*E-ISSN: 2476-9606*
***Abstract Proceedings International Scholars Conference***
*Volume 7 Issue 1*, October 2019, pp. 1729-1740

https://doi.org/10.35974/isc.v7i1.1372

# Membership Information System Using Node JS

Jay Idoan Sihotang[1], Yosua Richel Simanjuntak[2], Andrew Fernando Pakpahan[3]
Fakultas Teknologi Informasi, Universitas Advent Indonesia
*jayidoans@unai.edu*

## ABSTRACT

In the process of organizing membership activities, the speed and accuracy of member data processing is required. Problems in processing the data include, among other things, several organizations that have not used the system to process member's data, including managing and creating member data, organizational activity schedules, and member's finances. Node Js is the language used to design this information system aims to find out whether Node Js can be used in making membership information systems. With the UML design method, the author develops this system to be more structured and according to purpose. The results obtained are that Node Js is very capable of being used to design membership information systems with output results are reports of member data and financial data that has been recorded by the system administrator. The membership management information system has been running well and all functions run according to the design in which there are member data collection, finance, activities, and donations. The member data collection process is done by the admin by recording the activity history of the members including registration time, update time, login time, incoming and outgoing financial history, history of donations and activities followed by members. Advice from the author is that a membership management system that has been designed may be developed and a notification function can be added so that it can more easily access the requested data, and the mobile version is made.

**Keywords**: Membership System, Information System, Node Js, Adonis Js.

## INTRODUCTION

The rapid development of technology has caused a lot of research to develop technology that is flexible, especially in carrying out business processes in various business fields. This expectation arises because of the many technology users, one of whom uses web services to deliver information that is ready for presentation. Information system is a system within an organization that enabled the needs of daily transaction management, support the operation, managerial and strategic activities of an organization and provide certain outside parties with the necessary reports (Hutahaean, 2015). For example, Registration information systems and much more.

In collecting data on an organization, an online system is needed to regulate member registration, member finances, member activities, and member donations. Web-based systems

can help produce information that is fast and accurate, and can improve work effectiveness (A. Widodo, 2015). Membership information systems can also help the management of system users, by avoiding unauthorized access from unknown users (Graviardhi, Satoto, & Fatur, 2011). And it is proven that the system that was designed using PHP programming language and MySQL database system can work well (Tobing, 2017).

Based on previous studies, the authors will develop membership information system using Node Js and MySQL database to facilitate member's data collection, to ease the administrative work of the membership, and prevent the loss of data due to un-centralized data. The purpose of this research is to create a web-based membership information system. While the purpose of this study is to find out how to design and develop membership information system using the Node Js.

## METHODS

This research uses the Waterfall method which is also known as the Linear Sequential Model and is often referred to as the Classic Life Cycle (Pressman, 2012). This model is included in the generic model in software engineering and was first introduced by Winston Royce around 1970 so it is often considered obsolete, but it is the most widely used model in Software Engineering (SE). This model approaches systematically and sequentially. It is called waterfall because every next step requires the completion of the previous stage and walk in sequence. The stages of the waterfall model can be seen in Figure 1.

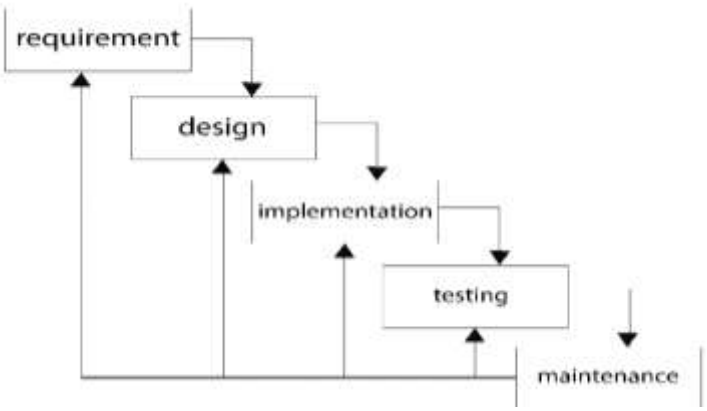


Figure 1. **Waterfall Model**

Business Process Engineering

Business process re-engineering is a methodology in redesigning existing business processes and adapted to system requirements. In other words, all business processes that do not produce value in the new system will be discarded or changed, so that the business processes that have been updated can produce optimal value (Hammer & Champy, 2009).

