**Software Design Document**

**1. INTRODUCTION**

**1.1 Purpose**

The purpose of this document is to present a detailed description of the product, online food order delivery.

**1.2 Scope**

The main objective of the Project on Online Food Ordering System is to managethe details of Food Item, Category, Customer, Order, Confirm Order. It manages all theinformation about Food Item, Payment, Confirm Order, Food Item. The project is totallybuilt at administrative end and thus only the administrator is guaranteed the access. Thepurpose of the project is to build an application program to reduce the manual work formanaging the Food Item, Category, Payment, Customer. It tracks all the details aboutthe Customer, Order, Confirm Order

**1.3 Overview**

.It generates the report on Food Item, Category, Payment

•Provide filter reports on Customer, Order, Confirm Order

•You can easily export PDF for the Food Item, Payment, Order

•Application also provides excel export for Category, Customer, Confirm Order

•You can also export the report into csv format for Food Item, Category, Confirm

Order

**1.4 Reference Material**

* SRS v2.0
* Feasibility report.

**1.5 Definitions and Acronyms**

|  |  |  |
| --- | --- | --- |
| 1 | Employer | Employer is an individual who has contacted the event organiser. |
| 2 | admin | admin is an individual who is responsible for the whole managing and can view the entirety of the event being planned on the software. |
| 3 | SQLite | SQLite is software library that implements a self-contained,serverless,zeroconfiguration, transactional SQL database engine. |
| 4 | Team Head | Team head is an individual who is responsible for all the actions undergoing under his/her team. |
| 5 | UI | UI stands for User Interface.It is defined as the space where interaction between humans and machines occurs. |
| 6 | customer | Customer can login and order delivery. |

**2. SYSTEM OVERVIEW**

The proposed system has the following requirements:

•System needs store information about new entry of Food Item.

•System needs to help the internal staff to keep information of Category and find

them as per various queries.

•System need to maintain quantity record.

•System need to keep the record of Customer.

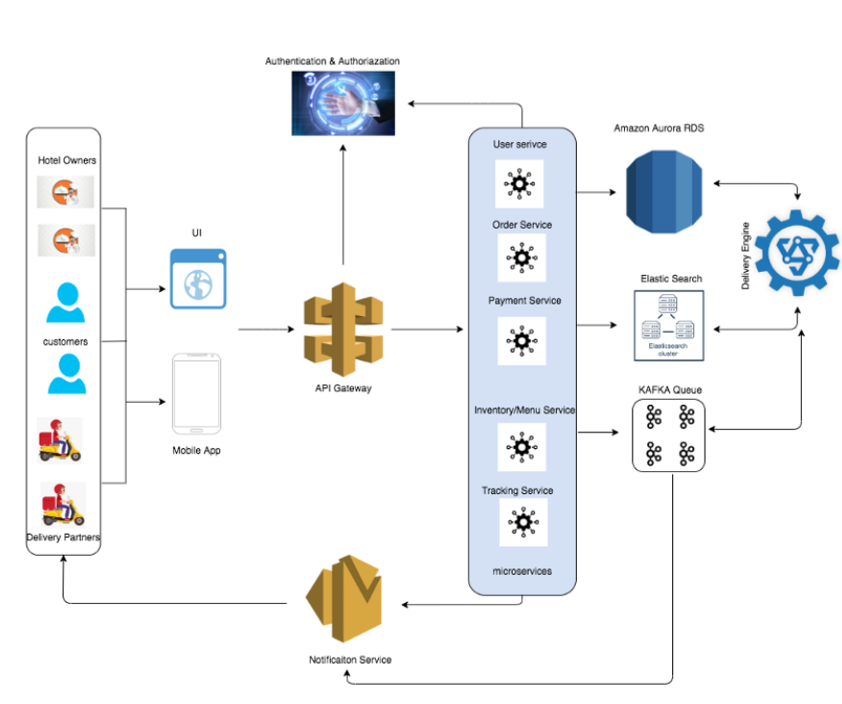
•System need to update and delete the record.

•System also needs a search area.

•It also needs a security system to prevent data.

