DOI <https://doi.org/10.18551/rjoas.2018-05.11>

DEVELOPING STUDENT FINANCIAL INFORMATION SYSTEM WITH THE BASIS OF CROSS-PLATFORM DESKTOP APPLICATION

Hardi Belgis Rahmania

Magister Program of Informatics Engineering, University of Muhammadiyah Malang, Indonesia

Kusuma Wahyu Andika

Lecturer of Informatics Engineering, Universitas Muhammadiyah Malang, Indonesia

*E-mail: belgis.rh@gmail.com

ORCID: 0000-0002-8861-5338

ABSTRACT

This study aims to develop student financial information system so that it can be applied in SMAN 1 Panggul. The electron framework is applied to student financial application with the basis of cross-platform desktop application. Based on the research findings, it can be concluded that if we can develop a website then it is not impossible if we want to create a desktop application. With the advancement of the technological era, we need to think twice in order to minimize the time and the effectiveness of something in order to work more functionally and flexibly. This study presents a comparison between two development types of cross-platform for desktop application: object-oriented solution and web-based solution.

KEY WORDS

Information system, application, cross-platform, web solutions.

Currently, information technology is growing so rapidly. The existing information system runs according to the level of user needs. What is meant here is computer-based information technology which is lately growing very fast, both hardware and software. However, based on the reality in daily life, the delay in receiving information is still occurring. Thus, decision-making is often too late. This will certainly disrupt the course of an institution or organization. One of the factors causing the information delay is the lack of organized information processing (Laudon and Laudon, 2016). In the world of agency work, computerization has many benefits. One of them is useful for SMAN 1 PANGGUL. Administration Staff of SMAN 1 PANGGUL is the section that process data related to student financial data.

In the processing of information systems, especially the collection of student financial data, constraints often occur. Processing information system should be handled effectively and efficiently. However, because of many processes are still done manually, then whenever the report of student financial data is required, they must first find the necessary data from piles of papers or archives and type them then. Incoming and outgoing data is also not well-organized. In addition, the author wants to have an experiment on the sophistication of today's technological world that runs very rapidly. It means that the author will apply some advanced java script technology (Huang, 2006).

System Analysis. The system that will develop in the research of this final project is a web-based desktop application (Leff and Rayfield, 2001). The system will run on PC and notebook. This system is developed to facilitate the financial staff (administrative staff) in processing the transactions of student financial payment and clarifying any details of the student financial reports at school SMAN 1 Panggul. The information system to be used is offline. The application is using the basis of cross-platform desktop application in which in its scope it can run on a number of OSs including Windows, Linux, and Mac.

Problem Analysis. Student financial report is a report that is needed by each school in which the report records all student funding activities, (Revsine, 2001). As the times progressed, the information system, carried out by taking notes on the paper then making the

archive, raises many obstacles. It is because the staffs will have difficulty looking for one by one sheet of financial records that are all archived in the data cabinet. Information system carried out in the traditional way is less effective for longer use. Therefore, the development of student financial information system application is expected to facilitate administrative staff (financial staff) in doing their jobs.

Functional Needs Analysis. Functional needs are needs that contain processes that will be carried out by the system (Setyawan and Wandyatmono, 2009). There is a major need that the system needs in order to run smoothly. This need is the core requirement of a system. If these needs are not met then the system cannot run properly.

Non-Functional Needs Analysis. Non-functional needs are complementary features that support the performance of a system and it has an indirect effect. The following is non-functional needs that exist within the system:

- Hardware needs;
- Software needs;
- Brain ware needs;
- Information about the system.

Figure 1 – Use Case Diagram

Table 1 – The Use Case Scenario of Student Financial System

Identification	
Use Case Name	Student financial system
Objective	The admin manage the financial system
Description	In this use case, the admin manages some matters that include payments, expenses, and financial reports
Actor	Admin / Administrative Staff
The Main Scenario	
Actor Action	System Action
-	1. Displaying the main menu of the system page
2. Selecting a menu	
-	3. Running the execution command of the selected menu
Final result: Admin back on the main page	

Table 2 – Data User Scenario

Identification	
Use Case Name	Data User
Objective	It contains all student financial reports
Description	This use case contains data user namely student and admin data
Actor	Admin / Administrative Staff
The Main Scenario	
Actor Action	System Action
-	1. Displaying user
2. Performing CRUD of data user	
-	3. Saving the action
Final result: the system saves every running command	

