ASHESI

ASHESI UNIVERSITY COLLEGE

Improving Scholarship Management Application for a Non-Governmental Organization

APPLIED PROJECT

B.Sc. Computer Science

George Ocran

2017

ASHESI UNIVERSITY COLLEGE

Improving Scholarship Management Application for Non-Governmental Organization

Applied Project

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

George Ocran

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 preparation and presentation of this applied project were supervised in accordance with the guidelines on supervision of applied project laid down by Ashesi University College.

Supervisor’s Signature:

……………………………………………………………………………………………

Supervisor’s Name:

……………………………………………………………………………………………

Date: ……………………………………………………………………………………………

Acknowledgements

A special thank you goes to my supervisor, Mr. Aelaf Dafla. Your support, guidance and leadership were invaluable in making this project a reality. Thank you for not giving up on me when the going got tough.

Also, a big thank you goes to my friends and especially my loving family. You are the reason I have made it this far. Thank you for being patient with me and being a part of my life.

Ultimately, all this will not be a possibility if not for God. To You be all the glory.

Abstract

This project was carried out for an NGO which aims to improve the lives of children and their families by providing scholarships to children who do not have the finances to get a quality education. This NGO's scholarship process was automated in order to improve the scholarship management process. Although the software was successfully deployed, it is not as widely used as expected and this threatens its success. Solving this problem is essential because subsequent versions and new functionalities implemented obviously hinge on the software's usage. As such, this project targets software deployment, a critical activity of software development process, as one of the possible causes of the lack of usage. Consequently, this project outlined the best practices of software deployment from research that can be applied for the deployment of the functionalities in this project and of future releases.

Another objective of this project was to add functionalities to the existing software to cater for a wider user base. The functionalities that were added are for donors of the organization in order to improve accountability and transparency and also to provide vital information for donors for decision-making. The functionalities include being able to view a list of students under a particular grant, viewing the various payments of the grant across the years amongst others. These will also include graphs that will display various information regarding the payments, communities and students.

Table of Contents

Chapter 1: Introduction

1.1 Background

This project is based on a Non-Governmental Organization (NGO) that specializes in empowering children to ensure a better future for themselves and their families. One of the ways this organization achieves this is through education. By providing scholarships, this NGO allows girls and boys to access quality education to give them a better chance of enjoying a higher standard of living in the future. The aim ultimate goal is to alleviate poverty for these individuals and their families, to increase their chances of survival for themselves and for future generations and to give them the platform to make a meaningful contribution to society.

The scholarship scheme was automated and scholarships are now awarded to students using software. The software has been deployed and is in use by the organization, however not to the extent that developers would have hoped. The focus of this project, thus, is to delve into the reasons why software may not be widely adopted by the target users. Specifically, this paper will explore software deployment in order to build features and deploy them effectively.

According to research, the software deployment process is a very important stage in the software development process. Unkelos-Shpigel and Hadar (2013) emphasize this point by pointing out that deployment warrants its own deployment architecture in the software development cycle. Medvidovic and Malek (2011) also point out that the deployment architecture can affect the non-functional requirements of software thereby affecting the Quality-of-Service (QoS). Similarly, Bloom and Clark (2008) also hammer on the importance of software deployment on vendors and customers alike stating that "the ease, speed and success of deployment are drivers of initial customer satisfaction, support calls and deployment costs."

As such, it is important that the updates to the software that will be made in this project be deployed efficiently and effectively to ensure the organization's staff enjoy optimum software.

Another objective of this project is to provide features for donors for the NGO's scholarship operation. Since the organization relies on donations, it is important that their scholarship disbursements be made readily available to all donors. This ensures that the operation is transparent and accountable which projects an image of trustworthiness for the organization. This will enable the organization to receive more funding and increase its impact. Additionally, donors will have more flexibility and have more power in deciding which students receive scholarships or which communities they would like to benefit from their donations. This project aims to implement these features for donors bearing in mind that they hold the potential to incentivize more donations and help the organization to have a bigger impact by providing more opportunities for children and their families. Ultimately, it is hoped that this seemingly small addition will improve the wellbeing of numerous people on the African continent in the long term.

1.2 Related Work

Some research has been conducted into the challenges involved in software deployment. Unkelos-Shpigel and Hadar (2013) conducted research involving deployment architects from seven different firms to glean out factors that will lead to successful software development. This research provided activities that would prove useful in the deployment of the features in this project.

One of these activities is brainstorming. The research suggests that brainstorming meetings be conducted at the time of software release in which major decisions involving all process partners, including developers and deployment architects, are taken (Unkelos-Shpigel and Hadar, 2013). This will be vital in the release of subsequent software updates with the organization’s software. Decisions made at this stage will have the prior knowledge of previous installations carried out at the organization. This will include the challenges faced at the time and also customer experience with the deployment. This step will be crucial for successful deployment and enhanced user experience.

Another important predictor to software deployment is the use of automated tools (Unkelos-Shpigel and Hadar, 2013). An example of an automated tool is GNU Make. GNU Make has a number of capabilities that make deployment easier. Amongst these is the ability to build and install files without knowing the details of how it is done; this significantly reduces the complexity of deployments (“gnu.org,” n.d.). Additionally, it can determine which files need to be updated based on the source files and can also update files in the proper order in case a file depends on another. This means that the entire program may not need to be recompiled, saving time and computing resources (“gnu.org,” n.d.). GNU Make is also conveniently language-independent and can be used to uninstall packages also (“gnu.org,” n.d.). Automated tools greatly simplify the software deployment process by cutting down the amount of human involvement during the deployment. This also reduces the likelihood of human errors occurring during the process thus increasing the chances of a successful deployment.

Mäntylä and Vanhanen (2011) in their research obtain information of four successful Finnish IT companies on their software deployment activities and challenges in order to answer research questions. The first question they sought to answer is: What deployment activities exist and how are they performed? The second was to find the main goals of software deployment and the challenges associated with them.

The deployment activities were grouped into four: stakeholder communication, installation preparations, installation and testing. Stakeholder communication involved informing the stakeholder of the contents of the deployment as well as user training and support. Installation preparations involved importing initial customer data, configuring and integrating the software, creating a deployment package and scheduling a deployment date. For three out of the four companies, a tool, such as GNU Make, was used in the creation of the deployment package. Installation involved preinstall checks, the actual installation of the software and ensuring that rolling back was possible if deployment was unsuccessful. The final activity, testing, required that the software was tested at the vendor and the customer's site. Depending on the complexity of the software it may be impossible to test the software at the vendor's site. Additionally, testing was carried out at the customer's site but only to a limited extent since the testing tools are not available to the users to test. Testing of this project should be significantly less complex. This is because dummy students can be added to the database and tested from the development site. Testing should be fairly straightforward at the organization's site after any updates are made because all the information required for the new features are already contained in the database.

The goals for the deployment can be divided into vendor goals and customer goals. Vendor goals were to reduce deployment efforts and decrease the customers' dependency on individual experts. For these companies, customers required Commercial off-the-shelf (COTS) software to be customized to suit their particular businesses or for a highly complex software to be developed for their use. In these cases, the deployment effort will be great due to the presence of many failure points. For this project, the deployment effort was considerably less because the software was built solely for this organization. Also, the software is already installed on their system thus new developments are only updates. Deploying updates are typically less problematic than clean installs and this is evidenced by the fact that customers usually "emphasize testing more during clean installs than updates" according to Mantyla and Vanhanen (2011). In the case studies, companies that deployed the software required the expertise of one highly knowledgeable worker who was always present. His or her absence usually led to significant problems. These companies are resigned to keeping these individuals because knowledge transfer to other employees is a laborious process (Mantyla & Vanhanen, 2011). Similarly, in deploying the new features, it is important that someone with prior knowledge in the previous installation and with the development of the software be heavily involved in its deployment. The reason for this is that problems that occur will be easier to rectify.

Customers' goals for deployment are:

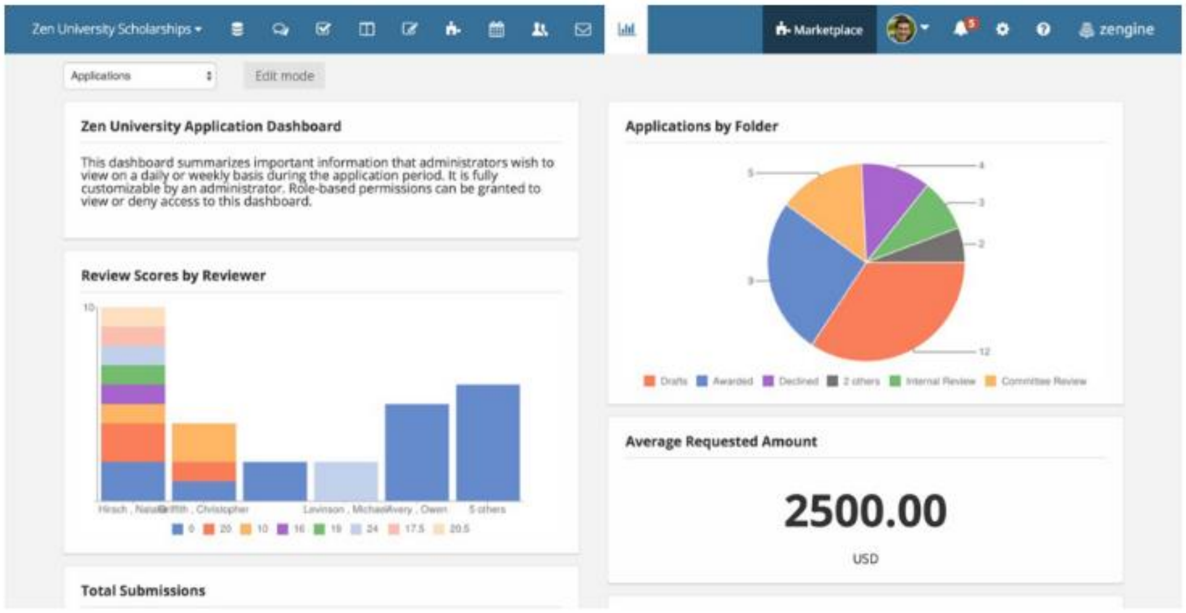
1. Deployment requires little attention from them
2. Updates contain no undesired changes
3. The downtime is properly scheduled

Of these three goals in the context of this project, the most important of these three goals is that the update contains no undesired change. This is very important because the software has not been as widely adopted as hoped and making any uniformed change could further damage its usage since customers generally do not want to change their current way of working (Mantyla & Vanhanen, 2011), and if they must do so, they must be notified. As a result of this, the layout and the existing functionalities have largely gone untouched in this project. The changes that involved the use of a framework have been minimal in the sense that they have been used in pages that present new functionalities to new users, specifically the functionalities for the donors. It is important also that the other customer goals are not ignored. This is because these also play a part in the customers' likelihood of patronizing the software. Silent software deployment may be an ideal way of making sure that installation will require no involvement from a customer however more research is required to know how to use it for this project.

The features that were implemented for the donors required the display of a lot of information on a single page. In order to get guidance on how to organize the information, an existing scholarship management web application was consulted. A picture of this existing solution can be seen in figure 1.1. From this application a few key observations were made. One of these is the minimalist approach employed by the page which avoids distractions and allows users to focus on the information. This was achieved by using muted background colors on the page and reserving the bright colors for the graphs. The page also employs text bolding and font sizes to good effect by highlighting information that is relevant and using larger fonts to ensure that certain information is clear to the reader.

In addition to this, the page also carefully organizes the information. It does so by placing all the relevant information and graphs in individual rectangles. These rectangles are also spaced out carefully to mark a clear demarcation in the information that is being displayed preventing the user from confusing different pieces of information. Also, the rectangles used to display the information are of different sizes which subtly introduces asymmetry and breaks up monotony.

The elements of Wizehive Scholarship Management application were carefully taken into consideration when the user interface was created to provide the needed information as well as provide a good user experience.



This dashboard summarizes important information that administrators wish to view on a daily or weekly basis during the application period. It is fully customizable by an administrator. Role-based permissions can be granted to view or deny access to this dashboard.

Figure 1.1 Wizehive scholarship management solution

1.3 Motivation

The aim of this project is to improve the scholarship management software utilized by the NGO to make its scholarship delivery more efficient. One way of achieving this is to add more functionality to the software to support a wider user base by catering for donors to utilize the software. This will make the organization more accountable and trustworthy which will improve chances of receiving donations and improving the lives of many more children and families. Ultimately, the goal of this project is to help this NGO in empowering children and give them the opportunity to increase their standard of living. The project aims to bring hope to children and their families for a better future.

Chapter 2: Requirements

2.1 Requirements Gathering

Requirements for the application were gathered through my supervisor who was in contact with the organization. We organized weekly meetings in which discussions were held on the features to be implemented. From these discussions, the major users of the features as well as the system and user requirements were identified. The major users of the features were identified to be the donors. The system and user requirements were also gleaned through analysis of the features.

Also, reports from the previous projects were reviewed. This provided a background of the work that had already been done on the software. It also provided indicated the areas of the software that could be improved.

2.2 Application Users

Within the scope of this project, the application users are the donors. The donors' role is to view information pertaining to the particular grant they provide to students.

2.3 Use cases and scenarios

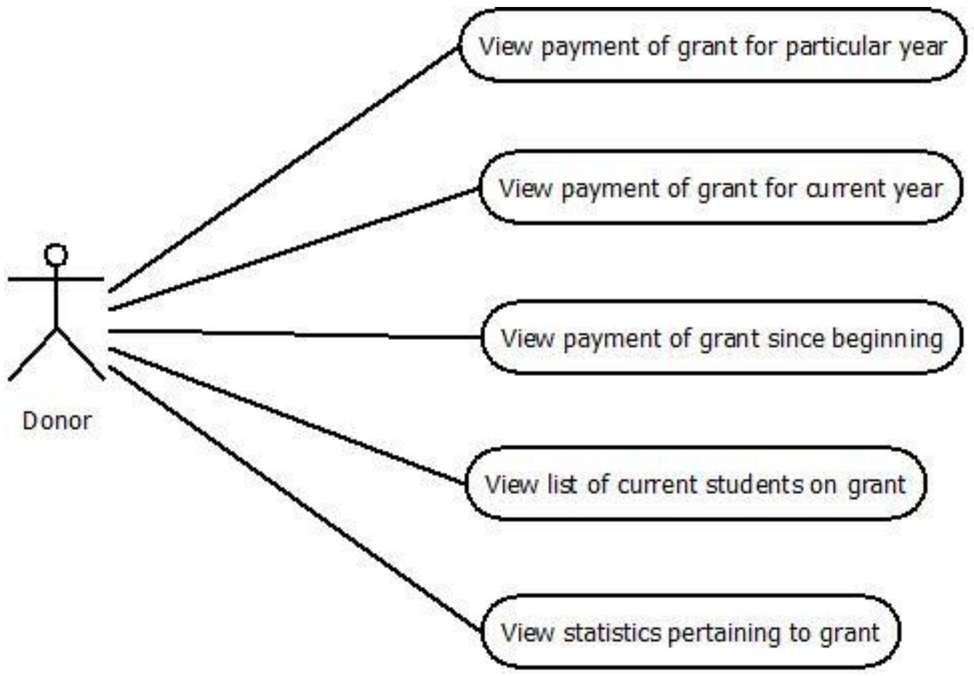

View payment of grant for particular year
View payment of grant for current year
View payment of grant since beginning
View list of current students on grant
View statistics pertaining to grant
Donor

Figure 1.1: Use case diagram for donor

2.4 Details of scenarios from use case

The table below gives further details about the requirement for the donor. It gives a description of what the requirement involves, what data is involved, how the data is generated and displayed. During implementation, this table serves as a guide to make sure that the feature is implemented to meet user requirements. The other tables outlining the other requirements can be found in the appendix.

Table 1.1: Tabular description of view payment of grant for particular year requirement

View Payment of Grant for particular year	
Actor	Donor
Description	Donor can view the amount of money that has been paid for an entire financial year in the past or present
Data	Cost of grant in Ghana Cedis
Stimulus	Option selected by donor
Response	Cost displayed on the screen
Comments	The donor can only view information pertaining to their grant

2.5 System Requirements

The system requirements can be divided into two: the functional requirements and the non-functional requirements.

