Applied Information Technology And Computer Science Vol. 0 No. 0 (2000) 1439-1450
© Universiti Tun Hussein Onn Malaysia Publisher's Office

**AITCS**

Homepage: http://publisher.uthm.edu.my/periodicals/index.php/aitcs
e-ISSN :2773-5141

# A Development of Core Assets Identification System for Software Product Line

**Siti Sarah Omar**[1], **Rabatul Aduni Sulaiman**[1]*

[1]Faculty of Computer Science and Information Technology,
Universiti Tun Hussein Onn Malaysia, Parit Raja, Batu Pahat, 86400, MALAYSIA

DOI: https://doi.org/10.30880/aitcs.2022.03.02.089
Received 00 Month 2020; Accepted 01 Month 2020; Available online 02 Month 2020

**Abstract**: The Software Product Line (SPL) is a collection of features that may be used to efficiently establish the software set in managing feature specific demands for producing common core assets products and reviewing software testing methods. Due to large sizes of products that are continuously being developed, testing is used to manage the core assets variability and commonality using effective ways in SPL. The framework for incorporating core assets into a strategic reuse concept, which includes critical activities such as core asset development, product development, and management in specific practice areas. SPL consists of two processes which are domain engineering and application engineering. Developing proper testing techniques to decrease and eliminate incorrect results with less efforts is critical to having manageable core assets. In software product lines, reusability is widely used to assess respective core assets.

**Keywords**: Information system, web-based, registration, product line

## 1. Introduction

Software Product Line (SPL) is a set of features that is used effectively to determine the software set in managing feature specific needs for developing common core assets product in evaluating ways of software testing. SPL managed sets of features in satisfying the needs of a specific requirements and objectives to be developed from a set of core assets in a prescribed way[1]. The framework of associating core assets product into strategic reuse concept including essential activities for example core asset development, product development and management in certain practice areas. SPL consists of two processes which are domain engineering and application engineering. Domain engineering is used to develop in identifying suitable core assets in reuse products. These products can be reused for developing product lines by using core assets. Meanwhile, application engineering used to perform requirements analysis in product development by using the core assets.

In SPL, the focus is to find the variability among commonality of products. Most products were invented to get the same result and require a mechanism to differentiate the characteristic function. Core assets are defined conceptually as reusing assets among products to identify the feature product line[2]. The important of having manageable core assets are developing proper testing technique to reduce and avoid error result with fewer efforts. Reusability is frequently used to evaluate respective core assets in

*Corresponding author: rabatul@uthm.edu.my software product lines. A framework is to enhance the selection of core assets for Model-Based Testing (MBT) in software product lines[3]. This approach helps the identification of appropriate core assets. MBT includes the information of variabilities and commonalities that illustrates core assets product.

Early detection of reusable life cycles offers extra benefits to the investment of core assets development and in other processes. The main purpose of the SPL is to build a set of products that allows to differentiate between current product assets including domain and application engineering. In real industry that implemented SPL, they faced difficulties in finding which core asset suits for development testing. Usually, multiple core assets are process to run testing when new product is developed. It leads to time wasted when extending the time to do re-testing and further effort needed. For instance, the mobile phones industry in SPL has various types of mobile phone models with multiple functionalities and names. Currently, mobile phones are continuously developed which initiate difficulties when choosing the suitable core assets appeal and differences in between for the company.

Hence, a core assets identification system in SPL will be proposed to help users to achieve the objectives of developing the guidelines to identify a suitable core asset based on reusability matric in SPL. Objectives of this project are defined as follows:

i. To design a system in identifying suitable core assets in Feature Model for SPL Testing.
ii. To develop a system with guidelines for core assets validation in Feature Model for SPL Testing.
iii. To test the developed system by using SPL smart farming core assets case studies.

This paper is organized into five sections. The first part is an introduction describing the context of the project. The second section describes the related work. In the third section, the methodology is explained. The result and discussion are described in the fourth section. In the last section, a conclusion

## 2. Related Work

### 2.1 Software Product Line (SPL)

Software Product Line (SPL) described as a set of application products which are defined the configuration using reusable software elements[4] to satisfy the specific needs of a particular market requirement in developing the core assets. In general, these products have the same functionalities, but some features may be different in certain aspects of system performance. All features of an SPL and the limitations assigned among features are represented by a feature model to describe in handling the commonalities with the variabilities of an SPL. A feature model (FM) defined the collection of products within an SPL by identifying the constraints among features thereby determining which combinations of features are valid. Reduce costs development, enhance software reusability, and promote maintenance to shorten marketing time can be listed as the advantages of SPL usage by software engineers[5]. SPL engineering provides software endless variations by embracing the patterns of number for software system corresponding to an SPL and resolving the differences among products. Domain engineering and application engineering are the two processes which constitute with SPL

### 2.2 SPL Process

The SPL development is typically involving each development process which affects all activities of the development itself in accorporating the products made. The framework helps the identification of variability and choosing pattern of design representation. SPL approaches process is divided into domain engineering and application engineering. The advantage of development processes is product generation costs will be lower in building the system.

