**As Partial Fulfillment of:**

**MSc (IT) - Master of Science In Information Technology**

**Submitted To:**

**Atmiya Institute of Technology and Science – 2017**

GUIDED BY: Prakash Gujarati

PREPARED BY: Dipen Parmar

**ACKNOWLEDGEMENT**

I would like to highlight the fact that this project is the product of many contributors. First, I’d like to thank Professor **Prakash Gujarati** who accepted to work with me on my Project idea, supervising my work.

Second, I would like to thank the stack overflow community and Google for providing tremendous help to the elaboration of this project.

**ABSTRAT**

The Lara-POS Restaurant Management software is a capstone project that aims towards developing an all-in-one application that addresses the various problems and challenges faced by high-end restaurant owners today.

In order to achieve this goal, this project addresses various aspects of the modern Restaurant Management System in India.

All in all, this projects main aim is to reduce the time overhead in high-end management restaurants by providing an alternative to the traditional management system based on physical record keeping and paper work.

**PROJECT PROFILE**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **PROJECT:** | Lara-POS Restaurant Management. | | **OBJECTIVE:** | Pendding | | **INSTITUTE:** | Atmiya Institute of Technology and Science | | **FRONT END:** | Bootstrap 4, jQuery, | | **BACK END:** | PHP 7.1.3 , Laravel 5.6 , MySql | | **DOCUMENTATION TOOL:** | Ms-Word,Excel,Visio, Draw.io | | **DEVELOPED BY:** | Dipen Parmar | | **GUIDED BY:** | Prakash Gujarati | | **SUBMITTED TO:** | Msc-it & ca (AITS) | | **-** |  | | **-** |  | |  |
|  |  |

**INDEX**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr** | **Particulars** | | **PageNo** |
|  |  |  |  |
| **1** | **Title Page** | | - |
| **2** | **ACKNOWLEDGEMENT** | | - |
| **3** | **ABSTRAT** | | - |
| **3** | **PROJECT PROFILE** | | - |
| **4** | **UML DIAGRAMS** | | () |
| 4.1 | Use Case | - |
|  | 4.2 | Activity Diagram | - |
|  | 4.3 | Wire Frame | - |
|  | 4.4 | User Interface | - |
|  | 4.5 | Data Dictionary | - |
|  | 4.6 | Data Flow Diagram (DFD) | - |
|  | 4.7 | E-R Diagram |  |
| **5** | **Report Softcopy** | | () |
| **6** | **----------------------------** | | - |
|  | 6.1 | User Interface | - |
|  | 6.2 | Database Structure, Migrations |  |
| **7** | **Implementation** | | () |
|  | 7.1 | Errors and Solutions |  |
|  | 7.2 | Challenging Tasks / Problems |  |

**HARDWARE REQUIREMENTS**

**Technical Details Hardware (**Requirements**)**

|  |  |
| --- | --- |
| **Particulars** | **Minimum Hardware**  **Requirements** |
| Processor Brand | Intel, Amd etc. |
| Processor Type | Core 2 Duo |
| Processor Speed | 1.0 GHz |
| RAM Size | 1 GB |
| Memory Type | DDR2 |
| Hard Drive Size | 30 GB |
| Hard Drive Interface | Any |

**Technical Details Software (**Requirements**)**

|  |  |
| --- | --- |
| **Particulars** | **Minimum Software**  **Requirements.** |
| Operating System | Windows, Linux, Mac, any |
| Browser | • Chrome\* 36+ • Edge\* 20+ • Mozilla Firefox 31+ |

**USE CASE**

A use case describes how a user uses a system to accomplish a particular goal. A use case diagram consists of the system, the related use cases and actors and relates these to each other to visualize: what is being described? (system), who is using the system? (actors) and what do the actors want to achieve? (use cases), thus, use cases help ensure that the correct system is developed by capturing the requirements from the user's point of view.

----

**What is a use case diagram?**

In the Unified Modeling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors. An effective use case diagram can help your team discuss and represent:

* Scenarios in which your system or application interacts with people, organizations, or external systems
* Goals that your system or application helps those entities (known as actors) achieve
* The scope of your system

**When to apply use case diagrams?**

A use case diagram doesn't go into a lot of detail—for example, don't expect it to model the order in which steps are performed. Instead, a proper use case diagram depicts a high-level overview of the relationship between use cases, actors, and systems. Experts recommend that use case diagrams be used to supplement a more descriptive textual use case.

UML is the modeling toolkit that you can use to build your diagrams. Use cases are represented with a labeled oval shape. Stick figures represent actors in the process, and the actor's participation in the system is modeled with a line between the actor and use case. To depict the system boundary, draw a box around the use case itself.

UML use case diagrams are ideal for:

* Representing the goals of system-user interactions
* Defining and organizing functional requirements in a system
* Specifying the context and requirements of a system
* Modeling the basic flow of events in a use case

Use case diagram components

To answer the question, "What is a use case diagram?" you need to first understand its building blocks. Common components include:

**Actors:** The users that interact with a system. An actor can be a person, an organization, or an outside system that interacts with your application or system. They must be external objects that produce or consume data.

**System:** A specific sequence of actions and interactions between actors and the system. A system may also be referred to as a scenario.

**Goals:** The end result of most use cases. A successful diagram should describe the activities and variants used to reach the goal.

**Use case diagram symbols and notation**

The notation for a use case diagram is pretty straightforward and doesn't involve as many types of symbols as other UML diagrams.

* **Use cases:** Horizontally shaped ovals that represent the different uses that a user might have.
* **Actors:** Stick figures that represent the people actually employing the use cases.
* **Associations:** A line between actors and use cases. In complex diagrams, it is important to know which actors are associated with which use cases.
* **System boundary boxes:** A box that sets a system scope to use cases. All use cases outside the box would be considered outside the scope of that system. For example, Psycho Killer is outside the scope of occupations in the chainsaw example found below.

**System**:  
 Draw your system's boundaries using a rectangle that contains use cases. Place actors outside the system's boundaries.



**Use Case**:  
 Draw use cases using ovals. Label the ovals with verbs that represent the system's functions.

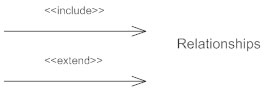
****

**Actors:** Actors are the users of a system. When one system is the actor of another system, label the actor system with the actor stereotype.



**Relationships:**

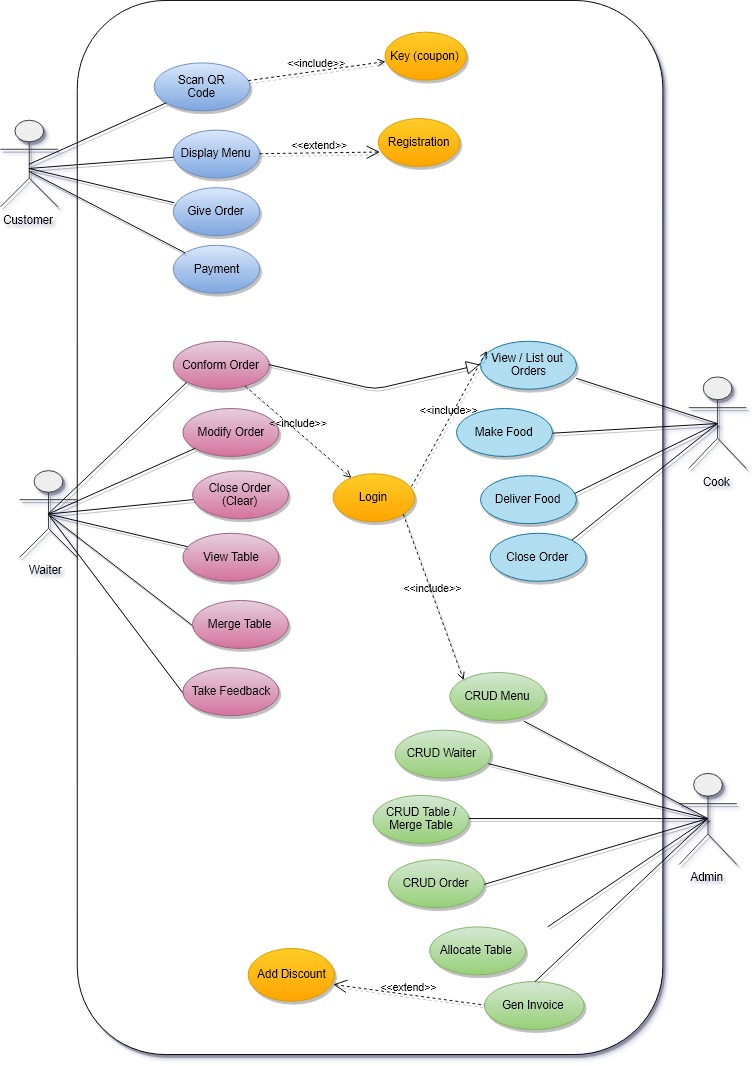
Illustrate relationships between an actor and a use case with a simple line. For relationships among use cases, use arrows labeled either "uses" or "extends." A "uses" relationship indicates that one use case is needed by another in order to perform a task. An "extends" relationship indicates alternative options under a certain use case.



Src 1: <https://www.lucidchart.com/pages/uml-use-case-diagram>

Src 2: <https://www.smartdraw.com/use-case-diagram/>

* **Use case diagram for Restaurant management system:**



Drawing Tool Used- : **Draw.io**

**Draw**.**io** is free online diagram software for making flowcharts, process diagrams, org charts, UML, ER and network diagrams.

Website: <https://www.draw.io/>

**ACTIVITY DIAGRAM**

**Activity diagram**: is another important behavioral diagram in UML diagram to describe dynamic aspects of the system. Activity diagram is essentially an advanced version of flow chart that modeling the flow from one activity to another activity.

## When to Use Activity Diagram?

Activity Diagrams describe how activities are coordinated to provide a service which can be at different levels of abstraction. Typically, an event needs to be achieved by some operations, particularly where the operation is intended to achieve a number of different things that require coordination, or how the events in a single use case relate to one another, in particular, use cases where activities may overlap and require coordination. It is also suitable for modeling how a collection of use cases coordinate to represent business workflows

1. Identify candidate use cases, through the examination of business workflows
2. Identify pre- and post-conditions (the context) for use cases
3. Model workflows between/within use cases
4. Model complex workflows in operations on objects
5. Model in detail complex activities in a high level activity Diagram

* Activity Diagram Notation Summary

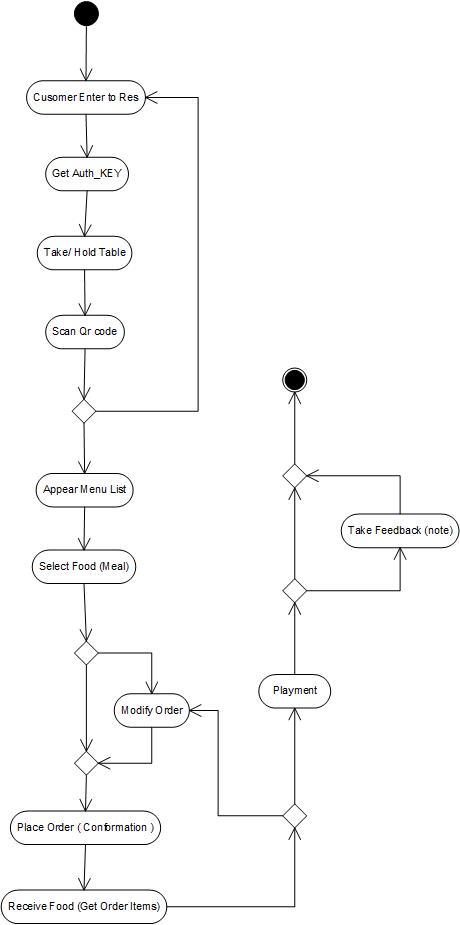
| Symbol | Name | Description |
| --- | --- | --- |
| start Symbol | Start symbol | Represents the beginning of a process or workflow in an activity diagram. It can be used by itself or with a note symbol that explains the starting point. |
| activity Symbol | Activity symbol | Indicates the activities that make up a modeled process. These symbols, which include short descriptions within the shape, are the main building blocks of an activity diagram. |
| connector Symbol | Connector symbol | Shows the directional flow, or control flow, of the activity. An incoming arrow starts a step of an activity; once the step is completed, the flow continues with the outgoing arrow. |
| joint Symbol | Joint symbol/ Synchronization bar | Combines two concurrent activities and re-introduces them to a flow where only one activity occurs at a time. Represented with a thick vertical or horizontal line. |
| fork Symbol | Fork symbol | Splits a single activity flow into two concurrent activities. Symbolized with multiple arrowed lines from a join. |
| decision Symbol | Decision symbol | Represents a decision and always has at least two paths branching out with condition text to allow users to view options. This symbol represents the branching or merging of various flows with the symbol acting as a frame or container. |
| note Symbol | Note symbol | Allows the diagram creators or collaborators to communicate additional messages that don't fit within the diagram itself. Leave notes for added clarity and specification. |
| flow final symbol | Flow final symbol | Represents the end of a specific process flow. This symbol shouldn’t represent the end of all flows in an activity; in that instance, you would use the end symbol. The flow final symbol should be placed at the end of a process in a single activity flow. |
| end symbol | End symbol | Marks the end state of an activity and represents the completion of all flows of a process. |

Src 1: <https://www.lucidchart.com/pages/uml-activity-diagram>

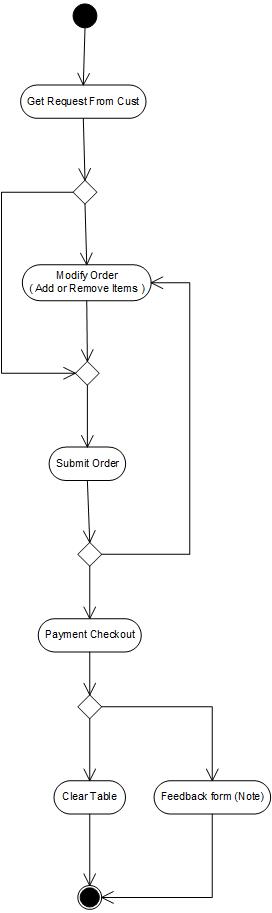
Src 2: <https://online.visual-paradigm.com/tutorials/activity-diagram-tutorial/>

Src 3: <https://www.lucidchart.com/pages/uml-activity-diagram>

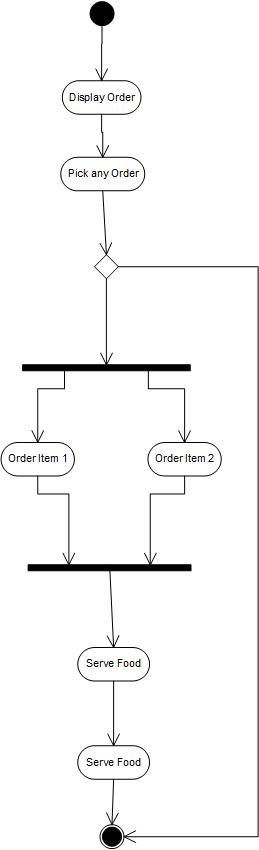
* Activity Diagram For **Customer**



* Activity Diagram For **Waiter**



* Activity Diagram For **Cook**



**USER INTERFACE**

Src: <https://techterms.com/definition/user_interface>

**What is User Interface Design?**

User Interface (UI) Design is the link between users and your website. It includes the basic design elements that need to be present in order to for someone to navigate your site and make decisions. It is the ever-evolving relationship between a person and the system that they are using. It includes the way that your website interacts with users, the overall design and how information is presented.

There are many different ways that you can look at user interface, but the basics always include the communication from a product to the user and vice versa1.

