

Meals on Map (MoM)

Spring 2016

Guided by: SAMAHA MAMOUN

|  |  |
| --- | --- |
| PREPARED BY:  DIPTI BHOSALE (87114)  SRUTHI PUNYAMURTHULA (87872)  VIVEK VYAS (87923) |  |

**INTERNATIONAL TECHNOLOGICAL UNIVERSITY**

**Department of Software Engineering**

05/08/2016

I HEREBY RECOMMEND THAT THE CAPSTONE PROJECT PREPARED UNDER MY

SUPERVISION BY

**Dipti Bhosale, Sruthi Punyamurthula, Vivek Vyas**

ENTITLED

**Meals on Map project**

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF

**MASTER OF SCIENCE IN SOFTWARE ENGINEERING**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

CAPSTONE ADVISOR

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

DEPARTMENT CHAIR

**Meals on Map**

by

**Dipti Bhosale, Sruthi Punyamurthula, Vivek Vyas**

CAPSTONE PROJECT REPORT

Submitted in partial fulfillment of the requirements

for the degree of

Master of Science in Software and Engineering

International Technological University

San Jose, California

May 8, 2016

Table of Contents

[**Abstract** 6](#_Toc449540892)

[**Acknowledgement** 6](#_Toc449540893)

[**List of Figures** 6](#_Toc449540894)

[**List of Tables** 7](#_Toc449540895)

[**1.** **INTRODUCTION** 7](#_Toc449540896)

[**1.1** **Scope** 8](#_Toc449540897)

[**2.** **DOCUMENT HISTORY** 8](#_Toc449540898)

[**2.1** **Document Status** 8](#_Toc449540899)

[**2.2** **Revision History** 8](#_Toc449540900)

[**2.3** **Approvals** 9](#_Toc449540901)

[**2.4** **Distribution:** 9](#_Toc449540902)

[**3.** **PROJECT PLAN** 9](#_Toc449540903)

[**3.1** **Scrum Team** 10](#_Toc449540904)

[**3.2** **Sprint Schedule** 10](#_Toc449540905)

[**4.** **RISK MANAGEMENT** 12](#_Toc449540906)

[**4.1** **Risk Identification** 12](#_Toc449540907)

[**4.2** **Risk Analysis** 13](#_Toc449540908)

[**4.3** **Risk Response Planning** 13](#_Toc449540909)

[**4.4** **Risk Management** 13](#_Toc449540910)

[**5.** **REQUIREMENTS OVERVIEW** 15](#_Toc449540911)

[**5.1** **Functional Requirements** 15](#_Toc449540912)

[**5.2 Non-Functional Requirements:** 17](#_Toc449540913)

[**5.2.1 Responsiveness Requirements** 17](#_Toc449540914)

[**5.2.2 Performance Requirements** 17](#_Toc449540915)

[**5.2.3 Security Requirements** 18](#_Toc449540916)

[**5.3** **Terminology** 18](#_Toc449540917)

[**6. Functional View:** 19](#_Toc449540918)

[**7. Design Overview** 46](#_Toc449540919)

[**7.1. Design Rationale** 46](#_Toc449540920)

[**7.2. Software Architecture** 47](#_Toc449540921)

[**7.2.1 High-level Architecture:** 47](#_Toc449540922)

[**7.2.2 4-tier Architecture:** 48](#_Toc449540923)

[**7.2.3 MVC Architecture:** 49](#_Toc449540924)

[**REFERENCES:** 50](#_Toc449540925)

## **Abstract**

Who doesn’t like fresh and tasty homemade food? Everybody loves food cooked by their mom. We are bringing unique solution to the people who can’t have homemade food on a daily basis. This solution is also going to create opportunity for all the home based cooks. These cooks can upload their schedule and menu in our system and our consumers can locate the fresh, tasty, healthy homemade food on our map based food locator aka MOM. After all who doesn’t love homemade food cooked by their mom?

Below are the highlights of our project:

* Fresh and tasty homemade food available at the click of a button!
* Unique solution for people who can’t have homemade food on a daily basis.
* An application where Cooks and Customers are the only users.
* Cooks can display their menu and Customers can order.
* Search based on Time and Location.
* Opportunity of placing an order in advance.
* Picking up from your nearest location, your favorite dishes, far cheaper and safer than restaurants!

## **Acknowledgement**

We would like to express our gratitude to our esteemed Professor, **Samaha Mamoun,** Core Faculty, Computer Science, International Technological University, San Jose, CA for his guidance, ethnicity, valuable suggestions that helped us in project preparation. We thank him for giving us the opportunity to learn and explore Software Engineering techniques used in real world.