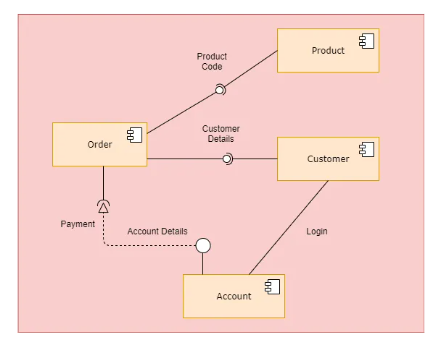
**3. SYSTEM ARCHITECTURE**

**3.1 Architectural Design**



**3.2 Decomposition Description**

Generalization hierarchy diagram for todo module



Activity diagram

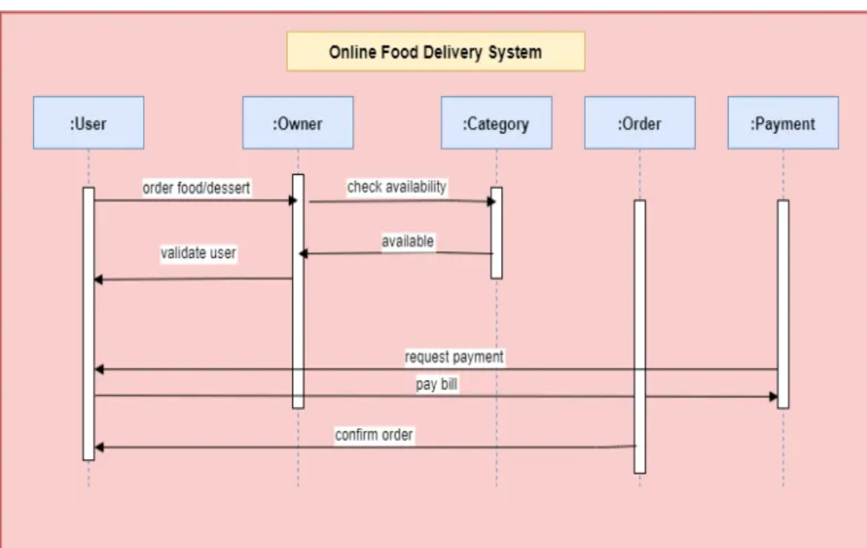


Fig 3.2(b) Sequence diagram

Diagram

Description automatically generated

Fig 3.2(c) Class diagram

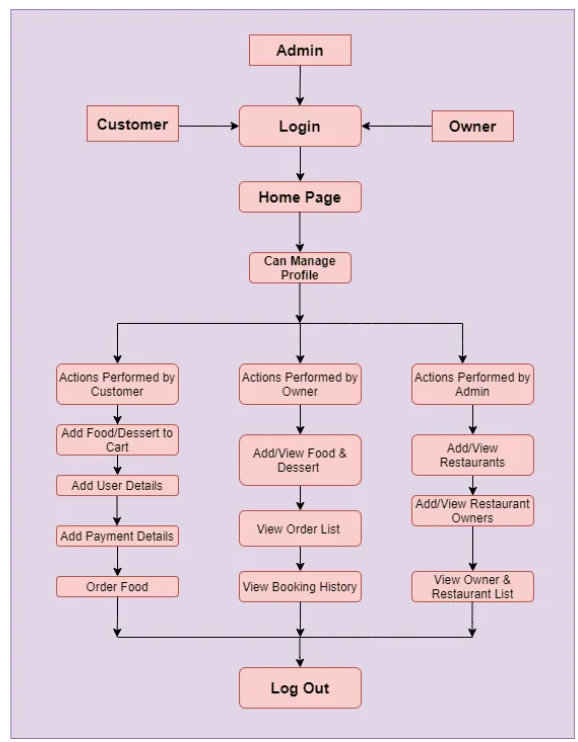


Fig 3.2(e) Data Flow Diagram

**4. DATA DESIGN**

**4.1 Data Description**

A Todo list has been designed to store all the tasks that are to be accomplished by a team and the current status of every task.This implies that a single todo form is associated with a single team.

Thus, the todo form is accessible only to authentic users who are a part of the particular event and the particular team.

A user can dynamically add tasks to the Todo list and can mark the task as done ,thus changing the status of a particular task.

