the certificate to read the encrypted record ID. The record ID is used for identifying the corresponding student record in the database server. The mobile application sends the encrypted record ID to a web server which decrypts the ID and uses the plain text to retrieve the corresponding encrypted student marks. The server also decrypts the encrypted student's marks and sends it to the mobile application. A human then manually compares the retrieved marks on the mobile application to the marks on the certificate to verify its authenticity.

The proposed approach by Yahya et al. for verifying student certificates can detect changes made to the hard copy of the certificate after they are issued. This is because the encrypted record ID stored in the QR code cannot be modified in any beneficial way. However, if the student marks are stored on the academic institution's database without strong protection, they will be susceptible to breaches. Hence, this approach does not provide a complete solution to the problem of academic institutions approving certificates that do not reflect the capabilities of its students. Additionally, the absolute reliance on a human to verify the authenticity of certificates by comparing marks on the student's certificate to marks retrieved from the internet database makes this approach error-prone. Moreover, any organization or individual that will want to verify certificate generated using this system will have to download and install the mobile application. However, certificate verification is not an activity that organizations and individuals do on daily basis. Hence, organizations and individuals will be reluctant to install an application just to verify the certificate of some few students or new employees. Hence, a web portal that can be accessed by organizations and individuals without any restriction for verifying certificates would be a better option.

1.5 Proposed Solution

The proposed solution to the problems identified with the current approach adopted by Ashesi University's academic registry to verify transcripts is a software system (hereinafter referred to as Ashesi Transcript System) that will automate the transcript verification process. The Ashesi Transcript System will be used to generate transcripts that will have embedded on it, a QR code. The QR code will store encrypted information that will be used to verify the authenticity of the transcript. When verifying a transcript, the system will read and decrypt the content of the QR code embedded on the transcript. The system will also read the text on the transcript and automatically compare it with the decrypted content of the QR code to verify the authenticity of the transcript.

This proposed approach of verifying transcripts will be able to detect changes made to transcripts after they are issued by the Ashesi academic registry. This is because, when the information on the transcript is changed, the QR code content cannot be changed in any beneficial way. In cases where the QR code content can be modified, the encrypted content of the QR code that the system will have generated given the information on the transcript cannot be produced without access to the encryption key, especially when the system uses standard encryption and secret key generator algorithms. The system will also detect falsified transcripts that were not issued by the Ashesi academic registry. Also, with this software system, the possibilities of the academic registry approving falsified transcripts as authentic due to distractions and tiredness will be minimal. The Ashesi Transcript System will also reduce the amount of time used to verify transcripts as compared to the currently adopted approach.

Hence, this proposed solution is an efficient, effective, and secured approach of generating and verifying transcripts. The proposed solution can also be used to secure PDF documents.

Chapter 2: Requirements

2.1 Requirement Design Overview

The requirements for the Ashesi Transcript System were gathered through personal interactions with the Ashesi academic registry personnel and the Provost. Interactions were mainly through interviews where the academic registry personnel described the current approach used to generate and verify transcripts, challenges associated with the current approach and their expectation of the new system that would address such challenges. Through these interactions, users, use cases, functional and non-functional requirements of the system were identified. These requirements were validated through subsequent meetings with the academic registry personnel, the Provost, and the Information Technology (IT) team at Ashesi University College. The Ashesi IT team provided more insights into the system requirements of the software given the setup of the university's IT infrastructure and how the new software can integrate with other systems deployed by the university. Through these subsequent engagements, some of the initial specified requirements were modified, and more were added. Also, new users and use cases identified during the subsequent meetings were included.

In the following subsections of this chapter, key features of the system, users and their characteristics, use cases, system requirements, functional and non-functional requirements are specified and elaborated.

2.2 Key features of the system

The Ashesi Transcript System will have the following key features.

- The system should support uploads of PDF file for processing.
- The system should be able to extract text and images from uploaded PDF document.

- The system should be able to generate and read QR code images.
- The system should be able to embed QR code on generated Ashesi transcript.
- The system should be able to read QR code embedded on PDF document, either electronic or scanned copies.
- The system should be able to encrypt and decrypt text with standard encryption and decryption algorithms.
- The system should be able to read text from scanned PDF document and images using Optical Character Recognition.
- The system should automate the process of verifying Ashesi transcripts.
- The system should always authenticate and only give access to people who have the authorization to use it.

2.3 Users /Actors

There are two users of the system: Ashesi academic registry personnel and a system administrator (Ashesi IT personnel). The roles and use cases of both the academic registry personnel and the system administrator are elaborated in section 2.3.1 and 2.3.2 respectively.

2.3.1 Ashesi academic registry personnel

The Ashesi academic registry personnel will have 'Registrar' as a role. With this role, the academic registry personnel will use the system to generate and verify transcripts. Figure 2.1 is a use case diagram that depicts how the academic registry personnel will use the system. Table 2.1 is a use case table that describes how each of the use cases of the academic registry personnel, will be accomplished with the corresponding system responses to each user action.

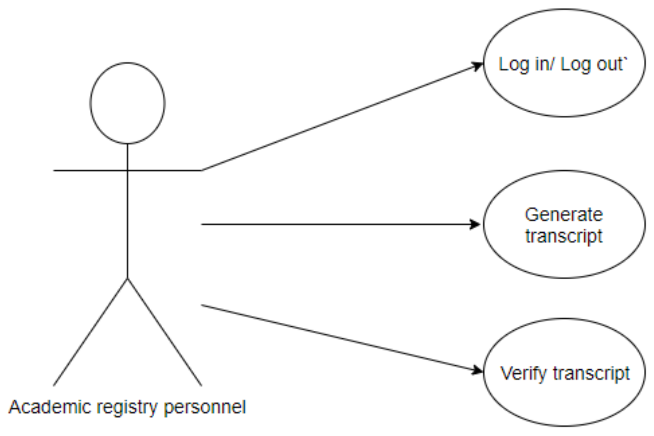


Figure 2.1: Use case diagram for academic registry personnel

The use case diagram above depicts the use cases of academic registry personnel. From the diagram, an academic registry personnel can log in and out of the system, generate and verify transcript.

Table 2.1: Use case table for academic registry personnel

Use case: Login/Logout	**Primary Actor: Academic registry personnel Requirement ID: DA 01 Priority: Highest**
Interested stakeholders:	Ashesi academic registry team
Description:	This use case describes how personnel of the Ashesi academic registry will log in and out of the system.
Goal:	Authenticate user credentials before granting access to the system.
Success measurement:	An academic registry who has access to the system successfully log in with correct credentials. Access to the system is denied for users with incorrect credentials.
Precondition:	The user has been given access to the software by a system administrator.
Trigger:	The user accesses the web application from a web browser.

Relationships:	Successful login leads to generate transcript webpage.
Event flow:	1. The user clicks on the login with Ashesi Office 365 credentials button. 1.1 The system displays Microsoft login page. 1.2 The user enters Ashesi Office 365 credentials. 1.3 The system displays user-management page after successful authentication of user credentials. 1.3.1 The user clicks on logout. 1.3.2 The system destroys user session, log the user out and display the login page. 2. The system displays error messages when authentication fails.
Use case: Generate transcript	**Primary Actor: academic registry personnel** **Requirement ID: DA 02 Priority: Highest**
Interested stakeholders:	Ashesi academic registry
Description:	This use case describes how an Ashesi academic registry personnel will use the system to generate a transcript.
Goal:	The system should generate a transcript with a QR code storing encrypted information for verifying transcript embedded on them.
Success measurement:	The academic registry personnel generate transcript in the correct and desired format.
Precondition:	The user has successfully logged in to the system.
Trigger:	The user uploads a PDF document with the student's academic information on the generate transcript webpage.
Relationships:	Read and encrypt information on the transcript, encode the encrypted data in a QR code and embed the QR code on the transcript.
Event flow:	1. The user clicks on generate transcript menu item. 2. The system displays generate transcript webpage. 3. The user selects a PDF document with student's academic information from the user's computer and clicks on the upload button. 3.1 The system reads the information on the uploaded transcript, compute its hash code, and encrypt the hash code. 3.2 The system generates QR code which stores the encrypted hash code and embeds the QR code on the transcript.

	3.3 The system displays the generated transcript in the browser for the user to download. 4. The user clicks on the upload button with no file selected. 4.1 The system displays errors on the generate transcript page without any file uploaded.
Use case: Verify transcript	**Primary Actor: academic registry personnel** **Requirement ID: DA 03 Priority: Highest**
Interested stakeholders:	Ashesi academic registry
Description:	This use case describes how an academic registry personnel will use the system to verify transcripts.
Goal:	Verification of transcripts.
Success measurement:	The system verifies the authenticity of the uploaded transcript and displays the verification status on a webpage.
Precondition:	The user has successfully logged into the system.
Trigger:	The user clicks on the verify transcript menu item after successful login to the system.
Relationships:	The system reads the encrypted QR code content and decrypt it, reads information on the transcript and compute its hash code, do Optical Character Recognition on scanned PDF document.
Event flow:	1. The user clicks on verify transcript on the menu bar. 2. The system displays the verify transcript page. 3. The user uploads a PDF document (scanned or original) of the transcript to be verified. 4. The user clicks verify transcript button. 4.1 The system reads the encrypted data stored in the QR code embedded on the transcript, decrypt the data, and compares it with the hash code of the information read from the uploaded transcript being verified. 4.2 The system displays the verification status of the transcript being verified (either correct or not). 5. The user clicks on the upload button with no file selected. 5.1 The system displays errors on the verify transcript page without any file uploaded.

2.3.2 System administrator

The system administrator will have a user role as ‘admin’ in the system. The system administrator can add new users to the system and also revoke the access of existing users; both registrars and other system administrators. Figure 2.2 is a use diagram that depicts how the system administrator will use the system. Table 2.2 is a use case table that describes how each of the use cases of the system administrator, will be accomplished with the corresponding system responses to each user action.

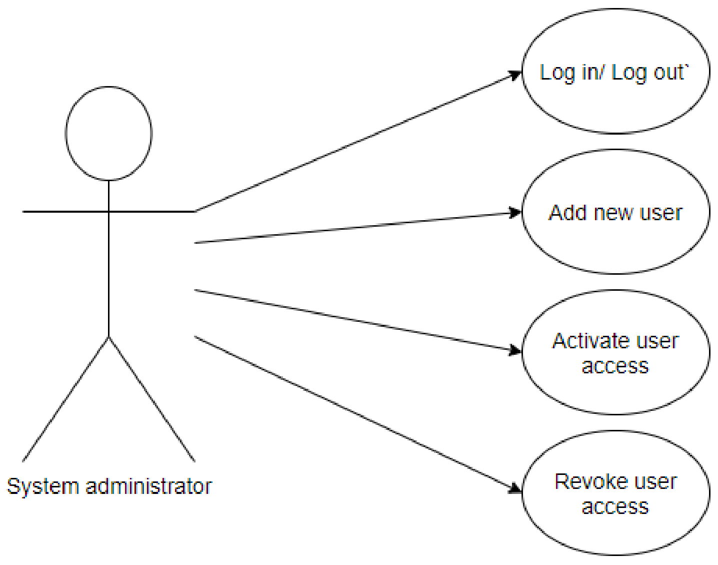


Figure 2.2: Use case diagram for the system administrator

The system administrator can log in and out of the system, add new users and revoke the access of existing user to the system.

Table 2.2: Use case table for the system administrator

Use case: Login /Log out	**Primary Actor: System administrator** **Requirement ID: DA 04 Priority: High**
Interested stakeholders:	The academic registry team, IT team and the Provost of Ashesi University.
Description:	This use case describes how a system administrator will log in and out of the system.
Goal:	Authenticates user credentials before granting access to the system.
Success measurement:	The user is granted access to the system after successful authentication of login credentials. If the login credentials are incorrect, the user is denied access to the system.
Precondition:	The user accesses the login screen of the system using the system domain name.
Trigger:	The user accesses the login page, enter correct login credentials and click on the login button.
Relationships:	Access to the user management page in the system.
Event flow:	1. The user clicks on the login with Ashesi Office 365 credentials button. 1.1 The system displays Microsoft login page. 1.2 The user enters Ashesi Office 365 credentials. 1.3 The system displays user-management page after successful authentication of user credentials. 1.3.1 The user clicks on logout. 1.3.2 The system destroys user session, log the user out and displays the login page. 1.4 The system displays error messages when authentication fails.
Use case: Add new user	**Primary Actor: System Administrator** **Requirement ID: DA 05 Priority: High**
Interested stakeholders:	The academic registry team, IT team and the Provost of Ashesi University.
Description:	This use case describes the process of how a system administrator will add new users to the system.
Goal:	Add a new user to the system.