In the business process of registering new members, prospective members must register on the register page, fill in all the forms correctly, after that the prospective member must activate the account through the link that has been sent to the prospective member's email, and the account is ready to use in the login page. The business process for member deactivation, starting from the member filling in the form of reasons to deactivate the account on the delete account page, then the system will change the status of the member to inactive and the system will delete the incoming session from the current member, and the system will redirect to the login page. The business process for account balance top-up, starting from opening the add balance page and fill in the form completely with the nominal balance to be filled, after that the system will store in the database and add into the request list on the admin page, then the admin will check the transfer receipt entered. If the transfer is valid, the admin will top-up the account balance. And if it is invalid, then the balance will not be added.

Refund business process, starting from members filling the refund form on the refund page, then the system will save and submit the queue list on the admin page. After validating the request, the admin will return the funds according to the amount stated by transferring funds to member's bank account. The business process of donations, starting from members filling out the donation form on one condition that they already have the contact they are going to, if they have not added contacts on the contact page. If they do, members can send donations to the selected contact, after filling out the form. After that the system will send donations to the destination. The business process of adding contacts starts from the member choosing which contacts to add or store as their contacts, then the request will be sent to the intended contact. The intended contact can accept the request or reject the request. Contact requests can be seen on My Contact > request page.

Use Case Diagram

The use of the Unified Modeling Language (UML) can help illustrating complex problems so that they are easier to learn and understand. And UML can also facilitate the communication between software and business processes by describing the system in detail to enable

understanding of system requirements (P. P. Widodo & Herlawati, 2011). In this study, the authors will describe the Use Case Diagrams and Class Diagrams of the designed application. Use Case diagram is used to describe the functional requirements of the system, describes the interaction between the users of the system with the system itself, and describes the relationship between the user (actor) to each of the functions in the system (Fowler, 2005). Figure 2 below illustrates the Use Case Diagram of the designed application.

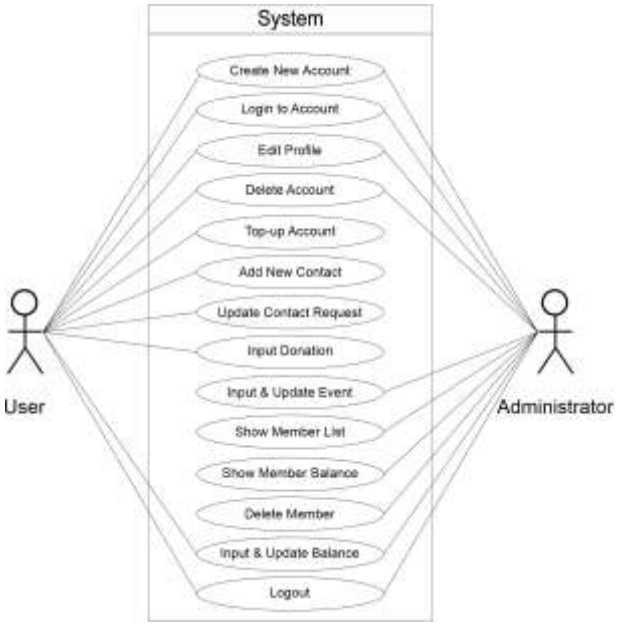


Figure 2. **Use Case Diagram**

Class Diagram

Class diagram according to Munawar (Munawar, 2005) is a set of similar objects. An object has a state and behavior moment. The state of an object is the condition of that object which is stated in the attributes / properties. While the behavior of an object defines how an object acts and reacts. The illustration about Class Diagram used in this System can be seen on Figure 3.