Domain engineering aim to reduce the amount of effort needed in implementing software while cutting cost and time saving of applying multiple scratch software when it is not required for single development[6]. In domain development, the process of producing core assets is determined by selecting a reusable process of developing the product. From the domain analysis, common domain requirements and information are identified which domain knowledge is accumulate for the next model process. Domain design or feature model establish to decide which features components are needed in the reference of architecture product line. Next, domain implementation is when system building, and infrastructure supports are processes in domain design as a complete software element[7]. In summary, domain engineering holds the collecting, organizing and stores memory of previous systems builds to form reusable core assets in providing reuse assets to build new system.

### 2.3 Model-Based Testing in SPL

Model Based Testing (MBT) is one of the techniques used to handle SPL testing where the requirements of the test model are identified for software product. Test cases can be generated based on uploaded model. Model based provides advantage in automated generation of test cases as no manual operations needed for abstracting the models. In the domain engineering, test model consists of the behavior variabilities and commonality that are used to generate the test cases. Furthermore, MBT can be implemented into application engineering where the requirements are bind in initiating product lines.

### 2.4 Software Product Line Online Tool (SPLOT)

Software product line online tools serves for product line systems which allows a web-based reasoning and configuration. The system is written in Java and use HTML based user interface engine. SPLOT also provides a database for the feature models that may be useful for effective line researchers[8]. SPLOT is use as an algorithm to build the feature model that contributes to complete the comparison of existing systems.

### 2.5 Comparison with the Existing Systems

The tool supports are which existed in current environment that are run by researchers before. The development of the product evaluation is depending on each tools compile and the functionalities by every software tools. Table 1 shows the example of tools existed and used before by the researchers.

**Table 1: Systems Comparison**

| Features/System | SPLOT | MetaEdit+ | FeatureIDE | Proposed System |
|---|---|---|---|---|
| Login | Unavailable | Available | Unavailable | Available |
| Uploading Module | Available | Available | Available | Available |
| File Management Module | Available | Available | Available | Available |
| Technique | Feature Model | Domain Specific Model | Feature Model | Feature Model |
| Reporting Module | Available | Available | Available | Available |
| Programming Language | Java | C | Java | C++ |
| Requirement Collection Module | Available | Available | Available | Available |
| Evaluation Module | Available | Available | Unavailable | Available |

## 3. Methodology

The system development consists phases in selecting proper development model that suitable for the framework project. The project approach identified that prototyping model is chosen to be implemented in the system. The prototyping model are often used to develop, tested, analyzed, and refined until suitable prototype define with the requirements needed. Figure 1 shows the phases involved starts with planning, analysis, design, implementation, and testing. Each phase in the prototype model has specific function represented to help in encourage the project development. To build the final framework, phases are taken to derive the core assets reusability tool-based framework furthered explain in the next section.

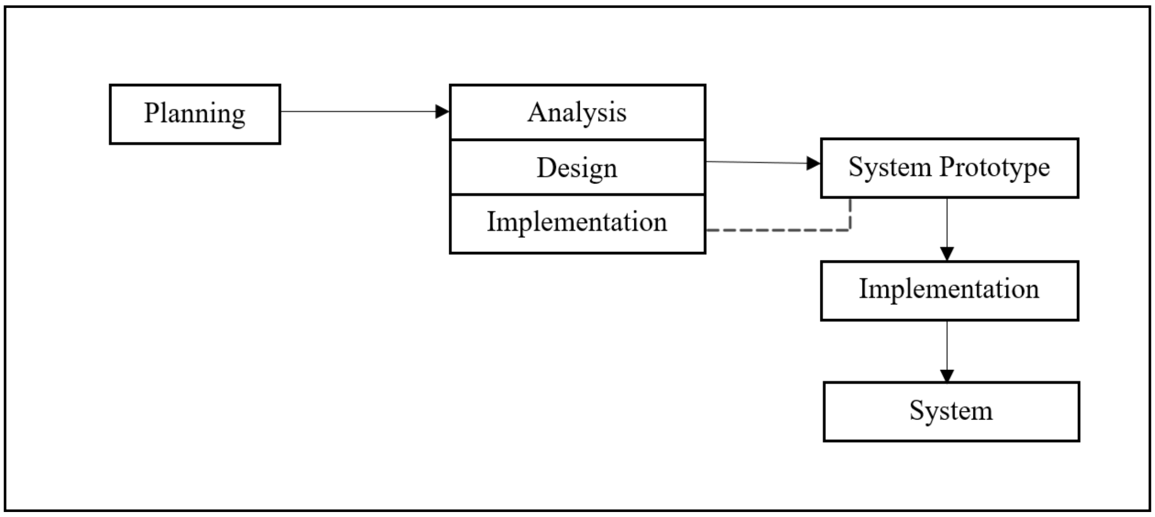


**Figure 1: Prototype Model**

## 3.1 Planning Phase

Identify key features needed to build the tool-based system. In this phase, the issues, and problems for core asset management in SPL are identified through research from the existing journal. The objectives and project scope are finalized by analyzing existing systems to evaluate core assets in SPL. Project planning and proposal are documented the existing problem of the project. In addition, this documentation also describes the requirements of the project. Project milestone is also defined in this phase to have clarification of the project planning which consists of activities and the duration of tasks based on start and end date. Planning phase is needed in providing information and requirements of the project.