Success measurement:	The newly added user can successfully log into the system with their correct Ashesi Office 365
Precondition:	The system administrator has successfully logged into the system.
Trigger:	The system administrator clicks on add new user button on the user management page.
Relationships:	
Event flow:	1. The system administrator logs into system. 2. The system administrator clicks on the add new user button on the user management page. 3. The system displays add new user page with a form. 4. The system administrator enters the Ashesi Office 365 credentials of the user to be added to the system. The system admin also selects the role of the new user and the status of their access to the system (either active or deactivated). 4.1 The system displays the user management with the account details of the newly added user listed among that of other users of the system. 4.2 The system displays the add new user page with error messages if the details of the new user entered have errors.
Use case: Revoke user access.	**Primary Actor: System Administrator** **Requirement ID: DA 06 Priority: High**
Interested stakeholders:	The academic registry team, IT team and the Provost of Ashesi University.
Description:	This use case describes how the system administrator will revoke a user's access to the system.
Goal:	Revoke user's access to the system.
Success measurement:	The user whose access to the system has been revoked can no longer log into the system.
Precondition:	The system administrator has successfully logged into the software.
Trigger:	The system administrator clicks on revoke user access button.
Relationships:	
Event flow:	1. System administrator logs into the system.

	2. System administrator clicks on the revoke user access button by the user's name.

2.4 Operating environment

The system will be deployed on Apache Tomcat server running on a Windows operating system. The system will also integrate with Ashesi Office 365 directory.

2.5 Functional requirements

This section documents functional requirements of the Ashesi Transcript System.

2.5.1 Generate transcript

Academic registry personnel should be able to use the system to generate transcripts with QR code embedded on it.

User requirement: Academic registry personnel should log in to the system and upload a PDF document with the student transcript information.

System requirement: The system should be able to generate transcript and render it in the browser at a very high speed.

Input/ Output: The input for this requirement should be an original PDF file with the student's transcript information. The output should be a PDF transcript with QR code embedded on it rendered in the browser for the academic registry personnel to save it on any storage device connected to the computer on which the system was accessed.

2.5.2 Verify transcript

The system should be able to verify the authenticity of transcripts.

User requirement: The academic registry personnel should log in to the system and upload a PDF of the transcript to be verified.

System requirement: The system should be able to correctly verify the authenticity of the uploaded PDF transcript. The transcript verification process should be completed at a faster rate than the manual approach of verifying transcript.

Input/ Output: The input for this requirement should be a PDF file (original or scanned) with the student transcript information. The output should be the authenticity status of the uploaded transcript; whether it is correct, or it needs manual inspection.

2.5.3 User management

The system administrator should be able to add new users and grant them access to the system. The system administrator can also revoke or deactivate existing users' access to the system.

User requirement: The user logs into the system as a system administrator.

System requirement: The system will use Ashesi office 365 API to authenticate the user credentials before given access to use the system.

Input/ Output: The system administrator should provide correct details of the new user and submit. When revoking access to the system of existing users, the system administrator should click on the revoke access button beside the user's name on the user management page. The details of the newly added user should be listed among other users of the application on the user management page. Also, the new user should be able to log in to the system.

2.5.4 User log in

Authorised users of the system should be able to log in to the system with their Ashesi office 365 credentials.

User requirement: The user accesses the login page of the system and enters login credentials.

System requirement: The system should be able to integrate with Microsoft Office 365 API.

Input/ Output: The user enters correct login Ashesi Office 365 credentials. Authenticated users are given access to the system. For academic registry personnel, generate transcript page will show after successful login. However, for the system administrators, the user management page will be displayed after successful login to the system.

2.6 Non-Functional requirements

The non-functional requirements of the system are grouped into the categories of security, performance, ethics, and error logging. Each of these categories is further explained in the sub-section below.

2.6.1 Security

The system should have standard security measures implemented to prevent unauthorized access. Additionally, keys for encryption and decryption must be protected from the reach of arbitrary users. The system should not in any instance change or maliciously manipulate the transcript information of any student in any way.

2.6.2 Performance

The system must be fast and robust to serve multiple users at the same time. Also, the system must be effective and efficient than the current system for generating and verifying transcripts.

Chapter 3: Architecture Design

3.1 Architecture Overview

The system is designed to have a modular structure with each module implemented to provide related services. For instance, all encryption and decryption operations are developed in one module while QR code operations are also developed in another module. This modular approach of designing the system allows the incremental development of the components of the system. Additionally, the modular structure makes testing and debugging of related functionalities easier and faster. This is because debugging errors related to a functionality will be focused on a module which will have a smaller code base as compared to the entire system's code base. This will fasten the detection and correction of errors.

3.2 Interface Design

This system has six interfaces. These are the login page, user management page, generate transcript page, verify transcript page, add new user page and transcript verification status page. The details of each of the interfaces and how users will interact with it are discussed in the following subsections.

3.2.1 Log in page

The users use the login page to submit their login credentials for authentication before granted access to the system. Figure 3.1 is the interface for the login page.

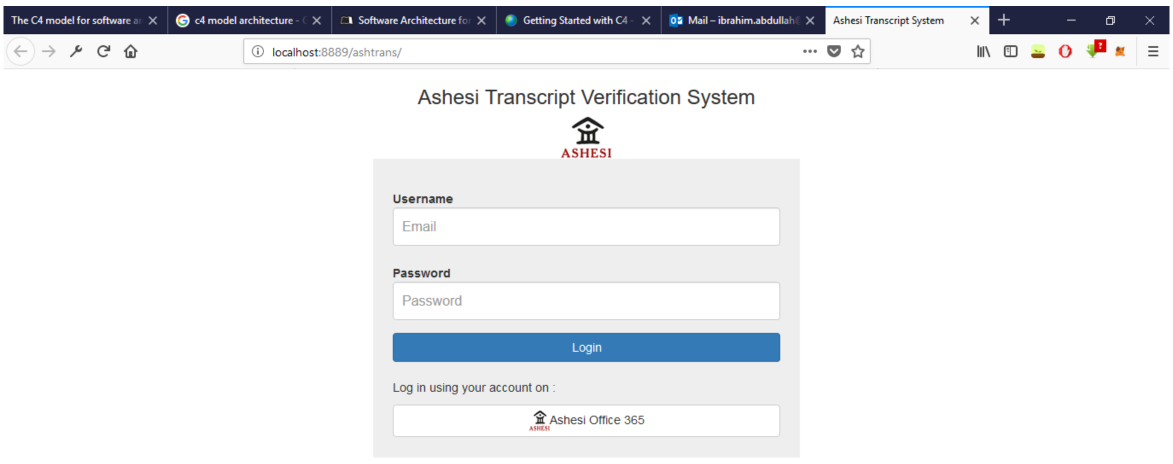
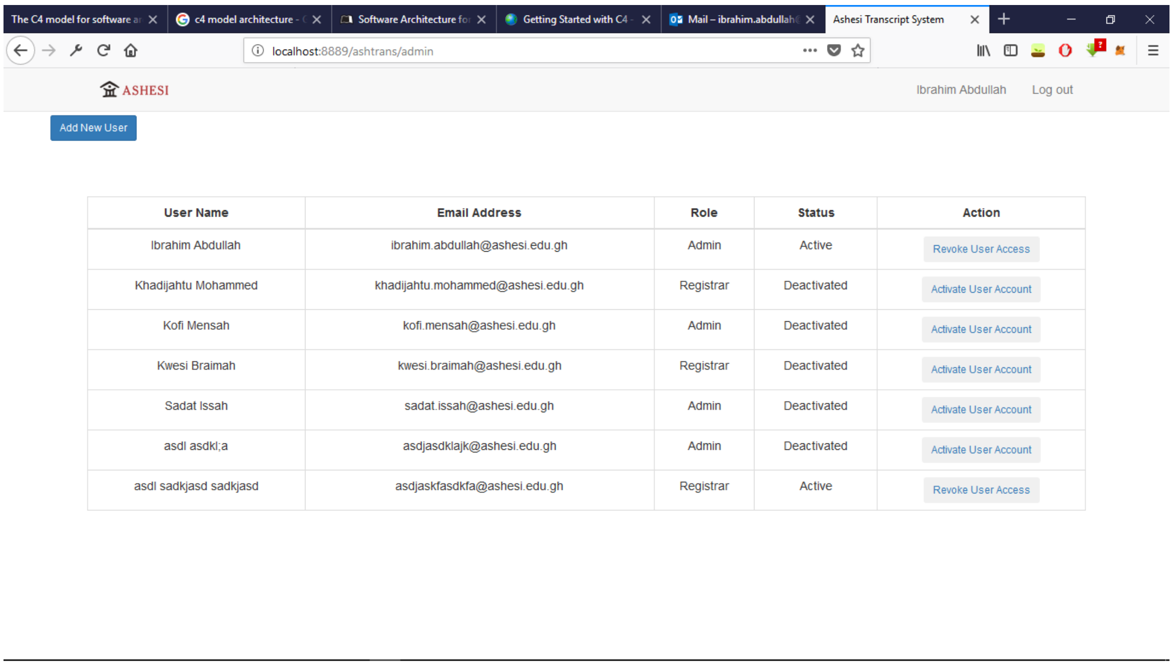


Figure 3.1: System login interface

3.2.2 User management page

The user management page provides functionalities through which the system administrator manages the users of the Ashesi Transcript System. The page shows a list of all users of the system in a table. Each user record has an associated button with which the system administrator can activate or revoke their access to the system. It also has a button which displays the add new user page when clicked. Figure 3.2 is the interface to the user management page. The current logged in system administrator can also revoke his or her access to the system. When that happens, the system administrator will not be able to log into the system again after logging out until another system administrator re-activates the account.

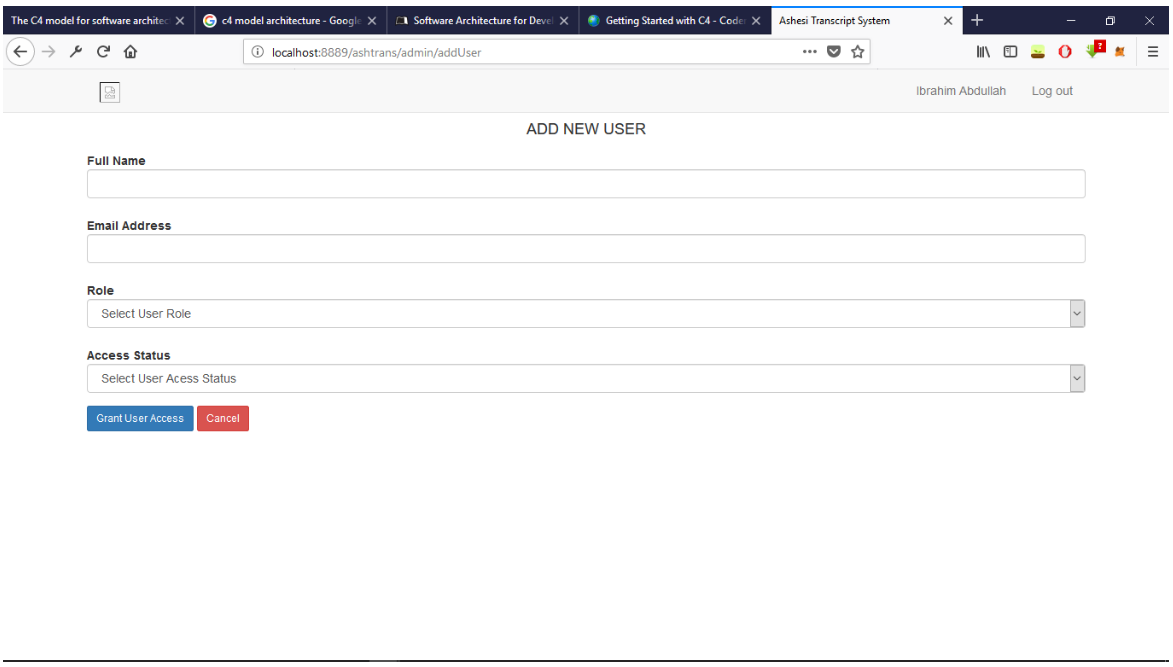


User Name	Email Address	Role	Status	Action
Ibrahim Abdullah	ibrahim.abdullah@ashesi.edu.gh	Admin	Active	Revoke User Access
Khadijahtu Mohammed	khadijahtu.mohammed@ashesi.edu.gh	Registrar	Deactivated	Activate User Account
Kofi Mensah	kofi.mensah@ashesi.edu.gh	Admin	Deactivated	Activate User Account
Kwesi Braimah	kwesi.braimah@ashesi.edu.gh	Registrar	Deactivated	Activate User Account
Sadat Issah	sadat.issah@ashesi.edu.gh	Admin	Deactivated	Activate User Account
asdl asdkl;a	asdjasdklajk@ashesi.edu.gh	Admin	Deactivated	Activate User Account
asdl sadkjasd sadkjasd	asdjaskfasdkfa@ashesi.edu.gh	Registrar	Active	Revoke User Access

Figure 3.2: User management interface

3.2.3 Add new user page

The system administrator uses this page add new users to the system. The new user can have a "Registrar" or an "Admin" role. The page has a form in which the system administrator enters the details of the new user. The systems administrator selects the role of the new user, either "Admin" or "Registrar" from a drop-down form input. Also, the user access status to the system must be specified, either active or not active. If the admin wants to add the new user but grant them access later, then the new user's access status must be set to "not active". The user will then not be log into the system until the account is set to active. Figure 3.3 is the interface for the add new user page.