Table 3 – Payment Scenario

Identification	
Use Case Name	Payment
Objective	It contains all the inputs of student payment
Description	This use case contains all inputs of student payment consisting of school fee, incidental, acceptance of new learners, mandatory savings, voluntary funds and other funds. There is also discount (school operational assistance) for students on certain types of payment
Actor	Admin
The Main Scenario	
Actor Action	System Action
1. Admin selects the payment menu	
	2. Displaying the payment menu
3. Admin selects the type of payment	
	4. Displaying the type of payment
5. Admin search for the number of the student who is paying	
	6. Showing the student numbers
7. Input the amount of the payment and give a discount to students who get a special discount	
	8. Saving the payment process
Final result: the system saves the data of payment process	

Table 4 – Expense Scenario

Identification	
Use Case Name	Expense
Objective	It contains the all outputs of expense from student funds
Description	This use case contains the expenses of the payment, among others, development, acceptance of new learners, learning & extracurricular activities, tests & exams, consumable purchases, power & service subscription, sanitation and rehabilitation, monthly payments, funds for underprivileged students, school maintenance, purchase & maintenance of computer and other expenses.
Actor	Admin
The Main Scenario	
Actor Action	Actor Action
1. Selecting the expense menu	
	2. Displaying the expense option
3. Selecting the type of expense	
	4. Displaying the expense form
5. Filling the expense form	
	6. Systems saves the expense process
Final result: the system saves the payment process	

Table 5 – Financial Report Scenario

Identification	
Use Case Name	Financial Report
Objective	It contains information about all financial statements
Description	This use case contains information on all student financial reports including balance sheets, profit-loss, journal books, ledgers, and school financial reports
Actor	Admin
The Main Scenario	
Actor Action	Actor Action
1. Selecting the information of financial menu	
	2. Displaying the menu of report options
3. Selecting the payment menu that will be selected	
	4. Processing the action
Final result: the system records every activity of the admin	

At this stage, system design will be carried out based on use case that has been developed before. The design starts from the creation of activity diagram, sequence diagram, class diagram, to the design of application interfaces.

Activity Diagram. Figure 2 is an activity diagram of report. The process starts when the admin login first and the system will display the main menu of the application of student financial information system. Admin must select one of the options from the main menu of the report, among which are the financial report, cash flow, balance sheet, income report, ledger and journal book. The database will display data from the menu.

Figure 2 – Activity Diagram of Report

Figure 3 – Activity Diagram of Payment

Figure 3 is the activity diagram of the payment. The process starts when the admin must login first and go into the main menu. Then, the admin chooses a payment menu consisting of acceptance of new learners, incidental, school fee, voluntary funds and mandatory savings. Admin inputs student's number and the database will process it. The system will display the payment form that will be inputted by the admin. The database stores the process if it has been completed. Then, there will be a successful payment notification. But, if it is failed to save, it will return to the payment menu.

Figure 4 is an activity diagram to organize the data users, the process starts when the admin login. If it is successful, it will go directly to the main menu and if it is failed it will repeat the login process. After the main menu, admin will select the menu of edit, upload and delete data users. After selecting one of these menus, the database will process and store every action that has been performed. If it is success, then the process is successful and done. But if it is failed, it will go back to the main menu.

Figure 4 – Activity Diagram of Data User Management

Figure 5 is an activity diagram that manages the expense data. The process starts when the admin selects the expense menu. One of them includes school activities, acceptance of new learner activity, learning activity, maintenance activity, etc. After that the admin inputs the nominal amount and the program will execute and then save it.

Figure 5 – Activity Diagram of Expense

Sequence Diagram. Sequence diagram is a diagram that describes the behavior of the object in the use case by describing the life line of objects and the sent and received messages between the objects. The following is a sequence diagram of the developed system:

Figure 6 – Sequence Diagram of Financial Report

Figure 7 – Sequence of Payment

Figure 8 – Sequence of Data User Management

Figure 9 – Sequence of Expense

Entity Relationship Diagram (ERD). Entity Relationship Diagram (ERD) is a graphical representation of an information system that shows the relationship between people, objects, places, concepts or events within a system, (Li and Chen, 2009). ERD is a data modeling technique that can help define business processes and it can be used as a relational database, (Song et al., 2007). The following is the ERD of the student financial system at SMAN 1 Panggul.