It is necessary for the changes made to the todo form to be reflected on to a Database. A todo form is unique for each team of a specific event and any changes made to the todo form should sustain when the user opens the form again. The Database used in the project, Event4u is SQLite and this maintains the attributes for the Todo list, which are as shown in the table below.

|  |  |
| --- | --- |
| **Attribute name** | **Data Type and its description** |
| task\_id | auto-incremental primary key (identifies each task uniquely) |
| team\_id | Foreign key (refers to the id of the team to which the task belongs to) |
| task\_descr | TextField (gives a brief description about the task ) |
| Status | CharField (a character depicting if the task is done or not.) |

The order of the objects to be retrieved from the database is defined in the view. View is linked to the template which provides the GUI for the Todo list.

**4.2 Data Dictionary**

Todo is a class which stores the task’s description and its status.

In order for the Todo to maintain referential integrity we have a foreign key relationship with the team ID to which the Todo belongs to.

*class Todo( ):*

*"""Todo accomplished by Teams of Event4u"""*

*task\_id = AutoField(primary\_key=True)*

*team = ForeignKey(Team, default= -1 )*

*STATS = (*

*(u'D',u'Done'),*

*(u'I',u'In progress'),*

*)*

*status =CharField(max\_length=10,choices=STATS)*

*task\_descr =CharField(max\_length=70)*

*def save(team,task\_descr):*

*””” Save the todo details into database”””*

*self.team = team*

*self.task\_descr = task\_descr*

*def get(team\_id):*

*””” Get a specific todoobjec”””*

*if self.team\_id = team\_id :*

*return self*

*else*

*return NULL*

**5. COMPONENT DESIGN**

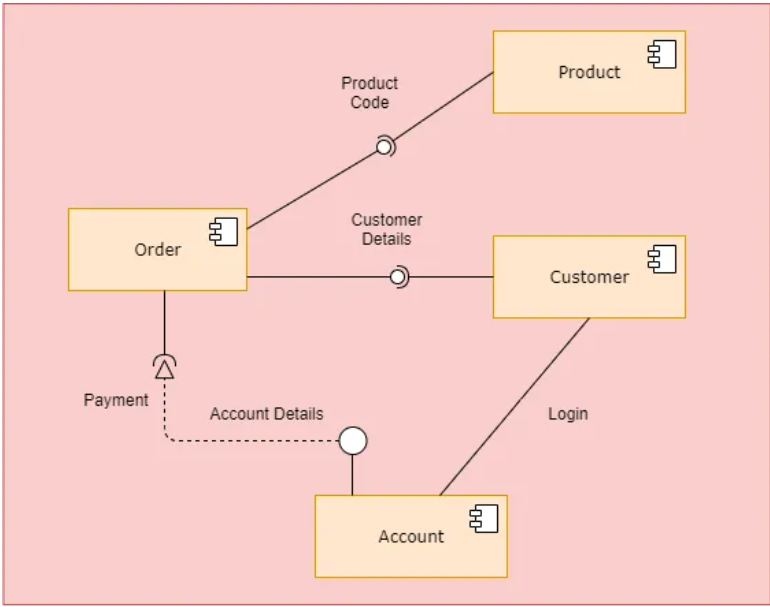