The C4 model for software architec
c4 model architecture - Google
Software Architecture for Devel
Getting Started with C4 - Code
Ashesi Transcript System
localhost:8889/ashtrans/admin/addUser
Ibrahim Abdullah
Log out

ADD NEW USER

Full Name

Email Address

Role

Select User Role

Access Status

Select User Acess Status

Grant User Access Cancel

Figure 3.3: Add new user interface

3.2.4 Generate transcript page

Ashesi academic registry personnel use this page to generate transcripts. The page has a file picker button for selecting files from the computer on which the web application was accessed. After selecting a file, the user clicks on the upload button to upload the file on to the server. The system uses the uploaded file to generate the student transcript which is rendered into the browser. The academic registry personnel can then download the PDF document. Figure 3.4 is the interface of the generate transcript page.

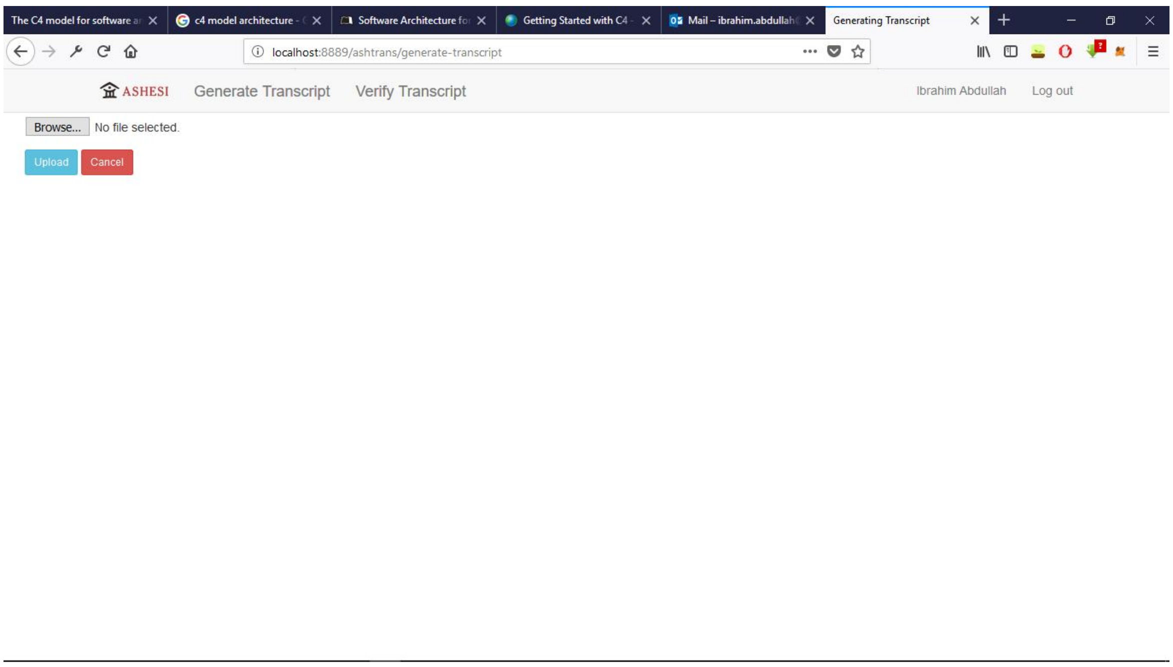


Figure 3.4: Generate transcript interface

3.2.5 Verify transcript page

Academic registry personnel use this page to verify the authenticity of transcripts. The page has a file picker button for selecting a file from the computer on which the web application was accessed. Only PDF documents and image files can be selected using the system. The upload button is used to upload the selected file on to the server for verification. The cancel button stops file upload and hence, the process of verifying the authenticity of the transcript. Figure 3.5 is the interface for the verify transcript page.

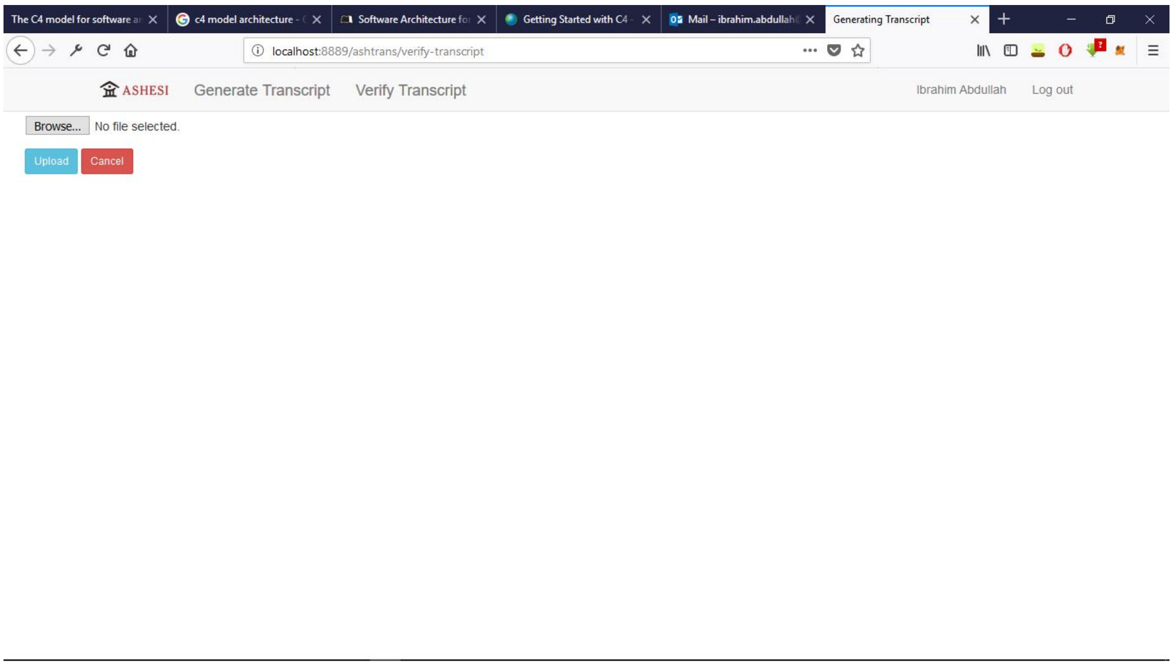


Figure 3.5: Verify transcript interface

3.3 System Architecture

The system architecture is designed using the C4 model, a widely used industry technique for describing and communicating the architecture of software systems intended to have a modular structure (Brown, n.d.). The technique provides a way for software developers to communicate software architecture at different levels of details (Brown, n.d.). The model describes the static structure of the software system in terms of the users, software system context, containers, components, and the individual classes that make up the system (Minutillo, 2015). Hence, the C4 model has four well-defined layers or levels of abstractions for describing the architecture of the software, each telling different stories about the architecture of the system to different types of audience. The levels of abstractions are software system, container, component and class. Each level of abstraction is further explained in the following sub-sections including diagrams that describe the architecture of the Ashesi Transcript System.

3.3.1 Software system

Software system is the first level of abstraction in the C4 model for describing software architecture. It describes the actual software that delivers value to the users. Hence, it describes the software being modelled and other software systems it depends on. System context diagram is used to describe the software system level of abstraction. It depicts the system being modelled as a box in the centre, surrounded by users and other software systems it interacts with. Figure 3.6 is the system context diagram of the Ashesi Transcript System. It interacts with Microsoft Office 365 API to use Ashesi Office 365 Directory for authentication.

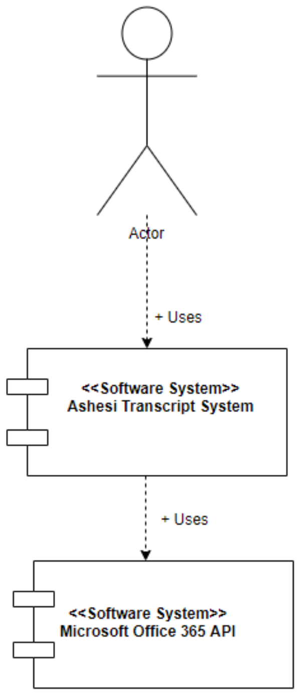


Figure 3.6: Software context diagram for the Ashesi Transcript System

3.3.2 Container

Container is the second level of abstraction in the C4 model. A container represents a separately runnable or deployable unit that host code or data and needs to be running for the entire software system to work. It shows high-level technology choices used to implement the software system, how the system is decomposed into different containers and how those containers communicate with each other (Minutillo, 2015). Container level of abstraction also shows how responsibilities are distributed across the containers that make up the system.

The Ashesi Transcript System comprises the following containers; a server-side Java /Spring model view controller (MVC) web application running on Apache Tomcat server, a client-side web application running in the browser using Thymeleaf template engine and windows file system. The Java/Spring MVC application delivers content to the client-side web application to display in the browser for the application users. The Java/Spring MVC web application also read and store data using windows file system. These containers together form the complete software system. Figure 3.7 is the container diagram for the Ashesi Transcript System.

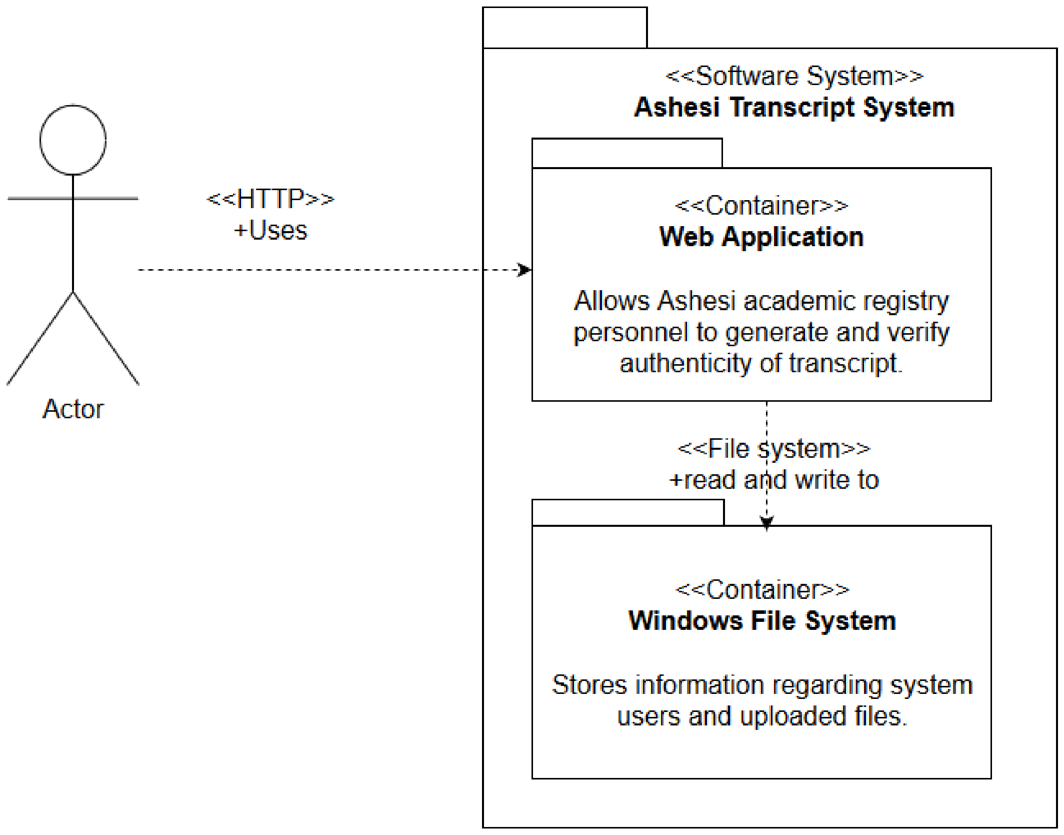


Figure 3.7 Container diagram for the Ashesi Transcript System

3.3.3 Component

Component is the third level of abstraction in the C4 model which shows the internal structure of the containers that make the overall software system and how the containers are decomposed into collaborating components (Minutillo, 2015). A component is a group of related functionalities; a collection of implementation classes behind one interface (Brown, n.d.). Components diagrams are used to describe the software architecture at this level.

