



# **ASHESI UNIVERSITY COLLEGE**

## **SCHOLARSHIP FUND MANAGEMENT SYSTEM**

### **APPLIED PROJECT**

B.Sc. Management Information Systems (MIS)

**Ayishetu Seidu**

**2017**

**ASHESI UNIVERSITY COLLEGE**

**Scholarship Fund Management System**

**APPLIED PROJECT**

Applied Project submitted to the Department of Computer Science,  
Ashesi University College in partial fulfilment of the requirements for the  
award of  
Management Information Systems (MIS)

**Ayishetu Seidu**  
**April 2017**

## Declaration

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

Candidate's Signature.....

Candidate's Name.....

Date.....

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

Supervisor's Signature.....

Supervisor's Name.....

Date.....

## Acknowledgements

I would like to give thanks to the almighty Allah for the gift of life he gave me to be able to undertake this project. My profound appreciation also goes to my invaluable supervisor, Mr Aelaf Dafla, whose guidance and supervision saw me through this project. I would also like to thank Mma Asana (my mother), Baba Seidu (my father), Sister Amina, Issah, and all my family members for their love and support throughout my stay in Ashesi. My gratitude also goes to all my friends, especially, Moses Kasanga, Alvin Ofori (who helped me anytime I got stuck with coding and my supervisor is not readily available), Kingsley Agyekum, David Okyere, Natasha Mabuza, Freda Manu, and all those whose support and encouragement made this project a success. My Final gratitude goes to the MasterCard Foundation, without whose support I could not have made it this far.

May Allah richly bless you all

## Abstract

This project seeks to improve an existing web based system used by a Non-Governmental Organisation in Ghana to manage their scholarship application process. The NGO, which sponsors High School students in Ghana, currently has a web based application that helps them process their scholarship application and awarding process. The existing application keeps record on applicants, generate points for applicants which are used as the basis for awarding the scholarships, and generate scholarship offer letters for these awardees. Given the current focus of the existing system, this project seeks to widen the scope of the system by adding functionalities to track payments. Therefore, instead of just being a **Scholarship Application System**, the system will be a **Scholarship Management System** (or a Scholarship Fund Management System). At the end of this project, the NGO should be able to use less time in processing payments, get a more transparent payment system, generate reports from the system and reduce costs of operation.

# Contents

Declaration .....	I
Acknowledgements .....	II
Abstract .....	III
List of Figures .....	VI
List of Tables.....	VII
Chapter 1: Introduction .....	1
1.1 Background .....	1
1.2 Motivation. ....	2
1.3 Related work and Research .....	3
1.3 Related Work .....	3
1.4 Objectives .....	4
1.5: Outline of project .....	4
Chapter 2: Requirement Analysis .....	5
2.1: Definition of Users .....	5
2.1.1 Programme Officer/Manager .....	5
2.1.2: Learning Advisor .....	6
2.1.3: The Accountant.....	7
2.1.4: IT Personnel/Administrator .....	8
2.2: System Requirements .....	9
2.2.1: Functional Requirements .....	9
2.2.2: Non-Functional Requirements.....	10
Chapter 3: Architecture and Design .....	11
3.1: Overview of the Mode View Controller (MVC) Architecture.....	11
3.1.1: The Model.....	11
3.1.2: The View .....	14
3.1.2: The Controller.....	16
Chapter 4: Implementation.....	18
4.1: Overview of implementation.....	18
4.1.1: Libraries and Frameworks .....	18
4.2 Email Implementation .....	19
4.2.1: Scope of Work .....	19
4.2.2: Overview.....	19
4.3: PDF Report.....	20
4.3.1: Scope of work .....	20

4.3.2: Implementation Overview. ....	21
4.4: Viewing Criteria for Generating Points.....	22
4.4.1: Scope of Work .....	22
4.4.2: Overview.....	22
4.5: Verifying Additions to Schools.....	24
4.5.1: Scope of Work .....	24
4.5.2: Overview.....	24
4.6: Disbursing Payment Requests. ....	25
4.6.1: Scope of Work .....	25
4.6.2: Overview of Implementation.....	26
Chapter 5: Testing .....	28
5.1: Unit Testing.....	28
5.2: Component Testing .....	30
5.3: Compatibility Testing.....	30
Chapter 6: Conclusion and Recommendations .....	31
6.1: Conclusion.....	31
6.2: Recommendations .....	31
References .....	33
Appendix .....	34

## List of Figures

Figure 2.1: Use case diagram for the programmes officer/manager .....	6
Figure 2.2: Use case diagram for the learning advisor.....	7
Figure 2.3: Use case diagram for the accountant .....	8
Figure 2.4: Use case diagram for the IT personnel or administrator .....	9
Figure 3.1: Entity-Relationship (ER) diagram for applicant details .....	12
Figure 3.2: Entity-Relationship (ER) diagram for payments .....	13
Figure 3.3: View of the home page .....	15
Figure 3.4: Interface for adding a new applicant .....	15
Figure 3.5: Interface for the list of applicants .....	16
Figure 3.6: High level architecture of the system .....	17
Figure 4.1: Email implementation result.....	20
Figure 4.2: PDF implementation result.....	21
Figure 4.3: Viewing point criteria implementation result.....	23
Figure 4.5: Results for verifying addition to school .....	25
Figure 4.6: Disbursing payment requests implementation result.....	27
Figure 4.7: Disbursing Payment Requests Implementation Result 1 .....	27
Figure A.1 Implementation of PDF report generation .....	34
Figure A. 2: Code for sending mail using PHP mail() function and object obscuring functions.....	34
Figure A.3 Login page for accounts.....	35
Figure A.4 All payment requests .....	35
Figure A.8 Result of unit test for verifying schools.....	41
Figure A.9 Results of unit test for getting all payment requests.....	41
Figure A.10 Unit test result for getting scholarship payment for a given payment request.....	41



## List of Tables

Table 5.1: Unit test for send email module .....	29
Table A.5: Unit test for view point criteria implementation .....	36
Table A.6: Unit test for PDF report .....	39
Table A.7 Unit Test for disbursing payments .....	40
Table A.11: Other functions under payments tested .....	42

## Chapter 1: Introduction

A Non-Governmental Organization (NGO), located in Ghana, provides scholarship for intelligent high school students from economically disadvantaged backgrounds. They operate from Greater Accra Region of Ghana with their programme units in four regions, namely, The Volta, Upper East, Eastern, and Central regions of Ghana. Like many organisations, the NGO has moved from manual processes and information system to a web based system. This existing system allows them to enter information on new applicants, award scholarships and track students' performance. This project therefore seeks to improve the system by adding functionalities to make it able to facilitate and track scholarship payments, and provide reporting functionalities for the scholarship application process.

### 1.1 Background

The pace of globalisation in recent times requires businesses to come up with cost effective operation strategies towards developing new products or services that meet acceptable standards, and appeal to target customers to deal with competition. To achieve this, businesses, whether for profit or not for profit, must constantly consider and improve the way they do business and change their information systems to buttress evolving processes. Business process automation is one way for businesses to effectively structure information systems and deal with the current trends. Its benefits are countless, ranging from cost saving to efficiency. A paper in the *Proceedings of the 32nd Annual ACM SIGUCCS Conference on User Services*, discussed a new registration system used in London School of Economics and its benefits compared to an old manual system. The new automated system reduced the number of days used for registration from twelve (12) days to seven (7) days. Again, the need for fifteen (15) IT support staff to facilitate the manual registration

was eliminated in the automated system, saving the school an amount of \$15270.18 (Forbes-Pitt, 2004).

Technologies like office automation systems, document management systems, workflow systems, web technology, and robotic automation exist for businesses to take advantage of, in automating their processes. Web Technology is lately gaining substantial attention among business process experts because it is a flexible and low-cost solutions to distributed collaborative work. With an available network, web technology allows computers not only to share resources but also enables people to interact with processes and users to achieve business goals (Takashi & Liang, 1997). Companies and institutions are taking advantage of this technology to achieve efficiency and cut down costs. Ashesi University, for example, uses an automated web based system for students' course registration and when registrations are open, in a less amount of time, they get more students to register than it would have been if they used manual system. Non-Governmental Organizations and non-profit-organizations are not an exception of these for-profit companies. They many not seek to maximize their profit, but they need to be efficient to maximize the use of the limited fund they have.

## 1.2 Motivation.

In view of the increasing number of applicants over the years, manually collaborating and managing the data from different schools in the ten regions of Ghana can be a tedious work for the NGO. Trivial jobs will have to be done by many people leading to a huge part of their funds for scholarships being used to pay these employees. Again, working manually takes time which the NGO could have used focus on more important aspects of the organisation

Again, when dealing with financial and academic records, consistency and accuracy are key to ensuring trust among stakeholders in the organisation. Their Donors need to be

sure that, financial statements and records are accurate and that monies they have donated are being used for the intended purposes. Beneficiaries also need to be sure that, their performances in their respective schools are exactly what is captured by the NGO. However, errors and inconsistencies are hard to avoid when processes are manual.

Ghana Passports office pose a clear example of errors that happen when business processes and Information Systems are made manual. People go to the Ghana passports office and spend hours just to submit their passport application form and they are given a day to come for their passport. The day arrives and when they go, they tell them they cannot find their documents so they should start the whole process again (Painstil, 2015). The money and time wasted in buying the form and waiting to submit the form respectively could have been invested in something profitable.

One key feature of doing business is governance and monitoring which is difficult if business processes are manual. From these motivations, this project seeks to help Plan Ghana save time and money, have a more accountable system and be effective in their processes.

### 1.3 Related work and Research

#### 1.3 Related Work

Some companies have come up with software that allows for the management of any kinds of fund. These applications are commercially off the shelf and are customizable for companies that buy them. Some of these existing applications are listed below;

##### *1.3.2.1: FluidReview*

This is a fund management system developed by Survey Monkey, which allows applicants to fill forms and submit every necessary document via a centralised portal. It allows for internal coordination of work and automates selection of qualified applicants. it

comes with features like online application forms, applicant matching, email and phone support, reporting, recommendation letter generation, and so on (Capterra, n.d.).