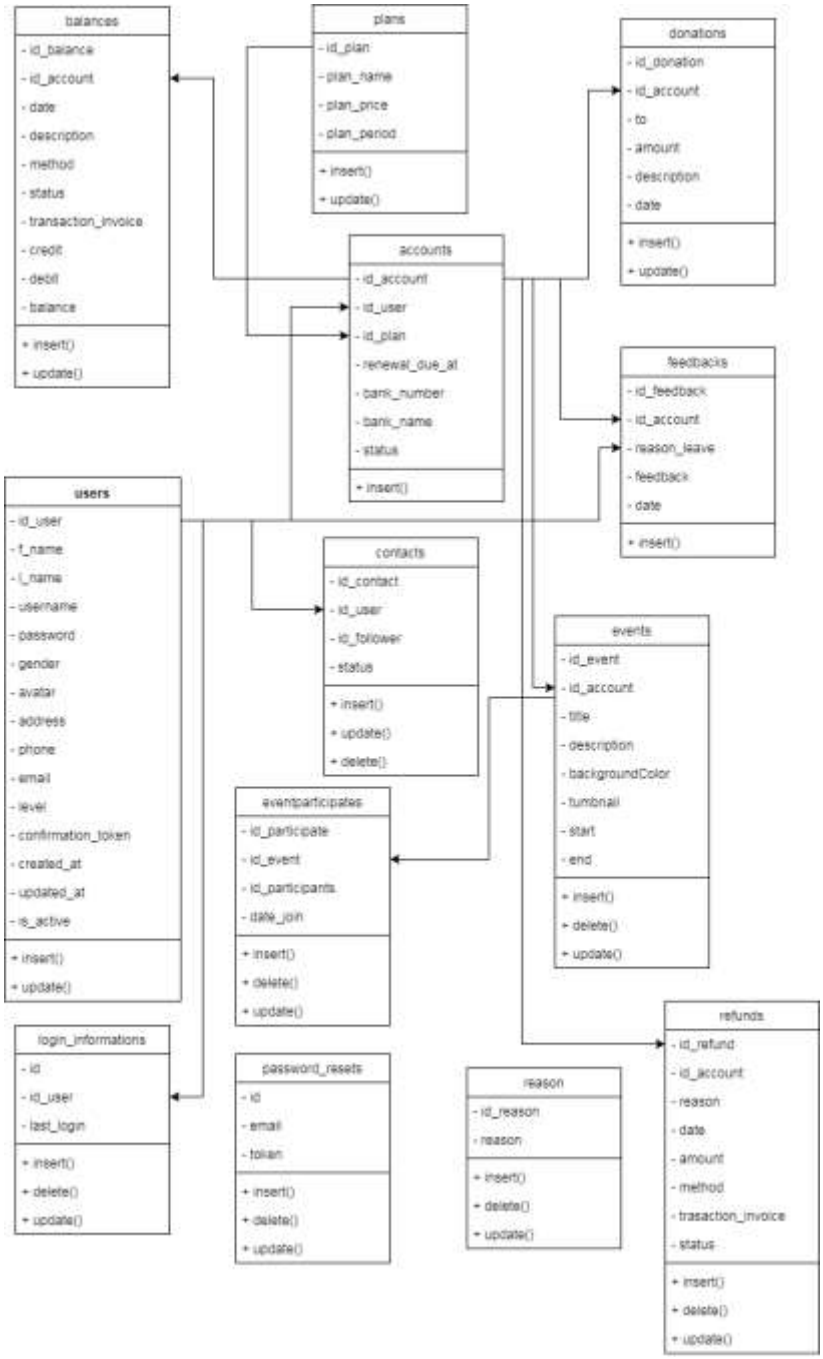


Figure 3. **Class Diagram**

Development Environment and Requirements

Information system development environment is one of the important factors, where the system that is developed has a need for several Software or Hardware configurations. In developing this information system, the author uses a computer with Windows 10 operating system, uses a local database server XAMPP, uses a MySQL database, the text editor used is Microsoft

Visual Code, the main programming language used is Node Js, and uses Adonis Js as a framework from the basic Node Js system.

## RESULTS

The system, which was designed based on the waterfall methodology, resulting in a working web-based membership information system using Node Js. Presentation of research results is divided into two parts, namely the results of the design which contains the appearance of the web, and the results of testing and evaluation of the system that has been conducted.

Design Result

The results of interface design of membership information systems using Js Node can be seen in Figure 4, 5, 6 and 7 below.