Figure 3.8 is the component diagram of the Ashesi Transcript System. The users interact with the web application through HTTP requests to the Spring MVC controllers which serve as entry points the web application. The controllers use the Spring service components to execute user requests. There are two Spring service components in the web application; user service and ashtrans-service. The user service component provides functionality regarding user account management such as adding a new user and reading details of users from the file system. The ashtrans Spring service component provides system focussed functionality regarding transcript generation and verification. It depends on the encryption and decryption module to provide services related to text encryption and decryption. It also depends on the QR code module to provide services related to generating and reading QR code and embedding QR code on the transcript. User data is stored in files on the server.

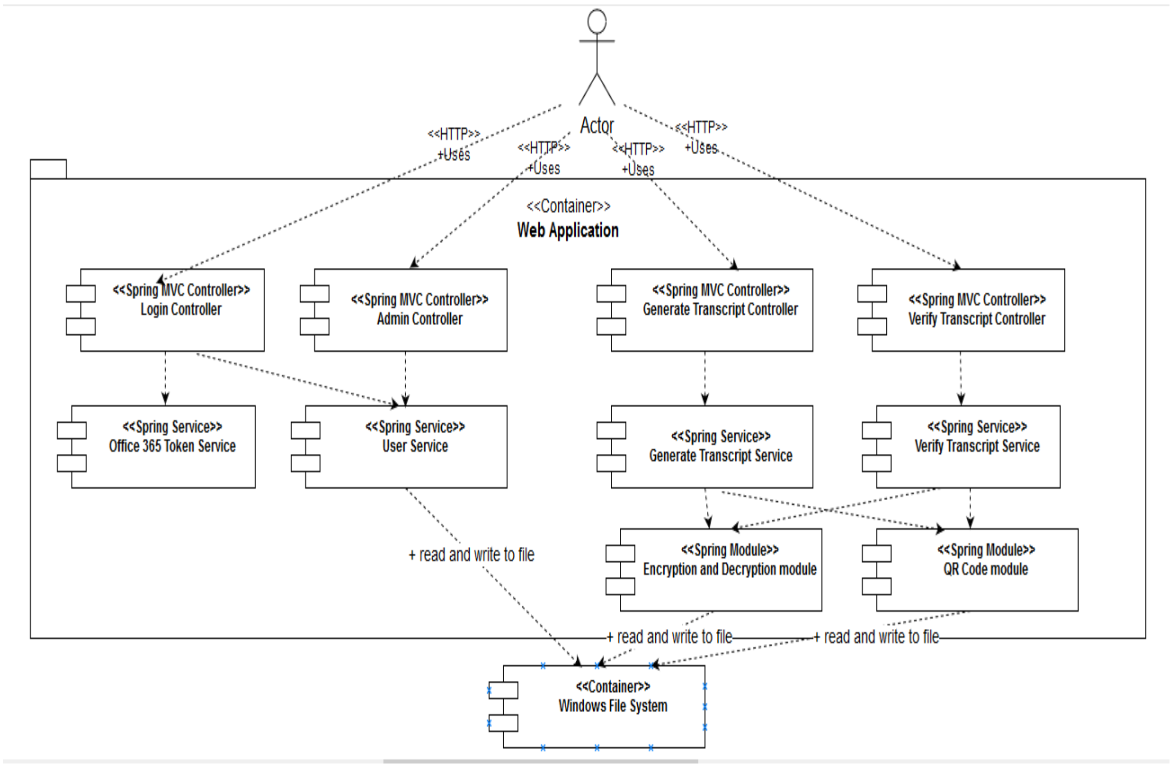

Actor
<<HTTP>> +Uses
<<HTTP>> +Uses
<<HTTP>> +Uses
<<HTTP>> +Uses
<<Container>> Web Application
<<Spring MVC Controller>> Login Controller
<<Spring MVC Controller>> Admin Controller
<<Spring MVC Controller>> Generate Transcript Controller
<<Spring MVC Controller>> Verify Transcript Controller
<<Spring Service>> Office 365 Token Service
<<Spring Service>> User Service
<<Spring Service>> Generate Transcript Service
<<Spring Service>> Verify Transcript Service
+ read and write to file
<<Spring Module>> Encryption and Decryption module
<<Spring Module>> QR Code module
+ read and write to file
+ read and write to file
<<Container>> Windows File System

Figure 3.8: Component diagram of Ashesi Transcript System.

3.4 System Logical View

In this section, sequence diagrams are used to model interactions between the users and the software system. It also shows interactions between system components and containers. The system interaction modelling is done for the following main functionalities of the Ashesi Transcript System; generating and verifying transcript.

3.4.1 Sequence diagram for generating transcripts

The sequence diagram below models the interactions between academic registry personnel and the Ashesi Transcript System. It also models interactions between the system's components and containers when generating transcripts. The interaction is designed based on the assumption that the academic registry personnel has logged in to the system.

On the generate transcript page, the academic registry personnel select a PDF document and uploads on to the server by clicking on the upload button. The uploaded file is sent to the generate transcript controller through Hypertext Transfer Protocol (HTTP) request. The controller sends the file to the generate transcript service to save it on the server and returns the path to the file for use. The generateTranscript method is invoked to use the uploaded file to a generate transcript. The system reads the encryption key from file, computes the hash code of the information on the uploaded file, encrypt the hash code and generates QR code which stores the encrypted hash code. The QR code is embedded on the generated transcript which the system displays in the browser.

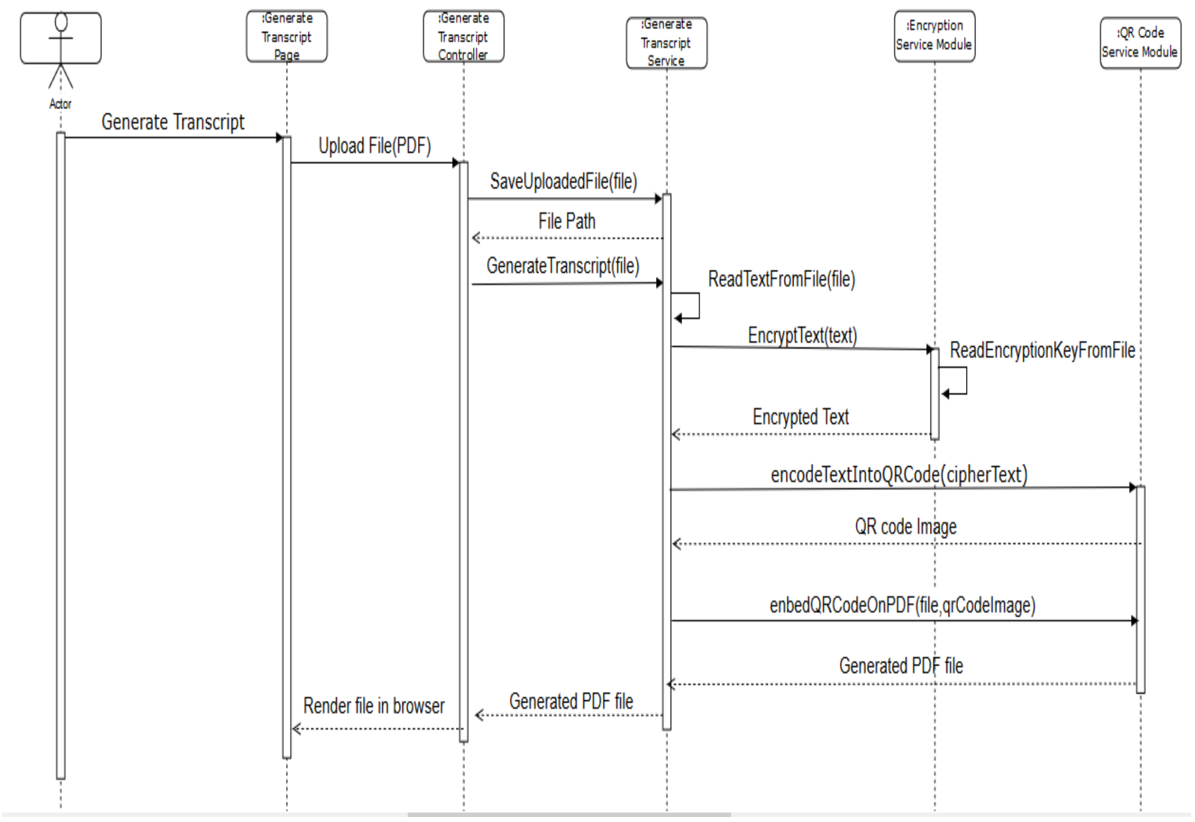
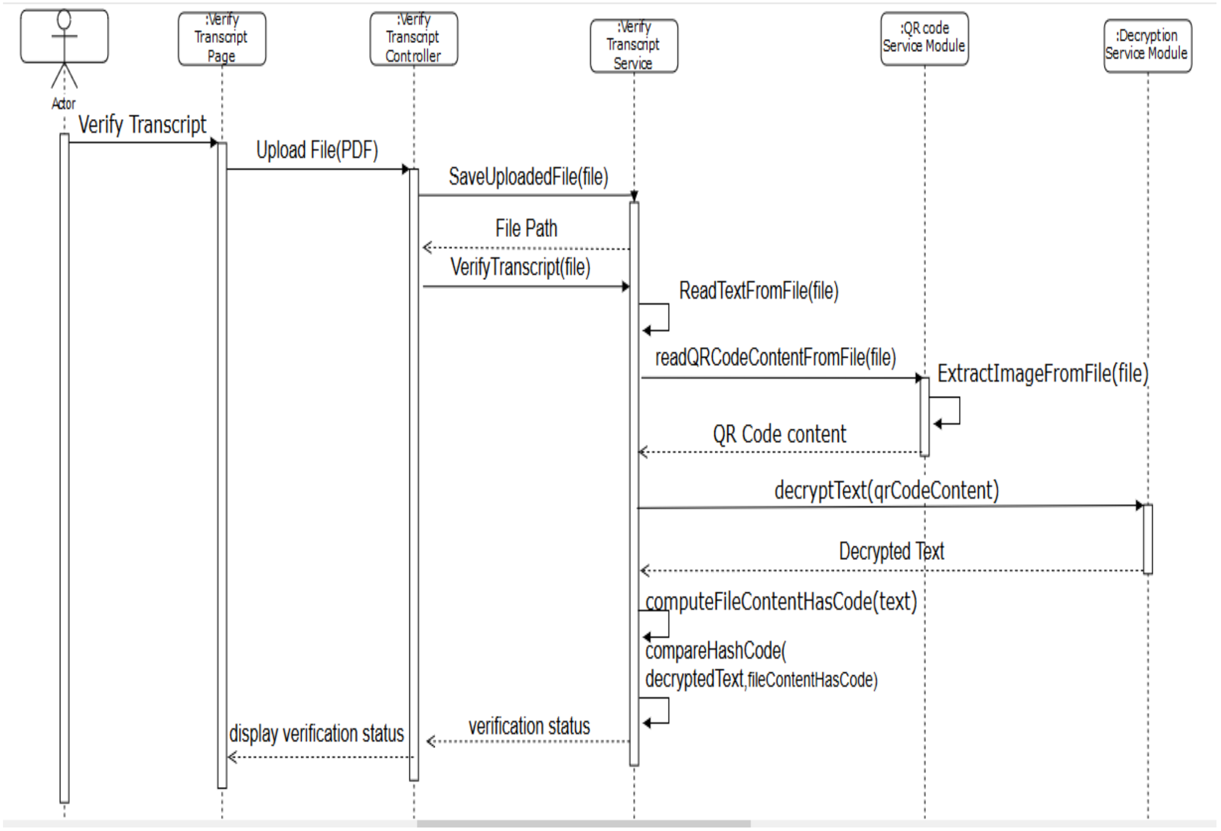


Figure 3.9: Sequence diagram for generating transcripts

3.4.2 Sequence diagram for verifying transcripts

The sequence diagram below models the interactions between academic registry personnel and the Ashesi Transcript System. It also models interactions between the system's components and containers when verifying the authenticity of transcripts. The interaction is designed based on the assumption that the academic registry personnel has logged in to the system.