#### *1.3.2.2: ICARIS Grant Management Software*

This is also another grant management system, developed by ICRARIS LTD, which uses SQL backend data structure for automating gift aid, grants processing, payments, and beneficiary management. It comes with features for reporting and dashboards (ICARIS LTD, n.d.).

### **1.4 Objectives**

By the end of this project;

- The NGO should be able to save time and costs in their processes which can be invested productively elsewhere.
- They should also be able to minimise errors which will promote trust among its major stakeholders; their donors and beneficiaries.
- Monitoring, on the part of the management of NGO, should be easier to minimise the tendency by their personnel to indulge in bad business practices like embezzlement of funds and accepting applicants who do not qualify to be awarded
- The system should be able to capture payments to scholarship students
- Current users should find the application and its interface usable.

### **1.5: Outline of project**

This paper has 6 chapters. Chapter 1, gave the background, related work, motivation and objectives for the project. Chapter 2 outlines requirements analysis and chapter 3 gives the architecture and design of the system. Chapter 4 describes the implementation; methods, technologies and frameworks used in building the system. The remaining chapters, 5 and 6 discuss Testing, Conclusions and Recommendation respectively.

## Chapter 2: Requirement Analysis

To come up with requirements for this version of the system, the initial developer of the system was interviewed to gain more insights about how the system works. The interview threw light on four main users of the system. The various users, their roles, scenarios, and use cases are elaborated below.

### 2.1: Definition of Users

#### 2.1.1 Programme Officer/Manager

The programme officer is one of the key persons in the operation of the organisation, regarding scholarships. He/she oversees a specific programme area, so he/she is more like a branch manager. When applicants apply for scholarships using hard copy forms, the programme officer enters the information on these applicants into the system, awards and confirms applicants' scholarships. When applicants are awarded scholarships, the programmes officer ensures that manages these sponsor students. He/she also is responsible for making payment requests on behalf of the scholars. The use case below summarises how the programme officer interacts with the system

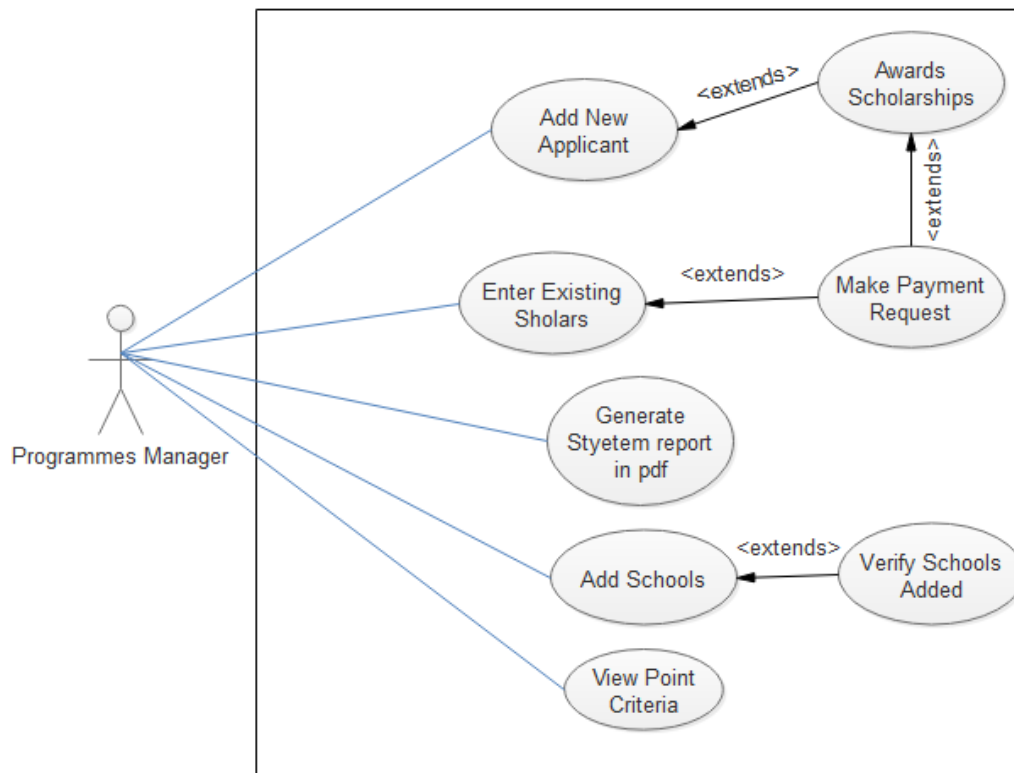
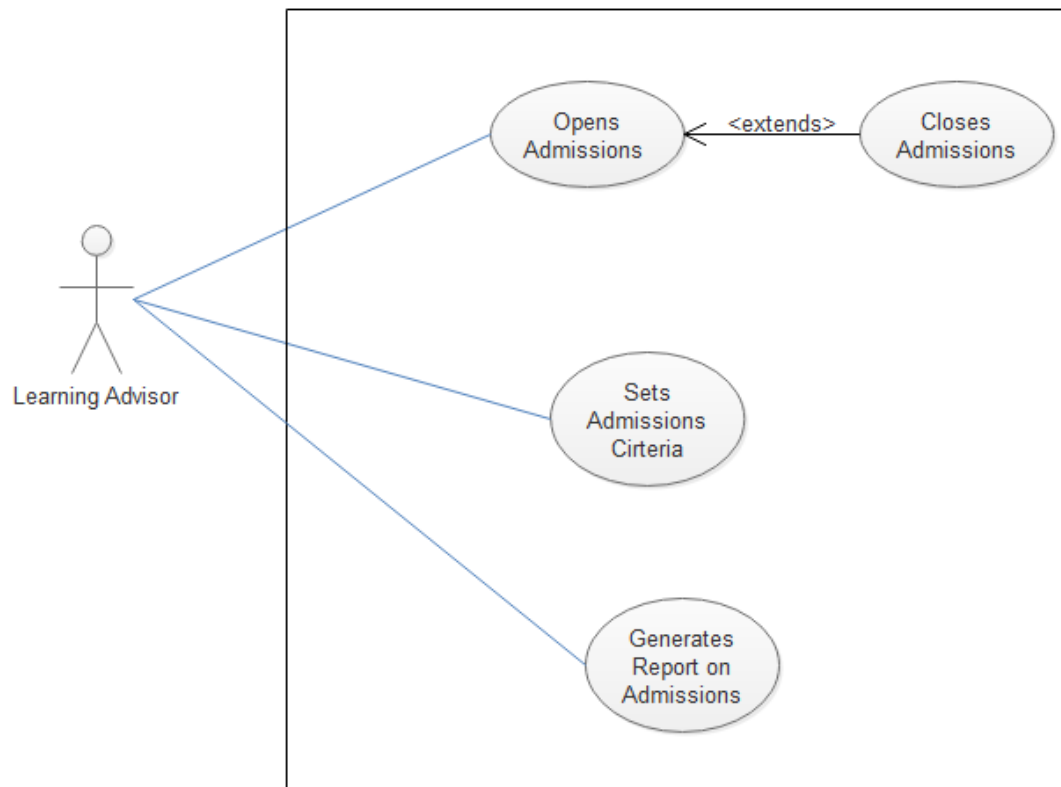


Figure 2.1: Use case diagram for the programmes officer/manager

#### 2.1.2: Learning Advisor

The learning advisor's main function is admissions. He/she is responsible for opening and closing new admissions. The learning advisor is also responsible for setting the criteria for awarding scholarships in each admission year. The use case diagram below shows the learning advisor's interaction with the system.

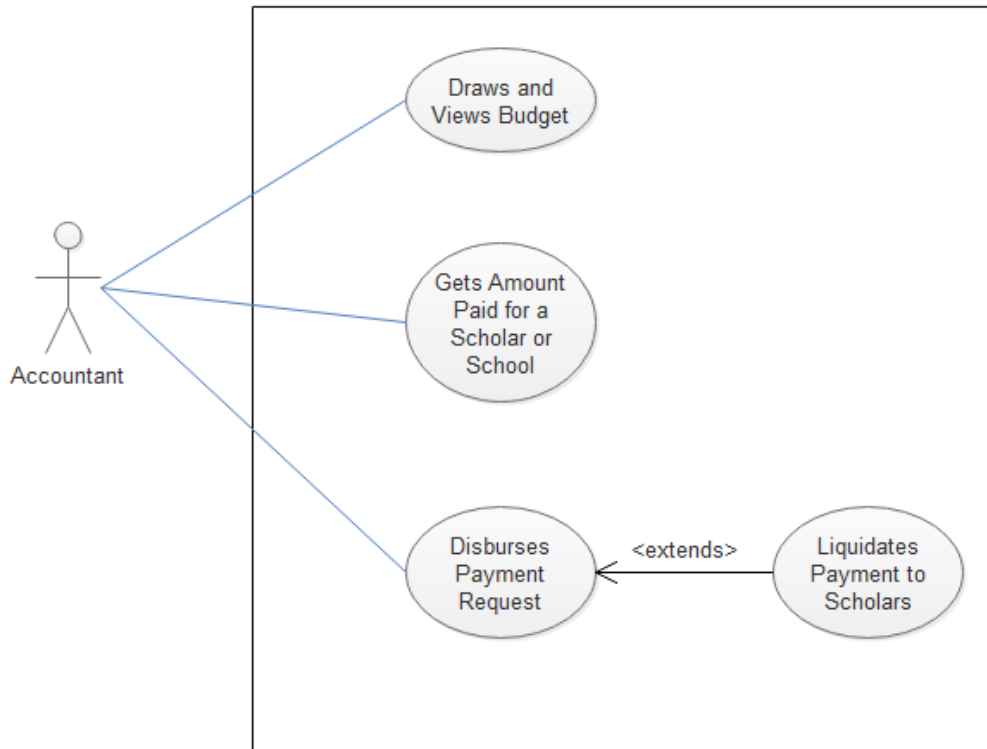


*Figure 2.2: Use case diagram for the learning advisor*

#### 2.1.3: The Accountant

The accountant handles all the monetary aspects of the organisation. He/she is responsible for disbursing payments requested by the programme officer after approving these payments. The accountant again prepares budgets and financial reports periodically for the NGO. The use case diagram below shows his/her typical interactions with the system.

