KDU UNIVERSITY COLLEGE PENANG CAMPUS

SCHOOL OF ENGINEERING, COMPUTING AND BUILT ENVIRONMENT

DEPARTMENT OF COMPUTING

DIPLOMA IN COMPUTER STUDIES

[DPT3054 & PROJECT]

[Ms Tan Phit Huan]

ASSIGNMENT : 1

[Documentation]

[MANICKAM MURUGAPPAN (0194907)
LAM E SHUEN (0195136)
LIM ZHI YI (0195602)
TEH ZHI HERN (0195308)
TEE YENNY (0194670)]

DUE DATE : [7 DECEMBER 2018]

TOTAL MARKS : [100 MARKS / 50%]

DOCUMENTATION

Table of Content

Abstract

The purpose of why the team decided to create Hub@KDU as a dual-platform system consisting of a mobile application and a website application for the students and admins of KDU Penang University College is because the team wants to make their lives easier with college related activities. The mobile application is designed for students to use to view their semester invoices online and to carry out payment through online payment methods. Students will also be able to get notifications of announcements posted by admins and will be able to view them in the app. Admins will be using the website to deploy invoices for students electronically through the website. The invoice will then be sent to the mobile application to be viewed by students. Admins will also be able to post announcements in a more conducive environment in terms of the User-Interface Design being more user-friendly.

Hub@KDU was came up with to solve the problems faced by admins and students of KDU Penang University College. The main problem faced by students is that they have to manually collect their invoices from their respective department by themselves and during invoice collection day, many students will crowd around the department office to collect the invoices which will cause havoc for the admins. Therefore, the team felt that there should be some sort of technology related application that can solve this problem and thus, Hub@KDU was created. Not only invoice collection but students face problems with payment of the invoices as well as there are times there will be long queues at the bursary of the college as everyone is queuing up to pay their invoice and there is usually only two counters. Hub@KDU was created to solve this problem as well. Keeping in mind admins as well, HUB@KDU was created to help admins deploy invoices electronically and to overcome the traditional way of printing students' invoice one by one. Announcement feature was also came up with to help admins notify students regarding important announcements. Most of the time, admins will just post announcements in the existing platform and students will not know about it since there is no notifications at all.

Therefore, the documentation below describes about how Hub@KDU was created by the team and how the functions and features were implemented for this project describing details such as the system development process, in depth explanation on the features and even diagrams that were used to model the structure and workflow of the activities within Hub@KDU.

Chapter 1: Introduction

1.1 Overview: current technologies/trends

Throughout the 21st century, technology has been evolving from time to time that it is now known as the driving force of change in the public's eyes. From old mobile phones to smartphones, to the Internet and many more, technology has been embraced by majority of people even though it is constantly changing and evolving as it is now incorporated into daily lives of many. This technological revolution has more pros over cons and is proven as it has changed conceptions of distance and time for example.

There are a few current technological trends that are very popular in the current technological industry. The first technological trend is the Internet of Things also known as IOT. The Internet of Things is the concept that all devices can be connected to the Internet and to each other to create the perfect link between the physical and digital worlds (Anon., n.d.). Devices like kitchen appliances for example a refrigerator and cars can be connected to the Internet making them the Internet of Things. With the Internet of Things, business industries for example can track how customers use and engage with their products by tracking customers' interaction with the products from a remote area. With this information, they can optimize their products to better suit customers' needs.

Machine learning is also another popular technological trend. In machine learning, a computer can self-learn by its own (Anon., n.d.). The computer has the ability to analyse data and tracking repeating patterns (Anon., n.d.). A good example of machine learning implementation is on social media platforms such as Facebook. Facebook uses machine learning to get a better understanding of how users are connected to other users in their social network. Facebook will analyse things that users had liked, shared or commented. With this information, Facebook will prioritize serving the content related to the information that it had learned to the user first rather than others.

Virtual reality also known as VR is popular among the technological world as well. Virtual reality is the term used to describe a three-dimensional, computer generated environment which can be explored and interacted with by a person. (Anon., n.d.). The person becomes immersed and a part of this generated virtual world and in the virtual world the person can manipulate objects or perform a series of actions. Virtual reality has been mainly implemented in video games for several years already and is continuing to expand further. It is

also now implemented in fields such as medicine and educational organizations with the ambition to lead to new and exciting discoveries that can help impact upon peoples' day to day lives.

1.2 Background of Project

For this project, Hub@KDU is the name given by the group. The reason Hub@KDU was proposed was because the group wanted to make students and admins life easier. This is in the sense of two things which is invoice collection and payment and important announcements. The current system practiced in the college requires students to manually collect their invoices in their respective department. After that, students will need to go to the bursary to pay their invoices which requires them to queue up when the line is long as there are usually only two counters. This is troublesome as it requires lots of time. For admins, they must print out each invoice respectively for each student which increases the cost of printing materials and usage of paper. They need to then manually hand out the invoices on collection day which is hectic as many students will crowd up in each department to collect their invoices. As for announcements, admins will usually not know if students are notified of an important announcement. They usually call a student to check if they know about the announcement. In conclusion, Hub@KDU project was proposed to solve these problems.

Therefore, this project is made to be used by two parties. The first party will be the students of KDU Penang University College where they can use a mobile application to see their invoices online for each semester, carry out online payment using the app and students will also be notified of important announcements in a more user-friendly way. The second party will be for the admins of the college where the admins will be using a website that is linked to the mobile application through a database. Admins can view and download invoices and make any changes to the invoices when there is error by editing the data that is related to students' invoices. They will be able to deploy the invoices through the website which will then be sent to the mobile application to be viewed by students. They can also post announcements on the website to better notify students through the mobile application.

For this project, a total of 5 members were assigned to work on this project. The members were then further broken down into 2 groups. The first group was in charge of

programming the project and connecting the database while the second group mainly oversaw the documentation consisting of written reports but helped in coding as well.

The members and their roles in this project are addressed as below:

1) Manickam Murugappan (1st group) → Team leader for this project and in charge of the setting up the database and coding the project.

2) Lam E Shuen (1st group) → Main leader for the coding group and in charge of coming up with the functionalities for this project

3) Teh Zhi Hern (1st group) → In charge of the UI design for this project and some coding sections of the project.

4) Lim Zhi Yi (2nd group) → Main leader for the documentation group and in charge of following up with the supervisor of the group.

5) Tee Yenny (2nd group) → In charge of project documentation and information searching for this project.

Finally, the group members will hope to create a system that provides fullest functionality for the users. The mobile application will be mainly coded in Java and will be coded using Android Studio, a software used for creating mobile application. The website meanwhile will be mainly coded in PHP with other programming languages as well such as HTML, CSS and Javascript. The group will be learning and researching on PHP as the group is not familiar with the programming language. Lastly, the group will also be researching on the area of databases and how to connect a database with a system and extracting data from the database while maintaining the database.

1.3 Problem Statements

In the world that society lives in today, the majority have their lives revolve around technology. Without technology, many activities that are carried out daily will be hard to carry out. Technology is created so that these activities will be easier and convenient. For example with technology, compared to writing a mail and sending it in the past, people can now easily send an email through mobile devices or computers. From young children to senior citizens, everyone has a mobile phone. Therefore, with technology such a huge focus in today's world, the group wants to create a system that can help students and admins of KDU Penang University College.

The first problem to why the system needs to be created is because at the start of every semester, there will be havoc around each department office and the bursary. This is because students need to collect their invoice from the department office manually and pay their fees at the bursary which may require them to queue up if the line is long. For admins, invoices must be printed out manually as well which is also a waste of paper. Admins will then have to go around informing students that the invoice can be collected, and some students would not be informed because they were not in class.

Furthermore, how can we make sure every student pays on time and ease the work of the admins? This is an important thing because students will be charged extra if they fail to pay before the dateline and they will be barred from using the school's facility. This project is aimed to overcome this problem.

The next problem is moodle as the main source of important announcements for KDU. There are times the system is down and cannot be accessed by the students, lecturers nor the admins. There are also instances where students do not see important announcements such as class cancelation because there is no notification from moodle regarding the announcement. The announcement will be posted there and that is all. Therefore, how can we make sure announcements from lecturers or admins get to students through this project?

1.4 Aims of the project

- To implement a mobile phone application that can provide students with electronic invoice and online payment.
- To allow admins to deploy invoices for students electronically.
- To allow admins to post announcements through a user-friendly website that will notify students through the mobile application.

1.5 Objectives of the project

For the objective for this project, there are a few objectives the group would like to achieve after making Hub@KDU. The first objective will be to help students with the collection and payment of their electronic invoice. This is with the objective that students do not need to waste their time to collect their invoice manually from their respective department and to queue up at bursary for their payment. To do that, the system must have the efficiency of auto generating the required electronic invoice in a PDF format for students.

The Hub@KDU project must also increase the efficiency of admins' work to post important announcements through the website and upload students' invoices electronically. The expected outcome through this project is to reduce paperwork for admins and to avoid admins handling out invoices manually to each student. The announcement function is also expected to increase the efficiency for admins to inform students regarding important announcements.

1.6 Summary of each chapter

In Chapter 1, we made an overview of the current technological trends. After having group discussions between the members, we came up with the idea for this project. With that, the members also discussed and came up with the background of the project and the problem statements. The name of this project is Hub@KDU and the motive of this project is to make students' and admins' life easier as stated in the background of the project. The information and the roles of the members that the members will be carrying out for this project were stated. For this project, several programming languages will be used such as HTML, PHP, CSS,

MQSQL and Javascript. The aims and objective of the project were also stated so that the members stay on track with the objectives and complete this project within the timeline given.

For Chapter 2, we did a feasibility study to determine if this project is technically feasible, whether we will be able to technically develop a well-built enough system to be used by other people. Operational feasibility was also done to see if students and admins of the college want to use the system the team will be developing. On top of that, we also carried out economical feasibility, listing the tangible and intangible benefits of the system we are developing. Thus, we have stated the results of our feasibility analysis in this chapter. We also researched in the current market to see whether there is an existing application that serves the college as well and found a mobile application named KDU. We then made a critical review of this application as it was relevant to the project.

In Chapter 3, we discussed about the requirement specifications for this project, listing the functional and non-functional requirements needed for our project. In requirement specification, we explained what the main functions of our project are. We will also be discussing about the process of our system development as well as providing justification of the chosen development tools and methods.

For Chapter 4, we will be discussing on the system design using Activity Diagrams, Use Case Diagrams, Class Diagrams and ERD diagrams. We will also be showing the user-interface design for this project, providing the sketches and comments we had on the design for this project for this section of the documentation.

In Chapter 5, we will be focusing on the implementation part of the project. We will be explaining the codes in this section as well as the selection of programming languages for this project and the integration methods.

Moving on to Chapter 6, we will be carrying out test cases for this project. We will be carrying out white box and black box testing mainly to ensure that our system runs perfectly for users to use without errors.

Finally for Chapter 7, we will be summarizing all the previous chapters in this section and recommend some things for future system enhancement.

Chapter 2: Primary study / Literature Review

2.1 Feasibility Study

Feasibility analysis is the analysis to identify the opportunities and limitations of the project and to decide whether to proceed with the project or not. For this project, there will be three types of feasibility analysis carried out.

i) Technical Feasibility

The first feasibility analysis will be technical feasibility. Technical feasibility answers the question of whether a project can be built and is used to identify to which extent the system being built can be successfully designed, developed and installed. After carrying out technical feasibility, the Hub@KDU project can be developed by our group. The first reason is because the platform our group is using to develop this project and the programming language used is familiar to the group. Our group will be using Android Studio which is a software the group is familiar with as we have learnt the knowhows of the software in our previous semester leading to this final year project. As for programming languages, the members of the group also know most of the languages needed for this project such as Java, html, css, MySQL and Javascript. Therefore, the group will be familiar with the development environment for this project which generates less risk for the project. Next, the project size of Hub@KDU is also not large. All system requirements that are needed for this project and think can be developed within the group's capabilities have been clearly stated to ensure it provides fullest functionality for the users. Extra requirements that are not important are not stated as it will complicate the project and make the project size too big. There is not much complexity for the systems that need to work together as well. Our group only needs to link a database and a e-payment system together for this project.

ii) Operational Feasibility

The second feasibility analysis is operational feasibility. Operational feasibility is an analysis that analyses how well a system will be accepted by other people. It answers the question of if our group builds the project, will it be used by other people? Our group carried out operational feasibility by carrying out two different surveys whether people would use our

application. The first survey was aimed at students of KDU Penang University College. In the survey, there was a question that asked whether students would use the application if the group were to develop it and out of 74 responses at the moment, 67 responded between 3 to 5 which was likely to most likely. Out of those 67 responses, 36 responded most likely which is a good response towards the project. In conclusion, students would want to use the application made by the group as it will be easier for them to pay their invoices and also get notified of important announcements. The second survey was aimed at the admins working in KDU Penang University College. As the admins are currently busy with their own work, rarely any admins had the time to carry out our survey but the group went to personally ask them ourselves and many of them responded yes to the project.

iii) Economic Feasibility

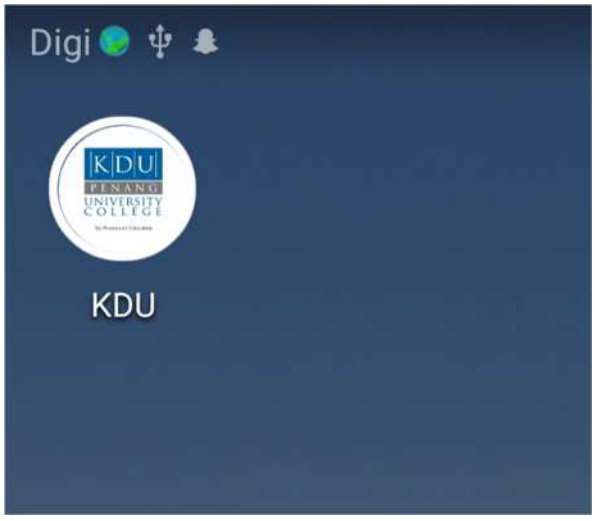
The third and final feasibility analysis is economic feasibility. Economic feasibility is also known as cost-benefit analysis and answers the question of should the group build the system? For this project, the group analysed that the estimated duration and cost for this project is feasible to build the system. The duration which is of 14 weeks is enough for the group to carry out planning, analysis, design and implementation. As for the cost for this project, since this project is made for educational purposes at the moment, developing it does not involve any cost. The software used is free and can be downloaded online while the server needed to host this project is given by the head supervisor for this project. If the project gets approved by KDU Penang University College, the estimated cost of activities such as maintenance of the server will only be analysed by the group then. Moving on, after carrying out economic feasibility, the group believes that the Hub@KDU project has intangible and tangible benefits. For intangible benefits, the system made will be important as it will help students and admins overcome the current system's problem at KDU Penang University College. The group believes that the system will be advantageous for admins to use thus improving services for students. For tangible benefits, a reduction in cost related to printing materials is expected to decrease with the development of Hub@KDU project.

2.2 Conclusion

In conclusion, the group has carried out three types of feasibility analysis which are technical, operational and economic feasibility. In terms of technical feasibility, the group has identified the platform development that will be used for developing this project as well as the programming languages that will be needed. Next, operational feasibility was also done and could be concluded that majority of students and admins in KDU Penang University College would use the system the group will be developing. Finally, the last feasibility analysis done was economic feasibility and the tangible and intangible benefits of the system could be identified after carrying out this analysis.

2.3 Critical review of literature relevant to the project

As Hub@KDU is a project which involves a mobile application, the team has researched in the current market whether there is an application that is made specifically for the college as well. After many researches, the members found a mobile app in the Android Play Store named KDU. Below will be the screenshots of the app as the team will be reviewing it as it is relevant to the current project.



Screenshot 2.3.1

The above is the KDU icon image for the app. The icon is the logo of KDU Penang University College. Since Hub@KDU is a mobile application for KDU, the logo of our mobile

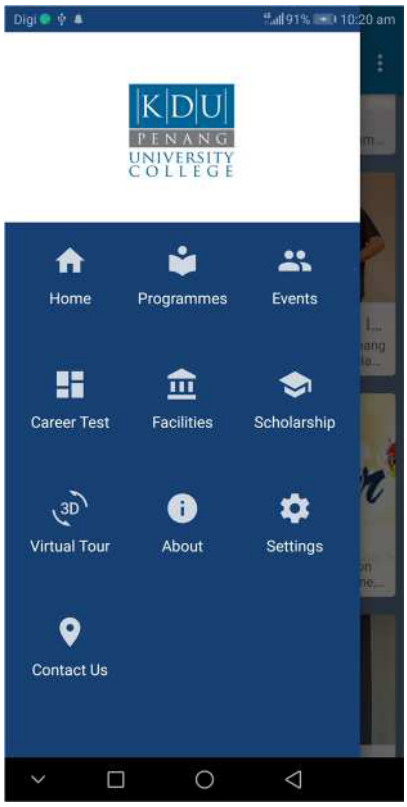
application will also have the word ‘KDU’ in it and the team will edit the icon for the mobile application using the software Adobe Photoshop.

Screenshot 2.3.2

This is the loading page of the KDU app. As the main page loads, users will be directed to this page for around 3 seconds. This is of course a nice touch to the app but our application will have a certain sort of animation as our application loads unlike the current application which is just a static page.

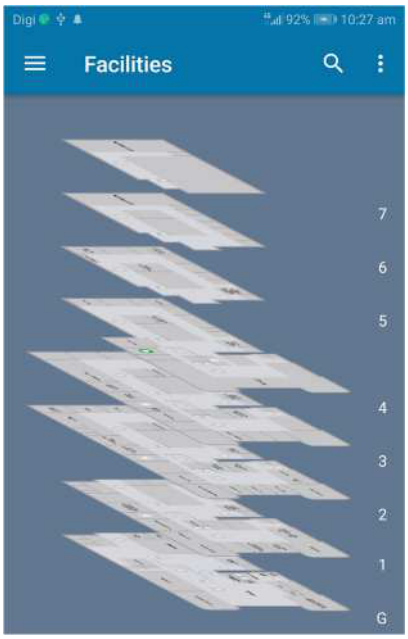
Screenshot 2.3.3

Moving on to the home page, the home page of this KDU app is designed such that users will be able to see the current events occurring in the college. They will be able to see events organised by KDU or achievements achieved by the students of the college. The overall UI design of the app is well done. There is a left sidebar for users to click to see other options to do in the application.



Screenshot 2.3.4

The above is the sidebar of the KDU app. There are a few options for users to choose such as a career test which can help users to discover about the career they would like to pursue. The app also shows the programmes available in KDU in the programmes option. There is also a 2D diagram representation of the facilities in KDU in the facilities option.



Screenshot 2.3.5

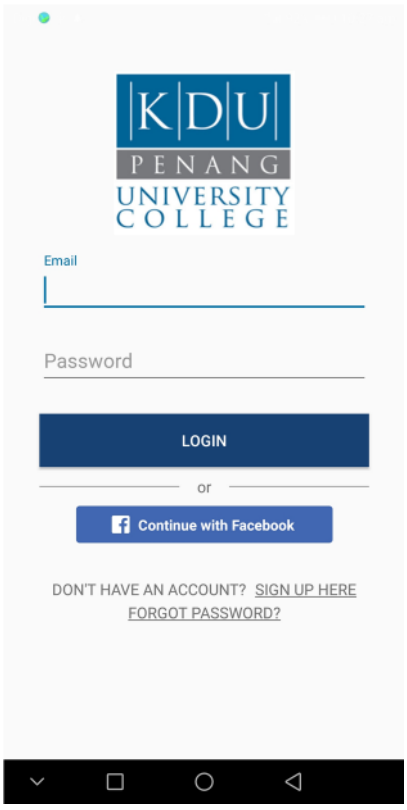
Screenshot 2.3.6

Users will be able to see the facilities available in the college through the facilities option. They can click on the eight diagrams available to further see in detail the facilities in each floor.

Screenshot 2.3.7

Screenshot 2.3.8

Moving on, in the virtual tour option of the app, users will be able to experience the environment in the college through virtual reality. They will be able to see the whole building of KDU Penang University College and can have a virtual tour of the facilities available there like for example the computer lab.



Screenshot 2.3.9

As the members were reviewing the app, the app requires users to sign up with an account to access the events and scholarship options in the sidebar menu. This to us is an inconvenient thing to do. In todays society, many of us do not like to do things which is a hassle such as signing up as there are many details needed to be fill up. Of course, maybe the app requires users to sign up for these two options as the admins of the app need to keep track of enquiries from users regarding the events and scholarships available but it will be nice if users could access these options without signing up as available in the other options if they have no enquires but would just like to see the events and scholarships available in the college.

2.4 Conclusion – Critical review of literature relevant to the project

In conclusion, after reviewing the app, it can be concluded that the application the group is making is different to the app that is already available. The KDU app was made very well with incredible features such as the virtual tour and the facilities option which is something the group will not be able to implement into Hub@KDU as the members have never been exposed to virtual reality concept. However, those features do not need to be implemented because Hub@KDU is created to solve the problems faced by students and admins of the college. Features like the virtual tour and facilities options in the KDU app was implemented as the KDU app is made more for users who want to know more about the college for example the programmes and facilities available before enrolling in it.

Although the app can be used by students of KDU as well to know more about the current events happening in the college, the app fails to provide a stronghold for why students should install the app because students can find out about these events themselves when they are in college. These events can be seen in the foyer of the college. As for other people who want to find out more about the college, they might visit the college itself rather than downloading the app to see the facilities through the screen as at times what is displayed on the screen is different from real life. People would rather be able to see and feel these things by themselves to be more convinced rather than the pictures in the app as some may think it is edited. People would also rather come to the college to enquire more about the programmes available from the admins rather than submitting their enquires through the app which will then take time for the app's admin to reply thus proving why there is an open day for any college in the world. Therefore, although the KDU app was well created, it fails to provide a stronghold in terms of why people should install the app based on our review. Hub@KDU will have a strong valid reason why people should download the app because it will benefit them if they do as this project prioritizes to make students and admins life easier. Being able to pay invoices through online payment compared to the current practice and also be notified of important announcements through the app are strong reasons why students will download the application as it will benefit them unlike the KDU app.

Chapter 3: System Analysis / Methodology

3.1 Requirement Specification

The group will be developing two things. The first will be a mobile application for students and the second will be a website for admins. These two platforms will be connected through a database that is hosted on the server that KDU provided. For the mobile application, students will be able to receive and view their invoices through e-invoices and they can pay through the mobile application as well with e-payment but will be implemented using a fake payment gateway because legit payment gateways need certain formal procedures that have to be done officially to be implemented. They will also be able to view their payment history and also download the invoices. Students will also be notified of important announcements from admins through the app. For admins, they will have a website where they can view and download invoices of students and make any changes to the invoices when there is an error. This can be done by editing the data related to students' invoices. They can also delete the invoices of students after students have paid their invoices. Admins will then be able to deploy the invoices for students using the website. They can deploy invoices for one student or for many students at once. Through the deployment of the invoices, admins can then unblock the payment for students to pay their invoices and will be able to control the due date of the invoice payment as well. Finally, admins will be able to post announcements on the website as well.

i) Functional Requirements

a) Mobile application (For students)

No	Function	Description
1.	Login and logout	Students will be able to login and logout of the mobile application. When the app is launched for the first time, students will be brought to the login page where they need to enter their Student ID and their default password that was set by KDU. Students can also logout of the mobile app at the main menu by pressing the logout button.

2.	Viewing invoice	Students will be able to view their invoices electronically. Invoices will be auto generated by the system by pulling the required data from the database and converting it to e-invoice for students to view.
3.	Downloading invoice	Students have the option to download their invoices in a PDF format to keep records of it. The downloaded invoices will be stored and saved in a specify folder dedicated to store all the .pdf invoice files.
4.	Making payment	In this system, one of the main functions is to allow students to pay their invoices through e-payment. Students can choose the payment method to their own preference. Payment methods such as paying through credit or debit card or through online banking is available but are implemented using a fake payment gateway.
5.	Viewing payment history	After payment, students can view their payment history as proof that they have paid their invoices and to check whether they successfully paid their invoices.
6.	Update invoice list	When students are using the application and they want to see if they have any new invoices, they can swipe down from the main page to update the invoice list to see if they have any new invoice deployed in the app.
7.	Notification for newly deployed invoice.	When a new invoice is deployed, students will get a notification saying that there is a new invoice in the invoice list.
8.	Blocking late payment	Students that do not pay within the deadline will be blocked by the system when they want to pay. They will need to go to the bursary to pay the late invoice payment. This is to adhere to the policy

		practiced by KDU where late payments will incur fines and extra charges.
9.	Viewing announcements	The system will notify students when admins post an important announcement. They will be able to view the announcements through the mobile application.
10.	Pin and unpin announcements	Students can pin announcements that are important so that it will be displayed as the first announcement that students will see. They can also unpin the announcements when the announcements are not important anymore.
11.	Viewing FAQ	There is a FAQ section within the application and students can go to this section to see some frequently asked questions regarding the app.

b) Website (For admins)

No	**Function**	**Description**
1.	Login and logout	Same as students, admins will be able to login and logout from the website. Admins will need to enter their username and password for authentication. Admins will be able to logout as well by clicking on the logout tab.
2.	Viewing invoice	Admins can view invoices to check if there is any error when needed. The concept is the same as the mobile app where the invoice is auto generated after extracting data from the database.
3.	Download invoice	Admins can download the generated invoices for future references for themselves.
4.	Editing student's data for invoice	The website allows admins to make changes to data related to students' invoices when there is an error. Admins can enter the Student ID of the

		student whose data need to be edited and from there, the information related to that particular Student ID will appear and from there, admins can edit the data which will update the database as well.
5.	Deleting student's invoices	Admins can delete the invoices of students after students have paid their invoices for example. This function is implemented so that admins do not need to keep redundant data.
6.	Deploying invoices	Admins will have the option to deploy invoices for either one student or many students at once. The deploy for one student function was implemented as a safety feature for admins to just deploy the invoice of one particular student if for example the admin accidentally deletes the invoice of that student. Through deploying the invoices, admins will unblock the payment for students in the mobile app and will also be able to control the due date payment for the invoice.
7.	Post announcements	Admins can use the website to upload and post announcements to notify students through their app. The announcements will be uploaded to a database which is the connection between the mobile application and the website. Once announcement is uploaded, students will get a notification from their mobile app. Admins will have a more user-friendly graphical user interface (GUI) for posting announcements.

ii) Non-Functional Requirements

For non-functional requirements, the mobile application and the website is very similar. Below are the non-functional requirements for Hub@KDU.

No	Non-functional requirement	Description
1.	Operational	Operational non-functional requirement means the physical and technical environment the system will be operating in. The mobile application is built using Android Studio and can run on Android version Lolipop (5.0 – 5.1.1) till the latest version of android Pie (9.0). As for the website, the website is built using programming languages like HTML, CSS, Javascript and PHP. It will be operating mainly in Windows and can be supported by any web browser. phpMyAdmin was used as it was a convenient administration tool to work with MySQL database management system. The website and the mobile app are coded using the media rule for responsive web design. This enables the system to adjust itself based on the device and web browser being used.
2.	Security	The security implemented for this system is that only admins can see the data fields that is in the database when editing student's data for invoice. Students will have no access at all to this section to prevent temperament of data. Finally, login and logout option are also implemented to increase security. The team will further improve the security in the future by using a hashing algorithm to hash students' and admins' credentials.
3.	User – friendly	The system made was made to be as user-friendly as possible. In this age of time, many users do not

		want to read instructions on how to use a system or app. Instead they would like to explore the system by themselves. Therefore, the team has made the system such that the system can be used by everyone in the society. Even a person who is not tech savvy will be able to navigate through the system easily as the system navigation is very simple as the team did not complicate the design for the system. There will be tabs and buttons that are clearly visible for users to click to navigate through the system.
4.	Performance	The performance for this system is fast. As the system has been coded with the required codes only, the running time is fast. The team did not put in extra codes that were redundant. The response time for the system is quick as well as there were not many pictures that were needed to load as the system starts. The only picture used was the logo of the college for the website and for mobile platform, only small images were used as icons in some sections. There were also no videos for this system.

3.2 Process of System Development

For this project, the team has decided to use Throwaway Prototyping method which is under Rapid Application Development (RAD) as the methodology method the team will adhere to in the process of system development for this project. The Throwaway Prototyping Method is suitable for this project as in Throwaway Prototyping Method, the analysis, design and implementation phases are done concurrently. As these 3 phases are done concurrently, the team believe it will save time as well as increasing the efficiency of the project.

After carrying out these three phases, the team will be able to develop a design prototype of the system. As Throwaway Prototyping Method is highly dependent on thorough analysis phase that is used to gather information and to develop more ideas for system concept, the design prototypes are useful for the team as the prototypes will be used to confirm the important issues to minimize the risk associated with the system before the real implementation of the system is built. This means that as the design prototype is built, the team will see if it meets the standard we are expecting. If the design prototype does not match the features the team is trying to implement, we will be able to go back to the 3 phases again and come up with another design prototype. This process is repeated until the team is satisfied with the design prototype and will proceed to the real implementation of features and functions of the system.

3.2.1 Throwaway Prototyping

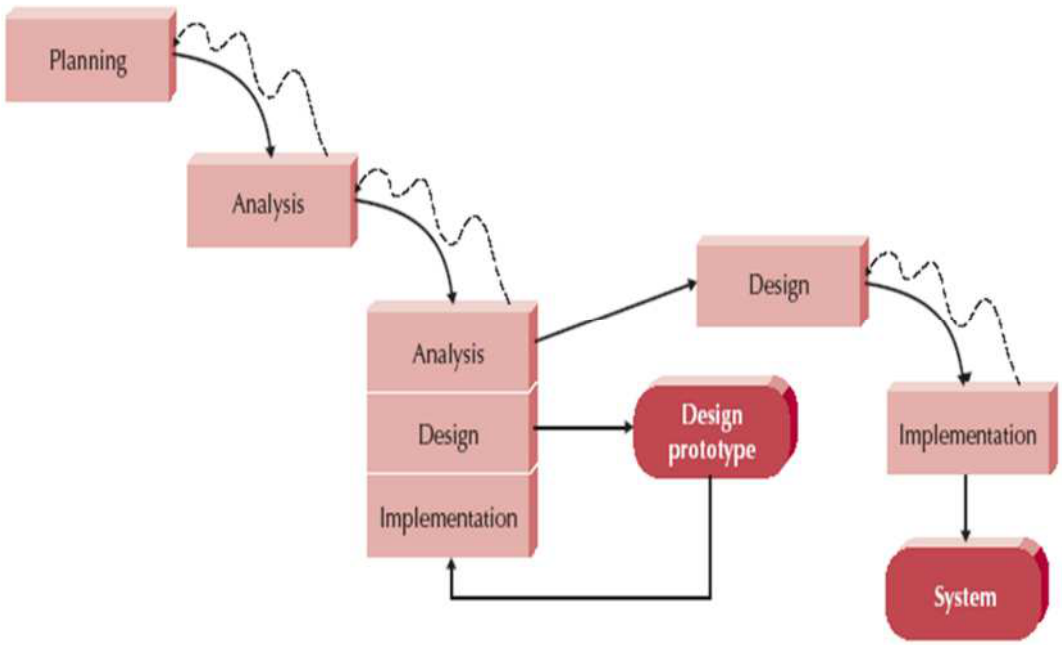


Diagram 3.2.1.1 Throwaway Prototyping Method (danny.chen, 2017).

The reason why the team chose Throwaway Prototyping under Rapid Application Development (RAD) as the methodology method the team will be following for the system development is because this method provides us with a few advantages. First and foremost, with a design prototype at hand, the team will be able to analyse whether the design prototype matches the standard of the features we are trying to implement. Since the team will have a

system prototype for us to interact and experience, the team and even the supervisor of the project can provide feedbacks or suggestions on how to better improve it. The team will be able to rectify the errors at an early stage and not only at the end of the project where the system is fully built.

Next, with throwaway prototyping, it will provide us with the advantage in the context where the team will be able to satisfy all the requirements at the end of the project and produce more stable and reliable systems. Since, throwaway prototyping focuses on thorough analysis that provides us with more information, this might help the team to prepare better solutions for the problems at hand. Without a prototype, the focus on a limited prototype can distract us from properly analysing the project as a whole which may lead to a system that is not properly developed.

Finally, since throwaway prototyping is under Rapid Application Development, it will help us to speed up the analysis, design and implementation phase which will help the team to complete the project faster.

3.2.2 Planning Phase

For the planning phase, the group will first hold a meeting with the supervisor of our project. During the meeting, we will discuss with our supervisor about the features we have for this project and what we intend to do so that our project can begin from the proposal stage. After the meeting, we will then hold a meeting between the team members to properly come up with the features of the project. The team will then proceed to sketching the design we have in mind for the system. From then on, we will properly distribute the tasks for each member with the guidance of our team leader for this project. Moving on, there will be a group meeting that will be done every week in order to keep track of the progress of the project and to discuss about the problems and ways to solve them. The team will also meet with our supervisor once every week for the duration of 14 weeks to keep track with the weekly log and ask our supervisor for opinions on the problems faced as well as things that are related to the progression of the project every week.

Since the team will be creating a system to be used by students and admins of the college, we will try and get the opinions from them regarding the system the team will be creating. We will be making two surveys one for students and one for admins. After the survey,

we will move on to the planning phase of the hardware and software required for this project. We will decide the programming languages the team will be using for this project as well as the tools we will be using to create this project before moving on to the analysis phase.

3.2.3 Analysis Phase

In a System Development Life Cycle (SDLC), the analysis phase is one of the most crucial phases in ensuring the project the team is creating is a success. This phase refers to requirements gathering for the system and also understanding what users want from the system that is being created. In the analysis phase as well, the team will be analysing and understanding the old system currently used and from there identify how to improve the system based on the problems faced by users when using the old system. This will then enable the team to develop requirements for the new system that the team will be creating.

Therefore after carrying out the analysis phase, the system the team will be creating, Hub@KDU, is created to be used by students and admins of KDU Penang University College. This system being a dual platform of mobile application and website will be able to do a few things such as providing students with e-invoice and e-payment through the mobile application. Students will also be notified of announcements from admins through the mobile application when the announcements are posted from admins. For admins, they can deploy invoices for students through the website which will then be sent to the mobile application to be viewed by students. Admins will be able to edit, delete, view and download the invoice. Admins will also be able to post announcements through the website. Therefore, from the team's analysis phase, there will be two types of main users using Hub@KDU which are:

i) Students (Mobile Application)
ii) Admins (Website)

3.2.4 Design Phase

Moving on to the design phase, the design phase is very important for the development of this system as well. The design phase covers designs such as program, user-interface and database design. In this phase, the team will be mainly deciding the designs that covers those aspects. Since the team is adhering to Throwaway Prototyping Method and the design prototype is crucial, the team will be expecting the design to be completed as fast as possible for these aspects mainly prioritizing the program and user-interface design.

For the program and user-interface design, the team will be looking at templates that are suitable to be implemented for this project. For the website design, the team will be looking for templates on Bootstrap, a free and open-source front-end framework that contains HTML and CSS codes, the two main programming languages the team will be using to design the front-end for the website. As for the mobile design, we will be designing our own icons and widgets for the mobile application.

Moving on to the database design, we will be using MySQL as the main relational database management system (RDBMS) to create the databases for this project. Then, the team will be using phpMyAdmin which is a free and open source administration tool for MySQL. The team will also be creating an ERD diagram and a data dictionary to further enhance the database design.

In conclusion, the design phase will be done several times until the team is satisfied with the overall design for this project in all aspects. This is to ensure that the end product of the system will be user-friendly for users to use. We truly believe that with the proper and right design for our system, users can easily adapt to our system as it is user-friendly. The team will also create relevant diagrams in this phase such as the use case diagram, activity workflow diagram and class diagram to show in detail and in depth the features and functions of Hub@KDU and to ensure the workflow of the system can be understood by users.

3.2.5 Implementation Phase (Starting Development)

This stage of the implementation phase can also be seen as the starting development of the project where at the end, the design or system prototype is developed. Once the design phase is done, the team will move on to the implementation phase, implementing certain features that are important to the full project first as well as some plugins. In our team's implementation phase, there were many times changes were made to the design phase as the prior design could not complement the features the team had in mind. Once the changes were done to the design, implementation carried on and we were able to implement features such as extracting data from the database in the server to auto generate the invoice in PDF format which was a crucial feature towards the completion of this project. Other features implemented include allowing admins to view and download invoices through the website.

The design protype was then developed and the team began testing it, fixing bugs and errors before moving on to the real implementation such as deploying the invoices through the website for students to view in the mobile app, editing invoices and ensuring a connection between the mobile application and the website. Login credentials and session management will also be worked on after the design prototype is approved by the team and the supervisor of the project.

3.2.6 Design/System Prototype

When the design prototype is done, the team will be testing the design prototype and analyse whether the design prototype can proceed to the real implementation of features and functions of the program. We will be testing and trying to fix the bugs that exist in the system during this stage.

3.2.7 Implementation Phase (Final)

During this final implementation phase, our system is ready to be used by users with no major errors or bugs. The features that the team has implemented for this project are ready to be used also. However during the real implementation of this project, our team had to drop one of our main functions for this system after discussing with our supervisor. We dropped our announcement function we had in mind for this project due to time constraint mainly because we could not give our full attention to this project as the members had other assignments that need to be submitted earlier, taking most of the time we had in mine for this project away which was a shame. Another reason the announcement feature was drop was because we took a lot of time learning a new programming language, PHP, a language we were not familiar with before. It can be concluded that the ideas we had for this project were there to be implemented but our technical skills were not up to par to the standard required and we needed more time to finish implementing the announcement feature.

We intend to finish implementing the announcement feature for Hub@KDU after submitting our project. From there, the team might change the methodology method we will be using to implement this feature changing from Throwaway Prototyping Method to Phased Development. Our submitted project can be considered as version 1 of Hub@KDU where the invoice feature is working. We can then move on to version 2 of Hub@KDU where the announcement feature will be implemented together with the invoice feature. Version 3 can be the version where an official payment gateway is implemented in our system. In the future as well, we plan to further develop Hub@KDU with more new features to create more system versions of Hub@KDU.

3.2.8 System

This is the final stage or phase where our project, Hub@KDU is ready to be used by users. It is ready to be installed without any major bugs or errors.

3.3 Justification of Chosen Development Tools and Methods

In this section, the tools and programming languages that were used to create this project will be listed with explanation as per below. Since Hub@KDU is a dual-platform system, the tools and programming languages can be divided into two parts, where the first part will be for mobile development and the second will be for website development.

3.3.1 Android Studio

For mobile development, the team has resorted to using Android Studio, as the software for us to code our mobile application. Android Studio is the official integrated development environment (IDE) for Google's Android operating system and is designed specifically for Android development (Anon., 2018). One of the reasons why the team will be using Android Studio is because the team members are familiar with this IDE. We have been taught on how to make mobile applications using Android Studio as our main IDE. Therefore, since Android Studio is free and available for download on Windows, macOS and even Linux based operating systems and since the team knows the knowhows of Android Studio, Android Studio is the main software the team will be using for mobile application development.

Diagram 3.3.1.1 Android Studio Logo

3.3.2 Java

Java is the programming language that will be used to code the mobile application. Android Studio is built on JetBrains' IntelliJ IDEA software, which is a Java integrated development environment (IDE) (Anon., n.d.). Therefore, most mobile applications that are created using Android Studio is through Java programming language as it is a versatile language that can be integrated for all types of systems.

Diagram 3.3.2.1 Java Logo

3.3.3 Notepad ++ / Sublime Text 3

Moving on to website development, the main software the team used for website development are Notepad++ and Sublime Text 3. Both are source code editors which supports several types of programming languages. The reason why the team is using this two software and not only one is due to preferences. Some of our members prefer using Notepad++ while some prefers Sublime Text 3 as each has their own unique features. Ultimately, the team used Notepad ++ to compile all the files created for the project. This two software were chosen as the software to be used for web development as both supported the programming languages needed for this project.

3.3.4 PHP

For website development, PHP programming language was mainly used for back-end programming. PHP is a server-side scripting language designed for web development and since our database is located on the server it will be easier to communicate to the database using PHP. The team will be coding PHP scripts to communicate to the database to pull the required information from the respective tables.

3.3.5 HTML/ CSS / JAVASCRIPT

Another crucial part in website development is the design of the website, the front-end programming. For front-end programming of the website, the programming language mainly used were Hypertext Markup Language (HTML), Cascading Style Sheets (CSS) and JavaScript. Each had their own roles like for example HTML was used as a markup to tell the web browser how to display text, images and other forms of multimedia on a webpage. CSS meanwhile was used to describe how HTML elements are to be displayed on the screen and was used to style these HTML elements. JavaScript was also used at time to further enhance the designs of the HTML elements and was also used at times in back-end programming.

3.3.6 MySQL

As we will need to create databases for Hub@KDU where information such as admin's credential and student's data related to invoice will be stored, the team will be using MySQL as the main relational database management system (RDBMS). One of the reasons why MySQL was chosen by the team is because of the user-friendly interface. The UI made it easy for us to learn how to use it which was useful as we were beginner students in terms of database management. Another reason is because there are a lot of tutorials online on ways to use MySQL which was very useful to the team for this project.

Diagram 3.3.6.1 MySQL logo

3.3.7 phpMyAdmin

To complement with the usage of MySQL, phpMyAdmin, a free software tool written in PHP with intentions to handle the administration of MySQL over the web was used by the team. With phpMyAdmin, the team was able to carry out typical operations such as MqSQL database management.

Diagram 3.3.7.1 phpMyAdmin logo

3.3.8 FileZilla Client

FileZilla is a free software that is available on the internet which can be downloaded for free. FileZilla is a tool used by the team to transfer files over the internet to the server that was given to the team for web hosting. The team used it as a File Transfer Protocol (FTP) client to upload the necessary files for our project such as PHP, CSS and image files to run from our local machines to the link which gives us access to store files on the college server which is allocated to us by our college IT technician.