## **List of Figures**

|  |  |  |
| --- | --- | --- |
| Figure No. | Figure Name | Page No. |
| 7.2.1 | High-level architecture |  |
| 7.2.2 | 4-tier architecture |  |
| 7.2.3 | MVC architecture |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## **List of Tables**

|  |  |  |
| --- | --- | --- |
| Table No. | Table Name | Page No. |
| [Table 2.1.1](#_Document_Status) | Document Status | 7 |
| [Table 2.2.1](#_Revision_History) | Revision History | 7 |
| [Table 2.3.1](#_Approvals) | Approvals | 7 |
| Table 3.1.1 | Team Details |  |
| Table 3.2.1 | Sprint 1 details |  |
| Table 3.2.2 | Sprint 2 details |  |
| Table 3.2.3 | Sprint 3 details |  |
| Table 3.2.4 | Sprint 4 details |  |
| Table 3.2.5 | Sprint 5 details |  |
| Table 3.2.6 | Sprint 6 details |  |
|  |  |  |
| Table 4.3.1 | Terminology |  |

## **INTRODUCTION**

Healthy and nutritious food is a basic need for every one of us. What could be better than a fresh, tasty and healthy homemade meal? We are introducing our unique application “Meals on Map (MOM)” which will satisfy this very need of our people.

The idea behind our application is to make fresh homemade food available to people in a neighborhood which is cooked by the chefs in the same neighborhood. Here we are not only satisfying food needs of people but also creating business opportunity for home based cooks.

The chefs in the neighborhood will cook food at their convenience and will upload their menu and schedule in advance to our website. Our consumers will check our application for the food available in their neighborhood for a particular time. They can also place an order in advance based on the schedule uploaded by our chefs. In short, the consumer will be able to search the food available in his/her neighborhood by entering the time, location and date he wants to pick up the food. For example, John will search for the lunch for 02/02/2016 and application will pop up all the homemade meals available in his neighborhood on the map for that time including details of the dishes, their prices and ratings and reviews of the local chefs and their dishes.

For this application, we are planning to use Google Maps APIs to locate the consumer and the meals available around him. Our MOM (Meals On the Map) app will show him available meals around him on the Google map inside our application. The meal suggestions will change based on the time entered by the consumer and the availability of the meals in his neighborhood.

MOM will be beneficial in a great way to not only the consumers but also the local chefs in various ways. They can cook at their own pace and convenience and in the quantities they are comfortable with. They can also restrict the number of pre-orders based on their capacity and cook solely based on the preorders. They can easily sell their food without investing much in their business by using our platform. In summary, our application will enable the home based cooks to unlock their passion of cooking food and make a business out of it without big investments and at their own pace. Consumers will also be happy since they will get tasty, healthy, homemade food choices at their fingertips which will cost them cheaper than the restaurants.

**Note**: Currently we are building website for this project (and not application). So every time whenever we refer as application in this document, it actually means our website.

## **Scope**

Currently, MOM (Meals on Map) is a website that can

1. Help consumer to find homemade food in his locality by asking the Zipcode and Date, which will locate the provider and the meals available around him.
2. Buy the homemade, fresh, healthy food at comparatively low price.
3. Manage both consumer & providers profile
4. Maintains history of consumer’s order
5. Allows consumers to rate the provider
6. Allows consumer to place an order in advance

## **DOCUMENT HISTORY**

## **Document Status**

Table 2.1.1

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Status** | **Date** | **Authors** |
| 1.0.0 | Under review | 04/27/2016 | Dipti Bhosale, Sruthi Punyamurthula, Vivek Vyas |

## **Revision History**

Table 2.2.1

|  |  |  |
| --- | --- | --- |
| **Version** | **Author** | **Summary of Changes** |
| 0.0.1 | Dipti Bhosale | Initial template |
| 0.0.2 | Dipti Bhosale | Added Abstract, Introduction, Scope |
| 0.0.3 | Dipti Bhosale | Added Risk management |
| 0.0.4 | Dipti Bhosale | Added Risk Identification |
| 0.0.5 | Dipti Bhosale | Added Requirements Overview |
| 0.0.6 | Dipti Bhosale | Added Functional Requirements |
| 0.0.7 | Dipti Bhosale | Added Non Functional Requirements |
| 0.0.8 |  |  |
| 0.0.9 | Dipti Bhosale | Added software architecture content. |
| 0.1.0 | Dipti Bhosale | Added Design Rationale |
| 0.1.1 |  |  |
| 0.1.2 |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## **Approvals**

Table 2.3.1

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Role** | **Date** | **Version Approval** |
| Samaha Mamoun | Instructor | 04/30/2016 | 1.0.0 |

## **Distribution:**

The document is distributed among all the teammates for continuous review and updating.

# **PROJECT PLAN**

E-commerce business have an evolving life cycle and hence we need to have a proper traceability matrix built over the period of the software development, a robust project plan and an effective Software Development Life Cycle. Thinking about all these criteria and keeping the current industry growth, we decided to go with a combination of scrum and agile methodology. According to the plan, we decided to have a phone or web meeting bi-weekly to analyze our work and discuss our milestones achieved.

One of the main reason to choose agile software development for our project was time-restriction and velocity required for project completion. In agile, the requirements gathering, design, development and testing all go hand in hand, it would be easier for us to perform the job with this methodology. It helps in adaptive planning, evolutionary development, and early delivery and encourages rapid and flexible response to change.

Scrum is a part of agile but is more of iterative and incremental agile software development framework for managing the development. Since the team is very small, each of the individual will equally play a role of the scrum team member and carry it forward till the end.