## 3.2 Analysis Phase

Based on the planning phase, next phase which is analysis phase will be started. In this phase, the problem related to core assets in SPL is formulated. Next, the objectives are identified based on problem formulation. From the planning phase, information and demands of the system are analyzed detail in this phase. Eight modules are identified which are requirement collection, test model development, registration, uploading, file update, file management, reporting, history, and evaluation module. The study of understanding important requirements and functionalities aspects is gathered to build the system during planning and analysis phases. In this section, the system design of the proposed system is described. Object oriented approach is used to generate the UML diagrams which are Use Case Diagram, Activity Diagram, Sequence Diagram and Class Diagram. Figure 2 shows the main system requirement that represent overall system activity of the developed system.

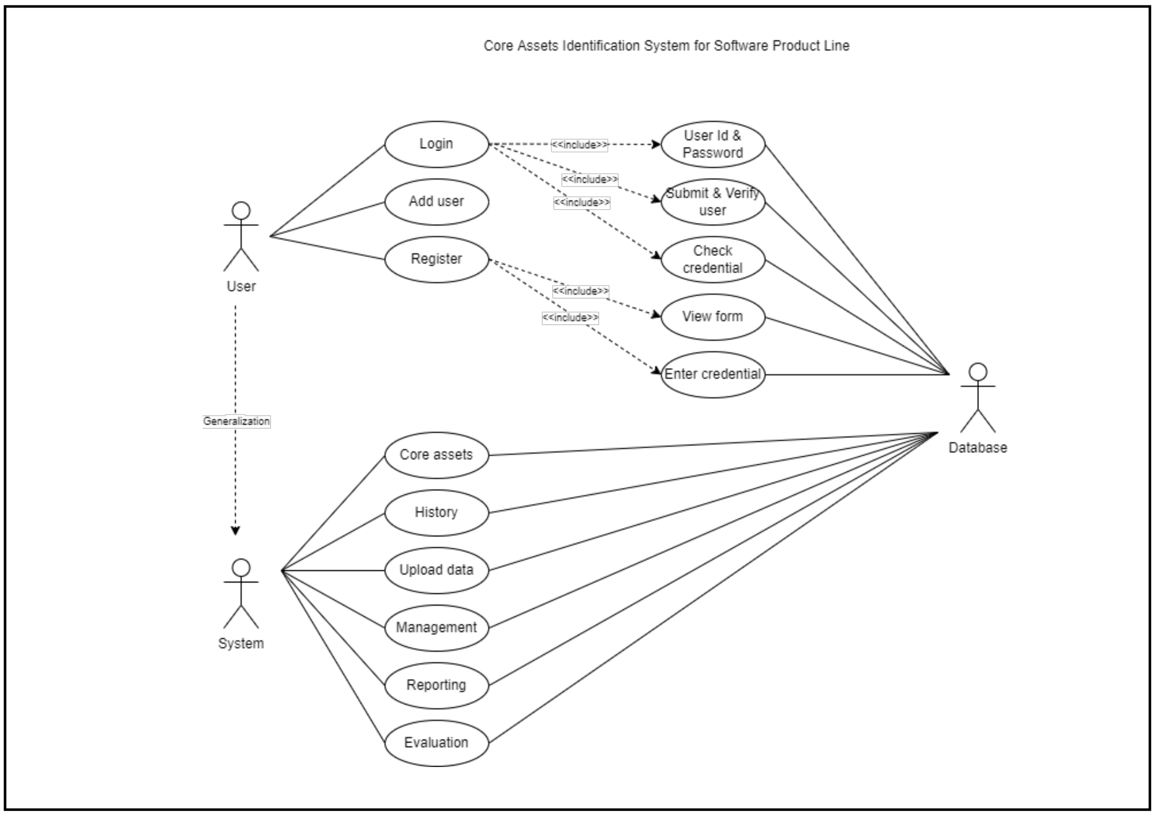


**Figure 2: Use case diagram for core assets identification system**

The Core Asset Identification System's system architecture is illustrated as a diagram which included layers, components and interactions of the web-based elements that interact with one another. As in Figure 3, the system design has two users which are software engineer or user and administrator. User must have their login credential to access the system. After the system validate position of the user, user will be directed into own responsibilities based on their roles. User can access the homepage and evaluate core asset using new uploaded file or current file from the history. Reports are periodically be checked if necessary. The administrator manages entire homepage system and function by performing create, read, update, and delete in the system where the data store in MySQL database. For file modification, it will direct access for permission to the database system and database will return specific data needed to complete the operation.