Main Display and Login Feature. When the app starts up, a browser will appear, some of which are: a login feature will appear in the navigation bar. Login will be successful if the username and password are inputted correctly. If it is wrong, the login will be failed. Figure 4.1 shows student financial information system application (SFISA) with the basis of desktop app running on windows OS and Linux OS.

When the application is running, the username and password that have been previously registered must be entered; the username and password that has been filled in the page admin.html bios. If it is a wrong input, then it cannot process the login. The following is a script for login.html as front-end and login.js as back-end.

```

Code login.html
<!DOCTYPE html>
<html>
<head>

```

```

</head>
<body>
<div id="loginbox" style="text-align:center;">
<form id="login-form" class="form-vertical">
<div class="control-group normal_text" >
<h4>A.S.I.K.S <br/></h4><b>Aplikasi Sistem Informasi Keuangan Siswa</b></div>
<input id="username" name="username" type="text" placeholder="Username" />
</div>
</div>
</div>
<div class="control-group" style="padding:0px 0;">
<div class="controls">
<div class="main_input_box">
<span class="add-on"><i class="icon-lock"></i></span>
<input id="password" name="password" type="password" placeholder="Password" />
</div>
</div>
</div>
<div class="form-actions">
<span style="text-align:center;"><input type="submit" class="btn btn-success" value="Login" /></span>
</div>
</form>
<form id="recoverform" action="#" class="form-vertical">
<p class="normal_text">Enter your e-mail address below and we will send you instructions how to recover a password.</p>

<div class="controls">
<div class="main_input_box">
<span class="add-on"><i class="icon-envelope"></i></span><input type="text" placeholder="E-mail address" />
</div>
</div>

<div class="form-actions">
<span class="pull-left"><a href="#" class="flip-link btn btn-inverse" id="to-login">&laquo; Back to login</a></span>
<span class="pull-right"><input type="submit" class="btn btn-info" value="Recover" /></span>
</div>
</form>
</div>

</body>
<script>>window.$ = window.jQuery = require('jquery');</script>
<script>require('bootstrap')</script>
<script src="../../js/login.js"></script>

```

```

Code login.js
const p = require('path');
const knexpath = p.resolve(__dirname, '../..', 'knexfile');
const knexfile = require(knexpath);
var knexConfig = {
  client: 'mysql',
  connection: {
    host: '127.0.0.1',
    user: 'root',
    password: '',
    database: 'siks'
  }
};
var knex = require("knex")(knexConfig);

const {webContents} = require('electron');

const electron = require('electron')
const path = require('path')
const app = electron.app
const url = require('url')
$(function($) {

```



```

var form = $('#login-form');
$(input[name="username"], input[name="password"]).on('focus', function() {
$('.help-block').addClass('hide');
$('.form-group').removeClass('has-error');
});
form.on('submit', function(e) {
e.preventDefault(), e.stopImmediatePropagation();
var formData = new FormData($('#form#login-form')[0]);
var gUsername = $('.group-username'),
gPassword = $('.group-password'),
username = formData.get('username'),
password = formData.get('password'),
buttonSubmit = $(this).find('button[type="submit"]');
buttonSubmit.button('loading');
gUsername.removeClass('has-error');
gPassword.removeClass('has-error');
var usernameErrorMsg = "";
var passwordErrorMsg = "";
var triggerError = function(message, owner) {
let wrapper = (owner == "username") ? gUsername : gPassword;
wrapper
.removeClass('has-error')
.addClass('has-error');
wrapper
.find('.help-block')
.removeClass('hide')
.text(message)
.css('font-style', 'italic');
};
if(username == "") triggerError("Username cannot be blank.", 'username');
knex('biodataadmin').where({
username: formData.get('username'),
password: formData.get('password'),
status : "aktif",
}).then(function(rows){
let tmp = false;
var id="";
$.each(rows, function(i, e) {
tmp=true;
id=e.id;
});
if(tmp){
let navigateUrl = url.format({
pathname: path.join(__dirname, 'pembayaran.html'),
protocol: 'file:',
slashes: true
})
window.location.href = navigateUrl;
localStorage.setItem("iduser",id);
localStorage.setItem("namauser",formData.get('username'));
}else{
alert("Username / Password Salah.");
}
});
buttonSubmit.button('reset');
});
});

```

Payment Feature. The payment feature is the space where all income processes from the school financial of SMAN 1 Panggul is carried out, which includes the data of school fee payment, incidental, acceptance of new learner, mandatory saving, voluntary fund, and other funds. The type of expense can also be managed via through account code page. In addition, the nominal amount of payments from each different student can be managed through the settings page. There is also the edit and delete function as the delete and change functions of payment data.