## **Scrum Team**

Team consisted of three people only so we decided to start of the work with making a schedule and dividing each of the tasks linked with a deadline of submission. We decided to use Google Drive for the co-ordination and sharing of our work across the team. Below is the table describing our work and responsibilities. Documentation part was done by each of the member in their respective expertise.

Table 3.1.1

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | Role | Email |
| 1 | Dipti Bhosale | Developer | 87114.bhosale@students.itu.edu |
| 2 | Shruti Punyamurthula | Data Admin | 87872.punyamurthula@students.itu.edu |
| 3 | Vivek Vyas | Developer | 87923.vyas@students.itu.edu |

## **Sprint Schedule**

In Agile methodology, sprints are a way to finalize on the requirements, focus on design and development and track down the progress. Sprints are typically from one week up to four weeks. We started with a free sprint schedule where the deadlines were not decided at the initial level to a more robust sprint schedule where we started on deciding and finalizing the requirements and track down on the progress. The initial phase of finalizing the project plan and going forward itself took quiet a long time to settle down. So we started with a sprint schedule from 1st February. The schedule for it is as follows:

**Sprint 1:**

Table 3.2.1

|  |  |
| --- | --- |
| February 1, 2016 | Brainstorming |
| February 6, 2016 | Finalizing the project |
| February 12, 2016 | Requirements gathering and documentation |
| February 18, 2016 | Finalizing the requirements |
| February 21, 2016 | Writing specification |

**Sprint 2:**

Table 3.2.2

|  |  |
| --- | --- |
| February 24, 2016 | Building the backbone of the documentations |
| March 2, 2016 | Sprint Planning, Design Sessions |
| March 6, 2016 | Designing the document |
| March 9, 2016 | Dividing the responsibility |
| March 15, 2016 | Start with the documentation |

**Sprint 3:**

Table 3.2.3

|  |  |
| --- | --- |
| March 18, 2016 | Gathering the Functional Requirements |
| March 21, 2016 | Refining the requirements to suit business needs |
| March 23, 2016 | Non-Functional Requirements Gathering |
| March 26, 2016 | Use case diagrams discussion |
| March 28, 2016 | Bug Cycle discussion |

**Sprint 4:**

Table 3.2.4

|  |  |
| --- | --- |
| March 29, 2016 | Class Diagrams |
| March 30, 2016 | Sequence Diagrams |
| April 2, 2016 | State Diagrams, Wireframes |
| April 4, 2016 | Django Environment setup |
| April 6, 2016 | Entity Relationship Diagrams |

**Sprint 5:**

Table 3.2.5

|  |  |
| --- | --- |
| April 7, 2016 | Development Started |
| April 9, 2016 | Developed the Admin page |
| April 10, 2016 | Built Signup Admin page for users |
| April 12, 2016 | Discussion on the version control setup |
| April 15, 2016 | Pushing the admin apps to real environment |

**Sprint 6:**

Table 3.2.6

|  |  |
| --- | --- |
| April 17, 2016 |  |
| April 19, 2016 |  |
| April 21, 2016 |  |
| April 23, 2016 |  |
| April 24, 2016 |  |

# **RISK MANAGEMENT**

Risk management is about pro-active dealing with risks (potential problems) as opposed to reactive dealing with risks. Below is the structured approach to risk management:

## **Risk Identification**

Firstly, we have to identify the risks that can jeopardize our project. For example, Database may crash causing loss of important customer data. There are some specific tools and techniques for identifying risk as listed below:

1. Documentation Reviews
2. Information Gathering Techniques - Brainstorming, Delphi Technique, Interviewing, Root cause analysis
3. Checklist analysis - previous similar project, lowest level RBS
4. Assumption analysis
5. Diagramming Techniques - cause and effect diagram, system and process flow chart, influence diagrams
6. SWOT Analysis
7. Expert Judgment

Risk management tools such as EasyRisk Manager, Risk Register can be used to document, distribute, and analyze the risks that we input in these tools!! ‘Risk Event’ will occur causing ‘Risk Result’ which affects ‘Schedule, Resources, or Scope’.

## **Risk Analysis**

In Risk Analysis, we define “how bad it is” part or we say the impact. We have get into the details of what will happen to us when the risk happens to us. We need to do both qualitative and quantitative risk analysis. Figure out the Probability (on a scale of 0 to 1) of each risk happening and Impact i.e., damage caused also on a scale of 0 to 1 of the risk happening. Then, come up with a number called the risk factor whose formula is: (P+I) – (P\*I). The higher the risk factor, the more concerned we will be about it.

## **Risk Response Planning**

We have to plan a response (what we will do about this risk) for each risk identified in Step 1 i.e., Risk Analysis. Below are the 4 ways of dealing with the risk or we can 4 kinds of responses:

1. Preventative actions: Here we can take steps to completely prevent the risk from occurring.
2. Mitigative actions: Steps taken to reduce the effect of (negative effects) from a risk.
3. Transference actions: Steps taken to transfer the liability to someone else. For example: Insurance company
4. Acceptance/Contingency planning actions: Steps taken to deal with the risk when it actually happens.