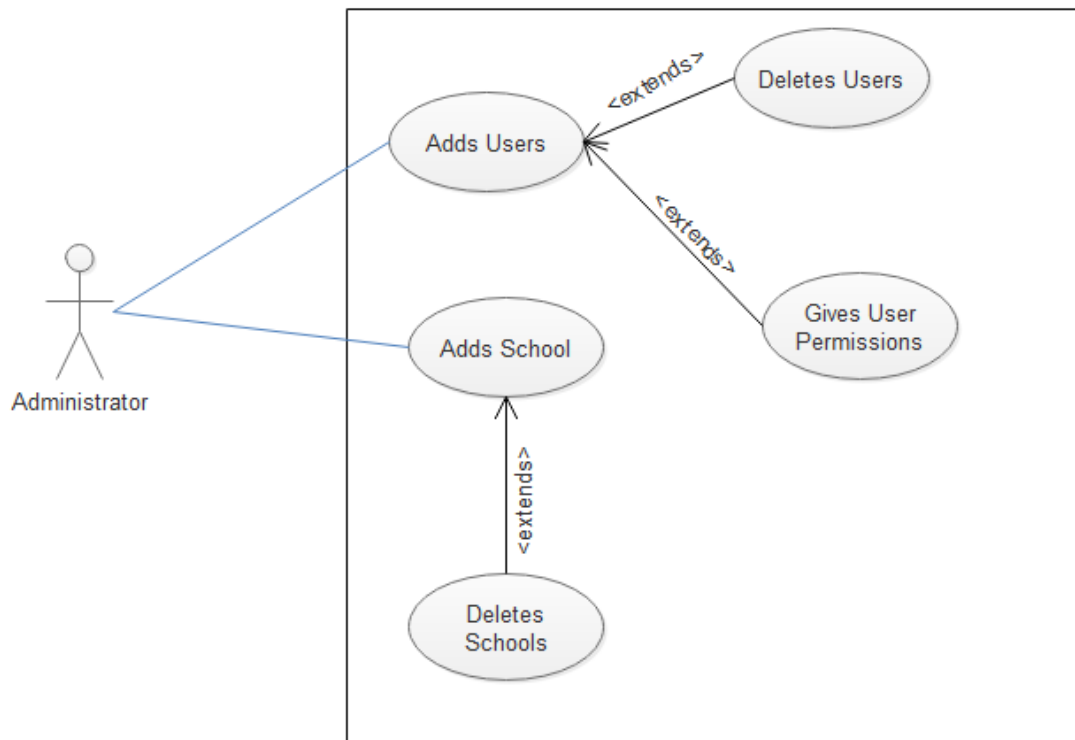




*Figure 2.3: Use case diagram for the accountant*

#### 2.1.4: IT Personnel/Administrator

This is the overall user of the system. This person can do almost everything on the system. He/she wants to be able to add users and delete users. He/she also want to be able to give users permission and withdraw user permissions. He/she also want to be able to manage support data like high schools and basic schools by adding and deleting schools. The use case Diagram below summarises the administrator's interactions with the system



*Figure 2.4: Use case diagram for the it personnel or administrator*

## 2.2: System Requirements

From the above analysis, the requirements below were generated to be added to the existing system.

### 2.2.1: Functional Requirements

- ⇒ The programmes manager should be able to see how an applicant got the points which is used as a basis for awarding the scholarship.
- ⇒ The country director should be notified, by mail, the summary of the entries made each week. This is to ensure that, offices in localities where the manager is not present to supervise the usage of the system feed the system with information. This will make monitoring and decision making easy for the programme's manager.
- ⇒ The Manager should be able to generate a pdf report on the summary of activities on the system.

- ⇒ When a data entry officer is adding a school to the list of schools, he should be able to verify that, that name does not exist in the database to avoid duplication of school names
- ⇒ The accountant/accounts department should be able to:
  - Login
  - Show budget for the year
  - Check payment requests and change and disburse them
  - See how much is paid for a student (given a scholarship package ID)
  - How much is paid this financial year for a school, and how much is left

#### 2.2.2: Non-Functional Requirements

- ⇒ The pages should be made faster and more interactive by reducing the number of page switches when using the system
- ⇒ The user interface should be appealing to encourage users to use the system.
- ⇒ Requests should be made asynchronous to make sure the system catches up with existing technology since synchronous requests are being wiped out from the new technologies

After gathering these requirements, a meeting was set up again with the initial developer to validate the requirements. This was to ensure that, the requirements are exactly what they should be, and to reconcile any disagreements.

## Chapter 3: Architecture and Design

### 3.1: Overview of the Mode View Controller (MVC) Architecture

This project is building on an existing system which uses the **Mode View Controller (MVC)** architecture. This system architecture divides a system into three components, namely, the model, the view, and the controller. It is one of the mostly used architecture in web based applications and it makes building of complex applications easy.

The model is the part of the application that implements the logic for the application's data management. This can be retrieving, updating, and storing data. The View represents that part of the application represents the user interface with which the user interacts. This includes all text boxes, buttons, dropdown menus, etc. which facilitates easy use. The controller serves as the interface between the model and the view. It listens to user actions and events and invoke changes to the model and the view.

#### 3.1.1: The Model

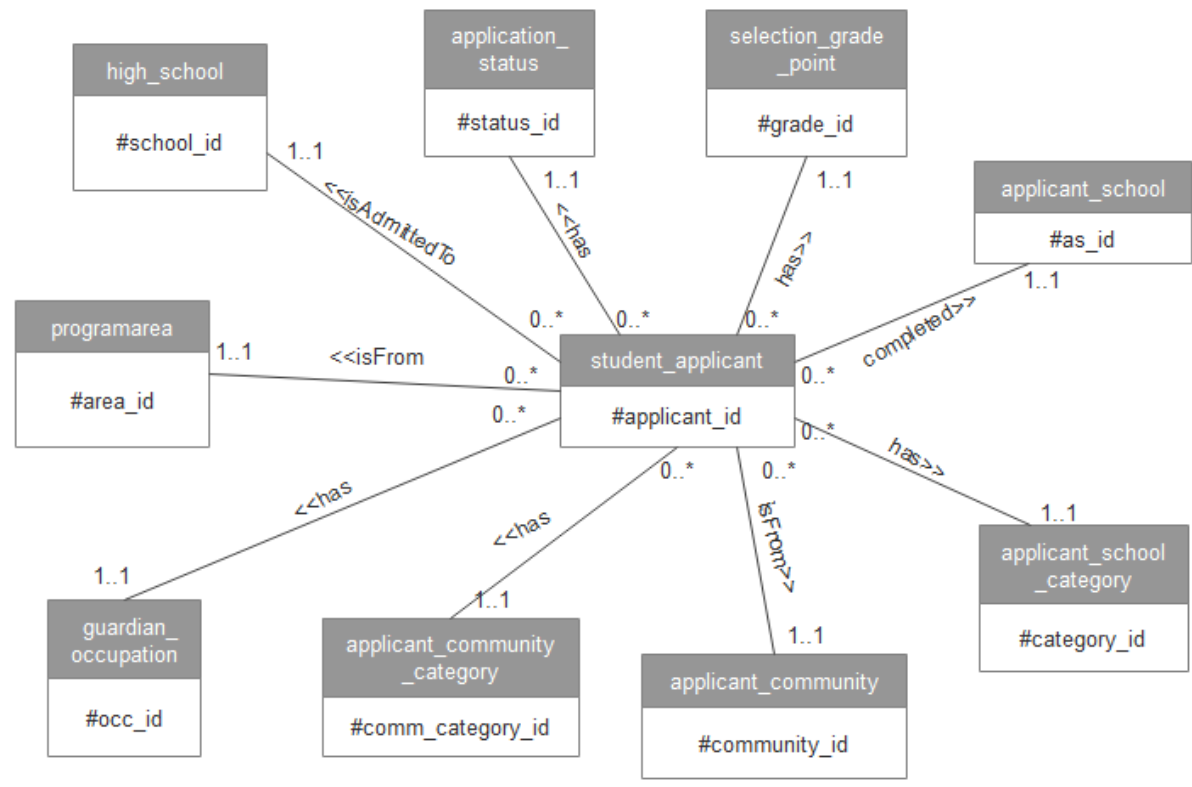
The model of this project is built using MySQL because it is free and mostly used in web application developments. It allows you to create relational database and tables, insert values, update, and delete these values and databases.

