
Indonesian Culinary Application System Design with UML Method

Djarot Hindarto^{1)*}

¹⁾Prodi Informatika, Fakultas Teknologi Komunikasi dan Informatika, Universitas Nasional Jakarta

¹djarot.hindarto@civitas.unas.ac.id

ABSTRACT

Indonesia has an important role as a country that produces various high-quality culinary items. Indonesia, with its plethora of natural resources and cultural diversity, is able to generate a wide range of culinary goods, including traditional foods and beverages as well as modern culinary inventions. Natural and cultural potentials are the foundation for the development of exceptional food and beverages in a country manufacturing culinary product. However, numerous issues with marketing are not global enough, despite the fact that Indonesian culinary items should be able to compete with well-known food products. Information technology and the internet have altered the culinary business scene in Indonesia, making it easier for local culinary products to obtain worldwide market share. This article investigates how the existence of information technology and the internet has opened up new avenues for Indonesian culinary manufacturers to broaden their marketing reach. Indonesian culinary items can now be accessed by consumers all over the world via an online platform, removing geographical barriers and presenting substantial economic growth possibilities. This study created an application system to advertise culinary products, with the first steps focusing on designing application systems. The Unified Modeling Language technique is used to create the culinary application system. This procedure aids in visualizing the system's structure, and interactions deployment. As a result, development teams and stakeholders have a more structured design and a better grasp of the system. Unified Modeling Language is quickly becoming a valuable tool for supporting thorough collaboration and documentation in application development.

Keywords: Culinary Goods; Interactions Deployment; Information Technology; Unified Modeling Language; Visualizing system's structure

INTRODUCTION

Indonesia is widely recognized for its diverse cultural landscape (Winata & McLafferty, 2023), (Estradivari et al., 2022), characterized by many ethnicities and customs. The culinary assortment seen within a specific location serves as a manifestation of its broader diversity. Indonesian cuisine showcases a distinctive flavor profile and is prominent in our cultural history, embracing traditional culinary offerings and modern adaptations. Despite the considerable potential inherent in the culinary industry, it faces several challenges, including promotional techniques, inventory management, ordering systems, and client interaction. In the current era marked by information and communication technology improvements, system applications have become essential instruments for facilitating several sectors of life, including the culinary industry. Indonesia's cultural diversity and gastronomic abundance provide a promising opportunity to advance system applications within the culinary area.

Indonesia is renowned for its rich cultural tapestry, encompassing various races, traditions, and customs. The culinary diversity of any region is indicative of the overall diversity present. Indonesian food exhibits a discernible flavor profile and holds a significant position within our cultural heritage, encompassing conventional culinary offerings and contemporary innovations. Despite its immense potential, the culinary business encounters several problems: promotional strategies, inventory management, ordering processes, and client engagement. In the contemporary period characterized by information and communication technology advancements, system applications have emerged as crucial facilitators in several domains of life, including the culinary sector. Within Indonesia's cultural diversity (Samaddar et al., 2020) and culinary (Sholihah & Heath, 2016), (Sukenti et al., 2016) abundance, there is significant potential for developing system applications in the culinary environment that is expanding rapidly.

Several problems necessitating a structured and visual approach require using the Unified Modeling Language (UML) method in designing Indonesian culinary applications. The successful implementation of culinary techniques necessitates a systematic and coherent methodology that accounts for many aspects, components, and interactions among diverse parts. UML methodologies offer a comprehensive set of symbols and graphical elements that facilitate the management of design intricacies. The absence of complete and organized documentation in the design of culinary

* Corresponding author



This is an Creative Commons License This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0).

applications might result in challenges in comprehending the creation, maintenance, and development of said applications in subsequent periods. The utilization of UML facilitates the generation of comprehensive and easily understandable documentation.

This study uses the Unified Modeling Language (UML) (Bosse, 2018), (Ferreira et al., 2015) method to design an information technology-based Indonesian culinary application system. The choice of UML as the modeling language was based on its robustness and efficacy in visually conceptualizing, articulating, and conveying system designs. In the present context, the application will be developed to encompass diverse information about gourmet products from different regions of Indonesia. This program facilitates the retrieval of information regarding regional specialties, enabling users to access such information conveniently. Designing culinary applications specific to Indonesian cuisine is paramount due to its substantial advantages to several stakeholders. This application will serve as a platform for promoting and introducing Indonesian cuisine worldwide. The utilization of technology facilitates the broader dissemination of Indonesian culinary items to diverse international markets, consequently enhancing the prospects for increased Indonesian culinary exports. Furthermore, this application will also have a beneficial impact on the preservation of Indonesian culinary culture.

The use of UML is limited to the Use Case Diagram and Class Diagram portions of the subject of this study. This research comprises multiple inquiries. The subsequent research inquiries can be presented:

Designing Indonesian culinary applications using Use Case Diagrams involves identifying and representing the many interactions between actors and the system, as well as the functionalities and behaviors of the system. (RQ1).

What primary constituents must be incorporated in Indonesian culinary apps utilizing the Unified Modeling Language (UML) methodology to facilitate culinary promotions and disseminate information about diverse regions? (RQ2).