## **Risk Management**

We should track, monitor, control and report the risk level throughout the project lifecycle. Also be aware of the risk that may be happening. The risk list as shown below will be maintained and reported as a component of the project status reporting process for this project.

To summarize the risk management plan, we can create a risk list/table as below:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Risk #** | **Risk Description** | **Likelihood**  **(L)** | **Impact**  **(I)** | **Risk Factor**  **(L+I)-(L\*I)** | **Risk Response Actions** | **Owner** | **Due Date** |
| 1 | Project Management: Personnel assigned to project do not have necessary skills to perform work, resulting in delays | 0.5 | 0.8 | 0.90 | 1. Send developers to Python training (Preventive)  2. Assign an in-house Python mentor to the project (Mitigation)  3. Increase schedule estimates to allow for skill level (Acceptance/Contingency) | Project Manager | March 10, 2016 |
| 2 | Customer is not acquainted with ordering online |  |  |  | Allow him/her to browse through the website for some time before actual ordering and also provide Customer support over phone. |  |  |
| 3 | Heavy flow of orders cannot be supported by providers. |  |  |  | Terms and conditions mention that the orders can be placed as long as the timings are maintained and providers are expected to update the application as and when the orders are sold-out. |  |  |
| 4 | Inspection of ordered meals is not possible. |  |  |  | Call to the providers is placed on request to talk and confirm about the cooking conditions and environment. |  |  |

There will be some unanticipated risks which each software will undergo and can only be known once the product goes live.

# **5.** **REQUIREMENTS OVERVIEW**

Business requirements are customer’s high level statements of needs and wants for a new or altered product. While gathering requirements we have to take into account of the conflicting requirements coming from various stakeholders, analyze them, validate and manage the requirements. To make a project successful, requirements gathering is a crucial step. There are various ways by which we can gather requirements and they may differ from one project to another. For this project, we have done the requirement gathering by below mentioned techniques:

1. Brainstorming
2. Document Analysis – Gap Analysis
3. Interview
4. Prototyping – Fast sketches
5. Observation
6. Survey

## **5.1** **Functional Requirements**

Business Architect (BA) converts the high level needs and wants of the end users (business requirements) into actionable items, which he gives to the developers. These requirements are called functional requirements. Functional requirements define what the system should do to fulfil the business requirements. They are the needs and wants of the programmers on what to do (how to code) to achieve a business need.

**FR1: Sign Up / Registration**

1.1: An anonymous user shall be able to Sign Up by simply entering his Name, Contact details, Email ID and Password

1.2: Once Signed Up, the user now becomes a Registered user and shall be able to Sign In with his registered Email ID and Password

1.3: The website shall display “Become a Provider” or,” Become a Customer” option for registered user, which he can choose and add his Bank details to become a provider

**FR2: Users**

2.1: The website shall allow an anonymous user to browse through the homepage and see various meals, their prices, ingredients, availability, picture of the meal & providers, and also ratings (ratings & feedback to be added in later implementations) given to the providers

2.2: The website shall allow the registered user to browse through the homepage and see various meals, their prices, ingredients, availability, picture of the meal & providers, and buy the meal (with valid credit card details). He shall also be able to view ‘Order History’ (history and current orders).

2.3: The website shall allow the provider to see his posted menu with respective details.

2.4: The website shall allow the provider to update the number of meals available in real time

**FR3: Filters or Search**

3.1: The website shall allow the consumers to search the orders based on the Date and Zipcode they enter so as to get the list of all possible options around him/her.

3.2: The website shall allow the consumer to put filters for veg, non-veg, vegan menu or to find a particular dish so that he can see only required menu

3.3: The website shall allow the consumer to filter the “available” and/or “sold out” meals

**FR4: Buy a Meal**

4.1: The website shall allow registered user (consumer) to add his credit card details in order to purchase a meal

4.2: The website shall not allow an unregistered user or a registered user with no/invalid card details to purchase a meal

4.3: The website shall allow the provider to add upcoming meal details like dish name, price, quantity available, ingredients, its picture and food type (veg, non-veg, vegan) on the website at-least 5 days prior to the pickup day so that it will be easy for customers to choose and order according to their needs

4.4: The website shall allow the consumers to pre-order the meals so that the provider can plan his meal according to the pre-orders received

**FR5: Feedback**

5.1: The website shall allow the consumers to rate and give review to a particular meal

5.2: The provider shall be able to receive and view the consumer’s feedback and ratings on an individual meal

**FR6: Offers/Deals**

6.1: Shall be able to add different kinds of offers or deals to the customer as a marketing policy.

## **5.2 Non-Functional Requirements:**

Non-functional requirements define how the software application/system will do the functional tasks. They are the kind of requirements that specifies the criteria that can be used to judge the operation of a system, rather than specific behaviors.

### **5.2.1 Responsiveness Requirements**

**NFR1: Calling External Services (API calls)**

1.1: When user selects any Social media (like Facebook, Twitter) for sign-in or sign-up, system needs to send user data and in response authenticate the user, this whole process of authentication shall happen with a latency of no greater than 1 minute.