2.5.1 Functional Requirements

- **The software should provide important information to donors**

The donor should be able to view the necessary information from the database and this information has to be displayed in the appropriate format. The information regarding the payments should be displayed as text to the donor. Information regarding the students should be displayed as a list containing the relevant information. Information about the statistics should be displayed using the appropriate graph so that the information will lend itself to analysis by the user. Bar graphs and pie charts are examples of charts that are to be used.

2.5.2 Non-Functional requirements

Reliability

In the event that graphs are not displayed, possibly due to a failure in internet connection, the information should still be displayed to the user either in text or table format.

Security

The system should ensure that only information pertaining to the grant in question is displayed. Information regarding other grants should not be displayed to donors.

Ease of use

The user interface should be easy to use. Button labels and descriptions should be clear and easy to understand.

Chapter 3: System Architecture

The scholarship management software was built by the previous developers using the 3-tier, layered architecture shown in figure 3.1. This architecture consists of three layers, namely: the user interface, the application and database layers. In order for the application to work effectively, these three layers need to communicate with each other.

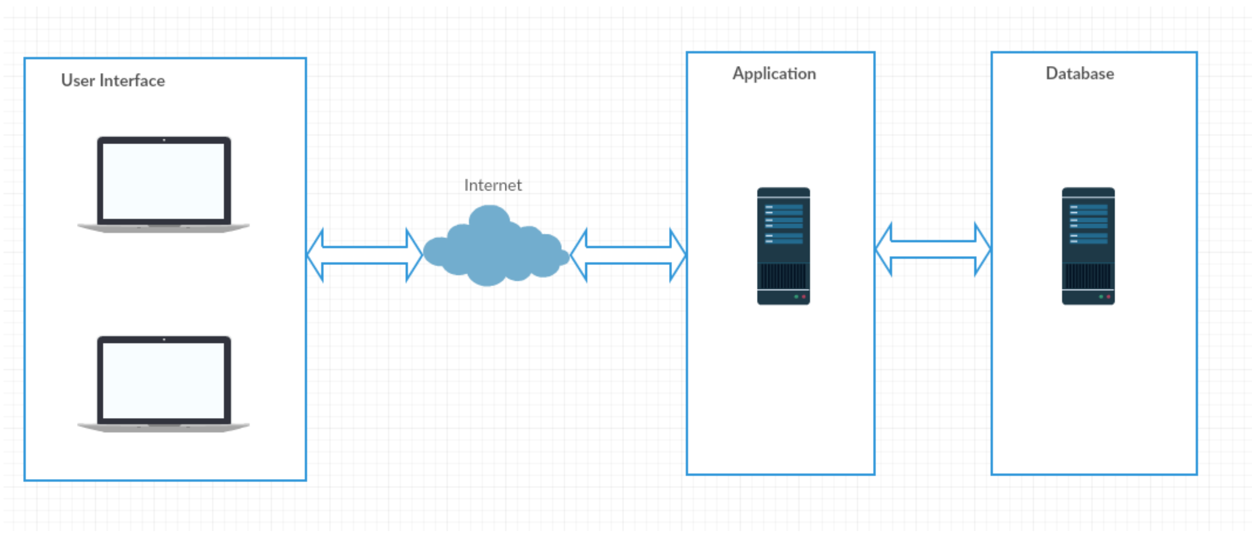


Figure 3.1: System Architecture

3.1 The User Interface Layer

This serves as the layer which allows users to make use of the system's functionalities. For instance, it is at this layer that the donor can view how much they have spent so far on scholarships or the gender distribution for the scholarships that have been awarded. These functionalities are accessed through a web browser such as Google Chrome and Mozilla Firefox. The pages are rendered using HTML, CSS and JavaScript. Information displayed from the screen using Ajax or PHP depending on the need to maintain a smooth user experience. If the information to be displayed did not require the page to be refreshed, then Ajax can be avoided, else if the information requires a page refresh, Ajax is used. Whether via Ajax or not, information is sent to and from the application layer using an HTTP connection. If information could not be retrieved, an error message will be displayed to the user. An example of a file that is used in the user interface is grantdetails.php.

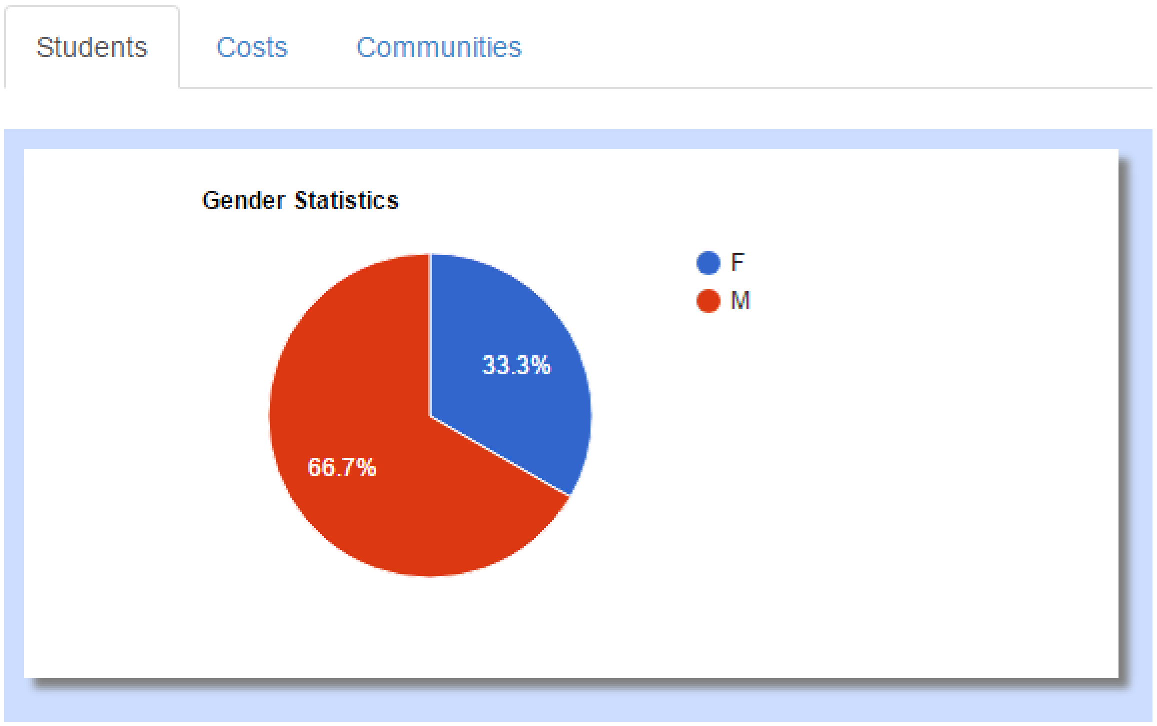


Figure 3.2: User Interface

Figure 3.2 above shows the user interface. The tabs are used to categorize the different sections of information that can be viewed by the user in a web browser. The tabbing was created using the Bootstrap framework, colors were displayed using CSS and graphs were generated using the Google Charts library. The graph shown is an example of a piece of information that is obtained from the database. This information was acquired directly from the server and displayed as a graph.

Ajax was used in order to provide a good user experience. In figure 3.3, the user is presented with a drop down from which he or she can select a financial year to view its associated cost. Ajax was used to prevent the page from requiring a reload before displaying the cost. The effect is that the amount changes seamlessly as the user changes the value in the drop down and an example of this is shown in figure 3.4.

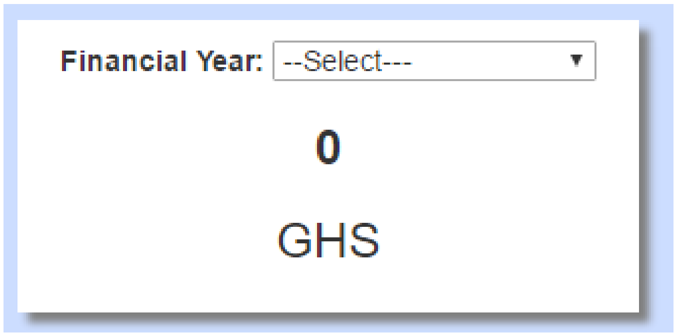


Figure 3.3: Dropdown and payment in particular year

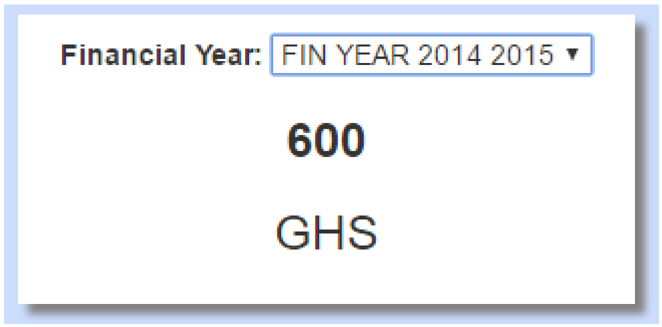
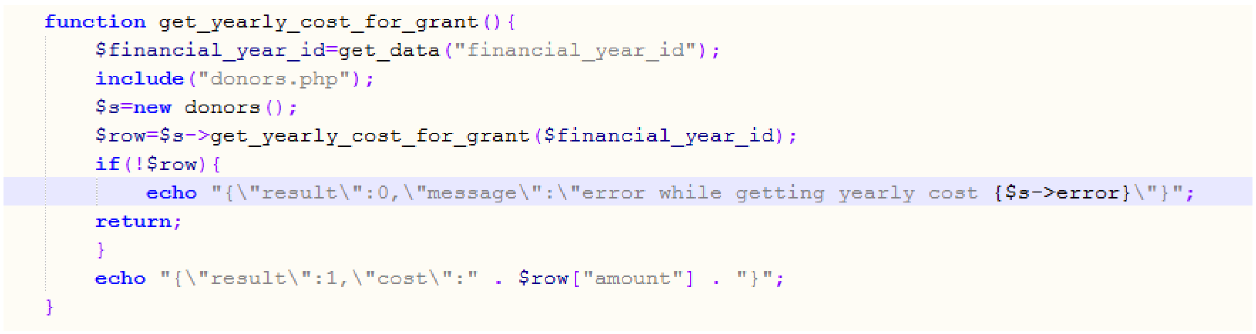


Figure 3.4: Selected year and payment for 2014/2015 financial year

3.2 Application Layer

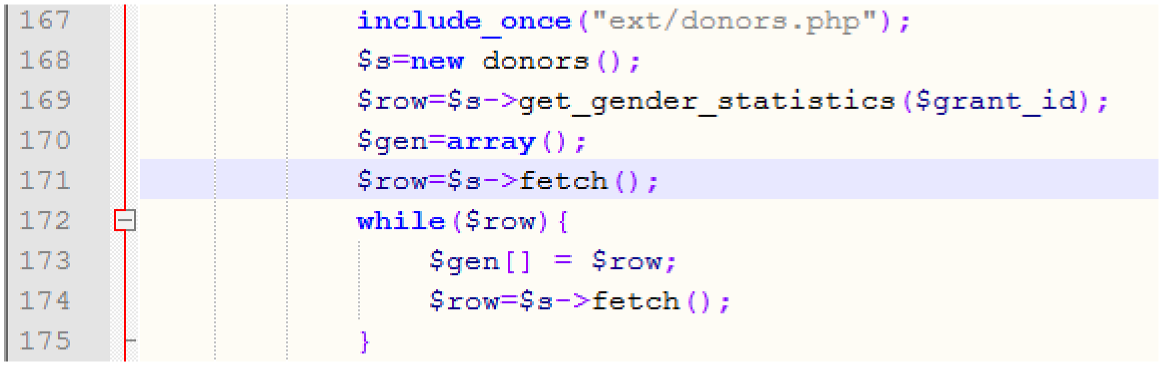
The application layer sits between the data layer and the user interface and contains the business logic of the software. This layer is where the web server operates and PHP functions that produce information based on requests received from the user interface. These functions are what communicate with the data layer to retrieve the information that is needed by the donor, such as, the gender distribution of the sponsored students. The application layer functions may either be client side or on a separate page. When the information retrieved does not require Ajax, the script is written on the client side otherwise it is written in a separate script An example of the file written as a separate script is ajaxgrants.php.



```
function get_yearly_cost_for_grant(){
    $financial_year_id=get_data("financial_year_id");
    include("donors.php");
    $s=new donors();
    $row=$s->get_yearly_cost_for_grant($financial_year_id);
    if(!$row){
        echo "{\"result\":0,\"message\":\"error while getting yearly cost {$s->error}\"}";
    return;
    }
    echo "{\"result\":1,\"cost\":" . $row["amount"] . "}";
}
```

Figure 3.5: Code for displaying payment for a year in JSON

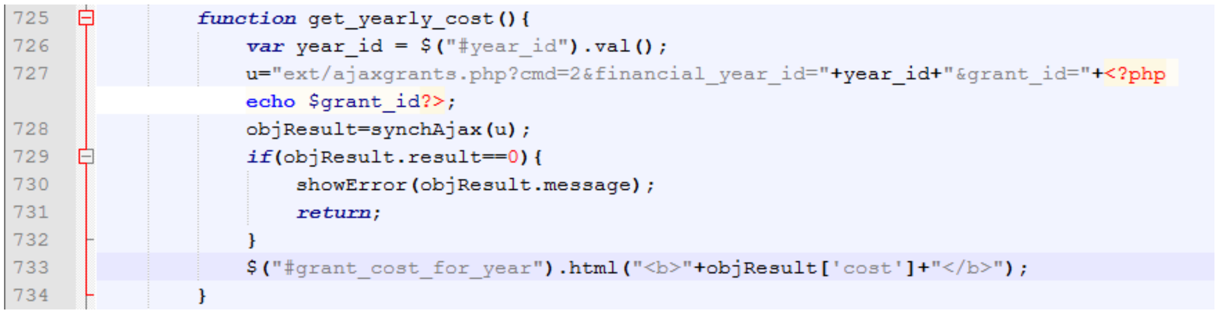
Figure 3.5 shows the code for providing the yearly cost for a particular grant which is displayed using Ajax. The information produced is passed to the browser as a JSON object that is displayed to the user via JQuery as seen in figure 3.7.



```
            include_once("ext/donors.php");
            $s=new donors();
            $row=$s->get_gender_statistics($grant_id);
            $gen=array();
            $row=$s->fetch();
            while($row){
                $gen[] = $row;
                $row=$s->fetch();
            }
```

Figure 3.6: Code showing information retrieval without Ajax

Figure 3.6 above is an example of retrieving information without Ajax. The information is retrieved from the data layer via an application layer function. The function that does this is on line 169.