The model of the existing system has tables such as

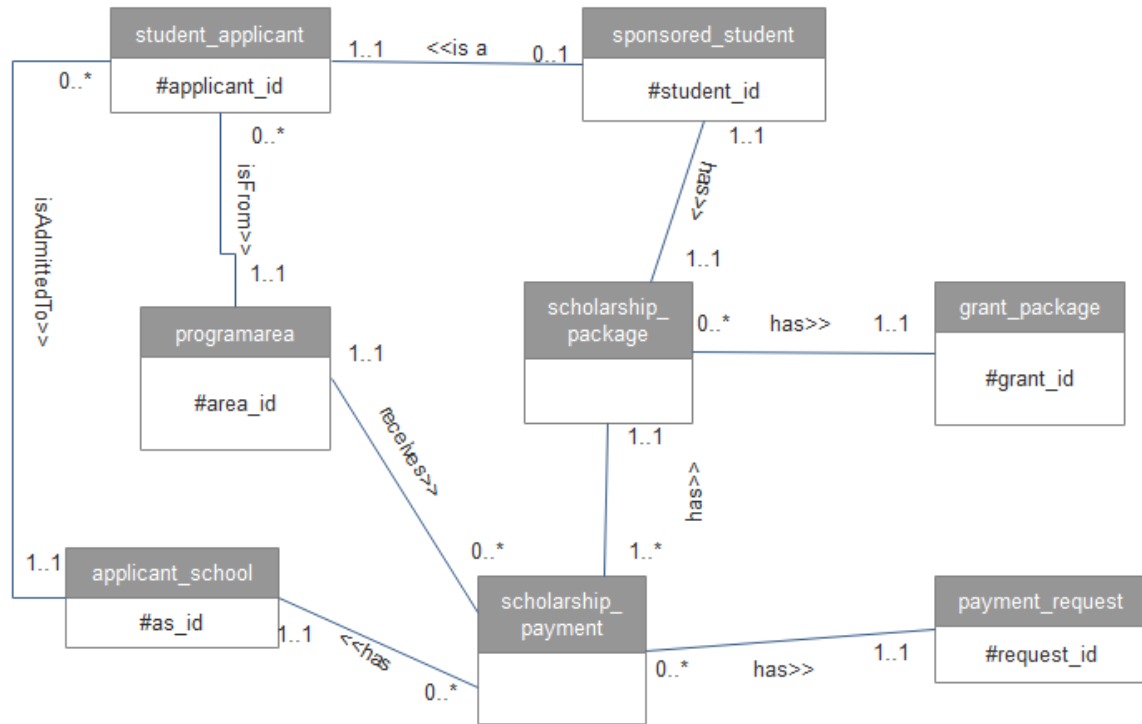
- student\_applicant- containing the data on applicants
- sponsored\_students – containing data on awardees
- users – containing data on employees that use the system
- userlevels – contains the permissions of the users
- scholarship\_package – contains data on each scholar and the details of the packages awarded to them
- grant\_packages – contains the various grants and their details

- scholarship\_payment – contains the details of a payment request on behalf of a scholar

The diagrams below show data is arranged in the model



**Figure 3.1: Entity-Relationship (ER) diagram for applicant details**



**Figure 3.2: Entity-Relationship (ER) diagram for payments**

Although the above database/model serves its purpose, the design is not the best. There are redundant relationships and tables which will make querying the database inefficient due to the many joins that will need to be done, and take up server space. An applicant, for example has information about his/her high school and programme area, but scholarship payments table, which refers to the applicant again makes reference to the school and programme area tables.

On redundant tables, status of an applicant could have been made an enumerator instead of creating a table for it. The information in the scholarship package table could have also been appended to the sponsored students' table.

The applicant's information is not also well normalised. For example, the school and community category tables could have been linked to the school and community tables respectively, instead of the direct link with the applicant table.

### 3.1.2: The View

This is built using Hypertext Mark-up Language (HTML) and Cascading Style Sheet (CSS).

The entry officer has an interface, represented by applicants.php, where he/she can enter information of new applicants using HTML forms. The applicants.js helps with validation of inputs made. Programme officers also have an interface (applicantlist.php) that helps them review applications and award scholarships. The system administrator also has an interface where he/she can add new users or delete existing users.

This version of the application will provide an interface for managing payments. A programme manager can make and liquidated payments in each active year for the scholars. The accountant will also have an interface where he/she can review payment requests and disburse them. There will also be an interface for the programme manager to generate report on the activities on the system.

Below are pictures of the view of some pages in the existing application.

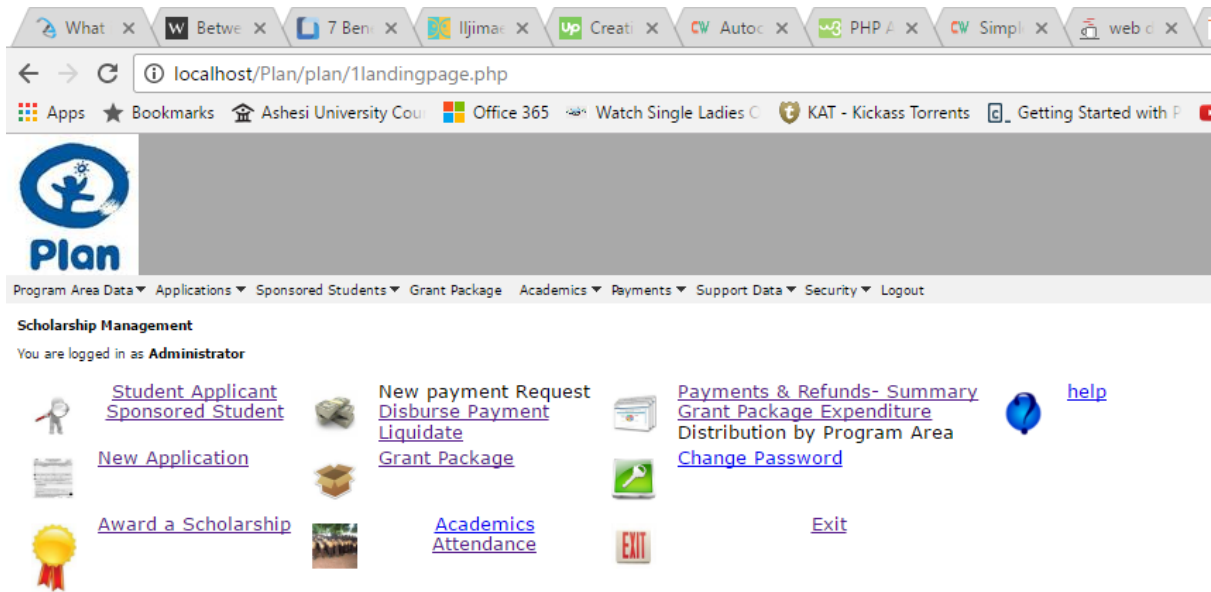



Figure 3.3: View of the home page

Figure 3.4: Interface for adding a new applicant





Program Area Data ▾ Applications ▾ Sponsored Students ▾ Grant Package ▾ Academics ▾ Payments ▾ Support Data ▾ Security ▾ Logout

Scholarship Management

3 records found

Program Area/Unit < --include all-- ▾ | Application Year : 2013 --all-- ▾ | scholarship ▾ new ▾ | 
search | export | Grants select grant ▾ | award | cancel | new applicant | refresh app points

[prev](#)
[next](#)

	PU	Year	Community	Name	Gender	Birth Date	Telephone	School Applied To	Status	Points	Grant	
<input type="checkbox"/>		2013	Aboano	Seidu, Ayishetu	F	17/05/1996	0247666425, 0507157798	A.M.E ZION GIRLS	New App	78	NO AWARD	<a href="#">detail</a>
<input type="checkbox"/>	Central	2011	Suromanya	Someone2In, Someone2In	M	01/03/2017	'024-9684639', "	ODORGONNO SHS	New App	0	GAD GRA 0095	<a href="#">detail</a>
<input type="checkbox"/>	Eastern	2011	Anyamoni	SomeoneIn, Someone	F	01/06/2001	0254171491, 022222	Wa SHS	New App	41	GAD GRA 0095	<a href="#">detail</a>

*Figure 3.5: Interface for the list of applicants*

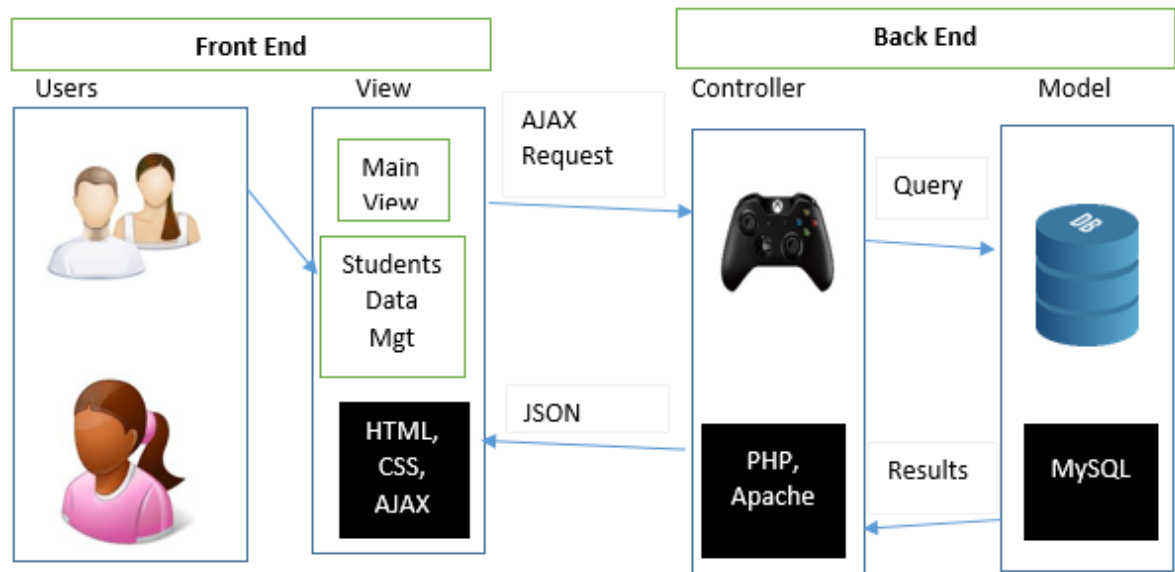
### 3.1.2: The Controller

The controller of the application uses PHP (read from the back as Hypertext Processor), facilitated by AJAX. PHP is a popular general-purpose scripting language that is especially suited to web development. It is fast, flexible, and practical. These two will make it easy to send and receive data from the model, that is MySQL and Apache Server, and display these data on the view using HTML and the CSS.

The system has a class called applicant.php which contains all methods and operations relating to applicants. It has functions that gets all applicants, get a specific applicant's details, counts applicants given a specific criterion and gets specific information on an applicant (e.g, school, community, grade, etc.).

The class "programareas.php" also contains the various operations on the Program Areas table in the model (database). At the NGO, a program area is a town or city whose students the NGO sponsors. Again, the class payments.php contains functions the pertain all or specific payments. Examples are getting payment requests (given a specific criterion), submitting a payment requests. All these have their corresponding JavaScript files which handle requests asynchronously in the background using AJAX.