**NFR2: Location Based Meals**

2.1: System shall display the meals based on the Zipcode and Date entered by the consumer with a latency of no greater than 30 seconds.

**NFR3: Data Filtration or Search**

3.1: System needs to filter meal search with a latency of no greater than 20 seconds.

3.2: System needs to filter the ‘available’ and ‘sold out’ meal with a latency of no greater than 20 seconds.

**NFR4: Sending Confirmation Receipt by Email**

4.1: If the customer buys a meal, system should send the confirmation receipt to opted Email ID with a latency of no greater than 3 minute.

**NFR5: Payment Gateway Services:**

5.1: Once the consumer confirms the payment method and proceeds to payment, system shall respond from payment gateway services with latency no greater than 30 seconds.

### **5.2.2 Performance Requirements**

Performance requirements define how well the system performs certain functions under specific conditions. Below are the performance requirements for our application.

1.1: Every page of the application (frontend) interacting with the customer or provider should respond in less than 8 seconds.

1.2: System should support ~100,000 users in total.

1.3: System should respond to ~10 parallel request in a second.

### **5.2.3 Security Requirements**

Security requirements relate to system confidentiality, integrity and availability.

1.1: All audit logs shall be verbose enough to support forensics.

1.2: System should mandate the sign-in process for every consumer who wants to buy a meal.

1.3: If the user enters a wrong password 3 times in a row, the account shall be locked out. The user can only unlock the account with the correct password after 15 minutes.

1.4: System should perform role based access check before permitting the users to do tasks that are allowed for them.

1.5: System should not expose passwords, and other sensitive payment details in log messages, error messages, session cookies etc.

### **5.3** **Terminology**

Table 5.3.1

|  |  |
| --- | --- |
| **Term** | **Description** |
| User | User is a generic term for consumer as well as provider |
| Consumer | He is someone who is looking for homemade meals available via our application |
| Provider | They are homemade based cooks/chefs who are willing to provide the fresh homemade food to the people in their neighborhood. |