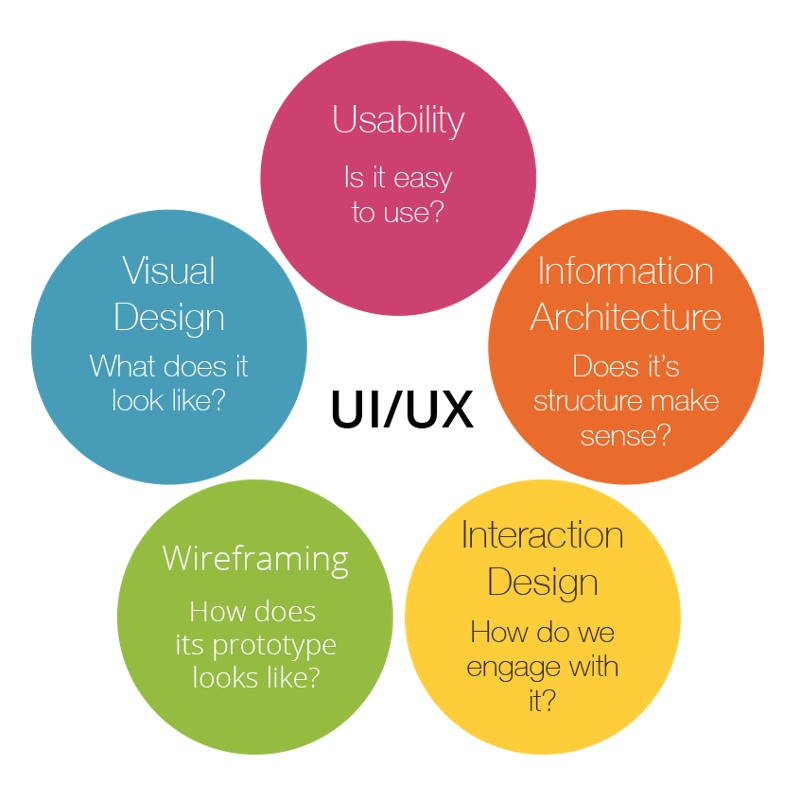
UI Design is all about structure, user manipulation and communication. This is one of the reasons it’s so important that you pay close attention to it. It is the basic building blocks of how your website is set up and functions when visited by your target audience. If it doesn’t go smoothly, problems tend to follow.

Some of the elements involved in User Interface Design include the input controls, navigational components, informational components and containers4. The devil is in the details, as they say. Buttons, lists, toggles, icons, breadcrumbs, tags and more also play a huge role. It is the way that you design your site from the back-end so that your audience can have a seamless, and enjoyable, front-end experience.

## ****There are 3 issues / parts of a user interface design, these being****

* how it looks,
* how it works,
* how easy it is to use.

**Key Components of UI/UX Design**



There are some core components of UX Design, which are as follows:

* Information Architecture
* Interaction Design
* Usability
* Wire framing
* Visual Design

**Interaction Design**

Interaction Design is about creating the conceptual design with which the users interact with the product and/or application. This interaction includes various elements such as aesthetics, color, font, icons, images, motion, sound, space, graphics etc.

**Usability**

Usability can be referred as the user-friendliness as well. Along with figuring out if the users get the information they want by using the application or visiting the site for the first time and if the application/website is easy to navigate, the usability also comprehends the ways to handle the errors.

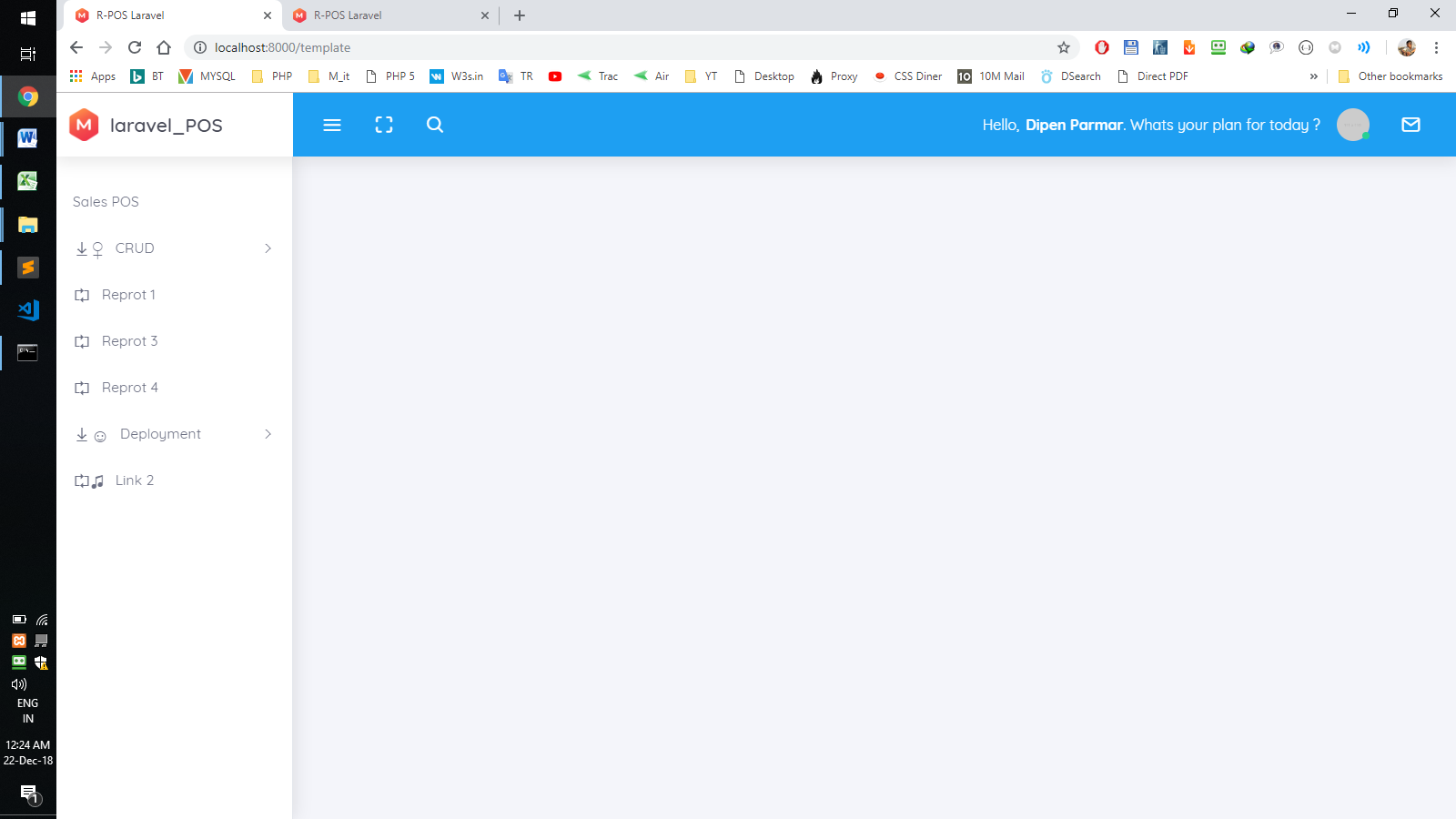
**Wire framing**

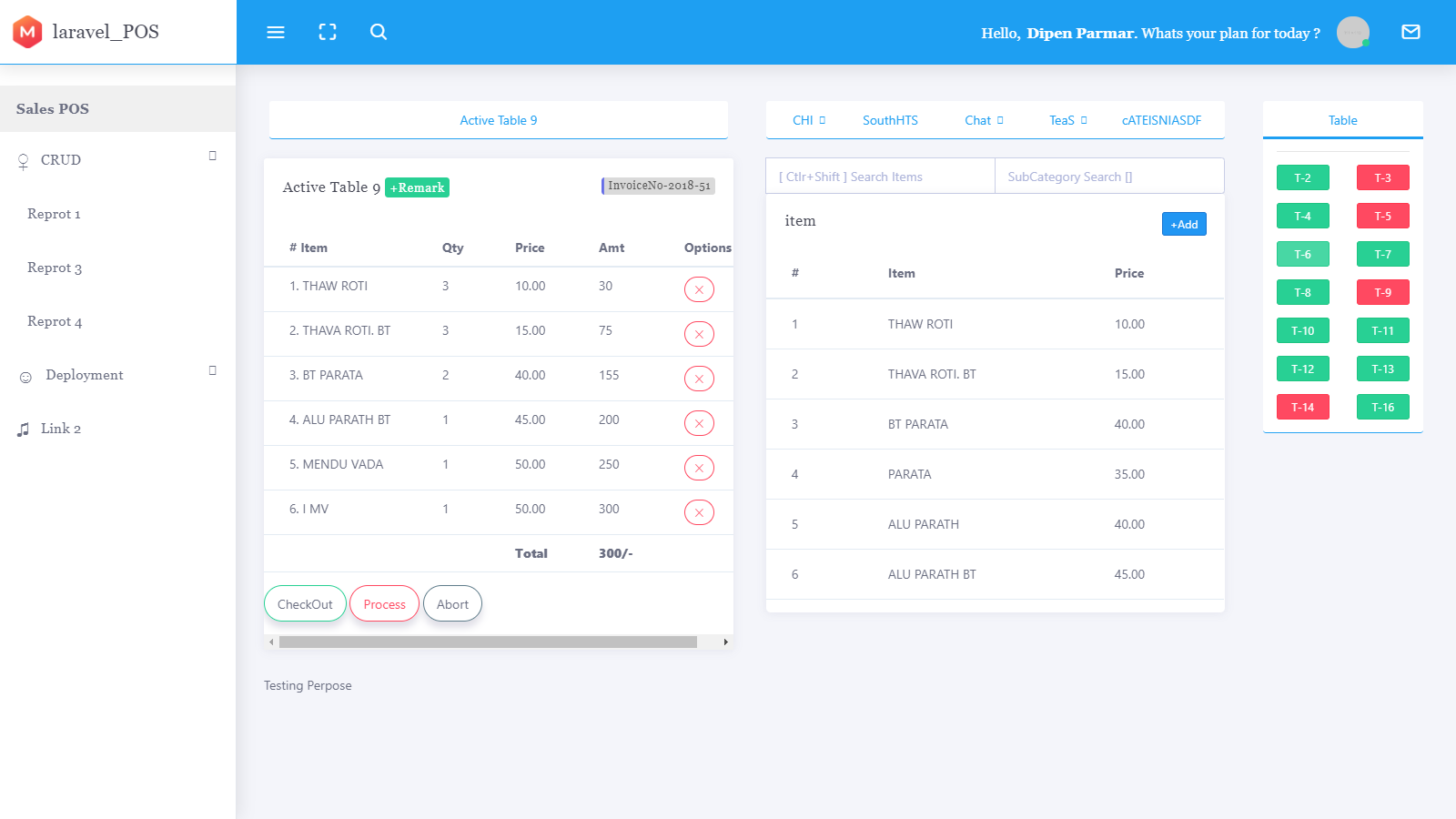
Wire framing is about creating a sample of the application to test the features, look and usability of the application before it is actually launched. It is a cheap way to test functionality and evaluate if the application serves the purpose.

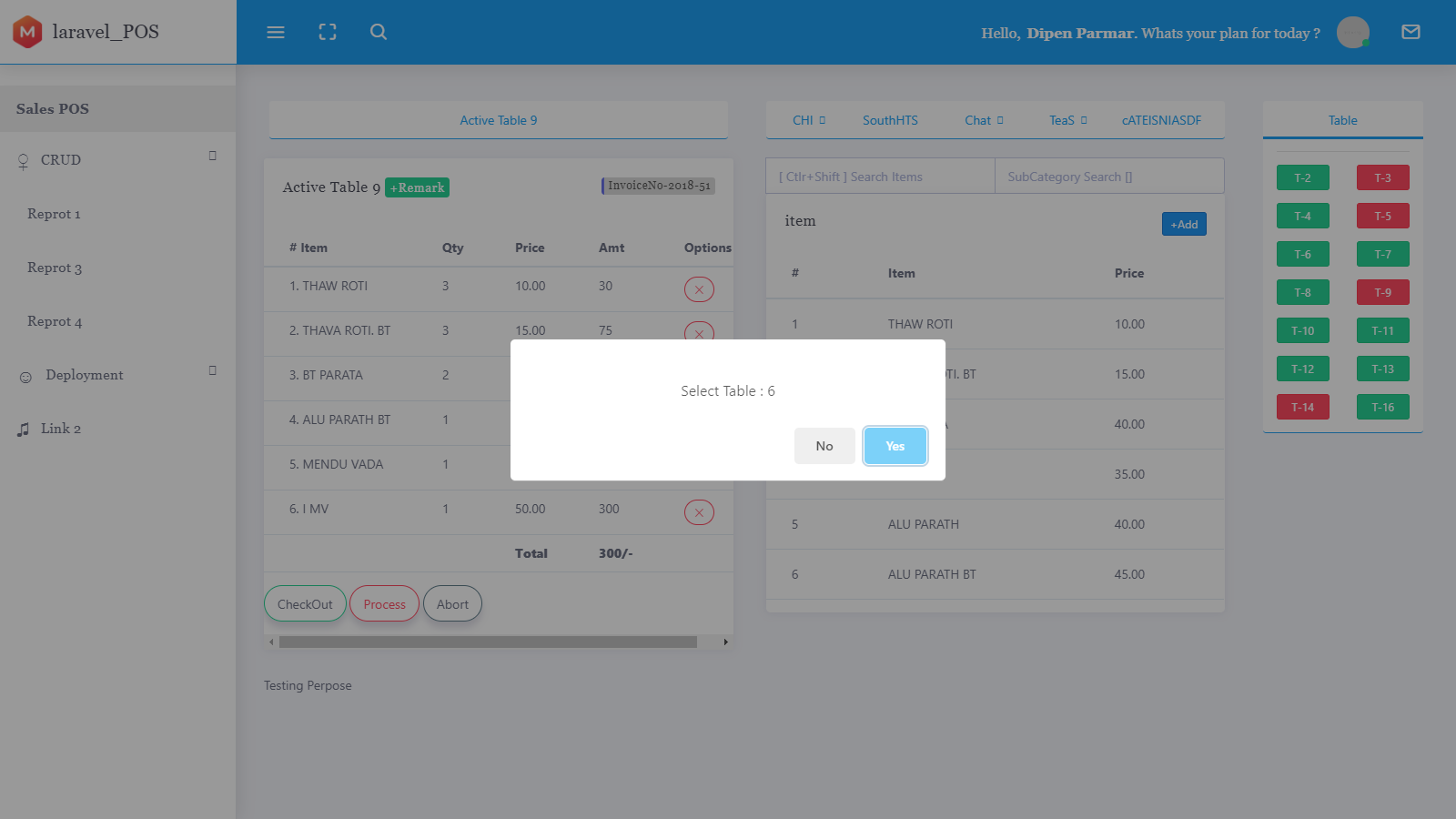
**Visual Design**

Visual design of the application or the website is like defining the company’s brand. Finalizing the visual design can affect the users’ behavior and hence it is the most important component of the UI design. Visual Design is not only about selecting the best images, colors, icons, fonts, but also identifying the appearance of the application impacts the interaction of its users. Now that you are aware with the UI/UX components, we would like to tell you how this whole procedure actually works.

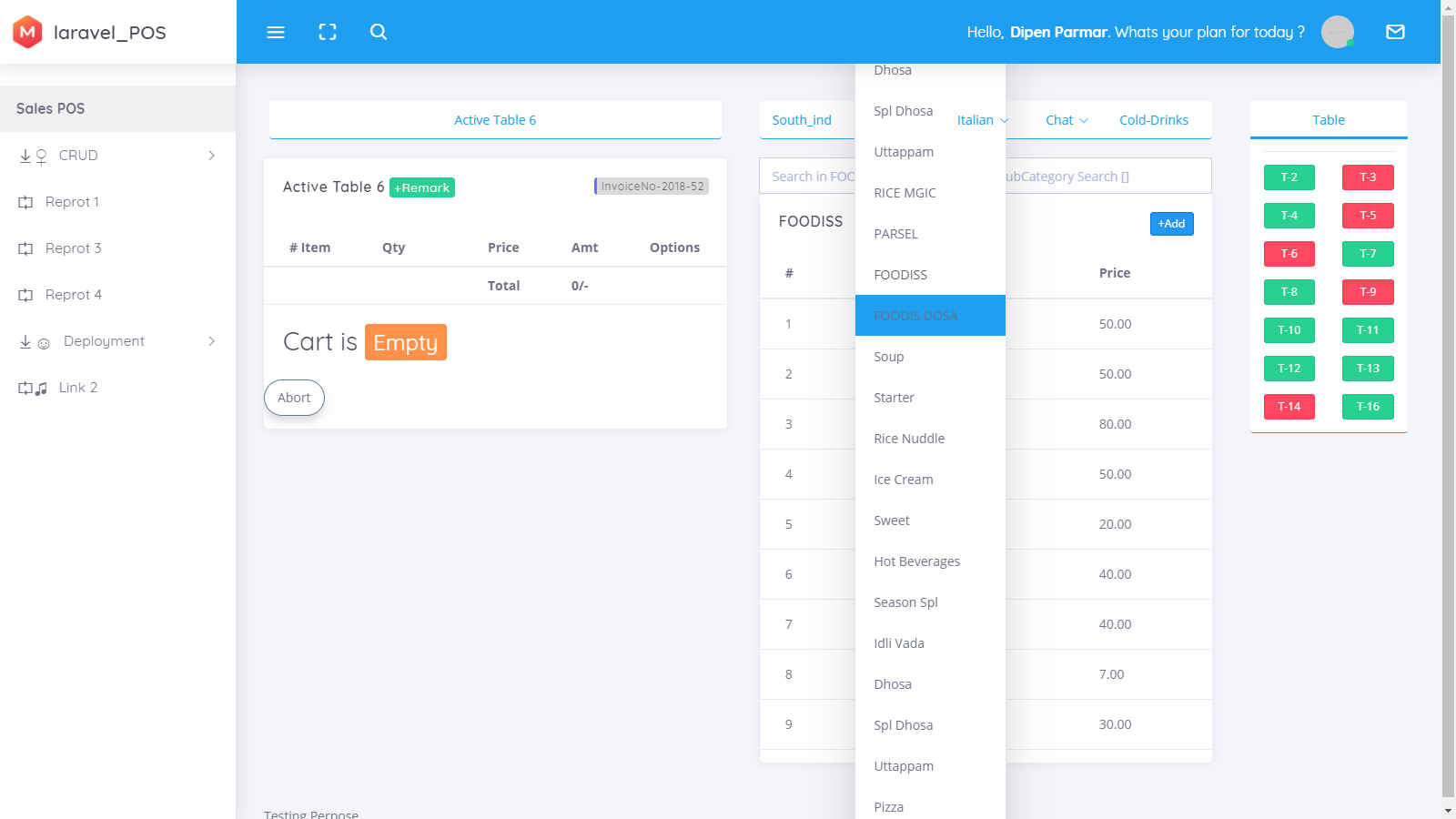
Src: <https://medium.com/@thinkwik/do-you-know-the-importance-of-ui-ux-development-773eae38436e>