Figure 4. **Login Page**

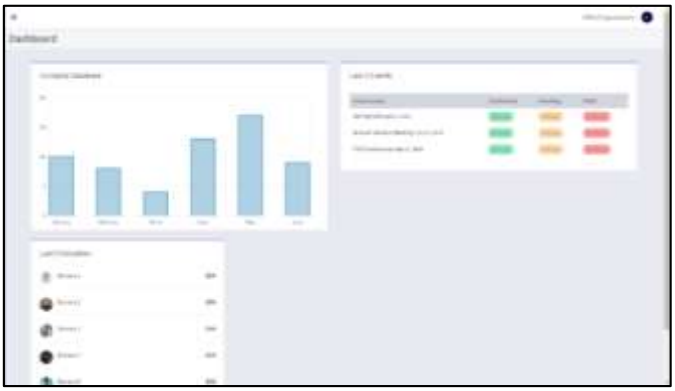

Figure 5. **Administrator Dashboard**

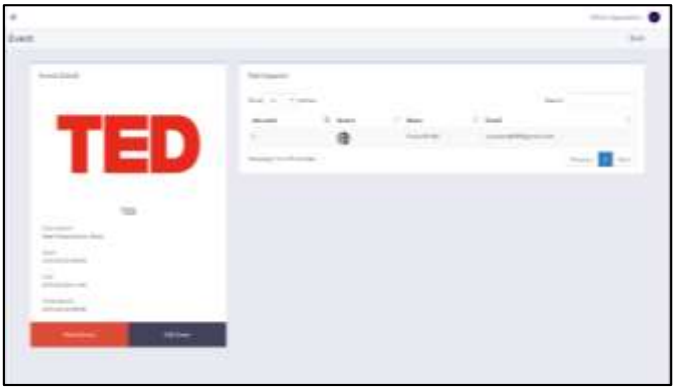


Figure 6. **Event Detail Page**

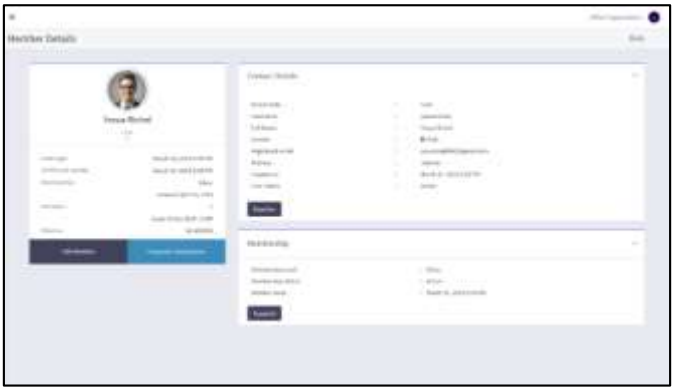

Figure 7. **Member Detail Page**

System Evaluation and Testing

Testing uses a set of validation activities, using the black box testing approach. According to Rosa and Shalahudin (Rosa & Shalahudin, 2014), black box testing is evaluating the software in terms of functional specifications without testing the design and program code. The test is intended to determine whether the functions, input, and output of the software comply with the required specifications. Black box testing is done by making a test case that evaluates all the functions by using software whether it meets the required specifications. Test cases developed in order to conduct black box testing must be made with both true cases and false cases. The author uses 2 black box methods that can help carry out testing on applications that have been developed; which is the Equivalence Class Testing Method and the Error Guess Method.

The first testing of the system using equivalence class testing on Membership Information System from the perspective of an Administrator. Table 1 below show the results of the testing.

Table 1. **Equivalence class testing method of Membership Information System (Administrator)**

| No. | Test Data | ***Input*** | Expected Results | ***Output*** | Conclusion |
|---|---|---|---|---|---|
| 1. | *Login admin* | Input *username* & *password* | Successfully *login* and *redirected* to admin dashboard page | successfully *login* and redirect to admin *dashboard page* | *Valid* |
| 2. | *Input plan* | Input *plan name, plan price, plan period* | *Plan data* is stored to *database* | Show a notification that indicates plan data was successfully stored to database | *Valid* |
| 3. | *Edit plan* | Update *plan name, plan price, plan period* | Plan data is changed to updated value from input | Show a notification that indicates plan data was updated in database | *Valid* |
| 4. | *Delete Plan* | Clicking delete plan button | Plan data will be deleted from database | Show a notification that indicates plan data was deleted from database | *Valid* |
| 5. | *Input Event* | Input event data on provided form | Successfully add Event data into database | Show a notification that indicates event data was stored to database | *Valid* |
| 6. | *Delete Event* | Clicking delete event button | Event data will be deleted from database | Show a notification that indicates event data was deleted from database | *Valid* |
| 7. | *Edit Event* | Update event data on provided form | Event data is changed to updated value from input | Show a notification that indicates event data was updated in database | *Valid* |
| 8. | *Input Member* | Input member data on provided form | Member data is stored on database | Show a notification that indicates member data was stored to database | *Valid* |
| 9. | *Edit Member* | Update member data on provided form | Member data is changed to updated value from input | Show a notification that indicates member data was updated in database. | *Valid* |
| 10. | *Input Balance* | Confirm requested account balance on provided form | requested balance data from user will be validated and accepted/denied | Show a notification that indicates a confirmation of accepted/denied member balance request, and stored in database | *Valid* |
| 11. | *Input Refund* | Confirm requested *refund* on provided form | Request refund data from user will be validated and accepted/denied | Show a notification that indicates a confirmation of accepted/denied member refund request, and updated in database | *Valid* |
| 12. | *Edit Profile* | Input updated administrator data on provided form | Administrator data will be updated in database | Show a notification that indicates administrator data was updated in database | *Valid* |