# **6. Functional View:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Use Case Id** | UC-01   |  |  | | --- | --- | | **Use Case Id** | UC-01 | | **Use Case Name** | Login | | **Actors** | Customer, Provider, Admin | | **Description** | This use case describes the interaction between the user and our MOM website.  The user has a valid username and password and logs in as one of the three roles in the system:   * A Customer * A Provider * An Admin. | | **Trigger** | The user wishes to   * Place/Change/Cancel an order for a meal (If a Customer) * Provide/Change/Cancel a menu for a meal (If a Provider) * Verify some information (If an Admin) | | **Pre-Condition** | The user shall enter his/her login details and also check the box provided for the role in the system. | | **Post-Condition** | The login is successful – the system should be able to provide access for the functionality depending on the roles. | | **Main Flow** | * The User enters his username and password to the system. * The User also checks the appropriate checkbox for his/her role in the system. * The system checks these with the database and proceeds to authorize. * If the authorization is successful, the user is logged in and provided access depending on his/her role in the system. * If the authorization is not successful, the user is not logged in and an appropriate error message is displayed. |  |  |  | | --- | --- | | **Use Case Id** | UC-02 | | **Use Case Name** | Logout | | **Actors** | Customer, Provider, Admin | | **Description** | This use case describes the interaction between the user and our MOM website.  The user clicks the logout button provided. | | **Trigger** | The user wishes to logout of MOM. | | **Pre-Condition** | The user shall click the logout button provided. | | **Post-Condition** | The logout is successful. | | **Main Flow** | * The User after doing what he intended to do in MOM, wishes to logout. * The user clicks the logout button present in the screen. * The database is checked if the user is already logged in. * If the User is logged in, then the appropriate entry is made in the database and the user is logged out of MOM. * If the User is not logged in, i.e. the database shows that the user is already logged out, then an appropriate error message is displayed, and user is prompted to the login screen again. |  |  |  | | --- | --- | | **Use Case Id** | UC-03 | | **Use Case Name** | View Homepage | | **Actors** | Customer, Provider, Admin, Un-registered User | | **Description** | This use case describes the interaction between the user and our MOM website.  The user wishes to know what the website is about and we provide the concept of MOM in the homepage, without the need to login to our system. | | **Trigger** | The user wishes to know about MOM. | | **Pre-Condition** | The user shall type the web address in a web browser. | | **Post-Condition** | The user shall be able to see the homepage where the concept of MOM is explained and can   * Sign-up to register into the system. * Login – if already registered. | | **Main Flow** | * The User types the web address of MOM in any web browser. * The user then sees the homepage with the concept of MOM explained. * The user shall also be able to see the upcoming meals, the area covered and also the sign up and login links available. * If the User wishes to sign up, he/she clicks the “Sign-up” link and is taken to the appropriate sign-up page. * If the User is already registered, he/she can click the “Login” link and proceed to login with their credentials. |  |  |  | | --- | --- | | **Use Case Id** | UC-04 | | **Use Case Name** | Sign-Up | | **Actors** | Un-registered User | | **Description** | This use case describes the interaction between the user and our MOM website.  The user wishes to sign-up into the MOM website to avail its services. | | **Trigger** | The user wishes to become   * A customer * A provider   in our MOM system. | | **Pre-Condition** | The user shall click the Sign-Up button on the homepage. | | **Post-Condition** | The user shall be able to see the sign-up page asking the following details:   * First Name * Last Name * E-Mail ID * Password * Confirm Password   An option to specify his/her role in the system   * If Customer   + Credit Card details * If Provider   + Bank Details   And once these are provided, the user is registered with MOM. | | **Main Flow** | * The User clicks the Sign-Up button on the homepage. * The user fills all the required fields and clicks on Submit. * The user is registered with the system and an entry is made in the database with the user’s E-Mail id and Password that he entered. * The role of the user is captured in the system at this time. |  |  |  | | --- | --- | | **Use Case Id** | UC-05 | | **Use Case Name** | Search | | **Actors** | Customer, Provider | | **Description** | This use case describes the interaction between the user and our MOM website.  The user wishes to search the MOM website for the areas where the service is available and the timings. | | **Trigger** | The user wishes to search the MOM based on   * Time * Location | | **Pre-Condition** | The user shall click search tab/button provided. | | **Post-Condition** | The user shall be able to see the details based on his/her selection of Time based or Location based search. | | **Main Flow** | * The User clicks the Search tab/button provided after logging in. * The user checks the option button provided – Time Based or Location Based. * When searched based on Time, the details should be displayed showing the next available meal prioritized by time, the earliest first. * When searched based on location – by clicking on the maps API provided, the details should be displayed showing the next available meal prioritized by distance, nearest first. | | **Use Case Id** | UC-06 | | **Use Case Name** | Place Order | | **Actors** | Customer, Provider | | **Description** | This use case describes the interaction between the user and our MOM website.  The user wishes to place an order for food. | | **Trigger** | The user shall click “order now” button provided. | | **Pre-Condition** | The user should have been registered in the system. | | **Post-Condition** | The user shall be able to place an order and an email confirmation sent to his registered mail ID. | | **Main Flow** | * The user provides all his preferences. * The user selects from the list of options available for him/her. * The user then selects the order of his preference. * The User clicks the “Order now” button provided. * If the user is not registered, he/she is redirected to the registration page.   + - * + Once the user completes the registration process, he/she is taken back to the ordering page. * The user receives an e-mail confirmation once the payment successful. |  |  |  | | --- | --- | | **Use Case Id** | UC-07 | | **Use Case Name** | Change Order | | **Actors** | Customer, Provider | | **Description** | This use case describes the interaction between the user and our MOM website.  The user wishes to change an order that is already placed. | | **Trigger** | The user shall click “change order” button provided, strictly not less than 3 days before the date of order. | | **Pre-Condition** | The user should have already placed an order and the order should appear in his/her order history. | | **Post-Condition** | The user shall be able to change an order and an email confirmation sent to his registered mail ID. | | **Main Flow** | * The user selects the order he/she wishes to change. * The user selects his/her preference from the new list of options available. * The User clicks the “Confirm” button provided. * The user receives an e-mail confirmation once the order is changed successfully. |  |  |  | | --- | --- | | **Use Case Id** | UC-08 | | **Use Case Name** | Cancel Order | | **Actors** | Customer, Provider | | **Description** | This use case describes the interaction between the user and our MOM website.  