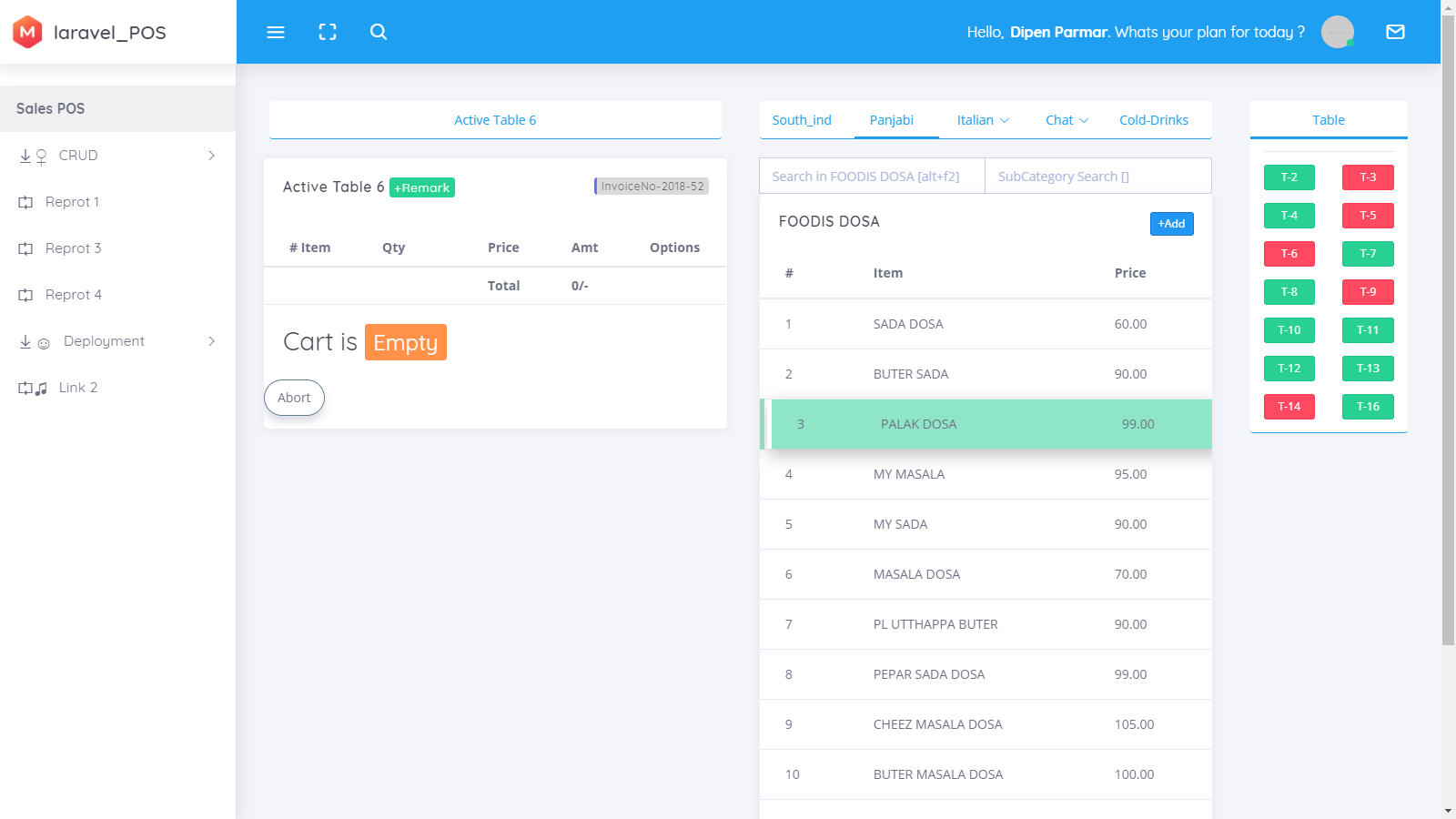
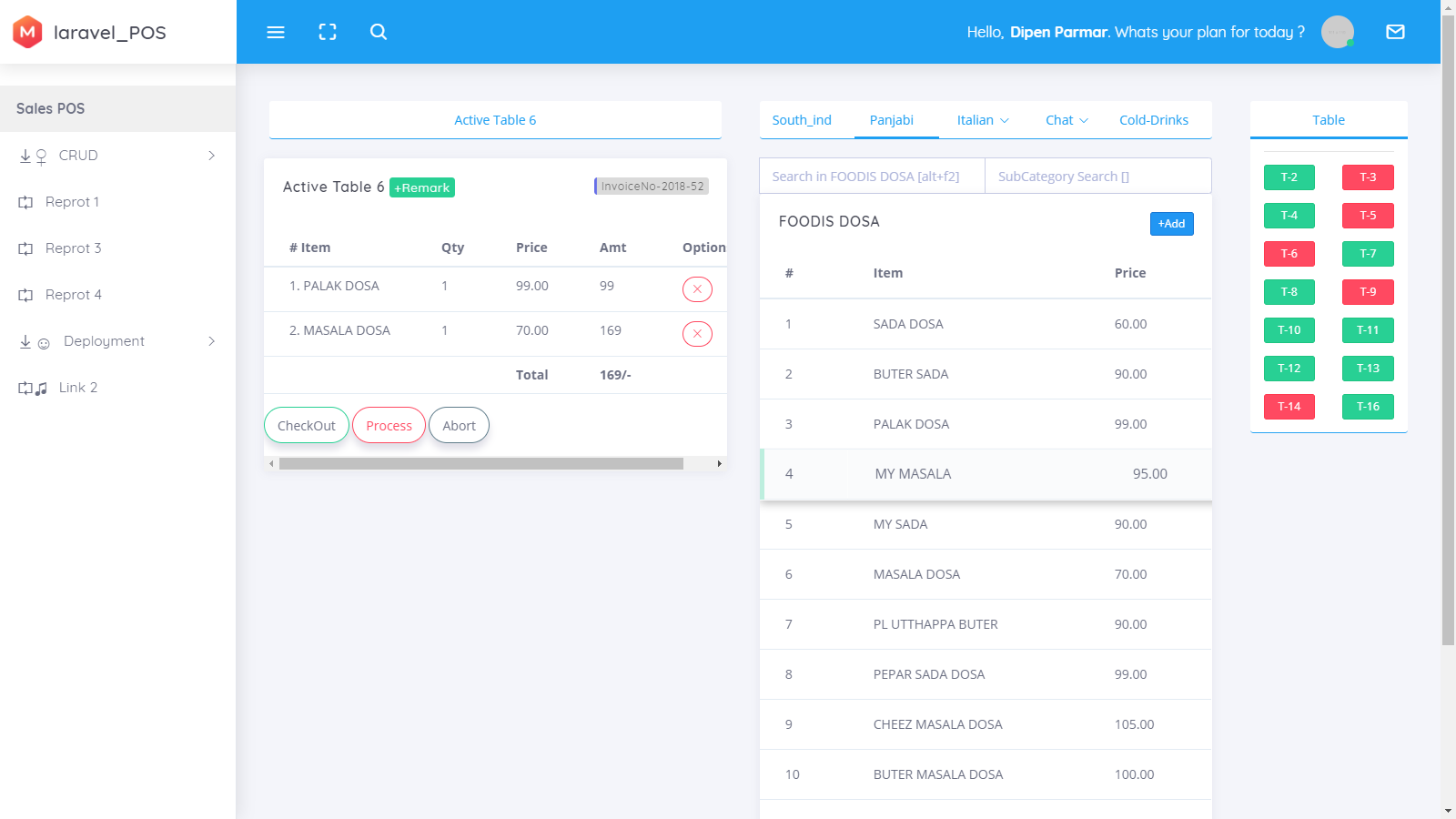