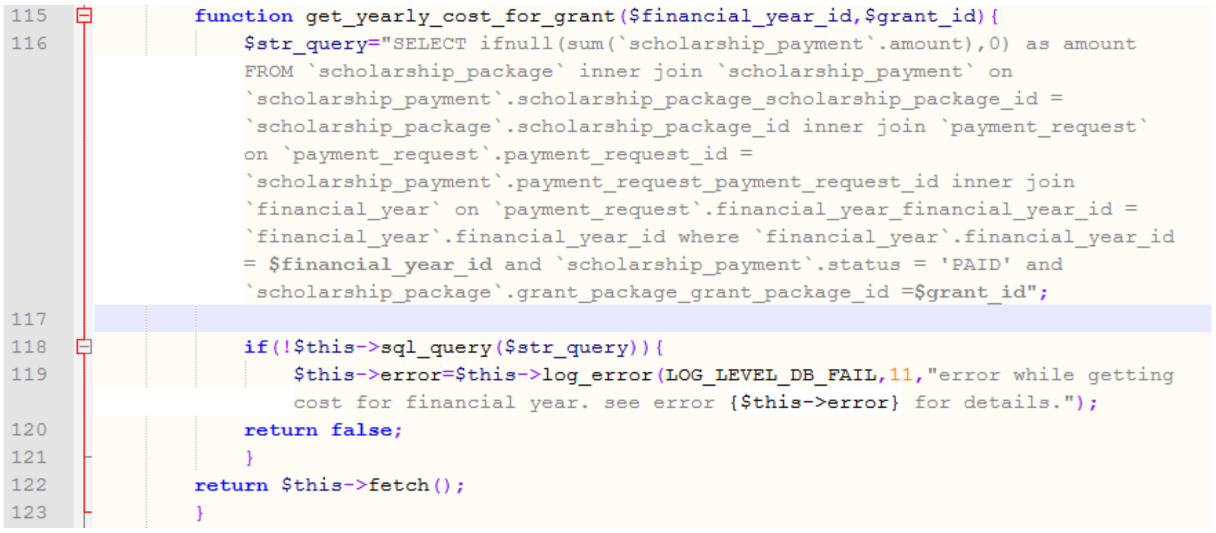


```
        function get_yearly_cost(){
            var year_id = $("#year_id").val();
            u="ext/ajaxgrants.php?cmd=2&financial_year_id="+year_id+"&grant_id="+<?php
            echo $grant_id?>;
            objResult=synchAjax(u);
            if(objResult.result==0){
                showError(objResult.message);
                return;
            }
            $("#grant_cost_for_year").html("<b>"+objResult['cost']+"</b>");
        }
```

Figure 3.7: Code used to retrieve and display information using Ajax

3.3 Data Layer

This layer consists of the MySQL database server that stores all information pertaining to the scholarships, students, schools, communities, etc. Any information that is displayed on the user interface is retrieved from this layer. MySQL is a relational database thus all the information is stored in tables and SQL queries are issued to retrieve information from either one or multiple tables. Figure 3.8 shows a function that exists on the data layer. This capability allows the application to pull information from many different tables to present as one view to the donor.



```php
function get_yearly_cost_for_grant($financial_year_id,$grant_id){
    $str_query="SELECT ifnull(sum(`scholarship_payment`.amount),0) as amount FROM `scholarship_package` inner join `scholarship_payment` on `scholarship_payment`.scholarship_package_scholarship_package_id = `scholarship_package`.scholarship_package_id inner join `payment_request` on `payment_request`.payment_request_id = `scholarship_payment`.payment_request_payment_request_id inner join `financial_year` on `payment_request`.financial_year_financial_year_id = `financial_year`.financial_year_id where `financial_year`.financial_year_id = $financial_year_id and `scholarship_payment`.status = 'PAID' and `scholarship_package`.grant_package_grant_package_id =$grant_id";

    if(!$this->sql_query($str_query)){
        $this->error=$this->log_error(LOG_LEVEL_DB_FAIL,11,"error while getting cost for financial year. see error {$this->error} for details.");
    return false;
    }
return $this->fetch();
}
```

Figure 3.8: Function to retrieve payment for a particular year in data layer

3.3.1 Database Architecture

Figure 3.7 shows the tables from the database that were used. The diagram shows the foreign key relationships between the tables involved. These relationships were essential to getting the right information that was required by the donors.

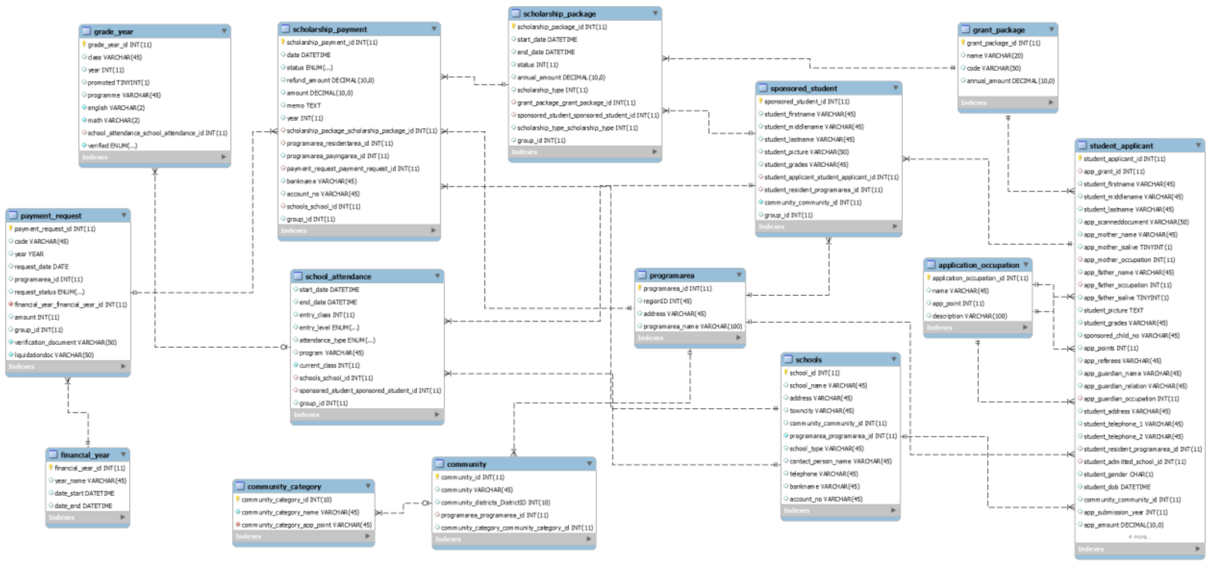

grade_year
grade_year_id INT(11)
class VARCHAR(45)
year INT(11)
promoted TINYINT(1)
programme VARCHAR(45)
english VARCHAR(2)
math VARCHAR(2)
school_attendance_school_attendance_id INT(11)
verified ENUM(...)
Indexes
scholarship_payment
scholarship_payment_id INT(11)
date DATETIME
status ENUM(...)
refund_amount DECIMAL(10,0)
amount DECIMAL(10,0)
memo TEXT
year INT(11)
scholarship_package_scholarship_package_id INT(11)
programarea_residentarea_id INT(11)
programarea_payingarea_id INT(11)
payment_request_payment_request_id INT(11)
bankname VARCHAR(45)
account_no VARCHAR(45)
schools_school_id INT(11)
group_id INT(11)
Indexes
scholarship_package
scholarship_package_id INT(11)
start_date DATETIME
end_date DATETIME
status INT(11)
annual_amount DECIMAL(10,0)
scholarship_type INT(11)
grant_package_grant_package_id INT(11)
sponsored_student_sponsored_student_id INT(11)
scholarship_type_scholarship_type INT(11)
group_id INT(11)
Indexes
grant_package
grant_package_id INT(11)
name VARCHAR(20)
code VARCHAR(50)
annual_amount DECIMAL(10,0)
Indexes
sponsored_student
sponsored_student_id INT(11)
student_firstname VARCHAR(45)
student_middlename VARCHAR(45)
student_lastname VARCHAR(45)
student_picture VARCHAR(50)
student_grades VARCHAR(45)
student_applicant_student_applicant_id INT(11)
student_resident_programarea_id INT(11)
community_community_id INT(11)
group_id INT(11)
Indexes
student_applicant
student_applicant_id INT(11)
app_grant_id INT(11)
student_firstname VARCHAR(45)
student_middlename VARCHAR(45)
student_lastname VARCHAR(45)
app_scanneddocument VARCHAR(50)
app_mother_name VARCHAR(45)
app_mother_isalive TINYINT(1)
app_mother_occupation INT(11)
app_father_name VARCHAR(45)
app_father_occupation INT(11)
app_father_isalive TINYINT(1)
student_picture TEXT
student_grades VARCHAR(45)
sponsored_child_no VARCHAR(45)
app_points INT(11)
app_referees VARCHAR(45)
app_guardian_name VARCHAR(45)
app_guardian_relation VARCHAR(45)
app_guardian_occupation INT(11)
student_address VARCHAR(45)
student_telephone_1 VARCHAR(45)
student_telephone_2 VARCHAR(45)
student_resident_programarea_id INT(11)
student_admitted_school_id INT(11)
student_gender CHAR(1)
student_dob DATETIME
community_community_id INT(11)
app_submission_year INT(11)
app_amount DECIMAL(10,0)
4 more...
Indexes
payment_request
payment_request_id INT(11)
code VARCHAR(45)
year YEAR
request_date DATE
programarea_id INT(11)
request_status ENUM(...)
financial_year_financial_year_id INT(11)
amount INT(11)
group_id INT(11)
verification_document VARCHAR(50)
liquidationdoc VARCHAR(50)
Indexes
school_attendance
start_date DATETIME
end_date DATETIME
entry_class INT(11)
entry_level ENUM(...)
attendance_type ENUM(...)
program VARCHAR(45)
current_class INT(11)
schools_school_id INT(11)
sponsored_student_sponsored_student_id INT(11)
group_id INT(11)
Indexes
programarea
programarea_id INT(11)
regionID INT(45)
address VARCHAR(45)
programarea_name VARCHAR(100)
Indexes
application_occupation
application_occupation_id INT(11)
name VARCHAR(45)
app_point INT(11)
description VARCHAR(100)
Indexes
schools
school_id INT(11)
school_name VARCHAR(45)
address VARCHAR(45)
towncity VARCHAR(45)
community_community_id INT(11)
programarea_programarea_id INT(11)
school_type VARCHAR(45)
contact_person_name VARCHAR(45)
telephone VARCHAR(45)
bankname VARCHAR(45)
account_no VARCHAR(45)
Indexes
financial_year
financial_year_id INT(11)
year_name VARCHAR(45)
date_start DATETIME
date_end DATETIME
Indexes
community_category
community_category_id INT(10)
community_category_name VARCHAR(45)
community_category_app_point VARCHAR(45)
Indexes
community
community_id INT(11)
community VARCHAR(45)
community_districts_DistrictID INT(10)
programarea_programarea_id INT(11)
community_category_community_category_id INT(11)
Indexes

Figure 3.9: Tables from database used for obtaining data

Chapter 4: Implementation and Testing

This chapter delves into the implementation details for the different features. It begins by detailing the different tools, frameworks and libraries that were used, how they were tested and details how features were implemented. Following each feature are the tests that were carried out to ensure that features worked correctly.

4.1 Implementation Tools and Libraries

Google Charts Library

This library was used to display data pulled from the server graphically in order to make analysis and decision-making easier for donors. It provided a breadth of chart types that enable displaying varied data to users. Another advantage is its ease of use, requiring simple JavaScript to make use of them. It also allowed for the customization of charts to better suit websites. It had the added bonus of being cross-platform for different browsers as well as hardware including tablets, phones and desktops.

Bootstrap Framework

Bootstrap is a framework that is useful in building responsive features in HTML, JavaScript and CSS (“Bootstrap,” n.d.). This framework was used in the implementation of the donors’ features because it provided the tools create a good user experience.

PhpMyAdmin and MySQL

PhpMyAdmin served as the web-based client for the administration of MySQL server. It allowed for the basic functionality of database management including inserting, deleting and updating data. This proved vital in both implementing the features and testing them. It was the preferred option for database administration because it is web-based and has a user-friendly interface making the implementation of the features easier. MySQL was the database server that was used to manage the data stored in the database. Since MySQL was used in the development of the previous features, it was more convenient to continue using as rather than using another which could potentially cause compatibility issues.

XAMPP

This is a PHP development environment conveniently comes packaged with PHP, MySQL and the Apache web server making development of web applications simpler. It also comes with a control panel that can be used to switch services on and off with one click.

PHP

This is the server-side scripting language that was used to retrieve data from the database and passed to the application layer. It was also used for coding the application logic. Despite being free, PHP was used primarily because the previous developers used it and it served its purpose of information storage, retrieval and production of dynamic content adequately. In addition to this, its wide usage ensured that there was enough support whenever problems were encountered.

Apache HTTP Server

The Apache HTTP web server is an open-source HTTP server for modern operating systems. For this project, it served as the main facilitator between the application and database layer by allowing communication through HTTP. Apache was used because it was used in the previous development and also because it comes bundled with XAMPP.

HTML

Hypertext Markup Language was used for the creation of the web pages displayed the relevant information on the client's browser. It was responsible for displaying the information gathered from the database. Tables, containers, buttons and text were displayed using HTML.

CSS and JQuery

Cascading Style Sheets were used to add styling to web pages written in HTML. Styling was added in order to improve the application's usability. Different colours, text font, shadow effects and positioning of elements were used to create a more pleasing visual experience and improve user experience.

JQuery is a JavaScript library that simplifies the use of JavaScript in client-side scripting ("Jquery.org", n.d.). JQuery was used to make calling HTML objects easier. It has the benefit of working across multiple web browsers.

JavaScript

JavaScript is a client-side scripting language that is used to create dynamic content of otherwise static HTML pages.

4.1.1 Testing of tools, libraries and frameworks

Table 4.1 the framework, library or tool that was tested, the test procedure and the outcome of the test. In most of these tests the result was displayed visually. In those situations the figures that display the result are stated and can be found in the appendix.

Table 4.1: Table showing tests and results for frameworks, tools and libraries

Framework / Library	Test	Result
Google Charts Library	Display line chart, bar chart and pie chart with consistent sample data. Test shown in figure 4.1.1 and 4.1.2	Line chart, bar chart and pie chart displayed as expected as shown in figure 4.1.3
Bootstrap Framework	Use tabbing to display different content. Test code is shown in figure 4.1.4	Different information displayed as different tabs were navigated. Results can be seen in figures 4.1.5, 4.1.6 and 4.1.7
PhpMyAdmin and MySQL[1]	Use select statement to view a list of academic years in database. SQL statement is: SELECT * FROM \`regions\`	List of regions produced as expected. Results shown in figure 4.1.8
XAMPP and Apache HTTP Server[2]	Navigate to the web server home page using a web browser. The URL is: http://localhost:1234/	Web server successfully displayed as seen in Figure 4.1.9
PHP	Create a script to display text in a browser.	Browser displays text as written in the test script.
HTML	Create a page that displays text in a web browser.	Browser displays test as written in the test page.
CSS and JQuery	Create an HTML page that contains three <p> elements. Two element have their background and font colours modified using CSS and JQuery. The script can be seen in figure 4.1.10	Two of the three <p> elements' background and font colors are successfully changed using CSS and JQuery. Figure 4.1.11 shows the result

[1] - This test proves the functionality of both MySQL and PhpMyAdmin. PhpMyAdmin successfully executed the query. The select statement produced the expected information from database

[2] – The testing of XAMPP and Apache Web Server were bundled due to convenience.

The sections below are the details for the implementation based on the system and user requirements described in chapter two.

4.2 Implementation of Application Features

4.2.1 Viewing Grant Cost for Particular Year

When the page is loaded, the donor selects the costs tab to view all information pertaining to costs. When the donor logs in, the page stores the grant id of the grant package associated with the donor. This id is used to display all the information that is relevant to the particular grant. The donor sees a dialog box as shown in the figure 4.1 and chooses the financial year that he or she wants to view the cost of. Once the donor selects the financial year, the year's id is retrieved to the application layer using jQuery and passed to the application layer using Ajax as shown in figure 4.2.