The user wishes to cancel an order that is already placed. | | **Trigger** | The user shall click “cancel order” button provided, strictly not less than 3 days before the date of order. | | **Pre-Condition** | The user should have already placed an order and the order should appear in his/her order history. | | **Post-Condition** | The user shall be able to cancel an order and an email confirmation sent to his registered mail ID. The payment is returned to the customer’s card in the next 3 to 4 business days. | | **Main Flow** | * The user selects the order he/she wishes to cancel. * The User clicks the “Confirm” button provided. * The date is checked, so that he doesn’t have to pay a fine. * If the date is less than three days before the actual order date, then he/she will have to pay a fine amount and if it is the exact day of the order, he/she ends up paying a fine of half the amount. * The user receives a confirmation e-mail once the order is cancelled. * If the cancellation is done 3 days prior to the day of order, the entire amount is returned to the customer’s card in the next 3 to 4 business days. |  |  |  | | --- | --- | | **Use Case Id** | UC-10 | | **Use Case Name** | Upload a meal | | **Actors** | Provider | | **Description** | This use case describes the interaction between the provider and our MOM website.  The provider wishes to upload a meal/meals in the system. | | **Trigger** | The provider clicks the “Upload a meal/meals” button provided after login to the provider account. | | **Pre-Condition** | The user logs into MOM with the provider role. | | **Post-Condition** | The provider shall be able to upload a meal for a particular day or meals for an entire week. | | **Main Flow** | * The user logs in to the system with the role of a “Provider”. * The MOM takes us to the provider homepage showing the list of meals he/she has already provided. * The provider can then upload the new menu onto the system where ingredients and quantity(headcount) and price will be a mandatory fields to fill. | | **Use Case Id** | UC-09 | | **Use Case Name** | Provide Feedback | | **Actors** | Customer | | **Description** | This use case describes the interaction between the user and our MOM website.  The user wishes to provide feedback for the order he has placed. | | **Trigger** | The user wishes to provide feedback for an order that he/she has placed by clicking on the stars or by entering his/her remarks in the box provided and clicking on “SAVE/ POST REMARKS”. | | **Pre-Condition** | The user should have been registered in the system, should have placed at least one order with the provider and the order should appear in his/her order history. | | **Post-Condition** | The user shall be able to provide feedback. | | **Main Flow** | * The user selects the order he/she wishes to provide feedback for. * The User clicks the stars from the five stars given/ gives rating for the his/her order. * The user can also fill the feedback box with any kind of feedback he would like to provide about the cook, the dish or the service. * The user clicks on the “SAVE/POST REMARKS” button provided for them to appear on the website for the particular provider. | | **Use Case Id** | UC-11 | | **Use Case Name** | Change a meal | | **Actors** | Provider | | **Description** | This use case describes the interaction between the provider and our MOM website.  The provider wishes to change a meal that has already been entered into the system. | | **Trigger** | The provider clicks the “Change the menu” button provided after logging in to the system. | | **Pre-Condition** | The user logs into MOM with the provider role. | | **Post-Condition** | The provider shall be able to change a meal for a particular day or meals for an entire week. | | **Main Flow** | * The user logs in to the system with the role of a “Provider”. * The MOM takes us to the provider homepage showing the list of meals he/she has already provided. * The provider can then change the menu and upload onto the system where ingredients and quantity(headcount) and price will be a mandatory fields to fill. * An e-mail notification will be sent to all the customers who ordered this meal, mentioning the change. * Customers who do not like the change, need to cancel this order and place another order according to their choice. * As this is a provider change, when a customer cancels the order, he will get a full refund back irrespective of number of days before the order. |  |  |  | | --- | --- | | **Use Case Id** | UC-12 | | **Use Case Name** | Cancel a meal | | **Actors** | Provider | | **Description** | This use case describes the interaction between the provider and our MOM website.  The provider wishes to cancel a meal that has already been entered into the system. | | **Trigger** | The provider clicks the “Cancel the menu” button provided after logging in to the system. | | **Pre-Condition** | The user logs into MOM with the provider role. | | **Post-Condition** | The provider shall be able to cancel a meal for a particular day or meals for an entire week, provided it is less than 3 days of the day of meal. | | **Main Flow** | * The user logs in to the system with the role of a “Provider”. * The MOM takes us to the provider homepage showing the list of meals he/she has already provided. * The provider can cancel the menu and will have to pay a fine amount if the cancellation is done less than 3 days of the meal date. * An e-mail notification will be sent to all the customers who ordered this meal and the amount credited back to their credit card within 3 to 4 business days. | | **Use Case Id** | UC-13 | | **Use Case Name** | View History | | **Actors** | Customer, Provider, Admin | | **Description** | This use case describes the interaction between the User’s and our MOM website. The User wishes to view his/her history of orders in MOM. | | **Trigger** | The user clicks the “View History” button provided after logging in to the system. | | **Pre-Condition** | The user logs into MOM with the appropriate role. | | **Post-Condition** | The user shall be able to view the order history. | | **Main Flow** | * The user logs in to the system with the appropriate role. * The MOM takes us to the provider their respective homepages. * The user can click on the “View History” button provided and can see the list of all his/her orders. * If the user is an Admin of the system, he still has the rights to view the order history of customers using the unique number generated and maintained at the backend and known only to the admin in case he needs information to resolve any issue. |  |  |  | | --- | --- | | **Use Case Id** | UC-14 | | **Use Case Name** | Checkout | | **Actors** | Customer | | **Description** | This use case describes the interaction between the MOM website and payment module. The payment module is out of scope of our system. Our MOM system captures only the customer’s Credit card details and provider’s bank details and shares it with the payment module. | | **Trigger** | The user clicks the “Checkout” button after placing an order in MOM. | | **Pre-Condition** | The user places an order and is ready to checkout. | | **Post-Condition** | The user shall be able to pay for his/her order successfully and then an e-mail confirmation is sent. | | **Main Flow** | * The user logs in to the system with the role of a “Customer”. * The MOM takes us to the provider customer home page. * The customer places his/her order and finally clicks the “Checkout” button provided. * If it is a registered customer, the site is redirected to the payment module, as the CC details are already available. * If unregistered, the site is redirected to the registration page and then goes to ask CC details which are then proceeded to the checkout. * Once the payment goes through successfully, confirmation mails are sent to the customers that their orders have been placed successfully. | |
| **Use Case Name** | Login |
| **Actors** | Customer, Provider, Admin |
| **Description** | This use case describes the interaction between the user and our MOM website.  The user has a valid username and password and logs in as one of the three roles in the system:   * A Customer * A Provider * An Admin. |
| **Trigger** | The user wishes to   * Place/Change/Cancel an order for a meal (If a Customer) * Provide/Change/Cancel a menu for a meal (If a Provider) * Verify some information (If an Admin) |
| **Pre-Condition** | The user shall enter his/her login details and also check the box provided for the role in the system. |
| **Post-Condition** | The login is successful – the system should be able to provide access for the functionality depending on the roles. |
| **Main Flow** | * The User enters his username and password to the system. * The User also checks the appropriate checkbox for his/her role in the system. * The system checks these with the database and proceeds to authorize. * If the authorization is successful, the user is logged in and provided access depending on his/her role in the system. * If the authorization is not successful, the user is not logged in and an appropriate error message is displayed. |