Fig 5(a) Component Design

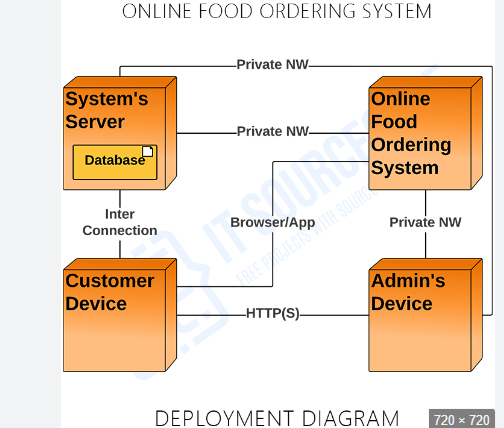


Fig 5(b) Deployment Diagram

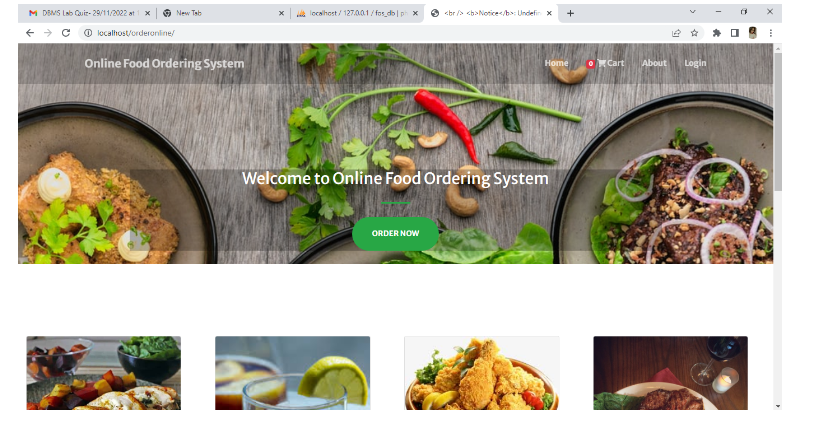
**6. HUMAN INTERFACE DESIGN**

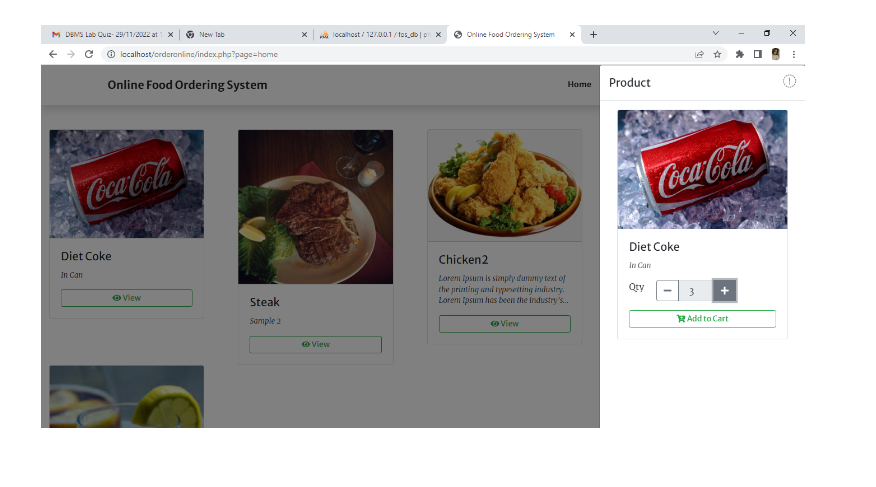
**6.1 Overview of User Interface**

User interface design is concerned with the dialogue between a user and the computer.It is concerned with everything from starting the system or logging into the system to the eventually presentation of desired inputs and outputs. The overall flow of screens and messages is called a dialogue.

**6.2 Screen Images**

l





**6.3 Screen Objects and Actions**

The user can view the tasks in a tabular form with a “Done” button placed next to each task.On clicking the “Done” button, the task is scratched out giving the client the impression of having completed a task.

**Graphical user interface, text, application, email

Description automatically generated**

**Graphical user interface

Description automatically generated with medium confidence**

A text box to dynamically add tasks is provided. Entering the task to be performed in the text box and clicking on “Add Tasks” button will add the task to the Todo list shown.

**7. REQUIREMENTS MATRIX**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Components  Requirement | Allauth. | DataBase | Users. | Task. | Add Task. | Mark  Task. |
| User interface. |  | 3.1.1 | 3.1.1 |  |  |  |
| Hardware interfaces. |  |  |  |  |  |  |
| Communications interfaces. | 3.1.4 |  |  | 3.1.4 | 3.1.4 | 3.1.4 |
| Event creation. | 3.2.1 | 3.2.1 |  |  |  |  |
| Task and team management | 3.2.2 | 3.2.2 | 3.2.2 | 3.2.2 | 3.2.2 | 3.2.2 |
| Budget. | 3.2.3 |  |  |  |  |  |
| Automatic notification. |  | 3.2.4 |  | 3.2.4 |  |  |
| Use case view. | 3.3.1 |  | 3.3.1 |  |  |  |