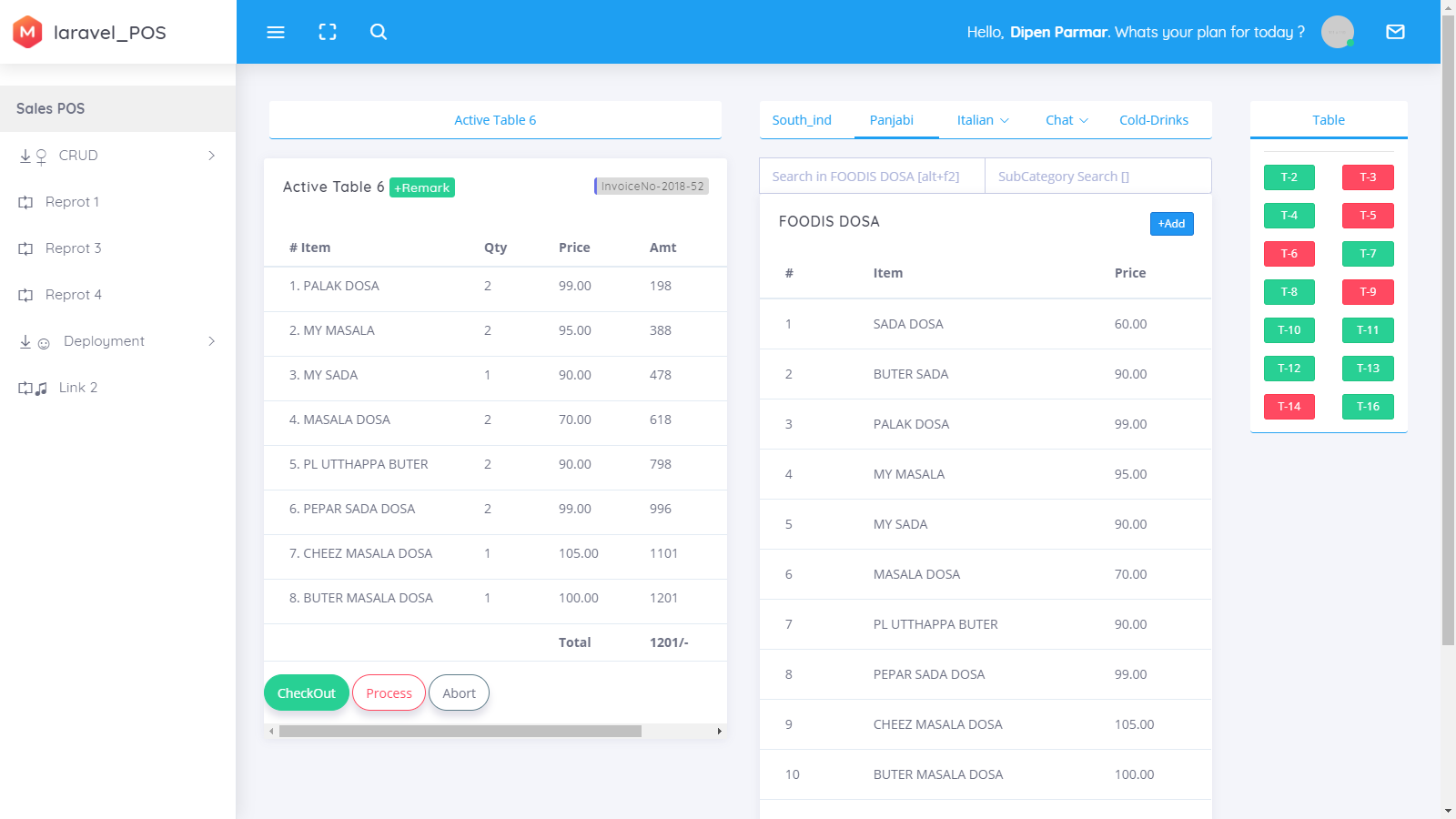


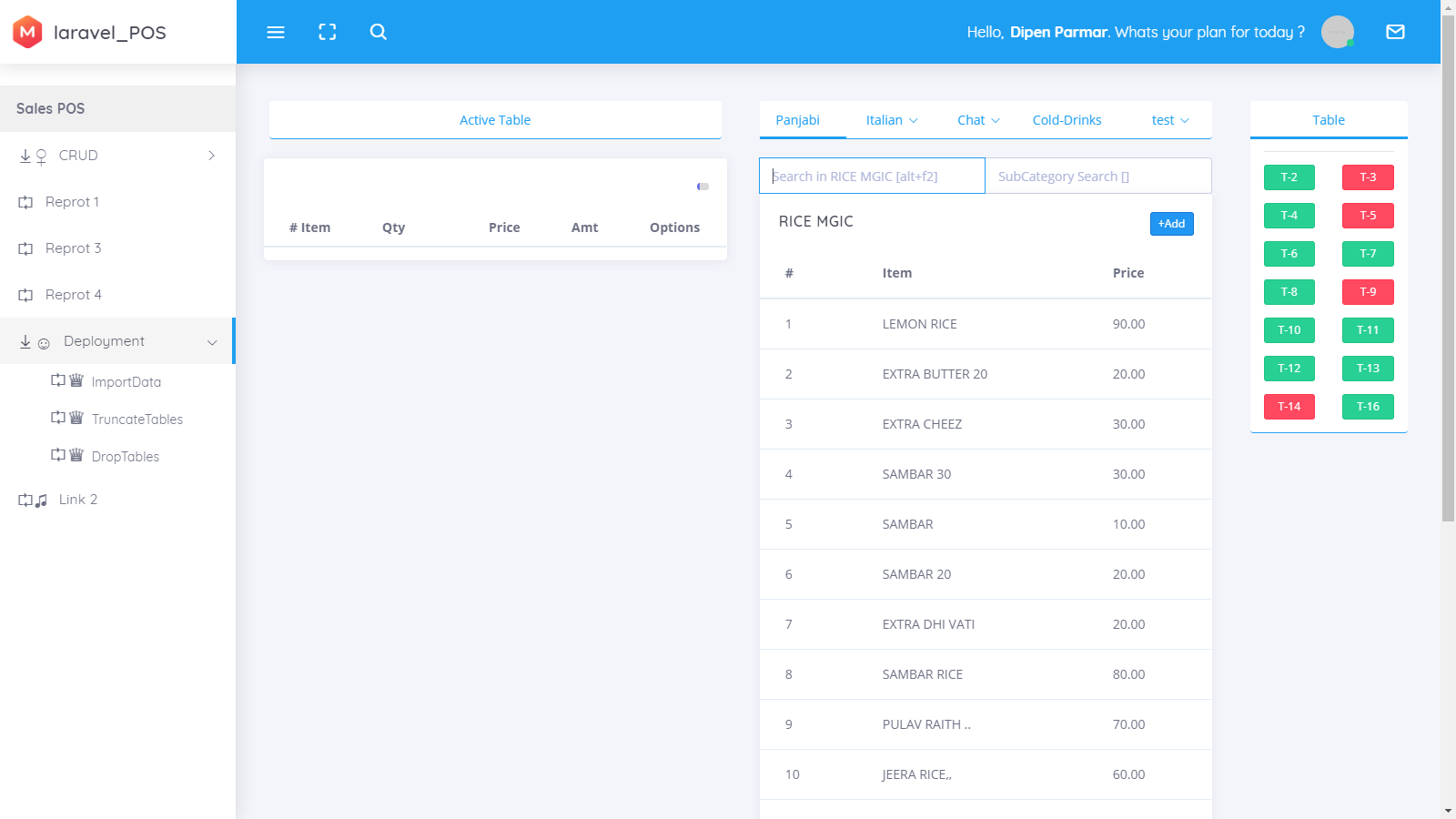


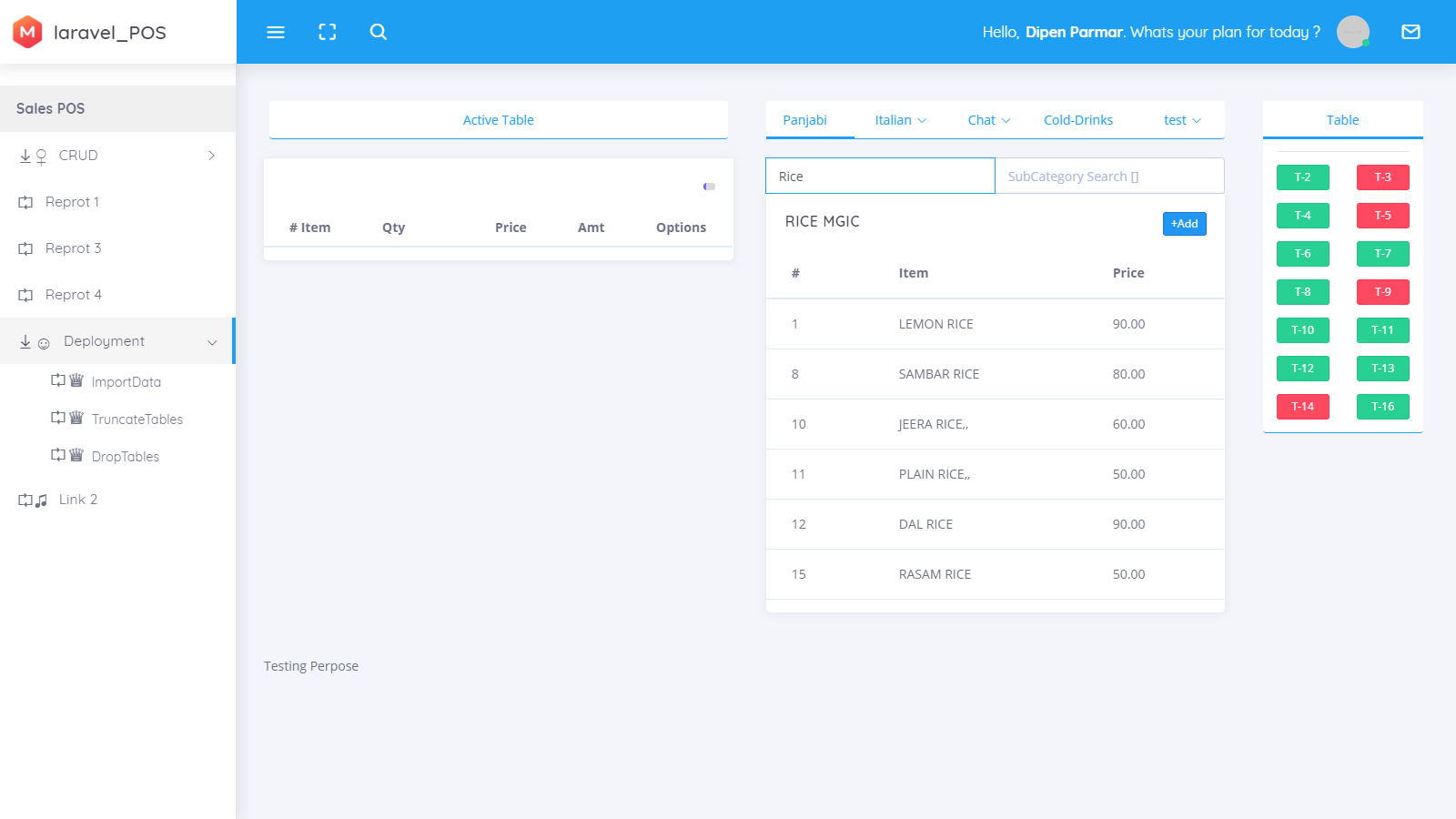


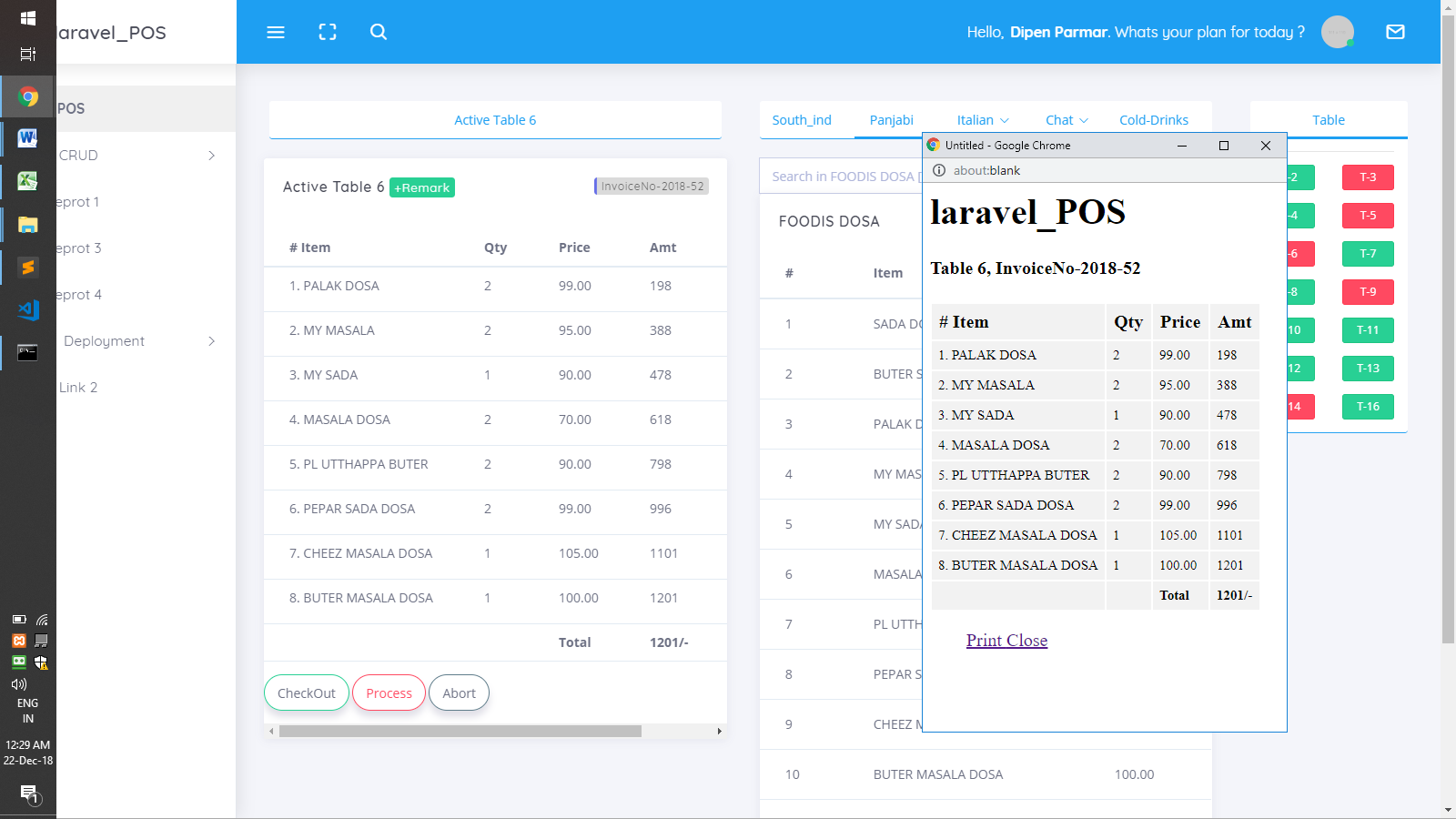


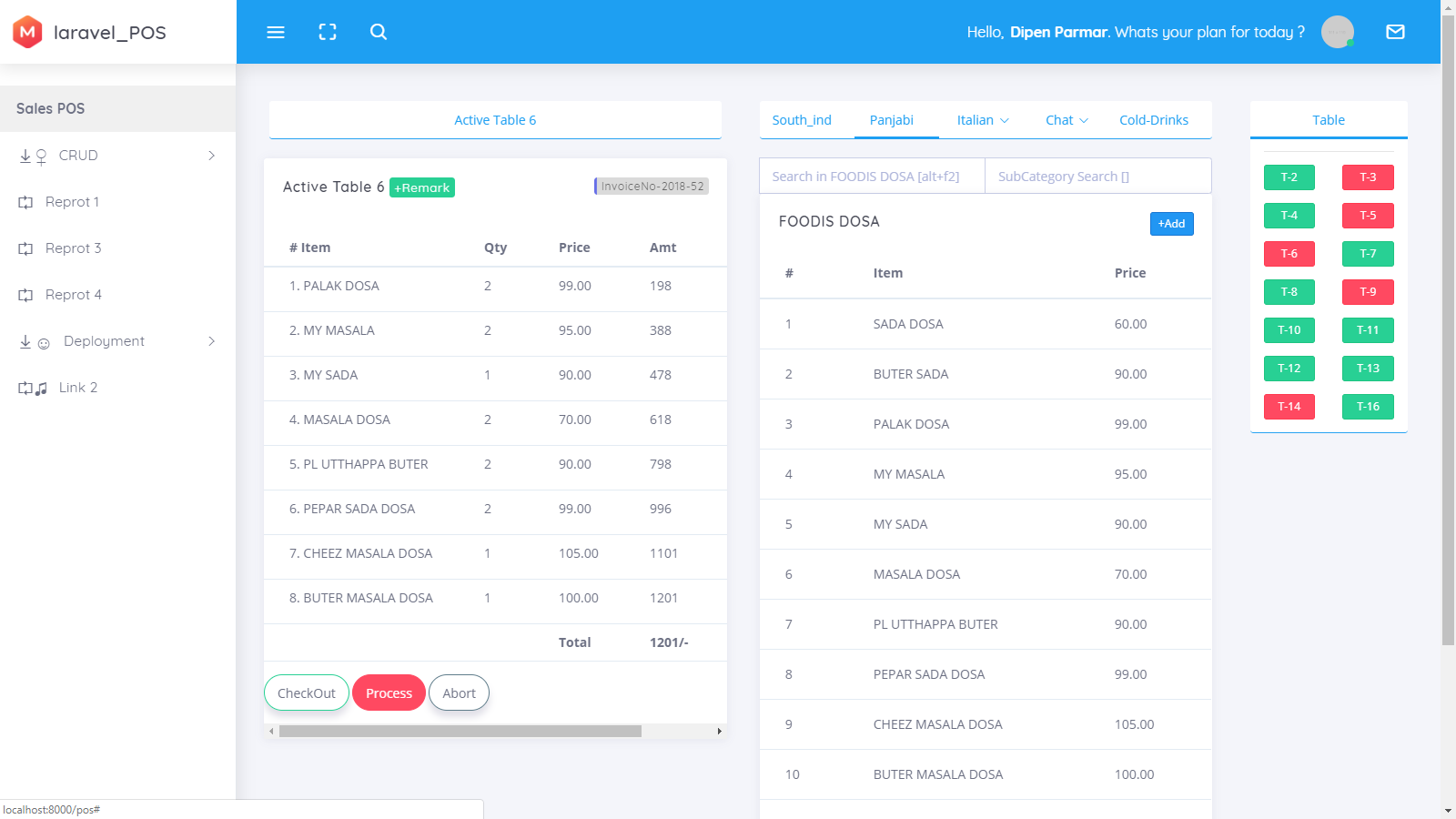


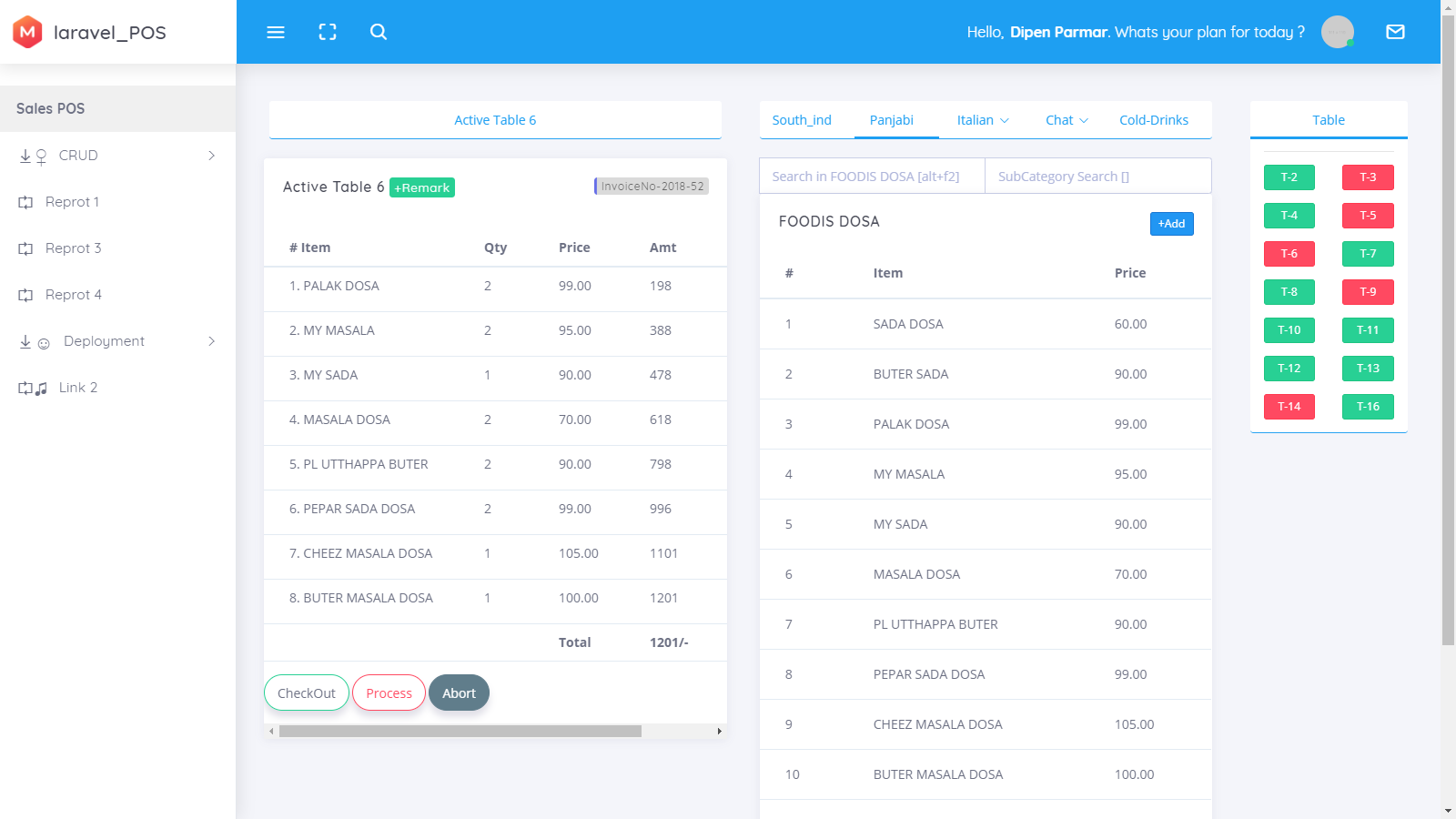




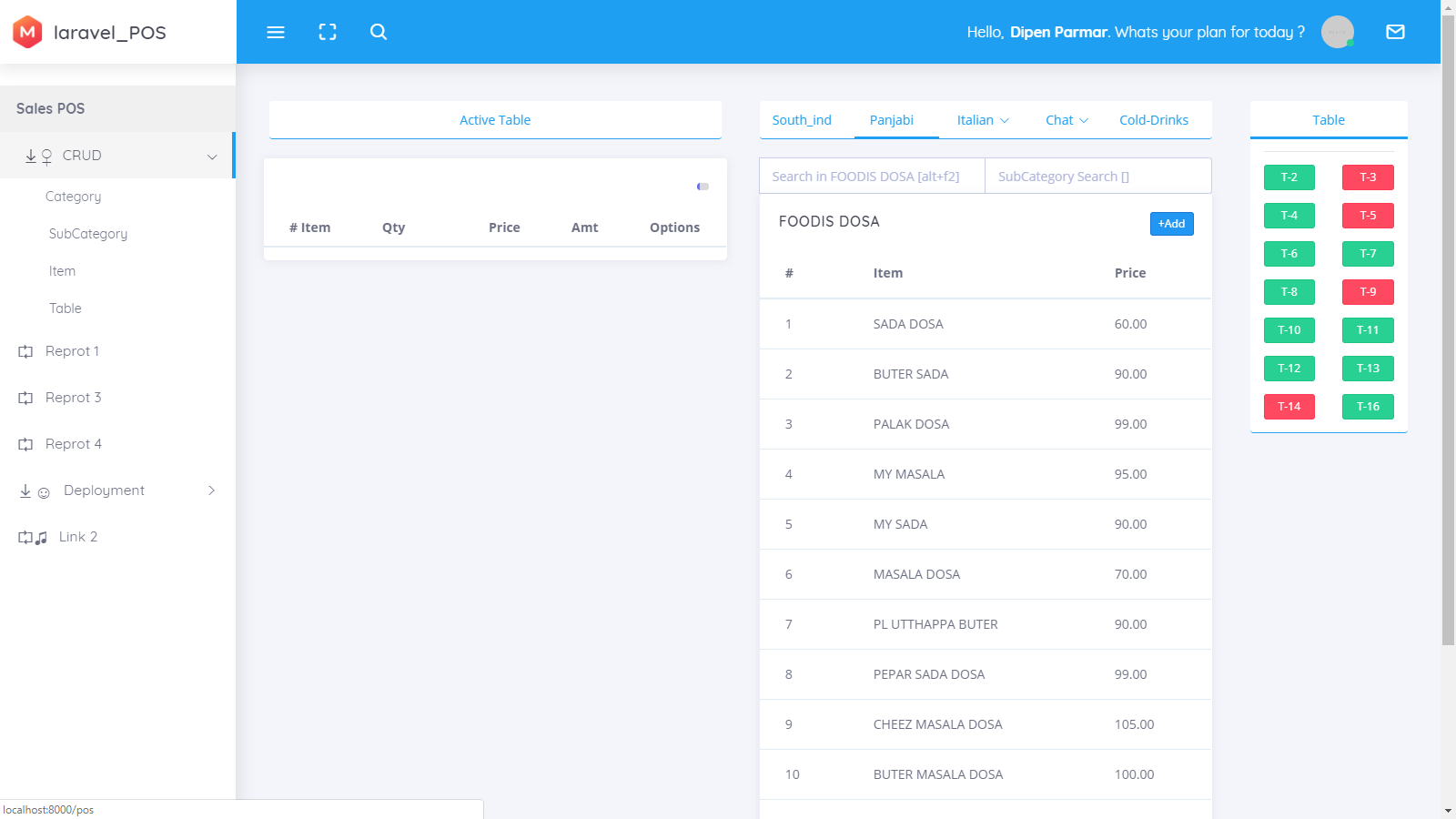