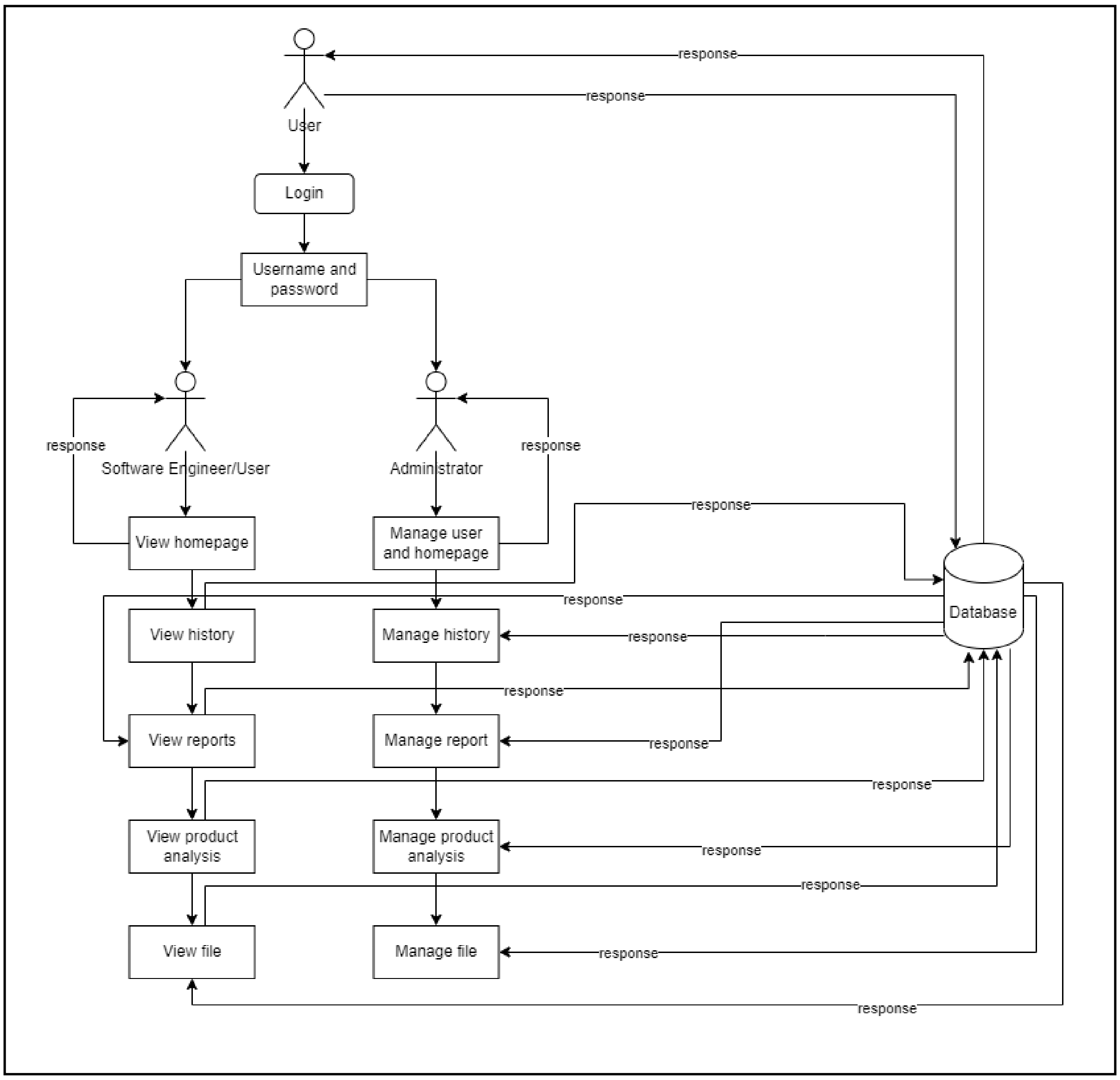
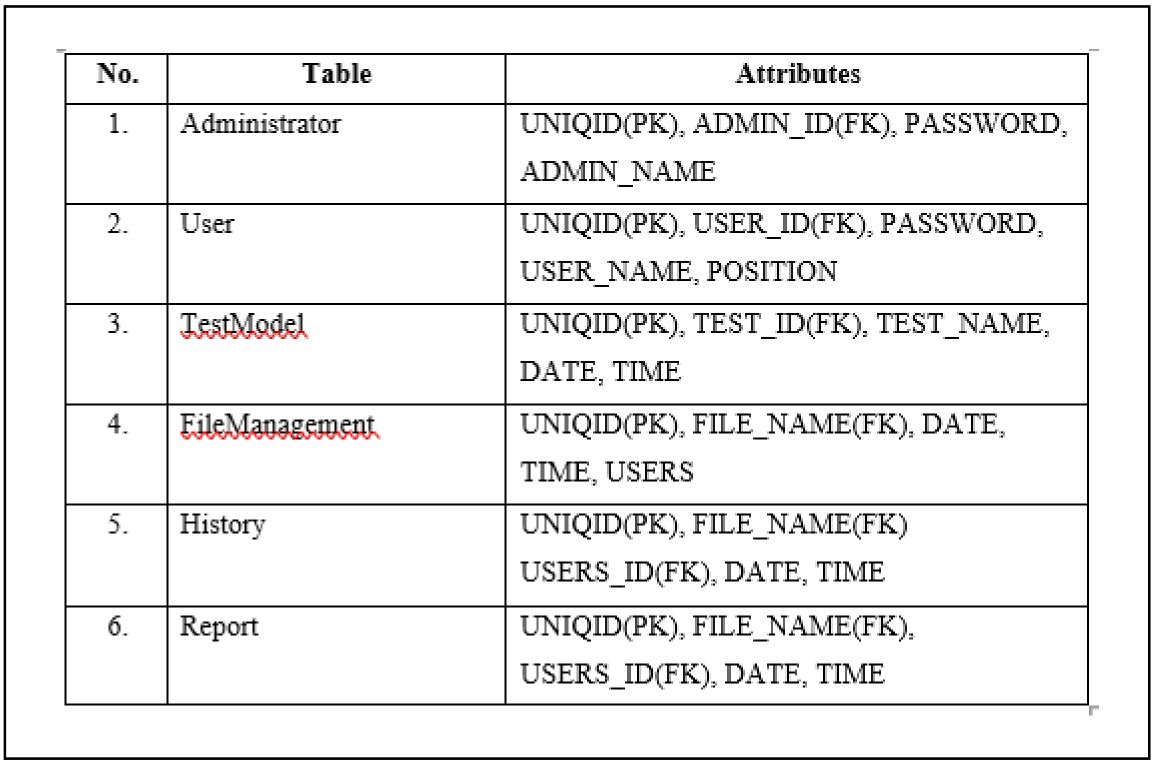


