1. Analysis
   1. Introduction to analysis

Analysis tries to dig deeper into a topic to get better understanding about it. A large topic is usually broken down into smaller parts and studied thoroughly. In our case, analysis deals with finding and understanding the requirements or needs of the project. Analysis is also done to find out the needs when a new product is made or an old product is altered so that it meets the requirements. In this computing age, analysis doesn't stop at gathering requirements but system analysts guide us through the entire software development process and provide us with the best solutions when necessary.

During the analysis phase, business analyst go through following steps:

* **Requirement Specification**

To find out problem specifications, system analysts go through different fact finding procedures like interviews and surveys with the client or company representatives. They discuss about the problems and the solutions that the upcoming software can provide.

This is important because if a business analyst does not get a customer's vision, then they might end up building a completely different product. This phase helps the analyst to know exactly what the customer want.

* **Needs identification**

This phase basically narrows down the problems or requirements that were gathered during the requirements specification. System analyst finds out actual problems and the ones that need to be addressed. We differentiate what client actually needs and not what he wants. The client's vision is then connected with the software development process and analysed if it is practically possible.

This phase is important because we get to understand the actual problems that needs to be solved and omit out the ones which are not necessary.

* **Translation**

In a software development process, a business analyst must have the idea of what the client want and our company must understand what we can have, and the development team must understand what they can offer out of our expectations. The job of the business analyst is to translate these needs of clients to development tasks for the development team in such a way that it meet the client's expectations.

This step is done to avoid misunderstandings that occur between the development team and the client. The business analyst who have idea about both the sides work as a communicator to remove such technical jargons.

* **Communication**

The development process of a software is often long and requires certain level of co-operation between the parties during the time period. Business analyst also works as a mediator between customer and developer who finds out the strong points and weak points and helps to improve the flow of communication.

Communication is very important because without it the customer never knows what is being developed and the end result may be disappointing. Business analysts during the analysis make sure the needs are met.

* 1. Methodology

Software system methodology basically is a process or a series of processes that describes how the life cycle of the software will be. We can also call it a framework where it is used to demonstrate, plan or control the software development process. During the development process of Canteen Automation System, I’m going to use incremental RAD as a development methodology.

* **Why not Evolutionary models?**

We use evolutionary models when the requirements are not clear. These models are mostly used for software development with frequently changing requirements. But, our system’s requirements are completely clear and we know exactly what we need. So, I’m not going to use any of the evolutionary models.

* **Why not Waterfall model?**

As waterfall model requires the need of knowledge of every step and that’s exactly in our case but I’m not going to adopt it because of the few limitations which makes it less feasible. Two main reasons for not using it are:

* Different stages cannot be performed parallelly.
* It is time consuming.
* **Why Incremental RAD model?**

Usually, Incremental model is preferred when we have less time but our needs are clear. Most intriguing feature in Incremental RAD is that every module can be completed and can be released in isolation if there is lack of time. In our system, there are many features which are not inter-related to each other which can be released separately. The motivates users to give updates about that certain module and it can be updated on its own which helps to make the system more user-friendly and we can also include and address the user needs. Also, as we have time period of just 3 months, we can’t go step by step like waterfall model to complete the project for which the manner of parallelism should be practised. So, looking at all these requirements, I found out that Incremental RAD is suited for my Canteen Automation System because it encourages to create a functional system within a short period of time.

* 1. Feasibility Study

Feasibility, by its name suggests that it is the degree in which it is measured or tested whether something can be done easily and conveniently. There are different factors in software development field which should be considered before moving towards the next step. We check whether the project is doable or not as well as checking the project’s success potential. There are different factors to be determined in a feasibility study which are as follows:

* Economic Feasibility

This assessment basically is used to determine the costs required during the software development process and the profits we may make after the completion of the system. Since this is a small project with no such costs and future benefits expectations, my system is feasible to develop.

* Technical Feasibility

Technical feasibility handles the technical parts like hardware and software that are required during the system development process. Usually, if the organization is willing to provide all the required tools, then the system development is considered technically feasible. I have all the tools required which makes my project technically feasible.

* Scheduling Feasibility

This part is considered most important part for the success of the project. After all, if there is no end-result after the time limit, the project will fail. An organization determines the time period that is required to complete the project during the scheduling feasibility. I can easily complete the project in time which makes my project feasible.

* Operational Feasibility

After the software development is complete, the system must be user-friendly or the users should be able to operate the system easily without any difficulties. My software is going to have a simple and user-friendly interface which makes it operationally feasible.