On the verify transcript page, the academic registry personnel select a PDF transcript and upload it on to the server by clicking on the upload button. The uploaded file is sent to the verify transcript controller through an HTTP request. The controller sends the file to the verify transcript service to save it on the server and returns the path to the file for use. The system read text on the uploaded file and the content of the QR code embedded on the file. The system then decrypts the content of the QR code and compare the hash code of the information on the uploaded file to the decrypted hash code which was stored in the QR code.



Actor
:Verify Transcript Page
:Verify Transcript Controller
:Verify Transcript Service
:QR code Service Module
:Decryption Service Module
Verify Transcript
Upload File(PDF)
SaveUploadedFile(file)
File Path
VerifyTranscript(file)
ReadTextFromFile(file)
readQRCodeContentFromFile(file)
ExtractImageFromFile(file)
QR Code content
decryptText(qrCodeContent)
Decrypted Text
computeFileContentHasCode(text)
compareHashCode(decryptedText,fileContentHasCode)
verification status
display verification status

Figure 3.10: Sequence diagram for verifying transcripts

Chapter 4: Implementation

4.1 System implementation tools, libraries and API's

In this subsection of chapter 4, tools, libraries, and APIs used for the development of the Ashesi Transcript System are discussed. These include Java, Spring MVC framework, Thymeleaf, Itext, PDFBox, Tess4j, and ZXing. For each of these technologies, the rationale for choosing it for this project is discussed.

4.1.1 Java

The system was developed using Java programming language. Java is a multi-purpose programming language. This means Java can be used to develop web applications, standalone console application and many more. Hence, the choice of using Java was to ensure that almost all components of the application are developed using one programming language. Additionally, Java has APIs with a collection of classes that efficiently implements standard encryption and decryption algorithms. Some of these security APIs are used for text encryption and decryption in the system.

4.1.2 Spring MVC framework

Spring framework is a technology that provides comprehensive programming and configuration model for a Java-based enterprise application that can be deployed on any deployment platform (Pivotal Software Inc., 2018). Spring MVC is a servlet-based web module in Spring framework for developing web application based on the standard Model View Controller architecture for developing web applications (Pivotal Software, Inc., 2018). The reason for choosing Spring MVC over other Java frameworks such as play (Lightbend, 2018) is due to its suitability for the modular system architecture of the Ashesi Transcript System. Spring framework can also be used to develop standalone applications which could be used as dependencies in the web application. Additionally, Spring Framework provides convenient means of managing application dependencies. Libraries added as dependencies in the Project Object Model file are automatically downloaded and included in the application.

4.1.3 Thymeleaf

Thymeleaf is a Java template engine for developing server-side web pages for both web and standalone environments (The Thymeleaf Team, 2017). Thymeleaf has modules that allow integration with Spring framework and hence enable the use of HTML, CSS, and JavaScript to develop web pages in Spring MVC applications. Thymeleaf, unlike Java Server Pages, allows the use of static content to build web page prototypes for testing purposes without deploying the application on a server. However, when the application is deployed, the static contents are replaced with contents generated from the server-side.

4.1.4 Itext, Apache PDFBox and ZXing libraries

Itext (iText Group NV, 2018) and Apache PDFBox (The Apache Software Foundation, 2009) are open source libraries that provide interfaces for creating, manipulating, and extracting text and images from PDF documents. Itext also provides interfaces for generating QR codes. In the Ashesi Transcript System, Itext is used to create new PDF documents, extract text on PDF documents, generate QR code and embed it on PDF documents. Itext library, however, does not support extracting images from PDF files. Hence, the Apache PDFBox library is used to extract QR code image from PDF transcripts during verification. ZXing (ZXing, 2018) library is used to read QR code content from the QR code image extracted from the transcript.

4.1.5 Tess4j

Tess4j is a Java Native Access (JNA) wrapper for Tesseract Optical Character Recognition (Tess4J, 2017). The library does optical character recognition for TIFF, JPEG, GIF, PNG, BMP image formats, Multi-page TIFF images, PDF document format (Tess4J, 2017).

4.1.6 Tomcat server

The Apache Tomcat software is an open source implementation of the Java Servlet, Java Server Pages, Java Expression Language and Java Web Socket technologies. Hence, Tomcat is a software that is used to host Java web applications.

4.2 System security

In this sub-section of chapter 4, further discussion on system implementation choices such as encryption schemes, encryption and decryption algorithms, encryption key generator algorithms and hash codes is documented. The rationale for making each implementation decisions are also discussed.

4.2.1 Hashing of transcript information

In the Ashesi Transcript System, hashing is used as the first level of securing the information stored in the QR code embedded on transcripts. In the process of generating transcripts, a hash code of the information on the transcript is computed and encrypted. This is done because the QR code embedded on transcripts could not store the encrypted student transcript information. The size of the resulting ciphertext is larger than the maximum amount QR code stores. Since hash codes are usually shorter in length than the original text and hash codes generated by the same hash function are of fixed length, encrypting the hash code of the information on the transcript reduces the size of information stored in the QR code embedded on the transcript. Hence, encrypting the hash code of the information on the transcript solves the issue with limited QR code storage space. Hashing also serves as a second level of security in detecting changes on transcripts. When the Ashesi Transcript System is verifying transcripts, a hash code of the information on the transcript is computed and compared with the decrypted hash code read from the QR code embedded on the transcript.

Java in-built Secure Hash Algorithm (SHA-256) is used to compute hash code of information on transcripts. The SHA hash functions are designed by the United State National Security Agency (NSA), and they are globally accepted as standard algorithms (mkyong, 2010). SHA-2, a family of the SHA hash functions to which SHA-256 belongs, are secure hashing algorithms (Oracle, 2018). Hence, SHA-256 was chosen over chosen over Message Digest algorithm version 5 (MD5) (Ragab, Nabil, & Osama, 2001) and other algorithms to hash information on transcripts in the Ashesi Transcript System. Figure 4.1 is the code snippet used to compute hash code of text String.

```
public static byte [] computeHashCode (String text) throws
NoSuchAlgorithmException {
        MessageDigest digest = MessageDigest.getInstance("SHA-256");
        return digest.digest(text.getBytes(StandardCharsets.UTF_8));
    }
```

Figure 4.1 Code snippet for computing hash code of text

4.2.2 Private Key encryption scheme

Private Key encryption is an encryption scheme where the same secret or private key is used for both encryption and decryption. Anyone who gains access to the encryption key can decrypt the encrypted text and hence, the encryption key must be protected from the reach of unauthorized people. This makes key management a major issue for private key or symmetric encryption. In the Ashesi Transcript System, private key encryption is used to encrypt and decrypt hash codes of the information on transcripts. The choice of private key encryption over public key encryption was because only the academic registry personnel will be using the system to generate and verify the authenticity of transcripts. Public key encryption is suitable for systems that enable communication with unknown clients and that their identity must be verified. However, the Ashesi Transcript System authenticate all users before they are given access to use the system. Advanced Encryption Standard algorithm is used for encryption and decryption in the system because it is the current industry standard.

4.2.3 Advanced Encryption Standard algorithm.

The Advanced Encryption Standard (AES) is a private or a symmetric key algorithm which uses a single key for both encryption and decryption. In November 2001, AES was accepted as a standard block cipher to replace Data Encryption Standard (DES) algorithm by the National Institute of Standard and Technology (Iguchi, Sasao, & Qin, 2006). AES has been used in many security-related applications such as Microsoft BitLocker Drive Encryption, Skype, Secure Socket Layer and Transport Layer Security (Isa, Bahari, Sufian, & Z'aba, 2012).

4.2.4 Encryption key generation and storage

The Ashesi Transcript System uses Java's key generator class to generate 128-bit long encryption key. The key generator class uses AES provider algorithm to generate the encryption key. No random seed is provided to the key generator algorithm. This allows Java cryptography library to use the highest priority Secure Random implementation installed for generating encryption keys for AES encryption, which is a better option as compared to choosing a random number as a seed which might not be good enough.

The encryption key is generated once when the application is deployed. It is then stored in a file as plain text in a different folder on the server on which the application is deployed. Access to the folder in which the key file is stored is restricted to only the system administrator. Also, to prevent unauthorised users from being able to use the key stored in the key file, the Secret Key object generated by the key generator class is modified in several ways before it is stored as a hexadecimal String in the key file. When the key is generated, the Secret Key object is encoded into an array of bytes which is also encoded into hexadecimal character array before converted to String and stored in the key file. The reverse process is used to turn the key String read from the key file to an instance of the Secret Key object so that it can be used for encryption and decryption.

The encryption key is stored in a file to prevent the academic registry personnel from entering it when generating or verifying transcripts. This prevents situations where a former academic registry personnel will have access to the encryption key. Usually, storing encryption keys in files are not recommended because of the insecurities it introduces into the system. For the Ashesi Transcript System, storing the encryption key in a file after the modifications done to it raises little concern. Additionally, the keys are used to encrypt hash codes of the information on the transcript. Since hashing is one way, the student information cannot be reproduced when unauthorized users are able to decrypt the encrypted hash codes.

```
private void generateNewEncryptionKey () throws Exception {
        KeyGenerator generator = KeyGenerator.getInstance("AES");
        generator.init(128); // The AES key size in number of bits
        SecretKey secKey = generator.generateKey();

        //Store the encryption key in a Keystore file
        storeGeneratedKeyInFile(secKey);
    }
```

Figure 4.2: Snippet of code for generating the encryption key

The other alternatives for storing the encryption key are; using an external Hardware Security Module (HSM), binding the encryption key to a hardware such as Trusted Platform Module (TPM) chips, storing the key on a different server or in a database (Information Security: where to store server side encryption key?, 2012). Using HSM means storing the keys on a more secure external device which is expensive for this project. Also, binding the encryption key to hardware such as TPM chips means encryption and decryption only works on that hardware. Even though the options of using an external device to store the encryption key are the most secure approaches for this project, they are expensive.

4.3 Module Implementations

In this subsection, details of the independent Java/Spring applications or modules are discussed. These are the encryption and decryption module and the QR code module.

4.3.1 Encryption and decryption module

The encryption and decryption module is an independent Java/Spring application that uses Java security and cryptography APIs for text encryption and decryption. The module is deployed as a Java Archive (jar file) and included in the ashtrans service module as a dependency in the Project Object Module (POM) file. In the ashtrans service module, an instance of any of the classes encapsulated in the resulting jar file of the encryption and decryption module is used to access functionalities provided by the module. The web application relies on the ashtrans service module to access the functionalities of the encryption and decryption module.