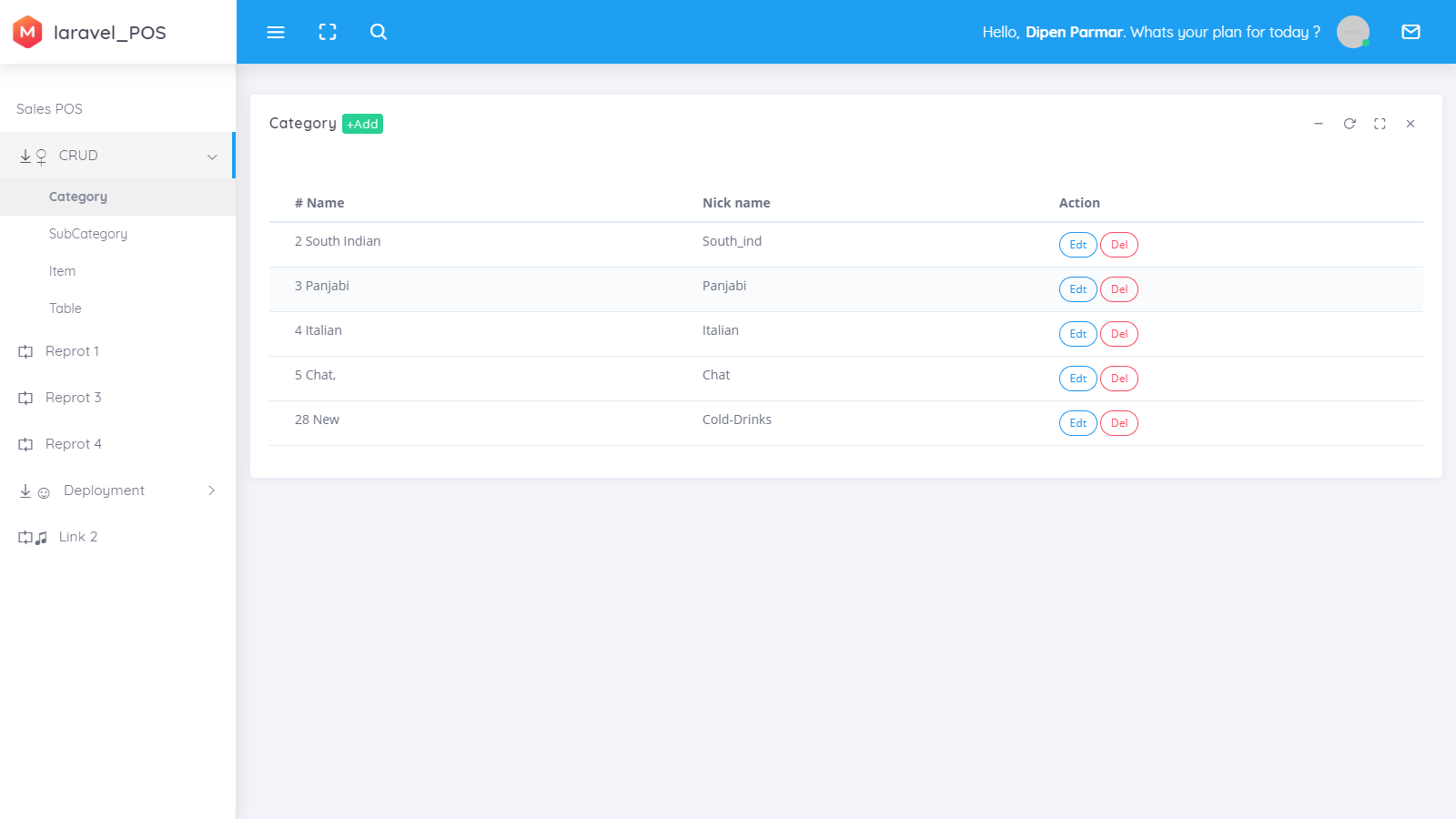




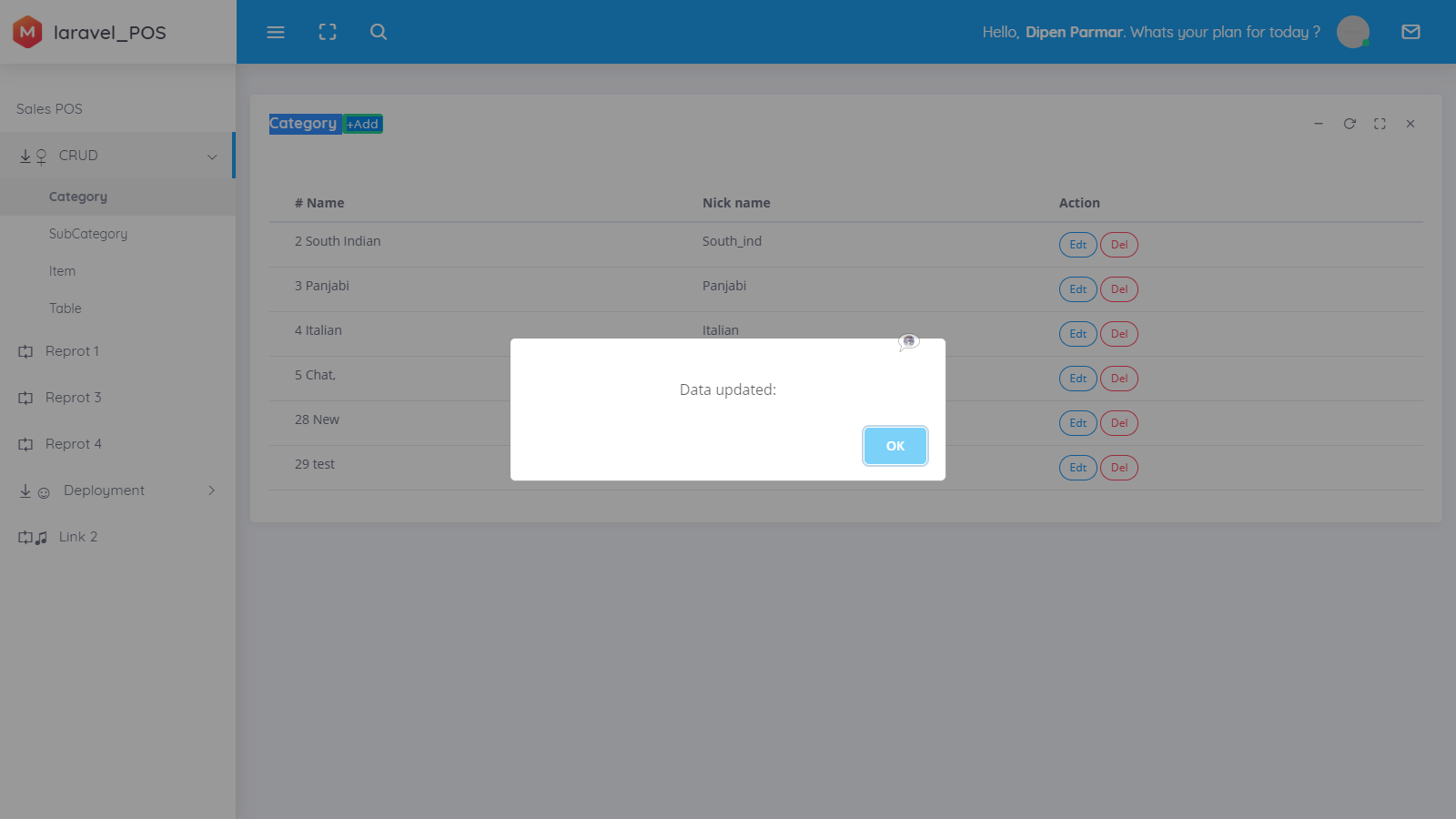


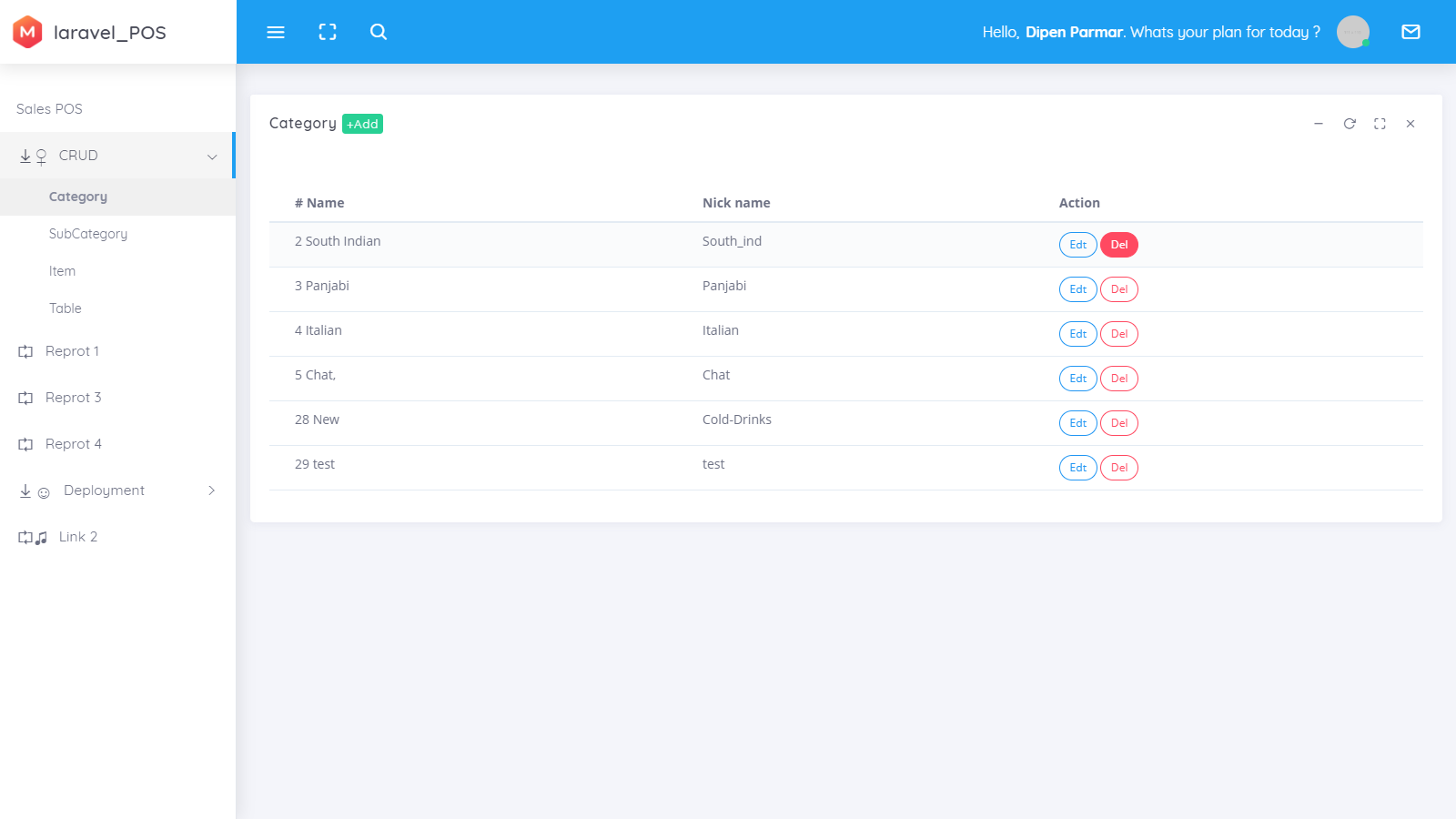


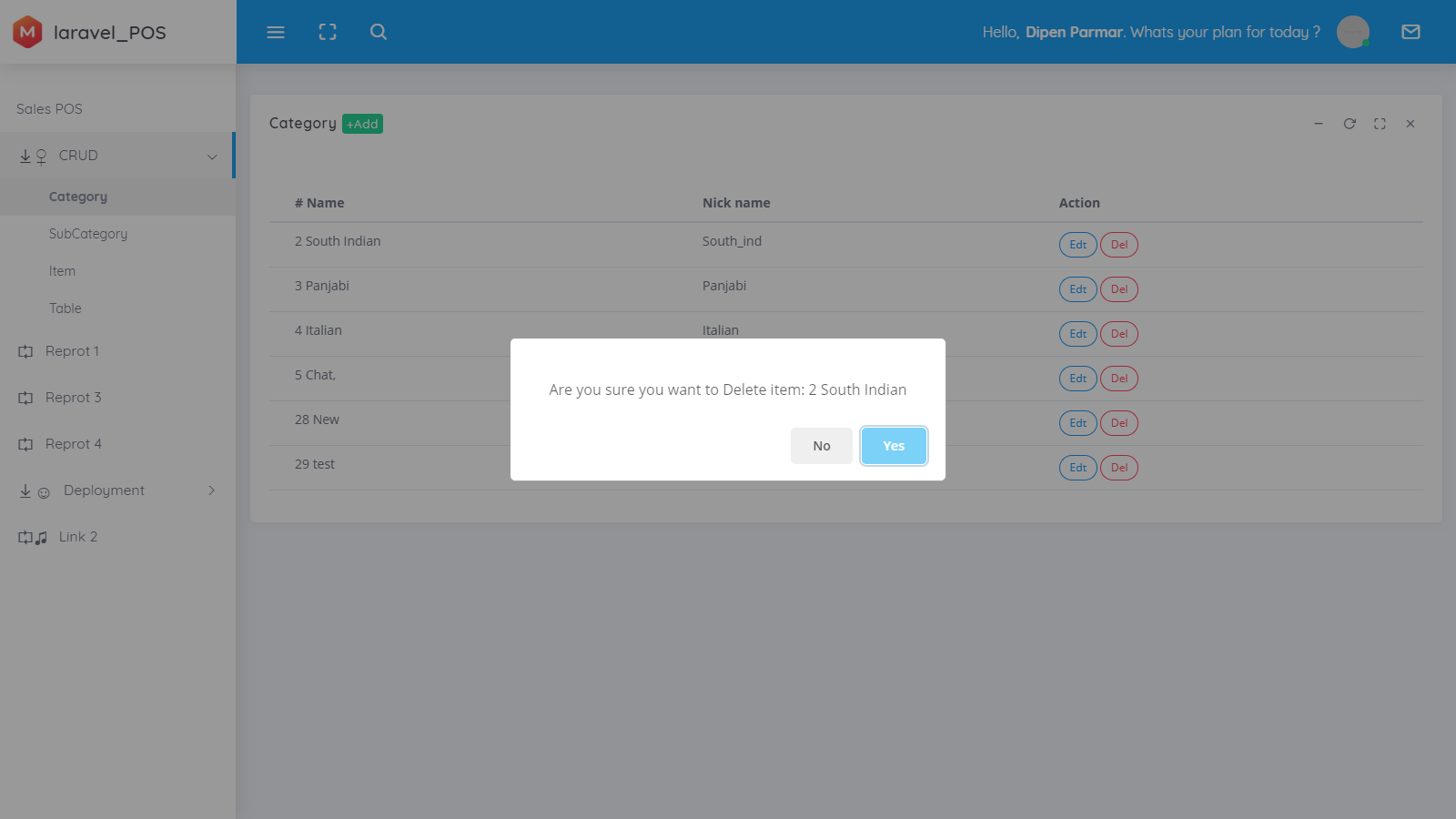


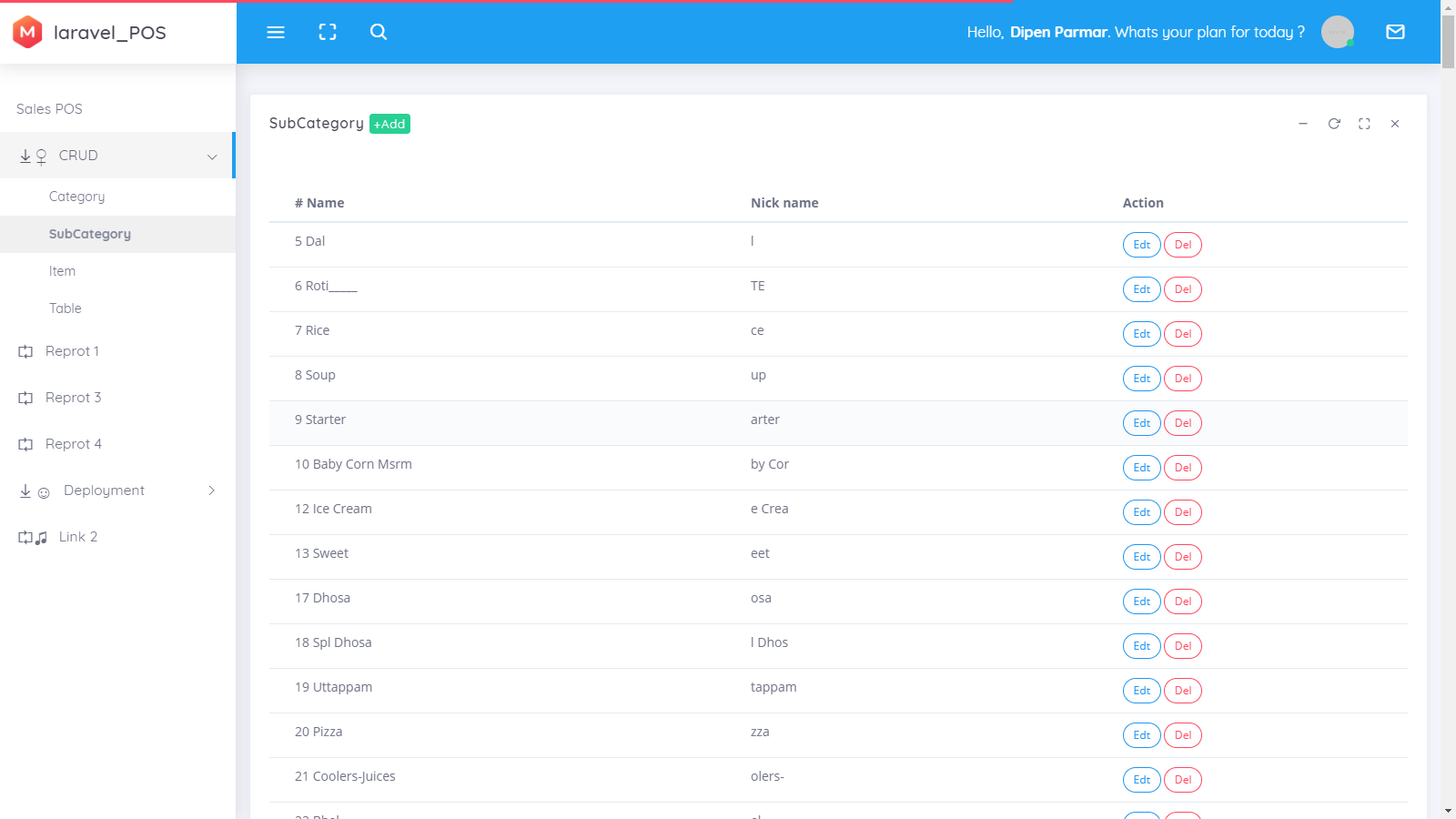


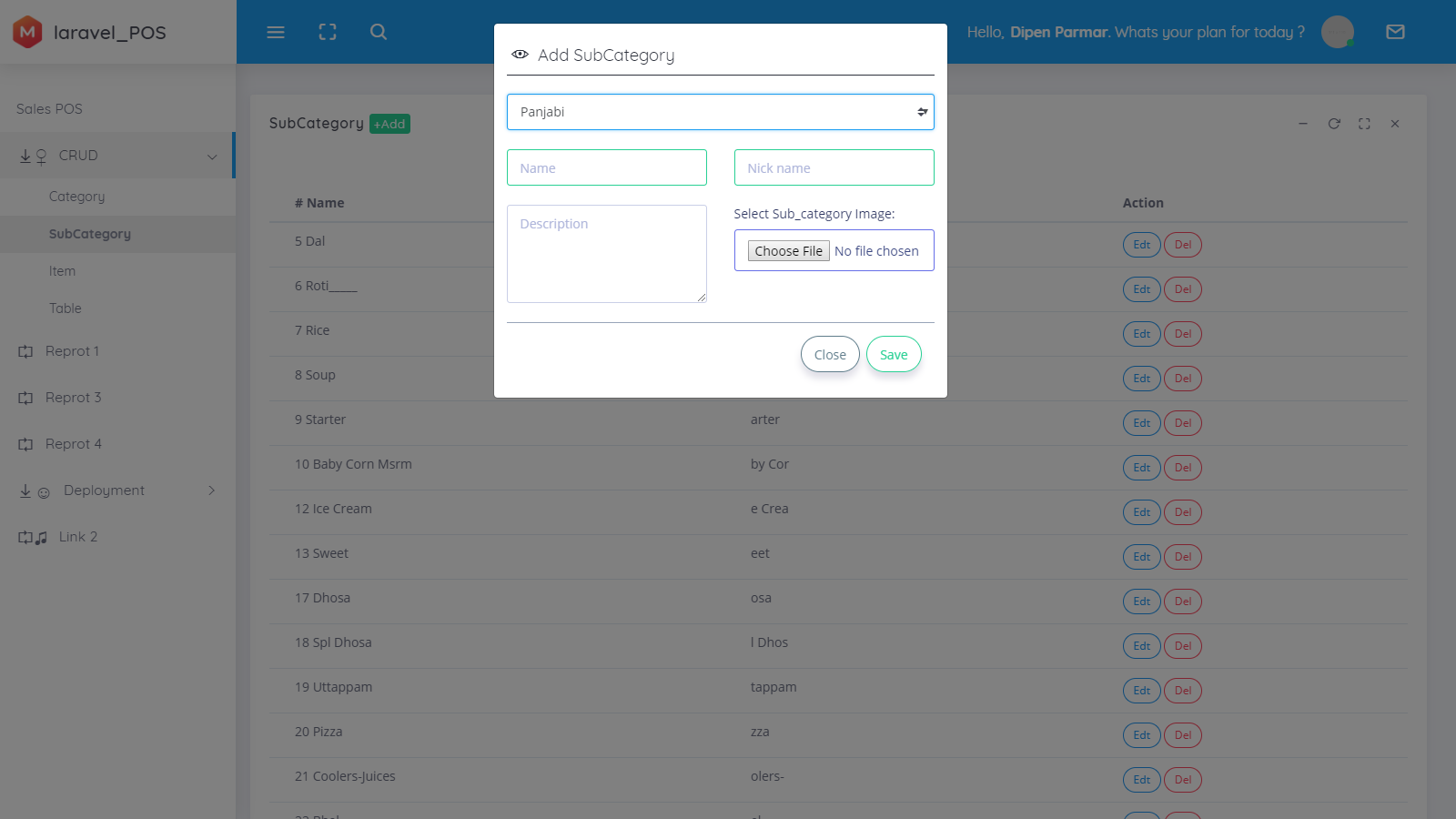