Source Part to Display Payment Data:

```

var retrievekodeAkun = function() {
  if(table.find('tbody').length != 0) {
    table.find('tbody').empty();
  }
  knex('kodeAkun')
    .then(function(rows){
      let tableRows = "";
      $.each(rows, function(i,e) {
        tableRows += '<tr>';
        tableRows += '<td>'+(i+1)+'</td>';
        tableRows += '<td>'+ e.id +'</td>';
        tableRows += '<td>'+ e.kelompok +'</td>';
        tableRows += '<td>'+ e.kategori +'</td>';
        tableRows += '<td>'+ e.kodeAkun +'</td>';
        tableRows += '<td>'+ e.namaAkun +'</td>';
        tableRows += '<td>';
        tableRows += '<button type="button" id="edit" class="btn btn-primary btn-mini" data-toggle="modal" data-target="#modal-edit-kodeAkun" data-id="'+ e.id +'> Edit</button> ';
        tableRows += '<button type="button" id="delete" class="btn btn-danger btn-mini" data-id="'+ e.id +'> Delete</button> ';
        tableRows += '</td>';
        tableRows += '</tr>';
      })
      table.append($('<tbody>').append(tableRows));
    });
  };

```

Source Part to Delete Payment Data:

```

(btnId == "delete") {
  knex('pembayaran').where('id', btnDataId).del().then(function(resp) {
    rekamAktifitas(localStorage.getItem("namauser")+" telah melakukan hapus data (kode pembayaran : "+btnDataKode+" )", "pembayaran");
    retrievepembayaran();
  });
}

```

The Part of Source Code to Edit Payment Data:

```

(btnId == "edit") {
  knex('pembayaran').where('id', btnDataId).then(function(response) {
    if(response.length > 0) {
      var pembayaran = response[0];
      modalEdit.find('form').attr('id', 'form-edit-pembayaran');
      modalEdit.find('form input[name="id"]').val(pembayaran.id);
      modalEdit.find('form input[name="cicilan"]').val(pembayaran.cicilan);
      modalEdit.find('form input[name="jumlahBayar"]').val(pembayaran.jumlahBayar);
      modalEdit.find('form input[name="jumlahPD"]').val(pembayaran.jumlahPD);
      modalEdit.find('form input[name="keteranganPembayaran"]').val(pembayaran.keteranganPembayaran);
      modalEdit.find('form input[name="namaPenyumbang"]').val(pembayaran.namaPenyumbang);
      modalEdit.find('form input[name="kodeTransaksi"]').val(pembayaran.kodeTransaksi);
      modalEdit.find('form select[name="userId"]').val(pembayaran.userId);
      modalEdit.find('form select[name="jenisBayar"]').val(pembayaran.jenisBayar);
      modalEdit.find('form select[name="jenisKas"]').val(pembayaran.jenisKas);
      modalEdit.find('form select[name="bulan"]').val(pembayaran.bulan);
      modalEdit.find('form select[name="piutangDiskon"]').val(pembayaran.piutangDiskon);
      jumlahPDnominaledit();
      modalEdit.find('form').data('id', btnDataId);
      modalEdit.modal('show');
    }
  });
}

```

Part of Source Code to Insert Payment Data:

```

form.on('submit', function(e) {
e.preventDefault();
e.stopImmediatePropagation();
var formData = new FormData(form[0]);
var idakun="";
knex('kodeakun').where('kodeAkun', formData.get('jenisBayar')).then(function(rows){
$.each(rows, function(i, e) {
idakun=e.id;
});
knex('pembayaran').insert({
userId : formData.get('userId'),
kodeAkunId : idakun,
id : formData.get('id'),
jenisBayar : formData.get('jenisBayar'),
jenisKas : formData.get('jenisKas'),
cicilan : formData.get('cicilan'),
bulan : formData.get('bulan'),
jumlahBayar : formData.get('jumlahBayar'),
piutangDiskon : formData.get('piutangDiskon'),
jumlahPD : formData.get('jumlahPD'),
nominal : formData.get('jumlahPDnominal'),
keteranganPembayaran : formData.get('keteranganPembayaran'),
namaPenyumbang : formData.get('namaPenyumbang'),
kodeTransaksi : formData.get('kodeTransaksi'),
tglPembayaran : formData.get('tglPembayaran')
})
})

```

Expense Feature. In the SFISA expense feature, the system records and stores all transactions and school expenses with the options listed on the Account Code page. The similar thing applies to edit and delete buttons which are used to change and delete data from payment.