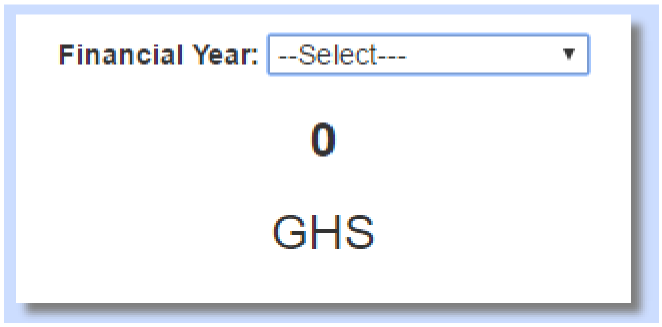
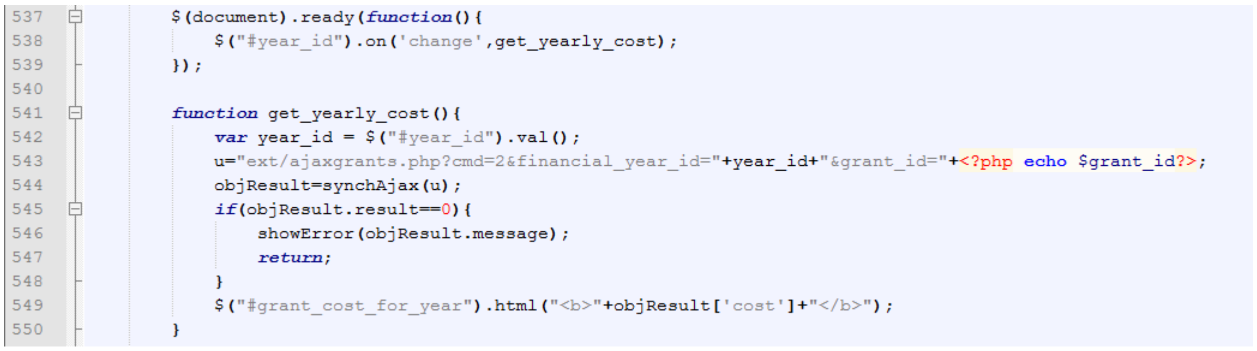


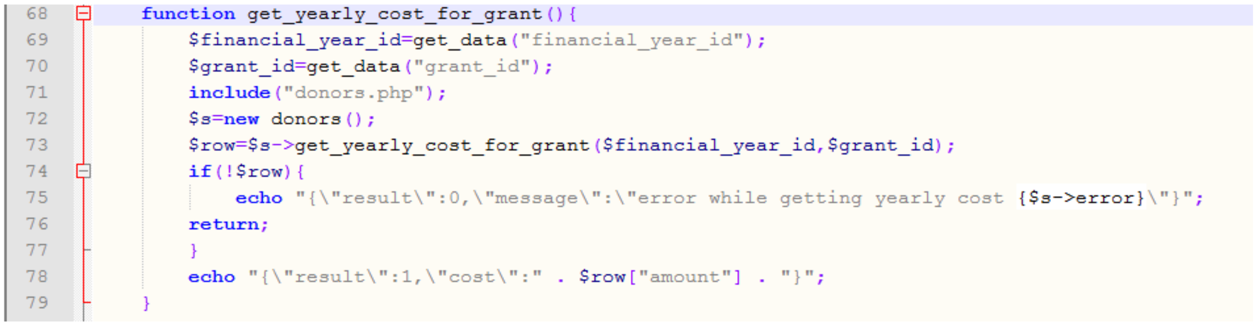
Figure 4.1: Dialog box to view payment for a particular year



```
$(document).ready(function(){
    $("#year_id").on('change',get_yearly_cost);
});

function get_yearly_cost(){
    var year_id = $("#year_id").val();
    u="ext/ajaxgrants.php?cmd=2&financial_year_id="+year_id+"&grant_id="+<?php echo $grant_id?>;
    objResult=synchAjax(u);
    if(objResult.result==0){
        showError(objResult.message);
        return;
    }
    $("#grant_cost_for_year").html("<b>"+objResult['cost']+"</b>");
}
```

Figure 4.2: Value from dropdown passed to application layer using JQuery

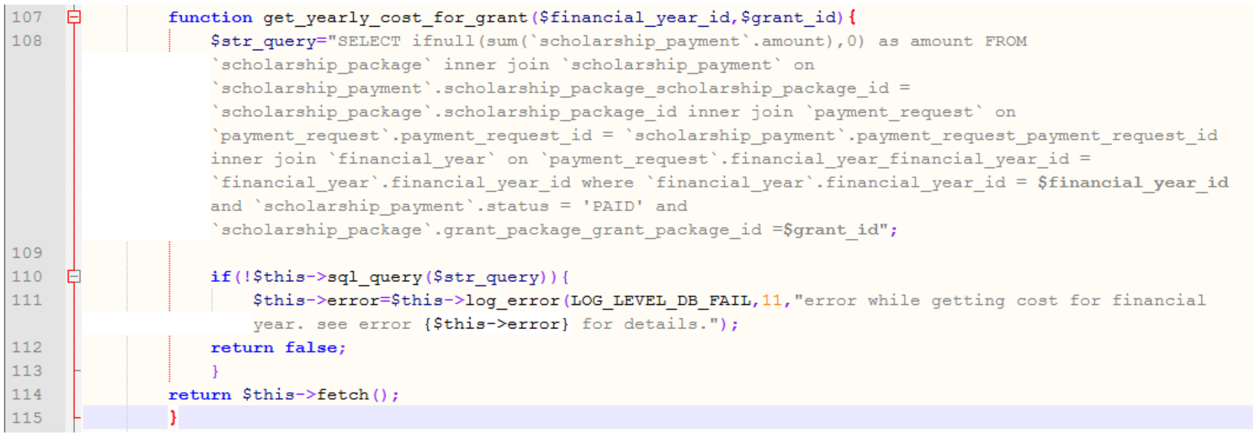
On the application layer page ajaxgrants.php, the appropriate function is selected using a select case from a parameter passed in the URL. In this case, the function in figure 4.3 is used to retrieve the data:



```
function get_yearly_cost_for_grant(){
    $financial_year_id=get_data("financial_year_id");
    $grant_id=get_data("grant_id");
    include("donors.php");
    $s=new donors();
    $row=$s->get_yearly_cost_for_grant($financial_year_id,$grant_id);
    if(!$row){
        echo "{\"result\":0,\"message\":\"error while getting yearly cost {$s->error}\"}";
    return;
    }
    echo "{\"result\":1,\"cost\":" . $row["amount"] . "}";
}
```

Figure 4.3: Application layer function

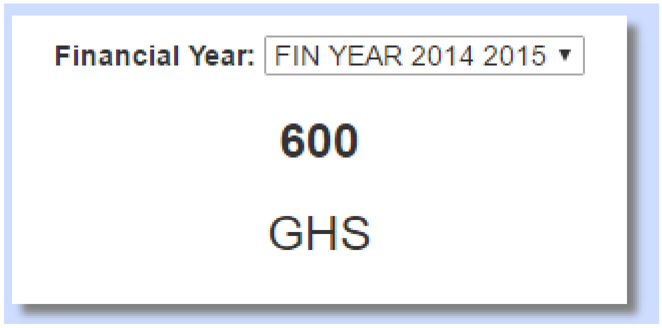
This function retrieves the financial year id and grant id from the URL and passes them to the data layer function, get_yearly_cost_for_grant, as seen in figure 4.4. This function inserts the ids into an SQL query that is used to retrieve the information from the SQL server.



```
function get_yearly_cost_for_grant($financial_year_id,$grant_id){
    $str_query="SELECT ifnull(sum(`scholarship_payment`.amount),0) as amount FROM
    `scholarship_package` inner join `scholarship_payment` on
    `scholarship_payment`.scholarship_package_scholarship_package_id =
    `scholarship_package`.scholarship_package_id inner join `payment_request` on
    `payment_request`.payment_request_id = `scholarship_payment`.payment_request_payment_request_id
    inner join `financial_year` on `payment_request`.financial_year_financial_year_id =
    `financial_year`.financial_year_id where `financial_year`.financial_year_id = $financial_year_id
    and `scholarship_payment`.status = 'PAID' and
    `scholarship_package`.grant_package_grant_package_id =$grant_id";

    if(!$this->sql_query($str_query)){
        $this->error=$this->log_error(LOG_LEVEL_DB_FAIL,11,"error while getting cost for financial
        year. see error {$this->error} for details.");
    return false;
    }
return $this->fetch();
}
```

Figure 4.4: Data layer function containing SQL to retrieve the cost

The end result is shown in figure 4.5 where the cost for the financial year is shown. In this instance, the financial year selected is that of 2014 to 2015 and the cost for that year was 600 Ghana Cedis:


Financial Year: FIN YEAR 2014 2015 ▾

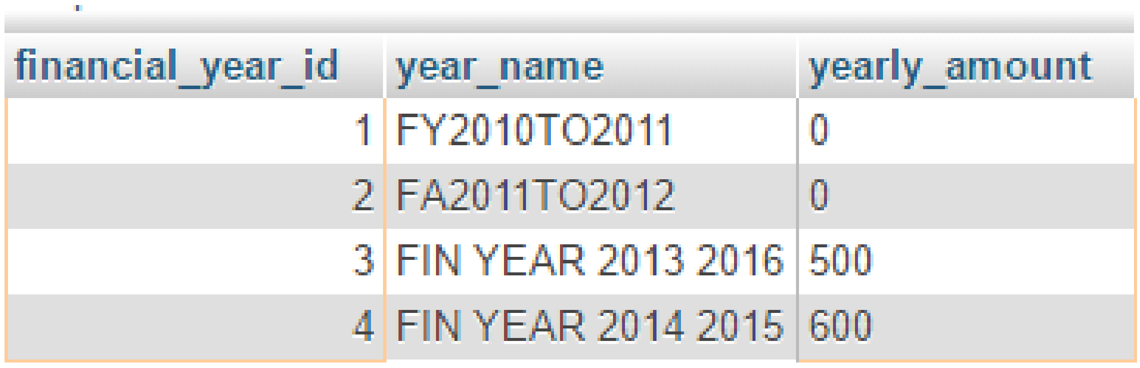
600

GHS

Figure 4.5: Updated dialog box containing payment for 2014/2015 financial year

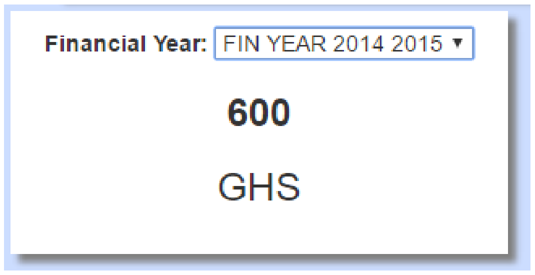
Testing to view grant for a particular financial year

In order to test this feature, a query was written to display all the amounts paid for a grant and matched against the result produced on the user interface. Figure 4.6 shows the result of the query that was created. Figures 4.7 to 4.10 show the result of the user's input on the user interface.



financial_year_id	year_name	yearly_amount
1	FY2010TO2011	0
2	FA2011TO2012	0
3	FIN YEAR 2013 2016	500
4	FIN YEAR 2014 2015	600

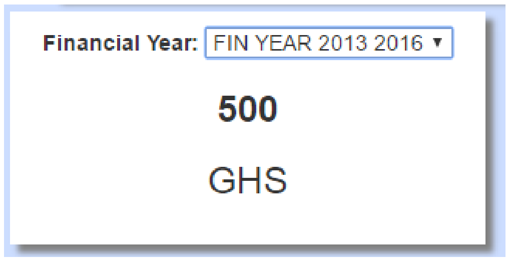
Figure 4.6: SQL showing the financial year ids, financial years and amounts retrieved from the database


Financial Year: FIN YEAR 2014 2015

600

GHS

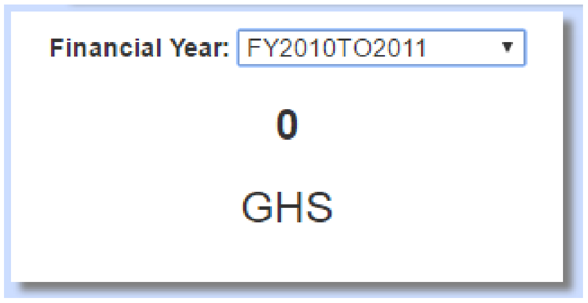
Figure 4.7: Dialog showing cost for 2014/2015


Financial Year: FIN YEAR 2013 2016

500

GHS

Figure 4.8: Dialog showing cost for 2013/2016


Financial Year: FY2010TO2011

0

GHS

Figure 4.9: Dialog showing cost for 2013/2016

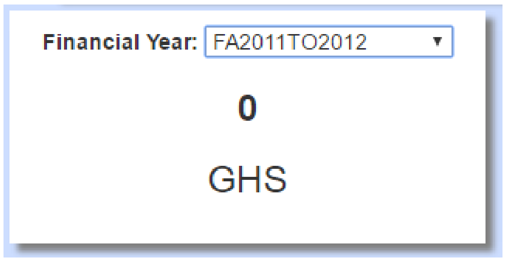
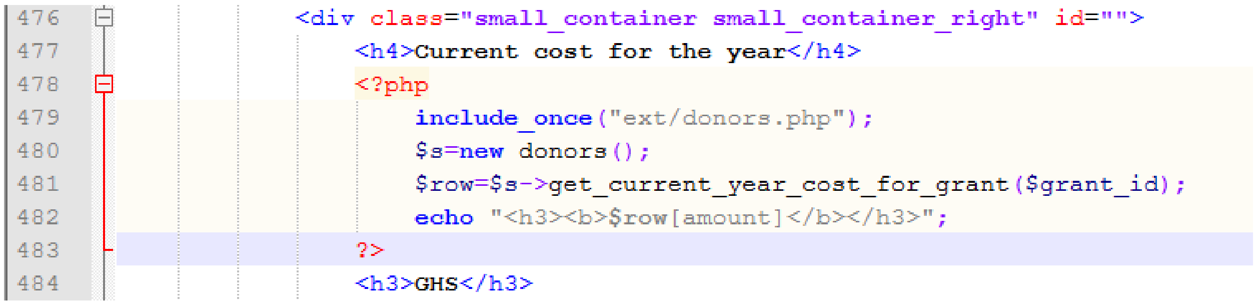


Figure 4.10: Dialog showing cost for 2011/2012

It should be noted that the years were input for testing purposes thus the years shown do not represent the actual years in the organization's database. From these diagrams it can be seen that the costs displayed with the different values of the drop down are consistent with those of the database meaning that this test was successful.

4.2.2 Viewing Grant Cost for Current Year

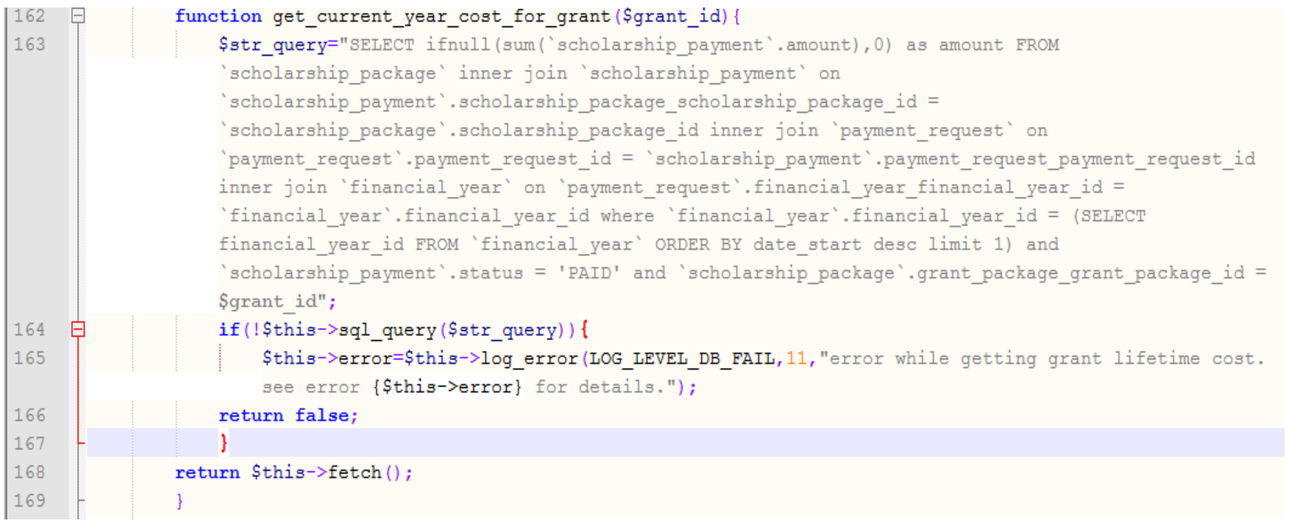
This information is displayed once the donor logs into the donor page. When the donor logs in, the page stores the grant id of the grant package associated with the donor. This id is used to display all the information. The id is passed to the application layer which inserts the id in a function, get_current_year_cost_for_grant:



```
<div class="small_container small_container_right" id="">
    <h4>Current cost for the year</h4>
    <?php
        include_once("ext/donors.php");
        $s=new donors();
        $row=$s->get_current_year_cost_for_grant($grant_id);
        echo "<h3><b>$row[amount]</b></h3>";
    ?>
    <h3>GHS</h3>
```