The second testing of the system using error guess method on Membership Information System from the perspective of an Administrator. Table 2 below show the results of the testing.

Table 2. **Error Guess method of Membership Information System (Administrator)**

| No. | Test Data | ***Input*** | Expected Results | ***Output*** | Conclusion |
|---|---|---|---|---|---|
| 1. | *Login admin* | input *username* = user *password = user123* | *Login failure* | Show a notification that indicates username / password are incorrect | *Valid* |
| 2. | *Input plan* | Input duplicate plan data on provided form | Plan data was not stored in database | Show a notification that indicates plan data was duplicate/not stored in database | *Valid* |
| 3. | *Edit plan* | Input same plan data value as previous data on provided form | Plan data was not updated in database | Show a notification that indicates updated plan data are the same and not updated in database | *Valid* |
| 4. | *Input Event* | Input duplicate event data on provided form | Event data was not stored in database | Show a notification that indicates event data was duplicate/not stored in database | *Valid* |
| 5. | *Edit Event* | Input same event data value as previous data on provided form | Event data was not updated in database | Show a notification that indicates updated event data are the same and not updated in database | *Valid* |
| 6. | *Input Member* | Input duplicate member data on provided form | Member data was not stored in database | Show a notification that indicates member data was duplicate/not stored in database | *Valid* |
| 7. | *Edit Member* | Input same member data value as previous data on provided form | Member data was not updated in database | Show a notification that indicates updated member data are the same and not updated in database | *Valid* |

The third testing of the system using equivalence class testing method on Membership Information System from the perspective of a User. Table 3 below show the results of the testing.

Table 3. **Equivalence class testing method of Membership Information System (User)**

| No. | Test Data | ***Input*** | Expected Results | ***Output*** | Conclusion |
|---|---|---|---|---|---|
| 1. | *Register user* | Input user data on provided registration form | Successfully register and waiting for user email confirmation | Show a notification that indicates registration was successful, and redirect to login page | *Valid* |
| 2. | *Login user* | Input *username* & *password* | Successfully *login* and *redirected* to user dashboard page | Successfully *login* and redirect to user *dashboard page* | *Valid* |

| No. | Test Data | *Input* | Expected Results | *Output* | Conclusion |
|---|---|---|---|---|---|
| 3. | *Input account plan* | Input account plan on provided form | Account plan data was stored in database | Show a notification that indicates account plan was stored in database | *Valid* |
| 4. | *Input Balance* | Input account balance top-up request on provided form | Account balance top-up data will be stored in database after confirmation from administrator | Show a notification that indicates balance top-up request was successfully submitted, and will be stored in database after confirmation from administrator | *Valid* |
| 5. | *Input Refund* | Input refund request on provided form | Deducted account balance data will be updated in database after confirmation from administrator | Show a notification that indicates refund request was successfully submitted, and will be updated in database after confirmation from administrator | *Valid* |
| 6. | *Delete Account* | Input account deactivation request on provided form | Requested account deactivation will be processed, dan account will be deleted in database | Show a notification that account was successfully deactivated, and will be redirected to login page | *Valid* |
| 7. | *Add Contact* | Clicking Add contact button on contact list | Request will be stored in database, dan shown a request notification on target account | Show a notification that indicates add contact request was successfully submitted | *Valid* |
| 8. | *Delete Contact* | Clicking delete contact button on my Contact | Selected contact will be deleted from contact list | Show a notification that indicates contact was successfully deleted | *Valid* |
| 9. | *Accept Contact* | Clicking on select box on contact request page | Selected contact will be added to my contact | Show a notification that indicates contact was successfully accepted | *Valid* |
| 10. | *Join Event* | Clicking on participate button on selected event page | Event participation data will be stored in database | Show a notification that indicates event participation data was successfully submitted | *Valid* |
| 11. | *Edit Profile* | Input updated user data on provided form | User data will be updated in database | Show a notification that indicates user data was updated in database | *Valid* |

The fourth testing of the system using error guess method on Membership Information System from the perspective of a User. Table 4 below show the results of the testing.