Diagram 3.3.8.1 FileZilla Client Logo

3.3.9 JSON

JSON also known as JavaScript Object Notation, is a minimal, readable format for structuring data and is primary used to transmit data between a server and a web application (Anon., n.d.). Since our website is hosted on a server, JSON was pivotal in ensuring that data was successfully transmitted between the server and the website as well as the mobile application when the mobile application is interconnected to the website.

Chapter 4: System Design / Proposed Solution

4.1 UML/Data Flow/Other Diagrams – Proposed Systems

In this section, the team will be showing all the appropriate diagrams needed for this project to model system requirements and to show the workflow of Hub@KDU using diagrams such as Use Case diagram and Activity Diagram.

4.1.1 Use Case Diagram

The first diagram, a use case diagram, is a diagram that represents what functions the users of a system will be able to do. From the use case diagram as well, we will be able to know the type of users that will be using the system. Each function is represented by a circle that is known as the use case. The users also known as the 'actors', will be modelled outside the whole diagram. In conclusion, a use case diagram specifies the behaviour of a system. Below is the use case diagram created by the team for Hub@KDU.

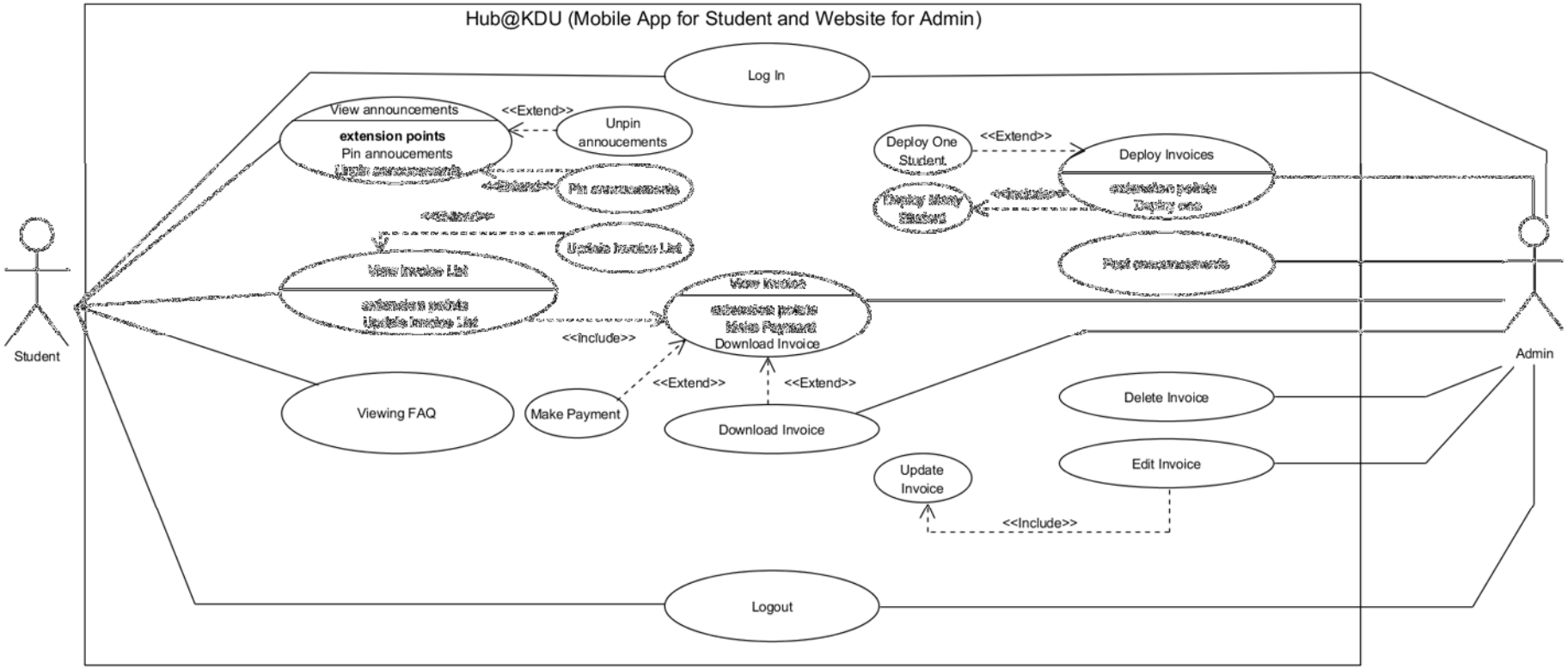


Diagram 4.1.1.1 Use Case Diagram for Hub@KDU

4.1.2 Use Case Diagram Explanation

There will be 2 types of users for the use case diagram above that is student and admin. Altogether, there are 18 use cases in total where some use cases can be carried out by both student and admin.

The functions that can be carried out by both student and admin are such as logging in and logging out from their respective platforms. Both are also able to view and download the invoices that are generated.

For student, they will be able to view the invoice list which is also the payment history. From this use case, there will be an include use case and an extend use case. Include use cases mean that it is compulsory to carry out this use case after carrying out the previous function related to it. Therefore, student must view their invoice after viewing the invoice list. Extend use cases mean that the function can be carried out by users if they want to or they can also choose to not carry out the use case meaning it is up to them. The extension point from the view invoice list use case is update invoice list where student can update their invoice list to check if there is new invoices deployed. Moving on, the view invoice use case will then have 2 extension points. Therefore after viewing the invoice, student can choose to make payment if they need to make the payment or download the invoice for their own references. Student will also be able view announcements. From this, they can choose to pin and unpin announcements based on their preferences. Finally, student can also view the FAQ section.

Moving on to admin, they will be able to deploy invoices for student. They must deploy the invoice for many student as that is the main thing to do under deploy invoice. They can then deploy the invoice for one student only if they need to. For example, they may accidentally delete a student's invoice and need to deploy that particular invoice again. Admins can also post announcements. Furthermore, admins will be able to view, download and delete the invoice. They will also be able to edit the invoice. Once they are done editing data related to the invoice, they must and will be able to update the invoice with the new data.

4.1.3 Activity Diagram

An activity diagram is a diagram that illustrates the processes or activities that are performed and how the workflow is in a system. An activity diagram can be viewed as a sophisticated data flow diagram as it portrays the primary activities of a system and the relationships among the activities.

The team has created 2 activity diagrams since our system is dual-platform. The first activity diagram is for the mobile application for students and the second will be for the website application for admins.

4.1.4 Activity Diagram (Mobile Application for Students)

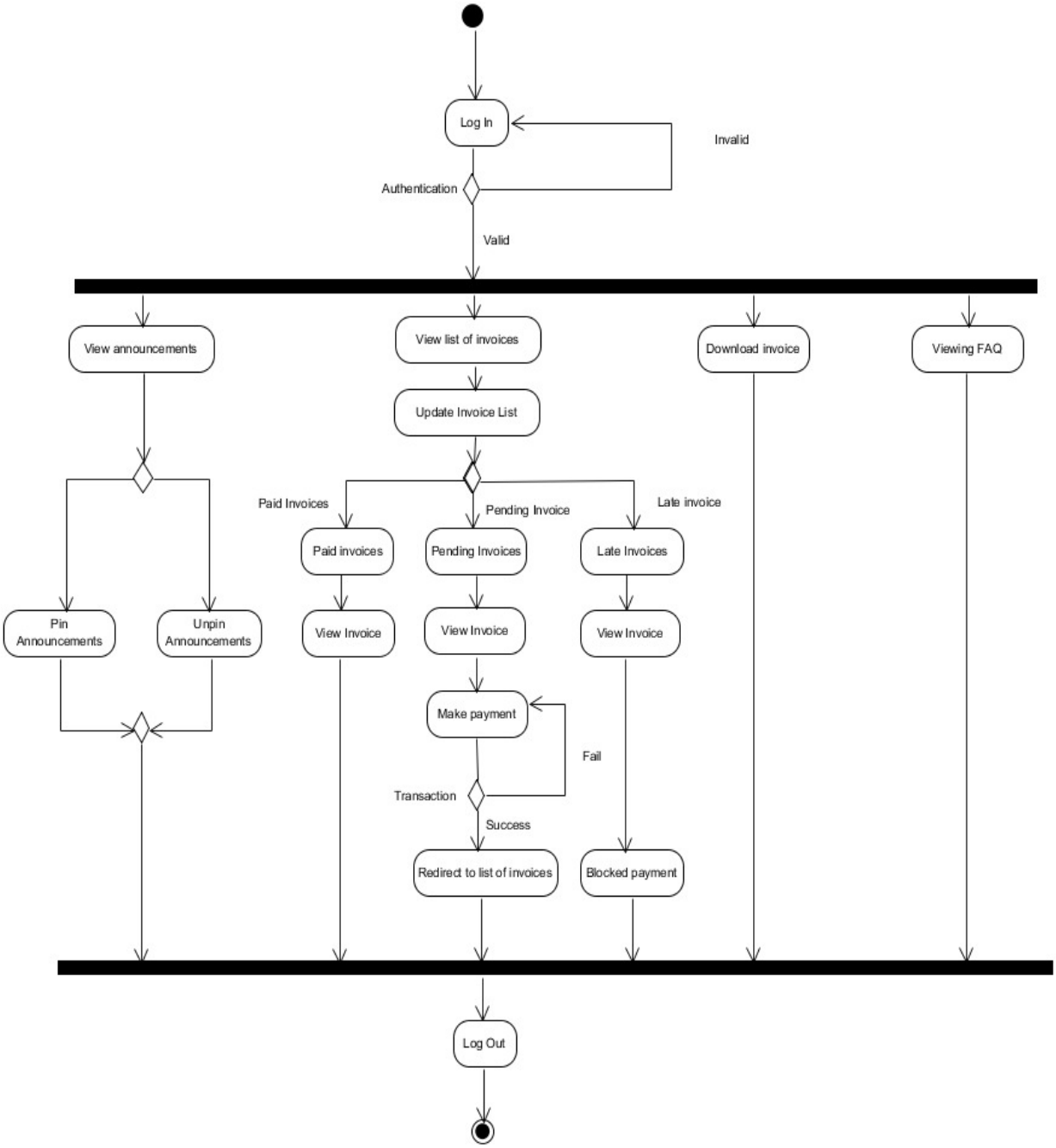


Diagram 4.1.4.1 Activity Diagram for Mobile Application for Students

4.1.5 Activity Diagram Explanation

In the activity diagram above, the first circle also known as the initial node portrays the beginning of a set of actions or activities. From the control flow, the first action to be done in the mobile application is to login. Then, a decision node is used to test the condition of the login. If authentication is valid, students will be able to proceed to other activities. If not, authentication will be invalid and students will be brought back to the login page and they need to try again.

Moving down the activity diagram, there will be 4 different activities for students to carry out. The first activity is view announcements through the mobile application. After viewing the announcements, there will be a decision node with 2 conditions where the first condition is students can pin announcements and the second condition is students can unpin announcements. A merge node is then used to bring back together the different decision paths which will then lead to the last activity in the activity diagram which is logout. Next students will also be able to view list of invoices which is the payment history. From there students will update their invoice list to check for new deployed invoices. From there, the control flow will go to a decision node with 3 conditions being paid invoice, pending invoice and late invoice. These 3 conditions will then lead to 3 different processes and based on which process it is, the next activity can be carried out. Therefore for paid invoice, the next activity will be that students can view the invoice details related to the payment. For pending invoice, students can proceed to view the invoice and then making payment. Although the online payment gateway system is fake, we decided to implement a decision node which will be used to test whether the transaction is successful or not as the concept of a real online payment gateway system will still be such that when the transaction is successful, the control flow will go to the next activity which is students will be redirected to the list of invoice page. If transaction fails, students will need to try and make the payment again until it is successful. Finally for late invoice, students will still be able to view the invoice and the control flow will lead to the next process which is blocked payment as the mobile application does not accept late payment as students with late payment need to pay their invoice at the bursary of the college in adherence to the policy at KDU Penang University College where late payments will incur fines and extra charges. Students can also download invoice and the final activity students can do is viewing the FAQ.

Lastly, the 4 different activities will then flow to the last activity which will be logout. Once students logout, a final activity node is used to stop all control flows in the activity diagram.

4.1.6 Activity Diagram (Website Application for Admins)

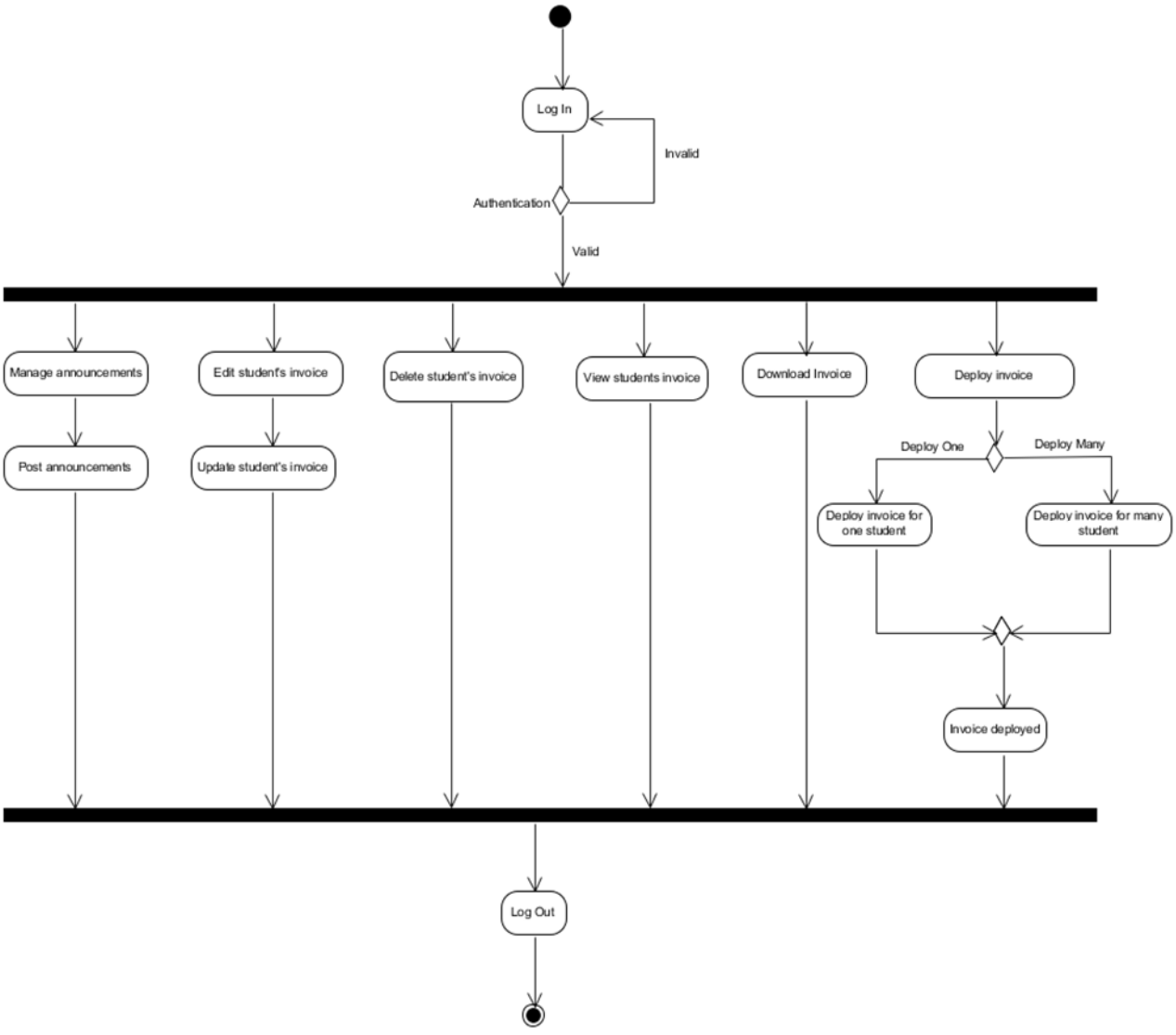


Diagram 4.1.6.1 Activity Diagram for Website Application for Admins

4.1.7 Activity Diagram Explanation

In the activity diagram above, the first circle also known as the initial node portrays the beginning of a set of actions or activities. From the control flow, the first action to be done in the website application is to login. Then, a decision node is used to test the condition of the login. If authentication is valid, admins will be able to proceed to other activities. If not, authentication will be invalid and admins will be brought back to the login page and they need to try again.

Next, admins will be able to carry out 6 different activities after successfully logging in. The first activity they can do is go to manage announcements. They will be able to post an announcement from that page. Moving on, admins can then edit student's invoice. After editing the data related to student's invoice, they can then update the student invoice with the new data. Admins can then delete, view and download the invoice where are 3 are separate activities. Finally the last activity will be deploy invoices for students. A decision node is then used where the 2 conditions are deploy one or deploy many. If the condition is to deploy the invoice for one student only, admins will proceed to deploy the invoice for one student only. If it is the other condition which is to deploy invoice for many students, admins will proceed to deploy the invoice for many students. A merge node is then used to bring back together the different decision paths which will then lead to the invoice being deployed.

Lastly, the 6 different activities will then flow to the last activity which will be logout. Once admins logout, a final activity node is used to stop all control flows in the activity diagram.

4.2 Database Design

As databases are very important in this project, the team will be showing the database structure within this project using diagrams such as ER Diagram and Data Dictionary.

4.2.1 ER Diagram

ER Diagram also knows as Entity Relationship Diagram (ERD), is a type of diagram which graphically illustrates the relationships of entity sets in a database (Anon., n.d.). An entity in this context is an object, which is a component of data (Anon., n.d.). These entities can then have their own attributes that define its properties (Anon., n.d.). There are 4 types of relationships in an Entity Relationship Diagram which are one-to-one relationship, one-to-many relationship, many-to-one relationship and many-to-many relationship. To summarize, ER Diagrams are used to sketch out the design of a database. Below will be the ER Diagram for Hub@KDU.

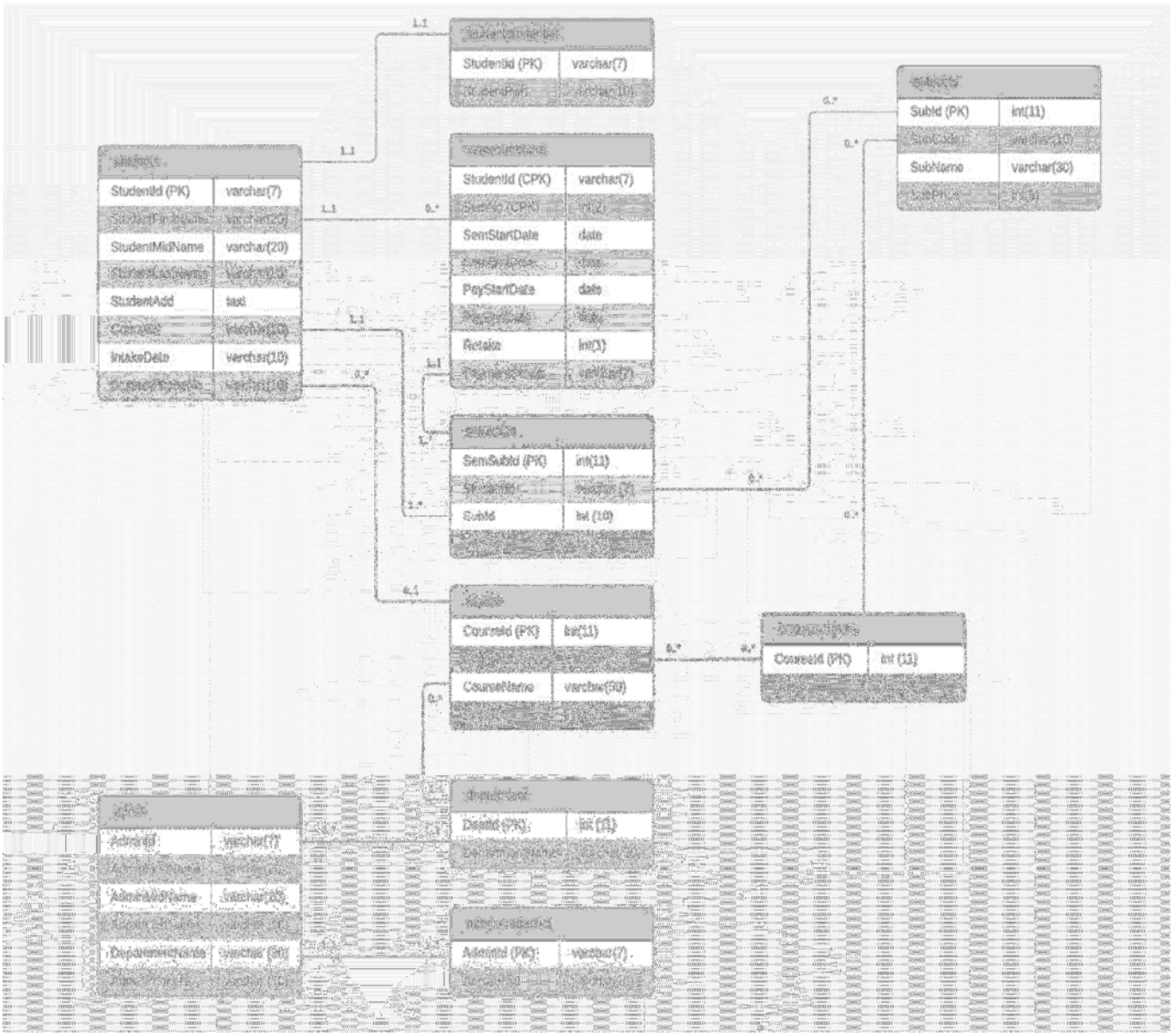


Diagram 4.2.1.1 ER Diagram for Hub@KDU

4.2.2 ER Diagram Explanation

In the ER Diagram that is shown above, there are 10 tables that are interconnected within the database. The students table which holds all the information about the student is connected to the studentcredential table with one to one relationship, meaning that one student can only have one set of credentials. Besides that, the students table is connected to the semestercount table which has all the details for each of their semesters with one to many relationships, which means a student can have zero or more semesters. The students table also has a one to many relationships with the semester table also known as the subjects taken table which holds the information of the subjects the students have taken in each semester. Besides that, the final connection for the students table is the connection with the course table which has the relationship of zero to one which can be defined that the student can be enrolled in one course or in no courses.

The course table also connects to the coursesubjects tables which is then connected to the subjects table which is used to filter the subjects based on the course to prevent duplicated data. The course table is also connected to the department table which aids in filtering out the courses based on the department it is in.

On the other hand, for the admin table, it is connected to the admincredential table with one to one relationship and to the department table to identify which department does the admin works for.

4.2.3 Data Dictionary

A data dictionary is a set of information describing the contents, format and structure of a database and the relationship between its elements, used to control access to the database as well as manipulation of it. Below is the data dictionary created by the team for Hub@KDU.

students					
Field Name	Data Type	Data Format	Field Size	Description	Example
StudentId	varchar	#######	7	Unique ID for students	0194907
StudentFirstName	varchar	n/a	20	First name of the student	Mary
StudentMidName	varchar	n/a	20	Middle name of the student	Elizabeth
StudentLastName	varchar	n/a	20	Last name of the student	Smith
StudentAdd	text	n/a	n/a	Address of the student	27, Perak Road 10150 Georgetown, Penang, Malaysia
CourseId	varchar	n/a	10	The id of the course they are currently enrolled in	12
IntakeDate	varchar	MMM YYYY	10	The date of their intake	Jan 2017
StudentPhoneNo	varchar	##########	10	Student’s mobile phone number	0165636985

admin					
Field Name	Data Type	Data Format	Field Size	Description	Example
AdminId	varchar	#######	7	Unique ID for admin	1234567
AdminFirstName	varchar	n/a	20	First name of the admin	Mary
AdminMidName	varchar	n/a	20	Middle name of the admin	Elizabeth
AdminLastName	varchar	n/a	20	Last name of the admin	Smith
DepartmentName	varchar	n/a	30	The name of the department the admin is working for	Computing
AdminPhoneNo	varchar	##########	10	Admin's mobile phone number	0165636985

semestercount					
Field Name	Data Type	Data Format	Field Size	Description	Example
StudentId	varchar	#######	7	Unique ID for students	0194907
SemNo	int	n/a	2	The particular semester of the student in the current course	2
SemStartDate	date	YYYY-MM-DD	n/a	The start date of a particular semester	2018-12-04
SemEndDate	date	YYYY-MM-DD	n/a	The end date of a particular semester	2019-12-04
PayStartDate	date	YYYY-MM-DD	n/a	The date when the system starts accepting payments for the invoice	2018-11-05
PayEndDate	date	YYYY-MM-DD	n/a	The date when the system declines payments for the invoice	2018-11-15
Retake	int	1/0	1	A boolean value to check if they are retaking subjects	1
PaymentStatus	varchar	n/a	7	The status of the payment whether it is paid, pending or late	Paid

semester					
Field Name	Data Type	Data Format	Field Size	Description	Example
SemSubId	int	n/a	11	The subject taken during the semester	52
StudentId	varchar	#######	7	Unique ID for students	0194907
SubId	int	n/a	10	Unique subject ID which is the foreign key of the subject ID in the subjects table	1
SemNo	int	n/a	5	The particular semester of the student in the current course	5

course					
Field Name	Data Type	Data Format	Field Size	Description	Example
CourseId	int	n/a	11	The unique id for the courses	1
CourseCode	varchar	n/a	10	The code for the courses	DCS
CourseName	varchar	n/a	50	The course name for the courses	Diploma in Computer Studies
DeptId	int	n/a	11	The unique id for the department	2360

subjects					
Field Name	Data Type	Data Format	Field Size	Description	Example
SubId	int	n/a	11	The unique ID for the subjects	12
SubCode	varchar	n/a	10	The subject code for each subject	DBB2234
SubName	varchar	n/a	30	The name of the subject	System Admin and Management System
SubPrice	int	n/a	5	The price of the respective subject in Ringgit Malaysia	1200

coursesubjects					
Field Name	Data Type	Data Format	Field Size	Description	Example
CourseId	int	n/a	11	The unique ID for each course	236
SubId	int	n/a	11	The unique ID for the subjects	12

department					
Field Name	Data Type	Data Format	Field Size	Description	Example
DeptId	int	n/a	11	The unique id for the department	2360
DepartmentName	varchar	n/a	30	The name of the department	Computing

studentcredential					
Field Name	Data Type	Data Format	Field Size	Description	Example
StudentId	varchar	#######	7	Unique ID for students	0194907
StudentPwd	varchar	#####@????	10	Password for students to login. The # is the 5 last letters of their full name followed by "@" then the ? is the last 4 digit of their identification card number	Appan@5483

admincredential					
Field Name	Data Type	Data Format	Field Size	Description	Example
AdminId	varchar	#######	7	Unique ID for admin	0194907
AdminPwd	varchar	#####@????	10	Password for admin to login. The hastag is the 5 last letters of their full name followed by the allias sign "@" then the question mark is the last 4 digits of their identification card number	Admin@1234

4.3 User-interface Design

In this section, the sketches of the design the team did for both platforms will be shown here. In the sketch, there were comments that were stated by the side of the sketch such as the colour code that will be used for certain components so that when team members work on the design, we will know what to implement in the design phase.

4.3.1 Layout Sketch Design for Website

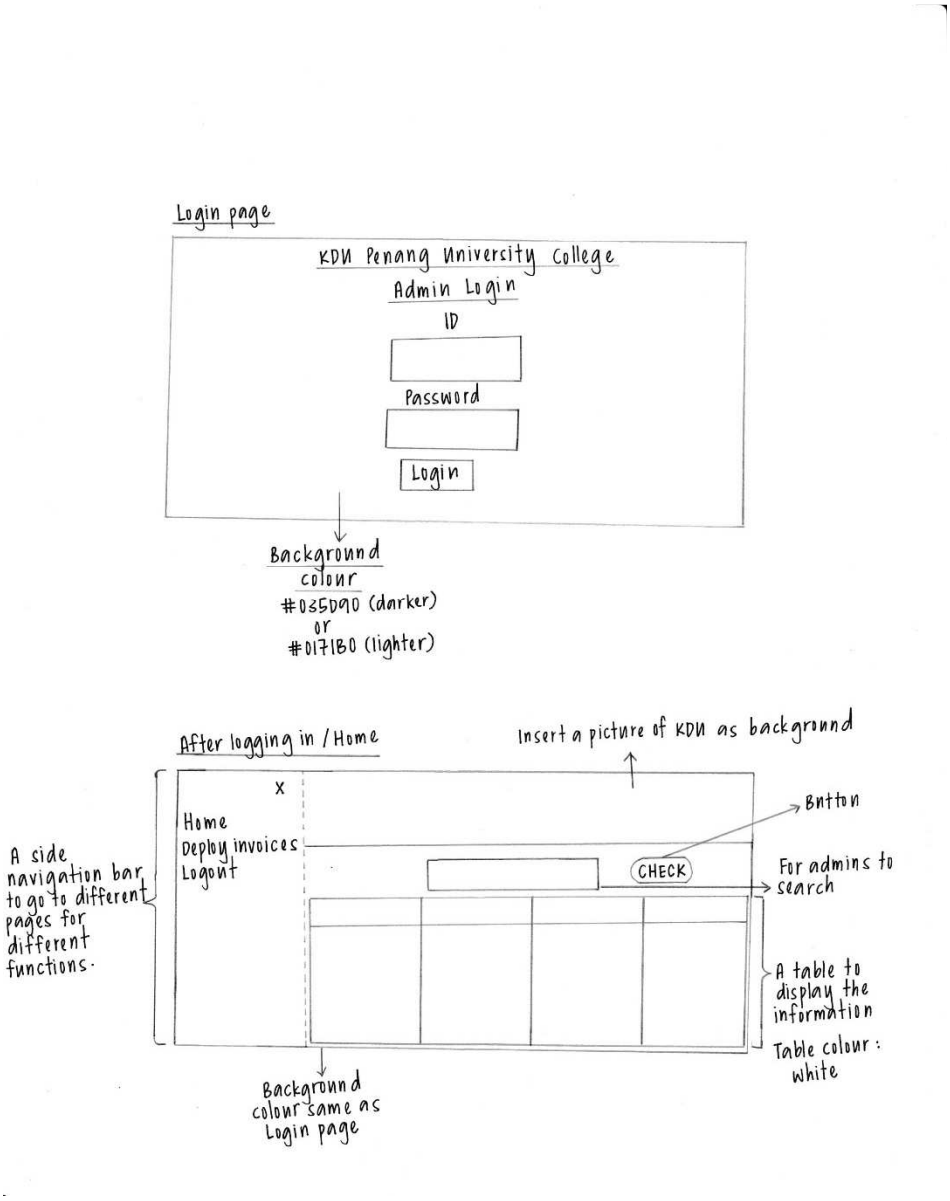


Diagram 4.3.1.1 Layout Sketch Design for Website

The above 2 sketch is the sketch for the login page of the website and the home page of the website after admins successfully log in. The login page was designed similar to the sketch above. For the home page after logging in, the team will be doing a side navigation bar on the left that can go to different pages when admins click on the tab functions they want. We will also be inserting a picture of KDU Penang University College at the header of the website. Other components for the home page are stated and drawn as per the sketch above. The team will be following this sketch closely as guidance when designing the front-end of the website.

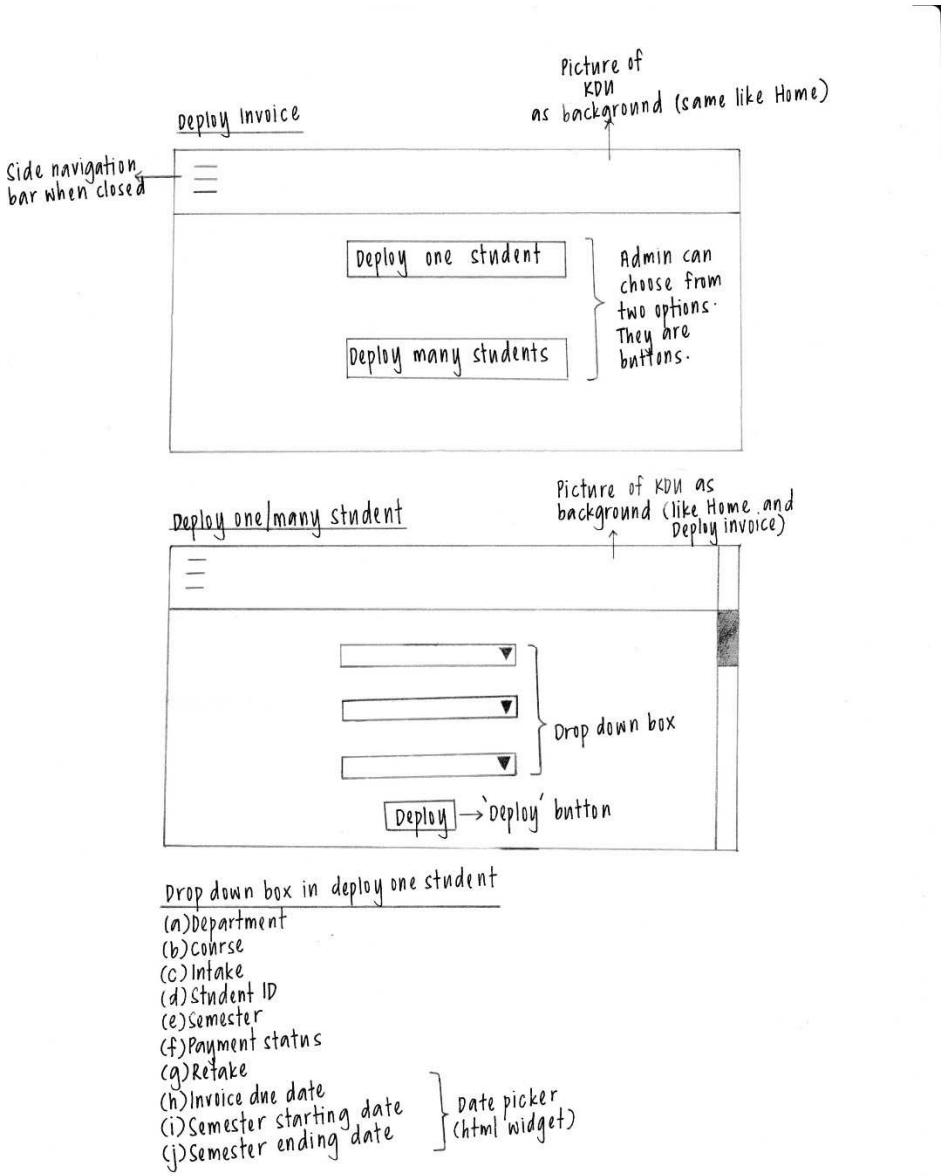


Diagram 4.3.1.2 Layout Sketch Design for Website

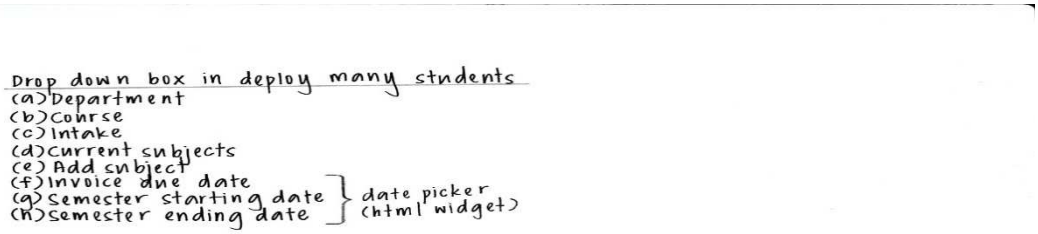


Diagram 4.3.1.3 Layout Sketch Design for Website

Next, the above 2 sketches are the sketch for the deploy invoice page of the website. The main page of the deploy invoice page has 2 buttons. The first button will lead to the page for deploying invoice for one student only while the second button will lead to the page for deploying invoice for many students. The next sketch shows the deploy one/many student page. Each function will have their own page but the two pages are similar in terms of the layout of the design. The only thing different between these 2 pages are the type of components that will be used for each page. Some of the drop-down box the team will be implementing for each page are listed in the comments.

4.3.2 Layout Sketch Design for Mobile

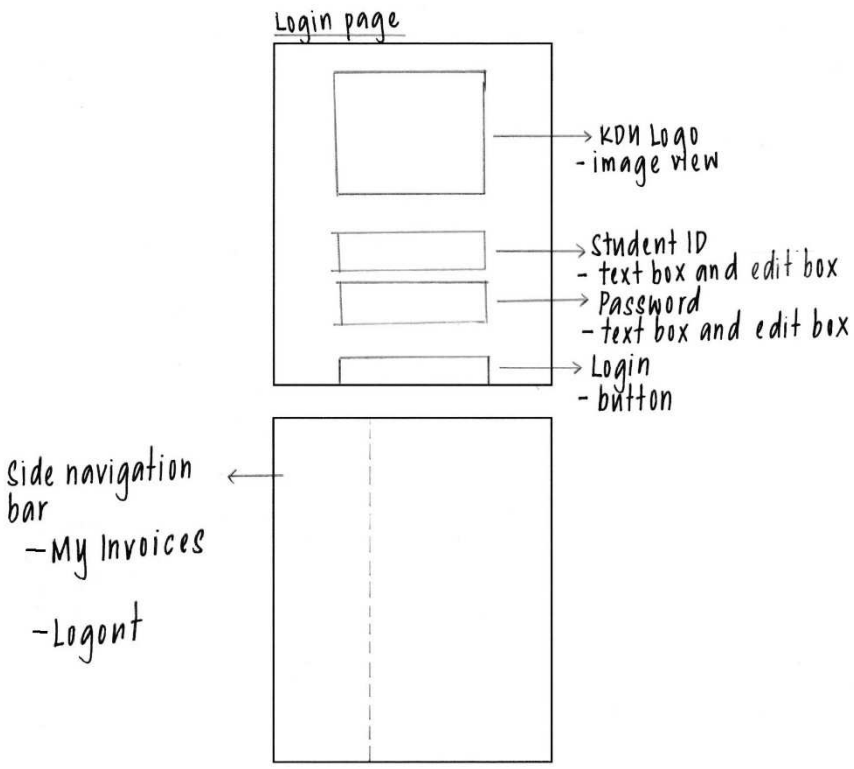


Diagram 4.3.2.1 Layout Sketch Design for Mobile

In the first sketch above for the login page, there will be 4 different user interface (UI) widgets being used. The image viewer is used to display the logo of the app There will be a pair of labels and followed by a pair of edit textboxes for the students to enter their username and password which are the necessary credentials for the students to login successfully. A login button is implemented at the end of the layout to trigger login function.

There will also be a side navigation bar which can be accessed by sliding the finger from left to right at the edge of the left area of the screen to slide in the menu from the left side. The menu will have 2 options which are My invoices and Logout. If they have been redirected to the payment page but wish to return to my invoices page, they can just slide the menu and select my invoices.

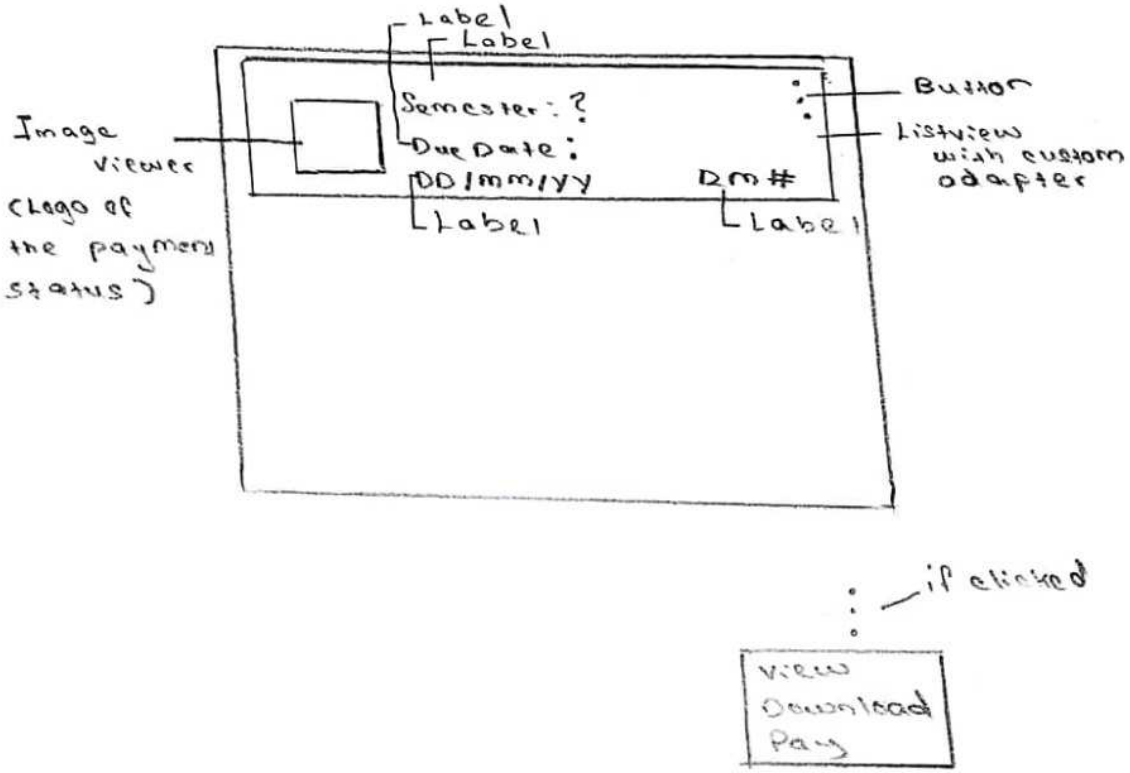


Diagram 4.3.2.2 Layout Sketch Design for Mobile

Under my invoices page there will be certain number of containers depending on the number of invoices that has been deployed containing information such as: -

a. Payment status whether the payment has been paid, pending or late

b. Semester No

c. Due date for the payment

d. The total amount that is due

Besides that, the container will also contain a symbol which represents three vertical dot which when pressed will pop out three options in a menu which are View, Download and Pay.