LITERATURE REVIEW

Numerous researchers have conducted studies utilizing the Unified Modeling Language (UML) methodology. The Unified Modeling Language (UML) is a graphical notation system employed to represent and communicate many aspects of models concisely and visually intuitively. Evaluating the layout quality of UML class diagrams using machine learning (Bergström et al., 2022). This paper aims to showcase the potential for automated assessment of the quality of UML class diagram layouts. It also seeks to examine the characteristics that have the greatest impact on the quality of UML class diagram layouts, assess the effectiveness of the developed layout evaluators, and offer labeled UML class diagram datasets for future research purposes. Automatically recognizing the semantic elements from UML class diagram images (Chen et al., 2022). The objective of this study is to create a systematic approach for the identification of semantic components inside UML class diagram visuals. This method aims to facilitate the utilization and advancement of software design models. The primary objective of this study is to assess the efficacy of the created methodology and conduct a comparative analysis with other established methods. QMaxUSE: A new tool for verifying UML class diagrams and OCL invariants (Wu, 2023). This study aims to present QMaxUSE, a novel verification tool designed for UML and OCL invariant class diagrams. Additionally, this research seeks to demonstrate the utility of QMaxUSE in addressing specific obstacles encountered in the formal verification process within the domain of Model-driven Engineering. Uncertain Decision-Making Requirements Formalizing with Complement Fuzzy UML Model (Abdelmadjid & Mimoun, 2021). In this study, we utilized the CFUML model and employed the representation of decision-maker preferences using one or more mass functions, employing belief theory. This study contributes to developing a versatile approach to articulate decision-making needs by systematically evaluating uncertainty through formalization. This article exclusively focuses on the presentation of our suggested model, CFUML. The CFUML model utilizes belief theory to express decision-maker preferences through one or more mass functions. A review of the generation of requirements specification in natural language using objects UML models and domain ontology (Abdalazeim & Meziane, 2021). This paper provides an overview of several studies conducted in generating natural language specifications from object UML models with the assistance of an ontology. This study examines and evaluates the resilience and constraints of the current methodologies. This encompasses examining natural language generation from a formal model, analyzing natural language generation from ontologies, and evaluating studies about the creation of natural language using OntoUML. Metamodel based approach to generate user interface mockup from UML class diagram (Thomas et al., 2021). This study aims to develop a method for automatically generating Human-Computer Interaction (HCI) mockups using class diagrams, intending to facilitate and expedite software design processes. For instance, the SEF (Schéma d'Enchaînement de Fenêtre) is an interface concept that enables rapid software design. An example of an interface model is the SEF (Schéma d'Enchaînement de Fenêtre), which facilitates the creation of a new interface prototype. It provides more widgets for designing the human-computer interface (HCI).

* Corresponding author



[Creative Commons Attribution-NonCommercial-ShareAlike 4.0
International License.](https://creativecommons.org/licenses/by-nc-sa/4.0/)

The MACAO (Method d'Analyse et de Conception) framework generates a novel interface mockup. It offers more devices for creating the human-computer interface (HCI). Macao (Méthode d'Analyse et de Conception d'Applications Orientées-objects) is a methodology utilized by computer scientists to facilitate the development of software applications. The Model d'Applications Orientées-objects (Model of Object-Oriented Applications) is a strategy utilized by computer scientists to facilitate the development of software. The model transformation process is rooted in Model Driven Engineering (MDE) principles. The UML notation is employed to design class diagrams. Additionally, we embrace. The transformation method is grounded in Model Driven Engineering (MDE) principles. UML notation is used to create class diagrams. In addition, we also embrace or incorporate. The MDA architecture is utilized to provide a model-based transformation process. Atlas Transformation is the tool utilized for the creation of transformation rules. The MDA architecture is used for the development of a model-based transformation process. Atlas Transformation is the tool utilized for the creation of transformation rules. Automatic Transformation of Language (ATL) is a technique used to produce a targeted interface based on a given class diagram. Automatic interface generation from a class diagram is a feature provided by the Language (ATL) framework.

The research gaps that have been found highlight the necessity for a cohesive methodology that utilizes automated approaches for evaluation, verification, transformation, and integration. This methodology aims to enhance the quality and dependability of UML class diagrams in software engineering. The identification and exploration of this research gap have the potential to make substantial contributions to the field of software design, hence enhancing the development of software systems characterized by increased robustness and efficiency. One approach involves prioritizing planning by using Use Case Diagrams and Class Diagrams.

METHOD

The Unified Modeling Language (UML) is a graphical representation tool utilized for system design, description, and communication. The Unified Modeling Language (UML) is a commonly employed standard language in the fields of software design and systems engineering. The Unified Modeling Language (UML) offers a comprehensive and structured visual representation for describing different facets of the system that is intended for development. UML, at its core, serves as a potent means of communication, facilitating collaboration, knowledge sharing, and discourse among developers, analysts, and various stakeholders. Its visual and lucid nature enhances the clarity and comprehensibility of system design discussions. The Unified Modeling Language (UML) is well recognized as a robust instrument in the field of systems development, offering numerous vital advantages. Firstly, the utilization of Unified Modeling Language (UML) facilitates enhanced communication among members of the development team as well as stakeholders. The utilization of precise and organized visual notation in the form of UML aids in the prevention of misinterpretations and guarantees a shared understanding among all stakeholders regarding system design. Furthermore, UML facilitates a more comprehensive comprehension of the system's structure, functionality, and interrelationships. UML diagrams are helpful in providing a more realistic visualization of concepts, expediting the learning process, and facilitating the analysis of needs. Moreover, UML serves as a catalyst for promoting organized planning and design. UML facilitates the development of efficient and easily implementable solutions by offering a well-defined framework for identifying components, relationships, and workflows. This framework aids development teams in mitigating the risk of errors during the first stages of development.