* 1. Requirement Analysis
     1. Functional Requirement

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Title** | **Description** | **Rational** | **Dependencies** |
| FR001 | Registration | Customers/Users can register their account. | It is needed to get user information for later use. | N/A |
| FR002 | Login | Customers/Users can login using their credentials. | It is needed for authentication of users. | FR001 |
| FR003 | Manage Account | Registered users can view and manage their profile. | It is needed for users to update or delete their account or data. | FR001 |
| FR004 | Navigate menu | Customers can go through the canteen menu. | It is needed to view the menu items. | N/A |
| FR005 | Select an item from the menu | Customers can select an item from the menu for further processing. | It is needed for the users to make an order. | FR001 |
| FR006 | Add an item to an order | Customers can add an item to a new or existing order. | It is needed for the users to make an order. | FR005 |
| FR007 | Review the order | Customers can review the order along with the prices. | It is needed for the customer to change the items if they change their mind. | FR006 |
| FR008 | Remove any item from the current order | Customers can remove any item from their order. | It’s important because customers should be able to remove the item they don’t want. | FR007 |
| FR009 | Provide payment details | After the order, customers can provide payment details. | It’s needed because we need to provide customers with different payment options. | N/A |
| FR010 | Place an order | Customers can place an order. | The sole purpose of this system is to order food. So, we need this. | FR006 |
| FR011 | Receive confirmation in the form of order number | After ordering the food, customers will get a confirmation of their order. | It is needed for the order number that’ll be needed in certain cases. | FR010 |
| FR012 | Pay online using a payment gateway | Customers can pay using a payment gateway that is available. | Payment is an important component in this system. This allows customers to pay using different options. | FR009 |
| FR013 | Add new/update/delete food from the menu | Canteen Manager can add, update or delete item from the menu. | This is needed because if the menu item change, canteen manager should be able to perform different actions. | FR001 |
| FR014 | Retrieve new orders from the database. | Kitchen staffs can retrieve new orders from the database. | It is needed because kitchen should be able to view order before processing. | FR001 |
| FR015 | Display the orders in easily readable, graphical way. | Kitchen staff can view the orders in easier and readable format. | The orders must be in an easier and readable format so that the staffs can understand. | N/A |
| FR016 | Mark an order as having been processed and remove it from the list of active orders. | Kitchen staffs will be able to mark or delete an order that has been processed. | It helps to make the order list more organizing and clean. | FR015 |
| FR017 | Update default options for a given food item. | Canteen Manager can update the different options available for the foods. | It allows the canteen owner to make changes to options that are available for different foods. | FR001 |
| FR018 | Add a new/update/delete food category to/from the menu. | Canteen Manager can add, update or delete food category from the menu. | It allows the canteen owner to make changes to food category that are available for different foods. | FR001 |
| FR019 | Option to credit the amount that a user has to pay. | Customers will be able to pay on credit and pay later on. | It’s needed because most of the people in schools or college canteen eat on credit. | FR001 |
| FR020 | Automatic reports generation. | The system will be able to generate profits, revenue and credit reports. | My system focuses on generating reports on different aspects and provide a general view to the manager. So this is important. | N/A |

Table 1: Functional Dependencies

* + 1. Non-functional Requirement

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Title** | **Description** | **Rational** | **Dependencies** |
| NFR001 | Time Limit | Displays the elapsed time between the submission of order process between the customer  and cashier | It gives the customers about the idea on the time that takes for the food. | N/A |
| NFR002 | More user friendly | Our canteen automation system should be more user friendly | The application will be friendlier for the users. | N/A |
| NFR003 | Flexibility | Our project should be so flexible that whenever we want to make changes in it very easily it can be done. | It’ll be easier to make future updates if necessary. | N/A |
| NFR004 | Portability | Our project should be portable on any platform and available on websites easily and at a faster speed than others. | More customers can be targeted if it supports multiple platforms. | N/A |
| NFR005 | Usability | Our website must be easy to learn and use. | It’ll be more effective if users learn easily. | N/A |
| NFR006 | Security | Website must be secured. | User security must be a concern which helps them to trust the application. | FR001 |
| NFR007 | Availability | Must be available anytime. | Users can use it whenever necessary. | N/A |
| NFR008 | Reliability | The system must be reliable so the customers would trust it. | It’ll increase more users and customer satisfaction. | N/A |
| NFR009 | Accuracy and precision | The data must be precise and accurate. | Correct data adds more value to the functioning of the application. | N/A |
| NFR010 | Scalability | The application must be scalable. | It’ll be useful for small as well as large canteens. | N/A |

Table 2: Non-functional Dependencies

* + 1. MoSCoW Prioritization

During a project management, MoSCoW prioritization is done to clearly understand the requirements and their priority. MoSCoW simply stands as Must have, Should have, Could have and Would have. This is done to find out which requirements come first, which one comes later on and which one we can exclude.

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Title** | **MoSCoW** | **Description** |
| FR001 | Registration | Must Have |  |
| FR002 | Login | Must Have |  |
| FR003 | Manage Account | Must Have |  |
| FR004 | Navigate menu | Must Have |  |
| FR005 | Select an item from the menu | Must Have |  |
| FR006 | Add an item to an order | Must Have |  |
| FR007 | Review the order | Should Have |  |
| FR008 | Remove any item from the current order | Must Have |  |
| FR009 | Provide payment details | Must Have |  |
| FR010 | Place an order | Must Have |  |
| FR011 | Receive confirmation in the form of order number | Should Have |  |
| FR012 | Pay online using a payment gateway | Should Have |  |
| FR013 | Add new/update/delete food from the menu | Must Have |  |
| FR014 | Retrieve new orders from the database. | Must Have |  |
| FR015 | Display the orders in easily readable, graphical way. | Should Have |  |
| FR016 | Mark an order as having been processed and remove it from the list of active orders. | Should Have |  |
| FR017 | Update default options for a given food item. | Must Have |  |
| FR018 | Add a new/update/delete food category to/from the menu. | Must Have |  |
| FR019 | Option to credit the amount that a user has to pay. | Must Have |  |
| FR020 | Automatic reports generation. | Must Have |  |
| NFR001 | Time Limit | Could Have |  |
| NFR002 | More user friendly | Should Have |  |
| NFR003 | Flexibility | Should Have |  |
| NFR004 | Portability | Should Have |  |
| NFR005 | Usability | Should Have |  |
| NFR006 | Security | Should Have |  |
| NFR007 | Availability | Could Have |  |
| NFR008 | Reliability |  |  |
| NFR009 | Accuracy and precision |  |  |
| NFR010 | Scalability |  |  |

Table 3: MoSCoW prioritization

* + 1. SRS