The architecture of the system is shown in the diagram below:



*Figure 3.6: High level architecture of the system*

## Chapter 4: Implementation

### 4.1: Overview of implementation

Building on an existing application, the system, together with the database was given to be deployed and run for familiarisation. XAMPP and the database on MySQL server which comes with XAMPP had to be installed for this. Apart from the familiarisation with the interface, the code had to also be understood before changes or additions could be made to it. This was challenging because people have different styles of programming so it is difficult understanding the programming practices of another person.

#### 4.1.1: Libraries and Frameworks

##### *4.1.1.1: jQuery*

“jQuery UI is a curated set of user interface interactions, effects, widgets, and themes built on top of the jQuery JavaScript Library” (jQuery User Interface, n.d). Its main function is to make it easier to use JavaScript on the website. This library was used to enhance the user interactions on the system. This library was chosen out of the many JavaScript Libraries because it is the most widely used and the most extendable.

##### *4.1.1.2: Bootstrap Framework*

Bootstrap is an open-source front end framework used for designing websites and web applications. It was created at Twitter in 2010 and has become one of the most widely used front-end frameworks (Bootstrap, n.d.). This library was chosen because it is widely used and thus, has a wider community of people who can help in case of any problem. Again, it has base styling for most of the html tags or elements and it supports JavaScript plugins.

After all the above, the actual implementation of the requirements outlined in **Chapter 2** started. The remaining parts of this chapter outlines the implementation of the individual requirements

## 4.2 Email Implementation

### 4.2.1: Scope of Work

The main aim is to report to the programme manager, by mail, every Friday on how many new applicants have been entered in total over the week. The report will also include how many entries were made at each programme area. This will ensure the system is used at all the programme areas so that the programmes manager can have enough information on the system to make informed decisions.

### 4.2.2: Overview

This requirement dealt with the `student_applicant` table in the database and the `applicant.php` class. There is a function in the `applicant` class which returns a count of records on applicants given a filter. This method then gets the all the records entered in a week and a breakdown for the specific programme areas. Also, on getting the specifics from the program areas,

To send the mail, PHP's in built function called `mail()` was used. This function sends a mail given the receivers' address, subject, the message and the header. Also, to get entries made in the past seven days, MySQL function called `DATE_SUB()` which subtracts a criteria (either month, day or year) from a given date was used.

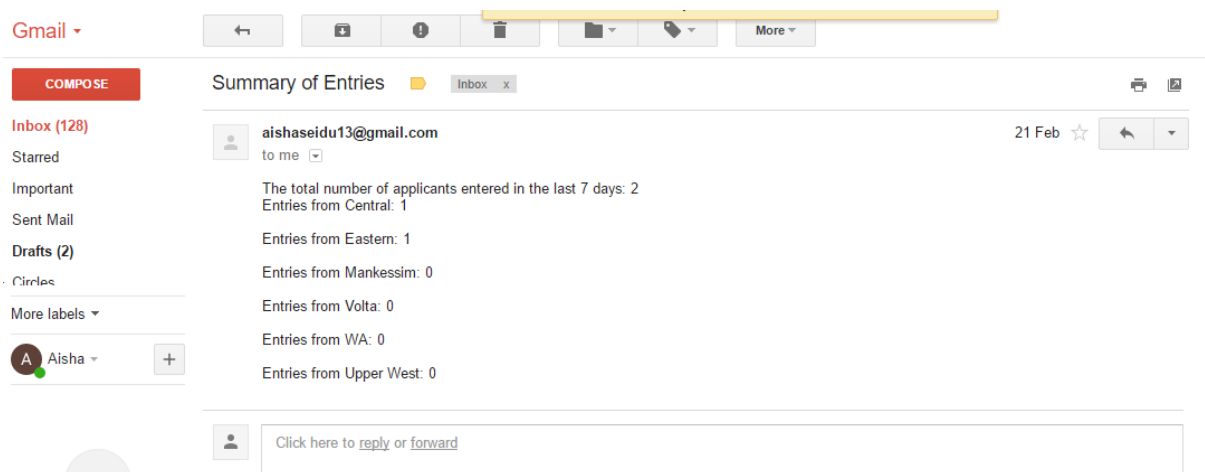
The main challenge was how to get the results from the query and add a message to it to be sent as a mail. After thorough research, output obscuring functions provided by PHP was discovered. These functions allow you to, instead of printing out a message from the

server side, store the message in a variable so that it can be used later. (See Appendix Figure A.1 for snippet of code showing how the functions work)

After getting all these information, the php.ini file was configured to allow mails to be sent from the localhost.

On every Friday, a mail should be sent to the given e-mail address giving records of applicants who have been added since the last period. Since the NGO uses a Linux server, they will use a **Cron** scheduler to trigger the mail every Friday. Cron is a system process, usually on Linux and UNIX, which automatically performs tasks given a set schedule (Computer Hope, n.d.).

The result of this implementation is shown in the screenshot below



*Figure 4.1: Email implementation result*

## 4.3: PDF Report.

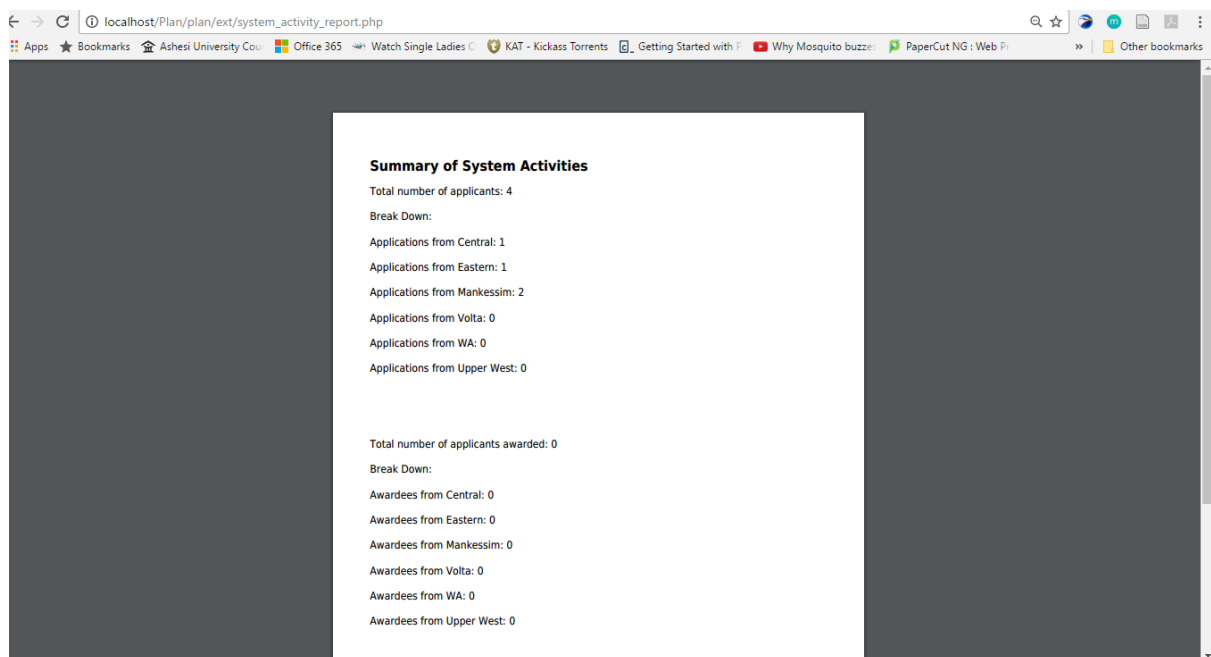
### 4.3.1: Scope of work

The main objective of this implementation was to generate a pdf report on activities that have been carried out on the system. These activities include entry of new applicants, awarded applicants and many more.

#### 4.3.2: Implementation Overview.

For this implementation, mPDF library was used. “mPDF is a PHP class which generates PDF files from UTF-8 encoded HTML. It is based on FPDF and HTML2FPDF with a number of enhancements” (mPDF, n.d). This was chosen because it has more functionalities than the FPDF (a basic PHP PDF library) Library. Again, the existing application already has a PDF functionality (pdf scholarship offer letter) which uses the mPDF Library so consistency needed to be ensured. The library is included in the beginning of the code so that instances of it can be created to access its functions. The data needed for the report was gotten from the applicant, program area, and packages tables. The classes needed for this implementation were the applicant.php and programarea.php

After this implementation, the manager is supposed to be able to generate pdf reports on important system activities. The result of this implementation is shown in the picture below:



*Figure 4.2: PDF implementation result*

## 4.4: Viewing Criteria for Generating Points

### 4.4.1: Scope of Work

Each applicant is awarded points based on certain information, and the total of these points for each applicant is what is used to award the scholarship. Applicants with high points are given priority over those with lower points. This implementation therefore sought to provide support data for programme manager to know how points awarded to each applicant are generated.

### 4.4.2: Overview

The parameters that are used to generate the points are the type of work the guardian does, the community the applicant comes from, the type of basic school applicant attended, and the aggregate grade the applicant got in the Basic Education Certificate Examination (BECE). The database that exists already have tables; “application\_occupation”, “community\_category”, “applicant\_school\_category”, and “selection\_grade\_point” respectively for this information.

There is a class, “occupation.php” that communicates with the “application\_occupation” table which has the occupation of the applicant’s guardian. It has a function called “get\_occupations” which returns all the occupations and the corresponding points they carry.

However, there was no class that communicated directly with the applicants' community, school type, and grade. Classes; "applicant\_community\_category.php", "applicant\_school\_category.php", and "applicant\_grade.php" were then created to communicate with the respective tables to get the details of each criteria.

After getting these details, a bootstrap template was used to format the tables to make it responsive and give it a once view.