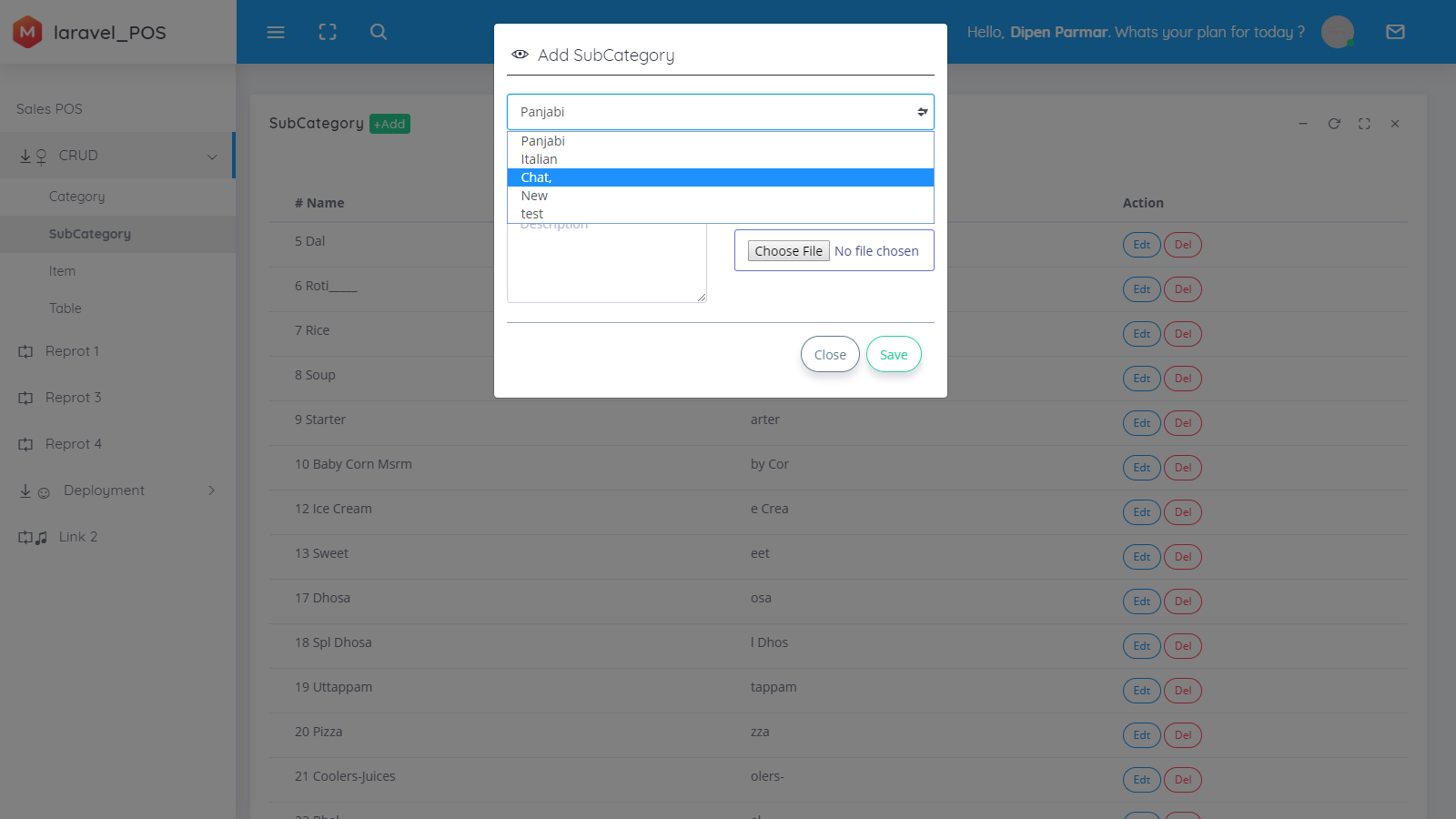


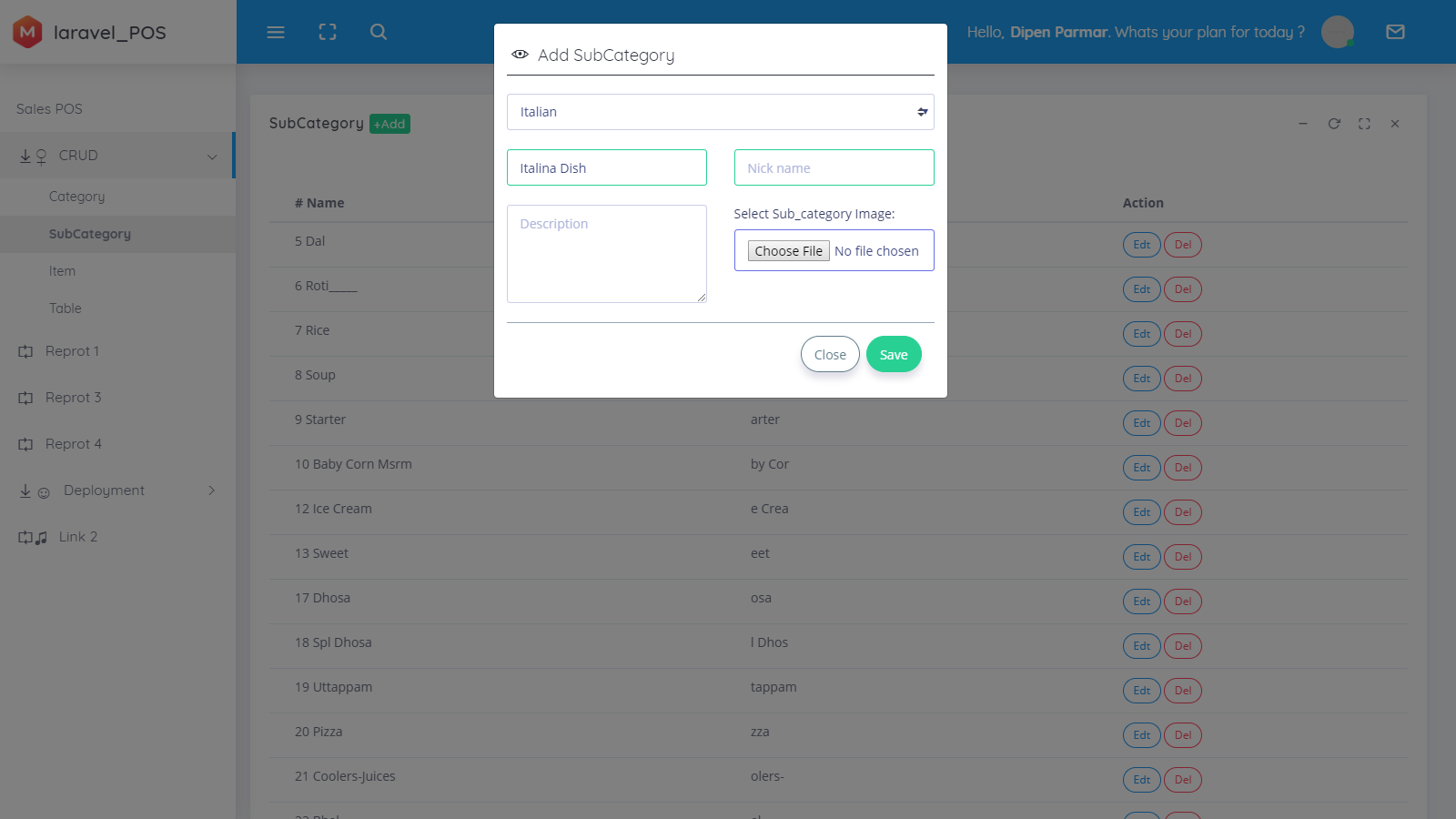


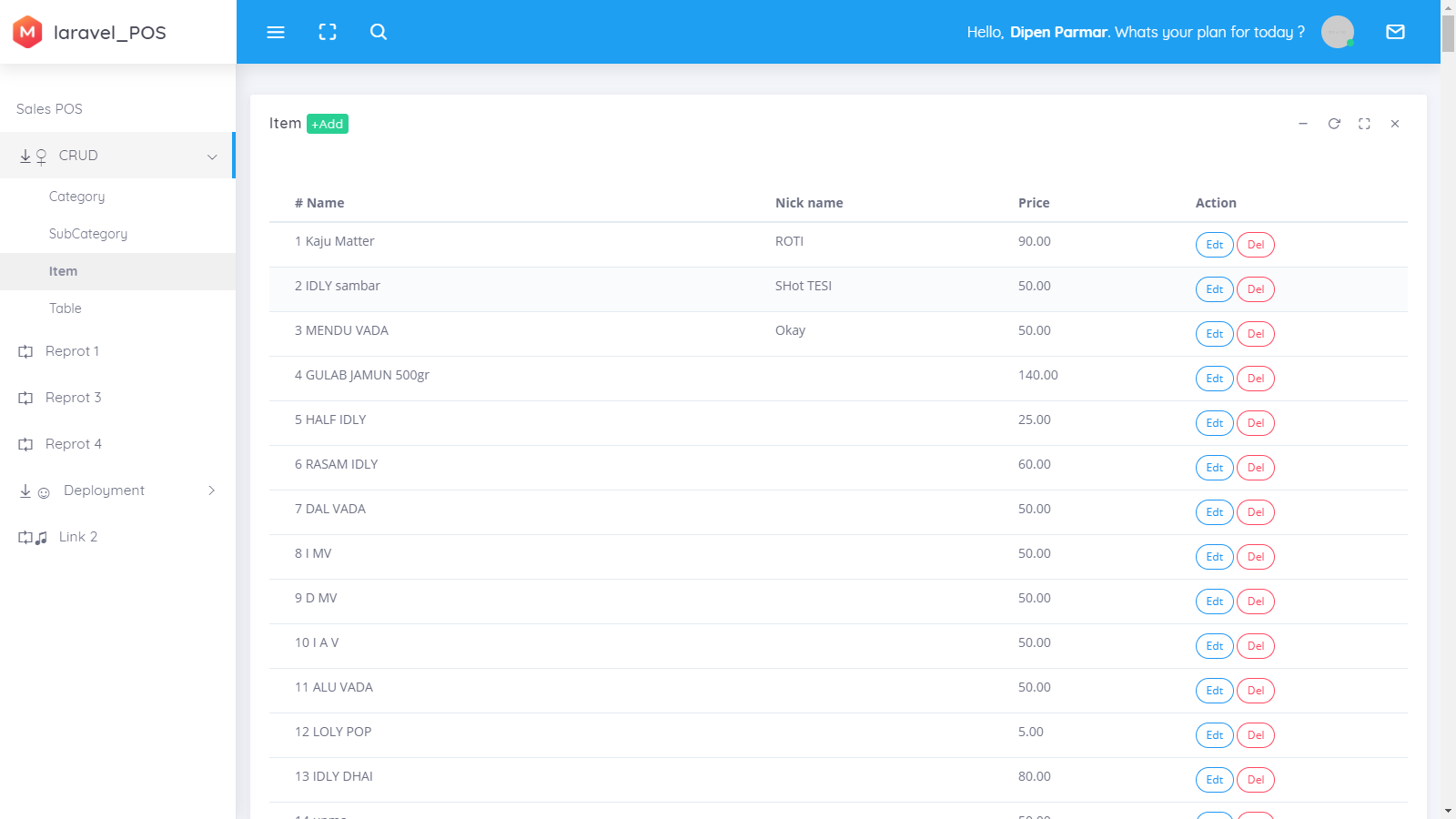


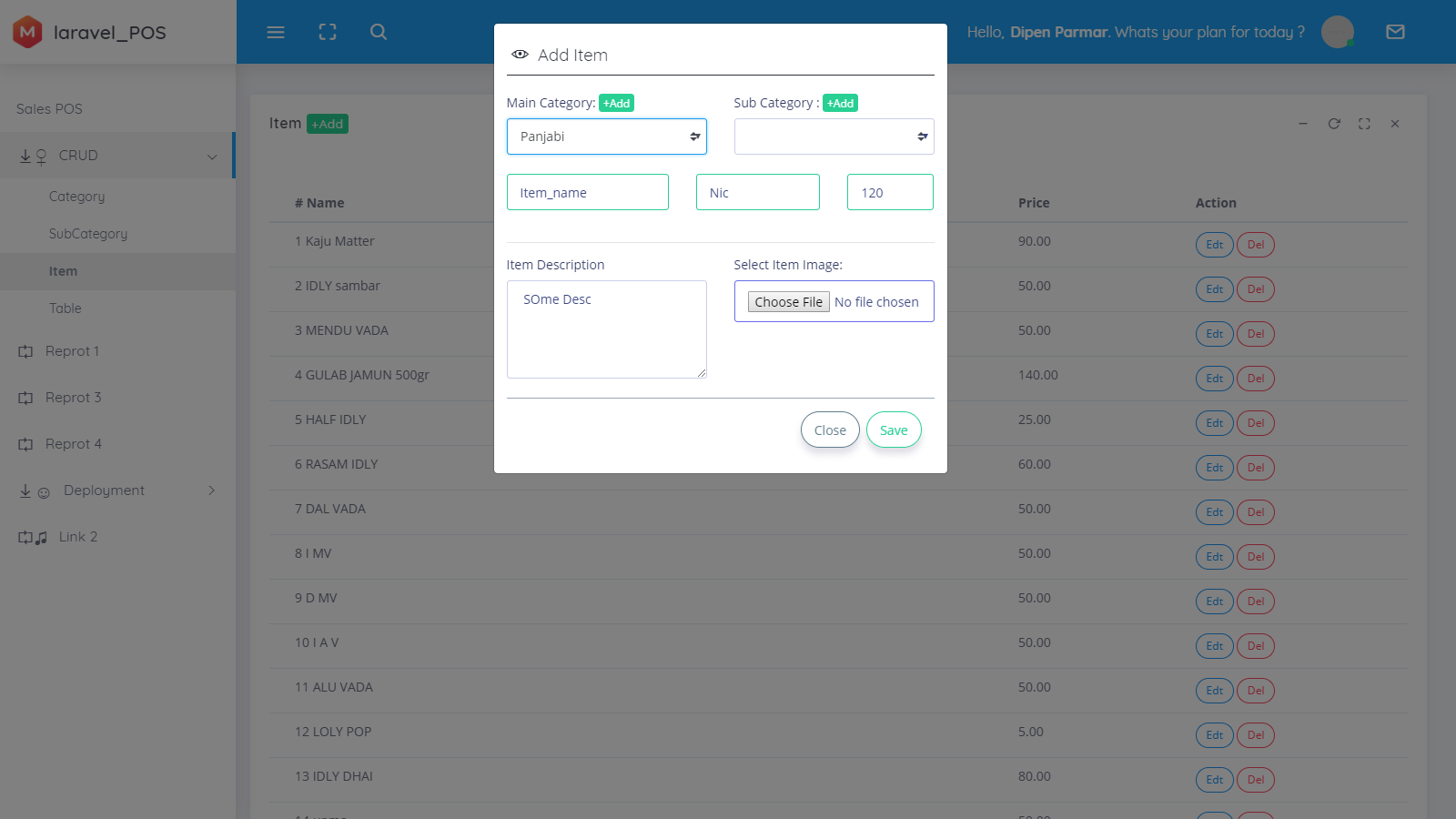


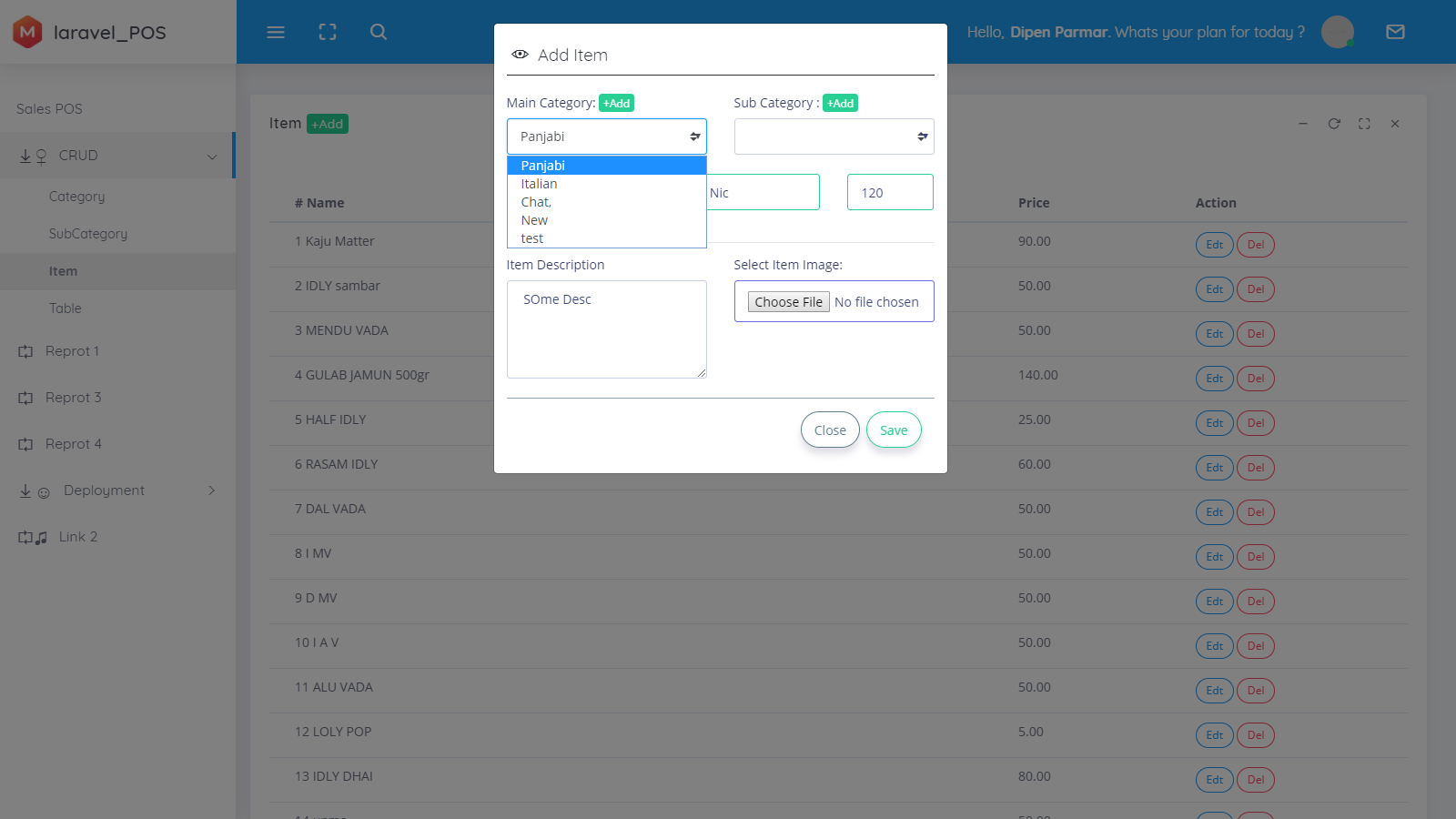


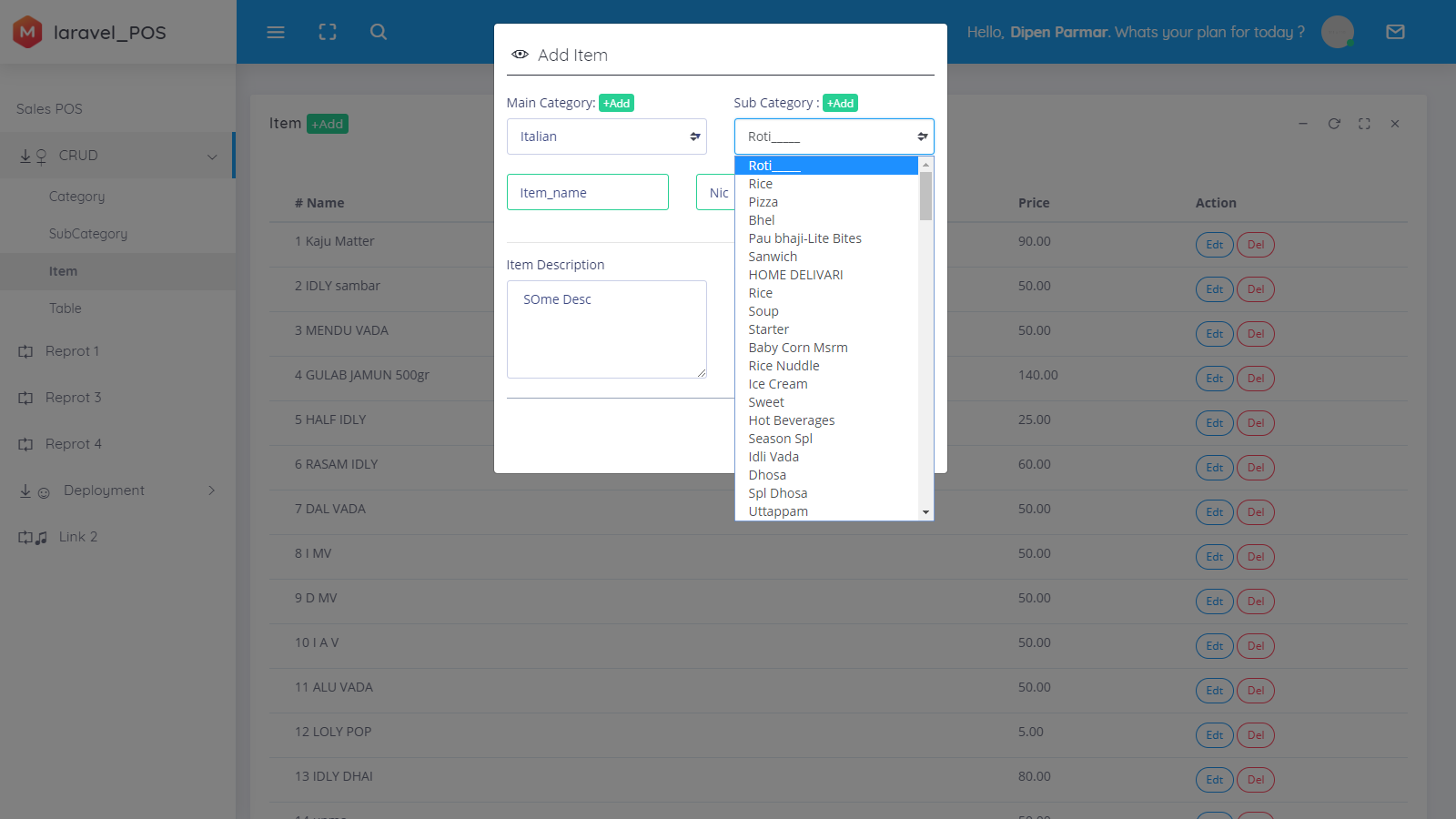


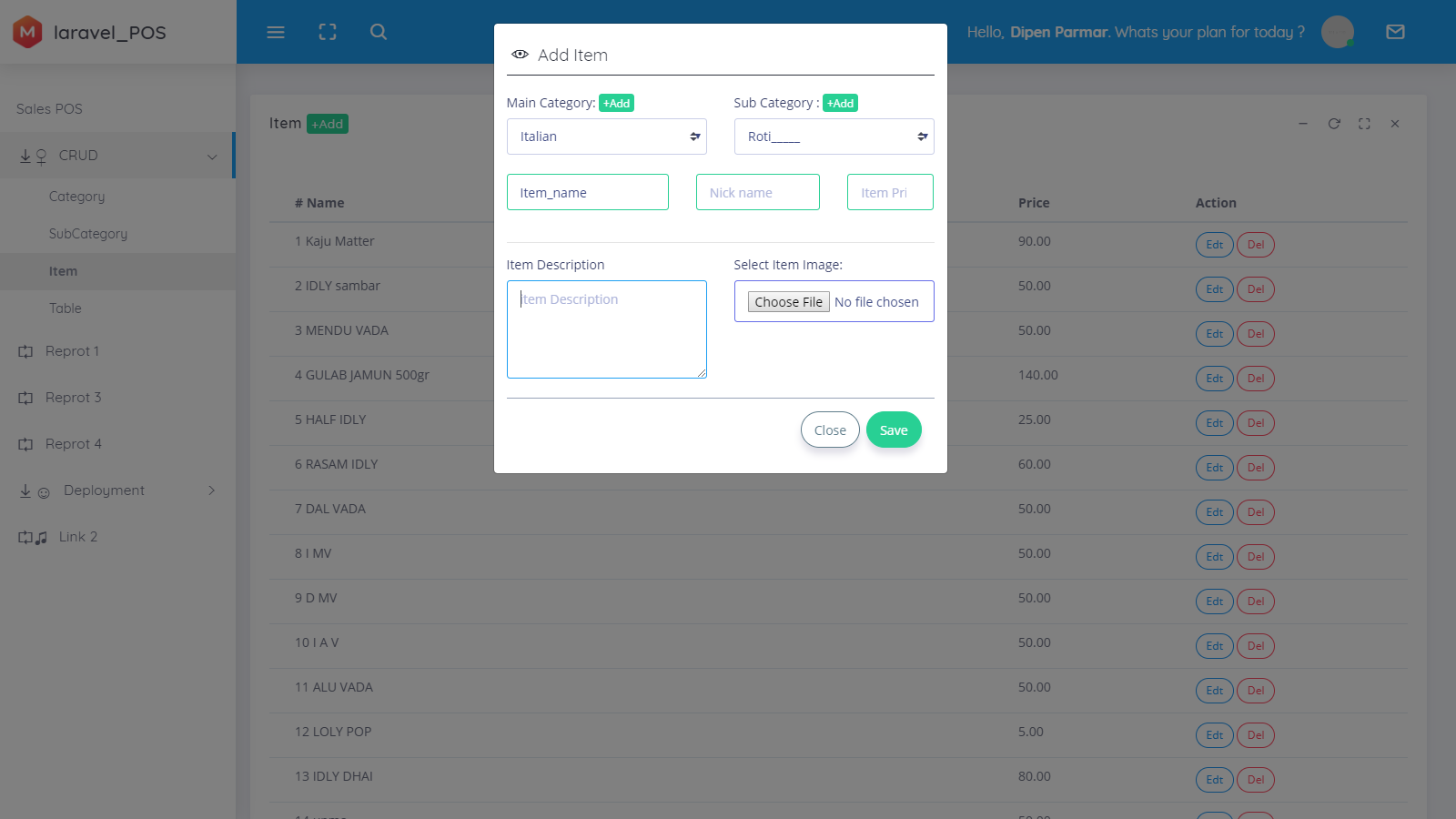












**DATA DICTIONARY**

**Data dictionary**: is a file or a set of files that contains a database's metadata. The data dictionary contains records about other objects in the database, such as data ownership, data relationships to other objects, and other data.