When the student presses view, they will be redirected to another page where the invoice will be viewed in the pdf version. On the other hand, if the users select download the pdf will be downloaded to their local storage. Finally, when the users click on pay, they will be redirected to pay layouts which is shown in the sketches below.

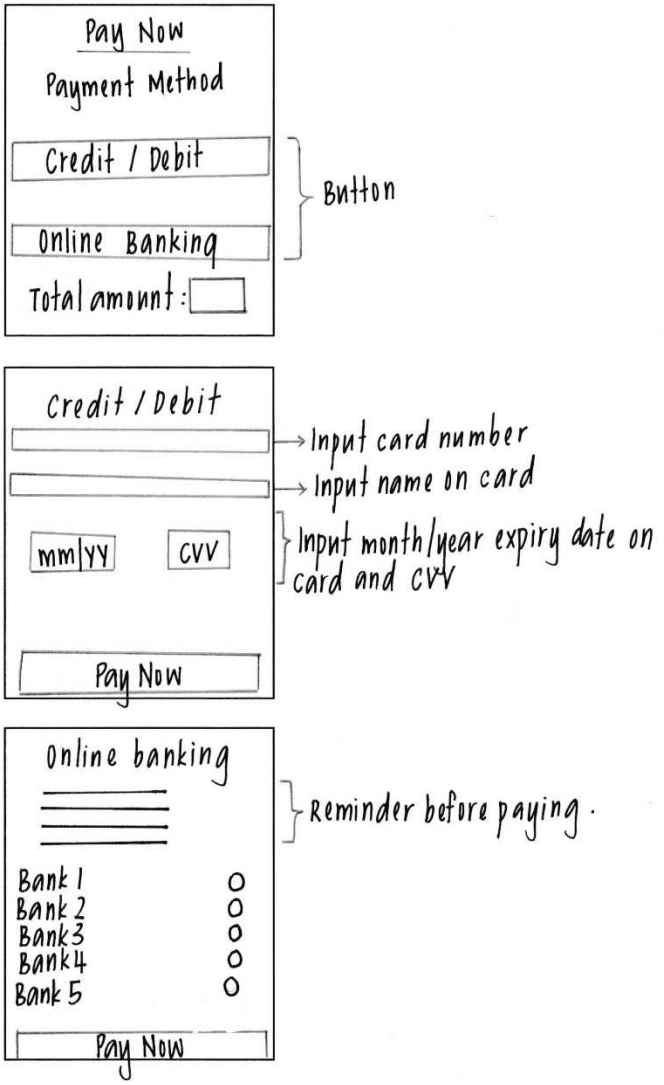


Diagram 4.3.2.3 Layout Sketch Design for Mobile

The pay now page will have an option for them to select whether they would like to complete the payment using Credit/Debit card or Online banking that is represented using buttons. Besides that, in the page the total amount that must be paid will be also shown again so that the students will know how much they will be charged.

If the students, select credit/debit option on the pay now page they will be redirected to this page which is shown in the next sketch where they are required to fill in the necessary information to verify the card details and to charge them if the details on their card is valid. Since our payment gateway system is fake, students can press pay now without filling in the details and will be redirected back to the sketch is Diagram 4.3.2.2.

Finally, if the students press on the online banking option they will be redirected to the page as shown in the last sketch above where they will be shown reminders before paying with their choice of bank which they will be using to carry out the transaction when real online payment gateway system is implemented.

Chapter 5: Implementation

In this section, the selection of programming language used will be covered. Then, significant section of codes that were important in implementing the functions and features of the mobile application and the website will be listed here with explanations on why the team implemented them in that manner in terms of integration methods. Finally, the installation settings that are needed to run this project will be shown as well.

5.1 Coding/Integration

5.1.1 Selection of programming language

For the selection of programming language, the team decided to code the back end of the website platform using PHP. PHP is a server-side scripting language designed for web development and since our database is located on the server it will be easier to communicate to the database using PHP programming language. HTML and CSS were used for front-end programming for the website together with JavaScript codes. For mobile development, Java was used as the main programming language as Android Studio, the software used to develop the mobile application is built on JetBrains' IntelliJ IDEA software, which is a Java integrated development environment (IDE). JSON was also used to transmit data between the server and the web application. Finally, MySQL was used to manage the database for this project.

5.1.2 Integration methods

In this section, the integration methods that were used to implement the codes for our system for both website and mobile application will be shown together with explanations on each screenshot.

5.1.2.1 Login (Website)

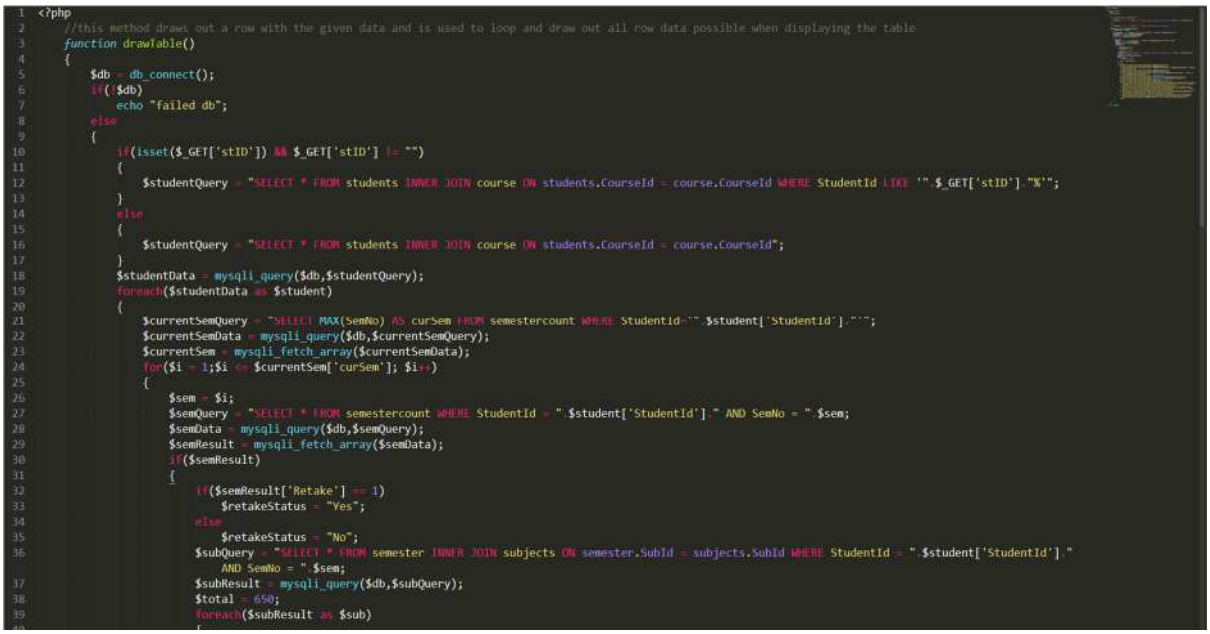
```php
<?php
//The initial page whr they have to login before using the functions of the website
    require("connection.php");
    $show_modal = false;
    if(isset($_POST['submit']))
    {
        $db = db_connect();
        if(!$db)
            echo "failed";
        else
        {
            $user = $_POST['userID'];
            $pw = $_POST['password'];
            $query = "SELECT * FROM admincredential WHERE AdminId = '$user' AND  AdminPwd = '$pw
                '"; //queries whether input by user is registered in the database
            $result = mysqli_query($db,$query);
            if(mysqli_num_rows($result) == 1)
            {
                $_SESSION["login"] = "set";
                if($_SESSION['login'] = "set")
                {
                    header("location:InformationModification.php");
                }
            }
            else
            {
                $header = "Invalid credentials";
                $footer = "Please try again";
                $show_modal = true;
            }
        }
    }
    if(isset($_GET['login']))
    {
        if($_GET['login'] = "not")
        {
            $header = "Not logged in";
            $footer = "Please login";
            $show_modal = true;
        }
    }
    if(isset($_GET['logout']))
    {
        if($_GET['logout'] = true)
        {
            $_SESSION['login'] = "unset";
        }
    }
?>
```

Screenshot 5.1.2.1.1

The flow of this page starts with users sending in their credentials after typing it in and pressing the submit button, the credentials are pulled through the $_POST method and then sent as variables to the SQL query to be checked against the server's database. If their credentials are correct and match the credentials in the database, they will be forwarded to the main page "InformationModification.php". With this the session will be set so that users do not have to login again when going back to the website's pages. If credentials are wrong, a message box will popup saying wrong credentials. Other than that, the bottom $_GET['login'] is triggered when a user tries to access the website without logging in, prompting a message

box saying to login. For $_GET[‘logout’], it is only triggered when the user has logged out to end the session.

5.1.2.2 Information Modification (Website)



```php
<?php
    //this method draws out a row with the given data and is used to loop and draw out all row data possible when displaying the table
    function drawTable()
    {
        $db = db_connect();
        if(!$db)
            echo "failed db";
        else
        {
            if(isset($_GET['stID']) && $_GET['stID'] != "")
            {
                $studentQuery = "SELECT * FROM students INNER JOIN course ON students.CourseId = course.CourseId WHERE StudentId LIKE '".$_GET['stID']."%'";
            }
            else
            {
                $studentQuery = "SELECT * FROM students INNER JOIN course ON students.CourseId = course.CourseId";
            }
            $studentData = mysqli_query($db,$studentQuery);
            foreach($studentData as $student)
            {
                $currentSemQuery = "SELECT MAX(SemNo) AS curSem FROM semestercount WHERE StudentId='".$student['StudentId']."'";
                $currentSemData = mysqli_query($db,$currentSemQuery);
                $currentSem = mysqli_fetch_array($currentSemData);
                for($i = 1;$i <= $currentSem['curSem']; $i++)
                {
                    $sem = $i;
                    $semQuery = "SELECT * FROM semestercount WHERE StudentId = ".$student['StudentId']." AND SemNo = ".$sem;
                    $semData = mysqli_query($db,$semQuery);
                    $semResult = mysqli_fetch_array($semData);
                    if($semResult)
                    {
                        if($semResult['Retake'] == 1)
                            $retakeStatus = "Yes";
                        else
                            $retakeStatus = "No";
                        $subQuery = "SELECT * FROM semester INNER JOIN subjects ON semester.SubId = subjects.SubId WHERE StudentId = ".$student['StudentId']."
                            AND SemNo = ".$sem;
                        $subResult = mysqli_query($db,$subQuery);
                        $total = 650;
                        foreach($subResult as $sub)
```

Screenshot 5.1.2.2.1

This code is taken from a php file called drawTable where a function called drawTable() is located. The drawTable() code is the main worker to building the table with all the data by using multiple SQL queries to pull the appropriate invoice data stored in different tables.

```
echo '<tr>
    <td style="text-align: center; vertical-align: middle;">'.$student['StudentId'].'</td>
    <td style="text-align: center; vertical-align: middle;">'.$semResult['SemNo'].'</td>
    <td style="text-align: center; vertical-align: middle;">'.$student['StudentFirstName'].' '.$student['StudentMidName'].' '.$student['StudentLastName'].'</td>
    <td style="text-align: center; vertical-align: middle;">'.$student['CourseName'].'</td>
    <td style="text-align: center; vertical-align: middle;">'.$student['IntakeDate'].'</td>
    <td style="text-align: center; vertical-align: middle;">'.date("d-m-Y",strtotime($semResult['SemStartDate'])).' - '.date("d-m-Y",strtotime($semResult['SemEndDate']))
        .'</td>
    <td style="text-align: center; vertical-align: middle;">'.$retakeStatus.'</td>
    <td style="text-align: center; vertical-align: middle;">'.date("d-m-Y",strtotime($semResult['PayStartDate'])).'</td>
    <td style="text-align: center; vertical-align: middle;">'.date("d-m-Y",strtotime($semResult['PayEndDate'])).'</td>
    <td style="text-align: center; vertical-align: middle;">'.$total.'</td>
    <td style="text-align: center; vertical-align: middle;">'.$semResult['PaymentStatus'].'</td>
    <td style="text-align: center; vertical-align: middle;"><p data-placement="top" data-toggle="tooltip" title="Edit"><button class="btn btn-primary btn-xs"
        data-title="Edit" data-toggle="modal" data-target="#edit" id="button-edit" onclick="editSem(\''.$student['StudentId'].'\','.$i.')" ><span class="glyphicon
        glyphicon-pencil"></span></button></p> </td>
    <td style="text-align: center; vertical-align: middle;"><p data-placement="top" data-toggle="tooltip" title="Delete"><button class="btn btn-danger btn-xs"
        data-title="Delete" data-toggle="modal" data-target="#delete" onclick="deleteSem(\''.$student['StudentId'].'\','.$i.')"><span class="glyphicon
        glyphicon-trash"></span></button></p></td>
    <td style="text-align: center; vertical-align: middle;"><p data-placement="top" data-toggle="tooltip" title="View"><button class="btn btn-primary btn-xs"
        data-title="View" data-toggle="modal" data-target="#view" ><span class="glyphicon glyphicon-eye-open" onclick="genPhp(\''.$student['StudentId'].'\','.$i.',\'view
        \')"></span></button></p></td>
    <td style="text-align: center; vertical-align: middle;"><p data-placement="top" data-toggle="tooltip" title="Download"><button class="btn btn-primary btn-xs"
        data-title="Download" data-toggle="modal" data-target="#download"><span class="glyphicon glyphicon-download" onclick="genPhp(\''.$student['StudentId'].'\','.$i.'
        ,\'download\')"></span></button></p></td>
    </tr>';
```

Screenshot 5.1.2.2.2

This part of the code is echoed using php, it builds a row of data into the appropriate columns based on the pulled data and is used repeatedly later on to load out and display all data inside the database.

```php
<?php
//The main page where all the student data is displayed with functions like delete, edit, view
include_once("connection.php");
include_once("drawTable.php");
$show_modal = false;
if(isset($_GET['updated']))
{
    if($_GET['updated'] == "updated")
    {
        $show_modal = true;
        $header = "Invoice update";
        $msg = "Invoice has been successfully updated";
    }
    else if($_GET['updated'] == "added")
    {
        $show_modal = true;
        $header = "Invoice added";
        $msg = "Invoice has been successfully added, proceed to add subjects manually";
    }
    else if($_GET['updated'] == "addedAll")
    {
        $show_modal = true;
        $header = "Mass Invoice Deploy";
        $msg = "Invoices successfully deployed!";
    }
}
if(isset($_SESSION['login']))
{
    if($_SESSION['login'] == "unset")
    {
        header("location:Login.php?login=not");
    }
}
else
{
    header("location:Login.php?login=not");
}


?>
```

Screenshot 5.1.2.2.3

Back on the main file InformationModification.php, the top of the file contains many if conditions to check where the user came from. It will display the appropriate message boxes based on where the user came from and the result of their actions. Besides that is also the session code to check for whether a user is logged in.

```
function deleteSem(studentId,semNo) //deleting a semester
{
    $.ajax({
            url:"deleteSem.php", //the page containing php script
            type: "post", //request type
            data: {studentId : studentId, semNo : semNo},
            success : function()
            {
                document.location.reload(true);
            }
        });
}

function editSem(studentId,semNo) //forward to editing a semester
{
    $('input#studentIdEdit').val(studentId);
    $('input#semNoEdit').val(semNo);
    $('#edit').submit();
}

function genPhp(studentId,semNo, type) //to generate the pdf
{
    $('input#studentIdPhp').val(studentId);
    $('input#semNoPhp').val(semNo);
    $('input#phpType').val(type);
    $('#pdf').submit();
}

function resubmitForm()
{
    $("#resubmit").submit();
}
```

Screenshot 5.1.2.2.4

The javascript/JQueries used in the bottom half of the page dictate how the page flows around. deleteSem() takes in data parameters and then sends a request to a PHP script to delete data from the server database, editSem() forwards the user to page where users can edit invoices. genPhp() is the function that forwards users to download or view PDFs of invoices. resubmitForm() is the function that runs when users are searching for a certain student's data.

5.1.2.3 Editing Invoices (Website)

```php
<?php
    //this sql here is to pull student data based on the student id and sem number given to load out
    //the whole php file in general is used to edit a certain invoice of a student
    include_once("connection.php");
    $db = db_connect();
    if(!$db)
        echo "failed";
    $studentId = $_GET['studentId'];
    $semNo = $_GET['semNo'];
    $studentQuery = "SELECT * FROM students INNER JOIN course ON students.CourseId=course.CourseId WHERE StudentId='".$studentId."'";
    $studentData = mysqli_query($db,$studentQuery);
    $student = mysqli_fetch_array($studentData);
    $semQuery = "SELECT * FROM semestercount WHERE StudentId = '".$studentId."' AND SemNo = ".$semNo;
    $semData = mysqli_query($db,$semQuery);
    $sem= mysqli_fetch_array($semData);
    $subQuery = "SELECT * FROM semester INNER JOIN subjects ON semester.SubId = subjects.SubId WHERE StudentId = '".$studentId."' AND SemNo = ".$semNo;
    $subResult = mysqli_query($db,$subQuery);
    $allSubQuery = "SELECT * FROM coursesubjects INNER JOIN subjects ON coursesubjects.SubId = subjects.SubId WHERE CourseId=".$student['CourseId']." OR
        CourseId=6";
    $allSubResult = mysqli_query($db,$allSubQuery);
    $total = 650;
if(isset($_SESSION['login']))
{
    if($_SESSION['login'] == "unset")
    {
        header("location:Login.php?login=not");
    }
}
else
{
    header("location:Login.php?login=not");
}
?>
```

Screenshot 5.1.2.3.1

Invoice editing's files start with loading in the specific invoice's data using SQL with the Student Id and Sem Number that was sent through the form GET method. This data is then displayed to the user in the appropriate layouts in the website.

```
function addSubject()
{
    var val = $("#subjectAdd").val();
    if($("#subjectsNow option[value='"+val+"']").length > 0)
        $("#popup-Modal").modal();
    else
    {
        $("#subjectsNow").append(new Option($("#subjectAdd :selected").text(), $("#subjectAdd").val()));
        var str = $("#subjectAdd :selected").text();
        var priceArr = str.split("-RM");
        total += parseInt(priceArr[1]);
        $("#PaymentAmount").text(total);
    }
}

function deleteSubject()
{
    var str = $("#subjectsNow :selected").text();
    var priceArr = str.split("-RM");
    total -= parseInt(priceArr[1]);
    $("#PaymentAmount").text(total);
    var val = $("#subjectsNow").val();
    $("#subjectsNow option[value='"+val+"']").remove();
}

function updateData()
{
    selectBox = document.getElementById("subjectsNow");
    var no = 0;
    for (var i = 0; i < selectBox.options.length; i++)
    {
        no++;
         $("#subject"+(i+1)).val(selectBox.options[i].value);
    }
    $("#SubjectTotal").val(no);
    $("#editForm").submit();
}
```

Screenshot 5.1.2.3.2

In the javascript/JQuery section, addSubject() is used to add subjects into the list in the website and has built in error messages for situations like when added subjects are being added again. Besides that, it also calculates the outstanding total of all the subjects including the resource fees. In deleteSubject() it does the reverse of addSubject() and deletes the subject from the list whilst also subtracting the amount from the outstanding total. The last one updateData() is what runs when users are done with editing the data and pulls the data out of the inputs in the website to send it over to the php script through the form submit.

```php
<?php
    //The php script called when Edit Invoice's form has been submitted to update a students invoice
    include_once("connection.php");
    $db = db_connect();
    $semNo = $_GET["SemNo"];
    $studentId = $_GET["StudentId"];
    $payStartDate = $_GET["InvoiceDate"];
    $payEndDate = $_GET["PaymentDate"];
    $semStartDate = $_GET["SemesterStart"];
    $semEndDate = $_GET["SemesterEnd"];
    $retakeStatus = $_GET["RetakeStatus"];
    $subjects = [];
    $PaymentStatus = $_GET["PaymentStatus"];
    for($i = 0; $i < $_GET['SubjectTotal']; $i++)
    {
        $subjects[] = $_GET['subject'.($i+1)];
    }
    $deleteSubQuery = "DELETE FROM semester WHERE StudentId='".$studentId."' AND SemNo=" $semNo;
    if(mysqli_query($db,$deleteSubQuery))
        echo "pass";
    for($i = 0; $i < count($subjects); $i++)
    {
        $addSubQuery = "INSERT INTO semester(StudentId, SubId, SemNo)
                        VALUES ('".$studentId."',".$subjects[$i].",".$semNo.")";

        if(mysqli_query($db,$addSubQuery))
            echo "pass2";
    }

    $updateSemQuery = "UPDATE semestercount
                       SET StudentId='".$studentId."',SemNo=".$semNo.",SemStartDate='".$semStartDate."',SemEndDate='".$semEndDate.
                       "',PayStartDate='".$payStartDate."',PayEndDate='".$payEndDate."',Retake=".$retakeStatus.",PaymentStatus='".$PaymentStatus."'
                       WHERE StudentId='".$studentId."' AND SemNo=".$semNo;
    if(mysqli_query($db,$updateSemQuery))
        header("location:InformationModification.php?updated=updated");
?>
```

Screenshot 5.1.2.3.3

This is the script that is fired when the form is submitted in the Edit Invoice page. It pulls all the data that was changed in the Edit Invoice page through the $_GET method and then builds SQL queries around them to be used to update the data inside the server database. After all is done it will send the user back to the InformationModification page where they can see the message on whether invoices were updates successfully.

5.1.2.4 Deleting Invoices (Website)

```php
<?php
    include_once("connection.php");
    $db = db_connect();
    if(isset($_POST['studentId']))
    {
        $stId = $_POST['studentId']; //pulls student id chosen
        $semNo = $_POST['semNo']; //pulls semester number chosen
        $deleteSemCountQuery = "DELETE FROM semestercount WHERE SemNo=".$semNo." AND StudentId='".$stId."'"; //sql query to delete chosen row from Invoice counts
        $deleteSemQuery = "DELETE FROM semester WHERE SemNo=" $semNo." AND StudentId='".$stId."'"; //delete all subjects related to the semester
        if(mysqli_query($db,$deleteSemCountQuery))
        {
            echo "1st del success";
        }
        if(mysqli_query($db,$deleteSemQuery))
        {
            echo json_encode(array("success"=>'successfuly registered')); //returns a message if both successful
        }
    }
?>
```

Screenshot 5.1.2.4.1

Deleting invoices has no UI and is fired using the deleteSem() function in InformationModification. Using the data sent over through the POST request, SQL queries are built around them then sent to the database server to delete the corresponding semester subjects and the semester data. A message is sent back in JSON to show whether deleting was successful.

5.1.2.5 View and Download PDF (Website)

```
<?php
require('fpdf17/fpdf.php');
//The pdf generator file
include_once("connection.php");
$db = db_connect();

//get invoices data
$query = mysqli_query($db,"SELECT * FROM students
        INNER JOIN course
        ON students.CourseId = course.CourseId
        WHERE
        StudentId = '".$_GET['studentId']."'");
$invoice = mysqli_fetch_array($query);

$query = mysqli_query($db,"SELECT * FROM semesterCount
        WHERE
        StudentId = '".$invoice['StudentId']."' AND
SemNo=".$_GET['semNo']);
$semesterCount = mysqli_fetch_array($query);
$amount = 0;
//A4 width : 219mm
//default amrgin : 10mm each side
//writable horizontal : 219-(10*2) = 189mm

$pdf = new FPDF('P','mm','A4');

$pdf->AddPage();

//set font to arial,bold,8pt , Title
//Cell(width,height,text,border,end line, [align](left is L, C is center,R
is right))

$pdf ->SetFont('Arial','B',6);
$pdf ->Cell (100,3,'',0,0);
$pdf ->Cell (89,3,'KDU University College (PG) Sdn Bhd (879357-
X)',0,1,'R');

$pdf ->SetFont('Arial','',6);
$pdf ->Cell (79,3,'',0,0);
$pdf ->Cell (110,3,'32, Jalan Anson,10400 Georgetown, Pulau
Pinang,Malaysia',0,1,'R');
$pdf ->Cell (59,3,'',0,0);
$pdf ->Cell (130,3,'Tel :04-2386368 Fax:04-2280362 Email:best@kdupg.edu.my
Website:kdupg.edu.my',0,1,'R');
```

```
$pdf ->Cell
(189,3,'_____
_____
_____',0,1);

$pdf ->SetFont('Arial','',10);
$pdf ->Cell (189,2,'',0,1);
$pdf ->Cell (189,5,'INVOICE',0,1,'R');
$pdf ->Cell (189,2,'',0,1);

$pdf ->Cell (100,5,$invoice['StudentFirstName'].'
'.$invoice['StudentMidName'].' '.$invoice['StudentLastName'],0,1);

$pdf ->Cell (100,5,'',0,0);
$pdf ->Cell (25,5,'Invoice Date',0,0);
$pdf ->Cell (5,5,' : ',0,0);
$pdf ->Cell (59,5,date("d-m-
Y",strtotime($semesterCount['PayStartDate'])),0,1);

$pdf ->Cell (100,5,'',0,0);
$pdf ->Cell (25,5,'Student No',0,0);
$pdf ->Cell (5,5,' : ',0,0);
$pdf ->Cell (59,5,$invoice['StudentId'],0,1);

$pdf ->Cell (100,5,'',0,0);
$pdf ->Cell (25,5,'Programme',0,0);
$pdf ->Cell (5,5,' : ',0,0);
$pdf ->Cell (59,5,$invoice['CourseName'],0,1);

$pdf ->Cell (100,5,'',0,0);
$pdf ->Cell (25,5,'Intake',0,0);
$pdf ->Cell (5,5,' : ',0,0);
$pdf ->Cell (59,5,$invoice['IntakeDate'],0,1);

$pdf ->Cell (100,5,'',0,0);
$pdf ->Cell (25,5,'Semester',0,0);
$pdf ->Cell (5,5,' : ',0,0);
$pdf ->Cell (59,5,date("d-m-Y",strtotime($semesterCount['SemStartDate'])).'
-  '.date("d-m-Y",strtotime($semesterCount['SemEndDate'])),0,1);

$pdf ->Cell (189,5,'',0,1);


$pdf ->Cell (159,5,'Item Description',1,0);
$pdf ->Cell (30,5,'Amount (RM)',1,1,'R');

$query = mysqli_query($db,"SELECT *
                FROM semester
                INNER JOIN subjects
                ON semester.SubId = subjects.SubId
                WHERE semester.StudentId ='".$invoice['StudentId']."' AND
semester.SemNo=".$_GET['semNo']);


while ($items = mysqli_fetch_array($query))
{
        $pdf ->Cell (159,5,$items['SubCode'].' - '.$items['SubName'],1,0);
        $pdf ->Cell (30,5,$items['SubPrice'],1,1,'R');
        $amount += $items['SubPrice'];
}
```

```
$pdf ->Cell (159,5,'Resource Fee',1,0);
$pdf ->Cell (30,5,'650',1,1,'R');
$amount += 650;

$pdf ->Cell (159,5,'Total Amount',1,0,'R');
$pdf ->Cell (30,5,$amount,1,1,'R');

$pdf ->Cell (189,15,'',0,1);

$pdf ->Cell(94.5,5,'Payment Due Date',1,0,'C');
$pdf ->Cell(94.5,5,'Payment Due Amount (RM)',1,1,'C');

$pdf ->Cell(94.5,5,date("d-m-
Y",strtotime($semesterCount['PayEndDate'])),1,0,'C');
$pdf ->Cell(94.5,5,$amount,1,1,'C');

$pdf ->Cell (189,5,'',0,1);

$pdf ->SetFont('Arial','B',6);
$pdf ->Cell (189,3,'Payment of Fees',0,1);

$pdf ->SetFont('Arial','',6);
$pdf ->Cell (189,3,'Payment can be made via the following in favour of KDU
University College(PG) Sdn. Bhd',0,1);
$pdf ->Cell (5,3,'1',0,0);
$pdf ->Cell (184,3,'Online banking / Bill payment (Maybank2u, CIMBClicks,
RHBNow)',0,1);

$pdf ->Cell (5,3,'2',0,0);
$pdf ->Cell (184,3,'Cash / crossed cheque/ bankers cheque',0,1);

$pdf ->Cell (5,3,'3',0,0);
$pdf ->Cell (184,3,'Debit card/ Credit card (Visa/Master/Amex)',0,1);

$pdf ->Cell (5,3,'4',0,0);
$pdf ->Cell (184,3,'Local online/telegraphic transfer and cash deposit
machine shall be made to the following accounts ONLY',0,1);

$pdf ->Cell (189,3,'',0,1);
$pdf->Cell (94.5,3,'Malaysian Students',1,0);
$pdf->Cell (94.5,3,'International Students',1,1);
$pdf->Cell (94.5,3,'Maybank (A/C No :507013013331/CIMB (A/C
No:8601003506)',1,0);
$pdf->Cell (94.5,3,'Citibank(A/C No:0165148002',1,1);

$pdf ->Cell (189,3,'',0,1);

$pdf ->SetFont('Arial','B',6);
$pdf ->Cell (189,3,'Payments using credit card or online banking can be
carried out using this app or can be performed at the counters in the
bursary',0,1);
$pdf ->Cell (189,3,'',0,1);
$pdf ->Cell (189,3,'Please fax a copy of payment advice together with
Student Number, Name and Contact Number to the Bursarys Office at (604)227-
6368 ',0,1);
$pdf ->Cell (189,3,'or email payment @kdupg.edu.my for transcation which
are carried out using the deposit machines ',0,1);

$pdf ->Cell (189,3,'',0,1);
$pdf ->Cell (189,3,'A late payment penalty of RM10 per day (including
Saturdays,Sundays, and Public holidays) ',0,1);
```

```
$pdf ->Cell (189,3,'will be imposed on ALL outstanding fees after the
payment due date stated in the invoice. ',0,1);

$pdf ->Cell (189,3,'',0,1);
$pdf ->Cell (189,3,'If fees remain unpaid,students will be barred from
suing the KDU University College facilities, ',0,1);
$pdf ->Cell (189,3,'classes and examinations and may be terminated from
their studies ',0,1);

$pdf ->Cell (189,3,'',0,1);
$pdf ->Cell (189,3,'Bursary operating hours 9am to 5.30pm (Mon-Fri,except
Public Holidays) ',0,1);
if($_GET['phpType'] == "view")
{
        $pdf->Output("invoice.pdf","I");
}
else
{
        $pdf->Output("invoice.pdf","D");
}
?>
```

Code Block 5.1.2.5.1

As the nature of the code is long, a formatted code is used instead of screenshots. For the View and Download parts we generate a PDF based on the button clicked which can be checked through the phpType variable sent through. The PDF is generated using the FPDF library and all the formatting is done through the code above. At the end based on the phpType variable, a downloaded pdf or an inline pdf is outputted to the user.

5.1.2.6 Deploying Invoices (Website)

```
<form method="get" action="DeployOne.php">
        <input type="submit" class="buttonCheck" value="Deploy One Student">
    </form>

    <form method="get" action="DeployAll.php">
        <input type="submit" class="buttonCheck" value="Deploy Many Students">
    </form>
```

Screenshot 5.1.2.6.1

Deploying invoices begins with a page that has 2 buttons in it that users can choose which choice they want between deploying only one student or many students.

```
var deptsAndCourses = {};
var coursesAndIntakes = {};
<?php
    $db = db_connect();
    $departmentQuery = "SELECT * FROM department";
    $departmentResult = mysqli_query($db,$departmentQuery);
    while($row = mysqli_fetch_array($departmentResult))
    {
        $courseQuery = "SELECT * FROM course WHERE DeptId=".$row['DeptId']; //load out all Department Course and Intake related data
        $courseResult = mysqli_query($db,$courseQuery);
        echo "deptsAndCourses['".$row['DeptId']."']=[";
        while($courseRow = mysqli_fetch_array($courseResult))
        {
            echo "'".$courseRow['CourseName']."',";
        }
        echo "];
";
    }

    $courseQuery = "SELECT * FROM course";
    $courseResult = mysqli_query($db,$courseQuery);
    while($row = mysqli_fetch_array($courseResult))
    {
        $intakeQuery = "SELECT DISTINCT(IntakeDate) AS IntakeDate FROM students WHERE CourseId=" $row['CourseId'] " ORDER BY IntakeDate";
        $intakeResult = mysqli_query($db,$intakeQuery);
        echo "coursesAndIntakes['".$row['CourseName']."']=[";
        while($intakeRow = mysqli_fetch_array($intakeResult))
        {
            echo "'".$intakeRow['IntakeDate']."',";
        }
        echo "];
";
    }

?>
```

Screenshot 5.1.2.6.2

For deploying only one student, a set of data is first echoed out before anything begins. An array built around departments as its columns and the courses in it as its row is made using the upper half of the above PHP code. The bottom half echoes out an array built around courses as its columns and the intakes as its rows. This is used later on to load out choices based on what the user chooses in the provided dropdown lists. An example would be deptsAndCourses[“Computing”] = [“Diploma in Computer Studies” , “Diploma in Games Technology”];

```
function changeCourseList()
{
    var deptList = document.getElementById("deptDropdown");
    var courseList = document.getElementById("courseDropdown");
    if(deptList.options[0].value == "")
    {
        deptList.remove(0);
    }
    var selDept = deptList.options[deptList.selectedIndex].value;
    while (courseList.options.length) {
        courseList.remove(0);
    }
    var courses = deptsAndCourses[selDept];
    if (courses)
    {
        var first = new Option("Choose One", "");
        courseList.options.add(first);
        var i;
        for (i = 0; i < courses.length; i++)
        {
            var course = new Option(courses[i], courses[i]);
            courseList.options.add(course);
        }
    }
}
```

Screenshot 5.1.2.6.3

changeCourseList() is assigned to the onChange event of department's dropdown list and runs when users have selected a department. It will load out options into the courses dropdown list based on the choice selected in the department dropdown.

```
function changeIntakeList()
{
    var courseList = document.getElementById("courseDropdown");
    var intakeList = document.getElementById("intakeDropdown");
    if(courseList.options[0].value == "")
    {
        courseList.remove(0);
    }
    var selCourse = courseList.options[courseList.selectedIndex].value;
    while (intakeList.options.length) {
        intakeList.remove(0);
    }
    var intakes = coursesAndIntakes[selCourse];
    if (intakes)
    {
        var first = new Option("Choose One", "");
        intakeList.options.add(first);
        var i;
        for (i = 0; i < intakes.length; i++)
        {
            var intake = new Option(intakes[i], intakes[i]);
            intakeList.options.add(intake);
        }
    }
}
```

Screenshot 5.1.2.6.4

changeIntakeList() is another version of the function mentioned above, it is instead assigned to the onChange event of the course dropdown list and runs when the user has selected a course. It loads out the intakes currently taking the selected course into the intake dropdown list.

```
function changeStudentList()
{
    var courseList = document.getElementById("courseDropdown");
    var intakeList = document.getElementById("intakeDropdown");
    var studentList = document.getElementById("studentDropdown");
    if(intakeList.options[0].value == "")
    {
        intakeList.remove(0);
    }
    $.ajax({
            url:"loadStudents.php", //the page containing php script
            type: "get", //request type
            data: {course : courseList.options[courseList.selectedIndex].value, intake: intakeList.options[intakeList.selectedIndex].value},
            success : function(data)
            {
                while (studentList.options.length) {
                    studentList.remove(0);
                }
                var count = Object.keys(data).length;
                var i;
                var first = new Option("Choose One", "");
                studentList.options.add(first);
                for(i = 0;i < count;i++)
                {
                    var id = new Option(data[i],data[i]);
                    studentList.options.add(id);
                }
            }
        });
}
```

Screenshot 5.1.2.6.5

As the nature of loading all the student ids corresponding to the selected choice is different, ajax is used to call a php script instead that will return student ids corresponding to the chosen department, course and intake through JSON then loaded into the student id dropdown list.

```
function deployForm()
{
    if($("#PayEndDate").val() == "")
        $("#date-Modal").modal();
    else if($("#SemStartDate").val() == "")
        $("#date-Modal").modal();
    else if($("#SemEndDate").val() == "")
        $("#date-Modal").modal();
    else if($("#paystatDropdown").val() == "" || $("#semesterDropdown").val() == "")
        $("#input-Modal").modal();
    else
        $("#deployForm").submit();
}
```

Screenshot 5.1.2.6.6

deployForm() is the last function called on the Deploy One page and checks if all the required inputs have been filled, sending out corresponding message boxes if they aren't. Once all inputs have been filled, it will submit the form that will lead to the script to deploy the invoice.

```
<?php
    //takes in a student's id, and all the given data for an invoice to manually add a semester
    include_once("connection.php");
    $db = db_connect();
    $StudentId = $_GET['Student'];
    $SemNo = $_GET['Semester'];
    $semCheckQuery = "SELECT SemNo FROM semestercount WHERE StudentId =\"".$StudentId."\" AND SemNo =" $SemNo;
    $semCheckResult = mysqli_query($db,$semCheckQuery);
    if(mysqli_num_rows($semCheckResult) > 0)
    {
        header("location:DeployOne.php?exists=exists");
    }
    $RetakeStatus = $_GET['Retake'];
    $PayEndDate = $_GET['datePay'];
    $PayStartDate = date("Y-m-d");
    $SemStartDate = $_GET['dateStart'];
    $SemEndDate = $_GET['dateEnd'];
    $PaymentStatus = $_GET['paystat'];
    $addSemQuery = "INSERT INTO semestercount (StudentId,SemNo,SemStartDate,SemEndDate,PayStartDate,PayEndDate,Retake,PaymentStatus)
                    VALUES ('".$StudentId."',".$SemNo.",'".$SemStartDate."','".$SemEndDate."','".$PayStartDate."','".$PayEndDate."',".$RetakeStatus.",'".$PaymentStatus."')";
    if(mysqli_query($db,$addSemQuery))
    {
        header("location:InformationModification.php?updated=added");
    }
?>
```

Screenshot 5.1.2.6.7

This is the script deployForm() from Deploy One leads to and takes all the data from the Deploy One form inputs through $_GET methods then builds an INSERT query around them before running it against the server database. Once successful it will forward the user towards the InformationModification page where they can see the corresponding success message and view their newly inserted data.

```
function changeSubjectList()
{
    var courseList = document.getElementById("courseDropdown"); //changes the subjects once an intake is selected
    var subjectList = document.getElementById("subjectDropdown");
    var intakeList = document.getElementById("intakeDropdown");
    $.ajax({
            url:"loadSubjects.php", //the page containing php script
            type: "get", //request type
            data: {course : courseList.options[courseList.selectedIndex].value, intake: intakeList.options[intakeList.selectedIndex].value},
            success : function(arr)
            {
                while (subjectList.options.length)
                {
                    subjectList.remove(0);
                }
                var count = Object.keys(arr).length;
                var i;
                var first = new Option("Choose One", "");
                subjectList.options.add(first);
                for(i = 0; i < count; i++)
                {
                    var subject = new Option(arr[i].SubCode +"-"+ arr[i].SubName +"-RM"+ arr[i].SubPrice, arr[i].SubId);
                    subjectList.options.add(subject);
                }
            }
        });
}
```

Screenshot 5.1.2.6.8

In the Deploy All part of the Deploy Invoices, all the codes are similar except in Deploy All instead of selecting student id, there is the choice to add and delete subjects. changeSubjectList() takes the place of changeStudentList() in the intake dropdown onChange, loading subjects that the can be taken by the course into the subject dropdown.

```
function addSubject()
{
    var val = $("#subjectDropdown").val();
    if($("#subjectsNowList option[value='"+val+"']").length > 0) //used to add a subject to the list
        $("#popup-Modal").modal();
    else
    {
        if(!val)
        {

            $("#subject-Modal").modal();
            return;
        }
        $("#subjectsNowList").append(new Option($("#subjectDropdown :selected").text(), $("#subjectDropdown").val()));
        var str = $("#subjectDropdown :selected").text();
        var priceArr = str.split("-RM");
        total += parseInt(priceArr[1]);
        $("#PaymentAmount").text(total);
    }
}

function deleteSubject()
{
    var str = $("#subjectsNowList :selected").text();
    if(str == "")
    {
        return;
    } //vice versa delete a subject from list once selected
    var priceArr = str.split("-RM");
    total -= parseInt(priceArr[1]);
    $("#PaymentAmount").text(total);
    var val = $("#subjectsNowList").val();
    $("#subjectsNowList option[value='"+val+"']").remove();
}
```

Screenshot 5.1.2.6.9

Similar to the Edit Invoice. There is addSubject() and deleteSubject() functions to manage the subject list and dropdown in Deploy All.

```php
<?php
    //Will take in Intake,Course and Subjocct data to mass deploy invoices
    include_once("connection.php");
    $db = db_connect();
    $intake = $_GET['Intake'];
    $semCheckQuery = "SELECT MAX(SemNo) AS CurrentSem FROM semestercount
                      INNER JOIN students ON semestercount.StudentId = students.StudentId
                      INNER JOIN course ON students.CourseId = course.CourseId
                      WHERE students.IntakeDate ='".$intake."' AND course.CourseName='".$_GET['Course']."'";
    $semCheckResult = mysqli_query($db,$semCheckQuery);
    $row = mysqli_fetch_array($semCheckResult);
    $currentSem = $row['CurrentSem'];
    $nextSem = (int)$currentSem + 1;
    $PayEndDate = $_GET['datePay'];
    $Course = $_GET['Course'];
    $PayStartDate = date("Y-m-d");
    $SemStartDate = $_GET['dateStart'];
    $SemEndDate = $_GET['dateEnd'];
    $Subjects = array($_GET['subject1'],$_GET['subject2'],$_GET['subject3'],$_GET['subject4'],$_GET['subject5'],$_GET['subject6']);
    $CourseIdQuery = "SELECT CourseId FROM course WHERE CourseName='".$Course."'";
    $CourseIdResult = mysqli_query($db,$CourseIdQuery);
    $courseRow = mysqli_fetch_array($CourseIdResult);
    $CourseId = $courseRow['CourseId'];
    $studentIntakeQuery = "SELECT StudentId FROM students WHERE IntakeDate='".$intake."' AND CourseId=".$CourseId;
    $studentIntakeResult=mysqli_query($db,$studentIntakeQuery);
    while($student = mysqli_fetch_array($studentIntakeResult))
    {
        $addSemQuery = "INSERT INTO semestercount (StudentId,SemNo,SemStartDate,SemEndDate,PayStartDate,PayEndDate,Retake,PaymentStatus)
                VALUES ('".$student['StudentId']."',".$nextSem.",'".$SemStartDate."','".$SemEndDate."','".$PayStartDate."','".$PayEndDate."',0,'PENDING')";
        mysqli_query($db,$addSemQuery);
        foreach($Subjects as $sub)
        {
            if($sub != "")
            {
                $addSubQuery = "INSERT INTO semester (StudentId,SemNo,SubId)
                        VALUES ('".$student['StudentId']."',".$nextSem.",".$sub.")";
                mysqli_query($db,$addSubQuery);
            }
        }
    }
    header("location:InformationModification.php?updated=addedAll");
?>
```

Screenshot 5.1.2.6.10

The script for deploying mass invoices is similar except the fact that it loops based on how many students are within the selected course and intake using the MAX() sql function to query how many students it has to loop through.