After this implementation, a table showing the criteria used in generating the points. should be able to be generated. Below is a snapshot of the implementation above:

### Criteria For Awarding Points

Criteria	Value	Point Assigned
Occupation	not applicable	0
Occupation	Farmer	20
Occupation	Trader	20
Occupation	Artisan and Tradesmen	15
Occupation	Civil Servant and Teachers	10
Occupation	Professionals	0
Occupation	Not working	0
Community Type	rural	10
Community Type	urban	8
School Category	public	10
School Category	private	8
Applicant Grade	6-12	20
Applicant Grade	13-18	15
Applicant Grade	19-22	10
Applicant Grade	23-30	5

*Figure 4.3: Viewing point criteria implementation result*



## 4.5: Verifying Additions to Schools

### 4.5.1: Scope of Work

Because the list of high schools and basic schools are many, if the entry officer is adding a new applicant does not easily identify a school name from the drop-down list there is a tendency to just add that school as a new school.

The objective of this implementation is to ensure that, when the entry officer is adding a new applicant, he or she does not add a school that already exist in the database. This is to help prevent duplications and save server space

### 4.5.2: Overview

This implementation was done using JavaScript and jQuery. It was done with the idea of a live search. A function was written to search and return all the names of the schools in the database that match a given criteria. This function will be triggered, using ajax, when a user starts typing the school's name so that, all the school names that match the entered values will be suggested to the user. With this, a user will only add schools that do not already exist in the database. The snapshot below show how High School names are verified when adding a new school. This implementation is the same for Basic Schools.



#### 4.6.2: Overview of Implementation

A login feature was implemented, using a bootstrap template, for the accountant to give him/her permission to access the payments before proceeding (see appendix Figure A.3)

The entities involved in this implementation are the scholarship\_package (containing detailed information on each student's scholarship), the program area, the high school, the payment request, and the scholarship payment tables. A function was written to get the payment requests currently in the system. Each payment request has several scholarship payments connected to it so a function was written to return all the scholarship payment given the payment requests ID. Again, another function was to change payment status, because the accountant is only in charge of making or cancelling disbursements, the function had to ensure that, the changes can be made if only it request status is either new request or disbursed.

After implementing the above functions, a Bootstrap modal and AJAX were used to enable the accountant to see the details of any payment request (without reloading the pages) before deciding to disburse it or not. AJAX request was also used to communicate with the server to disburse.

After this implementation, accountant, after logging in should see payment requests and be able to view all the scholarship payments associated with each request. The results of this implementation are shown in the images below.

## Accounts

### Payment Requests

DISBURSEDYES

Request Code	Request Year	Request Date	Amount	Request Status	Program Area	See More	Action
Fall 2017	2013	2017-03-06	800	REQUESTED	WA	<a href="#">See Details</a>	<a href="#">Disburse</a>
EASTER 2014 PAY 2	2014	2014-05-30	0	DISBURSED	Eastern	<a href="#">See Details</a>	<a href="#">Cancel Disbursement</a>
NorthTransport	2017	2017-04-11	150	REQUESTED	WA	<a href="#">See Details</a>	<a href="#">Disburse</a>
Spring Stipend	2017	2017-04-10	750	REQUESTED	Mankessim	<a href="#">See Details</a>	<a href="#">Disburse</a>
Boarders Fees	2017	2017-04-03	500	REQUESTED	Eastern	<a href="#">See Details</a>	<a href="#">Disburse</a>
SRCDues	2017	2017-04-03	40	REQUESTED	Eastern	<a href="#">See Details</a>	<a href="#">Disburse</a>

*Figure 4.6: Disbursing payment requests implementation result*

The screenshot shows a web browser at localhost/Plan/plan/ext/payment\_requests.php. A modal window titled "Scholarship Payments Under Payment Request" is displayed over a table of payment requests. The modal contains a table with the following data:

Firstname Name	Last Name	Year	Request Date	Amount	School	Program Area	Status
Ibrahim	Osman	2013	2013-03-06 00:00:00	800	ODORGONNO SHS	WA	NEW REQUEST
Ibrahim	Osman	2013	2013-03-06 00:00:00	800	ODORGONNO SHS	WA	NEW REQUEST

The modal has "Close" and "Save Changes" buttons. The background table shows various payment requests with columns for Request Code, Request Year, Request Date, Amount, Request Status, Program Area, and Action (See Details, Disburse).

*Figure 4.7: Disbursing Payment Requests Implementation Result 1*

## Chapter 5: Testing

After implementation, the necessary tests were done to ensure the application or the module does what it is expected to do. Testing also helps to see if there are any errors in the implantation and helps check software compatibility with its environment. The tests carried out on the system are Unit/Development Testing, Component Testing, and Compatibility Testing.

It should be noted that testing was done in parallel with implementation even though it is documented separately.

The requirements that were tested are listed below:

- Sending Mails to the Programmes Manager on entries made in the past week
- Generating PDF report on summary of system activities
- Viewing Criteria for Awarding points to scholars
- Verifying Additions to school
- Disbursing payment requests

### 5.1: Unit Testing

This test is done at the development stage. It involves testing the individual classes, functions, and objects to see. To do this testing, the test plan which is shown below was used.

**Table 5.1: Unit test for Send Email Module**

Function	Unit Testing Point	Testing	Results
get_applicants_count(\$filter)	SQL statement to get applicant count of applicants entered in the last 7 days	Write SQL statement with test and run it using phpMyAdmin <b>Test input:</b> none <b>Expected Results:</b> A message confirming successful selection of data	0
	Get_applicant_count(\$filter) method in the applicant class	Create an object of the applicant class and call the method to test <b>Inputs(s):</b> a filter <b>Expected Output:</b> a number	0
get_program_areas()	SQL statement to get the program areas	Write SQL statement with test and run it using phpMyAdmin <b>Test input:</b> none <b>Expected Results:</b> A message confirming successful selection of data	Showing rows 0 - 5 (6 total, Query took 0.0318 seconds.)
	get_programareas() method in the applicant class	Create an object of the programareas class and call the method to test <b>Inputs(s):</b> None <b>Expected Output:</b> a JSON array	Array ( [programarea_id] => 1 [programarea_name] => Central ) Array ( [programarea_id] => 2 [programarea_name] => Eastern ) Array ( [programarea_id] => 3 [programarea_name] => Mankessim ) Array ( [programarea_id] => 4 [programarea_name] => Volta ) Array ( [programarea_id] => 5 [programarea_name] => WA ) Array ( [programarea_id] =>

			6 [programarea_name] => Upper West )
--	--	--	--

The Unit Test for the various functional Requirements can be found in appendices (A.5 to A.11)

## 5.2: Component Testing

After the various classes have been put together to create a model, the models or functionalities were tested separately without integrating them into the bigger system. This was to see how the various classes and objects interact with each other so that, errors and conflicts can be resolved.

To do this, the URLs to the various components were entered in the browsers and loaded to see if the results meet expectation. For example, with the View Point Criteria implementation, if the URL to the PHP file is entered into the browser and loaded, it expected to return a table outlining the various criteria and the points allocated to each value. The results of this test can be found in the implementation of each requirement.

## 5.3: Compatibility Testing

This was done to see if the models will work in the external environments. These environments include the operating systems, browsers and so on. Since frameworks were also used in the implementation, the compatibility testing was also to see under which conditions they function. The focus of this testing was on browsers since the application is going to be used mostly on a web browser. All the modules and libraries worked perfectly on Google Chrome and Microsoft Edge except the mPDF library, which was not compatible with Microsoft Edge. Regarding Operating System, the application was tested on a Windows 10 Machine and it worked perfectly

## **Chapter 6: Conclusion and Recommendations**

### **6.1: Conclusion**

The requirements implemented in this project can be used by the NGO to help manage their payments to scholars. It will also help the organisation to easily generate report on actives to facilitate easy monitoring and accountability.

The modules implemented in this project can be integrated into the NGO existing system without any negative effects. Also, any company or institution whose processes are in line the activities of the organization for which this was developed can use the system with just a little modification.

### **6.2: Recommendations**

It is recommended that, anyone who works on this project to focus designing a better database for the system which the scope of this project as assigned did not permit. As already mentioned in the third chapter, the database design has some redundancies in it, so a better design will help ensure faster querying of the data. Beyond just managing scholarship application and funds, scholars' performances in their respective schools can be captured in the system.

Donors can also be made part of the system where they can donate and track the cause their monies are being used for.

Currently, applicants apply on paper before staff at the NGO enter the details onto their system. It will therefore be recommended that, applicants be made part of the system's users. This can be done by creating a portal or an interface where scholars can apply online from anywhere. This will reduce the workload on the programme managers. With the prevalence of mobile phones, a mobile application of the system can be created, especially for entering new applicant's information.