Figure 4.11: Application layer function to get payment for current year

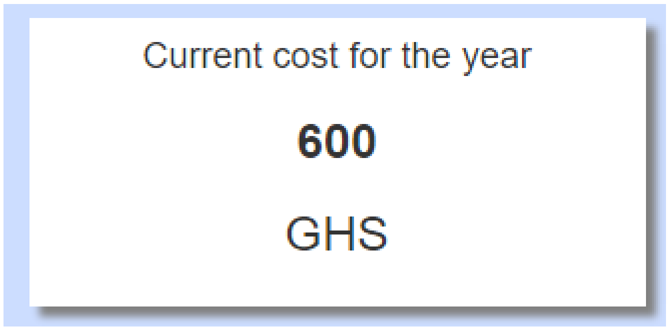
This function contains SQL that is used to retrieve the cost from the database, as seen in figure 4.12:



```
function get_current_year_cost_for_grant($grant_id){
    $str_query="SELECT ifnull(sum(`scholarship_payment`.amount),0) as amount FROM
    `scholarship_package` inner join `scholarship_payment` on
    `scholarship_payment`.scholarship_package_scholarship_package_id =
    `scholarship_package`.scholarship_package_id inner join `payment_request` on
    `payment_request`.payment_request_id = `scholarship_payment`.payment_request_payment_request_id
    inner join `financial_year` on `payment_request`.financial_year_financial_year_id =
    `financial_year`.financial_year_id where `financial_year`.financial_year_id = (SELECT
    financial_year_id FROM `financial_year` ORDER BY date_start desc limit 1) and
    `scholarship_payment`.status = 'PAID' and `scholarship_package`.grant_package_grant_package_id =
    $grant_id";
    if(!$this->sql_query($str_query)){
        $this->error=$this->log_error(LOG_LEVEL_DB_FAIL,11,"error while getting grant lifetime cost.
        see error {$this->error} for details.");
    return false;
    }
return $this->fetch();
}
```

Figure 4.12: Data layer function containing SQL

The information is displayed in the browser as shown below:



Current cost for the year

600

GHS

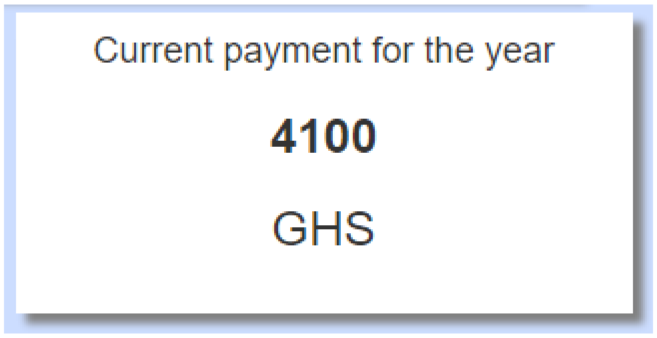
Figure 4.13: User interface dialog box displaying current year's payment

Testing for viewing grant cost for current year

To test this feature, the SQL used for the function in figure 4.14 was used to obtain information directly from the database and the result displayed as shown in figure 4.15. It can be seen that the result of the query is consistent with what is displayed on the user interface confirming that the feature works and the test was successful.

```
SELECT ifnull(sum(`scholarship_payment`.amount),0) as amount FROM
`scholarship_package` inner join `scholarship_payment` on
`scholarship_payment`.scholarship_package_scholarship_package_id =
`scholarship_package`.scholarship_package_id inner join
`payment_request` on `payment_request`.payment_request_id =
`scholarship_payment`.payment_request_payment_request_id inner
join `financial_year` on
`payment_request`.financial_year_financial_year_id =
`financial_year`.financial_year_id where
`financial_year`.financial_year_id = $financial_year_id and
`scholarship_payment`.status = 'PAID' and
`scholarship_package`.grant_package_grant_package_id = 5
```

Figure 4.14: SQL for obtaining payment for current year for grant with id 5


Current payment for the year

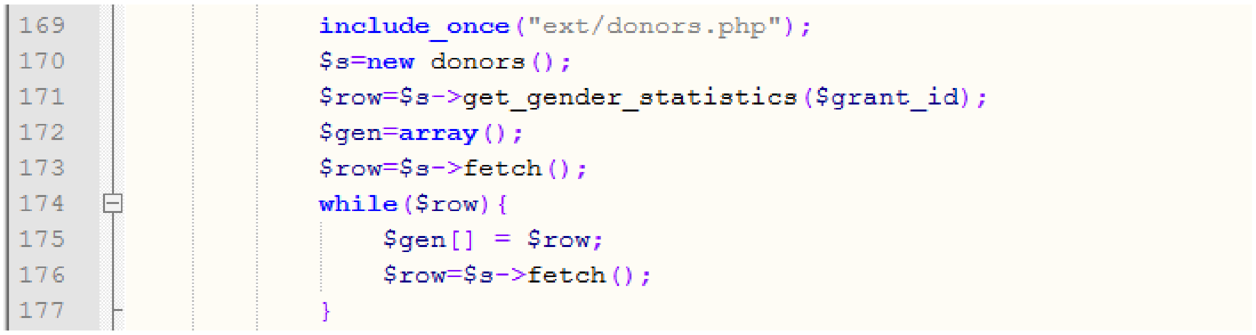
4100

GHS

Figure 4.15: Updated dialog box showing payment for current year

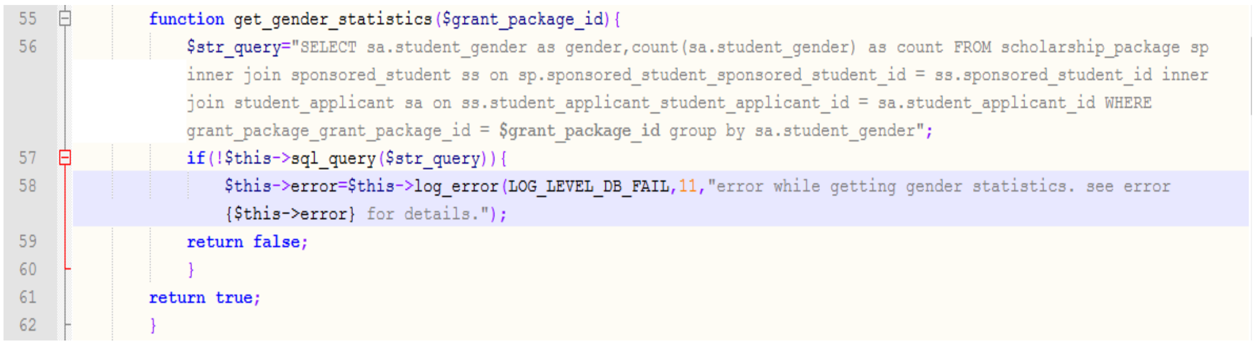
4.2.3 Viewing Gender Statistics for Grant

This information is displayed once the donor logs into the donor page. When the donor logs in, the page stores the grant id of the grant package associated with the donor. This id is used to display information relating to the gender. The id is passed to the application layer which inserts the id in a function, get_gender_statistics, which can be found in the data layer. Figure 4.16 shows the application layer code that passes the id to the data layer. Figure 4.17 shows the get_gender_statistics that can be found in the data layer.



```
include_once("ext/donors.php");
$s=new donors();
$row=$s->get_gender_statistics($grant_id);
$gen=array();
$row=$s->fetch();
while($row){
    $gen[] = $row;
    $row=$s->fetch();
}
```

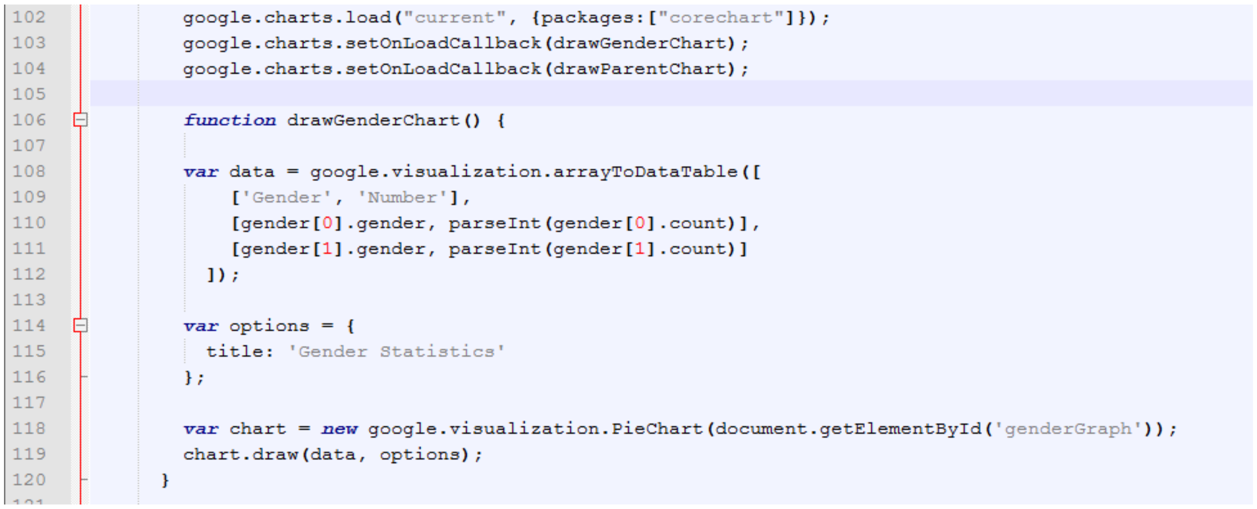
Figure 4.16: Application layer code to pass grant id to data layer



```
function get_gender_statistics($grant_package_id){
    $str_query="SELECT sa.student_gender as gender,count(sa.student_gender) as count FROM scholarship_package sp
    inner join sponsored_student ss on sp.sponsored_student_sponsored_student_id = ss.sponsored_student_id inner
    join student_applicant sa on ss.student_applicant_student_applicant_id = sa.student_applicant_id WHERE
    grant_package_grant_package_id = $grant_package_id group by sa.student_gender";
    if(!$this->sql_query($str_query)){
        $this->error=$this->log_error(LOG_LEVEL_DB_FAIL,11,"error while getting gender statistics. see error
        {$this->error} for details.");
    return false;
    }
return true;
}
```

Figure 4.17: Data layer code containing SQL for retrieving information on gender

The function exists in the data layer and passes the grant id into an SQL query which retrieves the information as an array. This array servers as the data that is used to generate the graph using the Google Charts library. Once the charts library is loaded, the data is passed to it, and the options for the graph are set. Finally, the id for the container for the graph is retrieved using jQuery and the graph is drawn:



```
102         google.charts.load("current", {packages:["corechart"]});
103         google.charts.setOnLoadCallback(drawGenderChart);
104         google.charts.setOnLoadCallback(drawParentChart);
105
106         function drawGenderChart() {
107
108         var data = google.visualization.arrayToDataTable([
109             ['Gender', 'Number'],
110             [gender[0].gender, parseInt(gender[0].count)],
111             [gender[1].gender, parseInt(gender[1].count)]
112           ]);
113
114         var options = {
115           title: 'Gender Statistics'
116         };
117
118         var chart = new google.visualization.PieChart(document.getElementById('genderGraph'));
119         chart.draw(data, options);
120       }
```

Figure 4.18: User interface code to draw chart. Gender variable contains two arrays, each identifying the gender and the count

In figure 4.18 the gender and the count is contained in the gender array as seen on line 110 and 111. The outcome of above code is seen figure 4.19 shown below:

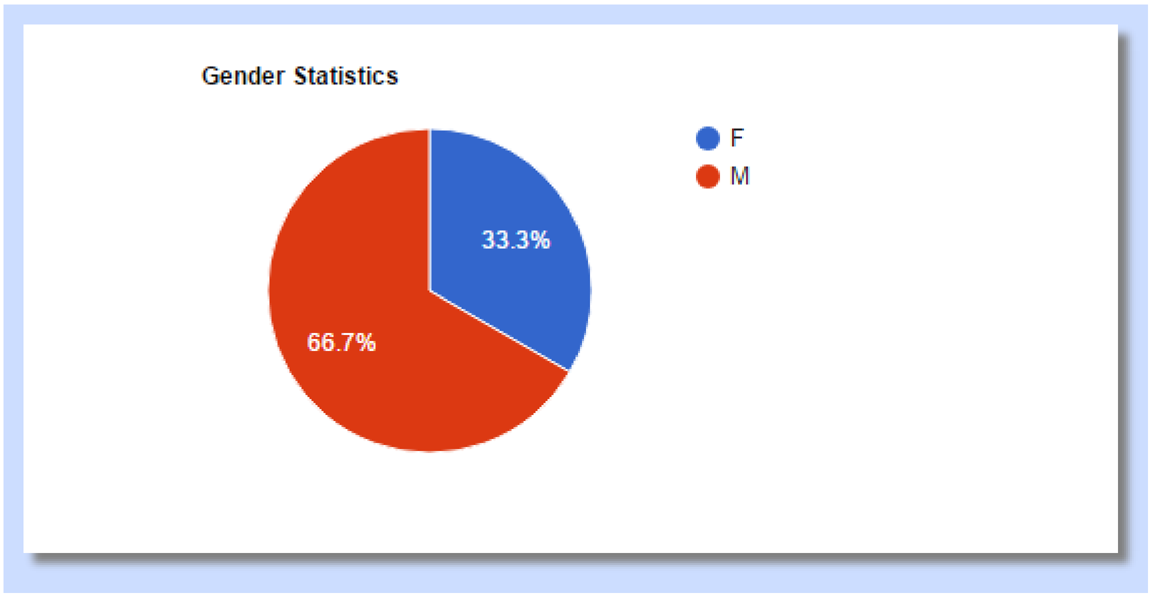
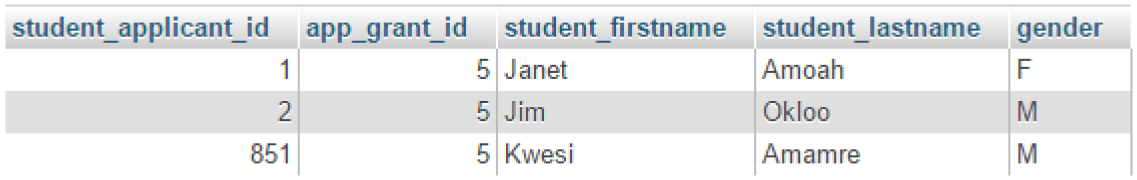


Figure 4.19: Graph showing gender distribution for grant in question

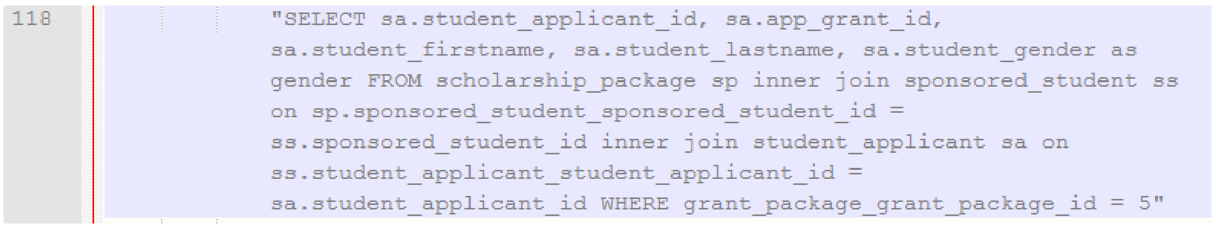
Testing for gender statistics

To test this feature, an SQL query was written directly into the database to generate the students, together with some of their details, whose fees are paid with a particular grant. This query is shown in figure 4.21. In this case the grant id that was used was the grant number 5. The result of the query is shown in figure 4.20 below:



student_applicant_id	app_grant_id	student_firstname	student_lastname	gender
1	5	Janet	Amoah	F
2	5	Jim	Okloo	M
851	5	Kwesi	Amamre	M

Figure 4.20: Result of test SQL query to get information about sponsored students and their gender