With UML, IT professionals can more effectively communicate, collaborate, and manage complexity in systems development. UML provides a powerful tool for visually representing and designing systems, ensuring that complex concepts can be conveyed in a more structured way and understandable to a broad range of parties involved in the project. UML is not just a set of notations and diagrams, it is also a tool that makes collaboration, communication, and management of complexity easier in software development and systems engineering projects. UML has various types of diagrams that represent different aspects of the system being designed. Here is an explanation of some of the main parts of the UML:

Use Case Diagram:

Use Case Diagrams (Meiliana et al., 2017) are used to describe interactions between actors (stakeholders or users) with the system. This diagram identifies the actions or scenarios that can be performed by actors and how the system responds to them. Using a Case Diagram helps in understanding the functionality of the system from the user's perspective. A Use Case Diagram is a type of diagram in the Unified Modeling Language (UML) that is used to describe interactions between actors (stakeholders or users) with the system to be developed. These diagrams help identify and visualize different usage scenarios, as well as describe how the system will respond to actions or requests

* Corresponding author



[Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.](https://creativecommons.org/licenses/by-nc-sa/4.0/)

from these actors. Use Case Diagram helps in a better understanding of system functionality from the user's perspective. Actors are external entities that interact with the system. These can be users, devices, or other entities involved in interactions with the system. Actors are depicted in the form of human symbols or devices and are given names that reflect their roles. Use cases are scenarios or usage flows that represent actions or activities that can be carried out by actors in the system. Use Case (Swain et al., 2012) describes the actions taken by the user to achieve certain goals in interaction with the system. Each Use Case has a descriptive name. Relationships: In Use Case Diagrams, there are relationships between actors and Use Cases. The main relationship is an association relationship that describes how actors interact with a particular Use Case. In addition, there are also other relationships such as generalization (decrease), extension (expansion), and inclusion (inclusion) which describe the special relationships between use cases. System Boundary: Circles or rectangles around actors and use cases that describe system boundaries or the context in which interactions occur. This helps in understanding the scope of the system being designed. With the Use Case Diagram (Li et al., 2013), the development team can identify and clarify the main usage scenarios, ensure that the developed system meets user needs, and communicate a clearer view of system interaction to all parties involved. Use Case Diagrams help in visualizing how users (actors) interact with the system and describe the main usage scenarios. Use Case Diagram helps in designing the system based on user needs and goals, leading to more relevant solutions.

Class Diagram:

Class Diagrams (Bergström et al., 2022) are used to describe the class structure in the system, including the relationships between classes, attributes, and methods that belong to each class. These diagrams help in visualizing the main components in the system and the interactions between them. Class Diagram (Chen et al., 2022) is a type of diagram in the Unified Modeling Language (UML) that is used to describe the class structure in the system, including the relationships and interactions between these classes. Class Diagram helps in static modeling of the system, namely how the components in the system are interrelated and how data is stored and processed in the system. This diagram is very useful in designing and understanding the structure and hierarchy of components in a system.

Following are some of the main elements in a Class Diagram:

- Class represents an entity that has attributes (data) and methods (functions or operations) associated with it. The class is described in a box with three parts: the class name at the top, the attributes in the middle, and the methods at the bottom.
- Attributes are data owned by a class. These can be variables, properties, or other information relevant to the class. Attributes are described in the form of a name and data type.
- Methods are functions or operations that can be performed by a class. It represents the actions that can be performed on objects of that class. Methods are described in the form of names and parameters.
- There are several types of relationships between classes in the Class Diagram, including association, generalization (decrease), realization (interface realization), and dependency (dependency). This relationship describes how classes interact and relate to each other.
- Multiplicity describes how many objects from one class can be related to objects from other classes in a specific relationship. This is indicated by a number or an "*" at the end of the relationship line.

The benefits of Class Diagrams are:

- Class Diagram (Thomas et al., 2021) helps in visualizing the structure of the components in the system and the existing class hierarchies.
- This diagram helps in understanding how classes relate to each other, including the relationship between parent and child classes in the inheritance hierarchy.
- Class Diagram helps in designing the data structure to be used in the system, including the required attributes and data types.
- This diagram helps in planning the system design in a structured manner, enabling the development team to divide and group related functions into separate classes.
- Class Diagrams serve as clear documentation of the structure of the system, facilitating future development, maintenance, and changes.

With the Class Diagram, the development team can design an organized and efficient structure in the system, describe the class hierarchy and the relationships between them, and understand better how data and functions are interrelated

* Corresponding author



in the system to be developed.

Sequence Diagram:

A sequence Diagram is used to describe the flow of messages or interactions between objects in the system all the time. These diagrams help in understanding the sequence of actions or messages that occur in a given scenario. A sequence Diagram is a type of diagram in the Unified Modeling Language (UML) that is used to describe interactions between objects in the system chronologically, showing the messages or method calls sent between these objects. Sequence Diagram helps in modeling the flow of execution in a scenario, thereby enabling a better understanding of how the objects interact with each other in the system.