## References

- Capterra. (n.d.). *FluidReview*. Retrieved November 1, 2016, from <http://www.capterra.com/grant-management-software/spotlight/123374/FluidReview/SurveyMonkey>
- Computer Hope. (n.d.). *Linux and Unix Crontab Command*. Retrieved March 14, 2017, from Computer Hope: [www.computerhope.com/unix/ucrontab.htm](http://www.computerhope.com/unix/ucrontab.htm)
- Forbes-Pitt, K. (2004). Getting Rid of Registration at LSE. *32nd Annual ACM SIGUCC Conference on User Services* (pp. 110-114). ACM.
- ICARIS LTD. (n.d.). *ICARIS GRANT MANAGEMENT MODULE*. Retrieved November 1, 2016, from ICARIS: <http://www.icaris.co.uk/grantmanagementsoftware.html>
- Takashi, K., & Liang, E. (1997). Analysis and design of Web-based information systems. *Computer Networks and ISDN Systems*, 29, 1167-1180. Retrieved January 19, 2017
- W3Schools. (n.d.). *jQuery Introduction*. Retrieved January 22, 2017, from W3Schools: [https://www.w3schools.com/jquery/jquery\\_intro.asp](https://www.w3schools.com/jquery/jquery_intro.asp)

## Appendix

```
require('mpdf/mpdf.php');

$app = new applicants();
//$app1 = new applicants();
$prog = new programareas();
$prog1 = new programareas();
$grant = new grants();
$num = $prog->get_area_count();
$area = $prog1->get_programareas();

if (!$area) {echo "could not get programme areas"; exit();}
$results = $prog1->fetch();
if (!$results) {echo "could not fetch programme areas"; exit();}

$pdf = new mPDF();
$pdf->AddPage();
$pdf->SetFont('Arial','B',16);
$pdf->Cell(40,10,'Summary of System Activities');
$pdf->Ln();

$pdf->SetFont('Arial','',12);
$pdf->Cell(40,10,'Total number of applicants: '.$app->get_applicant_count());
$pdf->Ln();

$pdf->SetFont('Arial','',12);
$pdf->Cell(40,10,'Break Down: ');
$pdf->Ln();
```

*Figure A.1 Implementation of PDF report generation*

```
40 $msg= ob_get_contents();
41 ob_end_clean();
42 //echo "<br>";
43 //getting entries from communities
44 $message = $message.$msg;
45 $filter = "e_date>=DATE_SUB(NOW(),INTERVAL 7 DAY) && programarea_id = ";
46
47 for ($i=1; $i<=$num; $i++) {
48     $filter = $filter.$i;
49     ob_start();
50     echo "Entries from ".$results['programarea_name'].": ".$applicant->get_applicants_count($filter)."\n \n";
51     //echo "<br>";
52     $msg= ob_get_contents();
53     ob_end_clean();
54     $filter = " e_date>=DATE_SUB(NOW(),INTERVAL 7 DAY) && programarea_id = ";
55     $results = $obj1->fetch();
56     $message = $message.$msg;
57 }
58
59 //Setting mail parameters and sending the mail
60 $To = 'aishaseidu13@gmail.com';
61 $Subject = 'Summary of Entries';
62 $Message = $message;
63 $Headers = 'From: aishaseidu13@gmail.com';
64 if (!mail($To, $Subject, $Message, $Headers)) {
65     echo "Mail not sent";
66     exit();
67 }
68 echo "mail sent";
69
```

*Figure A. 2: Code for sending mail using PHP mail() function and object obscuring functions*

Please sign in to Accounts

alvin.ofori

.....

☐ Remember me

Sign in

*Figure A.3 Login page for accounts*

## Accounts

### Payment Requests

Request Code	Request Year	Request Date	Amount	Request Status	Program Area	See More	Action
Fall 2017	2013	2017-03-06	800	REQUESTED	WA	<a href="#">See Details</a>	<a href="#">Disburse</a>
EASTER 2014 PAY 2	2014	2014-05-30	0	REQUESTED	Eastern	<a href="#">See Details</a>	<a href="#">Disburse</a>
NorthTransport	2017	2017-04-11	150	REQUESTED	WA	<a href="#">See Details</a>	<a href="#">Disburse</a>
Spring Stipend	2017	2017-04-10	750	REQUESTED	Mankessim	<a href="#">See Details</a>	<a href="#">Disburse</a>
Boarders Fees	2017	2017-04-03	500	REQUESTED	Volta	<a href="#">See Details</a>	<a href="#">Disburse</a>
SRCDues	2017	2017-04-03	40	REQUESTED	Eastern	<a href="#">See Details</a>	<a href="#">Disburse</a>
LabDues	2017	2017-04-03		REQUESTED	Volta	<a href="#">See Details</a>	<a href="#">Disburse</a>

*Figure A.4 All payment requests*

*Table A.5: Unit test for view point criteria implementation*

Function	Unit Testing Point	Testing	Results
get_community_category()	SQL statement to get community categories and points allocated to each category	Write SQL statement with test and run it using phpMyAdmin <b>Test input:</b> none <b>Expected Results:</b> A message confirming successful selection of data	Showing rows 0 - 1 (2 total, Query took 0.0546 seconds.)
	Get_community_category() method in community_category class	Write a code to create object of the class and call the get_community_category() method <b>Test input:</b> none <b>Expected Results:</b> An array of the details selected	Array ( [community_category_id] => 1 [community_category_name] => rural [community_category_app_point] => 10 ) Array ( [community_category_id] => 2 [community_category_name] => urban [community_category_app_point] => 8 )
		Write a php unit test to test the code Input: Diagnosis Details Output: true Assert: assertEquals	
get_applicant_school()	SQL statement to get applicants' school categories and points allocated to each category	Write SQL statement with test and run it using phpMyAdmin <b>Test input:</b> none <b>Expected Results:</b> A message confirming successful selection of data	Showing rows 0 - 1 (2 total, Query took 0.0651 seconds.)

	Get_applicant_category() method in applicant_school_category class	Write a code to create object of the class and call the get_applicant_school() method <b>Test input:</b> none <b>Expected Results:</b> An array of the details selected	Array ( [applicant_school_category_id] => 1 [applicant_school_category_name] => public [applicant_school_category_app_point] => 10 ) Array ( [applicant_school_category_id] => 2 [applicant_school_category_name] => private [applicant_school_category_app_point] => 8 )
get_applicant_grade()	SQL statement to get applicants' grade categories and points allocated to each it	Write SQL statement with test and run it using phpMyAdmin <b>Test input:</b> none <b>Expected Results:</b> A message showing the number of rows selected and the time the query took to execute	Showing rows 0 - 3 (4 total, Query took 0.0469 seconds.)
	get_applicant_grade() method in applicant_school_category class	Write a code to create object of the class and call the get_applicant_grade() method <b>Test input:</b> none <b>Expected Results:</b> An array of the details selected	Array ( [selection_grade_points_id] => 0 [grade_point] => 20 [min_grade] => 6 [max_grade] => 12 ) Array ( [selection_grade_points_id] => 2 [grade_point] => 15 [min_grade] => 13 [max_grade] => 18 ) Array ( [selection_grade_points_id] => 3 [grade_point] => 10 [min_grade] => 19 [max_grade] => 22 ) Array ( [selection_grade_points_id] => 4 [grade_point] => 5 [min_grade] => 23 [max_grade] => 30 )

Function	Unit Testing Point	Testing	Results
get_grants()	SQL statement to get applicant count of applicants entered in the last 7 days	Write SQL statement with test and run it using phpMyAdmin <b>Test Input:</b> None <b>Expected Results:</b> A message confirming successful selection of data	Showing rows 0 - 5 (6 total, Query took 0.0614 seconds.)
	Get_grants() method in the grants class.	Create an object of the grant class and call the get_grants() method <b>Test Input:</b> None <b>Expected Results:</b> An array of the various grant packages	Array ( [grant_package_id] => 1 [name] => GAD GRA 0095 [code] => GAD GRA 0115 [annual_amount] => 300 ) Array ( [grant_package_id] => 2 [name] => GAD GRA 0115 [code] => GAD GRA 0115 [annual_amount] => 300 ) Array ( [grant_package_id] => 3 [name] => GAD GRA 0047 [code] => GAD GRA 0047 [annual_amount] => 300 ) Array ( [grant_package_id] => 4 [name] => Honkook [code] => GHA 0160 [annual_amount] => 400 ) Array ( [grant_package_id] => 5 [name] => Sponsorship Fund [code] => SF [annual_amount] => 600 )

**Table A.6: Unit test for PDF report**

Function	Unit Testing Point	Testing	Results
get_grants()	SQL statement to get applicant count of applicants entered in the last 7 days	Write SQL statement with test and run it using phpMyAdmin <b>Test Input:</b> None <b>Expected Results:</b> A message confirming successful selection of data	Showing rows 0 - 5 (6 total, Query took 0.0614 seconds.)
	Get_grants() method in the grants class.	Create an object of the grant class and call the get_grants() method <b>Test Input:</b> None <b>Expected Results:</b> An array of the various grant packages	Array ( [grant_package_id] => 1 [name] => GAD GRA 0095 [code] => GAD GRA 0115 [annual_amount] => 300 ) Array ( [grant_package_id] => 2 [name] => GAD GRA 0115 [code] => GAD GRA 0115 [annual_amount] => 300 ) Array ( [grant_package_id] => 3 [name] => GAD GRA 0047 [code] => GAD GRA 0047 [annual_amount] => 300 ) Array ( [grant_package_id] => 4 [name] => Honkook [code] => GHA 0160 [annual_amount] => 400 ) Array ( [grant_package_id] => 5 [name] => Sponsorship Fund [code] => SF [annual_amount] => 600 )