5.1.2.7 Network Check (Mobile)

```
if ( conMgr.getNetworkInfo(ConnectivityManager.TYPE_MOBILE).getState() == NetworkInfo.State.CONNECTED
        || conMgr.getNetworkInfo(ConnectivityManager.TYPE_WIFI).getState() == NetworkInfo.State.CONNECTED ) {

    Intent intent = new Intent( packageContext: this,LoginActivity.class);
    startActivity(intent);
    finish();


}
else if ( conMgr.getNetworkInfo(ConnectivityManager.TYPE_MOBILE).getState() == NetworkInfo.State.DISCONNECTED
        || conMgr.getNetworkInfo(ConnectivityManager.TYPE_WIFI).getState() == NetworkInfo.State.DISCONNECTED) {

    AlertDialog.Builder builder;
    builder = new AlertDialog.Builder( context: this);
    builder.setTitle("No Internet Access Detected")
            .setMessage("We are sorry but there seems to be no internet connection right now." +
                    " Please try again later")
            .show();
}
```

Screenshot 5.1.2.7.1

On first launch the first activity that runs is the NetworkActivity that runs a network check on whether a user is connected to internet as the app requires internet connection for almost all its functions. If they do not have internet connectivity, they will be barred from entering the app with a dialog telling them to restart.

5.1.2.8 Login (Mobile)

```
SharedPreferences sp = this.getSharedPreferences( name: "data",MODE_PRIVATE);
String id = sp.getString( key: "id", defValue: null);
if(id != null)
{
    Intent intent = new Intent( packageContext: this,LoadingActivity.class);
    intent.putExtra( name: "id",id);
    startActivity(intent);
    finish();
}
else
{
    ReceivedAlarm alarm = new ReceivedAlarm();
    alarm.cancelAlarm( context: this);
    AlertDialog alertDialog = new AlertDialog.Builder( context: this).create();
    alertDialog.setTitle("IMPORTANT NOTES");
    alertDialog.setMessage("This app does not support late payments, all late payments should be done at the Bursary");
    alertDialog.show();
}
```

Screenshot 5.1.2.8.1

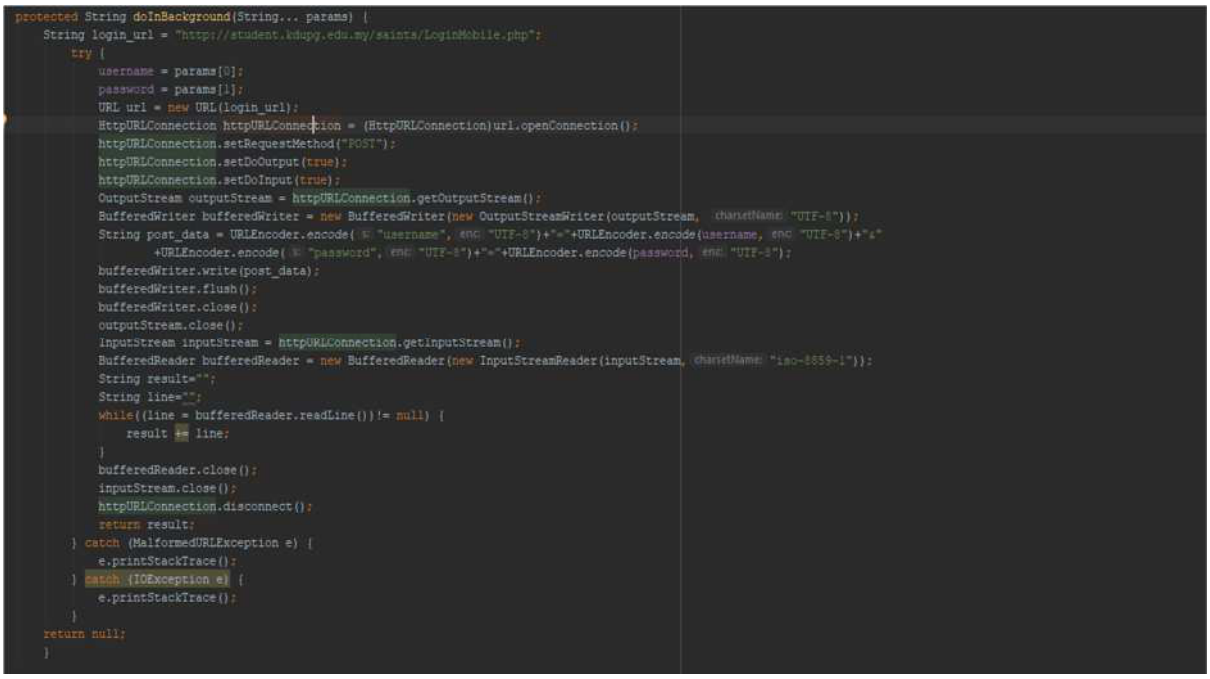
After the network check we enter the LoginActivity, the moment the activity launches, it will check whether a previous user has already logged in using SharedPreferences. If there was a logged user, it will skip the checks and move straight into the app, else it will put out a

dialog giving a notice to potential first time users about the app. The alarm is a routine check for new invoices and the routine check is cancelled here as no logged users means no there is no need for a routine check.

```
public void loadInvoices(String username,String password){
    String id = username.replace( target: " ", replacement: "");
    String pw = password.replace( target: " ", replacement: "");
    if(id.length() > 0 && pw.length() > 0)
    {
        LoginBackground login = new LoginBackground( ctx: this);
        login.execute(username,password);
    }
    else
    {
        Toast.makeText( context: this, text: "ID or Password is empty! Please try again",Toast.LENGTH_SHORT).show();
    }
}
```

Screenshot 5.1.2.8.2

loadInvoices() is a function in LoginActivity that will run when the login button is pressed, it will trim the inputs of white spaces and check whether it is empty, if empty it will put out an error message saying inputs are empty. When the inputs are legible it will send the values to a background task to check whether it's a legitimate credentials.



```
protected String doInBackground(String... params) {
    String login_url = "http://student.kdupg.edu.my/saints/LoginMobile.php";
        try {
            username = params[0];
            password = params[1];
            URL url = new URL(login_url);
            HttpURLConnection httpURLConnection = (HttpURLConnection)url.openConnection();
            httpURLConnection.setRequestMethod("POST");
            httpURLConnection.setDoOutput(true);
            httpURLConnection.setDoInput(true);
            OutputStream outputStream = httpURLConnection.getOutputStream();
            BufferedWriter bufferedWriter = new BufferedWriter(new OutputStreamWriter(outputStream,  charsetName: "UTF-8"));
            String post_data = URLEncoder.encode( s: "username", enc: "UTF-8")+"="+URLEncoder.encode(username, enc: "UTF-8")+"&"
                    +URLEncoder.encode( s: "password", enc: "UTF-8")+"="+URLEncoder.encode(password, enc: "UTF-8");
            bufferedWriter.write(post_data);
            bufferedWriter.flush();
            bufferedWriter.close();
            outputStream.close();
            InputStream inputStream = httpURLConnection.getInputStream();
            BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(inputStream, charsetName: "iso-8859-1"));
            String result="";
            String line="";
            while((line = bufferedReader.readLine())!= null) {
                result += line;
            }
            bufferedReader.close();
            inputStream.close();
            httpURLConnection.disconnect();
            return result;
        } catch (MalformedURLException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    return null;
    }
```

Screenshot 5.1.2.8.3

The background task will send data through POST method and OutputStream to a script on the server that will compare the credentials against the database to check whether it is a legitimate credential then send back a result through the InputStream. If correct it will move on to the app and if wrong it will go back to the login page.

```
<?php
//this page checks students logins on mobile and returns 1 or 0 based on whether it is correct or not
require "connection.php";
$con = db_connect();
$user_name = $_POST["username"];
$user_pass = $_POST["password"];
$mysql_qry = "select * from studentcredential where StudentId='".$user_name."' and StudentPwd='".$user_pass."'";
$result = mysqli_query($con ,$mysql_qry);
if(mysqli_num_rows($result) > 0) {
echo "1";
}
else {
echo "0";
}

?>
```

Screenshot 5.1.2.8.4

The script that checks the credentials sent through the app. If legitimate it will send 1 if wrong credentials it will send 0.

```
@Override
protected void onPostExecute(String result) {
    if(result.equals("0"))
    {
        alertDialog.cancel();
        alertDialog = new AlertDialog.Builder(context).create();
        alertDialog.setTitle("Failed Login");
        alertDialog.setMessage("Wrong credentials");
        alertDialog.show();
    }
    else
    {
        Intent intent = new Intent(context,LoadingActivity.class);
        intent.putExtra( name: "id",username);
        context.startActivity(intent);
        ((Activity)context).finish();
    }
}
```

Screenshot 5.1.2.8.5

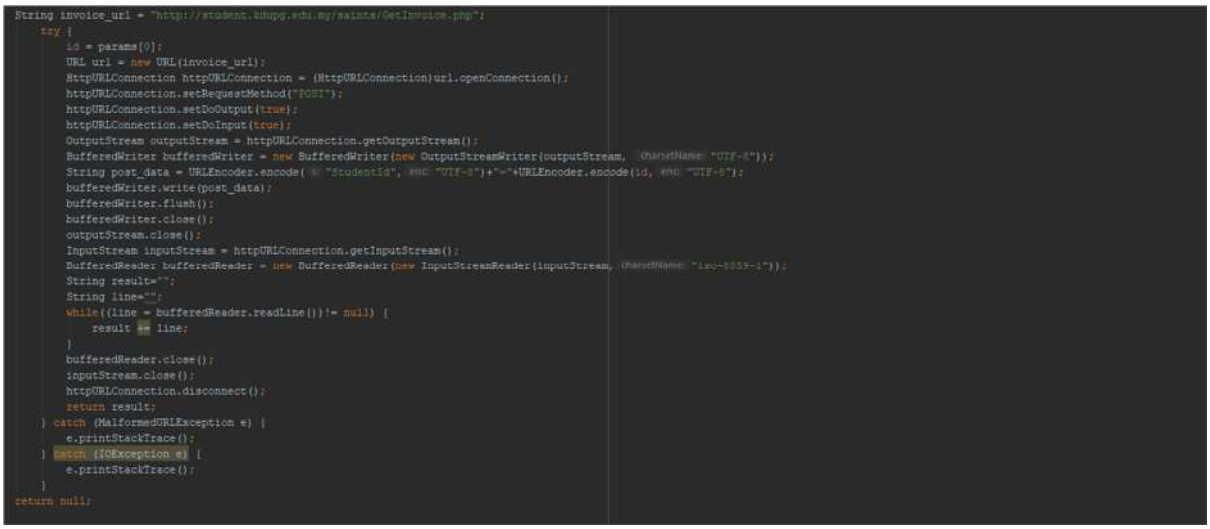
Based on the result from the request, it will output an AlertDialog saying Failed Login or start the next activity which is where the functions of the app are located at.

5.1.2.9 Loading Screen (Mobile)

```
if(username == null)
{
    SharedPreferences sp = this.getSharedPreferences( name: "data",MODE_PRIVATE);
    username = sp.getString( key: "id", defValue: null);
}
ImageView load = (ImageView) findViewById(R.id.loadingBlack);
Animation anim = AnimationUtils.loadAnimation( context: this,R.anim.rotate);
load.startAnimation(anim);
LoadInvoiceBackground loading = new LoadInvoiceBackground( context: this);
loading.execute(username);
```

Screenshot 5.1.2.9.1

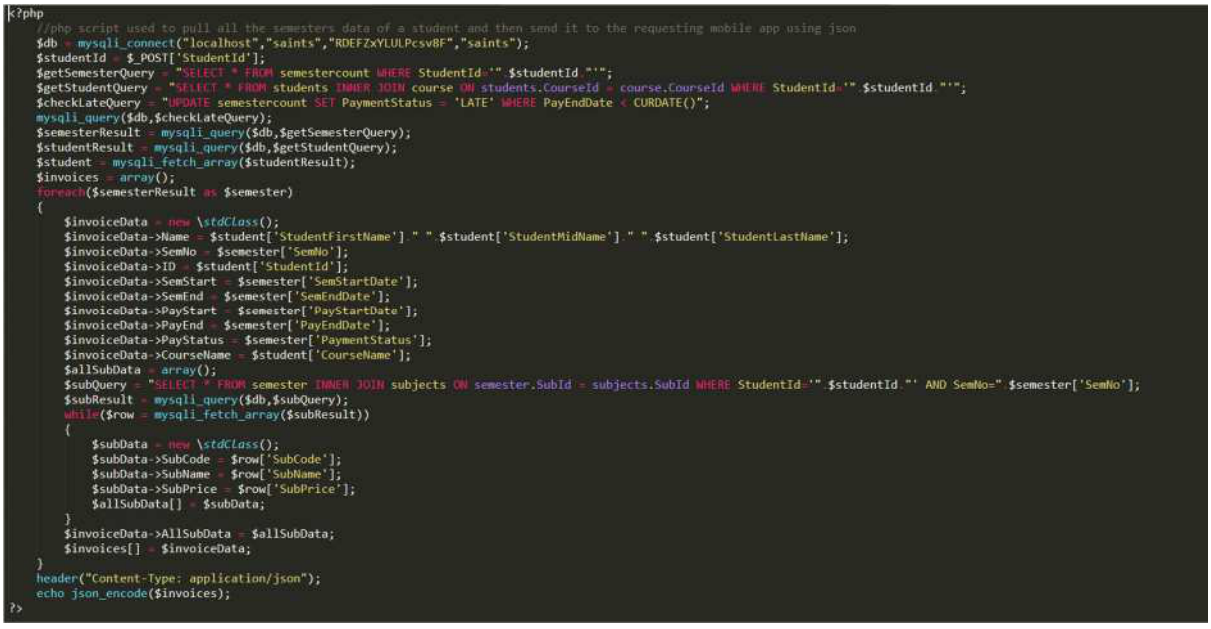
On the loading screen, logged in state will be saved as user has already successfully logged in, an animation will play during the time it takes to load the invoice data in a background task.



```
String invoice_url = "http://student.kdupg.edu.my/saints/GetInvoice.php";
    try {
        id = params[0];
        URL url = new URL(invoice_url);
        HttpURLConnection httpURLConnection = (HttpURLConnection)url.openConnection();
        httpURLConnection.setRequestMethod("POST");
        httpURLConnection.setDoOutput(true);
        httpURLConnection.setDoInput(true);
        OutputStream outputStream = httpURLConnection.getOutputStream();
        BufferedWriter bufferedWriter = new BufferedWriter(new OutputStreamWriter(outputStream,  charsetName: "UTF-8"));
        String post_data = URLEncoder.encode( s: "StudentId", enc: "UTF-8")+"="+URLEncoder.encode(id, enc: "UTF-8");
        bufferedWriter.write(post_data);
        bufferedWriter.flush();
        bufferedWriter.close();
        outputStream.close();
        InputStream inputStream = httpURLConnection.getInputStream();
        BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(inputStream,  charsetName: "iso-8859-1"));
        String result="";
        String line="";
        while((line = bufferedReader.readLine())!= null) {
            result += line;
        }
        bufferedReader.close();
        inputStream.close();
        httpURLConnection.disconnect();
        return result;
    } catch (MalformedURLException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
return null;
```

Screenshot 5.1.2.9.2

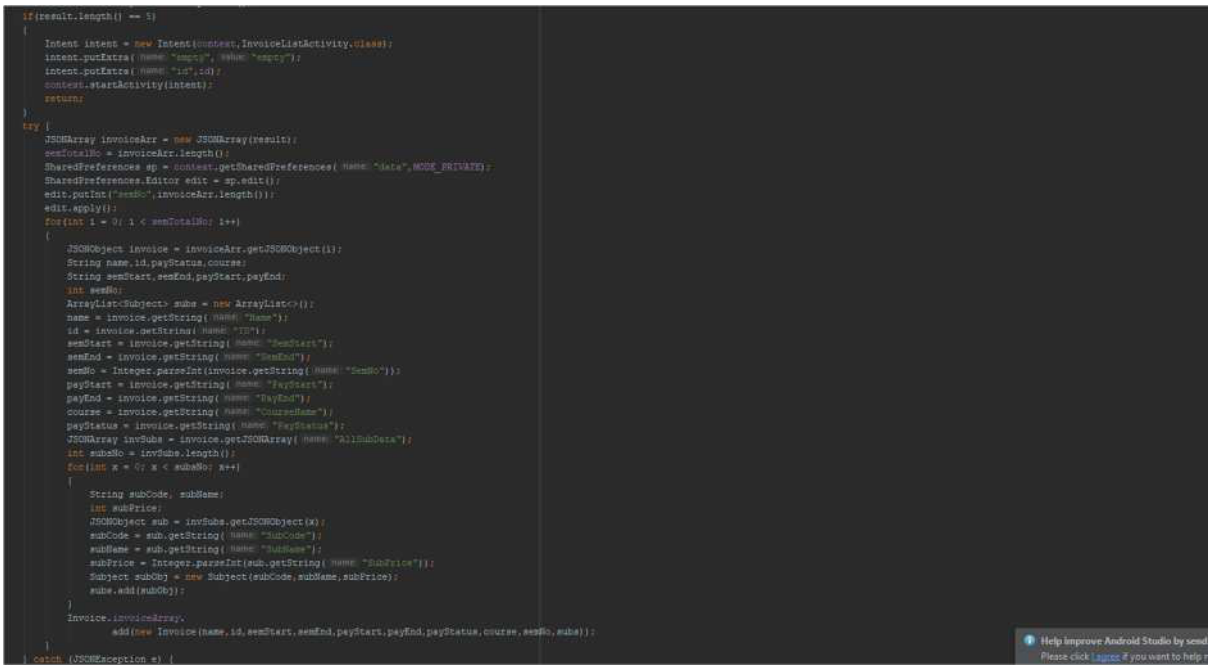
The background task that makes a request to the server. Same as login it will use the InputStream and OutputStream but the results returned is different.



```
<?php
    //php script used to pull all the semesters data of a student and then send it to the requesting mobile app using json
    $db = mysqli_connect("localhost","saints","RDEFZxYLULPcsv8F","saints");
    $studentId = $_POST['StudentId'];
    $getSemesterQuery = "SELECT * FROM semestercount WHERE StudentId='".$studentId."'";
    $getStudentQuery = "SELECT * FROM students INNER JOIN course ON students.CourseId = course.CourseId WHERE StudentId='".$studentId."'";
    $checkLateQuery = "UPDATE semestercount SET PaymentStatus = 'LATE' WHERE PayEndDate < CURDATE()";
    mysqli_query($db,$checkLateQuery);
    $semesterResult = mysqli_query($db,$getSemesterQuery);
    $studentResult = mysqli_query($db,$getStudentQuery);
    $student = mysqli_fetch_array($studentResult);
    $invoices = array();
    foreach($semesterResult as $semester)
    {
        $invoiceData = new \stdClass();
        $invoiceData->Name = $student['StudentFirstName']." ".$student['StudentMidName']." ".$student['StudentLastName'];
        $invoiceData->SemNo = $semester['SemNo'];
        $invoiceData->ID = $student['StudentId'];
        $invoiceData->SemStart = $semester['SemStartDate'];
        $invoiceData->SemEnd = $semester['SemEndDate'];
        $invoiceData->PayStart = $semester['PayStartDate'];
        $invoiceData->PayEnd = $semester['PayEndDate'];
        $invoiceData->PayStatus = $semester['PaymentStatus'];
        $invoiceData->CourseName = $student['CourseName'];
        $allSubData = array();
        $subQuery = "SELECT * FROM semester INNER JOIN subjects ON semester.SubId = subjects.SubId WHERE StudentId='".$studentId."' AND SemNo=".$semester['SemNo'];
        $subResult = mysqli_query($db,$subQuery);
        while($row = mysqli_fetch_array($subResult))
        {
            $subData = new \stdClass();
            $subData->SubCode = $row['SubCode'];
            $subData->SubName = $row['SubName'];
            $subData->SubPrice = $row['SubPrice'];
            $allSubData[] = $subData;
        }
        $invoiceData->AllSubData = $allSubData;
        $invoices[] = $invoiceData;
    }
    header("Content-Type: application/json");
    echo json_encode($invoices);
?>
```

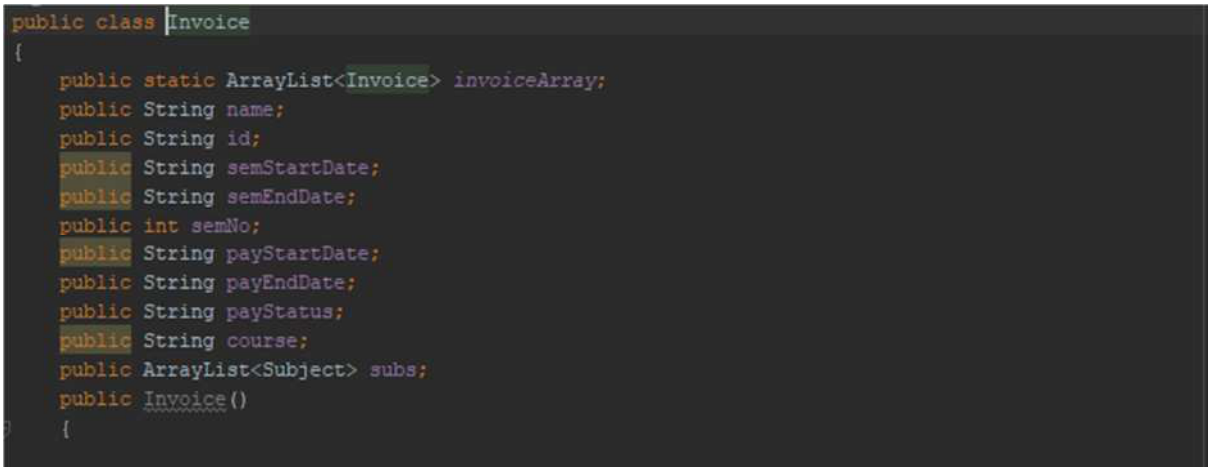
Screenshot 5.1.2.9.3

The script on the server that loads the data in neatly arranged arrays with objects in it that will be sent to the mobile through JSON. This script will also check for invoices that have passed the due date and update them to LATE accordingly



Screenshot 5.1.2.9.4

Here is where all the JSON data is recompiled into objects for easier data pulling later on. If an empty JSON is returned it would mean that the user currently has no invoices registered on the server. After compiling or if empty, it will then start the next activity where the data is laid out in a list.



Screenshot 5.1.2.9.5

Invoice object.

Screenshot 5.1.2.9.6

Subject object.

5.1.2.10 Invoice List (Mobile)

Screenshot 5.1.2.10.1

Once the InvoiceListActivity launches, they will first be asked for permission to write and read files as PDF functions require these permissions.

Screenshot 5.1.2.10.2

Next up is the list loading, if the student has no data then it'll just display a dialog. The list will be loaded through a ListView using a custom adapter made for the invoices with a custom layout to display the data. Below them theres a refresh listener that will fire when users pull down the list on the page.

```
public View getView(int position, View view, ViewGroup parent) {
        LayoutInflater inflater = LayoutInflater.from(context);
        View rowView=inflater.inflate(R.layout.layout_listrow, null,true);

        TextView semestertxt = (TextView)
rowView.findViewById(R.id.semester);
        TextView duetxt = (TextView) rowView.findViewById(R.id.duedate);
        TextView amountxt = (TextView) rowView.findViewById(R.id.amount);
        ImageView more = rowView.findViewById(R.id.more);
        ImageView statusimg = (ImageView)
rowView.findViewById(R.id.status);
        Drawable paidIcon =
rowView.getResources().getDrawable(R.drawable.succes);
        Drawable pendingIcon =
rowView.getResources().getDrawable(R.drawable.pending);
        Drawable lateIcon =
rowView.getResources().getDrawable(R.drawable.late);
        TextView dateLabel = (TextView)
rowView.findViewById(R.id.datelabel);
        int semNo = invoiceList.get(position).semNo;
        final String sem = String.valueOf(semNo);
        final String id = invoiceList.get(position).id;
        final String status = invoiceList.get(position).payStatus;
        dateLabel.setText("Due Date :");
        ArrayList<Subject> subs = invoiceList.get(position).subs;
        int subNo = subs.size();
        int total = 650;
        for(int i = 0; i < subNo; i++)
        {
            total += subs.get(i).subPrice;
        }
```

```
        final int price = total;
        amountxt.setText("RM"+total);
        semestertxt.setText("Semester : " + semNo);
        more.setOnClickListener(new View.OnClickListener()
        {
            @Override
            public void onClick(View v) {

                showMenu(v,sem,id,price,status);
            }
        });
        String payDate = invoiceList.get(position).payEndDate;
        Date date = null;
        try {
            date = new SimpleDateFormat("yyyy-MM-dd").parse(payDate);
            String formattedDate = new
SimpleDateFormat("dd/MM/yyyy").format(date);
            duetxt.setText(formattedDate);
        } catch (ParseException e) {
            e.printStackTrace();
        }
        if(status.equals("PENDING"))
            statusimg.setImageDrawable(pendingIcon);
        else if(status.equals("LATE"))
            statusimg.setImageDrawable(lateIcon);
        else if(status.equals("PAID"))
            statusimg.setImageDrawable(paidIcon);
        return rowView;
    }

    public void showMenu (View view,String sem,String id,int price,String
status)
    {
        final String semNo = sem;
        final String idNo = id;
        final int priceNo = price;
        final String statusNo = status;


        PopupMenu menu = new PopupMenu (context, view);
        menu.setOnMenuItemClickListener (new
PopupMenu.OnMenuItemClickListener ()
        {
            @Override
            public boolean onMenuItemClick (MenuItem item)
            {
                int itemId = item.getItemId();
                if(itemId == R.id.item_view)
                {
                    if(PermissionCheck.checkRead(context))
                    {
                        LoadPDFBackground download = new
LoadPDFBackground(context);
                        download.execute(idNo,semNo,"0");
                    }
                    else
                    {
                        Toast.makeText(context,"Access denied, Please head
to FAQ to fix this",Toast.LENGTH_LONG).show();
                    }
                }
```

```
                else if(itemId == R.id.item_download)
                {
                    if(PermissionCheck.checkRead(context))
                    {
                        LoadPDFBackground download = new
LoadPDFBackground(context);
                        download.execute(idNo,semNo,"1");
                    }
                    else
                    {
                        Toast.makeText(context,"Access denied, Please head
to FAQ to fix this",Toast.LENGTH_LONG).show();
                    }
                }
                else if(itemId == R.id.item_pay)
                {
                    if(statusNo.equals("PENDING"))
                    {
                        Intent intent = new
Intent(context,PaymentActivity.class);
                        intent.putExtra("price",priceNo);
                        intent.putExtra("id",idNo);
                        intent.putExtra("sem",semNo);
                        context.startActivity(intent);
                    }
                    else if(statusNo.equals("PAID"))
                    {
                        Toast.makeText(context,"This semester has already
been paid off",Toast.LENGTH_SHORT).show();
                    }
                    else
                    {
                        Toast.makeText(context,"The due date for this
invoice has passed, Please proceed to bursary",Toast.LENGTH_LONG).show();
                    }
                }
                return true;
            }
        });
        menu.inflate(R.menu.menu_invoice);
        menu.show();
    }
```

Code Block 5.1.2.10.3

Due to the nature of the code, formatting will be used instead of screenshots. The layout consists of an ImageViewer to display the status of the invoice and some TextViews to display the details. Besides that, the layout has a button that makes a popup menu with View, Download, Pay options in them.

```
public void refreshInvoices()
{
    LoadInvoiceBackground reload = new LoadInvoiceBackground( context: InvoiceListActivity.this); // your code
    reload.execute(id);
    pullToRefresh.setRefreshing(true);
    finish();
}
```

Screenshot 5.1.2.10.4

refreshInvoices is called when users swipe down to refresh the list. This will callback the background task that was used to load the invoices before this entering this Activity.

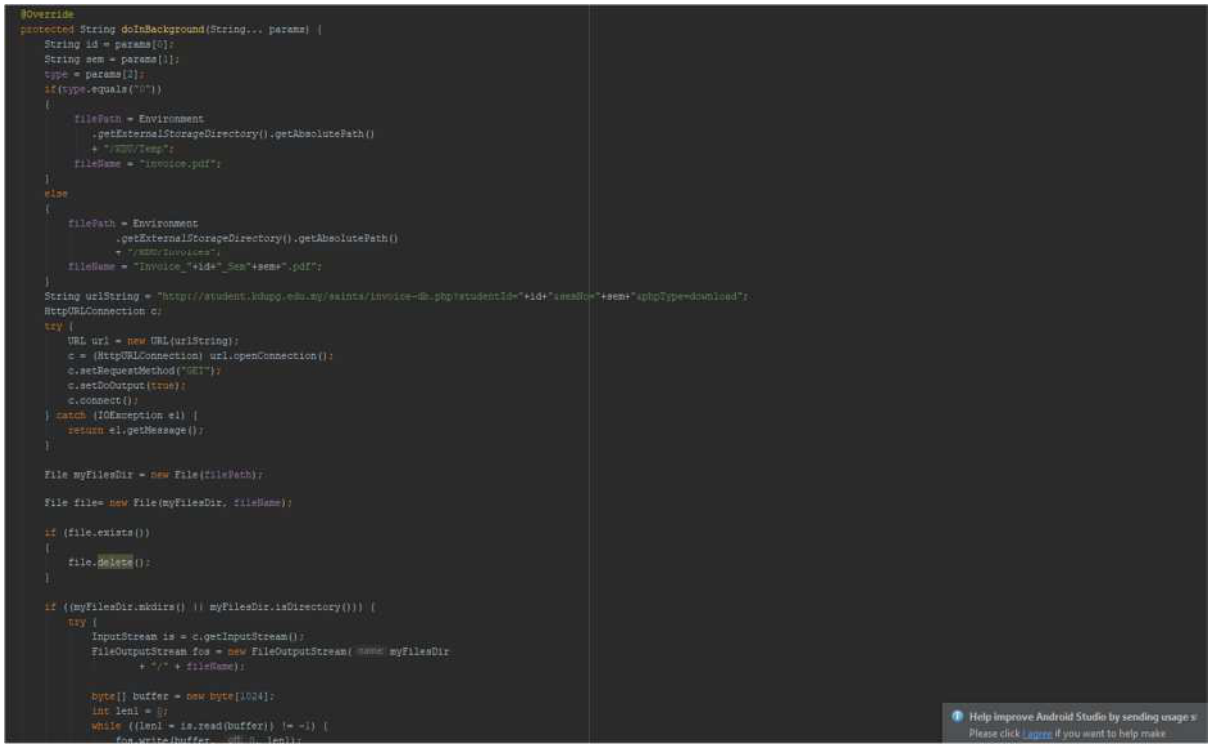
```
public class PermissionCheck{
    public static void readAndWriteExternalStorage(Context context){
        final Context ctx = context;
        if(!checkRead(context)) {
            AlertDialog.Builder builder = new AlertDialog.Builder(context);
            builder.setTitle("Permissions Are Needed")
                    .setMessage("This app requires storage permissions for some of its functions to work properly. Please allow when prompted")
                    .setCancelable(false)
                    .setPositiveButton( text: "OK", (dialog, id) ->
                            selfPerm(ctx);
                    });
            AlertDialog alert = builder.create();
            alert.show();
            return;
        }
    }
    public static boolean checkRead(Context context)
    {
        if(ActivityCompat.checkSelfPermission(context, Manifest.permission.READ_EXTERNAL_STORAGE) != PackageManager.PERMISSION_GRANTED
                && ActivityCompat.checkSelfPermission(context, Manifest.permission.WRITE_EXTERNAL_STORAGE) != PackageManager.PERMISSION_GRANTED)
        {
            return false;
        }
        else
        {
            return true;
        }
    }

    public static void selfPerm(Context context)
    {
        ActivityCompat.requestPermissions((Activity) context,
                new String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE,
                        Manifest.permission.READ_EXTERNAL_STORAGE}, requestCode: 1);
    }
}
```

Screenshot 5.1.2.10.5

The class that has all the code regarding permissions. selfPerm() brings up the permissions prompt while checkRead() is used to see whether user has already enabled permissions. readAndWriteExternalStorage() is used on initial startup to show a dialog so that users can know why they need to enable permissions on prompt.

5.1.2.11 PDF View and Downloading (Mobile)



```
@Override
protected String doInBackground(String... params) {
    String id = params[0];
    String sem = params[1];
    type = params[2];
    if(type.equals("0"))
    {
        filePath = Environment
           .getExternalStorageDirectory().getAbsolutePath()
           + "/KDU/Temp";
        fileName = "invoice.pdf";
    }
    else
    {
        filePath = Environment
                .getExternalStorageDirectory().getAbsolutePath()
                + "/KDU/Invoices";
        fileName = "Invoice_"+id+"_Sem"+sem+".pdf";
    }
    String urlString = "http://student.kdupg.edu.my/saints/invoice-db.php?studentId="+id+"&semNo="+sem+"&phpType=download";
    HttpURLConnection c;
    try {
        URL url = new URL(urlString);
        c = (HttpURLConnection) url.openConnection();
        c.setRequestMethod("GET");
        c.setDoOutput(true);
        c.connect();
    } catch (IOException e1) {
        return e1.getMessage();
    }

    File myFilesDir = new File(filePath);

    File file= new File(myFilesDir, fileName);

    if (file.exists())
    {
        file.delete();
    }

    if ((myFilesDir.mkdirs() || myFilesDir.isDirectory())) {
        try {
            InputStream is = c.getInputStream();
            FileOutputStream fos = new FileOutputStream( name: myFilesDir
                    + "/" + fileName);

            byte[] buffer = new byte[1024];
            int len1 = 0;
            while ((len1 = is.read(buffer)) != -1) {
                fos.write(buffer, off: 0, len1);
```

Help improve Android Studio by sending usage s
Please click I agree if you want to help make

Screenshot 5.1.2.11.1

When View or Download option is chosen from menu, the PDF background task is called where it will load the PDF from the same PHP script as the ones used on the website to View and Download.

If the user is viewing, it will check whether the file exists in a temp folder then delete and overwrite said file. If the user is downloading, it will give the invoice a unique name in an Invoice folder so that users can save the PDFs in their phones.

```
super.onPostExecute(result);
if(result.equals("1")) {
    if(type.equals("0"))
    {
        Intent intent = new Intent(context, PDFViewActivity.class);
        context.startActivity(intent);
    }
    else
    {
        Toast.makeText(context, text: "Invoice has been downloaded at" + filePath,Toast.LENGTH_LONG).show();
    }
}
else
    Toast.makeText(context, result, Toast.LENGTH_LONG).show();
```

Screenshot 5.1.2.11.2

After the PDF has been successfully pulled from the script, the result will be returned depending on which option was chosen. If it was View, an in-app PDF Viewer Activity will be run loaded with the PDF chosen. If it was Download, it will display a message to the user telling them where the PDF is located. Error messages are also implemented in the case the PDF failed to download.

```
public class PDFViewActivity extends AppCompatActivity{
    private WebView pdfViewer;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.layout_pdfview);
        Intent intent = getIntent();
            File filePath = new File( pathname: Environment
                    .getExternalStorageDirectory().getAbsolutePath()
                    + "/KDU/Temp");
            File file = new File(filePath, child: "invoice.pdf");
            PDFView pdfView = findViewById(R.id.pdfView);
            pdfView.fromFile(file).load();
    }
```

Screenshot 5.1.2.11.3

The PDF Viewer is a 3rd party library in the JCenter Repository called Android PDF Viewer. The PDF is loaded right after the PDFViewActivity is called, into the PDF View layout for the user to see.

5.1.2.12 Payment (Mobile)

```
public void paymentChoice(int choice) {
    if (choice == 1) {
        Intent intent = new Intent( packageContext: this, BankingActivity.class);
        intent.putExtra( name: "id",id);
        intent.putExtra( name: "sem",sem);
        startActivity(intent);
    } else {
        Intent intent = new Intent( packageContext: this, CardActivity.class);
        intent.putExtra( name: "id",id);
        intent.putExtra( name: "sem",sem);
        startActivity(intent);
    }
}
```

Screenshot 5.1.2.12.1

When users select the pay option in InvoiceListActivity's menu, it will bring them to a page with two buttons on it saying Online Banking and Card Payment. Depending on which option they choose the code above will redirect them to the respective payment pages. After payments become successful it will call a background task to update the server database that their chosen invoice has been successfully paid off.

```
protected String doInBackground(String... params) {
    String check_url = "http://student.kdupg.edu.my/saints/UpdateSem.php";
    try {
        username = params[0];
        sem = params[1];
        URL url = new URL(check_url);
        HttpURLConnection httpURLConnection = (HttpURLConnection)url.openConnection();
        httpURLConnection.setRequestMethod("POST");
        httpURLConnection.setDoOutput(true);
        httpURLConnection.setDoInput(true);
        OutputStream outputStream = httpURLConnection.getOutputStream();
        BufferedWriter bufferedWriter = new BufferedWriter(new OutputStreamWriter(outputStream, charsetName: "UTF-8"));
        String post_data = URLEncoder.encode( s: "StudentId", enc: "UTF-8")+"="+URLEncoder.encode(username, enc: "UTF-8") + "&" +
                           URLEncoder.encode( s: "SemNo", enc: "UTF-8")+"="+URLEncoder.encode(sem, enc: "UTF-8");
        bufferedWriter.write(post_data);
        bufferedWriter.flush();
        bufferedWriter.close();
        outputStream.close();
        InputStream inputStream = httpURLConnection.getInputStream();
        BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(inputStream, charsetName: "iso-8859-1"));
        String result="";
        String line="";
        while((line = bufferedReader.readLine())!= null) {
            result += line;
        }
        bufferedReader.close();
        inputStream.close();
        httpURLConnection.disconnect();
        return result;
    } catch (MalformedURLException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
    return null;
}
```

Screenshot 5.1.2.12.2

Same style as Login, the data is encoded to the link then sent to the script through the Input and Output Streams.

```
protected void onPostExecute(String result) {
    int amnt = Integer.parseInt(result);
    if(amnt > 0)
    {
        Toast.makeText(context, text: "Payment Successful",Toast.LENGTH_LONG).show();
        LoadInvoiceBackground load = new LoadInvoiceBackground(context);
        load.execute(username);
    }
    else
    {
        Toast.makeText(context, text: "Payment Failed",Toast.LENGTH_LONG).show();
        LoadInvoiceBackground load = new LoadInvoiceBackground(context);
        load.execute(username);
    }
}
```

Screenshot 5.1.2.12.3

The user will be notified whether it is successful or failure depending on what is returned from the script.

```
<?php
    //Update a semester to paid status when successful payment has been detected on mobile
    include("connection.php");
    $db = db_connect();
    $updateSemQuery = "UPDATE semestercount SET PaymentStatus='PAID' WHERE StudentId='".$_POST['StudentId']."' AND SemNo=".$_POST['SemNo'];
    mysqli_query($db,$updateSemQuery);
    echo mysqli_affected_rows($db);
?>
```

Screenshot 5.1.2.12.4

The script that updates the server of the payment and returns whether it was a success or not.