Here are some of the main elements in a Sequence Diagram:

- Objects are entities in the system that participate in interactions. These can be instances of classes or actors involved in scenarios. Objects are depicted as vertical blocks with the object's name above them.
- Lifeline is a dotted line that connects objects with messages or method calls. The lifeline describes the lifetime of the object in the scenario and shows how long the object is active in the interaction.
- Messages are communications sent between objects. Messages represent method calls or messages passing from one object to another. Messages are depicted as arrows with arrows pointing to the receiving object.
- Activation describes the period of time when the object is receiving or responding to a message. A vertical line on the lifeline of the object in question indicates activation.
- Return Message is a reply message sent by the receiving object in response to the original message. It shows the return value of the called method.

Sequence Diagram:

Activity Diagram is used to describe workflow or business processes in the system. This chart shows the activities, decisions, and actions taken in a given scenario. A sequence Diagram is a type of diagram in the Unified Modeling Language (UML) that is used to model the interactions between objects in the system chronologically. These diagrams provide a visual view of how objects communicate and interact with each other in a given scenario. Sequence Diagrams are primarily used to model the flow of system execution in response to messages or method calls passed between objects.

Activity Diagram:

An activity diagram is a type of diagram in UML that is used to model the flow of processes or activities in the system. These diagrams are helpful in depicting actions and the flow of work from one activity to another, including branch selection, looping, and parallel activities. Activity diagrams help in visualizing how objects interact in the form of activities, as well as how the flow of execution from one activity to another. The elements in the Activity Diagram include activities, transitions, forks and joins for parallel activities, decisions for branch selection, and start and end nodes to mark the beginning and end of activities. This diagram facilitates the modeling of workflows or processes in the system, helps in understanding the interaction of objects in the form of activities, as well as analysis of process optimization in the system. Activity diagrams are helpful as a communication and collaboration tool between development teams, analysts, and stakeholders, thus facilitating an understanding of the workflow of the system. This diagram also helps in optimizing workflows or processes in the system, so that potential problems or opportunities can be identified to increase efficiency. With Activity Diagrams, the development team can better design workflows or processes in the system, understand the interaction of objects in the form of activities, and support the analysis and optimization of processes in the system. This diagram plays an important role in supporting the development of an efficient and structured system.

State Diagram:

State Diagrams are used to describe changes in the status or state of objects or classes in the system. These diagrams help in modeling how objects move from one state to another based on certain events. State Diagram is a type of diagram in UML that is used to model various states (states) that can be experienced by an object or entity in the system as well as state changes that occur based on events (events). This diagram depicts an object's state as a circle with labels describing its state, and transitions between states as arrows labeled as triggering events that cause changes. The system or external environment can generate events. State Diagrams help visualize an object's interactions with its environment, understanding state changes and transition paths that objects may take. These diagrams also play a

* Corresponding author



role in system testing and debugging and as a communication and collaboration tool between the development team and stakeholders. With State Diagrams, development teams can better design object state changes and transitions, understand system dynamic behavior, and support efficient system development and maintenance processes.

Component Diagram:

Component diagrams are used to describe the structure of the physical or logical components in a system, as well as the relationships and dependencies between these components. A component Diagram is a diagram in UML that is used to model the physical structure of a software system or component-based system. These diagrams help in describing the components of the system and the relationships and interactions between them. Each component is a separate, independent unit within the system, which can be a class, package, or external system. Interface is used to describe a contract defined by a component to interact with other components. Dependencies show the relationship between components that are sheet, where a component uses or depends on another component without depending on its internal implementation. Realization is a relationship between a component and an interface that indicates that the component implements (fills in) a particular interface. The benefit of a Component Diagram is to understand the physical structure of the system, model the integration and interaction between components, and support system planning and design. These diagrams serve as a means of clear documentation of the physical structure of the system, facilitating collaboration and understanding between development teams. In addition, Component Diagram also helps in the analysis of dependencies between components and helps in identifying potential problems or errors in the system design. With the Component Diagram, the development team can better design the physical structure of the system, model the integration and interaction between components more efficiently, and support an effective design planning and analysis process. These diagrams become an important tool in the development of component-based systems and help in ensuring the system runs in an organized and coordinated manner.

Deployment Diagram:

Deployment diagrams are used to describe how system components are implemented in the physical environment, such as servers, hardware, and networks. Deployment Diagram is a UML diagram that is used to model the physical distribution of software components in a physical environment. This diagram helps in visualizing how the system components are implemented on the hardware and network infrastructure used. The elements in the Deployment Diagram include nodes that represent hardware or network nodes, components as software units, artifacts that are physical representations of components, associations that show relationships between nodes and components, and dependencies that describe relationships between components. This diagram helps in understanding the configuration and implementation of software in physical nodes, as well as mapping the components on the appropriate infrastructure. Deployment Diagrams also serve as a communication and collaboration tool between development teams, analysts, and stakeholders, facilitating understanding of system distribution and implementation. In addition, these diagrams also serve as clear documentation of the physical distribution structure of the software system, aiding in development, maintenance, and future changes. With Deployment Diagrams, development teams can plan and analyze how system components are distributed in the physical environment, understand component relationships and implementation, and support an efficient and structured infrastructure planning process. These diagrams become an important tool in the development and implementation of software systems, thus ensuring the system can be properly implemented in a suitable physical environment.