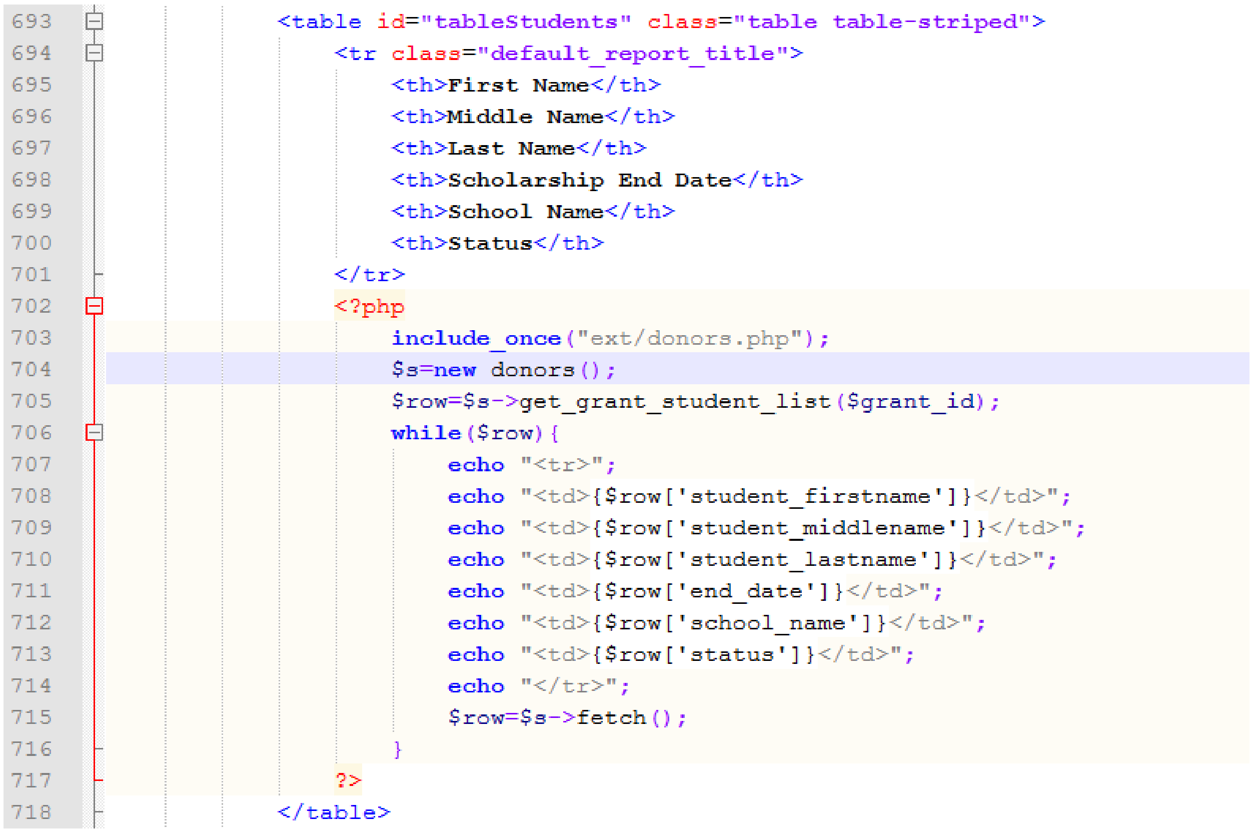
```
"SELECT sa.student_applicant_id, sa.app_grant_id,
sa.student_firstname, sa.student_lastname, sa.student_gender as
gender FROM scholarship_package sp inner join sponsored_student ss
on sp.sponsored_student_sponsored_student_id =
ss.sponsored_student_id inner join student_applicant sa on
ss.student_applicant_student_applicant_id =
sa.student_applicant_id WHERE grant_package_grant_package_id = 5"
```

Figure 4.21: Test SQL query to generate information about sponsored students

The query returns three students, two of who are male and one female. The percentages calculated based from these numbers are very similar to those that are displayed in the graph. From this result, we can see that the graph displays the correct information thus making our test a success.

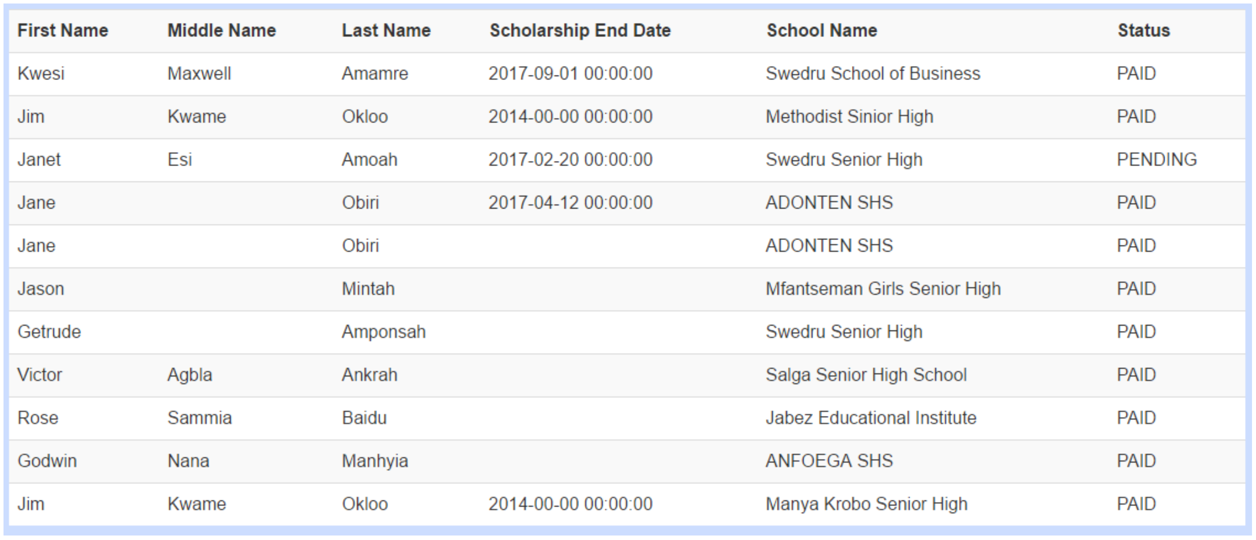
4.2.4 Viewing List of Students for Grant

This information is displayed once the donor logs into the donor page. When the donor logs in, the page stores the grant id of the grant package associated with the donor. This id is used to display the list of students who are current beneficiaries of the grant. The id is passed to the application layer which then passes it on to the data layer. The data layer retrieves the information via the get_grant_student_list function. Once obtained, this information is displayed in the HTML table styled with Bootstrap. Figures 4.22 and 4.23 show the code and the table output respectively.



```
<table id="tableStudents" class="table table-striped">
    <tr class="default_report_title">
        <th>First Name</th>
        <th>Middle Name</th>
        <th>Last Name</th>
        <th>Scholarship End Date</th>
        <th>School Name</th>
        <th>Status</th>
    </tr>
    <?php
        include_once("ext/donors.php");
        $s=new donors();
        $row=$s->get_grant_student_list($grant_id);
        while($row){
            echo "<tr>";
            echo "<td>{$row['student_firstname']}</td>";
            echo "<td>{$row['student_middlename']}</td>";
            echo "<td>{$row['student_lastname']}</td>";
            echo "<td>{$row['end_date']}</td>";
            echo "<td>{$row['school_name']}</td>";
            echo "<td>{$row['status']}</td>";
            echo "</tr>";
            $row=$s->fetch();
        }
    ?>
</table>
```

Figure 4.22: User interface layer code used to display information about sponsored students



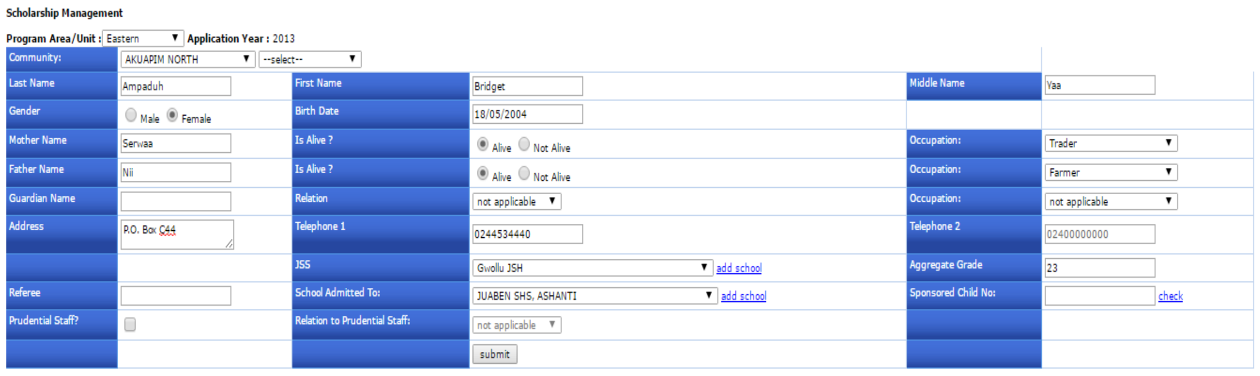
First Name	Middle Name	Last Name	Scholarship End Date	School Name	Status
Kwesi	Maxwell	Amamre	2017-09-01 00:00:00	Swedru School of Business	PAID
Jim	Kwame	Okloo	2014-00-00 00:00:00	Methodist Sinior High	PAID
Janet	Esi	Amoah	2017-02-20 00:00:00	Swedru Senior High	PENDING
Jane		Obiri	2017-04-12 00:00:00	ADONTEN SHS	PAID
Jane		Obiri		ADONTEN SHS	PAID
Jason		Mintah		Mfantseman Girls Senior High	PAID
Getrude		Amponsah		Swedru Senior High	PAID
Victor	Agbla	Ankrah		Salga Senior High School	PAID
Rose	Sammia	Baidu		Jabez Educational Institute	PAID
Godwin	Nana	Manhyia		ANFOEGA SHS	PAID
Jim	Kwame	Okloo	2014-00-00 00:00:00	Manya Krobo Senior High	PAID

Figure 4.23: Table of sponsored students

It is important to note that the scholarship end dates will ordinarily not be blank. They are blank only for testing purposes.

Testing for viewing list of students for grant

In order to test this feature, a student was added to the database under a specific grant. This required some data to be added to the database. Consequently, a new student was added as a student applicant before being added as a sponsored student under the grant. A scholarship package was created for that student to provide details regarding the scholarship and making an entry into the scholarship payment table which provides details of the payment for the particular scholarship package such as the amount. Once this data is entered, the student’s information was displayed in addition to the other students. Figure 4.24 shows the details of the student entered through the software.



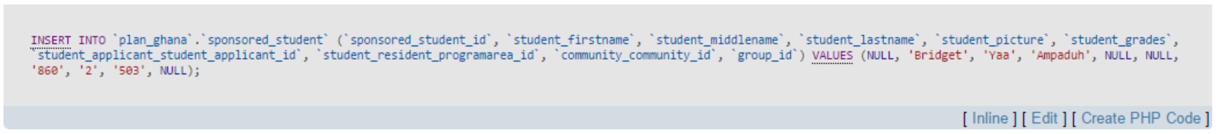
Scholarship Management

Program Area/Unit : Eastern Application Year : 2013

Community:	AKUAPIM NORTH --select--				
Last Name	Ampaduh	First Name	Bridget	Middle Name	Yaa
Gender	Male [✓ Female]	Birth Date	18/05/2004		
Mother Name	Serwaa	Is Alive ?	[✓ Alive] Not Alive	Occupation:	Trader
Father Name	Nii	Is Alive ?	[✓ Alive] Not Alive	Occupation:	Farmer
Guardian Name		Relation	not applicable	Occupation:	not applicable
Address	P.O. Box C44	Telephone 1	0244534440	Telephone 2	02400000000
		JSS	Gwollu JSH add school	Aggregate Grade	23
Referee		School Admitted To:	JUABEN SHS, ASHANTI add school	Sponsored Child No:	check
Prudential Staff?	☐	Relation to Prudential Staff:	not applicable		
			submit		

Figure 4.24: Details of newly entered student applicant

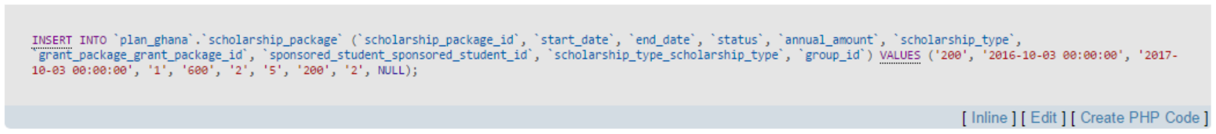
The student was then added into the sponsored student table. This was done using the SQL query in figure 4.25.



```
INSERT INTO `plan_ghana`.`sponsored_student` (`sponsored_student_id`, `student_firstname`, `student_middlename`, `student_lastname`, `student_picture`, `student_grades`,
`student_applicant_student_applicant_id`, `student_resident_programarea_id`, `community_community_id`, `group_id`) VALUES (NULL, 'Bridget', 'Yaa', 'Ampaduh', NULL, NULL,
'860', '2', '503', NULL);
```

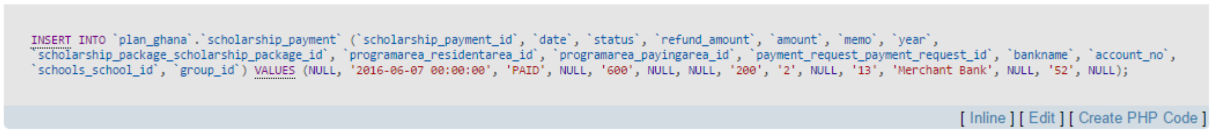
Figure 4.25: Student added as a sponsored student

Next, the SQL query shown in figure 4.26 was used to add a scholarship package pertaining to the student. Finally, the SQL query in figure 4.27 was used to add a scholarship payment on behalf of the student. Now, when the user interface is loaded, the student's details are added to the original table from figure 4.23. The updated table can be seen in figure 4.28.