5.1.2.13 FAQ (Mobile)

```
sp = this.getSharedPreferences( name: "data", MODE_PRIVATE);
edit = sp.edit();
context = this;
PDFView view = findViewById(R.id.faqPdf);
view.fromAsset( assetName: "FAQ.pdf").load();
```

Screenshot 5.1.2.13.1

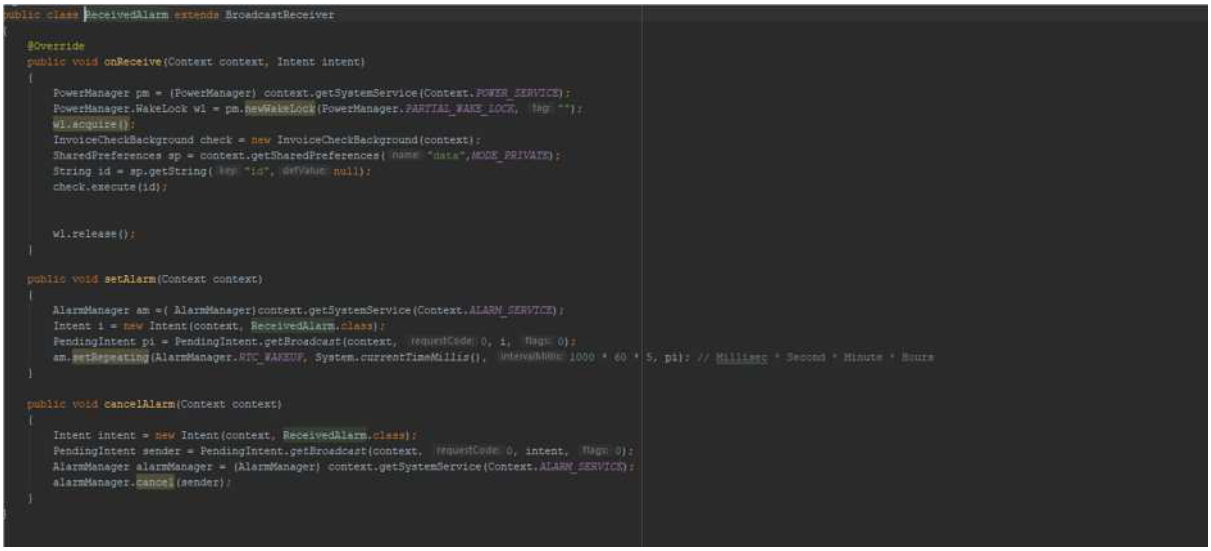
In the FAQActivity, a PDF manual is loaded from the assets into the Android PDF Viewer layout so that users can see what went wrong when using the app and its corresponding error messages.

```
public void triggerPermission(View v)
{
    if(!PermissionCheck.checkRead( context: this))
    {
        PermissionCheck.selfPerm( context: this);
        Toast.makeText( context: this, text: "If a prompt does not appear, read the rest of the FAQ",Toast.LENGTH_LONG).show();
    }
    else
    {
        Toast.makeText( context: this, text: "Permission already granted",Toast.LENGTH_LONG).show();
    }
}
```

Screenshot 5.1.2.13.2

A button to trigger permissions manually is also located in the FAQActivity, in case users miss pressed during the permissions prompt or disabled the prompt from showing again.

5.1.2.14 Notifications (Mobile)



```
public class ReceivedAlarm extends BroadcastReceiver
{
    @Override
    public void onReceive(Context context, Intent intent)
    {
        PowerManager pm = (PowerManager) context.getSystemService(Context.POWER_SERVICE);
        PowerManager.WakeLock wl = pm.newWakeLock(PowerManager.PARTIAL_WAKE_LOCK, tag: "");
        wl.acquire();
        InvoiceCheckBackground check = new InvoiceCheckBackground(context);
        SharedPreferences sp = context.getSharedPreferences( name: "data",MODE_PRIVATE);
        String id = sp.getString( key: "id", defValue: null);
        check.execute(id);


        wl.release();
    }

    public void setAlarm(Context context)
    {
        AlarmManager am =( AlarmManager)context.getSystemService(Context.ALARM_SERVICE);
        Intent i = new Intent(context, ReceivedAlarm.class);
        PendingIntent pi = PendingIntent.getBroadcast(context,  requestCode: 0, i,  flags: 0);
        am.setRepeating(AlarmManager.RTC_WAKEUP, System.currentTimeMillis(),  intervalMillis: 1000 * 60 * 5, pi); // Millisec * Second * Minute * Hours
    }

    public void cancelAlarm(Context context)
    {
        Intent intent = new Intent(context, ReceivedAlarm.class);
        PendingIntent sender = PendingIntent.getBroadcast(context,  requestCode: 0, intent,  flags: 0);
        AlarmManager alarmManager = (AlarmManager) context.getSystemService(Context.ALARM_SERVICE);
        alarmManager.cancel(sender);
    }
}
```

Screenshot 5.1.2.14.1

Notifications are triggered using an AlarmManager that broadcasts an intent to itself as a BroadcastReceiver. It will check every 5 minutes whether there is a new invoice registered on the server that belongs to the user.

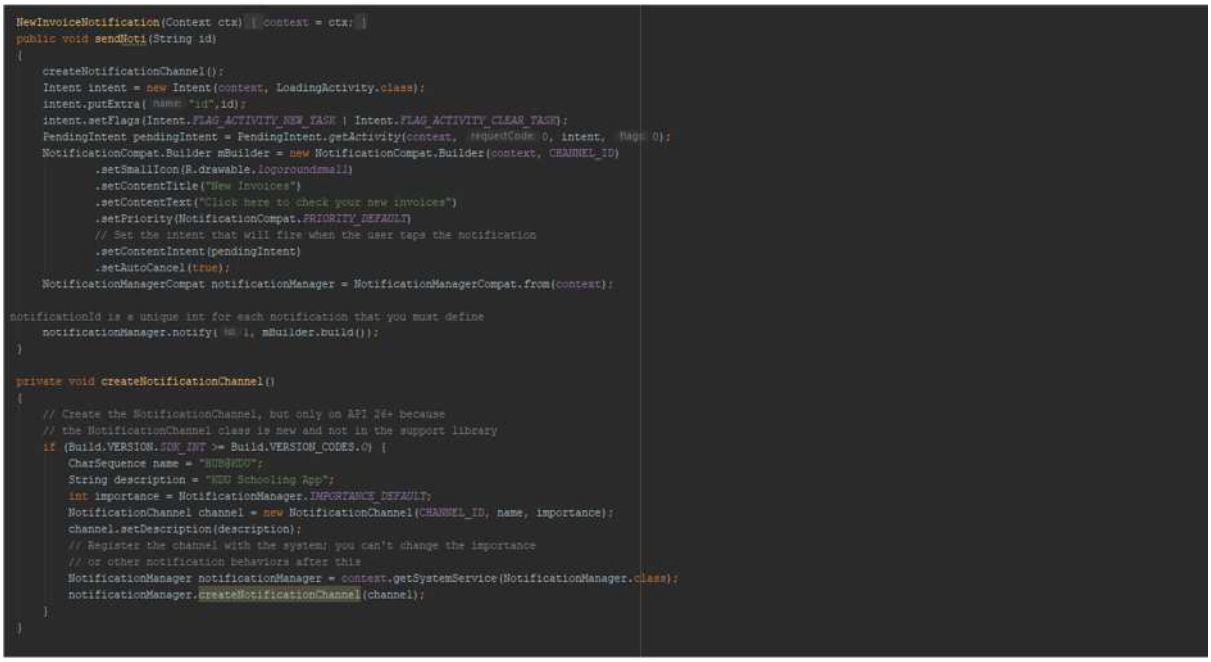
```
<?php
    include_once("connection.php");
    $db = db_connect(); //initialize connection
    $checksemquery = "SELECT COUNT(SemNo) AS SemNo FROM semestercount WHERE StudentId='".$_POST['StudentId']."'"; //finds how many invoices the student has
    $checkSemResult = mysqli_query($db,$checksemquery);
    $row = mysqli_fetch_array($checkSemResult);
    echo $row['SemNo']; //returns it to mobile app that is requesting this script
?>
```

Screenshot 5.1.2.14.2

The script that is called during the routine broadcast, it looks for the number of invoices registered to the student and returns this value to the mobile app.

Screenshot 5.1.2.14.3

Using the result returned from the script above it checks whether the number of invoices on the server is more than the number of invoices loaded on the app. If more then it will send a notification to the user on the phone. This notification will send despite the app being killed or in the background/not being used.



Screenshot 5.1.2.14.4

Notification class. sendNoti() is the main function that creates the notification then send it out. createNotificationChannel() is a function made to cater to Android Oreo and above where a notification channel with a channel id for notifications in the same group is mandatory.

5.2 Installation

The installation for Hub@KDU for both website and mobile platform is listed as below.

5.2.1 Configure SQL

Import the .sql file name HUB@KDU – DATABASE.sql file from the CD-ROM and import it to the respective database as a dataset.

5.2.2 Installation of FileZilla Client

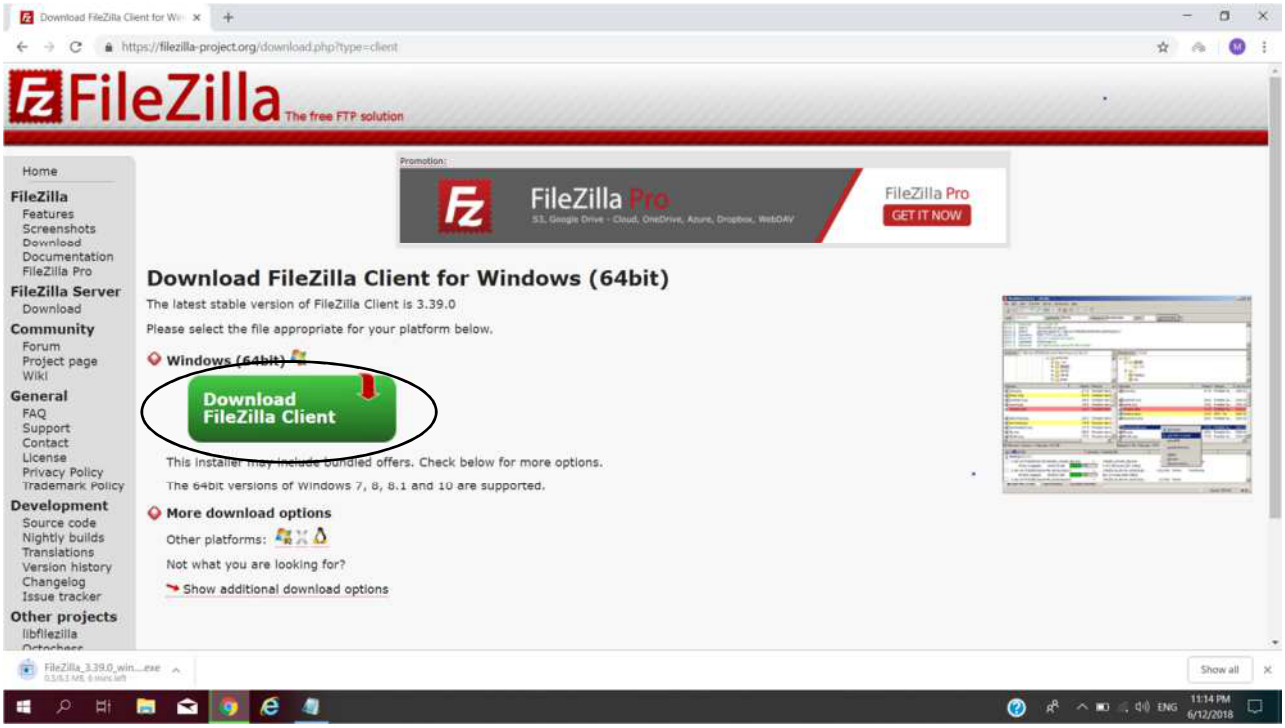


Diagram 5.2.2.1

To install FileZilla Client, user should go to this link (https://filezilla-project.org/download.php?type=client) and click on the download FileZilla Client which is circled in the diagram to download the .exe file.

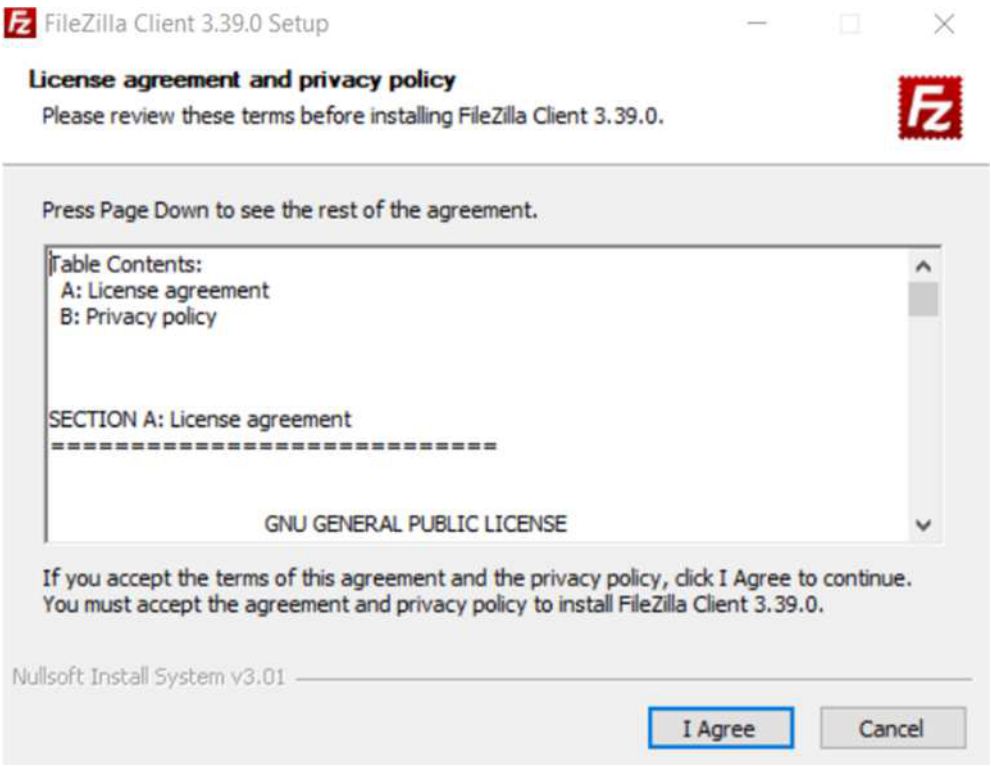


Diagram 5.2.2.2

Once the .exe file is done downloading, execute the file by running it. Press the button ‘I agree’ to move to the next page of the installation.

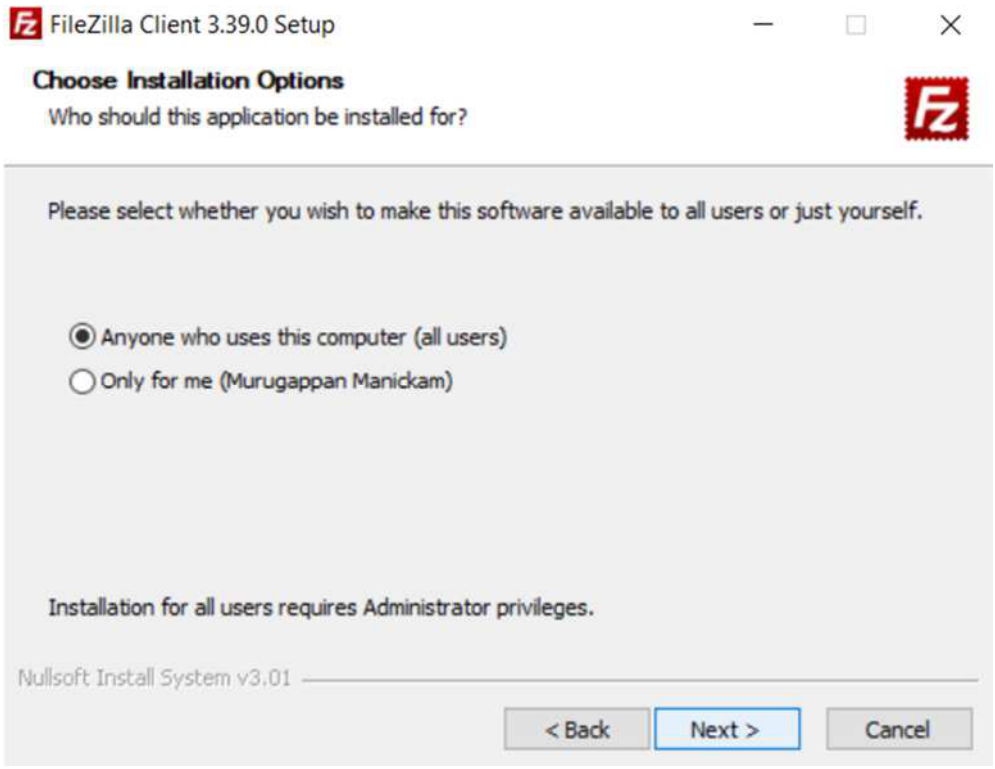


Diagram 5.2.2.3

The user can customise whether the software is only available for the user who are installing it or available for anyone that is accessing the workstation. Press “Next” to proceed to the next page.

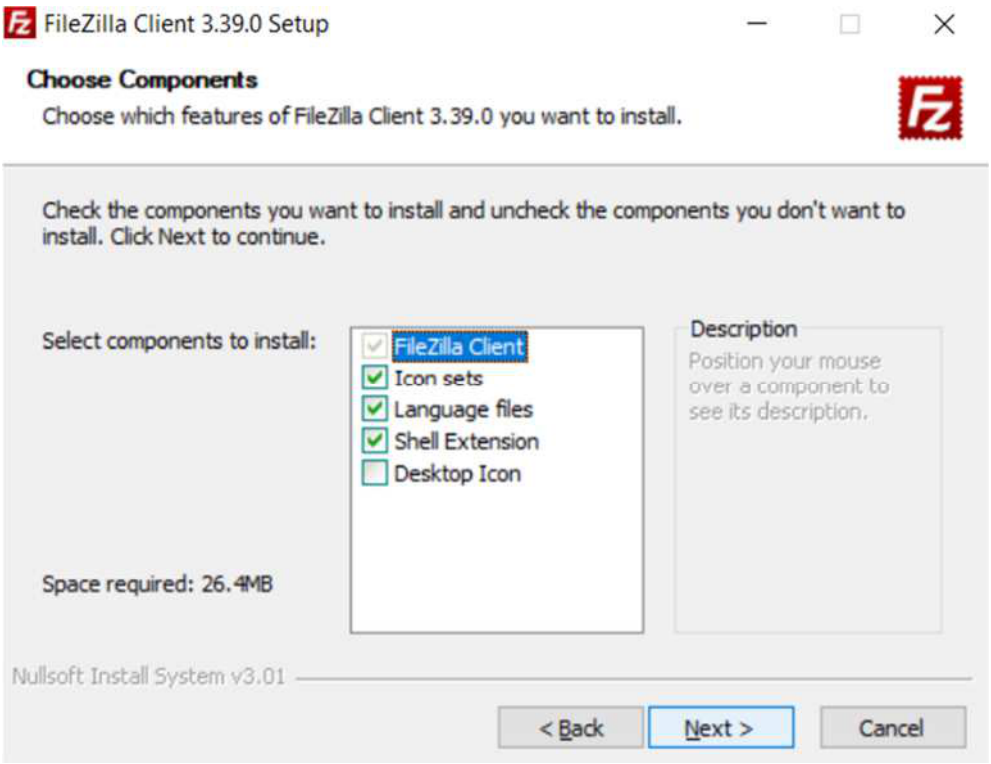


Diagram 5.2.2.4

You can customise the components you want to be installed but for a smooth operation of the FileZilla Client, we advise to just proceed with what has been checked by default. Press “Next” to proceed to the next page.

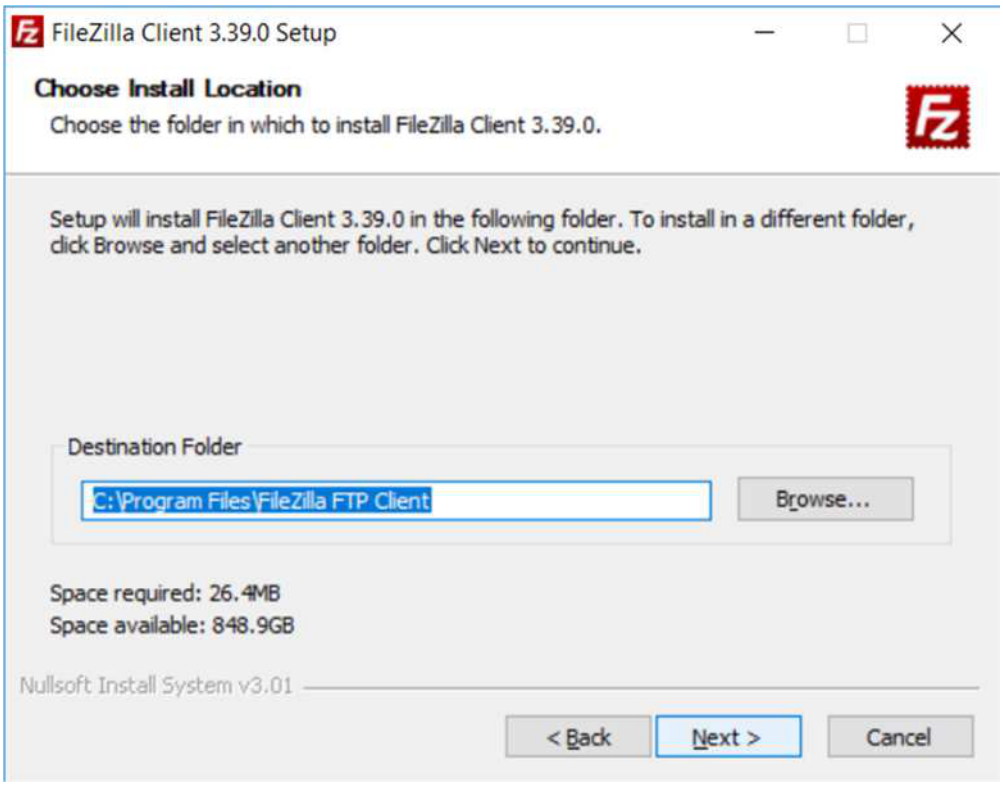


Diagram 5.2.2.5

In this page of the installation, you can select where should the installation of the software be done. Press “Next” to proceed to the next page.

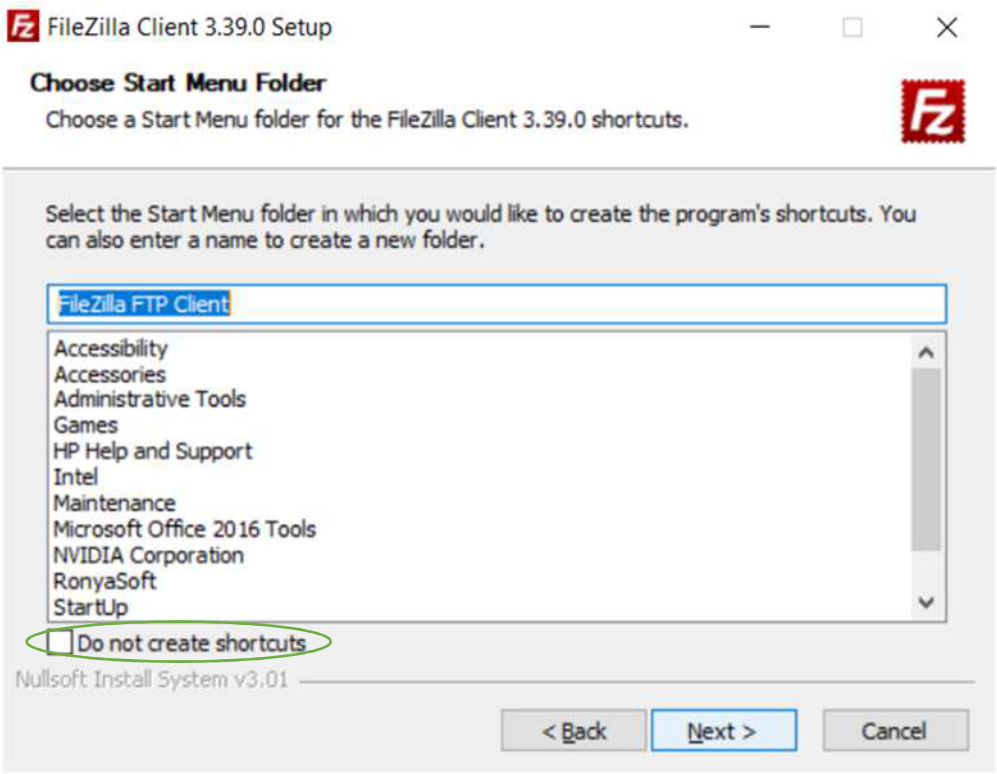


Diagram 5.2.2.6

In this page, the user can choose start menu folder and if the user does not want a shortcut to be created, they can check the box which is circled in the diagram above. Press “Next” to proceed to the next page.

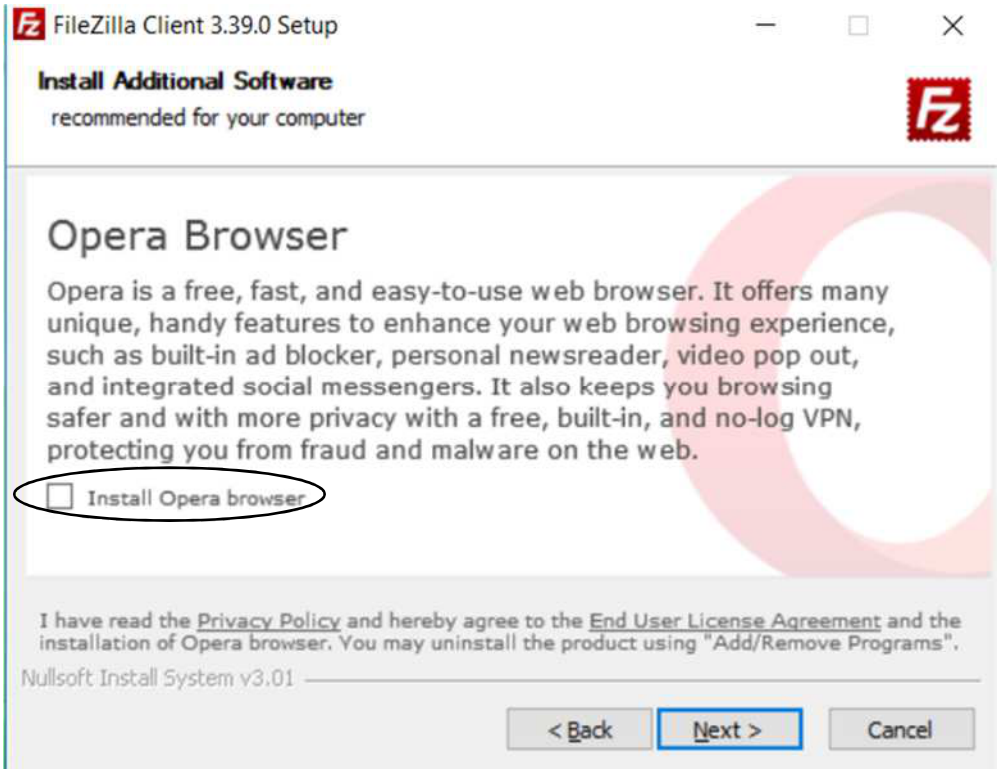


Diagram 5.2.2.7

Be aware that FileZilla might trick you into installing other software while installing FileZilla, so if you see the title Install Additional Software, make sure that the box which is circled in the diagram is unchecked. Press “Next” to proceed to the next page.

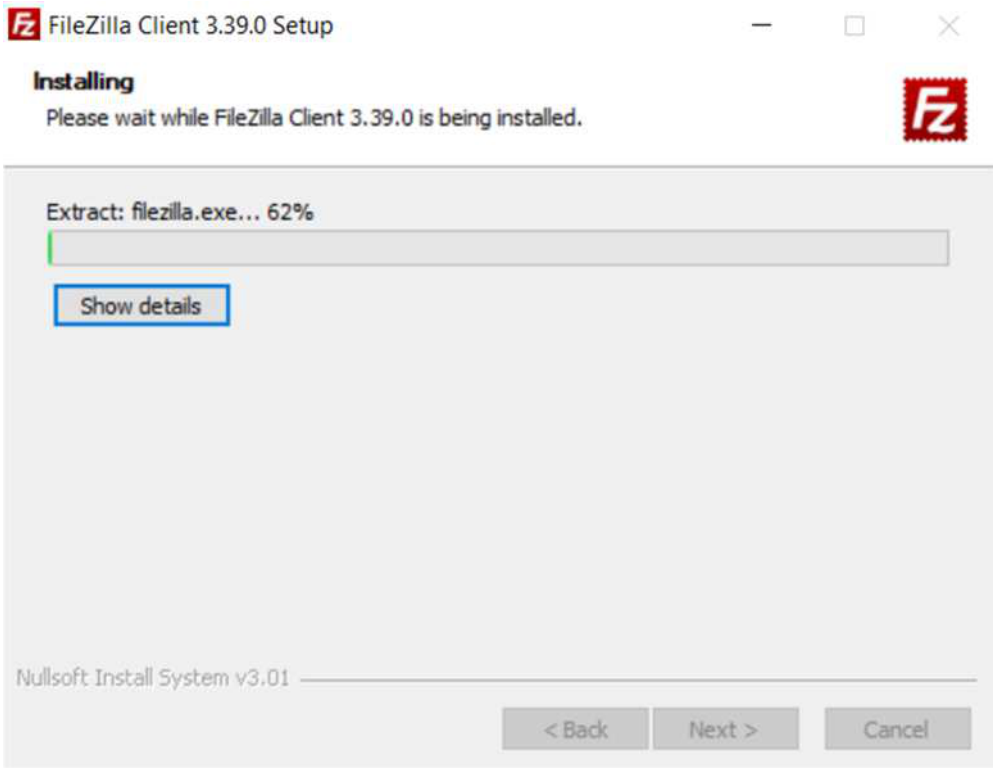


Diagram 5.2.2.8

Now wait for the files to be installed.

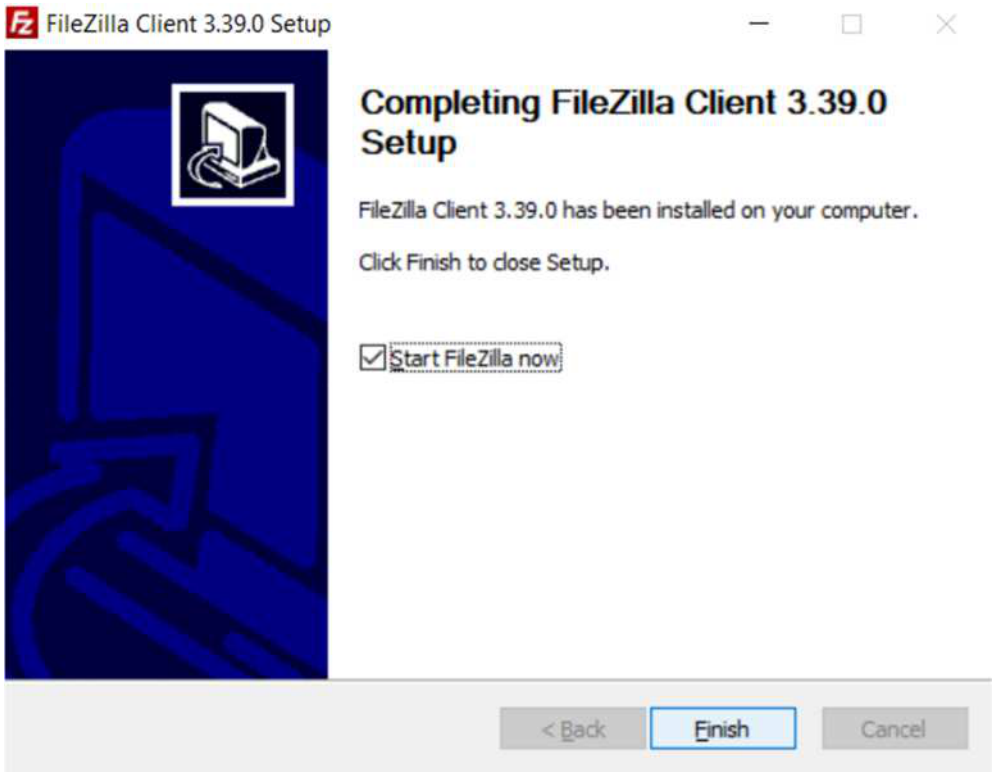


Diagram 5.2.2.9

Check the box as shown in the diagram to immediately start FileZilla and then click the Finish button to fire up FileZilla.

5.2.3 Uploading files to the server

Copy the folder called HUB@KDU – WEBSITE in the CD-ROM to the desktop. Edit the file called connection.php and edit the variable in lines 3,4,5 according to your database hosting credentials.

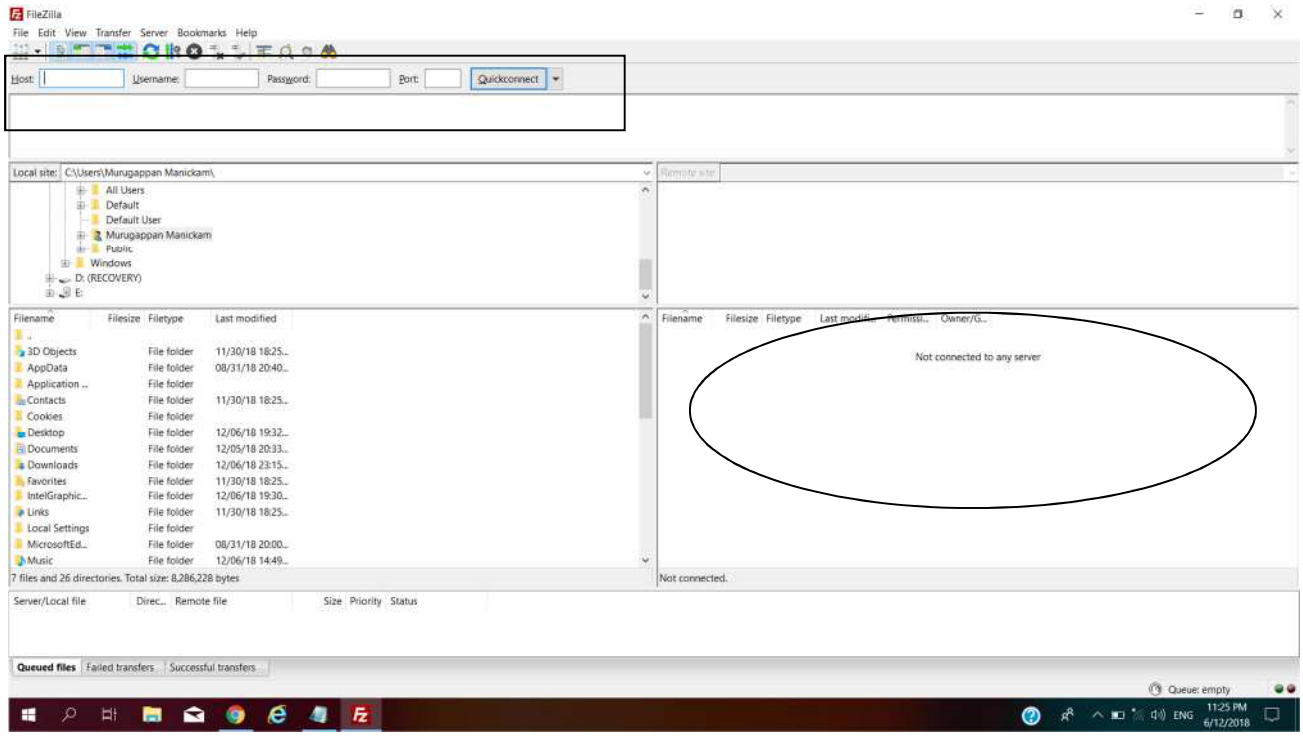


Diagram 5.2.3.1

Connect to your ftp server by entering your necessary credentials to the rectangled area in the diagram above.

Drag and drop all the files from the folder called HUB@KDU – WEBSITE which you copied to the desktop to the circled area in the diagram above and you have completed the process of uploading the files to the server.

5.2.4 Installation of Mobile App

To install the mobile app, just connect your phone to the workstation and drag and drop the .apk file which is stored in the CD-ROM as HUB@KDU.apk in your local storage. Redirect yourself to the place where you stored the .apk file in your mobile phone and execute the .apk file to install it. Finally, you can start using the app once the installation is done.

Chapter 6: Testing

The testing process for Hub@KDU will be shown in this section. There will be different types of testing that will be done. For example there will be different rounds of testing of the system. The bugs and errors will be listed and the team will be fixing these errors before releasing the final version of the project. There will also be blackbox testing done for both platforms.

6.1 System Testing (Website)

The system testing done for website requires minimum testing. This is because most of the components used does not require keyboard input from users. The only components that requires keyboard input from users are the student ID field in the home page and the username and password for login. Therefore, hardcore testing will be done for both the login page and the student ID field in the home page which will be shown in the blackbox testing later on.

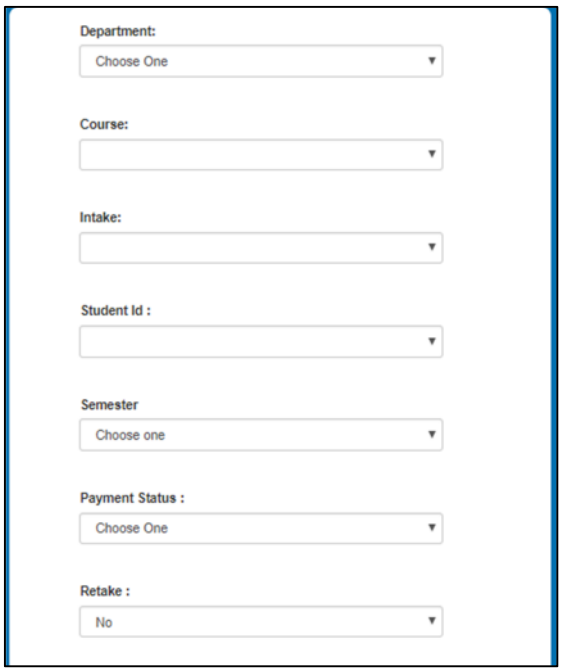


Diagram 6.1.1

As shown in Diagram 6.1.0, most of the components used in our website for inputs is drop-down box only for deploying the invoice and even updating the invoice after editing. These inputs are actually stored in databases already and are extracted to be stored in the drop-down box. Therefore, admins need to only select from the existing data in the drop-down box. We chose to implement drop-down boxes over text boxes which are editable because if we let the admins to input data, there will be different ways of admin inputting data when the outcome will be the same such as inputting the course as “Diploma in Computer Studies” and “diploma in computer studies” is the same for humans but for the computers, it is different just because one is uppercase and the other is lowercase. Therefore, to prevent all these mishaps we decided to use drop down boxes instead.

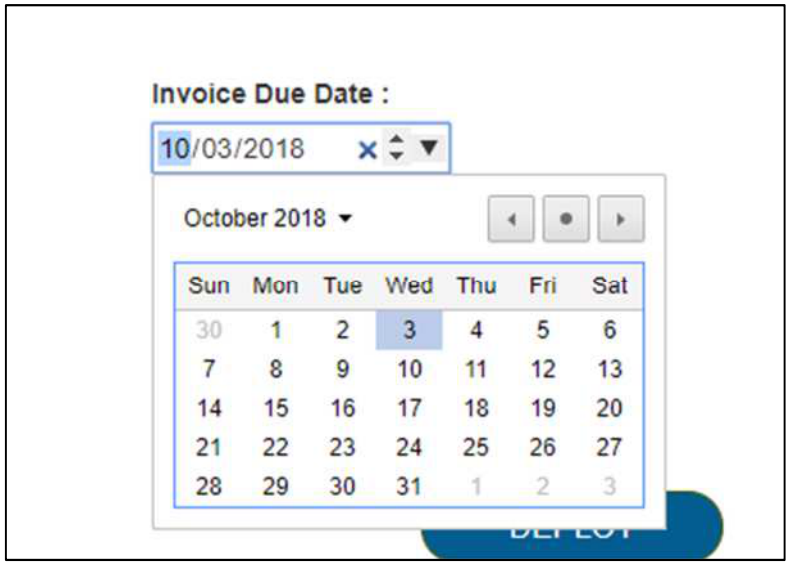


Diagram 6.1.2

The same was done for date selections for the invoice as well. The team used date pickers that can be implemented using HTML. Instead of manually inputting the dates, admins can just select the date using the date picker as shown in Diagram 6.1.1.

6.1.1 Errors Found during System Testing (Website)

During a few rounds of system testing for the website, there were a few bugs and errors found. Below are the errors that have since been fixed by the team.