Package Diagram:

Package diagrams are used to organize and group elements in the system into packages or modules. Package Diagram is a type of diagram in UML that is used to organize and describe the organizational structure or hierarchy of elements in a software system. This diagram uses packages as an organizing unit for related elements, such as classes, objects, use cases, and components. Package diagrams help in understanding the interrelationships and dependencies between packages in the system, enabling efficient and well-organized structural modeling. These diagrams also serve as a communication and collaboration tool between development teams, facilitating an understanding of how system elements interact and relate. Package diagrams serve as clear documentation of the organizational structure of a software system, assisting in planning, developing, maintaining, and changing in the future. With the Package Diagram, the development team can design an efficient organizational structure, group software elements properly, and support the system development process in an efficient and structured manner.

* Corresponding author



Object Diagram:

Object diagrams are used to describe the concrete instances of the classes in the system and the relationships between these instances. An object diagram is a type of diagram in UML that is used to model the static structure of a system at a certain point in time. This diagram displays the objects that exist in the system along with the attributes and relationships that are owned by these objects. Objects are described by their names and attributes, and the relationships between objects can be association, composition, or aggregation. The benefits of Object Diagrams are as a tool to understand the static structure of the system, describe the relationships between objects, and represent instances of classes in the system. These diagrams assist in system testing and debugging and are used for communication and collaboration between the development team and stakeholders. With the Object Diagram, the development team can analyze the components and relationships in the system at a certain point in time, and get a clear picture of the concrete state of the system. This diagram has an important role in the system design planning and analysis process.

RESULT

The study's findings employing the Unified Modeling Language for Use Case Diagrams and Class Diagrams are depicted in Figure 1 and Figure 2, respectively. The talk concerns two specific domains: Use Case Diagrams and Class Diagrams. The utilization scenario A schematic representation of an online meal ordering system illustrating the dynamic exchange between actors (users) and the design across various usage scenarios. The primary performer in this context is the "User," who can engage in the "Food Order" process through menu selection, cart addition, and order confirmation. The system can effectively handle the "Menu List" by presenting a comprehensive array of meal options. The actor, "Admin," can manage the menu by adding, changing, or deleting menu items. The utilization scenario The figure presented above facilitates the visualization of the interaction flow among users, food menus, and online ordering systems, hence aiding in comprehending the requisite system functioning.

The class diagram of an online food ordering system delineates the arrangement of classes within the system and the interconnections among these classes. The diagram encompasses several classes, namely "User," "FoodMenu," "Shopping Cart," "Orders," and "Admin." The "User" class has characteristics including name and address, as well as methods such as "FoodOrder()." The "FoodMenu" class possesses attributes, including the name of the meal and its price, as well as methods such as "AddMenu()" and "RemoveMenu()." The class named "Shopping Cart" has characteristics like lists of menus that have been selected by the user, as well as methods such as "AddToCart()" and "RemoveFromCart()." The "Order" class encompasses several characteristics, such as the date of the order, and methods, such as "ConfirmOrder()." The interconnection of these classes is conveyed by associations, wherein the class "User" can establish a link with both "Shopping Cart" and "Orders." In contrast, the "Shopping Cart" is associated with the class "Food Menu." The "Admin" class possesses the necessary privileges to oversee the management of the "MenuFood" entity through a suitable association. Furthermore, the "User" class can establish a relationship with the "Admin" class to facilitate communication and interaction between users and administrators. The presented Class Diagram simplifies designing class structures inside online food ordering systems. It visually represents the interconnections of classes, properties, and methods essential to the system's functionality. The explicit definition of the relationships and interactions between elements in the system facilitates both system development and maintenance.