The module has three classes and two interfaces which are implemented to adhere to abstraction and encapsulation principles in Object Oriented Programming. The EncryptionServiceImp class implements EncryptionServiceApi interface. EncryptionServiceApi interface specifies methods that the EncryptionServiceImp class must implement to provide encryption services. The DecryptionServiceImp class implements the EncryptionServiceApi interface. The Utility has static methods that implement functions common to both the EncryptionServiceImp and DecryptionServiceImp classes. For example, both encryption and decryption require the system to read the encryption key from file. Instead of implementing two different methods to read the encryption key from file, a static method that performs the action is implemented in the utility class for the two different classes to access. Figure 4.3 the snippet of code used to encrypt text while Figure 4.4 is a snippet of code used to decrypt text.

```
@Override
public String encryptText (String plainText) throws Exception {

if (isTimeToGenerateNewKey ()) {
      generateNewEncryptionKey ();
         }
      //Read encryption key from file
      SecretKey secretKey =
      Utility.readEncryptionKeyFromFile(keyFile);
      //Generate a Hashcode for the plaintext
      byte [] hashcode = Utility.computeHashCode(plainText);

      //Encrypt the generated Hascode
      Cipher aesCipher = Cipher.getInstance("AES");
      aesCipher.init(Cipher.ENCRYPT_MODE, secretKey);
      byte [] byteCipherText = aesCipher.doFinal(hashcode);

      //Convert the bytes array of cipher text to hexadecimal
      return DatatypeConverter.printHexBinary(byteCipherText);
}
```

Figure 4.3: Code snippet for encrypting text

```
@Override
public String decryptText (String cipherText) throws
NoSuchAlgorithmException, NoSuchPaddingException {
        try {
                //Read the encryption key from file
                SecretKey secKey =
                Utility.readEncryptionKeyFromFile(keyFile);
                Cipher aesCipher = Cipher.getInstance("AES");
                aesCipher.init(Cipher.DECRYPT_MODE, secKey);
                byte [] bytePlainText = null;
                bytePlainText =
                aesCipher.doFinal(DatatypeConverter.parseHexBinary(cipherTe
                xt));
                //Convert the byte array to string
                return DatatypeConverter.printHexBinary(bytePlainText);
            } catch (InvalidKeyException ex) {

                Logger.getLogger(DecryptionServiceImpl.class.getName()).log
                (Level.SEVERE, null, ex);
                return null;
      }
```

Figure 4.4: Code snippet for decrypting text

4.3.2 QR code module

The QR code module is an independent Java/Spring application that uses Itext, PDFBox, Tess4J and other libraries to create and manipulate PDFs. The module is also used to generate and read QR codes and embed QR code on PDF documents. The module uses Tess4J JNA wrapper library to do OCR on images and scanned PDF documents. The module is deployed as a Java Archive (jar file) and included in the ashtrans service module as a dependency. Hence, services provided by this module can be accessed in the web application through the ashtrans-service module.

The QR code module has three interfaces and three classes. The QRCode class implements the QRCodeAPI interface. The QRCode class represent QR code object and hence, has QR code properties as instance variables. The QRCodeServiceImp class implements the QRCodeServiceApi interface. The QRCodeServiceApi interface specifies methods the QRCodeServiceImp class must implements. The ImageRendererListener class implements Render Listener interface. It is used to write QR code images extracted from PDF document as files so that it can be read to extract their content. Figure 4.5, 4.6 and 4.7 are code snippets that show how QR codes are generated, embedded onto and extracted from PDF document.

```
@Override
public Qrcode generateQRCode(String text, String errorLevel) {
      //Check text size as a form of validation
      if ((text == null) || ("".equals(text)) || (text.length() <= 100))
      {
            return null;
      }
      try {
            //set properties for generating qr code
            Map<EncodeHintType, Object> qrcodeProperties =
            new HashMap<>();
            ErrorCorrectionLevel level =
            this.getQrcodeErrorCorrectionLevel(errorLevel);
             if (level == null) {
                  return null;
             }
           qrcodeProperties.put(EncodeHintType.ERROR_CORRECTION, level);
            BarcodeQRCode barcodeQRCode = new BarcodeQRCode(text, 100,
            100, qrcodeProperties);
            Image imageFile = barcodeQRCode.getImage();
            Qrcode qrcode = new Qrcode(text, imageFile);

            return qrcode;
            } catch (BadElementException bee) {
                  System.err.println("Error: " + bee.toString());
            } catch (Exception exp) {
                  System.err.println("Error: " + exp.toString());
            }
            return null;
     }
```

Figure 4.5: Code snippet for generating QR Code

```
@Override

public void embedQRCodeOnPDF(Qrcode qrcode, File pdfFile, File
destFile)throws IOException, DocumentException {

      PdfReader fileReader = new PdfReader(pdfFile.getAbsolutePath());

      //FileOutputStream fos = new FileOutputStream(destFile);

      PdfStamper pdfStamper = new PdfStamper(fileReader, new
      FileOutputStream(destFile));

      //embed image of the pdf file

      Image imageFile = Image.getInstance(qrcode.getQrCodeImageFile());

      imageFile.setAbsolutePosition(36, 400);

      int numberOfPages = fileReader.getNumberOfPages();

      PdfContentByte under = pdfStamper.getUnderContent(numberOfPages);

      under.addImage(imageFile);

      //Close all resources

      pdfStamper.close();

      fileReader.close();

}
```

Figure 4.6: Code snippet for embedding QR code on generated PDF transcript

```
private String extractQRcodeImage (File file) throws IOException,
DocumentException {
      //Get filename withoyt extension
      String filenameWithOutExtension =
      getFileNameWithoutExt(file.getName());
      //Generate PDF reader
      PdfReader reader = new PdfReader(file.getAbsolutePath());
      PdfReaderContentParser parser = new
      PdfReaderContentParser(reader);


      ImageRendererListener listener = new ImageRendererListener
      (imageDestinationFolder+ "/" +filenameWithOutExtension +
      "%s.%s");


      //Last page number
      int lastPageNumber = reader.getNumberOfPages();
      parser.processContent(lastPageNumber, listener);
      reader.close();


      //Return the path to the image file
      return imageDestinationFolder + "/" + filenameWithOutExtension +
      lastPageNumber + ".png";
}
```

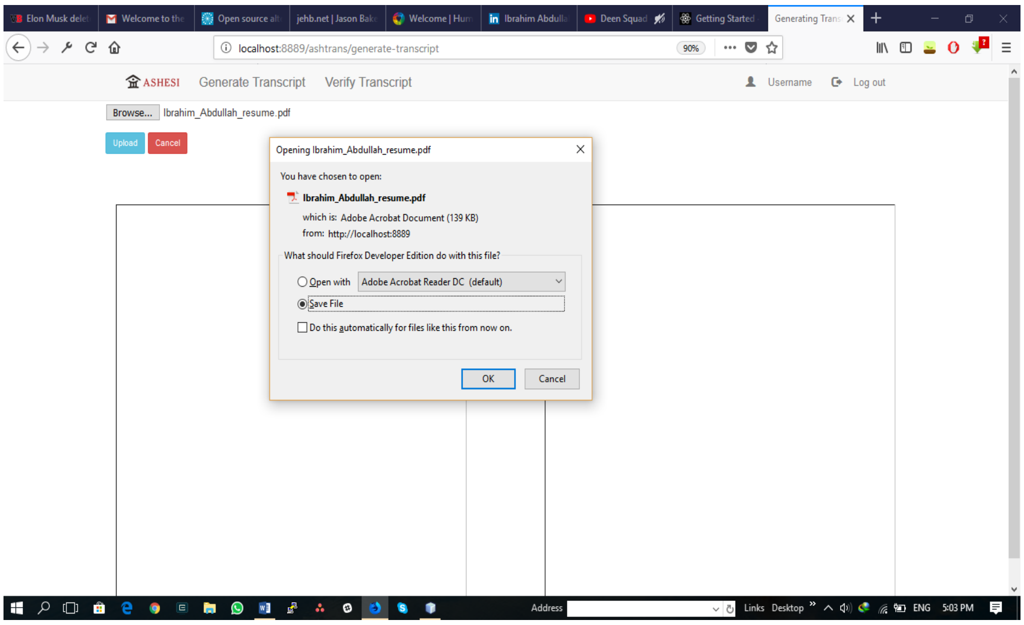
Figure 4.7: Code snippet for extracting QR code image from PDF transcript

4.4 Implementation of core functionalities

This section of the report documents implementation details of the core functionalities of the Ashesi Transcript System namely, transcript generation, transcript verification and Optical Character Recognition.

4.4.1 Transcript generation

The system can be used to generate transcripts with QR code embedded on it. The generated PDF transcript is displayed in the browser for the academic registry personnel to save on a computer. Figure 4.8 is a sample PDF document rendered in the browser for the user to save on the computer on which the application was accessed.



localhost:8889/ashtrans/generate-transcript

Ibrahim_Abdullah_resume.pdf

Opening Ibrahim_Abdullah_resume.pdf

You have chosen to open:

Ibrahim_Abdullah_resume.pdf

which is: Adobe Acrobat Document (139 KB)
from: http://localhost:8889

What should Firefox Developer Edition do with this file?

( ) Open with Adobe Acrobat Reader DC (default)
(●) Save File

☐ Do this automatically for files like this from now on.

Figure 4.8: A sample generated PDF document rendered in the browser for download

4.4.2 Transcript verification

The Ashesi Transcript System verifies transcripts by reading the content of the QR code image embedded on the transcript. It checks if the content of the QR code is a hexadecimal String. This is because the encrypted hash codes of the information on transcripts generated with the Ashesi Transcript System are converted into hexadecimal String before they are stored in the QR code embedded on the transcript. If the content of the QR code is not a hexadecimal String, then the transcript was not generated with the Ashesi Transcript System. As a result, the system halts the verification process and declare the transcript to be incorrect. Otherwise, the system continues with the verification process. The systems decrypt the encrypted hash code read from the QR code image embedded on the transcript. The decrypted hash code is the hash code of the information on the transcript when it was generated. The system also read and compute hash codes of the information on the transcript to be verified. If changes have been made to the transcript after it was issued, then the hash code of the information on the transcript to be verified will be different from the hash code decrypted from the QR code image embedded on the transcript. Hence, the system compares the two hash codes to verify the authenticity of the transcript. If there is a mismatch, then the transcript is considered incorrect and hence, further verification must be done by the academic registry personnel.

4.5 Optical Character Recognition (OCR)

Optical Character Recognition technique is used in the Ashesi Transcript System to read text on scanned PDF documents. If the PDF is scanned, Itext is not able to read the text on it and hence return an empty String. In such cases, the system tries to do OCR on the PDF document. The system uses Tess4J Java Native Access wrapper for Tesseract OCR library to do OCR. The library is trained to recognize English letters using test data retrieved from Git Hub repository of the Tesseract project. The accuracy rate of doing OCR on scanned PDFs is less than 100 percent. The system reads the Ashesi logo as the letter 'M' and stray marks as text. Additionally, since the information on the transcript is displayed in a tabular form, the library reads text on the same line but in different table cells as one line. This displaces the position of the characters. As a result, hash codes generated for the information on the transcript retrieved using OCR is always different from hash code stored in QR codes embedded on the transcript. Therefore, the Ashesi Transcript System verifies all scanned PDF transcripts as incorrect.

Chapter 5: Testing and Results

This section documents the testing of Ashesi Transcript System to assess the applications performance and correctness. The testing procedures used are component testing, system testing. The procedures are further elaborated with their corresponding result in the following subsections.

5.1 Component testing

The Ashesi Transcript System consists of different components or modules with each of them performing specific functions. Each of the modules was tested to ensure it works correctly. The following subsections detail the testing of the system's components and the results.

5.1.1 QR code module

The QR code module is used to generate QR code images and PDF documents. It is also used to embed QR code images on PDF transcripts and read QR code images embedded on PDF transcripts. Since the module is an independent Java application, it was tested by creating an instance of the classes it encapsulates and invoking their methods in the main class of the application. Given a dummy text String, the module successfully generates QR code image storing the text and embeds it on a PDF document it creates. Using PDF documents created using the module, the module successfully extracts and read the QR code image on the PDF. Figure 5.1 is a sample PDF document generated using the module with QR code embedded on it. Hence, the system passed the test of generating QR code and embedding it on PDF transcript. It is also passed the test of reading QR codes embedded on PDF transcripts.

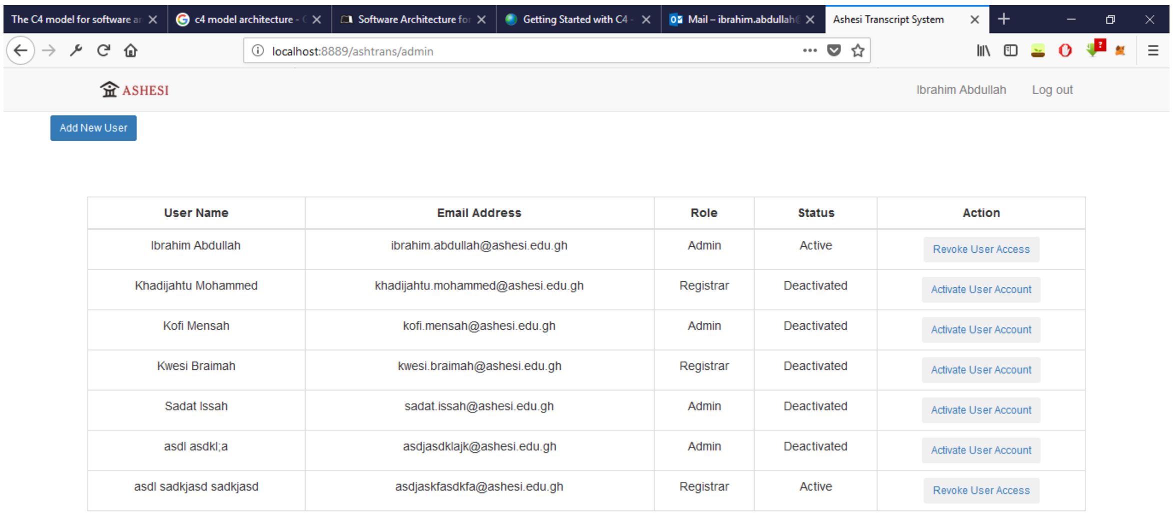
5.1.2 Encryption and Decryption module

The encryption and decryption module was tested as an independent Java application by creating an instance of the classes in the module and invoking their methods. The encryption key generated by the module was stored in a file in a directory on the host machine. Given a dummy text String, the module successfully computes and encrypts the hash code of the text using the encryption key stored in the file. Using the same encryption key, the module successfully decrypts the encrypted hash code which matches the hash code of the original text.