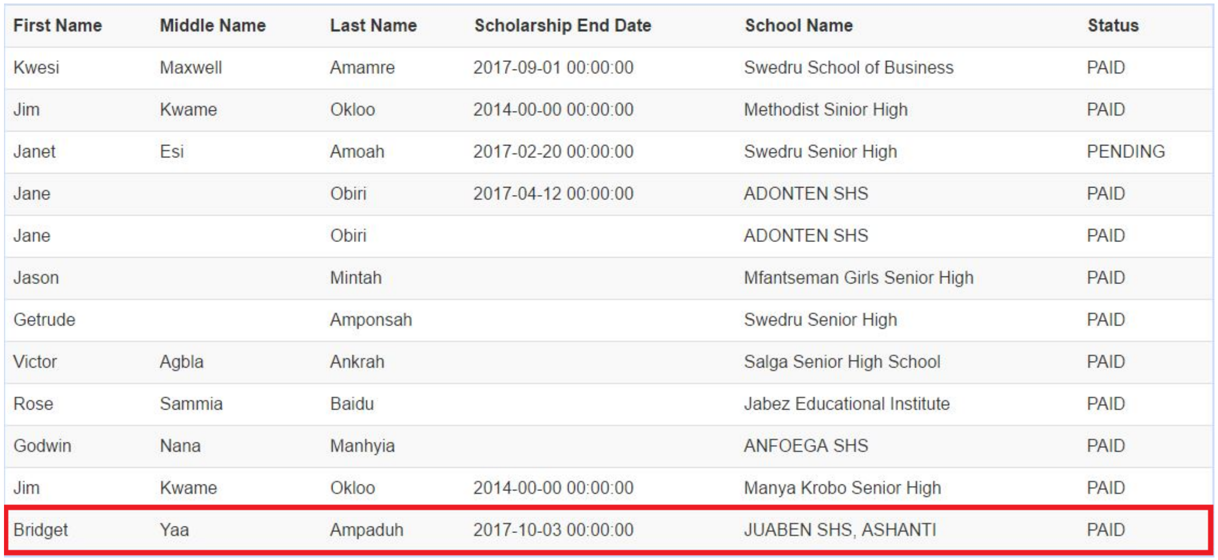
```
INSERT INTO `plan_ghana`.`scholarship_package` (`scholarship_package_id`, `start_date`, `end_date`, `status`, `annual_amount`, `scholarship_type`,
`grant_package_grant_package_id`, `sponsored_student_sponsored_student_id`, `scholarship_type_scholarship_type`, `group_id`) VALUES ('200', '2016-10-03 00:00:00', '2017-
10-03 00:00:00', '1', '600', '2', '5', '200', '2', NULL);
```

Figure 4.26: Scholarship package created for student



```
INSERT INTO `plan_ghana`.`scholarship_payment` (`scholarship_payment_id`, `date`, `status`, `refund_amount`, `amount`, `memo`, `year`,
`scholarship_package_scholarship_package_id`, `programarea_residentarea_id`, `programarea_payingarea_id`, `payment_request_payment_request_id`, `bankname`, `account_no`,
`schools_school_id`, `group_id`) VALUES (NULL, '2016-06-07 00:00:00', 'PAID', NULL, '600', NULL, NULL, '200', '2', NULL, '13', 'Merchant Bank', NULL, '52', NULL);
```

Figure 4.27: Scholarship payment made on student’s behalf



First Name	Middle Name	Last Name	Scholarship End Date	School Name	Status
Kwesi	Maxwell	Amamre	2017-09-01 00:00:00	Swedru School of Business	PAID
Jim	Kwame	Okloo	2014-00-00 00:00:00	Methodist Sinior High	PAID
Janet	Esi	Amoah	2017-02-20 00:00:00	Swedru Senior High	PENDING
Jane		Obiri	2017-04-12 00:00:00	ADONTEN SHS	PAID
Jane		Obiri		ADONTEN SHS	PAID
Jason		Mintah		Mfantseman Girls Senior High	PAID
Getrude		Amponsah		Swedru Senior High	PAID
Victor	Agbla	Ankrah		Salga Senior High School	PAID
Rose	Sammia	Baidu		Jabez Educational Institute	PAID
Godwin	Nana	Manhyia		ANFOEGA SHS	PAID
Jim	Kwame	Okloo	2014-00-00 00:00:00	Manya Krobo Senior High	PAID
Bridget	Yaa	Ampaduh	2017-10-03 00:00:00	JUABEN SHS, ASHANTI	PAID

Figure 4.28: Table showing details of newly added student

Chapter 5: Conclusions and Recommendations

5.1 Challenges

One main challenge with the implementation was writing the SQL queries. This involved joining multiple tables and including conditions and groupings that did not always work well together. As a result, retrieving data was a complicated process that would often result in data that looked correct but was in fact wrong. Adding to the difficulty was the lack of foreign key constraints on the provided database. The lack of foreign key constraints required that most inputs be double-checked to ensure that the correct data was input across all the tables to ensure that results of tests were consistent.

5.2 Future Work

The user interface can be improved in order to create a better user experience for the donors. Research needs to be undertaken to gain insights into how to create web pages that involve a lot data and how to present that data to the users in the most effective manner.

The non-functional requirement of reliability could not be achieved in this project. In future versions of this application, reliability should be ensured to allow donors to view data in the event of failure of the Google Charts library.

In addition, information needs to be provided on donors' phones. The bootstrap framework can be used to make the page responsive to different screen sizes. Another pressing need is to be able to display information to donors if graphs cannot be properly displayed on phones. A suitable means to display information should be found in the event that this happens.

The donor page also can be used as a platform for donors to communicate with recipients, their parents or guardians. This would help ensure that donors know how some of their students are faring and would also help donors keep regular contact with students that they take a personal interest in.

Furthermore the donors can gain more value from the data if data mining is employed. As the number of scholarship offerings grows and the data increases, data mining tools can be employed in order to provide insightful information that can aid donors in making more informed decisions. For example, connections between student grades and program areas may be made in order to determine why students from particular areas may or may not be as successful as others. This may prompt research and may provide further opportunities for the organization and donors to make an impact beyond offering scholarships.

5.3 Conclusion

This project set out to include an important group of users into the NGO's application: the donors. It provided useful features that allowed donors to view information that would help them in the decision-making and also improve the organization's trustworthiness. These features included being able to view payments made for a particular financial year, for the current financial year and for entire grant's lifetime. Using text or graphs depending on the data, trends and data were made available to donors in a clear and concise manner.

Additionally, information pertaining to students was also displayed to the donor. This information included gender statistics of the students, the number of communities that grant serves and also provides a list of students currently enjoying the scholarship. This information is essential for donors because it enables them to directly see the impact of their contribution.

Based on the test results, it is safe to say that the project succeeded in achieving its goal.

The project also set out to propose activities to undertake during deployment. The aim was to ultimately improve the usage of the application. In this light, the project suggested activities and software that would be useful in the deployment of subsequent versions.

6 References

Bloom, L., & Clark, N. (2008). IT-Management Software Deployment: Field Findings and Design Guidelines. Retrieve March 25, 2017, from http://delivery.acm.org/10.1145/1480000/1477985/a8-bloom.pdf

Bootstrap: The world's most popular mobile-first and responsive front-end framework. (n.d.). Retrieved May 2, 2017, from http://getbootstrap.com/

Gnu.org. (n.d.). Retrieved April 26, 2017, from https://www.gnu.org/software/make/

Jquery.org. (n.d.). Retrieved April 10, 2017, from https://jquery.com/

Mantyla, M. V., & Vanhanen, J. (2011). Software Deployment Activities and Challenges - A Case Study of Four Software Product Companies. In 2011 15th European Conference on Software Maintenance and Reengineering (pp. 131–140). https://doi.org/10.1109/CSMR.2011.19

Unkelos-Shpigel, N., & Hadar, I. (2013). A multitude of requirements and yet sole deployment architecture: Predictors of successful software deployment. In Twin Peaks of Requirements and Architecture (TwinPeaks), 2013 2nd International Workshop on the (pp. 19–23). https://doi.org/10.1109/TwinPeaks.2013.6614719

7 Appendix

Table 1.1: Tabular description of feature to view payment of grant for particular year

View Payment of Grant for current year	
Actor	Donor
Description	Donor can view the amount of money that has been paid till date for the current financial year
Data	Payment of grant in Ghana Cedis
Stimulus	Displayed on page load
Response	Cost displayed on the screen
Comments	The donor can only view information pertaining to their grant

Table 1.2: Tabular description of view payment of grant since first payment requirement

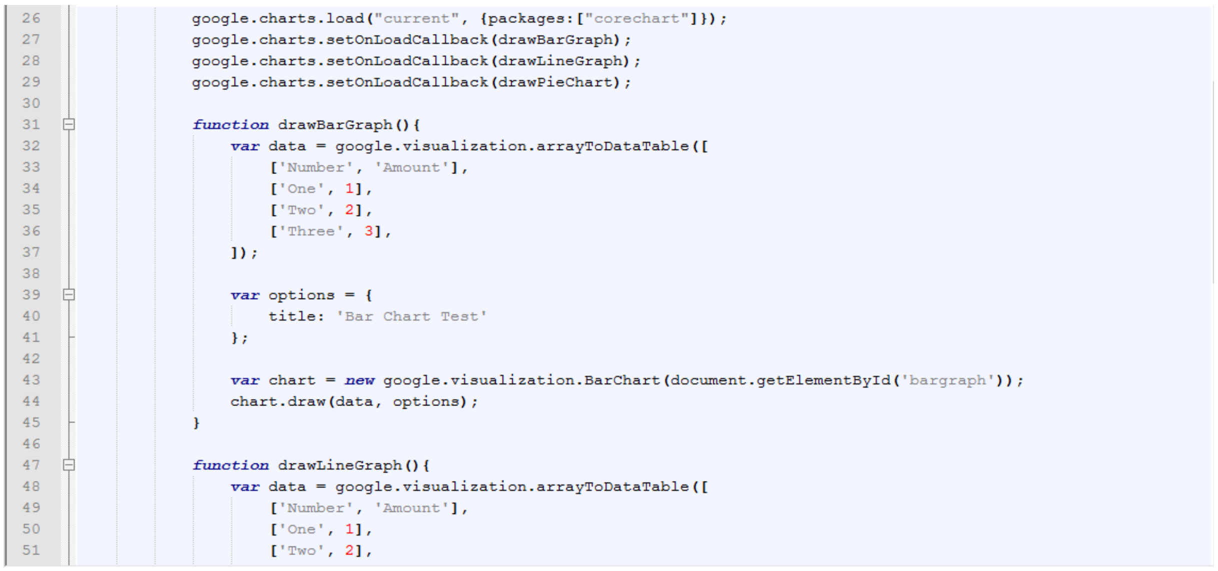
View Payment of Grant since first payment	
Actor	Donor
Description	Donor can view the amount of money that has been paid till date since the inception of the grant
Data	Payment of grant in Ghana Cedis
Stimulus	Displayed on page load
Response	Cost displayed on the screen
Comments	The donor can only view information pertaining to their grant

Table 1.3: Tabular description of view list of current students requirement

View List of current students on the grant	
Actor	Donor
Description	Donor can view a list of students whose fees are currently funded by the grant. This list will contain information such as the students' full name, school, expected scholarship end date and status of payment
Data	Information pertaining to students
Stimulus	Displayed on page load
Response	Information displayed on the screen
Comments	The donor can only view information pertaining to students

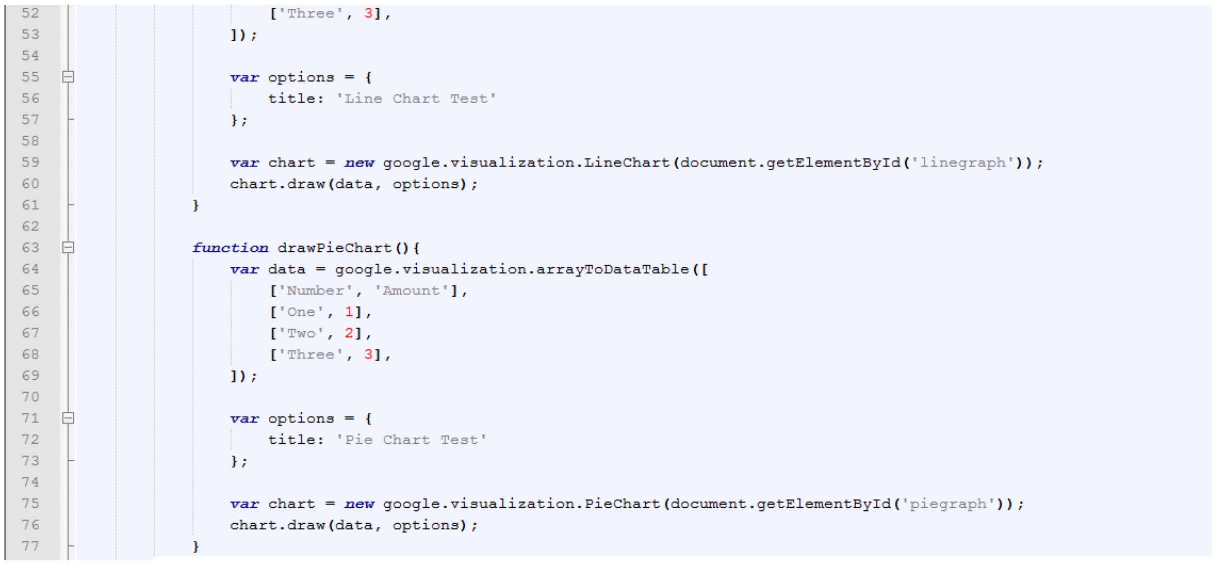
Table 1.4: Tabular description of view payment of grant for particular year requirement

View Statistics pertaining to grant	
Actor	Donor
Description	Donor can view statistics based on how the grant has been disbursed. Information includes the gender distribution of sponsored students, type of communities that have been awarded grants, occupations of students' guardians, etc.
Data	Information pertaining to students, communities and schools
Stimulus	Displayed on page load
Response	Information displayed on screen
Comments	The donor can only view information pertaining to their grant



```
google.charts.load("current", {packages:["corechart"]});
google.charts.setOnLoadCallback(drawBarGraph);
google.charts.setOnLoadCallback(drawLineGraph);
google.charts.setOnLoadCallback(drawPieChart);

function drawBarGraph(){
    var data = google.visualization.arrayToDataTable([
        ['Number', 'Amount'],
        ['One', 1],
        ['Two', 2],
        ['Three', 3],
    ]);

    var options = {
        title: 'Bar Chart Test'
    };

    var chart = new google.visualization.BarChart(document.getElementById('bargraph'));
    chart.draw(data, options);
}

function drawLineGraph(){
    var data = google.visualization.arrayToDataTable([
        ['Number', 'Amount'],
        ['One', 1],
        ['Two', 2],
```

Figure 4.1.1: Test code for displaying graph using Google Charts



```
            ['Three', 3],
        ]);

        var options = {
            title: 'Line Chart Test'
        };

        var chart = new google.visualization.LineChart(document.getElementById('linegraph'));
        chart.draw(data, options);
    }

    function drawPieChart(){
        var data = google.visualization.arrayToDataTable([
            ['Number', 'Amount'],
            ['One', 1],
            ['Two', 2],
            ['Three', 3],
        ]);

        var options = {
            title: 'Pie Chart Test'
        };

        var chart = new google.visualization.PieChart(document.getElementById('piegraph'));
        chart.draw(data, options);
    }
```

Figure 4.1.2: Test code for displaying graph using Google Charts

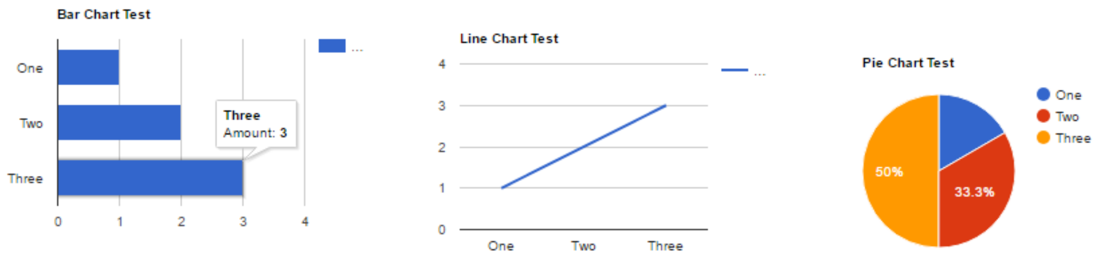
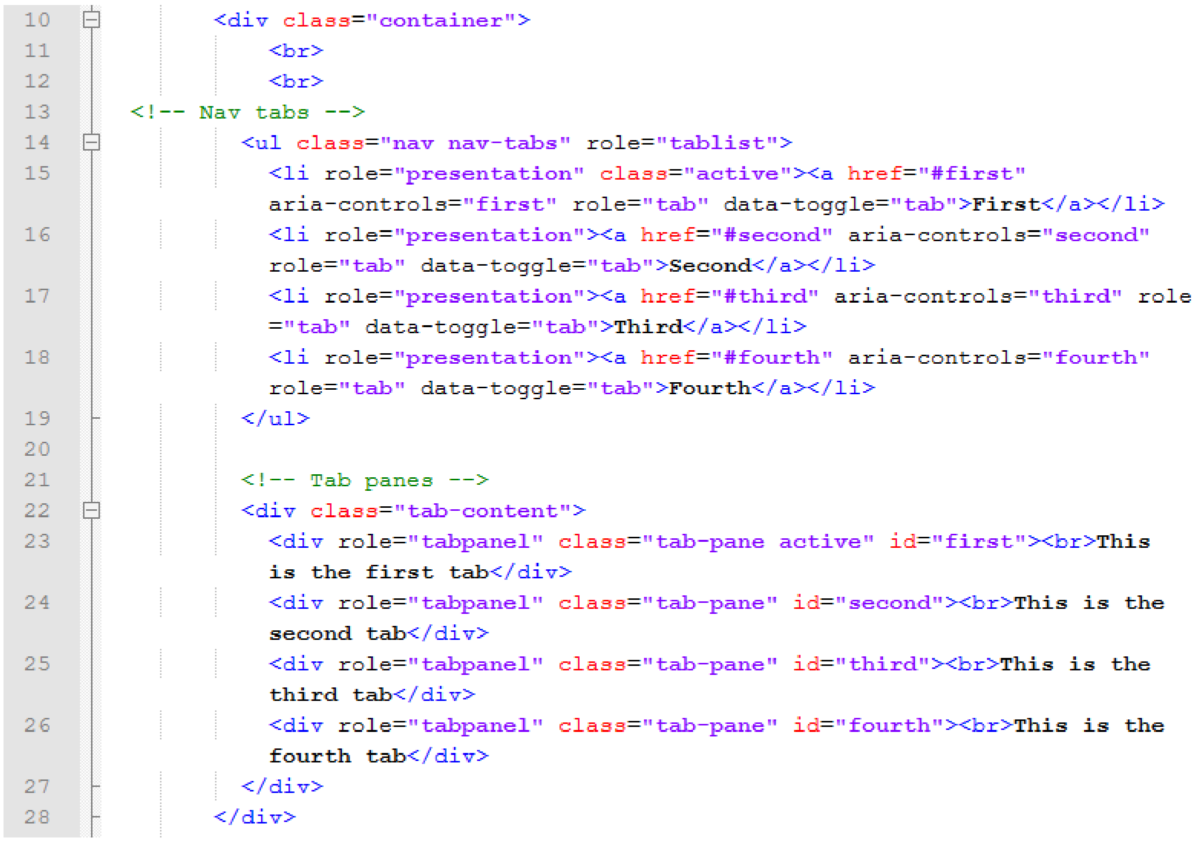
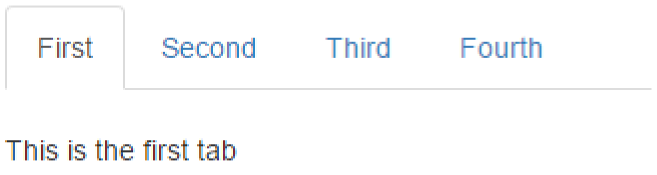