* Corresponding author



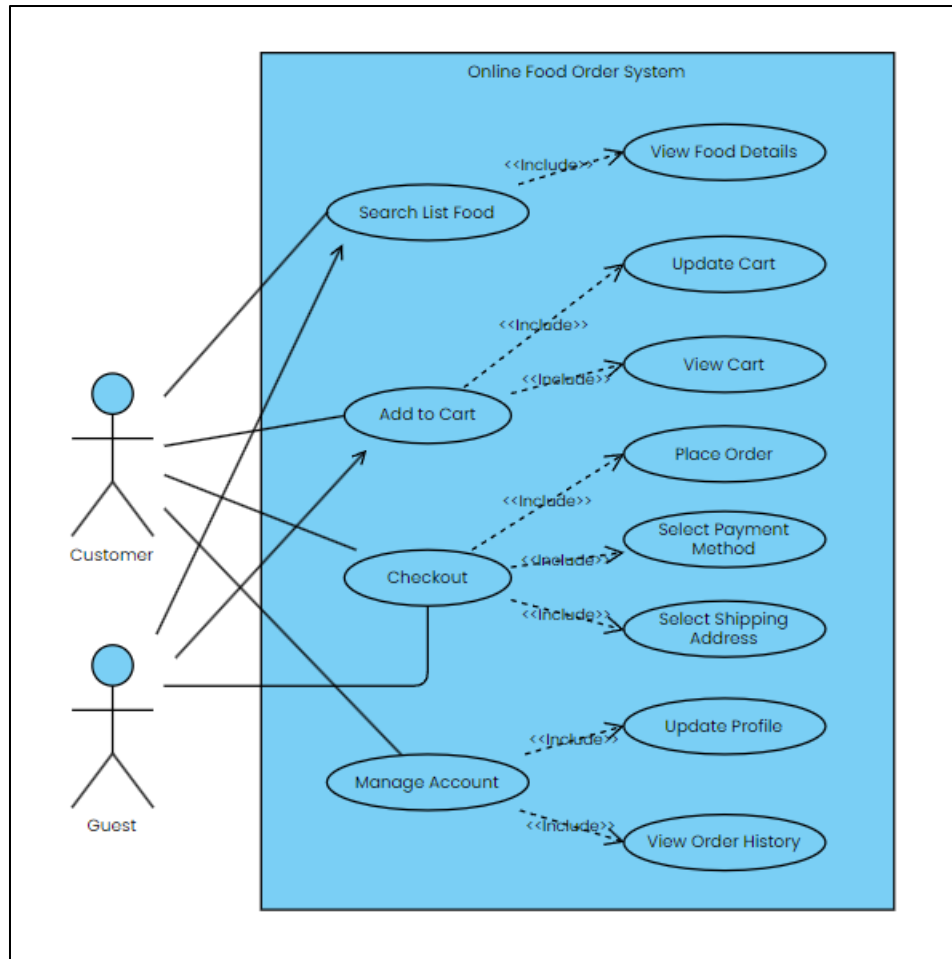


Fig 1. Use Case Diagram

Figure 1 comprehensively depicts the various actions Customers and Guests perform during their engagement with the system. These actions include searching for food items, adding selected items to the cart, proceeding to the checkout process, and managing their respective accounts. The Search List meal application allows users to choose and examine various meal options and access detailed information on each food item. The Food List provides a venue for consumers and guests to engage in multiple activities. The "Add to Cart" feature encompasses two components: the "Update Chart" function and the "View Cart" function. The checkout process contains three main steps: placing an order, selecting a payment method, and choosing a shipping address. The Manage Account feature encompasses two main functionalities: updating the user's profile information and seeing the order history.

* Corresponding author



[Creative Commons Attribution-NonCommercial-ShareAlike 4.0
International License.](https://creativecommons.org/licenses/by-nc-sa/4.0/)

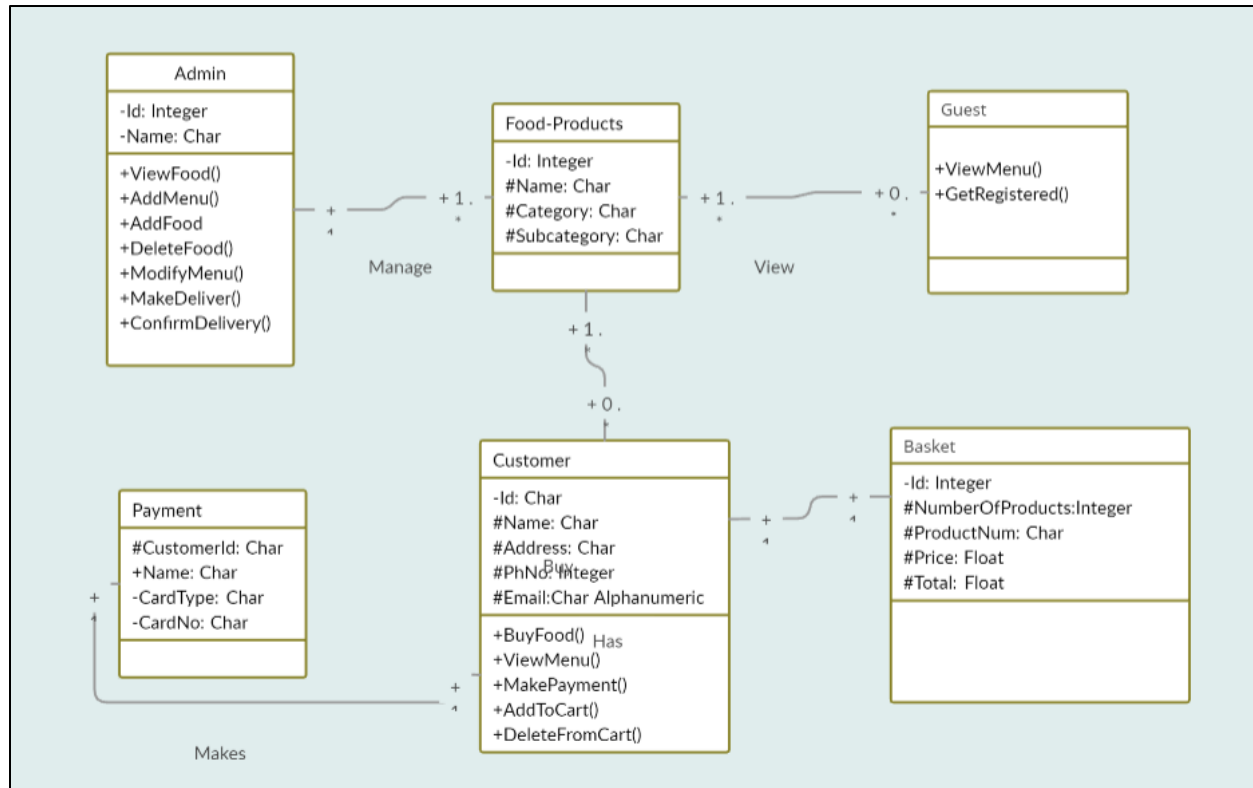


Fig 2. Class Diagram

Fig 2, aims to display Class Diagrams for Admin, Payment, Food-Products, Guest, Customer, Basket. Class Diagram in Admin consists of Id (type: Integer), Name (type: Char). Procedures ViewFood(), AddMenu(), AddView(), DeleteFood(), ModifyMenu(), MakeDeliver(), ConfirmDelivery(). Customer consists of Id (type: Char), Name (type: Char), Address (type: Char), PhNo (type: Integer), Email (type: Charnumeric). BuyFood(), ViewMenu(), MakePayment(), AddToCart(), DeleteFromCart() procedures. The basket consists of id (type Integer), NumberofProduct (type: Integer), ProductNum (type: Char), Price (type: Float), Total (type: Float). Payment consists of CustomerId (type: Char), Name (type: Char), CardType (type: Char), CardNo (type: Char).