5.2 System testing

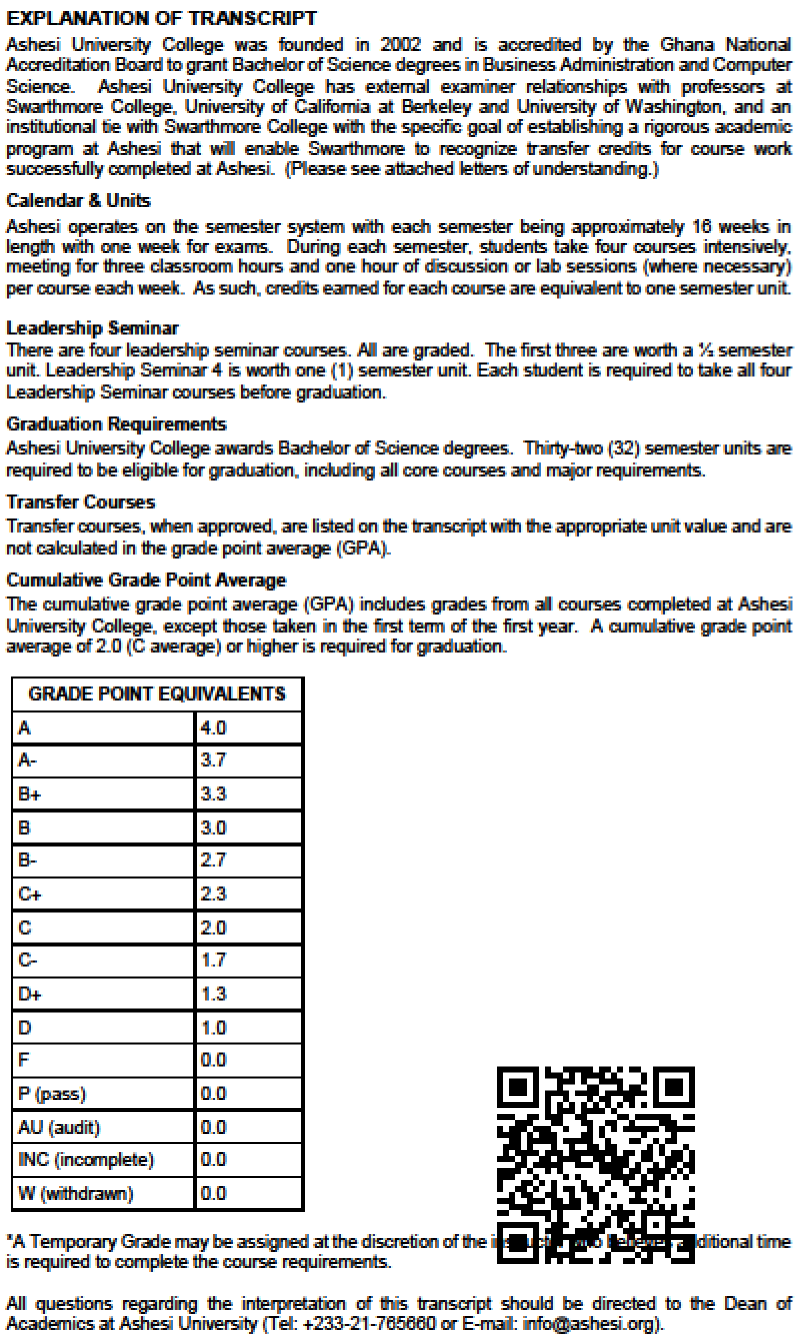
System testing procedure involves testing the complete application with all the components integrated. At this level, the system was tested as if it is in production mode. This system can successfully generate PDF transcripts which have embedded on it a QR code storing the encrypted hash code of the information the transcript. Figure 5.1 is a sample of a generated PDF transcript with QR code embedded on it. For original PDF transcripts, the system successfully reads QR codes embedded on them to verify the authenticity of the transcript. However, verification of scanned PDF's does not function correctly because of errors in doing OCR on the scanned PDFs. The OCR library is not able to read correctly text in a tabular format. It also misinterprets some of the English alphabets. For example, it read the letter 'f' as '?'. The library also read stray marks such dot and the Ashesi logo on transcripts as the letter 'M'.

Additionally, system administrators can successfully add new users to the system and revoke access of existing users. Users added to the system can log in to the system using their Ashesi Office 365 credentials. Figure 5.1 shows a list of users added to the system by the system administrator. In figure 5.1, the user currently logged in to the system has the username Ibrahim Abdullah displayed on the top right corner of the page. Additionally, the buttons in the action column displays ‘Activate user account’ for users whose access to the system has been revoked.



User Name	Email Address	Role	Status	Action
Ibrahim Abdullah	ibrahim.abdullah@ashesi.edu.gh	Admin	Active	Revoke User Access
Khadijahtu Mohammed	khadijahtu.mohammed@ashesi.edu.gh	Registrar	Deactivated	Activate User Account
Kofi Mensah	kofi.mensah@ashesi.edu.gh	Admin	Deactivated	Activate User Account
Kwesi Braimah	kwesi.braimah@ashesi.edu.gh	Registrar	Deactivated	Activate User Account
Sadat Issah	sadat.issah@ashesi.edu.gh	Admin	Deactivated	Activate User Account
asdl asdkl;a	asdjasdklajk@ashesi.edu.gh	Admin	Deactivated	Activate User Account
asdl sadkjasd sadkjasd	asdjaskfasdkfa@ashesi.edu.gh	Registrar	Active	Revoke User Access

Figure 5.1: User management interface showing list of users added to the system



EXPLANATION OF TRANSCRIPT

Ashesi University College was founded in 2002 and is accredited by the Ghana National Accreditation Board to grant Bachelor of Science degrees in Business Administration and Computer Science. Ashesi University College has external examiner relationships with professors at Swarthmore College, University of California at Berkeley and University of Washington, and an institutional tie with Swarthmore College with the specific goal of establishing a rigorous academic program at Ashesi that will enable Swarthmore to recognize transfer credits for course work successfully completed at Ashesi. (Please see attached letters of understanding.)

Calendar & Units

Ashesi operates on the semester system with each semester being approximately 16 weeks in length with one week for exams. During each semester, students take four courses intensively, meeting for three classroom hours and one hour of discussion or lab sessions (where necessary) per course each week. As such, credits earned for each course are equivalent to one semester unit.

Leadership Seminar

There are four leadership seminar courses. All are graded. The first three are worth a ½ semester unit. Leadership Seminar 4 is worth one (1) semester unit. Each student is required to take all four Leadership Seminar courses before graduation.

Graduation Requirements

Ashesi University College awards Bachelor of Science degrees. Thirty-two (32) semester units are required to be eligible for graduation, including all core courses and major requirements.

Transfer Courses

Transfer courses, when approved, are listed on the transcript with the appropriate unit value and are not calculated in the grade point average (GPA).

Cumulative Grade Point Average

The cumulative grade point average (GPA) includes grades from all courses completed at Ashesi University College, except those taken in the first term of the first year. A cumulative grade point average of 2.0 (C average) or higher is required for graduation.

GRADE POINT EQUIVALENTS	
A	4.0
A-	3.7
B+	3.3
B	3.0
B-	2.7
C+	2.3
C	2.0
C-	1.7
D+	1.3
D	1.0
F	0.0
P (pass)	0.0
AU (audit)	0.0
INC (incomplete)	0.0
W (withdrawn)	0.0

*A Temporary Grade may be assigned at the discretion of the i[illegible]ditional time is required to complete the course requirements.

All questions regarding the interpretation of this transcript should be directed to the Dean of Academics at Ashesi University (Tel: +233-21-765660 or E-mail: info@ashesi.org).

Figure 5.2: Generated PDF transcript with QR code embedded on it.

5.3 Summary of implemented requirements and test results

This section provides a summary of the functional requirements of the system. For each requirement, it is indicated whether it has been implemented and whether it passed testing.

Table 5.1: System requirement implemented and tested

Requirement	Description	Implementation Status	Tested
Text encryption	The system encrypts hash code of information on PDF a transcript.	Implemented	Test passed
Generate QR code and embed on PDF	The system generates QR code storing encrypted hash code of information on the transcript.	Implemented	Test passed
Generate Transcript	The system generates transcripts with QR code embedded on it.	Implemented	Test passed
Verify original PDF Transcript	The system verifies correctly the authenticity of transcript original PDF transcript	Implemented	Test passed
Verify scanned PDF transcript.	The system verifies correctly the authenticity of scanned PDF transcript	Implemented	Test failed. OCR error in reading text on scanned PDF result in hash code different from the actual hash code of the information on the transcript.
Add new users to the system.	System administrator add	Implemented	Test passed

	new users to the system		
Revoke user access	System administrator revoke existing users' access to the system	Implemented	Test passed
User login using Ashesi Office 365 credentials	Users log in to the system using their Ashesi Office 365 credentials	Implemented	Test passed

Chapter 6: Conclusion and Recommendations

This project set out to develop a software system to automate the process used by the academic registry to verify transcripts at Ashesi University College. The objective was to reduce the possibilities of the academic registry approving incorrect transcripts as genuine. The developed system uses QR code and encryption techniques to generate transcripts that makes it possible for automatic verification. In the following sub-sections, limitations of the developed system and recommendations for future work are discussed.

6.1 Limitations

The Ashesi Transcript System does not operate well when the transcript to be verified is a scanned PDF due to Optical Character Recognition errors. The OCR library is not able to read correctly text in a tabular format. It also misinterprets some of the English alphabets. For example, it reads the letter ‘f’ as ‘?’. The library also reads stray marks such as dot and the Ashesi logo on the transcripts as the letter ‘M’. The OCR errors are partly due to limited training data to perfect the predictions of the library. Since some of the characters were not interpreted correctly, computed hash codes of the information on transcripts to be verified are always different from the decrypted hash code from the QR code image embedded on the transcript. As a result, even correct transcripts are considered incorrect due to OCR errors.

Also, the encryption key is stored as plain text on the host computer on which the application runs. Even though the file has restricted access to only the admin user of the host computer, it is still not secure. Additionally, the keys will be lost if the host computer crashes.

Moreover, the system does not support user authentication using Ashesi LDAP server. Since most of the software systems deployed by the university use the LDAP server to provide alternative login options for users, it would have been better for the Ashesi Transcript System to have that option as well.

6.2 Future Work

Future work on this project should seek to address the limitations stated above. More training data will have to be provided to the OCR library to perfect its readings from scanned PDFs. Otherwise, a different approach that could solve the problems introduced by Optical Character Recognition such as Computer Vision could be explored.

Additionally, a more secure approach for storing the encryption key should be explored. The encryption key is stored as a plain text in a file on the computer on which the application is deployed. The key can be lost if the server crashes. Hence, alternative ways of storing the encryption key such as using Hardware Security Module, binding the key to a hardware such as TPM chips or storing the key on different servers could be explored. Also, the system uses one key for encryption. This introduces vulnerabilities into the system as patterns in the ciphertext could be explored to predict the key. Future works should consider generating new keys for encryption after a specified period.

Also, future works on the system should retrieve transcripts information directly from the Ashesi Student Management System. This will make the system more usable and save more time as compared to the academic registry personnel generating transcript information from a different system and uploading the PDF file on the Ashesi Transcript System.

6.3 Conclusions

The success of this project shows that the process for verifying transcripts can be improved and made error-free with technology. The problem of academic institutions mistakenly approving incorrect transcripts as genuine can be solved with an improved version of this system. Hence, this project has served as a proof of concept that the process of verifying student academic transcripts can be automated with encryption techniques and QR code technology.