**Figure 3: System architecture for core asset identification system**

## 3.3 Design Phase

The process of specifying a system software and hardware architecture, components, modules, interfaces, and data will be done after user's requirements were successfully analyze. In this phase, both interface and database had been designed to help visualize the system before proceeding with the coding the system. It focuses in determining client requests and functionalities that required, documentation requirements and design followed by validating the system. The following Figure 4 is the database schema which are planned from the database that have been designed and extracted from the class diagram to determine which elements are required and how they will have interacted.



| No. | Table | Attributes |
|---|---|---|
| 1. | Administrator | UNIQID(PK), ADMIN_ID(FK), PASSWORD, ADMIN_NAME |
| 2. | User | UNIQID(PK), USER_ID(FK), PASSWORD, USER_NAME, POSITION |
| 3. | TestModel | UNIQID(PK), TEST_ID(FK), TEST_NAME, DATE, TIME |
| 4. | FileManagement | UNIQID(PK), FILE_NAME(FK), DATE, TIME, USERS |
| 5. | History | UNIQID(PK), FILE_NAME(FK) USERS_ID(FK), DATE, TIME |
| 6. | Report | UNIQID(PK), FILE_NAME(FK), USERS_ID(FK), DATE, TIME |

**Figure 4: Schema Table of the system**

Following is the system interface for the development. User can log into the system using the log-in interface as shown in Figure 5 for user and administrator by entering valid username and password.

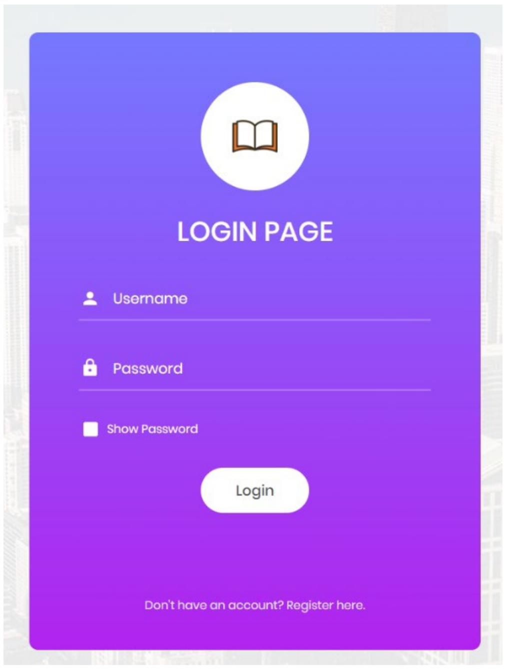
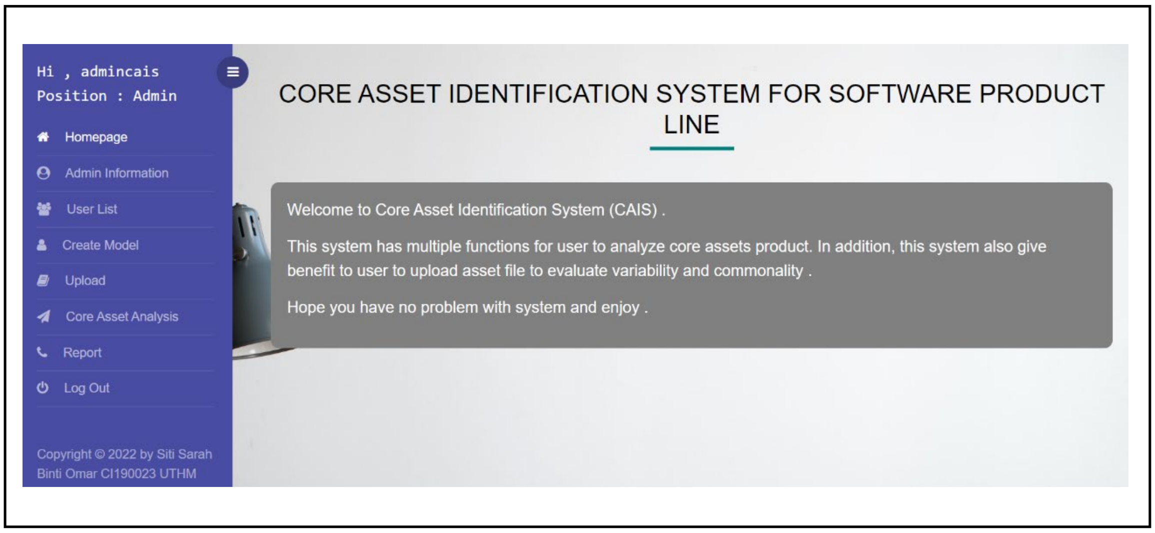