DISCUSSIONS

How can Indonesian culinary applications be designed using Use Case Diagrams that identify and represent the numerous interactions between actors and systems as well as system functions and behavior? (RQ1).

The primary objective of this study, denoted as RQ1, is to concentrate on developing Indonesian culinary apps by utilizing the Use Case Diagram methodology. The Use Case Diagram effectively illustrates the dynamic relationships between actors, such as users or other entities, and the system. It also provides a comprehensive depiction of the system's functionality and behavior. The study will encompass the procedure of discerning the individuals or entities engaged in the application, ascertaining the many scenarios in which they interact with the system, and graphically representing the interactions between actors and systems in diverse usage contexts. This study aims to identify the stakeholders who can engage with culinary applications, including users, administrators, and potentially external entities. Subsequently, the utilization scenarios that exemplify the primary functionalities of culinary applications will be delineated, encompassing activities such as recipe exploration, tutorial perusal, culinary knowledge acquisition, and similar endeavors. A comprehensive elucidation of each use case will encompass several usage situations. This will entail thoroughly examining the actions undertaken by both the actor and the system during their interaction. The outcomes of Research Question 1 will be presented in the format of Use Case Diagrams, which serve as visual representations illustrating the interactions between actors and systems within culinary applications. The graphic presented herein offers a comprehensive depiction of the diverse usage situations and the corresponding responses of

* Corresponding author



the program to the activities undertaken by various participants. Utilizing this Use Case Diagram will also aid in identifying system functions and behaviors that necessitate implementation within the application. Consequently, it will serve as a comprehensive framework for designing Indonesian culinary apps that effectively cater to the requirements of users and other relevant stakeholders.

Research Question 2 (RQ2) explores the fundamental components that must be incorporated into an Indonesian culinary application employing the Unified Modeling Language (UML) methodology. The objective is to enhance culinary promotion and facilitate the distribution of varied regional culinary information. The crux of the matter is developing a comprehensive and influential application that adeptly demonstrates the depth of Indonesian cuisine while fostering cultural interaction and comprehension.

The inclusion of fundamental components inside the culinary application is crucial for the attainment of these objectives. Establishing a comprehensive Culinary Product Database is a fundamental resource, encompassing a wide array of data encompassing recipes, ingredients, and regional variations. The User Interface assumes a significant role, necessitating an intuitive design to facilitate smooth navigation and interaction, enabling users to discover and interact with the application effortlessly. The integration of robust Search and Culinary Categories functions enables customers to navigate culinary options efficiently, boosting their overall exploration experience. Multimedia recipe tutorials enhance user engagement by including visual aids, such as photographs and videos, which facilitate the understanding and practical application of recipes. To cultivate a more profound affiliation with Indonesian culture, the application integrates Regional and Cultural Information, illuminating perspectives on the origins, histories, and cultural import of many culinary traditions around the country.

The application also promotes User Engagement by facilitating user feedback, recipe ratings, and user-generated contributions, cultivating a communal atmosphere and generating a collective culinary experience. Security and privacy measures are used to safeguard user data, hence fostering confidence in the utilization of the program. In addition, the Content Management function enables administrators to effectively manage and modify the application's content in real-time, ensuring its timeliness and pertinence. The design process will be guided by addressing Research Question 2, ultimately resulting in the development of an Indonesian culinary application that effectively incorporates these essential elements. Utilizing the UML approach, the application can deliver an immersive, safe, and user-centric platform, effectively fulfilling its objective of presenting Indonesia's gastronomic abundance and cultural variety to an international audience.

CONCLUSION

In summary, utilizing the Unified Modeling Language (UML) methodology in developing an Indonesian culinary application system is a very efficient strategy for generating comprehensive and well-organized solutions. The UML methodology offers a robust framework for delineating diverse facets of culinary applications, encompassing the interactions between actors and systems through Use Case Diagrams and the class structure and relationships between components via Class Diagrams. The development of Indonesian culinary applications can be facilitated by employing Unified Modeling Language (UML) to comprehensively outline essential components, including culinary databases, user-friendly interfaces, search functionalities, culinary categorizations, and cultural and regional data. This approach enables the integration of novel functionalities, such as multimedia recipe instruction, user engagement, and content administration, potentially enhancing the overall user experience. Furthermore, Unified Modeling Language (UML) facilitates the creation of efficient user interfaces, guaranteeing the seamless integration of user interactions with the system. Moreover, this approach enables enhanced surveillance, examination, and verification, fostering the development of apps with robust security measures, dependable functionality, and adherence to regulatory standards. In general, the utilization of the UML method in designing an Indonesian culinary application system offers a robust framework for creating scalable and well-organized solutions. This approach facilitates the development of culinary apps that are informative, visually appealing, and capable of adapting to various devices and user interactions. This approach enables the fulfillment of culinary promotion requirements, the dissemination of cultural knowledge, and the ongoing maintenance and advancement of the application. The use of UML in the design of Indonesian culinary applications plays a significant role in advancing and enhancing the cultural significance of Indonesian cuisine within the context of the information technology era.