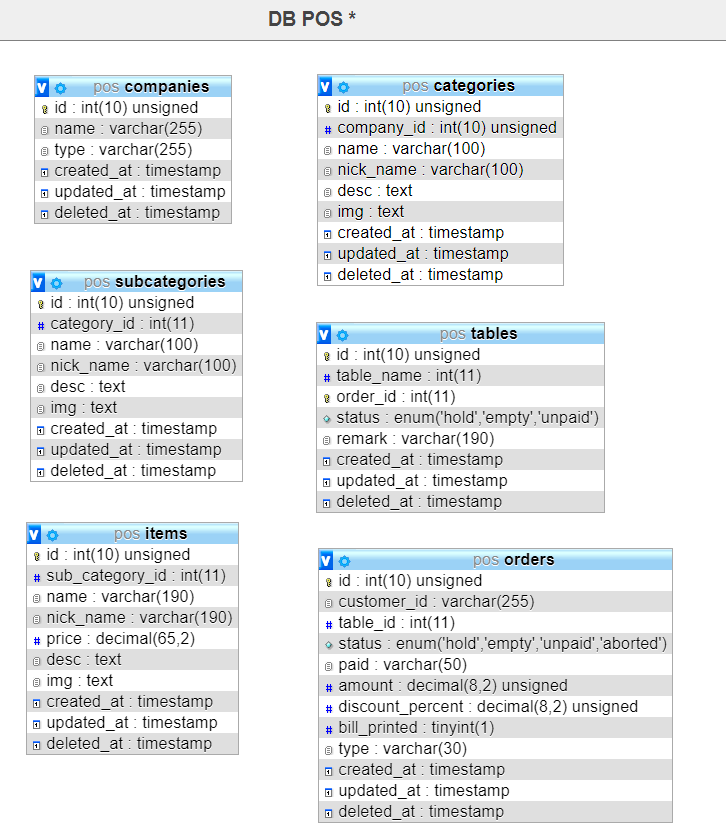
The data dictionary is a crucial component of any relational database. Ironically, because of its importance, it is invisible to most database users. Typically, only database administrators interact with the data dictionary.

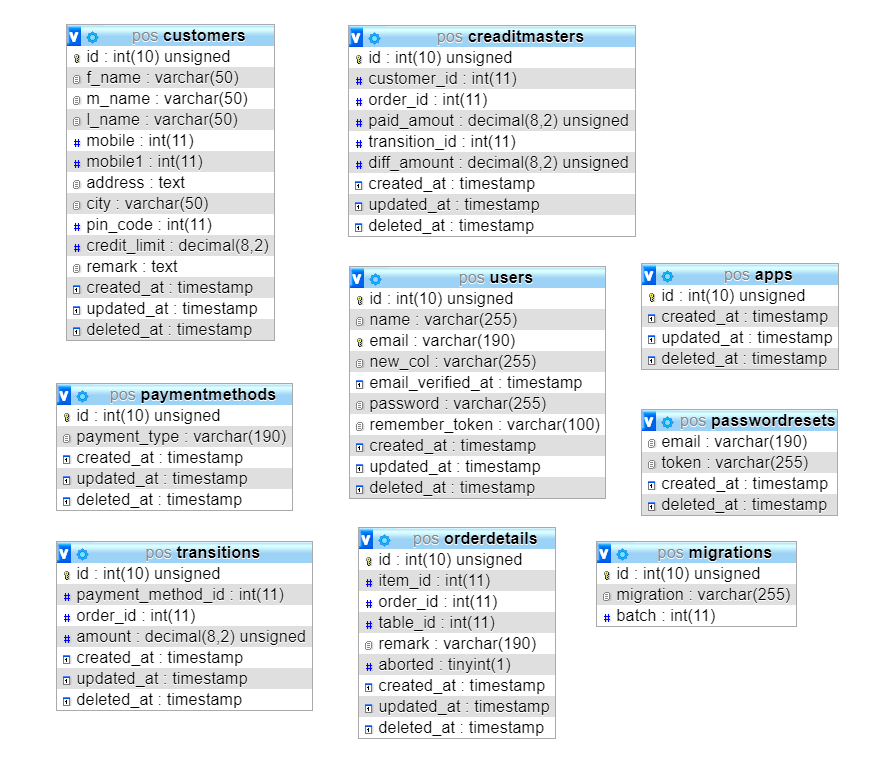
In a relational database, the **metadata** in the data dictionary includes the following:

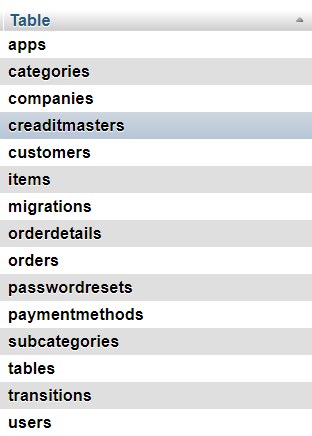
* Names of all tables in the database and their owners
* Names of all indexes and the columns to which the tables in those indexes relate
* Constraints defined on tables, including primary keys, foreign-key relationships to other tables, and not-null constraints

For most relational database management systems (RDBMS), the database management system software needs the data dictionary to access the data within a database. For example, the Oracle DB software has to read and write to an Oracle DB. However, it can only do this via the data dictionary created for that particular database.

Src: <https://www.techopedia.com/definition/27752/data-dictionary>





**DATA FLOW DIAGRAM (DFD)**

A data flow diagram shows the way information flows through a process or system. It includes data inputs and outputs, data stores, and the various sub processes the data moves through. DFDs are built using standardized symbols and notation to describe various entities and their relationships.

Data flow diagrams visually represent systems and processes that would be hard to describe in a chunk of text. You can use these diagrams to map out an existing system and make it better or to plan out a new system for implementation. Visualizing each element makes it easy to identify inefficiencies and produce the best possible system.

**Physical and logical data flow diagrams**

Before actually creating your data flow diagram, you’ll need to determine whether a physical or logical DFD best suits your needs. If you’re new to data flow diagrams, don’t worry—the distinction is pretty straightforward.

**Logical data flow** diagrams focus on what happens in a particular information flow: what information is being transmitted, what entities are receiving that info, what general processes occur, etc. The processes described in a logical DFD are business activities—a logical DFD doesn’t delve into the technical aspects of a process or system. Non-technical employees should be able to understand these diagrams.

**Physical data flow** diagrams focus on how things happen in an information flow. These diagrams specify the software, hardware, files, and people involved in an information flow. A detailed physical data flow diagram can facilitate the development of the code needed to implement a data system.

**Symbols and Notations Used in DFDs**

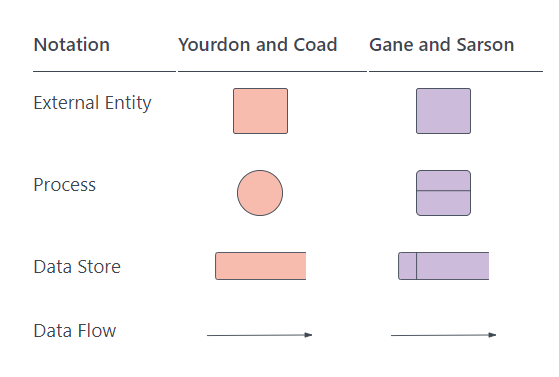
Two common systems of symbols are named after their creators:

* Yourdon and Coad
* Yourdon and DeMarco
* Gane and Sarson

One main difference in their symbols is that Yourdon-Coad and Yourdon-DeMarco use circles for processes, while Gane and Sarson use rectangles with rounded corners, sometimes called lozenges. There are other symbol variations in use as well, so the important thing to keep in mind is to be clear and consistent in the shapes and notations you use to communicate and collaborate with others.

Using any convention’s DFD rules or guidelines, the symbols depict the four components of data flow diagrams.

1. **External entity:** an outside system that sends or receives data, communicating with the system being diagrammed. They are the sources and destinations of information entering or leaving the system. They might be an outside organization or person, a computer system or a business system. They are also known as terminators, sources and sinks or actors. They are typically drawn on the edges of the diagram.
2. **Process:**any process that changes the data, producing an output. It might perform computations, or sort data based on logic, or direct the data flow based on business rules. A short label is used to describe the process, such as “Submit payment.”
3. **Data store:** files or repositories that hold information for later use, such as a database table or a membership form. Each data store receives a simple label, such as “Orders.”
4. **Data flow:** the route that data takes between the external entities, processes and data stores. It portrays the interface between the other components and is shown with arrows, typically labeled with a short data name, like “Billing details.”



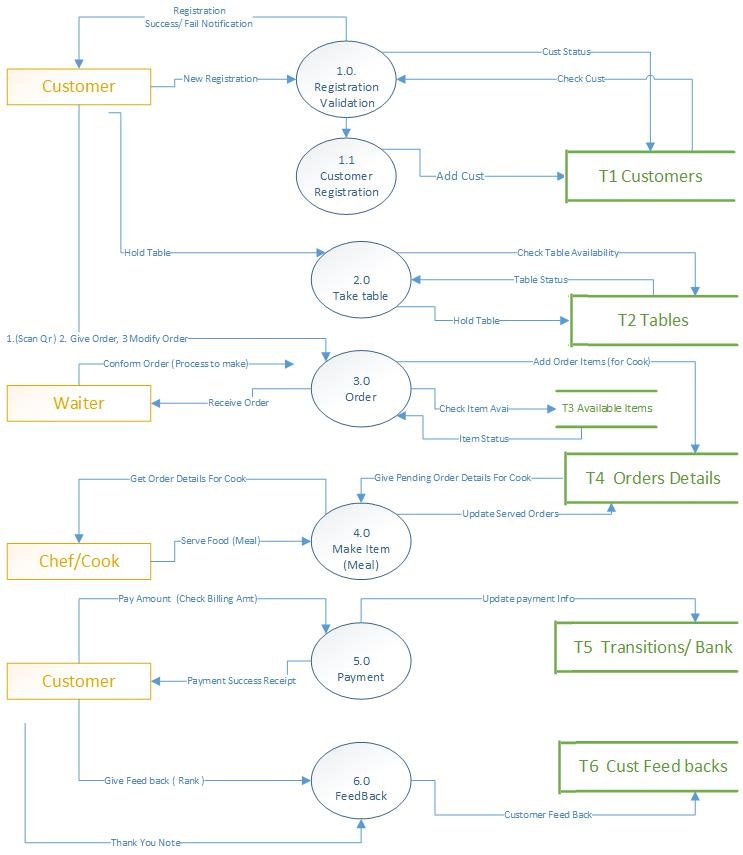
## DFD rules and tips

* Each process should have at least one input and an output.
* Each data store should have at least one data flow in and one data flow out.
* Data stored in a system must go through a process.
* All processes in a DFD go to another process or a data store.

Src: <https://www.lucidchart.com/pages/data-flow-diagram>

Src: <https://www.lucidchart.com/blog/data-flow-diagram-tutorial>

**Data flow diagram Restaurant Management System**



**E-R DIAGRAM**

An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system. ER Diagrams are most often used to design or debug relational databases in the fields of software engineering, business information systems, education and research. Also known as ERDs or ER Models, they use a defined set of symbols such as rectangles, diamonds, ovals and connecting lines to depict the interconnectedness of entities, relationships and their attributes. They mirror grammatical structure, with entities as nouns and relationships as verbs.

ER diagrams are related to data structure diagrams (DSDs), which focus on the relationships of elements within entities instead of relationships between entities themselves. ER diagrams also are often used in conjunction with data flow diagrams (DFDs), which map out the flow of information for processes or systems.

## Uses of entity relationship diagrams

* **Database design:** ER diagrams are used to model and design relational databases, in terms of logic and business rules (in a logical data model) and in terms of the specific technology to be implemented (in a physical data model.) In software engineering, an ER diagram is often an initial step in determining requirements for an information systems project. It’s also later used to model a particular database or databases. A relational database has an equivalent relational table and can potentially be expressed that way as needed.
* **Database troubleshooting:**ER diagrams are used to analyze existing databases to find and resolve problems in logic or deployment. Drawing the diagram should reveal where it’s going wrong.
* **Business information systems:**The diagrams are used to design or analyze relational databases used in business processes. Any business process that uses fielded data involving entities, actions and interplay can potentially benefit from a relational database. It can streamline processes, uncover information more easily and improve results.
* **Business process re-engineering (BPR):**ER diagrams help in analyzing databases used in business process re-engineering and in modeling a new database setup.
* **Education:**Databases are today’s method of storing relational information for educational purposes and later retrieval, so ER Diagrams can be valuable in planning those data structures.
* **Research:** Since so much research focuses on structured data, ER diagrams can play a key role in setting up useful databases to analyze the data.

**The component’s and features of an ER Diagram**

ER Diagrams are composed of entities, relationships and attributes. They also depict cardinality, which defines relationships in terms of numbers. Here’s a glossary:

**Entity:**

### 

A definable thing—such as a person, object, concept or event—that can have data stored about it. Think of entities as nouns. Examples: a customer, student, car or product. Typically shown as a rectangle.

**Entity type:**

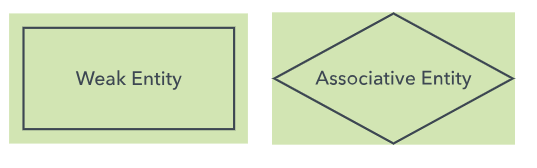
A group of definable things, such as students or athletes, whereas the entity would be the specific student or athlete. Other examples: customers, cars or products.

**Entity set:**

Same as an entity type, but defined at a particular point in time, such as students enrolled in a class on the first day. Other examples: Customers who purchased last month, cars currently registered in Florida. A related term is instance, in which the specific person or car would be an instance of the entity set.

**Entity categories:**

Entities are categorized as strong, weak or associative. A **strong entity** can be defined solely by its own attributes, while a **weak entity** cannot. An associative entity associates entities (or elements) within an entity set.



**Entity keys:** Refers to an attribute that uniquely defines an entity in an entity set. Entity keys can be super, candidate or primary. **Super key:**A set of attributes (one or more) that together define an entity in an entity set. **Candidate key:**A minimal super key, meaning it has the least possible number of attributes to still be a super key. An entity set may have more than one candidate key. **Primary key:**A candidate key chosen by the database designer to uniquely identify the entity set. **Foreign key:**Identifies the relationship between entities.

**Relationship**

How entities act upon each other or are associated with each other. Think of relationships as verbs. For example, the named student might register for a course. The two entities would be the student and the course, and the relationship depicted is the act of enrolling, connecting the two entities in that way. Relationships are typically shown as diamonds or labels directly on the connecting lines.