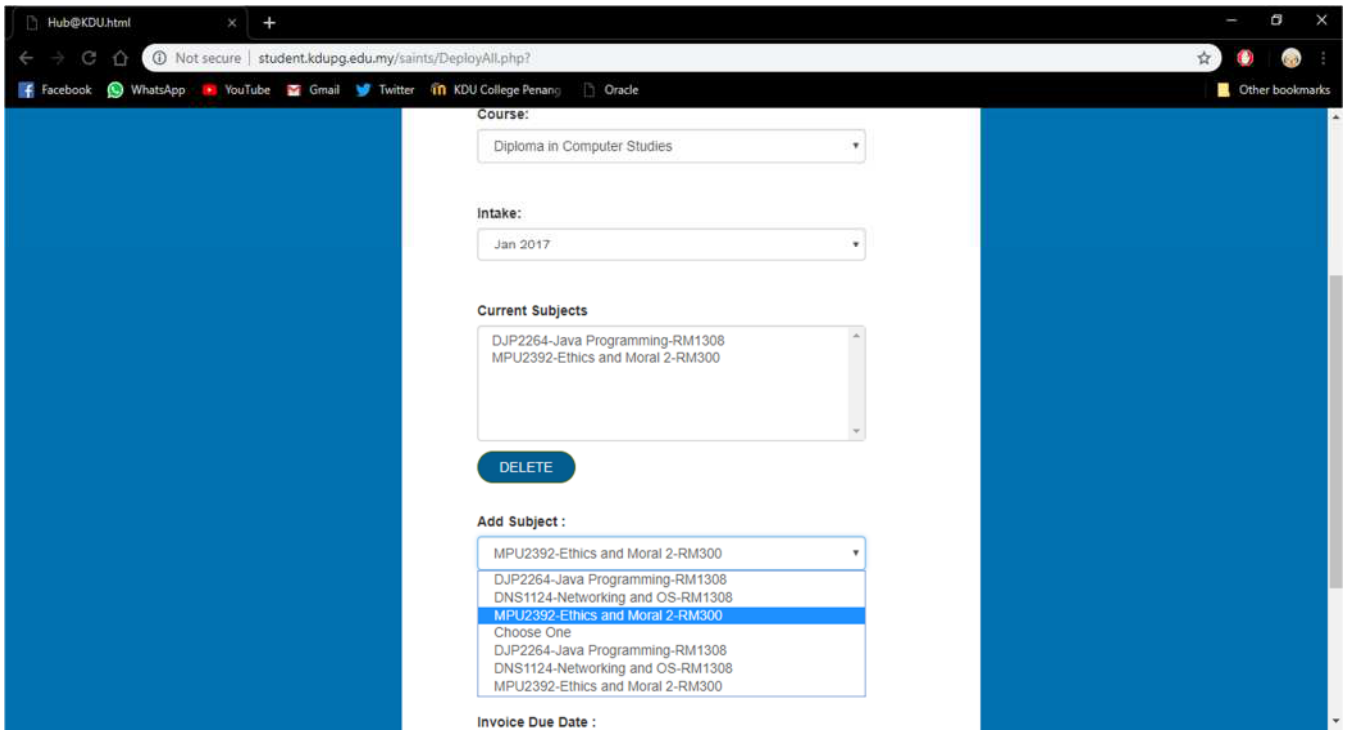


Diagram 6.1.1.1

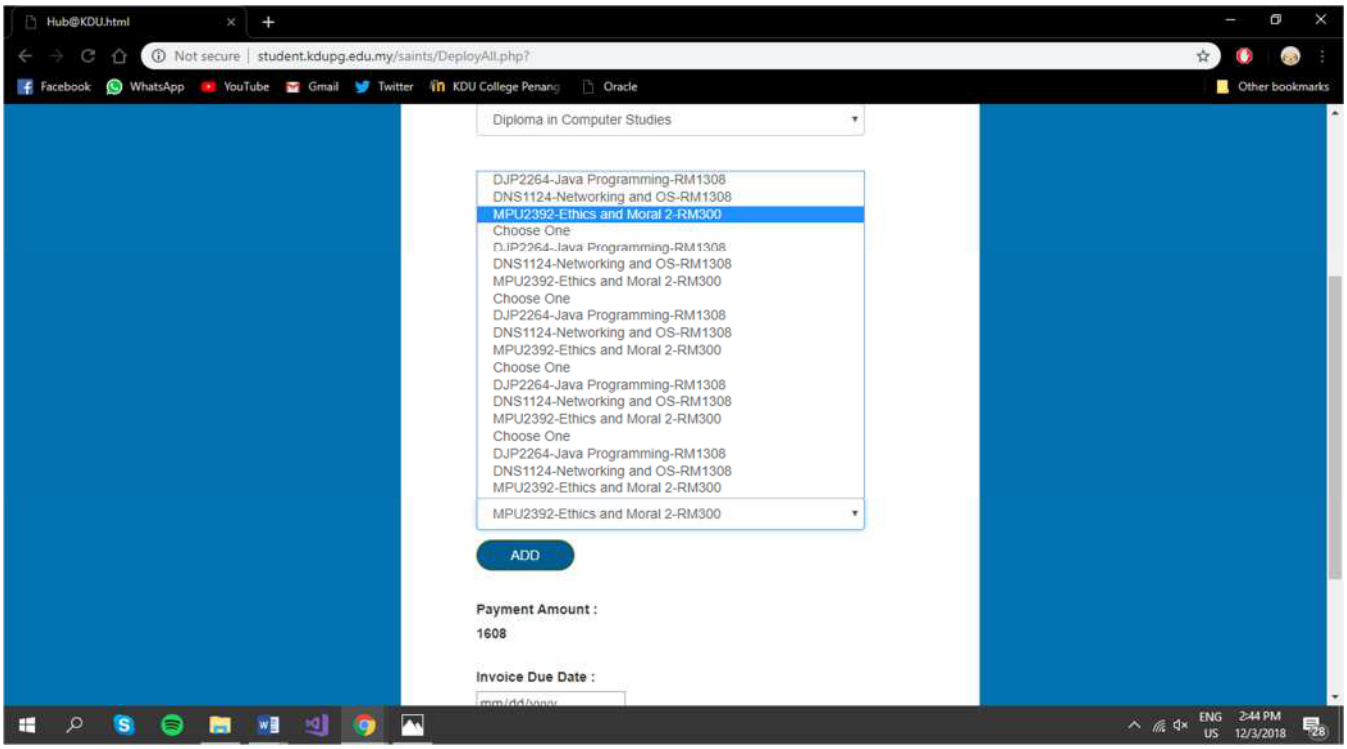


Diagram 6.1.1.2

In the two screenshots above, one of the errors found when deploying the invoice was that if users changes the intake field, the Subject dropdown list keeps getting refreshed because every time the user changes the value in the intake field, it triggers the action for the Subject dropdown list and it doesn't clear the previous items in the list so it keeps adding them. So, we just added a line of codes which clears all the items in the list before adding them. So now every time it is triggered the first line of code it executes is the line which clears all the items in the dropdown list then only it adds the necessary subjects in it.

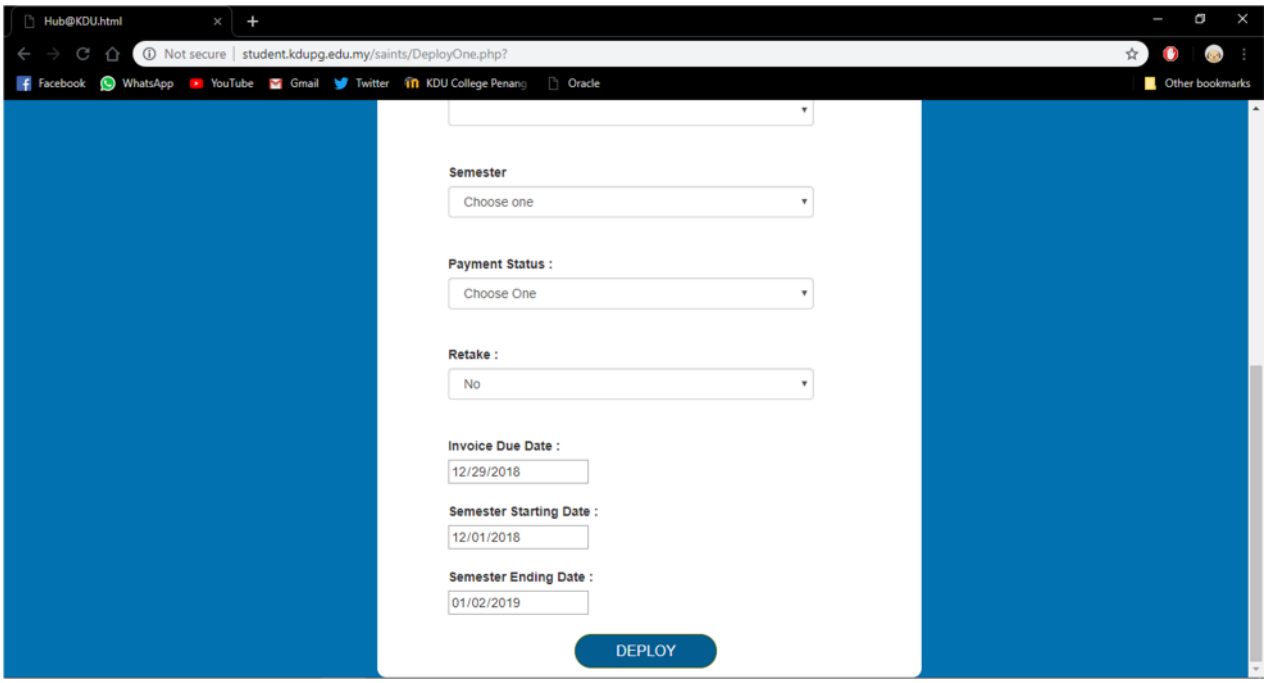


Diagram 6.1.1.3

Diagram 6.1.1.4

Another error found was that for deploying invoice for both deploy one and deploy many students, if admins just fill up the date field and ignore the rest of the fields and then press the deploy button, the system will redirect to a blank white page as shown in Diagram 6.1.1.3. Therefore the problem that lead to this huge error was because the deploy button was coded inside the form. When the button is pressed, since the button is inside the form, the

button together with the whole form gets submitted without manually checking whether the required fields have been filled in. The codes behind the button do not work leading to the blank white page. The solution to this was putting the button outside the form so that when the deploy button is pressed, the codes behind the button to check all fields are filled in before submitting the form will work.

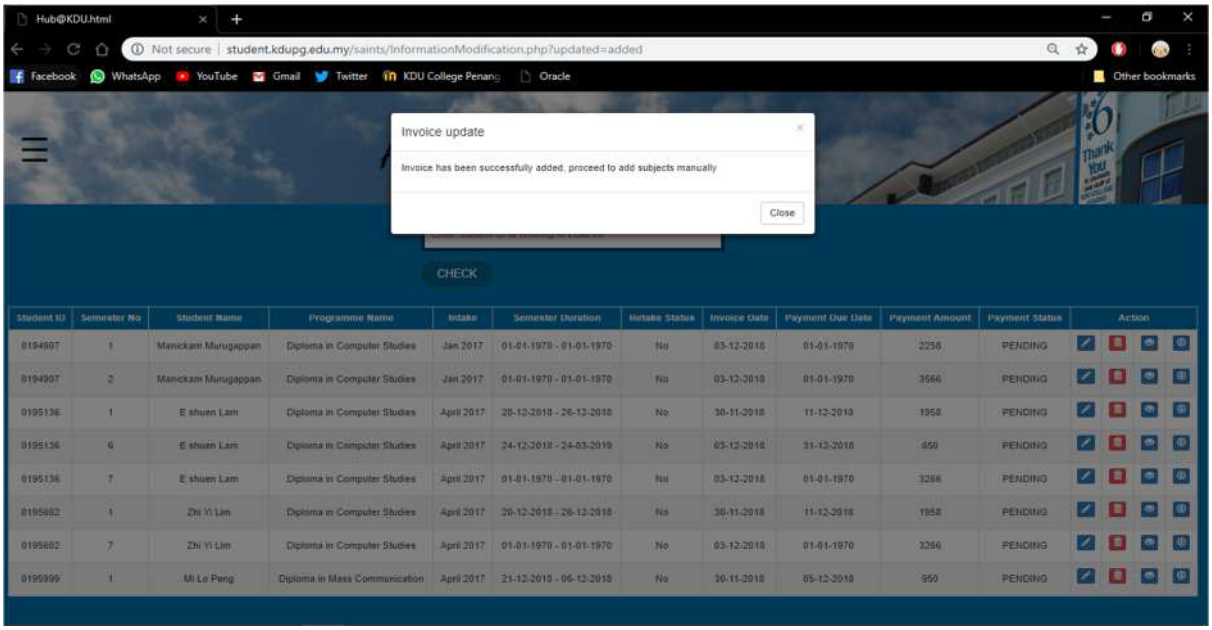


Diagram 6.1.1.5

Another error was that in deploy invoice for many students, if admins leave the date field out in the form and press deploys, the invoice will still be deployed as shown in Diagram 6.1.1.4. In the diagram, the payment due date for some of the invoices has year 1970 which is totally wrong. Therefore the error was that the team forgot to check that the date validation where it needed to be filled up as well before deploying the invoice. Therefore, the team ensured to implement the code where date needs to be filled up as well before deploying the invoice for many students.

Another error found was related to the side navigation bar. The hitbox for triggering the side navigation bar was not accurate. Therefore, wherever admins click on the image at the header of the website, the side navigation bar will slide in from the left as well and not only when admins click on the 3 horizontal lines that represents the side navigation bar. To fix this, we fix the CSS that handled the width and height variable to fix the width and height of the hit

box ensuring that only when admins click on the 3 horizontal lines, the side navigation bar will load.

6.2 System Testing (Mobile)

The system testing done for mobile platform requires minimum testing as well. Most of the functions available in the mobile application do not require input manually from students. They only need to navigate through the functions of the mobile platform using buttons on the layouts and the side bar navigation menu. The online payment requires input from students regarding card details for example but since the payment will be implemented using a fake payment gateway because legit payment gateways need certain formal procedures that have to be done officially to be implemented, the team will not be testing the payment with blackbox testing since it is unnecessary. Therefore, hardcore testing will only be done for the login page which will be shown in the blackbox testing later on.

6.2.1 Errors Found during System Testing (Mobile)

Due to time constrain on the project, the mobile application was coded and tested on the spot. Therefore, no rounds of testing were done on the mobile application but the team ensured that as we were coding the app, that there will be no errors. Errors that were found during coding were fixed on the spot, therefore no screenshots of the errors were taken for the documentation.

6.3 Blackbox Testing

Our team decided to implement backbox testing because this type of testing attempts to find errors or flaws in the systems which may trigger behaviour or performance errors which will in conclusion make the programme crash because the software is tested without knowing the internal structure of code or the programme. Therefore, we will be testing the system as if we are the users who have no knowledge of the internal coding structure of the system. Below will be the blackbox testing done for both platforms.

6.3.1 Blackbox Testing (Website)

6.3.1.1 Login Page

Input Condition	Valid Equivalence Class	Valid Boundaries	Invalid Equivalence Class	Invalid Boundaries
Username				
Type	Integer		Non-Integer(3)	
Size	7	7(1)	<7(4), >7(5)	6(6),8(7)
Password				
Type	String		Empty(8)	
Size	9	9(2)	<9(9),>9(10)	8(11),10(12)

Valid Test Case	Boundaries Covered
• **0195888**	1
• **8888@Abcde**	2

Invalid Test Case	Boundaries Covered
• **Abc** • **123456** • **12345678**	3,4,5,6,7
• **(White Space)** • **12345678** • **0123456789**	8,9,10,11,12

6.3.1.2 Invoice List / Main Page (Student ID field)

Input Condition	Valid Equivalence Class	Valid Boundaries	Invalid Equivalence Class	Invalid Boundaries
studentID				
Type	Integer		Non-Integer(2)	
Size	7	7(1)	<7(3), >7(4)	6(5),8(6)

Valid Test Case	Boundaries Covered
• **0195888**	1

Invalid Test Case	Boundaries Covered
• **Abc** • **123456** • **12345678**	2,3,4,5,6

6.3.2 Blackbox Testing (Mobile)

6.3.2.1 Login Page

Input Condition	Valid Equivalence Class	Valid Boundaries	Invalid Equivalence Class	Invalid Boundaries
Username				
Type	Integer		Non-Integer(3)	
Size	7	7(1)	<7(4), >7(5)	6(6),8(7)
Password				
Type	String		Empty(8)	
Size	9	9(2)	<9(9),>9(10)	8(11),10(12)

Valid Test Case	Boundaries Covered
• **0195888**	1
• **8888@Abcde**	2

Invalid Test Case	Boundaries Covered
• **Abc** • **123456** • **12345678**	3,4,5,6,7
• **(White Space)** • **12345678** • **0123456789**	8,9,10,11,12

Chapter 7: Conclusions and Future Recommendation

7.1 Conclusions

In conclusion, this project, Hub@KDU was aimed to help students' and admins' of KDU Penang University College overcome the problems faced with the current system practiced for invoice collection and invoice payment. It was also aimed to increase admins' efficiency in terms of notifying students with new and important announcements using the mobile application. Unfortunately, the team was not able to implement this announcement feature due to time constraint faced by the team but will be looking to implement in the upcoming versions of Hub@KDU. For invoice collection and invoice payment for students, they can now use the mobile application to receive their invoice electronically and do not need to manually collect the invoice themselves during invoice collection day. Admins need only to use the website to deploy invoice and students will receive the invoice through the mobile application. For payment, the concept of the fake payment gateway has been implemented and the logic is there meaning that when students presses the pay now button for pending invoices, the application will notify students that the invoice has been successfully paid but any real transactions of payment for the invoice has not been implemented due to formal procedures that is needed to implement an official payment gateway system. Therefore, our e-payment function will be worked on in upcoming versions of the application as well.

Compared to the current system practiced at KDU Penang University College, the system the team developed has functions that can solve problems such as manually collecting invoice as well as manually paying the invoices. As this project is for educational purposes in terms of our final year project, our features may currently lack the standard or quality to be fully implemented as the main system for the college but the features and the way Hub@KDU works has been implemented and they can be further improved on. Therefore, the team will continue to work on perfecting these features and upcoming features in the upcoming versions of Hub@KDU so that this system can be truly considered to be the system that the college can implement to replace the current system that is adhere at the college.

Last but not least, we would like to thank our supervisor for this project, Ms Norayu Binti Abdul Talib for truly guiding us towards the completion of this project. All her comments were crucial in ensuring that the project could be completed during the 14 weeks duration we

had and we are truly grateful that she understood our team's decision to drop the announcement feature due to time constraint. Finally, the team would just like to say thank you to the team members that have worked hard and contributed towards the completion of this project. Although our system may currently lack some functionalities and maybe our User-Interface Design can be considered simple as well, but we will continue to strive and work hard on Hub@KDU to improve the system as a whole.

7.2 Future Recommendation

As stated in the final implementation phase in the process of system development, the team was not able to implement the announcement feature due to time constraint. Therefore, the team will be looking to implement the announcement feature in the coming versions of Hub@KDU. We will be looking to implement the initial idea the team had for the announcement feature which was admins will be able to post announcements through the website and the students will be notified of the announcements through the mobile application. Besides that, we also want to successfully implement a real payment gateway in both the payment methods that are currently available in the mobile application that is the online banking and card payment for students when they proceed to making payment through the application to pay their invoices. Another feature we want to try and implement is that we want to implement notifications for students when the payment due date for their invoice is near and they have not paid yet. There were plans to implement this feature in the implementation phase but the team just could not seem to get it to work. We hope to implement this feature in the future version of Hub@KDU.

In conclusion, these are the features we want to implement in the upcoming versions of Hub@KDU as there was no time to implement it during the timespan of this project. We already have the idea therefore we will need to focus and work hard on successfully implementing it. We will also aim to improve the user-interface design of the system in upcoming versions.

References

Anon., 2018. *USER GUIDE: Meet Android Studio.* [Online] Available at: https://developer.android.com/studio/intro/ [Accessed 4 December 2018].

Anon., n.d. *5 Technology Trends You Need to Know to Work in Any Industry: Way Up.* [Online] Available at: https://www.wayup.com/guide/deloitte-313295-sponsored-1-5-technology-trends-need-know-work-industry/ [Accessed 13 September 2018].

Anon., n.d. *JetBrains: ENJOY PRODUCTIVE JAVA.* [Online] Available at: https://www.jetbrains.com/idea/ [Accessed 4 December 2018].

Anon., n.d. *smartdraw: Entity Relationship Diagram.* [Online] Available at: https://www.smartdraw.com/entity-relationship-diagram/ [Accessed 5 December 2018].

Anon., n.d. *Squarespace: What is JSON?.* [Online] Available at: https://developers.squarespace.com/what-is-json [Accessed 4 December 2018].

Anon., n.d. *What is Virtual Reality?: Virtual Reality Society.* [Online] Available at: https://www.vrs.org.uk/virtual-reality/what-is-virtual-reality.html [Accessed 18 September 2018].

danny.chen, 2017. *Object Oriented System Analysis and Design*, s.l.: s.n.

ADMIN USER MANUAL

TABLE OF CONTENT

1.0 LOGIN PAGE

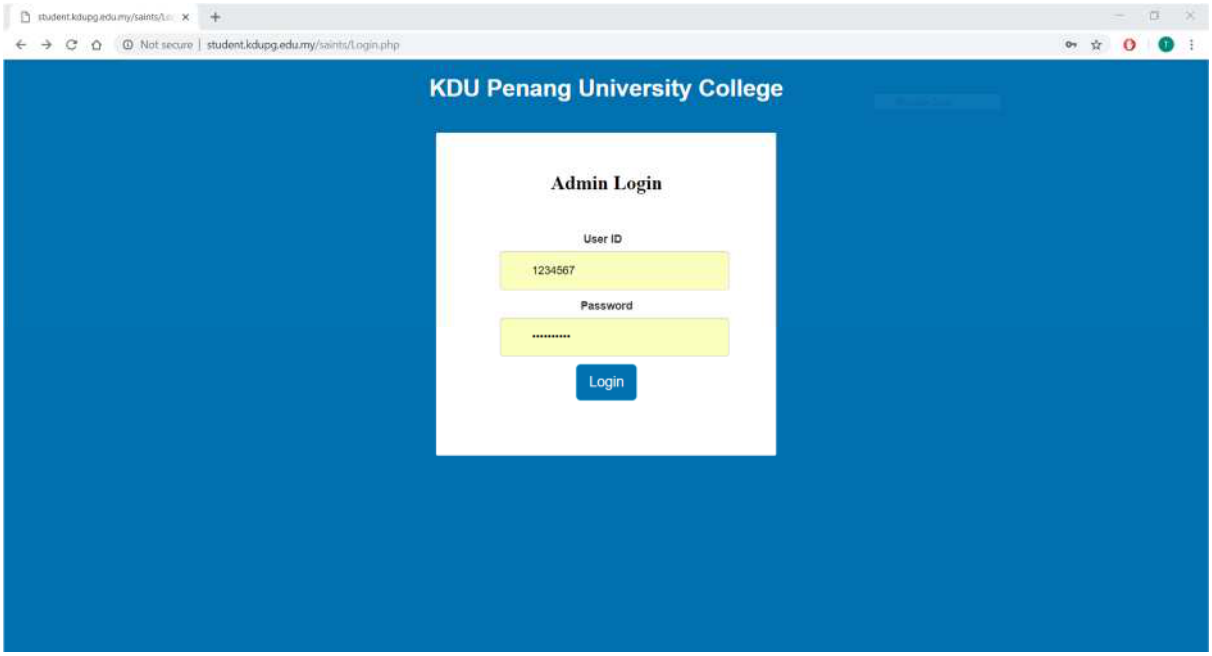


Diagram 1.1

According to the diagram above user is required to insert their respective user ID and password to be granted further access into the website. Click on the login button after inputting your credentials.

2.0 HOME PAGE

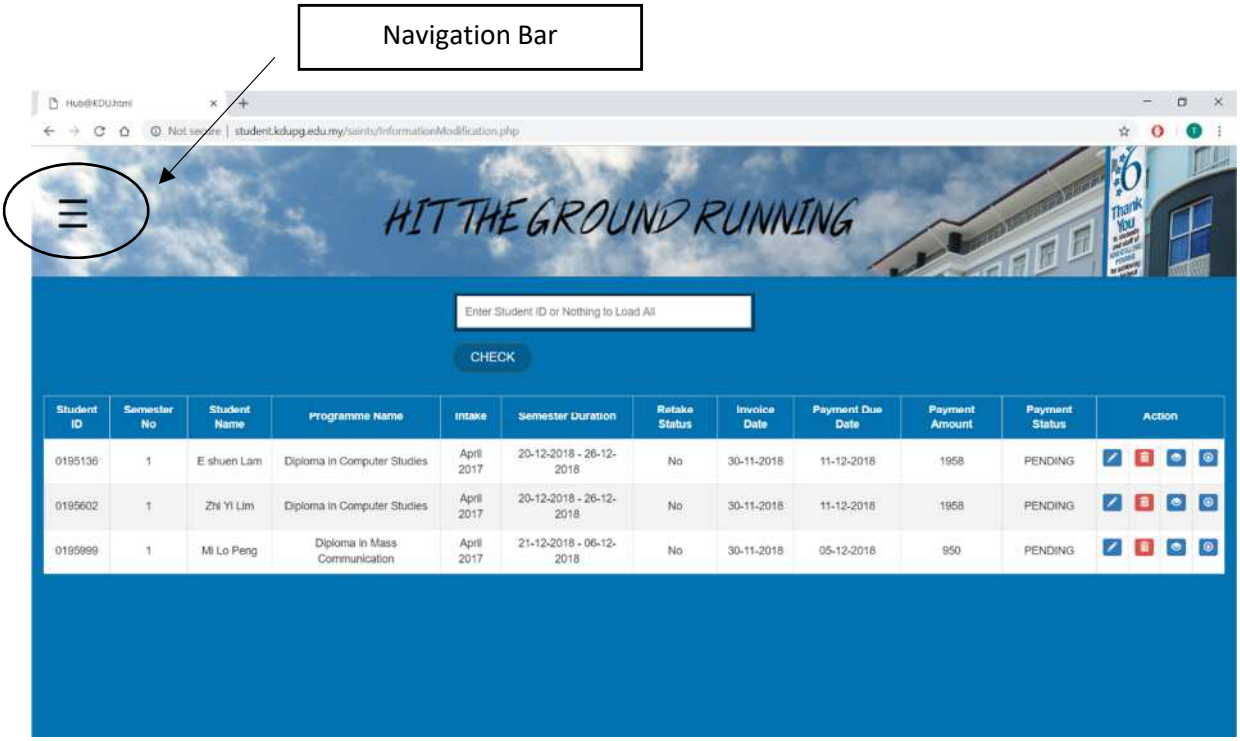


Diagram 2.1

After successfully logging in, you will be redirected to the home page as shown above. This page represents a list of current invoices that are currently stored in the database and also displays the side navigation bar. Click on the 3 horizontal lines that is circled in the diagram to view the options available on the side navigation bar.

3.0 SIDE NAVIGATION BAR

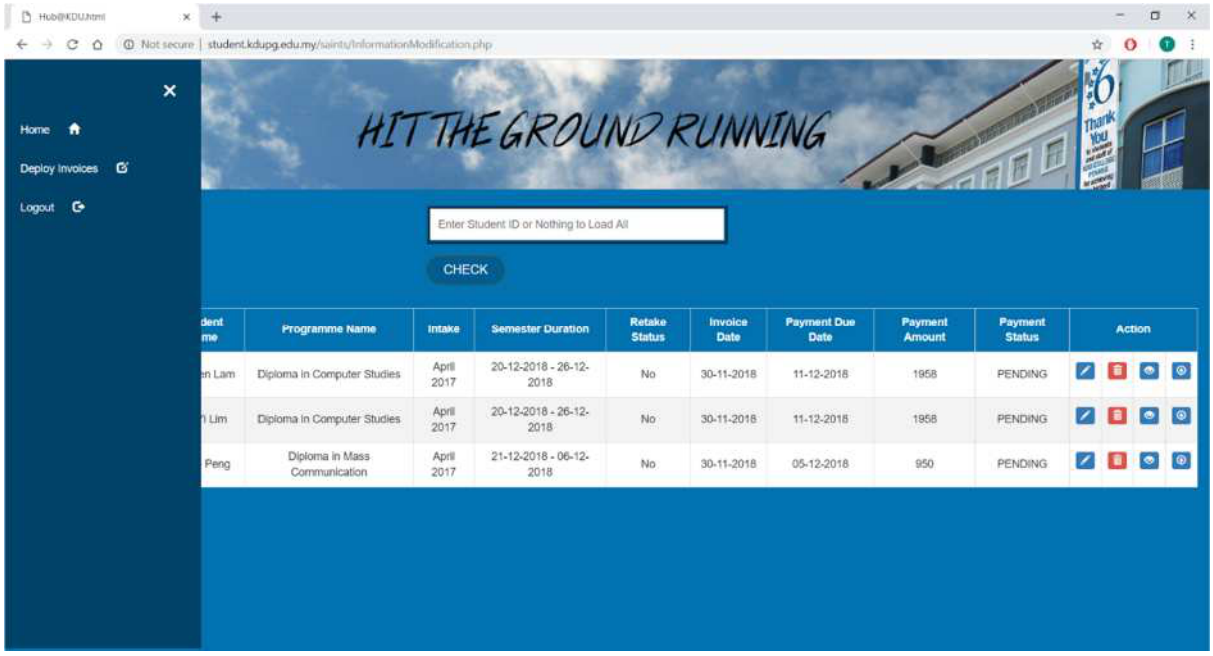


Diagram 3.1

The side navigation bar will slide in from the left side as shown above and there will be three different options that you can choose from the following. If you select **Home**, you will be redirected to the page in Diagram 2.1.

3.1 DEPLOY INVOICE

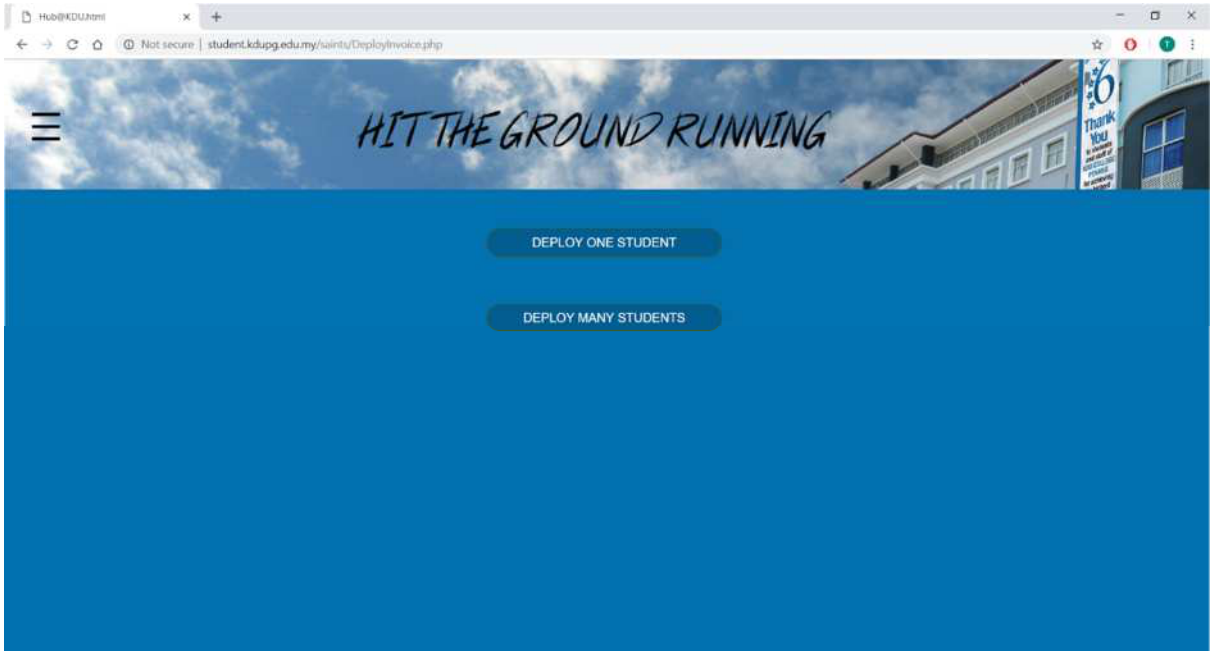


Diagram 3.2

If you select **Deploy invoices**, you will be redirected to this page as shown above where you will have to choose either to deploy one student's invoice or deploy many students' invoices. Further explanation regarding the two functions as shown in the button above are located under Section 4.0 Deploy Invoices (Page 7 for Deploy One and Page 10 for Deploy Many) .

3.2 LOGOUT

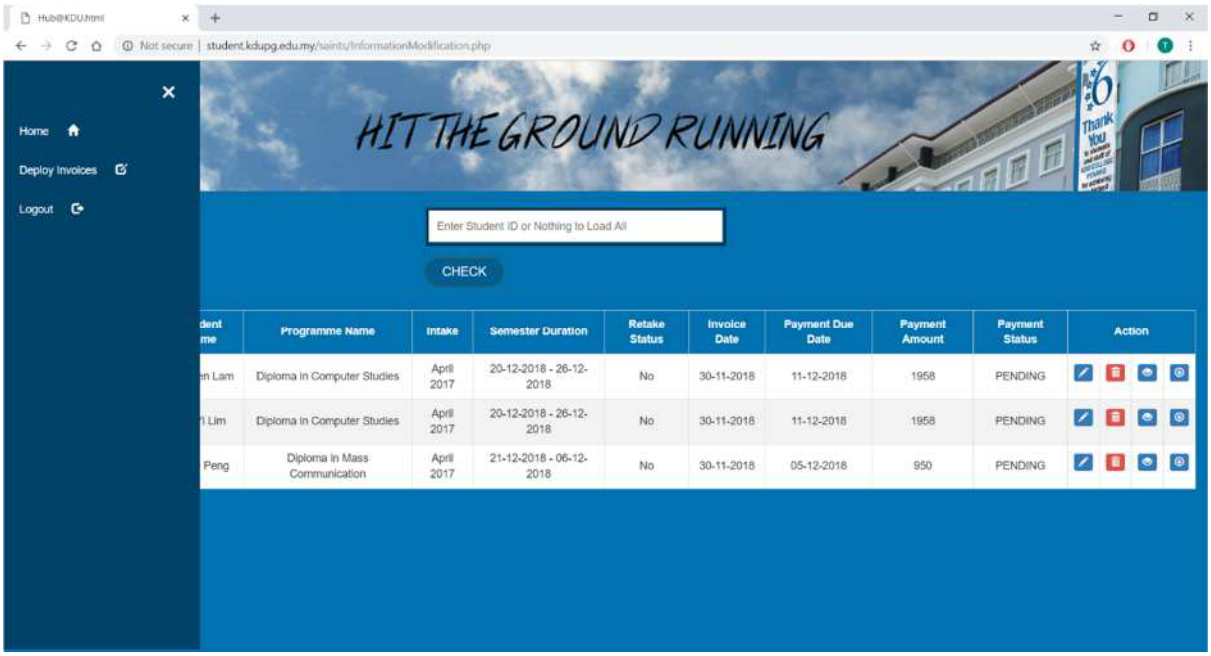


Diagram 3.3

If you select **Logout** then you will be logged out from the website and will be redirected to the page which is shown in Diagram 1.1.

4.0 DEPLOY INVOICE

4.1 DEPLOY ONE STUDENT'S INVOICE

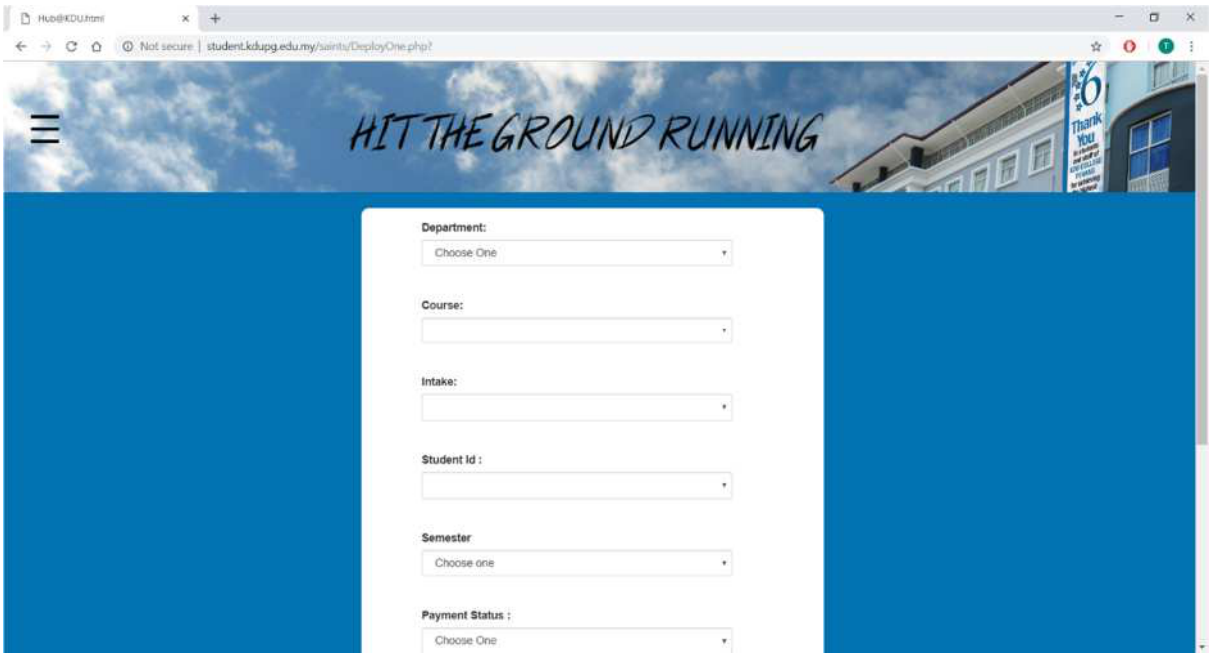


Diagram 4.1

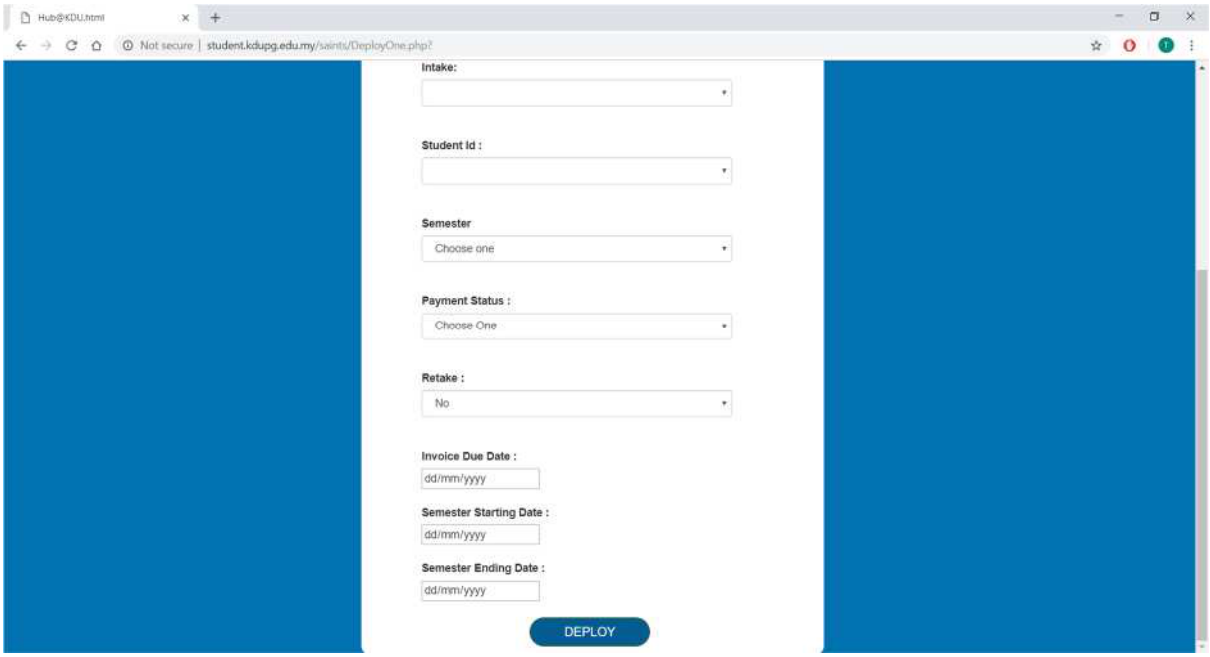


Diagram 4.2

If you were to click on the button which states deploy one student's invoice you will be redirected to the page shown in Diagram 4.1 / 4.2. The function Deploy One was created because admins might have unusual cases where students' invoices are not deployed or have any flaws / errors in it or invoices are accidentally deleted. As shown above you are required to fill up all the fields on the form to proceed further on. When filling up the form please fill in the **form** from **top to bottom** as other fields are dependent on the previous fields.