**Figure 5: Login page for user**



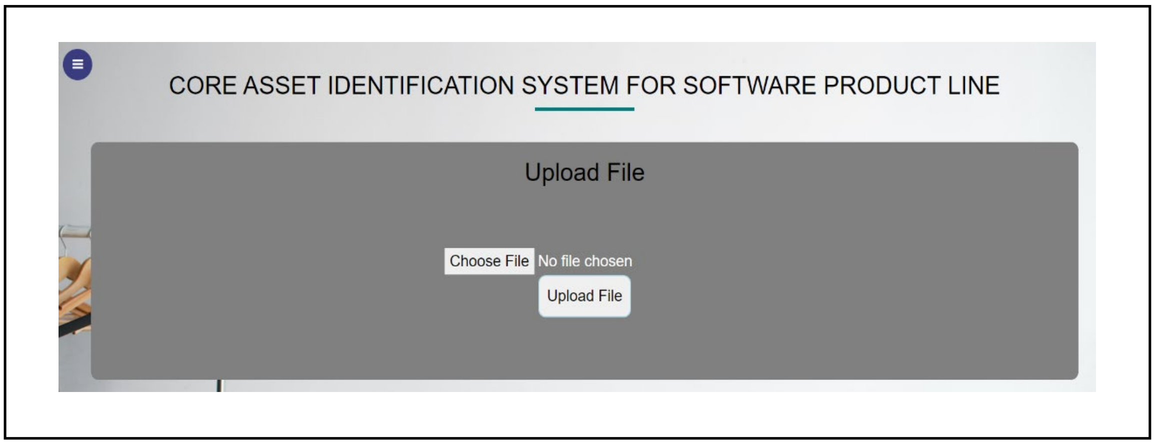
CORE ASSET IDENTIFICATION SYSTEM FOR SOFTWARE PRODUCT LINE

Welcome to Core Asset Identification System (CAIS) .

This system has multiple functions for user to analyze core assets product. In addition, this system also give benefit to user to upload asset file to evaluate variability and commonality .

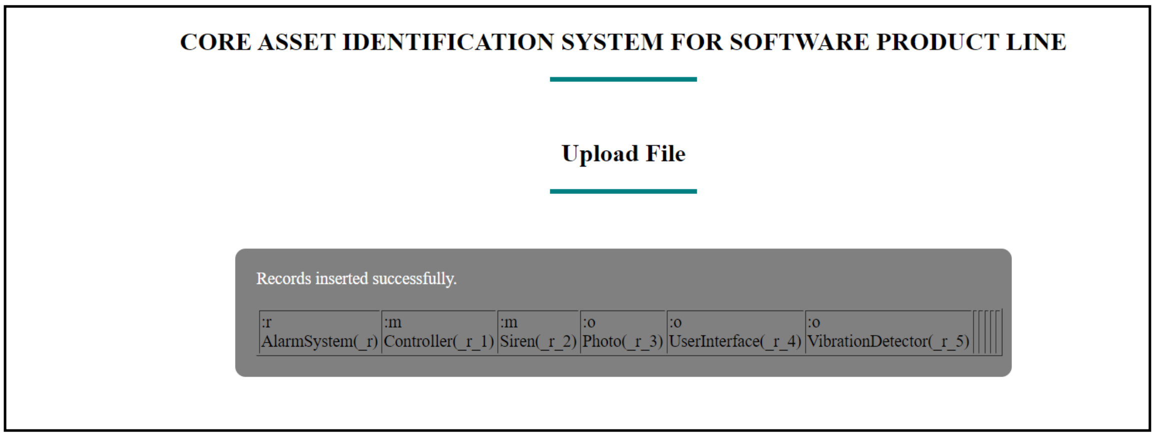
Hope you have no problem with system and enjoy .