Account Code Feature. The following features contain codes of SFISA's various assets, incomes, fees, receivables and debts that are still interconnected with income, expense, and financial report pages.

User Feature. The following features contain pages from admin and student users. User data contains data from student and admin (financial staff) in which the data serves as a complementary data from payment pages and SFISA expenses.

Financial Report Feature. The journal report is a systematic and chronological record of financial transaction that occurs in a company. The journal is a record of debiting and crediting the effect of the transaction with the required explanations within the transaction.

Ledger Report. The ledger is a collection of interconnected accounts (estimates) that constitute an entity compiled and grouped according to the accounts of the company's financial reports. The ledger source is the journal that has been created before. After the transactions are recorded in the journal, the next step is to move all the journals into their respective accounts in the ledger.

Feature of Profit and Loss Report. Profit or loss is the result of periodical calculation. The following income is derived from the termination of the liquidated activity. In the following report, we can see whether the school is experiencing loss or gain.

Feature of Balance Sheet Report. The balance sheet is prepared after all journals are posted in the ledger and before journal entries are made. The preparation of accounts in the trial balance is based on an arrangement in which it will facilitate the preparation of financial reports.

SFISA Navigation Bar Feature. The following feature is a function of the navigation bar that consists of a profile (complete data from the admin that is currently logged in) and record activity (a page that records all activities that the admin does such as adding payment data, deleting, altering or changing, with the aim that any data changes can be recorded for instance, an error can be minimized by knowing the log activity).

CONCLUSION

Based on the research findings, it can be concluded that if we can develop a website then it is not impossible if we want to create a desktop application. With the advancement of the technological era, we need to think twice in order to minimize the time and the effectiveness of something in order to work more functionally and flexibly. This study presents a comparison between two development types of cross-platform for desktop application: object-oriented solution and web-based solution. JavaFX refers to the Java programming language that has the slogan "write once, run anywhere". On the other hand, Electron is a web-based technology that is currently used in many companies such as Microsoft, Facebook, Slack, and Docker. Electron combines node.js with chromium and uses JavaScript, HTML5, and CSS to create desktop applications. Two versions of the same application were developed using JavaFX and Electron frameworks. The application is used to compare the technology based on the criteria described in the section 4.3. This comparison aims to find the differences among the solutions and find its benefits and drawbacks.

REFERENCES

1. Hitchner, J. R. (2010). Financial Valuation,+ Website: Applications and Models (Vol. 545). John Wiley & Sons.
2. Huang, Y., Xie, Z., & Zhong, S. B. (2006). Application of .NET, VML, JavaScript in WebGIS map display. *Science of Surveying and Mapping*, 4, 045.
3. Laudon, K. C., & Laudon, J. P. (2016). *Management information system*. Pearson Education India.
4. Leff, A., & Rayfield, J. T. (2001). Web-application development using the model/view/controller design pattern. In *Enterprise Distributed Object Computing Conference, 2001. EDOC'01. Proceedings. Fifth IEEE International* (pp. 118-127). IEEE.
5. Li, Q., & Chen, Y. L. (2009). Entity-relationship diagram. In *Modeling and Analysis of Enterprise and Information Systems*(pp. 125-139). Springer, Berlin, Heidelberg.
6. Revsine, C. (2001). Johnson. *Financial Reporting and Analysis*. New Jersey: Prentice Hall.
7. Setyawan, A., & Wandyatmono, J. (2009). Sistem Informasi Penggajian Pegawai Kecamatan Geneng Kabupaten Ngawi. *Speed-Sentra Penelitian Engineering dan Edukasi*, 1(3).
8. Song, I. Y., Khare, R., & Dai, B. (2007). SAMSTAR: a semi-automated lexical method for generating star schemas from an entity-relationship diagram. In *Proceedings of the ACM tenth international workshop on Data warehousing and OLAP*(pp. 9-16). ACM.