REFERENCES

Adamchik, V. S. (2009). *Concept of Hashing*. Retrieved from Carnegie Mellon University School of Computer Science: https://www.cs.cmu.edu/~adamchik/15-121/lectures/Hashing/hashing.html

Ashesi University. (2018). Retrieved from Ashesi University College: http://www.ashesi.edu.gh/student-life-5/career-services-4/for-alumni.html

Brown, S. (n.d.). *Background and History:The C4 model for software architecture*. Retrieved from The C4 model for software architecture: https://c4model.com/

Chatterjee, T., Das, T., Dey, S., Nath, A., & Nath, J. (2011). Symmetric key cryptosystem using combined cryptographic algorithms - Generalized modified Vernam Cipher method, MSA method and NJJSAA method: TTJSA algorithm. In 2011 World Congress on Information and Communication Technologies (pp. 1175–1180). https://doi.org/10.1109/WICT.2011.6141415

Dey, S., Agarwal, S., & Nath, A. (2013). Confidential encrypted data hiding and retrieval using QR authentication system. In 2013 International Conference on Communication Systems and Network Technologies (pp. 512–517). https://doi.org/10.1109/CSNT.2013.112

Dey, S. (2013). A new generation of digital academic-transcripts using encrypted QR code ^{TM}: Use of encrypted QR code ^{TM} in mark-sheets (academic transcripts). In *2013 International Multi-Conference on Automation, Computing, Communication, Control and Compressed Sensing (iMac4s)* (pp. 313–317). https://doi.org/10.1109/iMac4s.2013.6526429

Diffie, W., & Hellman, M. E. (1976). Multiuser cryptographic techniques. *AFIPS* (pp. 7-10). New York: ACM New York. doi:10.1145/1499799.1499815

Guhr, D. (2012, February 12). Fraud in international education-The tip of the iceberg? *University World News* (208). Retrieved from http://www.universityworldnews.com/article.php?story=20120210094015109

Iguchi, Y., Sasao, T., & Qin, H. (2006). A Design of AES Encryption Circuit with 128-bit Keys Using Look-Up Table Ring on FPGA. *IEICE Transactions on Information and Systems, E89-D*(3). doi:10.1093

Information Security: where to store server side encryption key? (2012, March 1). Retrieved from StackExchange: https://security.stackexchange.com/questions/12332/where-to-store-a-server-side-encryption-key

Isa, H., Bahari, I., Sufian, H., & Z'aba, M. R. (2012). AES: Current security and efficiency analysis of its alternatives. *Journal of Information Assurance and Security, 7*, 52-60.

iText Group NV. (2018). *Itext Developers*. Retrieved from Itext.

Kapsalis, I. (2013). Security of QR Codes. Trondheim, Norway: Institutt for telematikk.

Lightbend. (2018). *Play:High velocity web framework*. Retrieved from Lightbend: https://www.lightbend.com/play-framework

Minutillo, M. (2015, January 29). *Getting Started with C4 - History: Coder Mike*. Retrieved from Coder Mike: http://codermike.com/starting-c4

mkyong. (2010, February 24). *Java SHA Hashing Example*. Retrieved from Mkyong.com: https://www.mkyong.com/java/java-sha-hashing-example/

Mullen, J., & Fiegerman, S. (2017, October 4). Yahoo tops the list of largest-ever data breaches - Oct. 4, 2017. Retrieved October 29, 2017, from http://money.cnn.com/2017/10/04/technology/yahoo-biggest-data-breaches-ever/index.html

Oracle. (2018). *Java™ Cryptography Architecture Standard Algorithm Name Documentation*. Retrieved from Oracle Java SE Documentation: https://docs.oracle.com/javase/7/docs/technotes/guides/security/StandardNames.html#KeyGenerator

Pivotal Software Inc. (2018). *Spring Framework Overview*. Retrieved from Spring Framework Documentation.

Pivotal Software, Inc. (2018). *Web on Servlet Stack*. Retrieved from Spring Framework Documentation: https://docs.spring.io/spring/docs/current/spring-framework-reference/web.html

Pornin, T. (2013, February 3). *Information Security: What are the differences between MD5, SHA and RSA*. Retrieved from StackExchange: https://security.stackexchange.com/questions/2298/what-are-the-differences-between-md5-sha-and-rsa

Pornin, T. (2013, March 2). *Information Security:The Theory*. Retrieved from StackExchange: https://security.stackexchange.com/questions/211/how-to-securely-hash-passwords/31846#31846

Ragab, A. H., Nabil, I., & Osama, A. (2001). An efficient message digest algorithm (MD) for data security. *TENCON 2001. Proceedings of IEEE Region 10 International Conference on Electrical and Electronic Technology. 1.* Singapour: IEEE. doi:10.1109/TENCON.2001.949578

Singh, U., & Garg, U. (2013). An ASCII value based text data encryption. *International Journal of Scientific and Research Publications, 3*(11), 5. Retrieved from https://s3.amazonaws.com/academia.edu.documents/37602567/ijsrp-p2397.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1523465140&Signature=lVmjxA3K06iJ8pPWJ2jXPxXREjg%3D&response-content-disposition=inline%3B%20filename%3DAn_ASCII_value_based_text_data_enc

Tess4J. (2017). Retrieved from Sourceforge: http://tess4j.sourceforge.net/

The Apache Software Foundation. (2009). Retrieved from PDFBox: https://pdfbox.apache.org/

The Thymeleaf Team. (2017, November 5). *Home:Thymeleaf*. Retrieved from Thymeleaf: https://www.thymeleaf.org/index.html

Verma, M., & Gedam, R. (2014). Data Protection by Multi-Level Encryption Algorithm. *International Journal of Latest Technology in Engineering, Management & Applied Science, III*, 156-159. Retrieved from http://www.ijltemas.in/DigitalLibrary/Vol.3Issue5/156-159.pdf

Yahya, Z., Kamarzaman, N. S., Azizan, N., Jusoh, Z., Isa, R., Shafazand, M. Y., … Mokhtaruddin, S. Z. S. (2017). A New Academic Certificate Authentication Using Leading Edge Technology. In *Proceedings of the 2017 International Conference on E-commerce, E-Business and E-Government* (pp. 82–85). New York, NY, USA: ACM. https://doi.org/10.1145/3108421.3108428.

ZXing. (2018). *ZXing ("Zebra Crossing") barcode scanning library for Java, Android*. Retrieved from Git Hub: https://github.com/zxing/zxing

APENDICES

7 Appendix A: Office 365 API request and user authentication class

```
/**

* @author Ibrahim-Abdullah

*/

public class Office365Authentication {

      //Setting Outlook API parameters for accessing user
      credentials

      private static final String AUTHORIZE_URL
      ="https://login.microsoftonline.com/common/oauth2/v2.0/autho
      rize";

      private static final String AUTHORITY
      ="https://login.microsoftonline.com";

      private static final String APP_ID = "7be47dc6-8bed-426d-
      988f-90dfdab8338c";

      private static final String APP_PASSWORD =
      "andIBPR91@%cszjBON426_=";


      //Change this to app url in production mode

      private static String redirectUrl = null;

      //User account properties to request access for

      private static final String[] scopes = {

         "openid",

         "offline_access",

         "profile",

         "User.Read"

    };


    /**

     * Build API request url for retrieving user account data
     * @param state

     * @param nonce

     * @return Url for making request

     */

public static String getOffice365LoginUrl(UUID state, UUID nonce){

           UriComponentsBuilder urlBuilder =
           UriComponentsBuilder.fromHttpUrl(AUTHORIZE_URL);

           urlBuilder.queryParam("client_id", APP_ID);


           urlBuilder.queryParam("redirect_uri",getOffice365APIRed
           irectUrl());

           urlBuilder.queryParam("response_type", "code
           id_token");

           urlBuilder.queryParam("scope", getApiAccessScopes());

           urlBuilder.queryParam("state", state);

           urlBuilder.queryParam("nonce", nonce);

           urlBuilder.queryParam("response_mode", "form_post");

           return urlBuilder.toUriString();

    }


      /**

      * Get the scope of user account details to request access
     in API request

     * @return String scope parameters

     */

public static String getApiAccessScopes() {

        StringBuilder sb = new StringBuilder();

        for (String scope : scopes) {

            sb.append(scope).append(" ");

        }

        return sb.toString().trim();

    }
```

```
/**

* Get the Redirect URL that Office365 API will send API response
to

* @return Redirect URL

*/

private static String getOffice365APIRedirectUrl() {

        if (redirectUrl == null) {

            try {

                loadOffice365ConfigurationFromFile();

            } catch (IOException e) {

                return null;

            }

        }

        return redirectUrl;

    }

/**

* Load Office365 API redirect url from the auth.properties file

* @throws IOException

*/

private static void loadOffice365ConfigurationFromFile() throws
IOException {

           String authenticationConfigFile = "auth.properties";


           //Read configuration file

           InputStream authConfigStream =
           Office365Authentication.class.getClassLoader().

           getResourceAsStream(authenticationConfigFile);


           //Check if the file was

           if (authConfigStream != null) {

           Properties authenticationProperties = new Properties();

            try {
                authenticationProperties.load(authConfigStream);

                redirectUrl =
                authenticationProperties.getProperty("redirectUrl
                ");

            } finally {

                authConfigStream.close();

            }

        } else {

                throw new FileNotFoundException("Property file '"
                + authenticationConfigFile + "'reading error.");

        }

    }


    /**

     * Used to exchange authentication code obtained from API
request for Token

     * to access user data

     *

     * @param authenticationCode

     * @param clientId The client ID for

     * @return Office365TokenResponse

     */

public static Office365TokenResponse getTokenFromAuthCode(String
authenticationCode, String clientId) {

      //Interceptor to log HttpServelet request and response

      HttpLoggingInterceptor interceptor = new
      HttpLoggingInterceptor();

      interceptor.setLevel(HttpLoggingInterceptor.Level.BODY);

      OkHttpClient client = new
      OkHttpClient.Builder().addInterceptor(interceptor).build();


      // Create and configure the Retrofit object

      Retrofit retrofit = new
      Retrofit.Builder().baseUrl(AUTHORITY).client(client)
```

```
  .addConverterFactory(JacksonConverterFactory.create()).build
  ();


  // Generate the token service

  Office365TokenService tokenService =
  retrofit.create(Office365TokenService.class);


  try {

        // Get Access Token from authentication Code

        return
        tokenService.getAccessTokenFromAuthCode(clientId,
        APP_ID,APP_PASSWORD, "authorization_code",
        authenticationCode,getOffice365APIRedirectUrl()).execut
        e().body();

    } catch (IOException e) {


        //Return error when token could not be accessed

        Office365TokenResponse error = new
        Office365TokenResponse();

        error.setError("IOException");

        error.setErrorDescription(e.getMessage());


        return error;

    }

}


/**

 * This method refresh API tokens after they've expired

 *

 * @param tokens Already obtained tokens to refresh

 * @param tenantId client id for making API request

 * @return Office365TokenResponse

 */
```

```
      public static Office365TokenResponse
      ensureTokens(Office365TokenResponse tokens, String tenantId)
      {

        Calendar calender = Calendar.getInstance();

        //Check if the tokens are still valid

        if (calender.getTime().before(tokens.getExpirationTime()))
{

            return tokens;

        } else {

           //Interceptor to log HttpServelet request and response

           HttpLoggingInterceptor interceptor = new
           HttpLoggingInterceptor();


      interceptor.setLevel(HttpLoggingInterceptor.Level.BODY);

           OkHttpClient client = new
           OkHttpClient.Builder().addInterceptor(interceptor).buil
           d();

           // Create and configure the Retrofit object

           Retrofit retrofit = new
           Retrofit.Builder().baseUrl(AUTHORITY).client(client)
           .addConverterFactory(JacksonConverterFactory.create()).
           build();

           // Generate the token service

           Office365TokenService tokenService =
           retrofit.create(Office365TokenService.class);

           try {

                //Get refreshed tokens

                return
                tokenService.getAccessTokenFromRefreshToken(tenan
                tId, APP_ID, APP_PASSWORD,"refresh_token",
                tokens.getRefreshToken(),getOffice365APIRedirectU
                rl()).execute().body();

            } catch (IOException e) {



                //Return an error

                Office365TokenResponse error = new
                Office365TokenResponse();

               error.setError("IOException");
```

```
                    error.setErrorDescription(e.getMessage());

                    return error;
               }
          }
     }
}
```