Figure 4.1.3: Test results for displaying graph using Google Charts



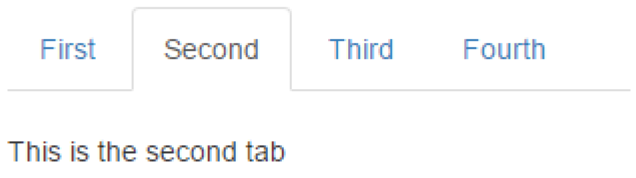
```
        <div class="container">
            <br>
            <br>
<!-- Nav tabs -->
          <ul class="nav nav-tabs" role="tablist">
            <li role="presentation" class="active"><a href="#first"
            aria-controls="first" role="tab" data-toggle="tab">First</a></li>
            <li role="presentation"><a href="#second" aria-controls="second"
            role="tab" data-toggle="tab">Second</a></li>
            <li role="presentation"><a href="#third" aria-controls="third" role
            ="tab" data-toggle="tab">Third</a></li>
            <li role="presentation"><a href="#fourth" aria-controls="fourth"
            role="tab" data-toggle="tab">Fourth</a></li>
          </ul>

          <!-- Tab panes -->
          <div class="tab-content">
            <div role="tabpanel" class="tab-pane active" id="first"><br>This
            is the first tab</div>
            <div role="tabpanel" class="tab-pane" id="second"><br>This is the
            second tab</div>
            <div role="tabpanel" class="tab-pane" id="third"><br>This is the
            third tab</div>
            <div role="tabpanel" class="tab-pane" id="fourth"><br>This is the
            fourth tab</div>
          </div>
        </div>
```

Figure 4.1.4: Test code to create tabbing effect with Bootstrap Framework



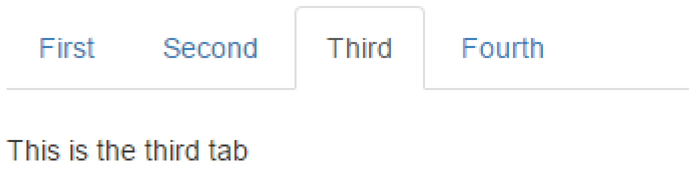
This is the first tab

Figure 4.1.5: Test result for creating tab effect in Bootstrap Framework



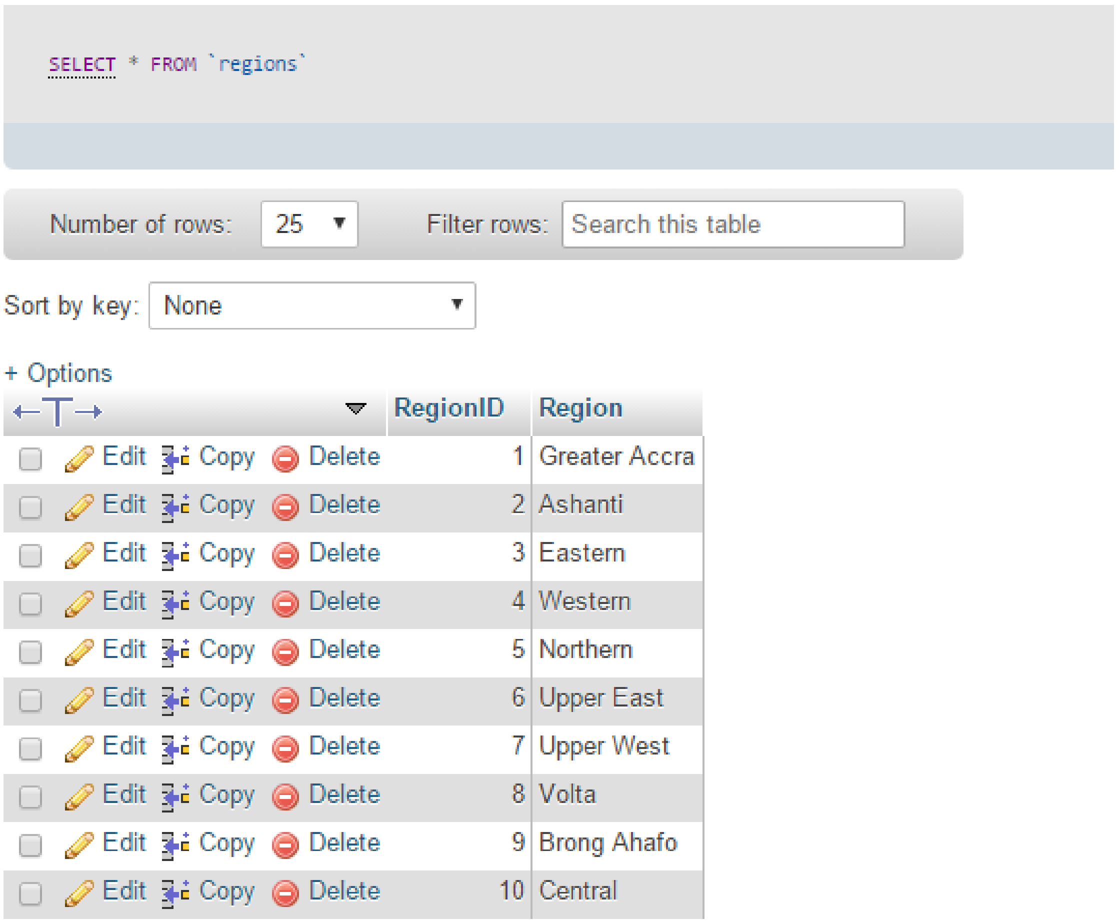
This is the second tab

Figure 4.1.6: Test result for creating tab effect in Bootstrap Framework



This is the third tab

Figure 4.1.7: Test result for creating tab effect in Bootstrap Framework



```
SELECT * FROM `regions`
```

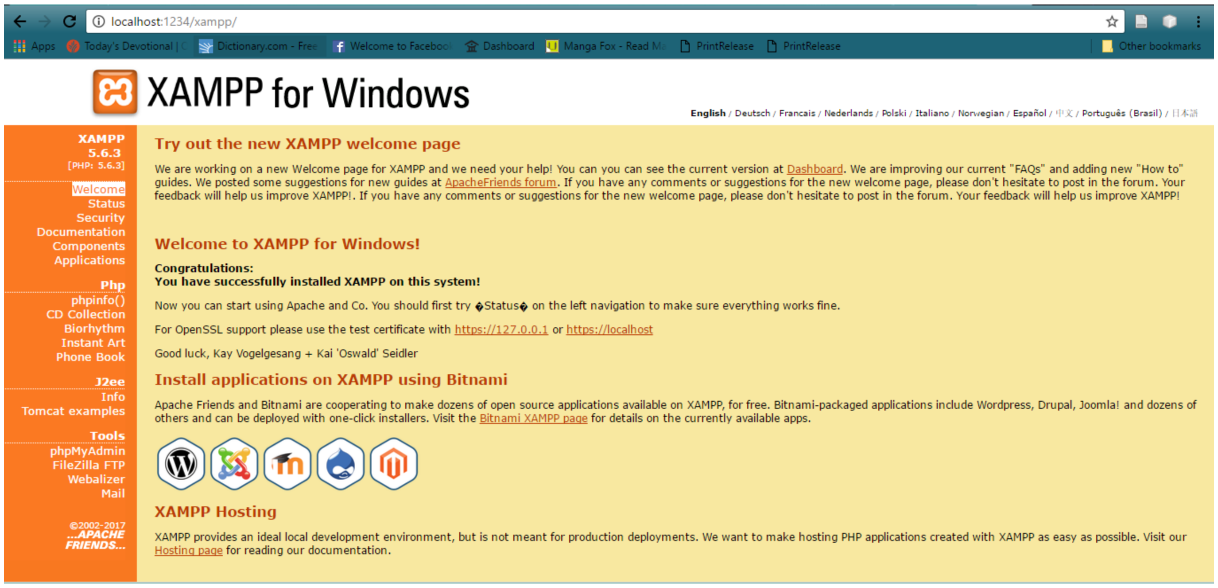
Number of rows: 25 Filter rows: Search this table

Sort by key: None

+ Options

	RegionID	Region
Edit Copy Delete	1	Greater Accra
Edit Copy Delete	2	Ashanti
Edit Copy Delete	3	Eastern
Edit Copy Delete	4	Western
Edit Copy Delete	5	Northern
Edit Copy Delete	6	Upper East
Edit Copy Delete	7	Upper West
Edit Copy Delete	8	Volta
Edit Copy Delete	9	Brong Ahafo
Edit Copy Delete	10	Central

Figure 4.1.8: Test SQL query and result for phpMyAdmin and MySQL



XAMPP for Windows

English / Deutsch / Francais / Nederlands / Polski / Italiano / Norwegian / Español / 中文 / Português (Brasil) / 日本語

XAMPP
5.6.3
[PHP: 5.6.3]

Welcome
Status
Security
Documentation
Components
Applications

Php
phpinfo()
CD Collection
Biorhythm
Instant Art
Phone Book

J2ee
Info
Tomcat examples

Tools
phpMyAdmin
FileZilla FTP
Webalizer
Mail

©2002-2017
...APACHE
FRIENDS...

Try out the new XAMPP welcome page

We are working on a new Welcome page for XAMPP and we need your help! You can you can see the current version at Dashboard. We are improving our current "FAQs" and adding new "How to" guides. We posted some suggestions for new guides at ApacheFriends forum. If you have any comments or suggestions for the new welcome page, please don't hesitate to post in the forum. Your feedback will help us improve XAMPP!. If you have any comments or suggestions for the new welcome page, please don't hesitate to post in the forum. Your feedback will help us improve XAMPP!

Welcome to XAMPP for Windows!

Congratulations:
You have successfully installed XAMPP on this system!

Now you can start using Apache and Co. You should first try �Status� on the left navigation to make sure everything works fine.

For OpenSSL support please use the test certificate with https://127.0.0.1 or https://localhost

Good luck, Kay Vogelgesang + Kai 'Oswald' Seidler

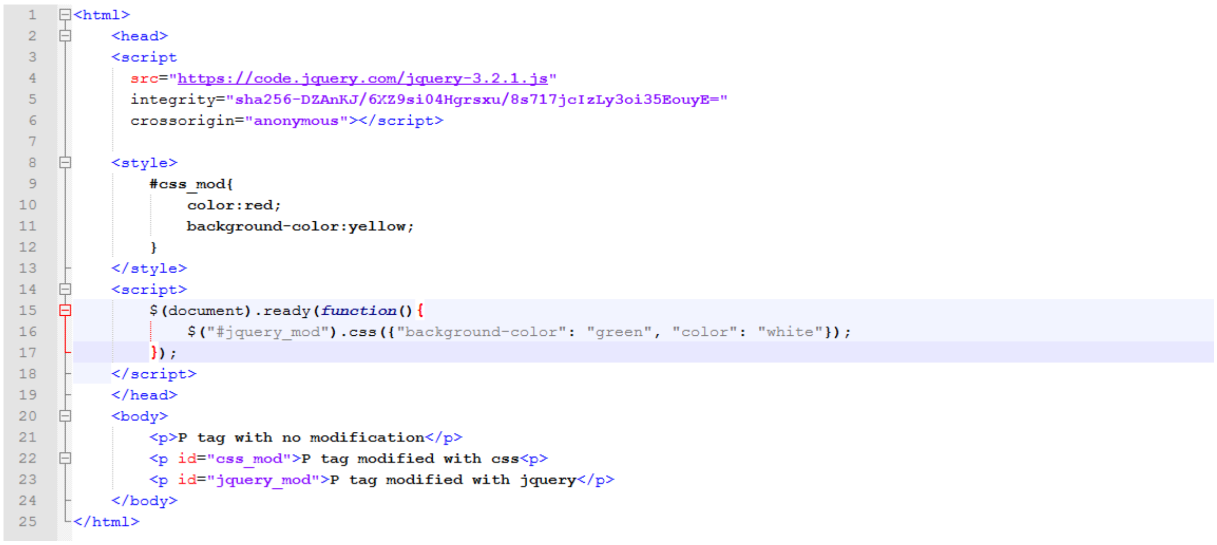
Install applications on XAMPP using Bitnami

Apache Friends and Bitnami are cooperating to make dozens of open source applications available on XAMPP, for free. Bitnami-packaged applications include Wordpress, Drupal, Joomla! and dozens of others and can be deployed with one-click installers. Visit the Bitnami XAMPP page for details on the currently available apps.

XAMPP Hosting

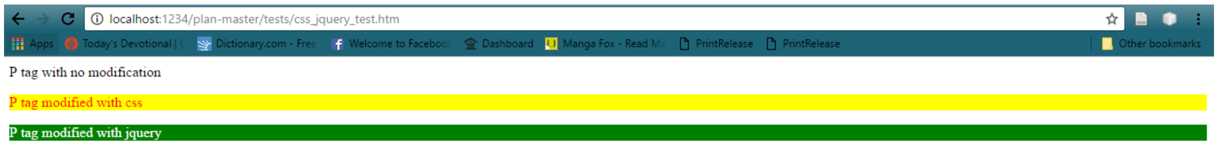
XAMPP provides an ideal local development environment, but is not meant for production deployments. We want to make hosting PHP applications created with XAMPP as easy as possible. Visit our Hosting page for reading our documentation.

Figure 4.1.9: Test result for Apache Server and XAMPP



```
<html>
    <head>
    <script
      src="https://code.jquery.com/jquery-3.2.1.js"
      integrity="sha256-DZAnKJ/6XZ9si04Hgrsxu/8s717jcIzLy3oi35EouyE="
      crossorigin="anonymous"></script>

    <style>
        #css_mod{
            color:red;
            background-color:yellow;
        }
    </style>
    <script>
        $(document).ready(function(){
            $("#jquery_mod").css({"background-color": "green", "color": "white"});
        });
    </script>
    </head>
    <body>
        <p>P tag with no modification</p>
        <p id="css_mod">P tag modified with css<p>
        <p id="jquery_mod">P tag modified with jquery</p>
    </body>
</html>
```

Figure 4.1.10: Code for testing jQuery and CSS



```
P tag with no modification
P tag modified with css
P tag modified with jquery
```

Figure 4.1.11: Test result of CSS and jQuery test