Figure 6: Homepage interface of the system


CORE ASSET IDENTIFICATION SYSTEM FOR SOFTWARE PRODUCT LINE

Upload File

Figure 7: Upload file page

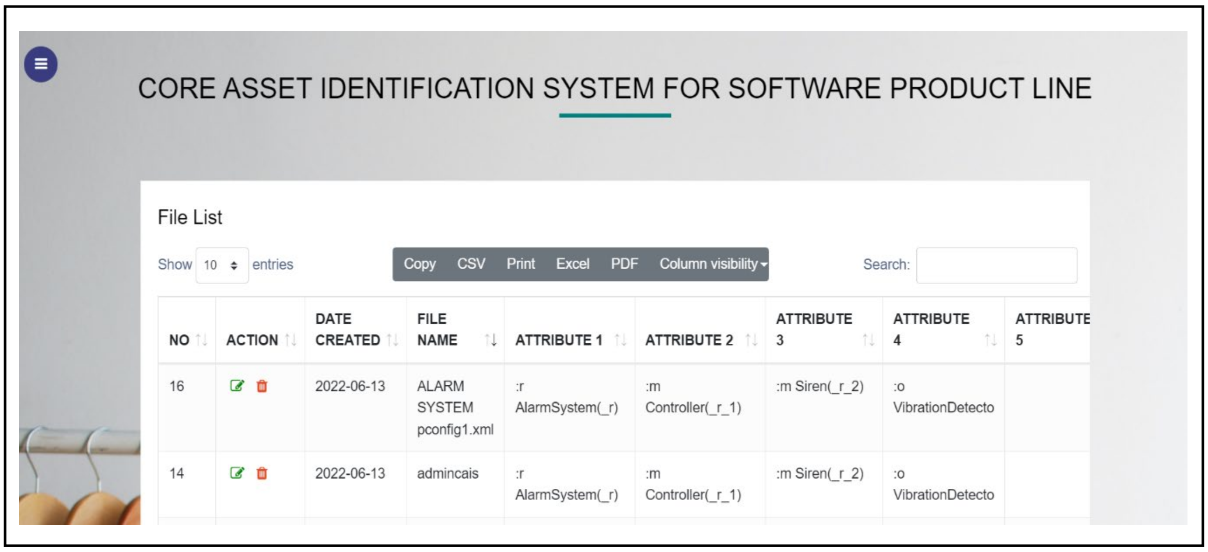

CORE ASSET IDENTIFICATION SYSTEM FOR SOFTWARE PRODUCT LINE

Upload File

Records inserted successfully.

| | | | | | |
|---|---|---|---|---|---|
| :r | :m | :m | :o | :o | :o |
| AlarmSystem(_r) | Controller(_r_1) | Siren(_r_2) | Photo(_r_3) | UserInterface(_r_4) | VibrationDetector(_r_5) |

Figure 8: Output display for uploaded file



| NO | ACTION | DATE CREATED | FILE NAME | ATTRIBUTE 1 | ATTRIBUTE 2 | ATTRIBUTE 3 | ATTRIBUTE 4 | ATTRIBUTE 5 |
|---|---|---|---|---|---|---|---|---|
| 16 | | 2022-06-13 | ALARM SYSTEM pconfig1.xml | :r AlarmSystem(_r) | :m Controller(_r_1) | :m Siren(_r_2) | :o VibrationDetecto | |
| 14 | | 2022-06-13 | admincais | :r AlarmSystem(_r) | :m Controller(_r_1) | :m Siren(_r_2) | :o VibrationDetecto | |

**Figure 9: Table of file list**

Figure 6 shows the homepage of the system with seven option to select in the menu such as create model, upload file, core asset analysis, file management, report generation, and user management. Figure 7 shows the page of uploading file and Figure 8 shows the screen of output which successful uploaded. Figure 9 shows the table of file list with attributes of certain functions.

### 3.4 Implementation Phase

System developments are implemented in this phase. This phase will represent the development results. Evaluate proposed guidelines and research studies referring to several case studies. As the system will be tested multiple time, every version will be added at least one features to meet the objectives. The implementation phase is when the system made to be avail to users by demonstrating the actual system. Maintenance for the system is done such as backup and recovery to enhance the features for each module functions when needed.

### 3.1.5 Testing Phase

The testing phase is to validate the implementation of element functionalities which has applied in the prototype. Each error and failure will be identified and written as note in the test result. User acceptance testing will be used to carry out an effective testing. The stakeholder will give their feedback based on the tested prototype and request for changes if any. New request and demands are analyzed to fit the requirement needed that will be used for maintaining purpose.

## 4. Results and Discussion

The result and discussion section describes the analysis of the study. In this section, the result is explained in quantitative research method as it relates to case studies as shown in Table 2.