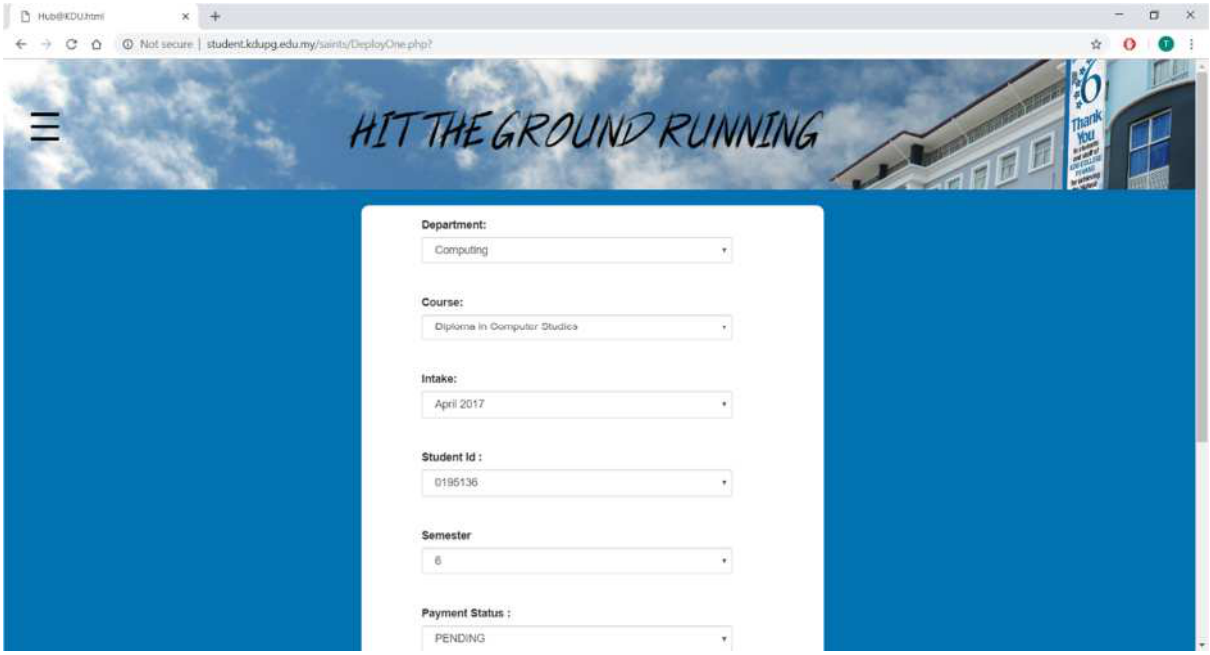


Diagram 4.3

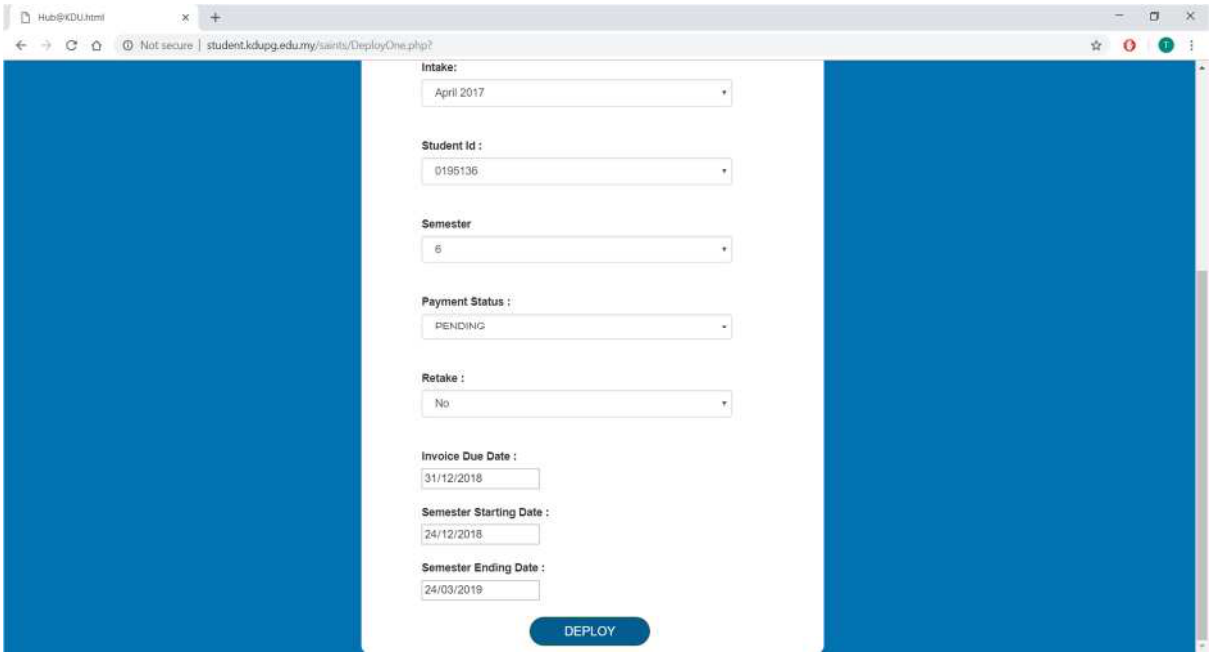


Diagram 4.4

Diagram 4.3 and Diagram 4.4 shows a form that has been fully filled as an example. After you have filled the form, click on the **DEPLOY** button.

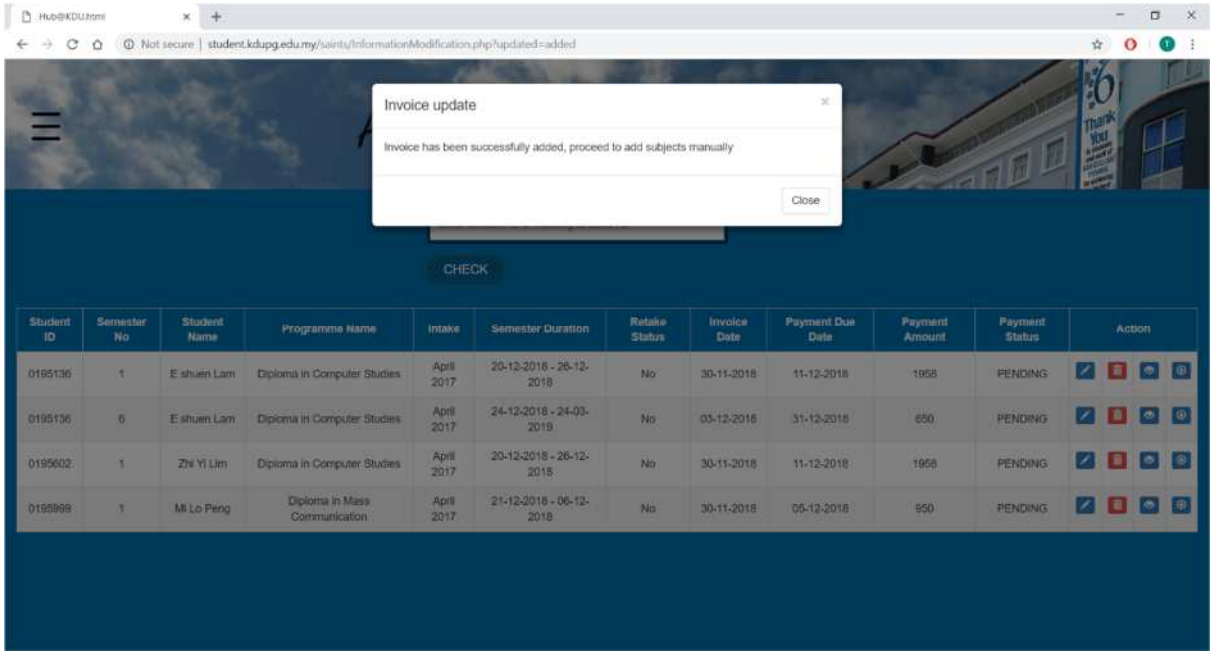


Diagram 4.5

Diagram 4.5 shows a pop-up message to notify users that the invoices is successfully deployed and are requested to add subjects manually which will be further explained under Section 7.1 Editing Invoices (Page 18).

4.2 DEPLOY MANY STUDENTS' INVOICE

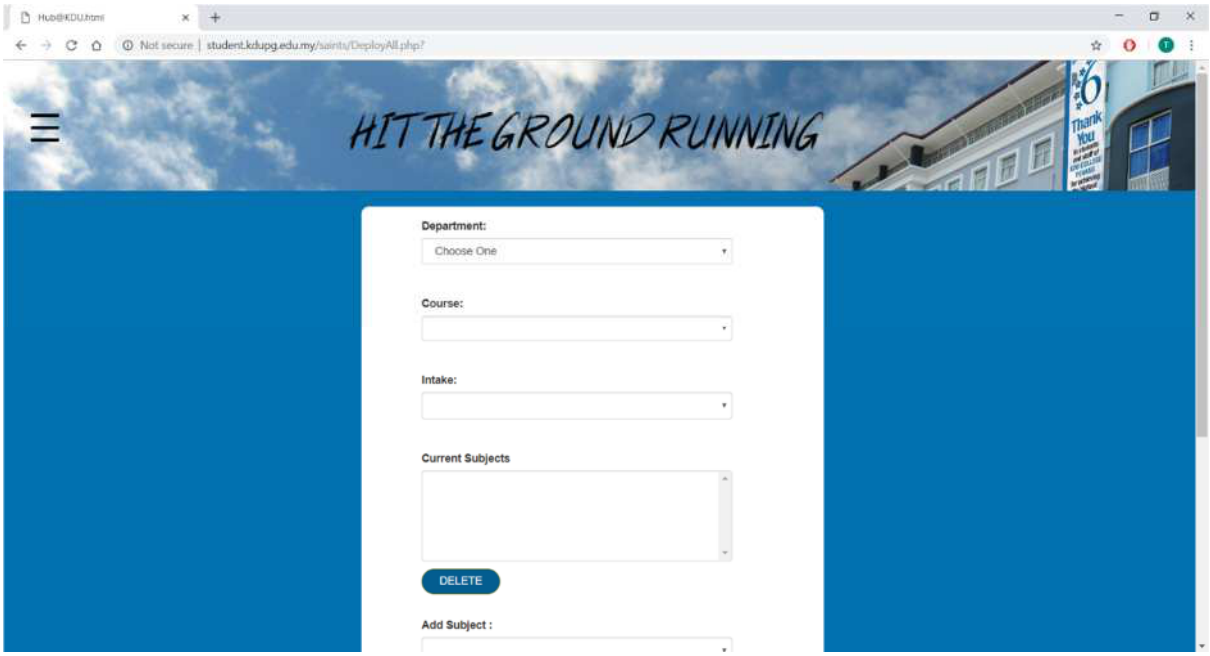


Diagram 4.2.1

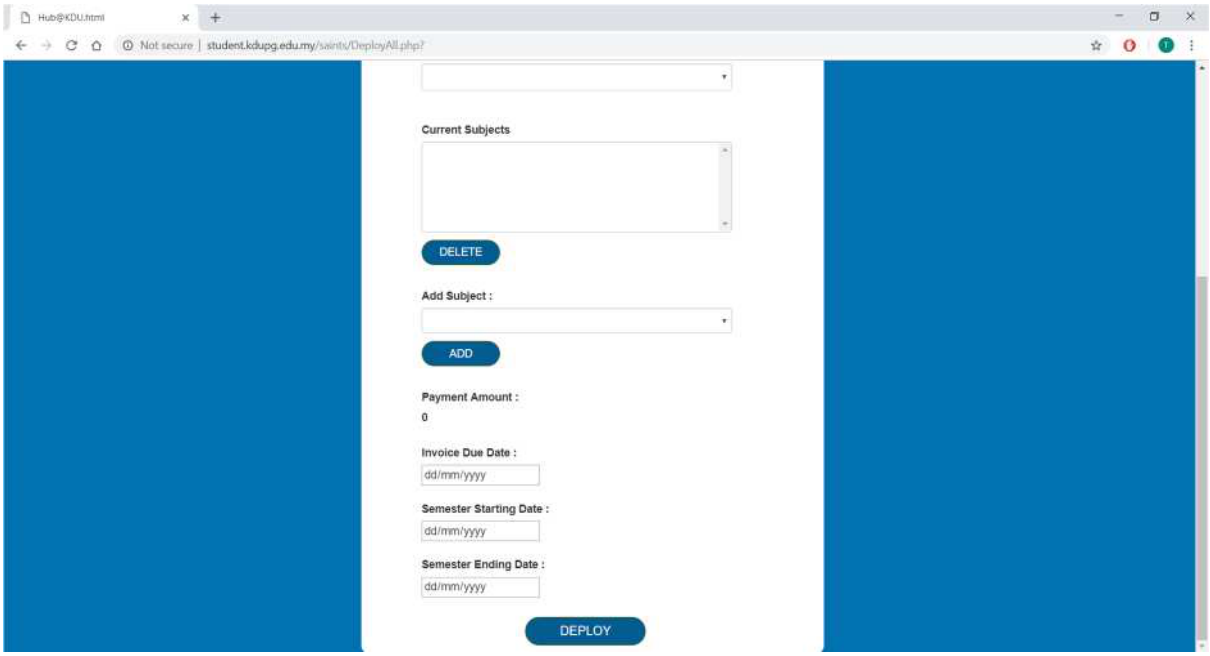


Diagram 4.2.2

If you were to click on the button which states deploy many students' invoice you will be redirected to the page shown in Diagram 4.2.1 / 4.2.2 As shown above you are required to

fill up all the fields on the form to proceed further on. When filling up the form please fill in the **form** from **top to bottom** as other fields are dependent on the previous fields.

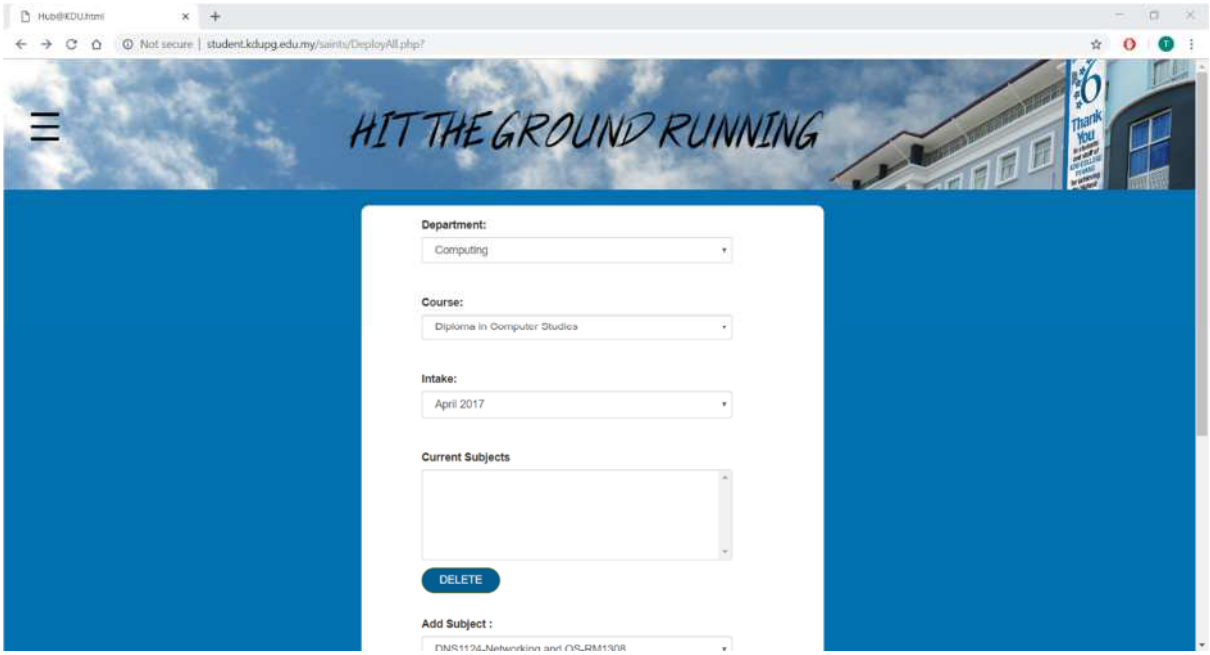


Diagram 4.2.3

You can first choose which department to deploy the invoice to, followed by the course that is related to the selected department and finally the respective intake based on the filter that is done as shown in Diagram 2.4.

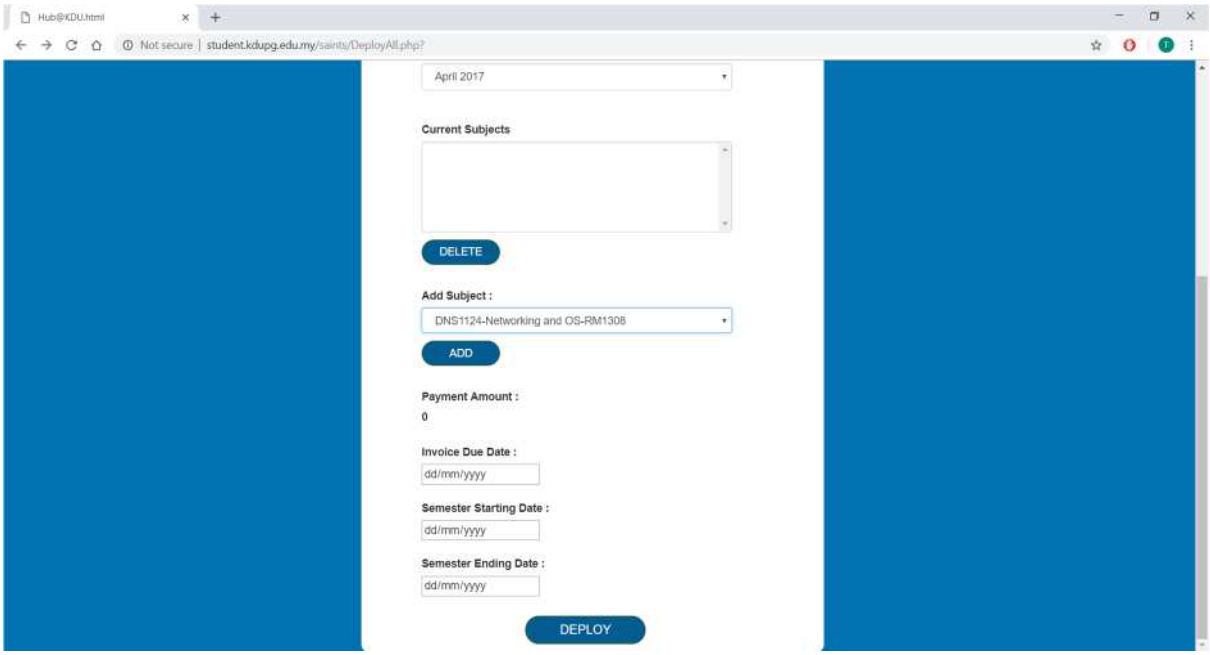


Diagram 4.2.4

You can then add the subjects available that students need to take in the add subject drop-down list box as shown in Diagram 4.2.2.

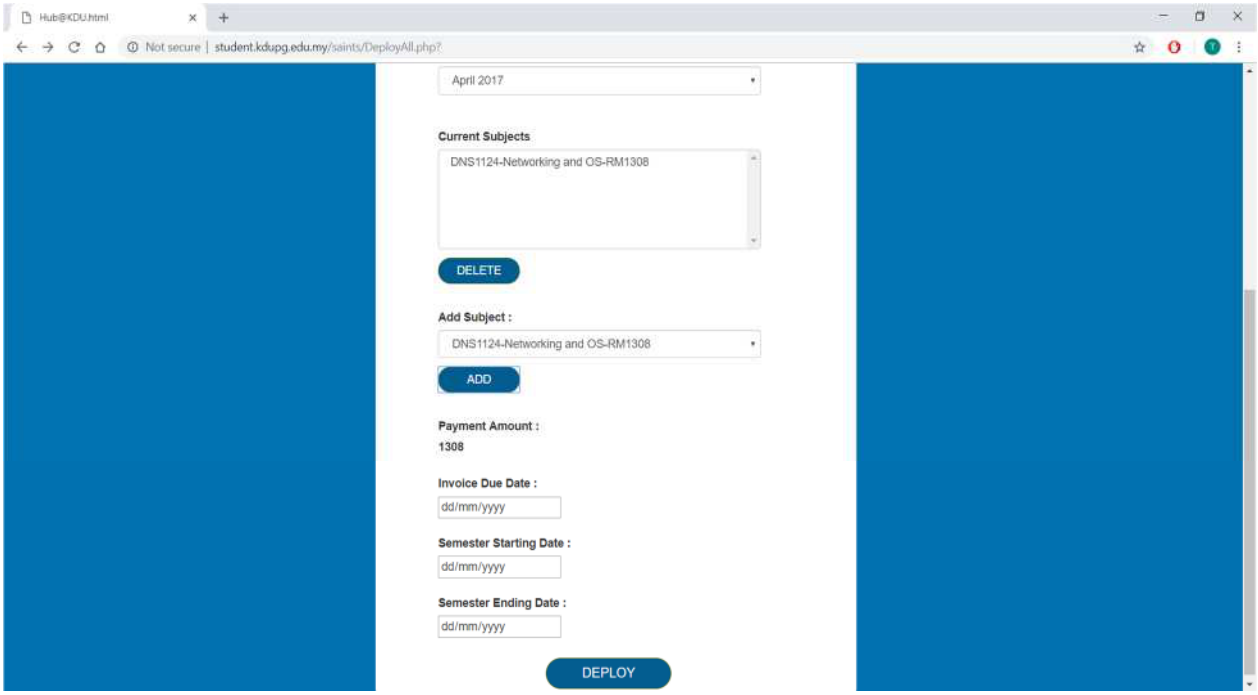


Diagram 4.2.5

The subjects will be added to the current subjects field which is non-editable so if you want to delete the subjects, please refer to Diagram 4.2.6.

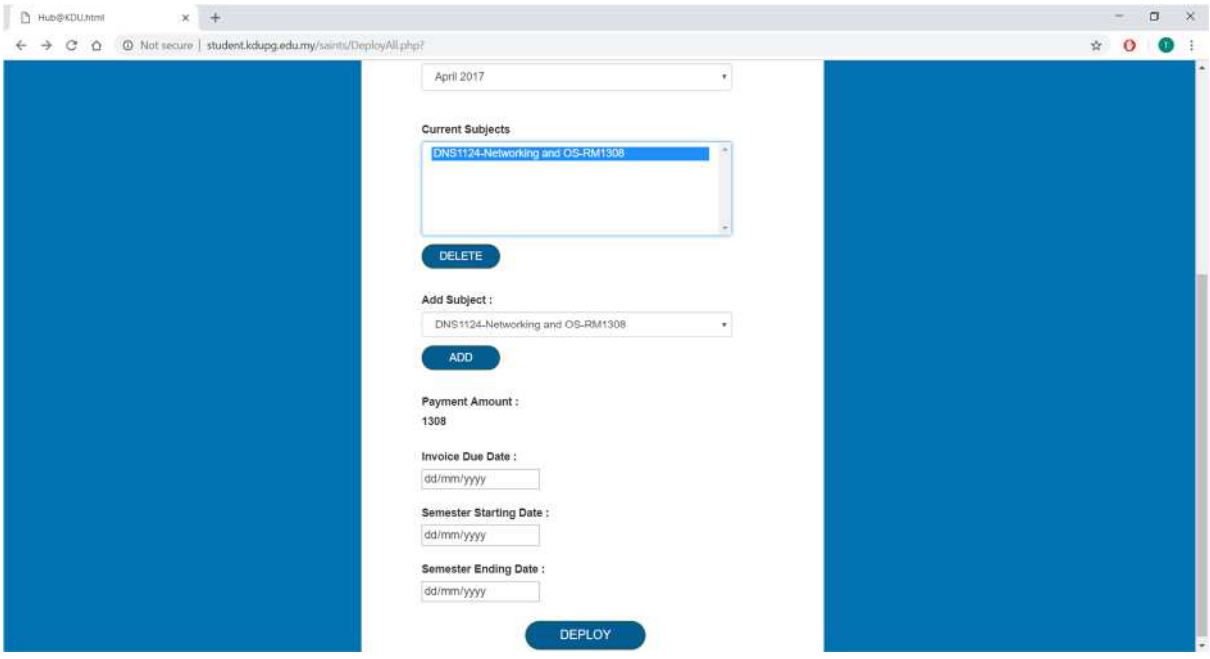


Diagram 4.2.6

Diagram 4.2.6 shows that you can delete the current subject if you accidentally added a wrong one. First, you must click on the subject that you want to delete. Then, after the subject is highlighted, click on the delete button.

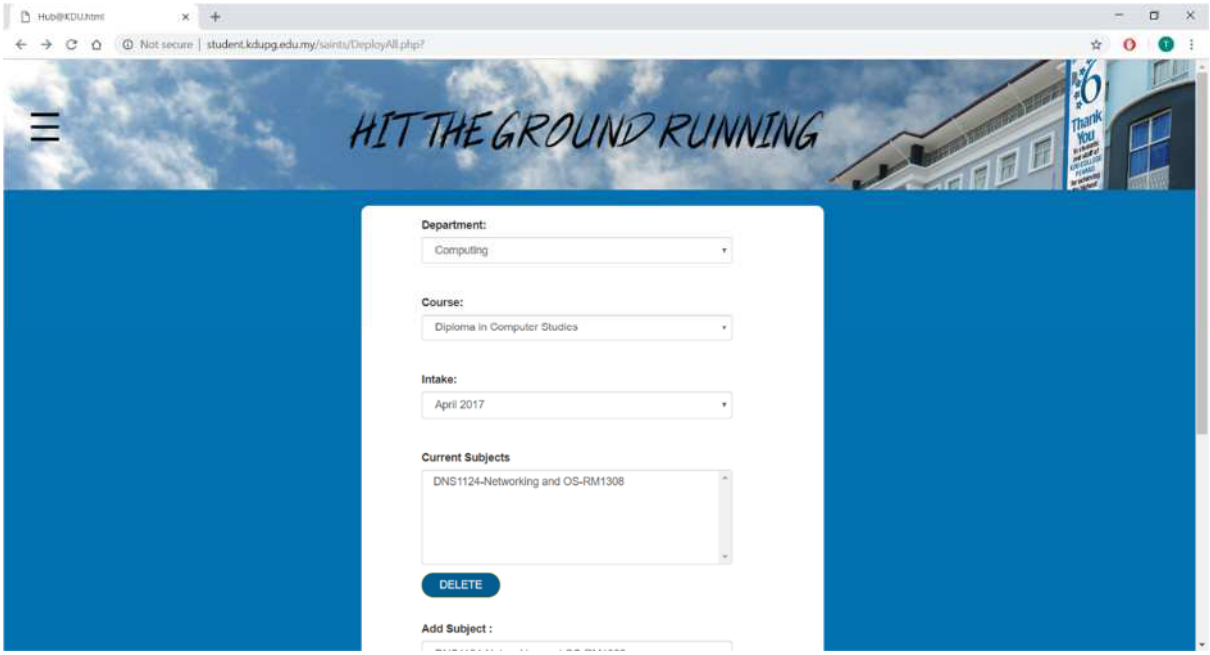


Diagram 4.2.7

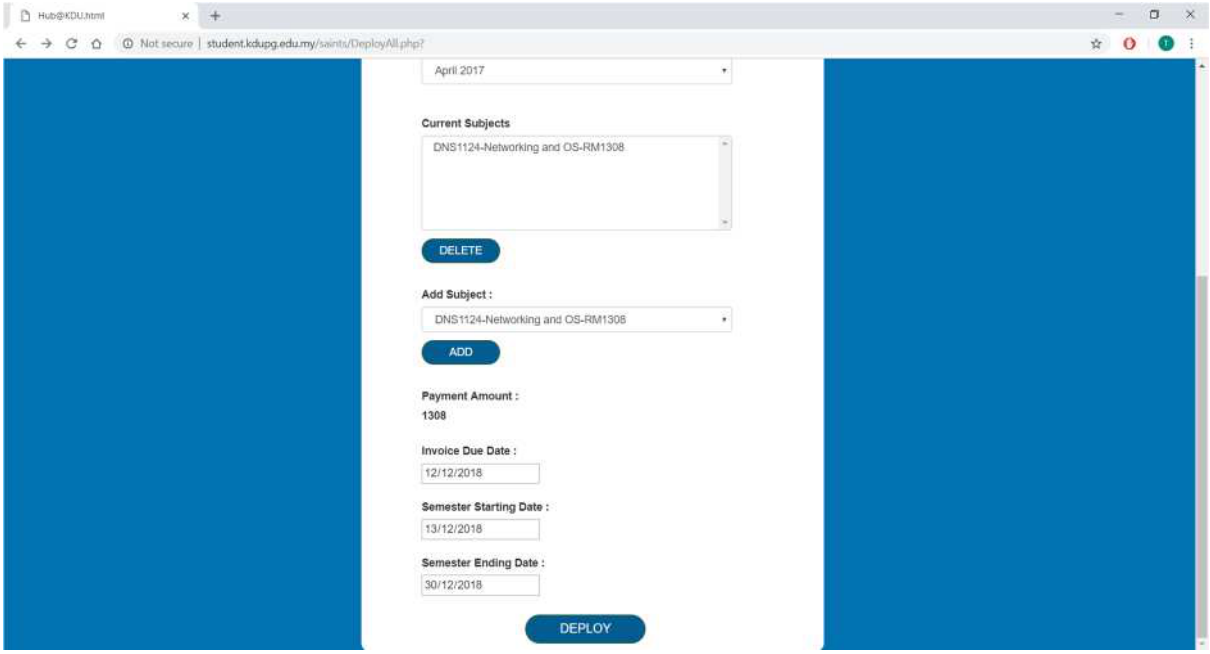


Diagram 4.2.8

Diagram 4.2.7 and Diagram 4.2.8 shows an example of a filled form. After you have completely filled the form, click on the **DEPLOY** button.

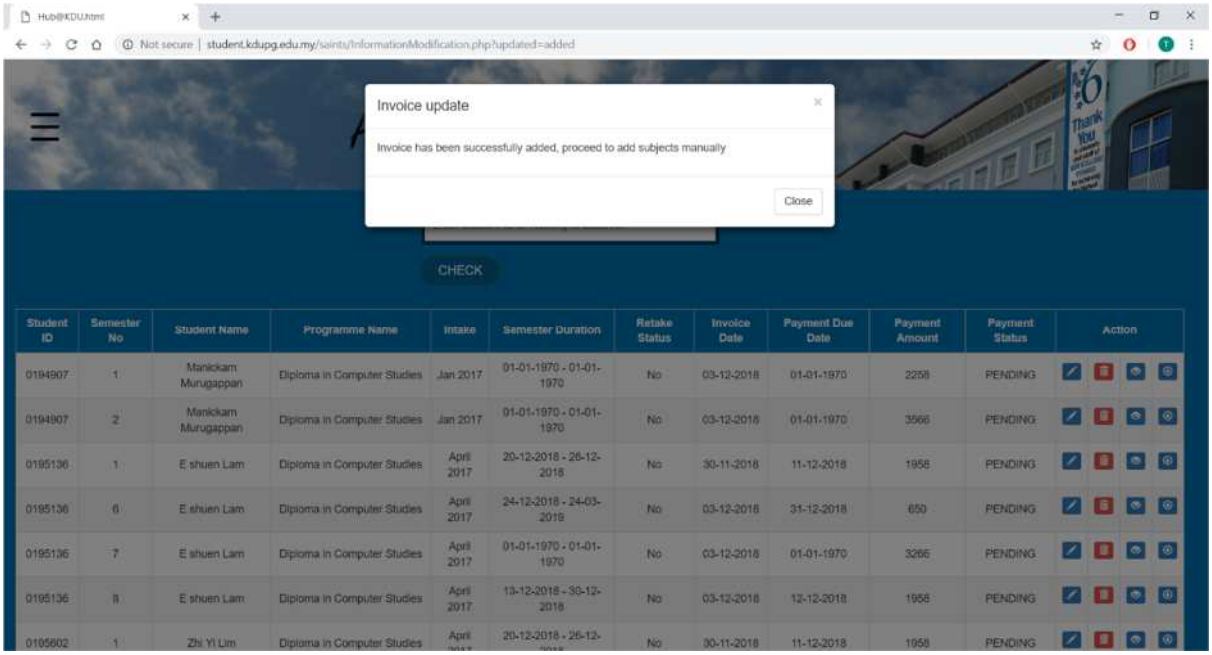


Diagram 4.2.9

The above diagram shows that the invoices are successfully deployed.

5.0 TO FILTER INVOICES BASED ON A PARTICULAR STUDENT ID

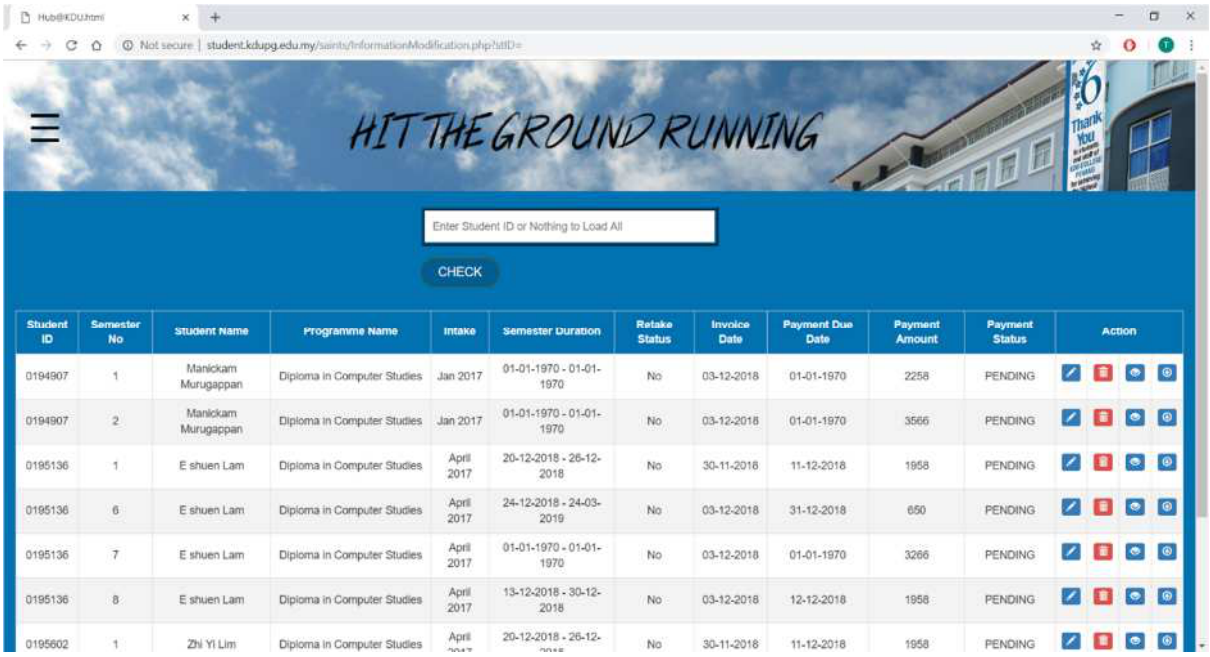


Diagram 5.1

Diagram 5.1 shows the main page of the website which consists of all the invoices that are deployed.

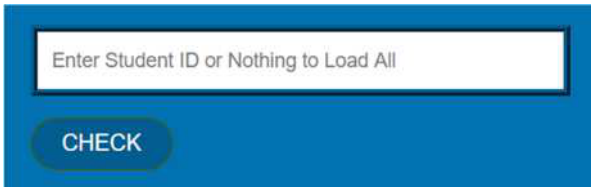


Diagram 5.2

Diagram 5.2 shows that you can enter the particular student's ID so that the results that are displayed are dynamic. Click on the check button after you have filled in the student ID field.

Diagram 5.3

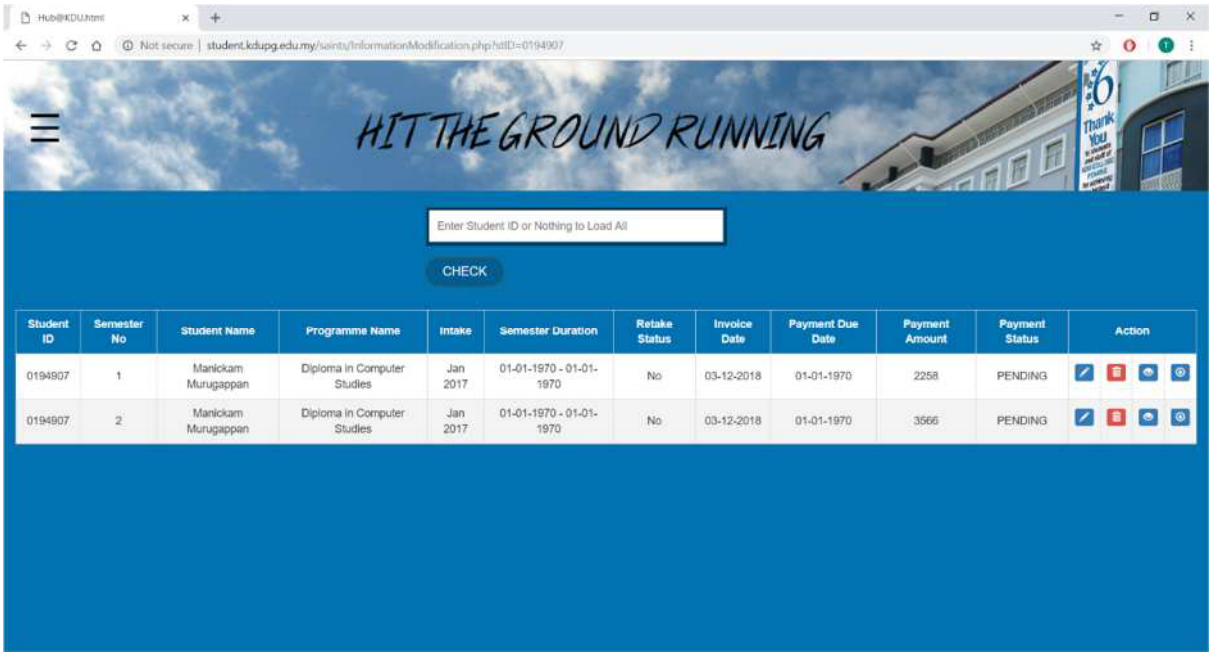


Diagram 5.4

Diagram 5.4 shows that the results are displayed based on the student ID keyed in the field which is shown in Diagram 5.3.

6.0 TO DISPLAY ALL INVOICES THAT HAVE BEEN DEPLOYED

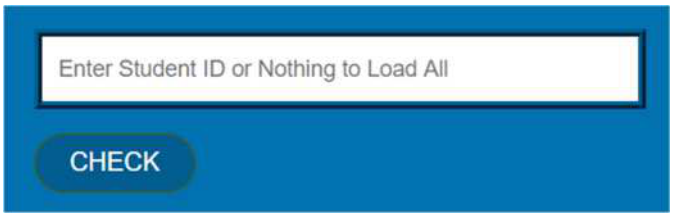


Diagram 6.1

To display all students' information, leave the student ID field empty and click the **CHECK** button.

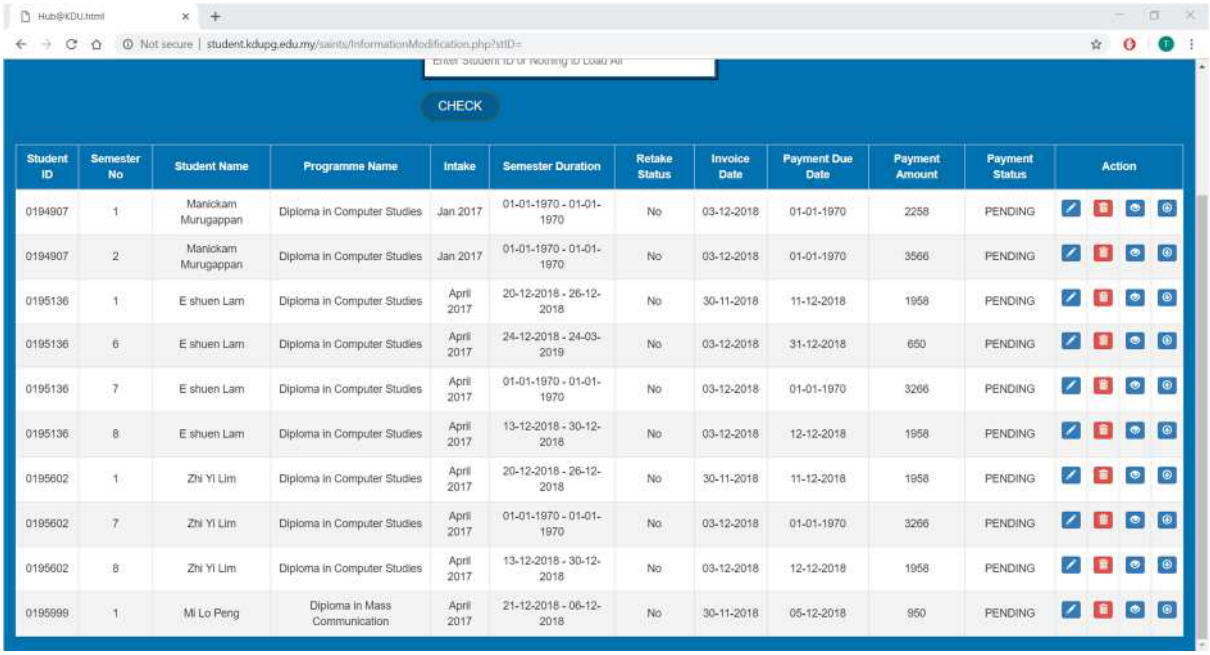


Diagram 6.2

All invoices that have been deployed will be shown as in Diagram 6.2

7.0 INVOICE ACTIONS

7.1 EDITING INVOICES

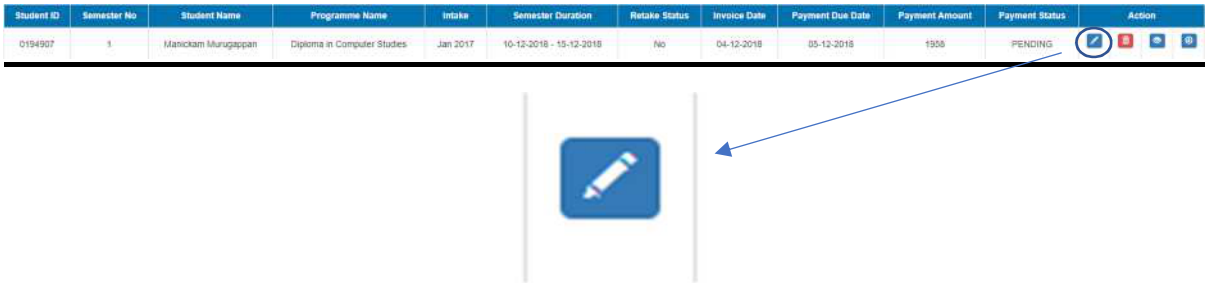


Diagram 7.1.1

If you want to edit a student's invoice or add subjects if you were to deploy invoice for one student click on the icon on the respective row of the invoice list as shown in Diagram 7.1.1.