REFERENCES

Abdalazeim, A., & Meziane, F. (2021). A review of the generation of requirements specification in natural language using objects UML models and domain ontology. *Procedia CIRP*, 189(2019), 328–334.

* Corresponding author



[Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.](https://creativecommons.org/licenses/by-nc-sa/4.0/)

- <https://doi.org/10.1016/j.procs.2021.05.102>
- Abdelmadjid, L., & Mimoun, M. (2021). Uncertain Decision-Making Requirements Formalizing with Complement Fuzzy UML Model. *Procedia Computer Science*, 198, 317–322. <https://doi.org/10.1016/j.procs.2021.12.247>
- Bergström, G., Hujainah, F., Ho-Quang, T., Jolak, R., Rukmono, S. A., Nurwidyantoro, A., & Chaudron, M. R. V. (2022). Evaluating the layout quality of UML class diagrams using machine learning. *Journal of Systems and Software*, 192, 111413. <https://doi.org/10.1016/j.jss.2022.111413>
- Bosse, S. (2018). A Unified System Modelling and Programming Language based on JavaScript and a Semantic Type System. *Procedia Manufacturing*, 24, 21–39. <https://doi.org/10.1016/j.promfg.2018.06.005>
- Chen, F., Zhang, L., Lian, X., & Niu, N. (2022). Automatically recognizing the semantic elements from UML class diagram images. *Journal of Systems and Software*, 193, 111431. <https://doi.org/10.1016/j.jss.2022.111431>
- Estradivari, Agung, M. F., Adhuri, D. S., Ferse, S. C. A., Sualia, I., Andradi-Brown, D. A., Campbell, S. J., Iqbal, M., Jonas, H. D., Lazuardi, M. E., Nanlohy, H., Pakiding, F., Pusparini, N. K. S., Ramadhana, H. C., Ruchimat, T., Santiadji, I. W. V., Timisela, N. R., Veverka, L., & Ahmadi, G. N. (2022). Marine conservation beyond MPAs: Towards the recognition of other effective area-based conservation measures (OECMs) in Indonesia. *Marine Policy*, 137, 104939. <https://doi.org/10.1016/j.marpol.2021.104939>
- Ferreira, L., Miranda, I., Simoes, R., & Cruz-Cunha, M. M. (2015). GuiMarket Specification Using the Unified Modeling Language. *Procedia Computer Science*, 64, 1263–1272. <https://doi.org/10.1016/j.procs.2015.09.241>
- Li, L., Li, X., He, T., & Xiong, J. (2013). Extensics-based test case generation for UML activity diagram. *Procedia Computer Science*, 17, 1186–1193. <https://doi.org/10.1016/j.procs.2013.05.151>
- Meiliana, Septian, I., Alianto, R. S., Daniel, & Gaol, F. L. (2017). Automated Test Case Generation from UML Activity Diagram and Sequence Diagram using Depth First Search Algorithm. *Procedia Computer Science*, 116, 629–637. <https://doi.org/10.1016/j.procs.2017.10.029>
- Samaddar, A., Cuevas, R. P., Custodio, M. C., Ynion, J., Ray (Chakravarti), A., Mohanty, S. K., & Demont, M. (2020). Capturing diversity and cultural drivers of food choice in eastern India. *International Journal of Gastronomy and Food Science*, 22(March), 100249. <https://doi.org/10.1016/j.ijgfs.2020.100249>
- Sholihah, A. B., & Heath, T. (2016). Assessing the Quality of Traditional Street in Indonesia: A case study of Pasar Baru Street. *Procedia - Social and Behavioral Sciences*, 234, 244–254. <https://doi.org/10.1016/j.sbspro.2016.10.240>
- Sukenti, K., Hakim, L., Indriyani, S., Purwanto, Y., & Matthews, P. J. (2016). Ethnobotanical study on local cuisine of the Sasak tribe in Lombok Island, Indonesia. *Journal of Ethnic Foods*, 3(3), 189–200. <https://doi.org/10.1016/j.jef.2016.08.002>
- Swain, R. K., Panthi, V., & Behera, P. K. (2012). Test Case Design Using Slicing of UML Interaction Diagram. *Procedia Technology*, 6, 136–144. <https://doi.org/10.1016/j.protcy.2012.10.017>
- Thomas, M., Mihaela, I., Andrianjaka, R. M., Germain, D. W., & Sorin, I. (2021). Metamodel based approach to generate user interface mockup from UML class diagram. *Procedia Computer Science*, 184, 779–784. <https://doi.org/10.1016/j.procs.2021.03.096>
- Winata, F., & McLafferty, S. L. (2023). Therapeutic landscapes and networks in restricted lives: Constructing restorative experiences among Indonesian female domestic workers in Hong Kong. *Wellbeing, Space and Society*, 5(May), 100163. <https://doi.org/10.1016/j.wss.2023.100163>
- Wu, H. (2023). QMaxUSE: A new tool for verifying UML class diagrams and OCL invariants. *Science of Computer Programming*, 228, 102955. <https://doi.org/10.1016/j.scico.2023.102955>

* Corresponding author