**Table 2: Result evaluations**

| | CAIS | FM_State Tool | FeatureIDE |
|---|---|---|---|
| User Interactive | Yes | Yes | No |
| Import files | Yes | Yes | Yes |
| File management | Yes | No | No |
| User management | Yes | No | No |
| Reporting | Yes | No | No |

**Table 3: Test cases of the system**

| Test Case | Test Case ID | Test case description | Test Result | |
|---|---|---|---|---|
| | | | Success | Fail |
| User Login | TEST_UC1_1 | The system able to verify the users. | ✓ | |
| | TEST_UC1_2 | The system should redirect validated users to the respective homepage based on their identity. | ✓ | |
| | TEST_UC1_3 | The system able to reset the form when login is invalid. | ✓ | |
| | TEST_UC1_4 | While exception occurs. The system shall return to previous state. | ✓ | |
| | TEST_UC1_5 | The system should provide a button to indicate user received the system message. | ✓ | |
| User Registration | TEST_UC1_6 | The system allows user to register an account with complete username and password. | ✓ | |
| | TEST_UC1_7 | The system should provide a button to indicate user received the system message. | ✓ | |
| Administrator Login | TEST_UC2_1 | The system able to verify the users. | ✓ | |
| | TEST_UC2_2 | The administrator should enter a valid username and password to success login into the system. | ✓ | |
| | TEST_UC2_3 | The system able to reset the form when login is invalid. | ✓ | |
| | TEST_UC2_4 | While exception occurs. The system shall return to previous state. | ✓ | |
| Uploading File Module | TEST_UC3_1 | The system should be able to upload and view the file. | ✓ | |
| | TEST_UC3_2 | The system should be able to save the file into the database. | ✓ | |
| File Management Module | TEST_UC4_1 | The system should be able to view file list that has been uploaded. | ✓ | |
| | TEST_UC4_2 | The system allows user to update or delete file. | ✓ | |

| Test Case | Test Case ID | Test case description | Test Result | |
|---|---|---|---|---|
| | | | Success | Fail |
| Test Model Development | TEST_UC5_1 | User able to select test model or xml file to be uploaded into the system. | ✓ | |
| | TEST_UC5_2 | The system successfully saves files into the database. | ✓ | |
| | TEST_UC5_3 | Users can view the file uploaded. | ✓ | |
| Reporting Module | TEST_UC6_1 | The system should be able to view report list accordingly. | ✓ | |
| | TEST_UC6_2 | The system should be able to generate document to print the report directly from the system. | ✓ | |

The testing process was held for CAIS to prove in meetings the objectives by solving the common industrial problems based on case studies. Table 3 shows the user acceptability testing based on a set of test cases to utilize the testing approach. The result is judged by satisfaction of user with the outcome of the test case.

## 5. Conclusion

The project framework will improve the base guideline where too many core assets existed leads to difficulty in identifying proper core assets to be use in testing. Continuously applying reusability method in product reuse while evaluating commonalities and variabilities for SPL. Hence, effective approach of core assets is maintained in developing the guidelines tool. Time constraints will be minimized as fewer effort is needed to do multiple runs of testing and high quality assure for compatible guidelines implementation. Enable validation core assets that are mandatorily used for test case generation in SPL. This system also consists of extra functionalities. For instance, upload and update the core assets that help users to easily store and manage test model files.

**Acknowledgment**

The authors would like to thank the Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia for its support.

## References

[1] M. Marques, J. Simmonds, P. O. Rossel, and M. C. Bastarrica, "Software product line evolution: A systematic literature review," *Information and Software Technology*, vol. 105, pp. 190–208, 2019, doi: 10.1016/j.infsof.2018.08.014.

[2] A. R. S. Ali, "A Framework for Evaluating Reusability of Core Assets using SPL and SOA," no. November, 2018, [Online]. Available: http://repository.sustech.edu/handle/123456789/22811

[3] N. Siegmund, M. Rosenmüller, M. Kuhlemann, C. Kästner, S. Apel, and G. Saake, "SPL Conqueror: Toward optimization of non-functional properties in software product lines," *Software Quality Journal*, vol. 20, no. 3–4, pp. 487–517, 2012, doi: 10.1007/s11219-011-9152-9.

[4] Y. Xiang, Y. Zhou, Z. Zheng, and M. Li, “Configuring software product lines by combining many-Objective optimization and SAT solvers,” *ACM Transactions on Software Engineering and Methodology*, vol. 26, no. 4, pp. 1–47, 2018, doi: 10.1145/3176644.

[5] F. Roos-frantz, D. Benavides, and A. Ruiz-Cortés, “Feature Model to Orthogonal Variability Model Transformation towards Interoperability between Tools,” *Knowledge Industry Survival Strategy Initiative, Kiss Workshop @ ASE2009, Auckland, New Zealand*, no. February 2014, 2009.

[6] J. Guo *et al.*, “SMTIBEA: a hybrid multi-objective optimization algorithm for configuring large constrained software product lines,” *Software and Systems Modeling*, vol. 18, no. 2, pp. 1447–1466, 2019, doi: 10.1007/s10270-017-0610-0.

[7] S. Wang, S. Ali, and A. Gotlieb, “Cost-effective test suite minimization in product lines using search techniques,” *Journal of Systems and Software*, vol. 103, pp. 370–391, 2015, doi: 10.1016/j.jss.2014.08.024.

[8] D. C. Marcilio Mendonca, Moises Branco, “S.P.L.O.T Software Product Line ONline Tools,” 2010. http://www.splot-research.org/.