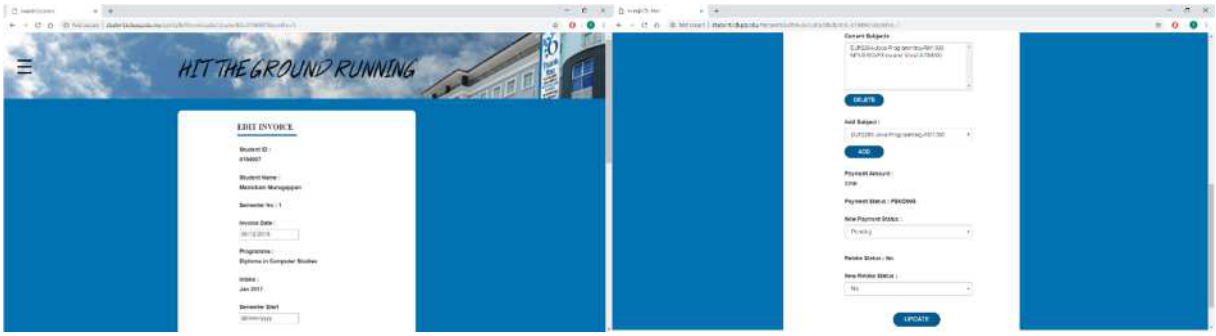


Diagram 7.1.2

In the diagrams shown above, it shows the edit page where you can edit the invoice date, semester start, semester end, payment date, delete current subject, add to current subject, change the payment status and change the new retake status. After the modification has been done, click on the **UPDATE** button.

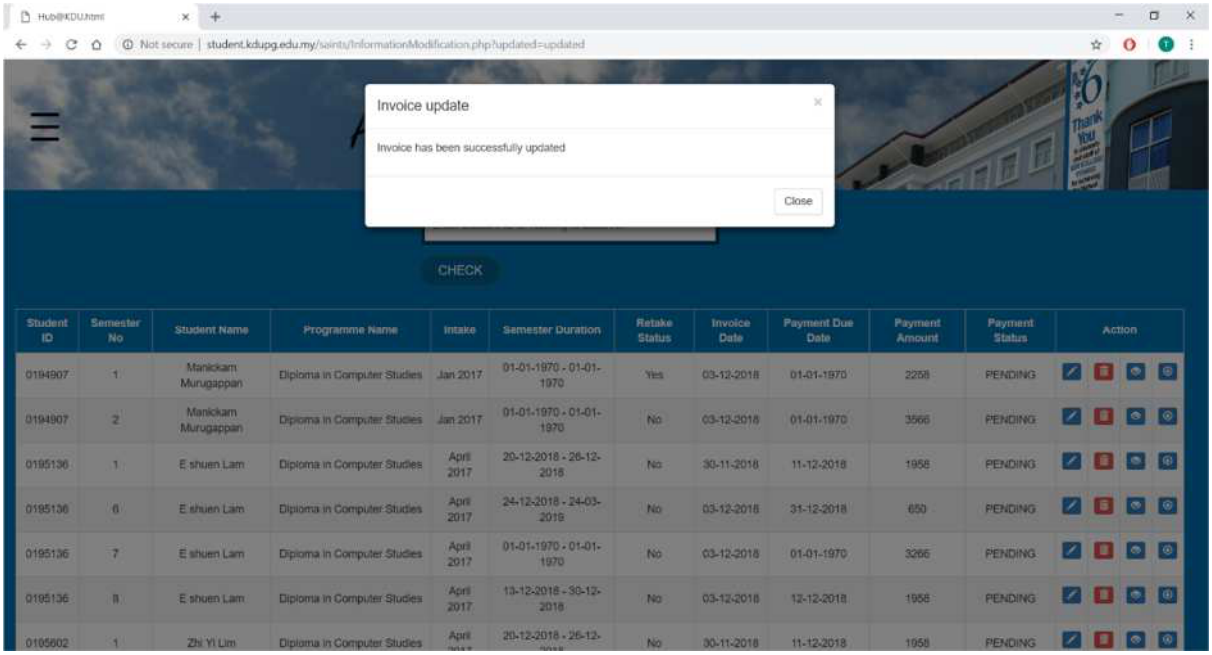


Diagram 7.1.3

Diagram 7.1.3 shows that you will be notified whether the invoice has been successfully updated.

7.2 DELETING INVOICES

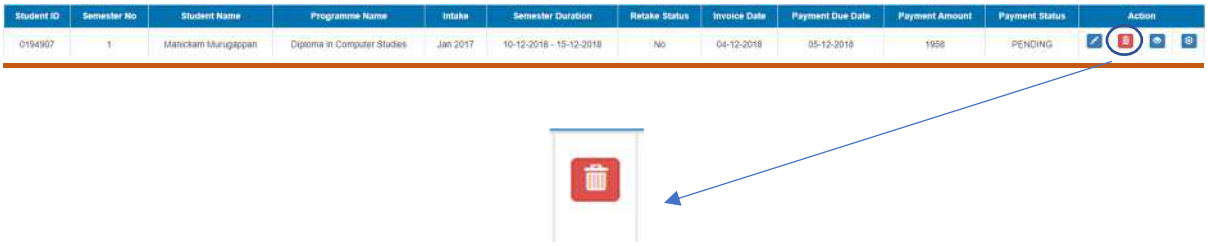


Diagram 7.2.1

You can click on the icon on the respective row of the invoice list which is shown in Diagram 7.2.1 to delete the particular invoice.

7.3 VIEWING INVOICES

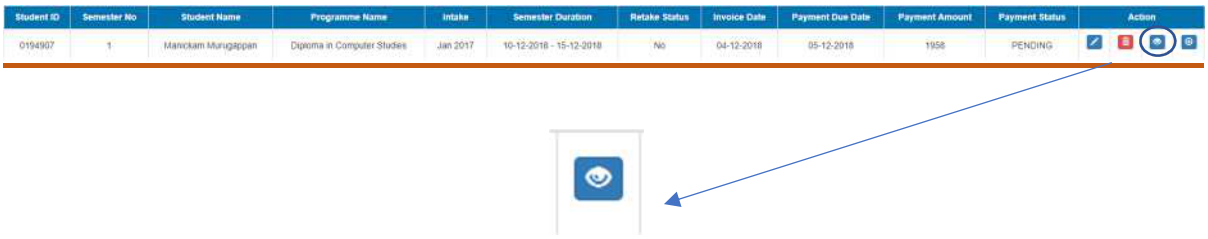


Diagram 7.3.1

You can click on the icon above on the respective row of the invoice list which is shown in Diagram 7.3.1 if you would like to view the respective student's invoice. When you have clicked on it you will be redirected to your default browser which displays the invoice in a pdf format.

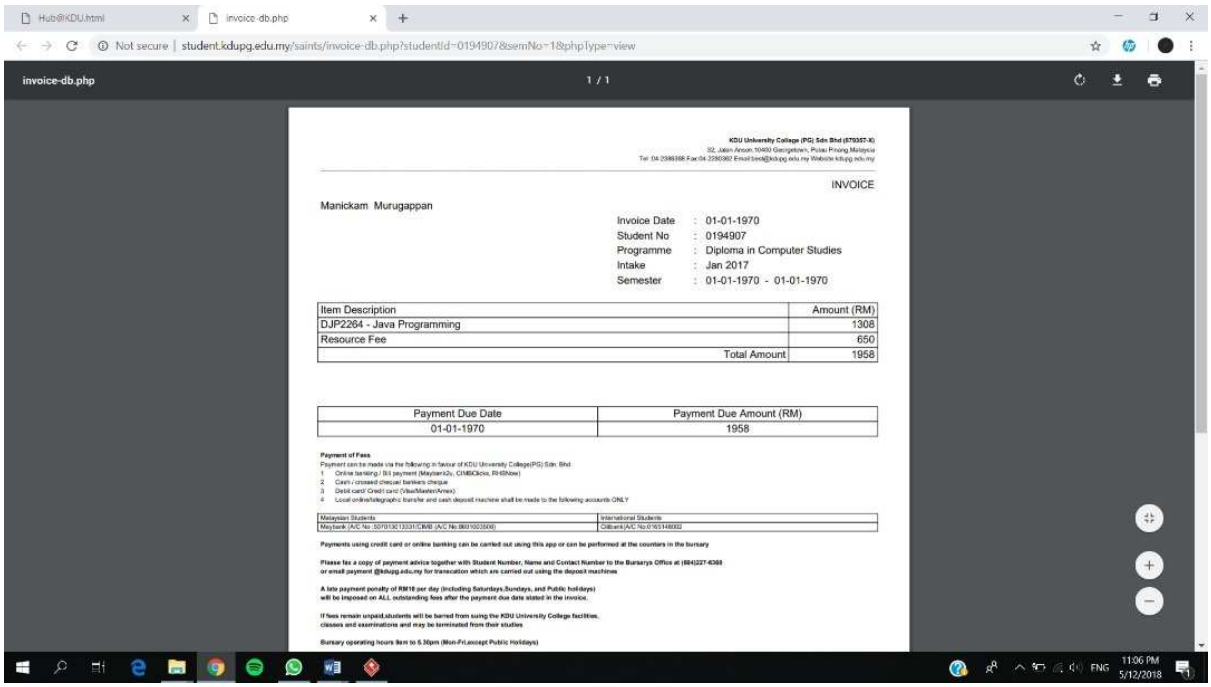


Diagram 7.3.2

Diagram 7.3.2 is an example on how the invoice would be displayed. In our case the default browser is google chrome. The display of the pdf varies depending on your default web browser.

7.4 DOWNLOADING INVOICES

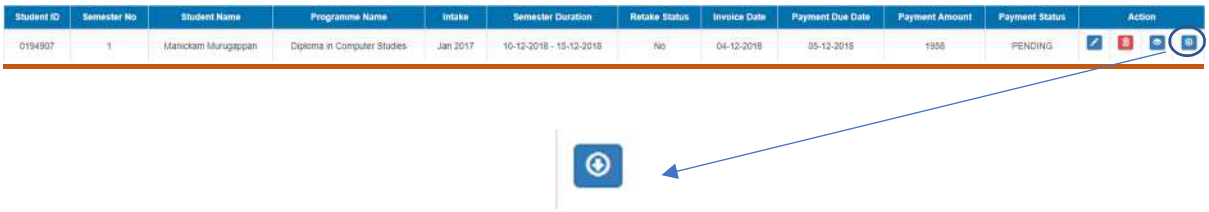


Diagram 7.4.1

To download the invoice of a particular student just click on the icon on the respective row which is shown in Diagram 7.4.1 to download the respective students invoice.

STUDENT USER MANUAL

TABLE OF CONTENT

1.0 LOG IN

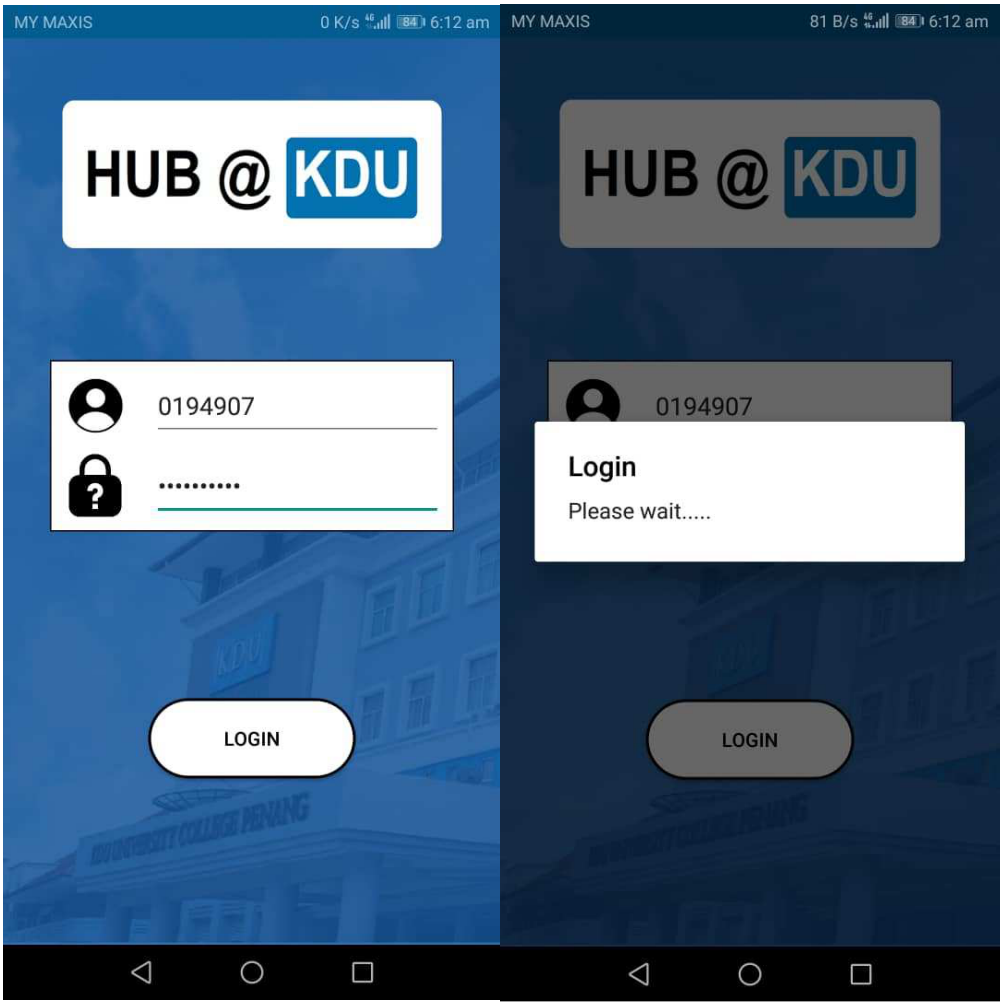


Diagram 1.1

According to the diagram above, you will be required to insert your respective user ID and password to be granted further access into the website. Click on the login button after inputting your credentials and there will be an alert box stating that you are logging in if the credentials are correct.

1.1 PERMISSION FOR STORAGE

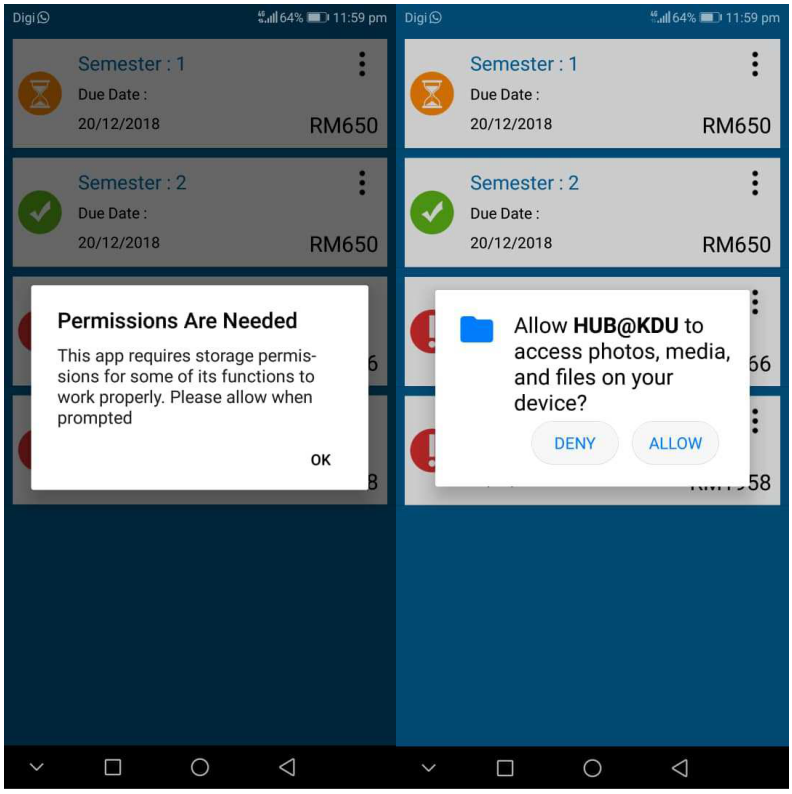


Diagram 1.1.1

When you have successfully logged in, you will be brought to this screen as shown above where there is a notice stating the reason why the permission is needed and followed by the permission dialog where the users have to press on **ALLOW** for the app to work efficiently. If you have pressed **DENY**, please refer to Section 2.4.1 (Page 18)

2.0 MY INVOICES

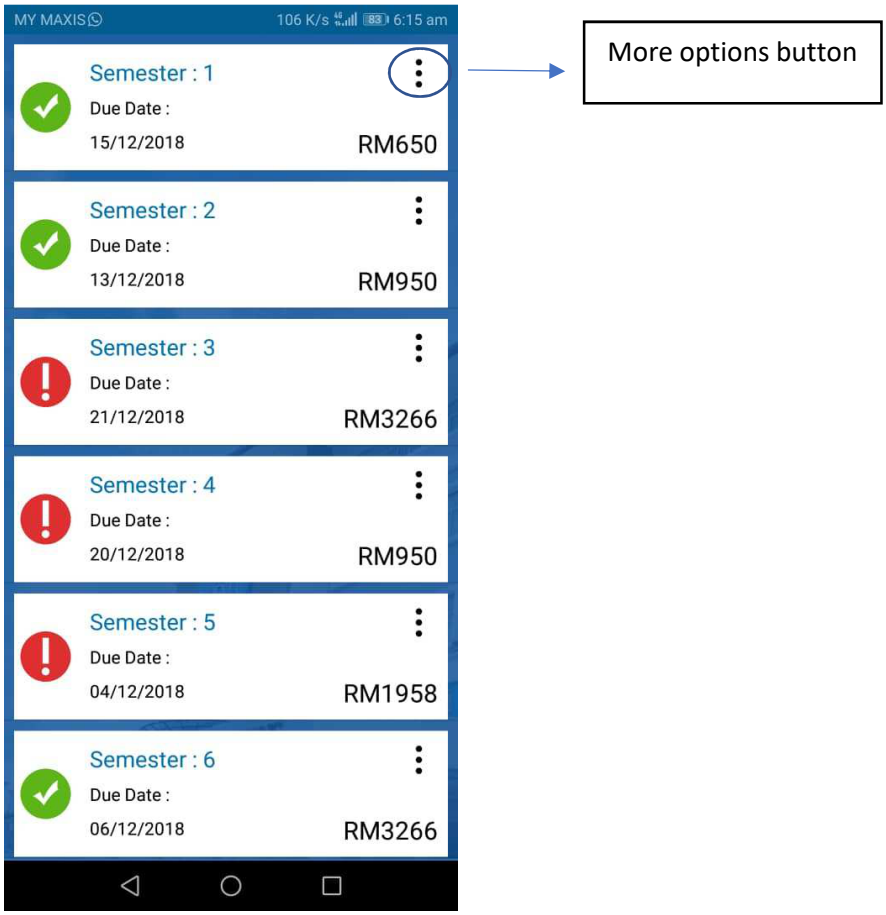


Diagram 2.0.1

Once you have log in successfully, you will be redirected to this page which is known as the My Invoices page or also known as the main page of the mobile application. In this page the user can view their respective invoice details such as the: -

a. Semester No
b. Due date for the payment
c. Amount Due

If the user presses on the more options button there will be a menu as shown as in Diagram 2.0.2 below.

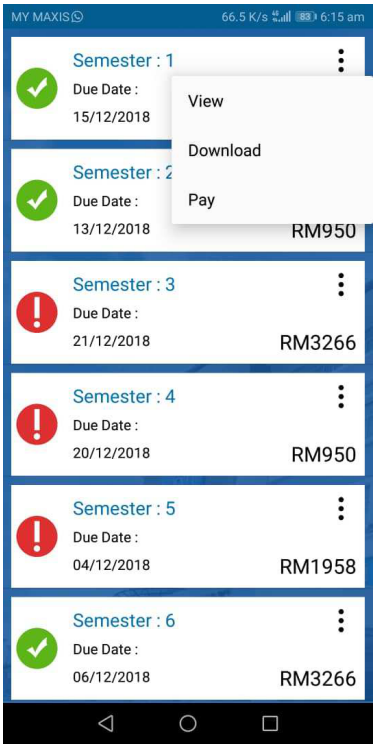


Diagram 2.0.2

For further details on the activity in the menu, if you press **View** under the menu please refer to Section 2.1 My Invoices (View) (Page 7). On the other hand, if you press **Download** please refer to Section 2.2 (Download) (Page 8). Finally, if you press **PAY** please refer to Section 2.3 (PAY) (Page 9).

2.1 MY INVOICES (VIEW)

Diagram 2.1.1

The respective invoice is displayed based on which semester container they pressed the view button on. The invoice is displayed in .pdf format and can also be zoomed in for better visibility.

2.2 MY INVOICES (DOWNLOAD)

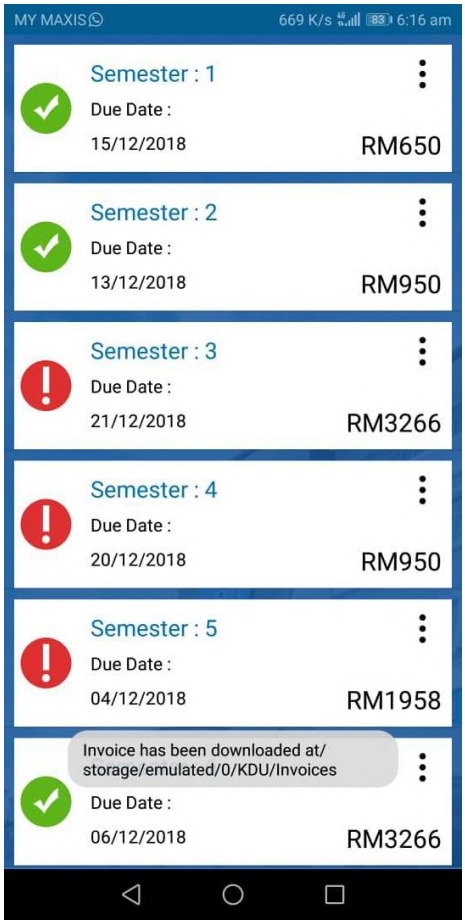


Diagram 2.2.1

In the screenshot above it can be seen that there is a toast message stating the directory of where the .pdf invoice file has been downloaded. So, if you would like to open the .pdf file, you can redirect yourself to the directory as shown above or go to your local storage and navigate to the KDU folder and under Invoices folder you can find your Invoices files downloaded.

2.3 MY INVOICES (PENDING/PAY)

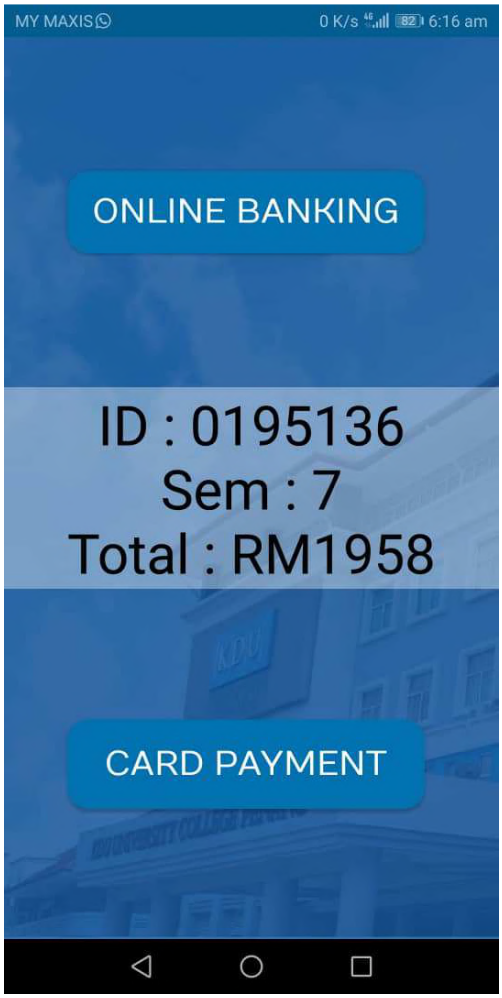


Diagram 2.3.1

When you click on the pay under the menu for semester's that the payment status is pending, you will be redirected to the diagram as shown above to choose your prefer method of payment. For further explanation on Online Banking Method, please refer to Section 2.3.1 (Page 10) and for further explanation on Card Payment Method, please refer to Section 2.3.2 (Page 12).

2.3.1 MY INVOICES (ONLINE BANKING / PAY)

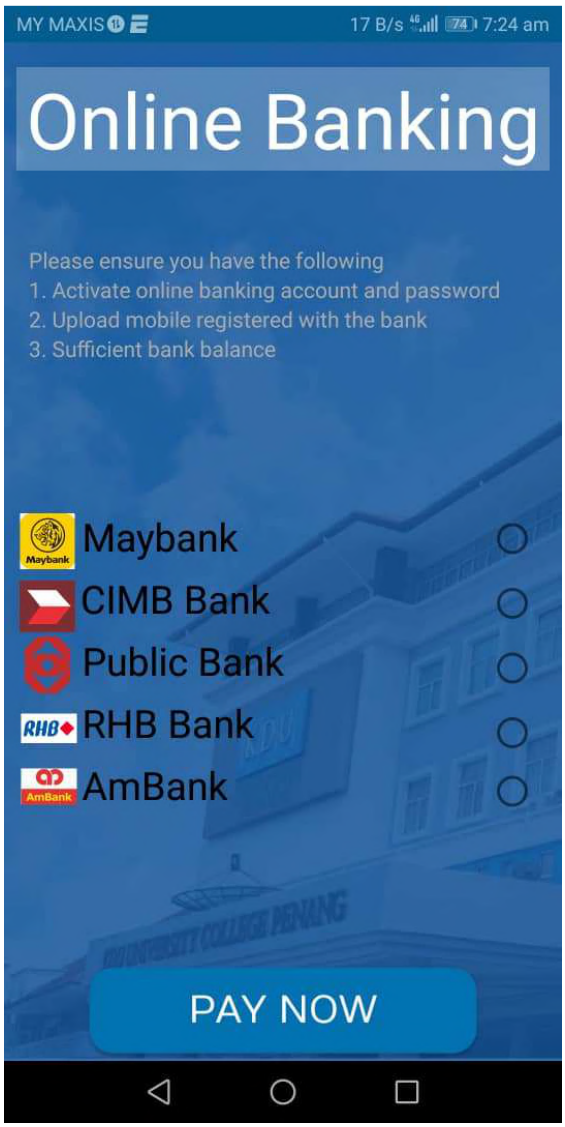


Diagram 2.3.1.1

If you clicked on the Online Banking method in Section 2.3, then you will be redirected to the page as shown in the diagram above.

Note: There will be no error checking because this is a fake payment gateway so all payment will be successful once you click pay now but the concept is there for further development in future versions of the application.

So once the payment is successful, they will be redirected to the page as shown in the diagram below which is Diagram 2.3.1.2

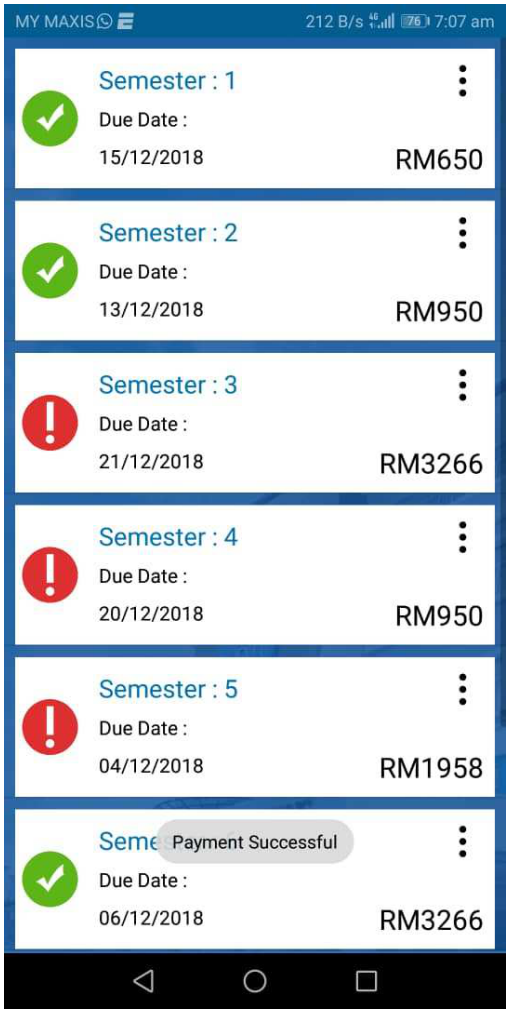


Diagram 2.3.1.2

So once the payment has been approved, you will be redirected to the page as shown above and there will be a toast message saying the payment is successful and if you realize the icon status in the respective container that you pressed **PAY**, the icon has changed to a paid icon.

2.3.2 MY INVOICES (CARD PAYMENT / PAY)

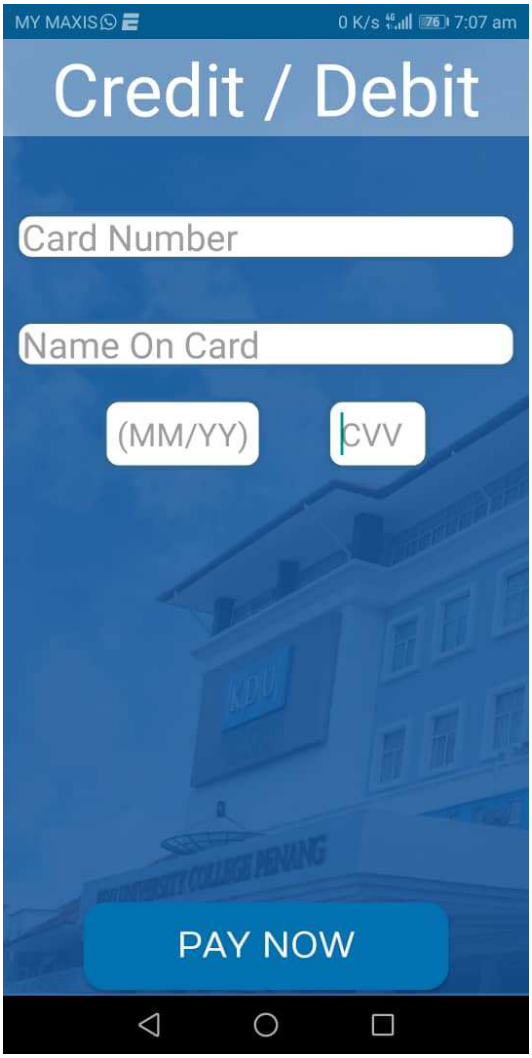


Diagram 2.3.2.1

If you clicked on the card payment method in Section 2.3 then you will be redirected to the page as shown in the diagram above.

Note: There will be no error checking because this is a fake payment gateway so all payment will be successful once you click pay now but the concept is there for further development.

So once the payment is successful, they will be redirected to the page as shown in the diagram below which is Diagram 2.3.2.2

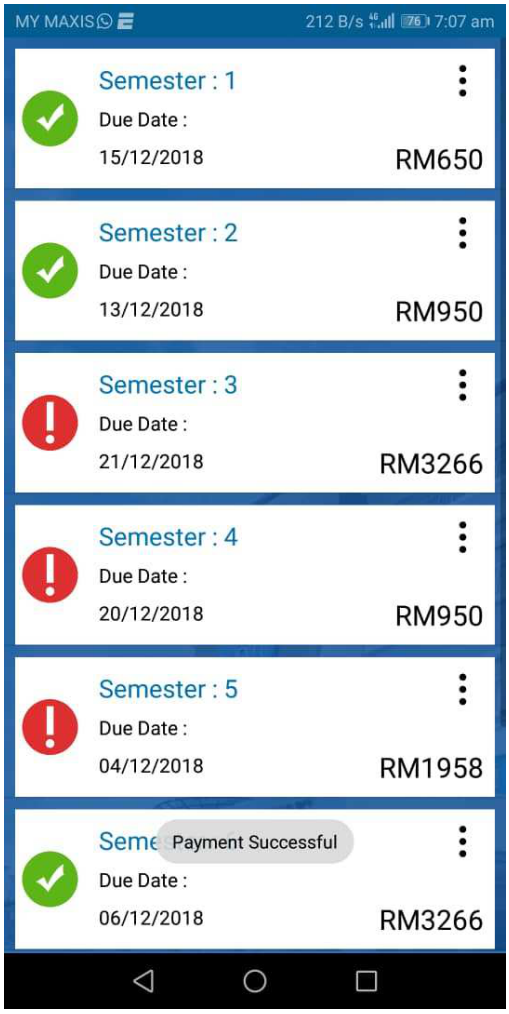


Diagram 2.3.2.2

So once the payment has been approved, you will be redirected to the page as shown above and there will be a toast message saying the payment is successful and if you realize the icon status in the respective container that you pressed **PAY**, the icon has changed to a paid icon.

2.3.3 MY INVOICES (PAID/ PAY)

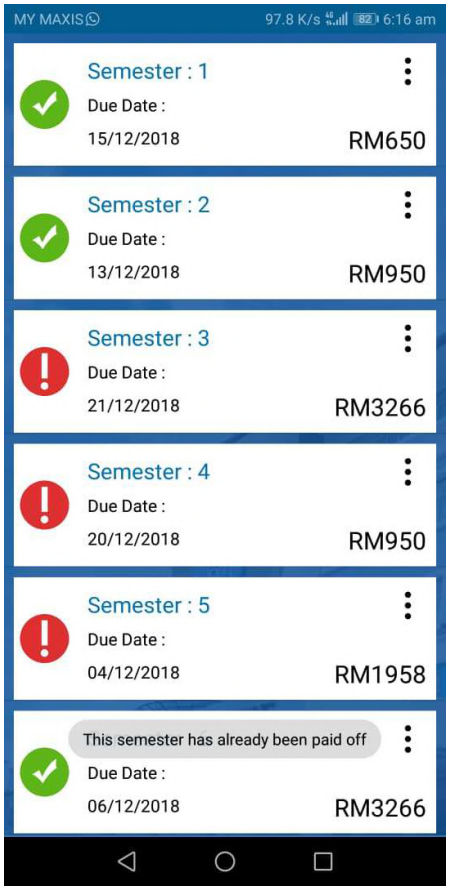


Diagram 2.3.3.1

When you click on the pay under the menu for semester's that the payment status is paid, there will be a toast message saying that the semester has already been paid off meaning that there already has been a successful transaction for the invoice and there is no amount due for payment.

2.3.4 MY INVOICES (LATE/ PAY)

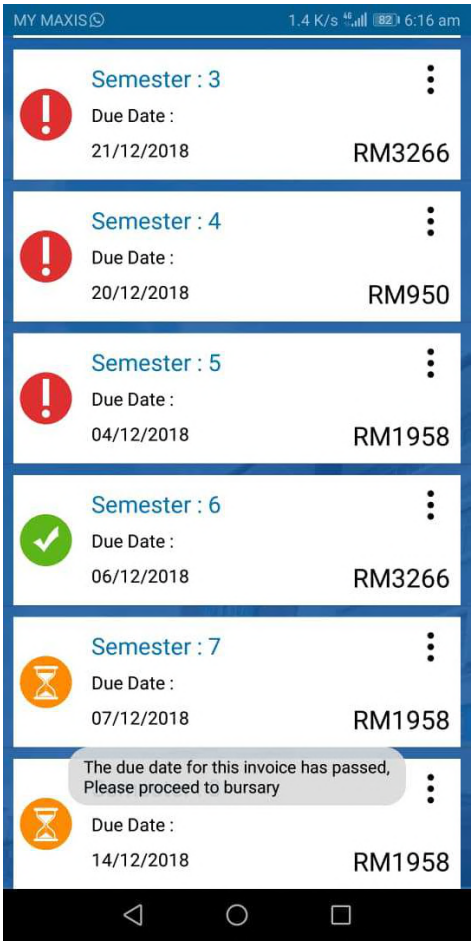


Diagram 2.3.4.1

When you click on the pay under the menu for semester's that the payment status is late, there will be a toast message saying that the due date for the invoice has passed and to make the transaction you have to proceed to the bursary located at the college. This is to adhere to the policy practiced by KDU where late payments will incur fines and extra charges.

2.3.5 MY INVOICES (UPDATE)

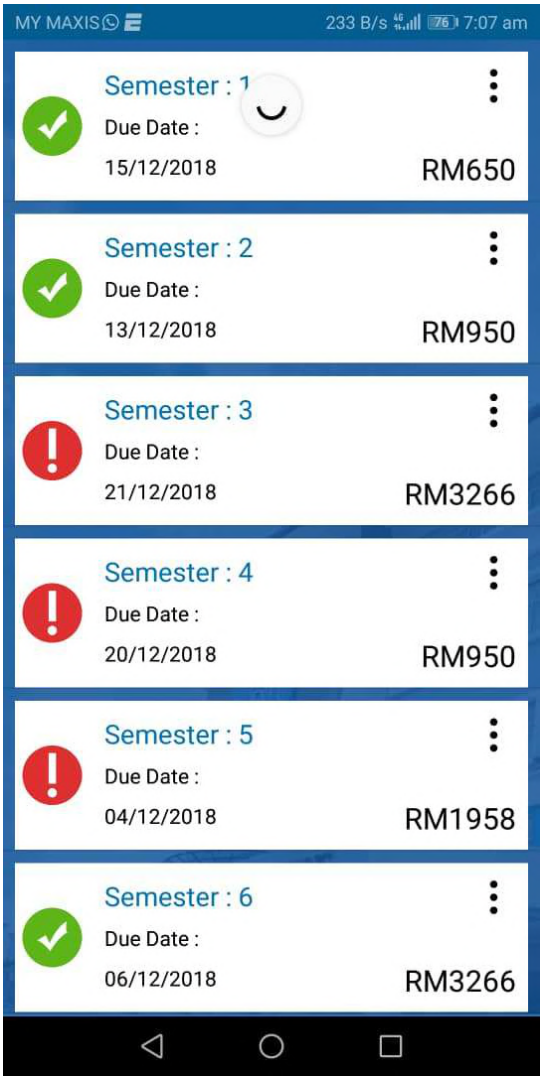


Diagram 2.3.5.1

When you would like to refresh or update the list of invoices in your list to see newly deployed invoices, you can simply swipe down from the top with your index finger. If there are any new invoice deployed, the invoice list will be updated.

2.4 SIDE NAVIGATION BAR

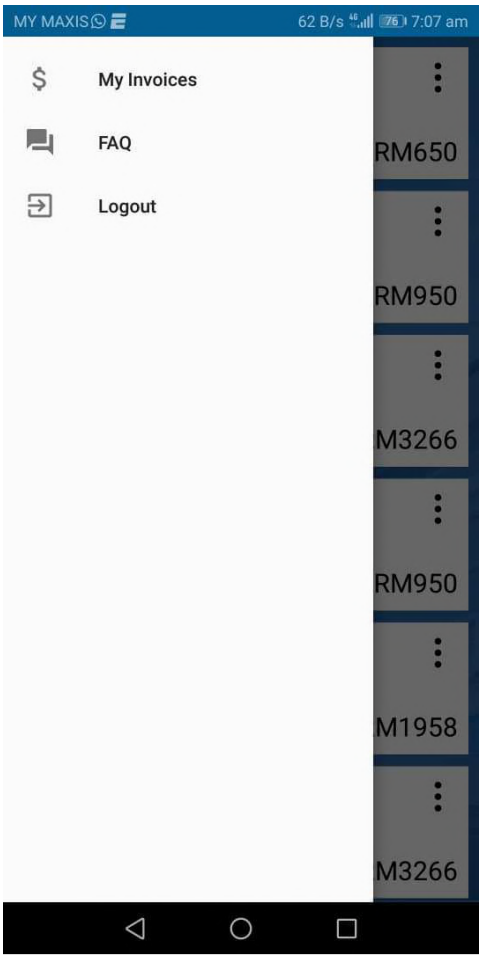


Diagram 2.4.1

To view the navigation bar just slide your finger from left to right from the left side of screen to slide in the side navigation bar. The navigation bar has 3 options, which are My invoices, FAQ and Logout. If you press on **My Invoices,** you will be redirected to Diagram 2.0.1 under Section 2.0 My Invoices. On the other hand, for FAQ explanation please refer to Diagram 2.4.1.1 under Section 2.4.1. Finally, for logout you will be logged out from the app and for further explanation, please refer to Section 2.4.2.

2.4.1 SIDE NAVIGATION BAR (FAQ)

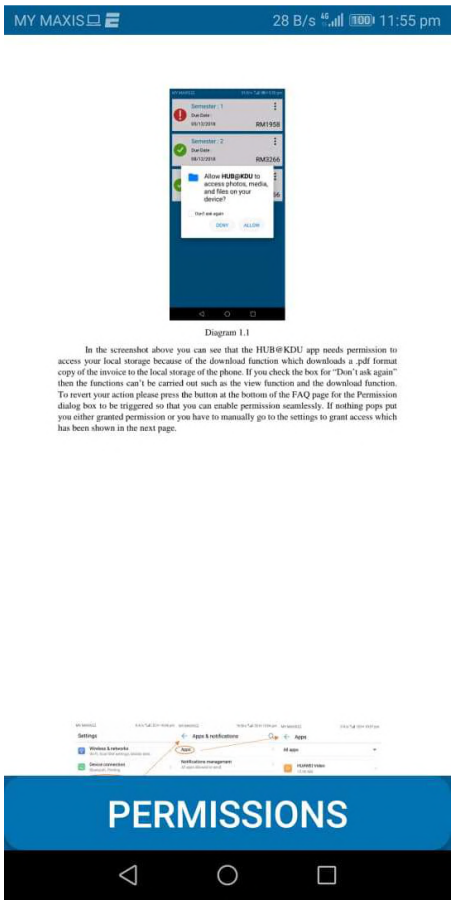


Diagram 2.4.1.1

Once you have pressed the FAQ under the side navigation bar you will be redirected to this page where there is a detail explanation on how to grant permission for the function of accessing the storage. The button "PERMISSIONS" can also trigger the permission dialog when pressed.

2.4.2 SIDE NAVIGATION BAR (LOGOUT)

When you have clicked the button logout, you will be logged out from your account and be redirected to the log in page again.