Table A.7 Unit Test for disbursing payments

Function	Unit Testing Point	Testing	Results
get_all_payment_requests()	SQL statement to get all payment requests	Write SQL statement with test and run it using phpMyAdmin <b>Test Input:</b> None <b>Expected Results:</b> A message showing number of rows returned	Showing rows 0 - 6 (7 total, Query took 0.0281 seconds.)
	get_all_payment_requests() method in the payments class.	Create an object of the payments class and call the get_all_payment_requests() method <b>Test Input:</b> None <b>Expected Results:</b> An array of all payment requests	<b>See Appednix Figure A.9</b>
get_payment_request_students()	SQL statement to get all students on a given requests	Write SQL statement with test and run it using phpMyAdmin <b>Test Input:</b> None <b>Expected Results:</b> A message showing number of records returned	Showing rows 0 - 0 (1 total, Query took 0.0985 seconds.)
	get_payment_request_students() method in the payments class	Create an object of the payments class and call the get_all_payment_requests() method Test Input: payment request id Expected Results: An array of all scholarship payments on relating to that particular payments request	<b>See Appednix Figure A.10</b>

```
localhost/Plan/plan/ext/school.php

Array ( [school_id] => 186 [address] => [school_name] => ;lllll [towncity] => Kpando ) Array ( [school_id] => 143 [address] => BOX KU 74 [school_name] => A.M.E ZION GIRLS [towncity] => WINNEBA ) Array ( [school_id] => 109 [address] => BOX 29 [school_name] => ABAKRAMPA SHS [towncity] => ABAKRAMPA C/R ) Array ( [school_id] => 161 [address] => [school_name] => Ab Senior High School [towncity] => Abor ) Array ( [school_id] => 66 [address] => BOX AB 39 [school_name] => ABURAMAN SHS [towncity] => ABURA DUNKWA ) Array ( [school_id] => 108 [address] => P OBOX 18 ABURI AKUAPEM [school_name] => ABURI SEC / TECH [towncity] => TOWN ) Array ( [school_id] => 86 [address] => BOX CT 772 [school_name] => ACADEMY OF CHRIST THEKING [towncity] => CAPE COAST ) Array ( [school_id] => 61 [address] => [school_name] => ACCRA ACADEMY [towncity] => ) Array ( [school_id] => 92 [address] => PMB [school_name] => ACCRA GIRLS SHS [towncity] => ACCRA ) Array ( [school_id] => 62 [address] => [school_name] => ACCRA SENIOR HIGH SCHOOL [towncity] => ) Array ( [school_id] => 81 [address] => [school_name] => ACHIMOTA SHS [towncity] => ACCRA ) Array ( [school_id] => 176 [address] => [school_name] => Ada Sec. Tech. school [towncity] => Sege Ada ) Array ( [school_id] => 100 [address] => BOX 14 [school_name] => ADANKWAMAN SHS [towncity] => ASSIN DARMANG ) Array ( [school_id] => 118 [address] => [school_name] => ADEISO SHS [towncity] => ) Array ( [school_id] => 215 [address] => [school_name] => ADIDOME SHS [towncity] => VOLTA ) Array ( [school_id] => 129 [address] => PO BOX83,CAPECOAST [school_name] => ADISADEL COLLEGE [towncity] => CITY ) Array ( [school_id] => 208 [address] => [school_name] => ADOM PRINT SHS [towncity] => ) Array ( [school_id] => 22 [address] => [school_name] => ADONTEN SHS [towncity] => ) Array ( [school_id] => 199 [address] => [school_name] => ADUKROM SHS [towncity] => ADUKROM ) Array ( [school_id] => 163 [address] => [school_name] => Afadjato Sec. Tech. School [towncity] => Liua ) Array ( [school_id] => 164 [address] => [school_name] => Afadjato Sec. Tech. school [towncity] => Liati ) Array ( [school_id] => 23 [address] => [school_name] => AGGREY MEMORIAL SHS [towncity] => ) Array ( [school_id] => 24 [address] => [school_name] => AGONA NSABA PRESBY SHS [towncity] => ) Array ( [school_id] => 102 [address] => [school_name] => AHANTAMAN GIRLS SHS [towncity] => SEKONDI ) Array ( [school_id] => 214 [address] => [school_name] => AHANTAMAN SHS [towncity] => TAKORADI ) Array ( [school_id] => 69 [address] => BOX 29 [school_name] => AHMADIYAA SENIOR HIGH SCHOOL [towncity] => CENTRAL ) Array ( [school_id] => 179 [address] => [school_name] => Akatsi Sec. Tech. Sch. [towncity] => Akatsi ) Array ( [school_id] => 1 [address] => BOX 47 [school_name] => AKIM SWEDRU SHS [towncity] => AKIM SWEDRU ) Array ( [school_id] => 166 [address] => [school_name] => Akpafu S.H.T.S [towncity] => Akpafu ) Array ( [school_id] => 16 [address] => none [school_name] => Akro Secondary Technical Sch. [towncity] => none ) Array ( [school_id] => 156 [address] => [school_name] => AKUSE METHODIST SHS [towncity] => AKUSE ) Array ( [school_id] => 141 [address] => BOX 46 [school_name] => AKWAMUMAN SHS [towncity] => AKOSOMBO ) Array ( [school_id] => 112 [address] => [school_name] => AL-UMMAH SHS [towncity] => EGUAFU ABREM ) Array ( [school_id] => 25 [address] => [school_name] => ANFOEGA SHS [towncity] => ) Array ( [school_id] => 190 [address] => [school_name] => ANLO SHS [towncity] => ANLO, V/R ) Array ( [school_id] => 99 [address] => BOX 593 [school_name] => ANTOA SHS [towncity] => KUMASI ) Array ( [school_id] => 140 [address] => PO BOX10 ANUM [school_name] => ANUM PRESBYSHS [towncity] => TOWN ) Array ( [school_id] => 64 [address] => [school_name] => APAM SHS [towncity] => ) Array ( [school_id] => 198 [address] => [school_name] => APEGUSO SHS [towncity] => APEGUSO, E/R ) Array ( [school_id] => 211 [address] => [school_name] => ARCHBISHOP PORTER GIRLS [towncity] => TAKORADI ) Array ( [school_id] => 26 [address] => [school_name] => ASAMANKESE SHS [towncity] => ) Array ( [school_id] => 17 [address] => Asesewa [school_name] => Asesewa Senior High School [towncity] => nor ) Array ( [school_id] => 82 [address] => [school_name] => ASSIN MANSO SHS [towncity] => ASSIN ) Array ( [school_id] => 70 [address] => BOX 30 MANKESSIM [school_name] => ASSIN NSUTA SENIOR HIGH [towncity] => MANKESSIM ) Array ( [school_id] => 27 [address] => [school_name] => ASSIN STSTE COLLEGE, A BEREKU [towncity] => ) Array ( [school_id] => 87 [address] => )
```

Figure A.8 Result of unit test for verifying schools

```
localhost/Plan/plan/ext/payments.php

Array ( [payment_request_id] => 8 [code] => Fall 2017 [year] => 2013 [request_date] => 2017-03-06 [programarea_id] => 5 [request_status] => REQUESTED [financial_year_financial_year_id] => 2 [amount] => 800 [group_id] => 1 [verification_document] => [liquidationdoc] => [programarea_name] => WA [year_name] => FA2011To2012 ) Array ( [payment_request_id] => 7 [code] => EASTER 2014 PAY 2 [year] => 2014 [request_date] => 2014-05-30 [programarea_id] => 2 [request_status] => DISBURSED [financial_year_financial_year_id] => 3 [amount] => 0 [group_id] => -1 [verification_document] => [liquidationdoc] => [programarea_name] => Eastern [year_name] => FIN YEAR 2013 2016 ) Array ( [payment_request_id] => 9 [code] => NorthTransport [year] => 2017 [request_date] => 2017-04-11 [programarea_id] => 5 [request_status] => REQUESTED [financial_year_financial_year_id] => 4 [amount] => 150 [group_id] => [verification_document] => [liquidationdoc] => [programarea_name] => WA [year_name] => FY2016To2017 ) Array ( [payment_request_id] => 10 [code] => Spring Stipend [year] => 2017 [request_date] => 2017-04-10 [programarea_id] => 3 [request_status] => REQUESTED [financial_year_financial_year_id] => 4 [amount] => 750 [group_id] => [verification_document] => [liquidationdoc] => [programarea_name] => Mankessim [year_name] => FY2016To2017 ) Array ( [payment_request_id] => 11 [code] => Boarders Fees [year] => 2017 [request_date] => 2017-04-03 [programarea_id] => 2 [request_status] => REQUESTED [financial_year_financial_year_id] => 4 [amount] => 500 [group_id] => [verification_document] => [liquidationdoc] => [programarea_name] => Eastern [year_name] => FY2016To2017 ) Array ( [payment_request_id] => 12 [code] => SRCDues [year] => 2017 [request_date] => 2017-04-03 [programarea_id] => 2 [request_status] => REQUESTED [financial_year_financial_year_id] => 4 [amount] => 40 [group_id] => [verification_document] => [liquidationdoc] => [programarea_name] => Eastern [year_name] => FY2016To2017 ) Array ( [payment_request_id] => 13 [code] => LabDues [year] => 2017 [request_date] => 2017-04-03 [programarea_id] => 4 [request_status] => REQUESTED [financial_year_financial_year_id] => 4 [amount] => [group_id] => [verification_document] => [liquidationdoc] => [programarea_name] => Volta [year_name] => FY2016To2017 )
```

Figure A.9 Results of unit test for getting all payment requests

```
localhost/Plan/plan/ext/payments.php

Array ( [payment_request_payment_request_id] => 11 [scholarship_payment_id] => 56 [date] => 2017-04-15 00:00:00 [status] => DISBURSED [amount] => [schools_school_id] => 15 [scholarship_package_id] => 403 [start_date] => 2014-09-02 00:00:00 [end_date] => 2017-09-02 00:00:00 [scholarship_annual_amount] => 1000 [ifnull(spacak_status',0)] => 1 [sponsored_student_id] => 191 [student_firstname] => Alvin [student_middlename] => Selorm [student_lastname] => Ofori [student_applicant_student_applicant_id] => 851 [programarea_id] => 2 [programarea_name] => Eastern [DistrictID] => 1 [District] => AGONA EAST [community_id] => 2 [community] => Aboano [student_telephone_1] => [student_telephone_2] => [student_dob] => 2016-05-17 00:00:00 [app_submission_year] => 2014 [student_gender] => M [app_grant_id] => 5 [grant_package_id] => 5 [grant_name] => Sponsorship Fund [annual_amount] => 600 [school_id] => 15 [school_name] => Jabez Educational Institute )
```

Figure A.10 Unit test result for getting scholarship payment for a given payment request

Table A.11: Other functions under payments tested

Function	Role
user_is_verified(\$n, \$p)	Verifies if a user has access or not to accounts and payments given a username and a password
change_request_status(request_id)	Changes the status of a given payment request to either <b>"Disbursed"</b> or <b>"Requested"</b> and the corresponding scholarship payments associated with it to <b>"Disbursed"</b> or <b>"New Request"</b> . Only the accountant can invoke this method
change_payment_status(payment_id)	Changes the status of a given scholarship payment given an ID to either <b>"Disbursed"</b> or <b>"New Request"</b> . Only the accountant can do this
liquidate_payment_request(\$payment_request_id)	liquidates a given payment request given an id. <b>Modified to liquidate all corresponding scholarship payments</b>
create_payment_request(\$financial_year_id, \$programarea_id,\$request_name,\$owner_id=0)	Allows the programme manager to add a new payment request given all the details of a payment request