Table 4. **Error Guess method of Membership Information System (User)**

| No. | Test Data | *Input* | Expected Results | *Output* | Conclusion |
|---|---|---|---|---|---|
| 1. | *Register user* | Input duplicate user data on provided registration form. | Member data was not stored in Database | Show a notification that indicates data submitted was a duplicate of other user data on database | *Valid* |

| No. | Test Data | *Input* | Expected Results | *Output* | Conclusion |
|---|---|---|---|---|---|
| 2. | *Login user* | input incorrect user login *username* = user *password = user123* | *Login failure* | Show a notification that indicates username / password are incorrect | *Valid* |
| 3. | *Input account plan* | Input an account plan while having insufficient account balance | Account plan data was unable to updated | Show a notification that indicates insufficient account balance | *Valid* |
| 4. | *Input Balance* | Input another account balance top-up request while previous top-up request not confirmed by Administrator | Account balance top-up request will not be stored in database | Show a notification that indicates unsuccessful account balance top-up request | *Valid* |
| 5. | *Input Refund* | Input another account balance refund request while previous refund request not confirmed by Administrator | Account balance refund request will not be stored in database | Show a notification that indicates unsuccessful account balance refund request | *Valid* |
| 6. | *Edit Member* | Input same member data value as previous data on provided form | Member data was not updated in database | Show a notification that indicates updated member data are the same and not updated in database | *Valid* |

# DISCUSSION

After reviewing the development of a web-based membership information system using node js, there are several suggestions for further development of the system in the future. The membership management system that has been designed can be developed and added a notification function so that it enables easy access to the requested data. And it is expected that in future developments, web applications that currently exist can be made in mobile applications, so that the system can be accessed anywhere without using a computer..

## Conclusion

Through the development of the membership information system that has been designed and tested, there are several conclusions from this study. First, the membership information system has been running well and all functions were functioned according to the design in which there is a data collection of members, finance, activities, and donations. Second, the data collection process is carried out by the admin by recording the history of activities of members including registration time, update time, login time, financial history of entry and exit, history of donations and activities participated by members. It is hoped that the design of membership information systems based on Node Js can help the effectiveness and efficiency of the organization member administration.

## REFERENCES

Fowler, M. (2005). UML Distilled, *Panduan Singkat Bahasa Permodelan Objek Standar* (3rd ed.). Yogyakarta, Indonesia: Penerbit Andi.

Graviardhi, K. P., Satoto, K., & Fatur, R. A. (2011). Sistem Informasi Fitness Center Hotel Ciputra Semarang (Thesis (Undergraduate), Universitas Diponegoro). Retrieved from http://eprints.undip.ac.id/25254/

Hammer, M., & Champy, J. (2009). *Reengineering the Corporation: Manifesto for Business Revolution*. US: HarperCollins e-books.

Hutahaean, J. (2015). Konsep Sistem Informasi. Deepublish.

Munawar. (2005). Pemodelan Visual dengan UML. Yogyakarta, Indonesia: Graha Ilmu.

Pressman, R. S. (2012). *Rekayasa Perangkat Lunak: Pendekatan Praktisi* (7th ed.). Yogyakarta, Indonesia: Penerbit Andi.

Rosa, A. S., & Shalahudin, M. (2014). *Rekayasa Perangkat Lunak: Terstruktur dan Berorientasi Objek. Bandung, Indonesia*: Informatika.

Tobing, Y. R. (2017). *Perancangan Sistem Informasi Membership Pada Vitka Fitness Berbasis Web* (Tugas Akhir, STMIK GICI Batam). Retrieved from http://library.stmikgici.ac.id/ tugas_akhir/21000517.pdf

Widodo, A. (2015). Perancangan Sistem Informasi Membership PT. Gold Gym. Jurnal STMIK Pembangunan, 3(1). Retrieved from http://ojs.ipem.ecampus.id/ojs_ipem/index.php/ stmik-ipem/article/view/98

Widodo, P. P., & Herlawati. (2011). Pemodelan Sistem Berorientasi Obyek Dengan UML. Yogyakarta, Indonesia: Graha Ilmu.