|  |  |
| --- | --- |
| **Use Case Id** | UC-02 |
| **Use Case Name** | Logout |
| **Actors** | Customer, Provider, Admin |
| **Description** | This use case describes the interaction between the user and our MOM website.  The user clicks the logout button provided. |
| **Trigger** | The user wishes to logout of MOM. |
| **Pre-Condition** | The user shall click the logout button provided. |
| **Post-Condition** | The logout is successful. |
| **Main Flow** | * The User after doing what he intended to do in MOM, wishes to logout. * The user clicks the logout button present in the screen. * The database is checked if the user is already logged in. * If the User is logged in, then the appropriate entry is made in the database and the user is logged out of MOM. * If the User is not logged in, i.e. the database shows that the user is already logged out, then an appropriate error message is displayed, and user is prompted to the login screen again. |

|  |  |
| --- | --- |
| **Use Case Id** | UC-03 |
| **Use Case Name** | View Homepage |
| **Actors** | Customer, Provider, Admin, Un-registered User |
| **Description** | This use case describes the interaction between the user and our MOM website.  The user wishes to know what the website is about and we provide the concept of MOM in the homepage, without the need to login to our system. |
| **Trigger** | The user wishes to know about MOM. |
| **Pre-Condition** | The user shall type the web address in a web browser. |
| **Post-Condition** | The user shall be able to see the homepage where the concept of MOM is explained and can   * Sign-up to register into the system. * Login – if already registered. |
| **Main Flow** | * The User types the web address of MOM in any web browser. * The user then sees the homepage with the concept of MOM explained. * The user shall also be able to see the upcoming meals, the area covered and also the sign up and login links available. * If the User wishes to sign up, he/she clicks the “Sign-up” link and is taken to the appropriate sign-up page. * If the User is already registered, he/she can click the “Login” link and proceed to login with their credentials. |

|  |  |
| --- | --- |
| **Use Case Id** | UC-04 |
| **Use Case Name** | Sign-Up |
| **Actors** | Un-registered User |
| **Description** | This use case describes the interaction between the user and our MOM website.  The user wishes to sign-up into the MOM website to avail its services. |
| **Trigger** | The user wishes to become   * A customer * A provider   in our MOM system. |
| **Pre-Condition** | The user shall click the Sign-Up button on the homepage. |
| **Post-Condition** | The user shall be able to see the sign-up page asking the following details:   * First Name * Last Name * E-Mail ID * Password * Confirm Password   An option to specify his/her role in the system   * If Customer   + Credit Card details * If Provider   + Bank Details   And once these are provided, the user is registered with MOM. |
| **Main Flow** | * The User clicks the Sign-Up button on the homepage. * The user fills all the required fields and clicks on Submit. * The user is registered with the system and an entry is made in the database with the user’s E-Mail id and Password that he entered. * The role of the user is captured in the system at this time. |

|  |  |
| --- | --- |
| **Use Case Id** | UC-05 |
| **Use Case Name** | Search |
| **Actors** | Customer, Provider |
| **Description** | This use case describes the interaction between the user and our MOM website.  The user wishes to search the MOM website for the areas where the service is available and the timings. |
| **Trigger** | The user wishes to search the MOM based on   * Time * Location |
| **Pre-Condition** | The user shall click search tab/button provided. |
| **Post-Condition** | The user shall be able to see the details based on his/her selection of Time based or Location based search. |
| **Main Flow** | * The User clicks the Search tab/button provided after logging in. * The user checks the option button provided – Time Based or Location Based. * When searched based on Time, the details should be displayed showing the next available meal prioritized by time, the earliest first. * When searched based on location – by clicking on the maps API provided, the details should be displayed showing the next available meal prioritized by distance, nearest first. |
| **Use Case Id** | UC-06 |
| **Use Case Name** | Place Order |
| **Actors** | Customer, Provider |
| **Description** | This use case describes the interaction between the user and our MOM website.  The user wishes to place an order for food. |
| **Trigger** | The user shall click “order now” button provided. |
| **Pre-Condition** | The user should have been registered in the system. |
| **Post-Condition** | The user shall be able to place an order and an email confirmation sent to his registered mail ID. |
| **Main Flow** | * The user provides all his preferences. * The user selects from the list of options available for him/her. * The user then selects the order of his preference. * The User clicks the “Order now” button provided. * If the user is not registered, he/she is redirected to the registration page.   + - * + Once the user completes the registration process, he/she is taken back to the ordering page. * The user receives an e-mail confirmation once the payment successful. |

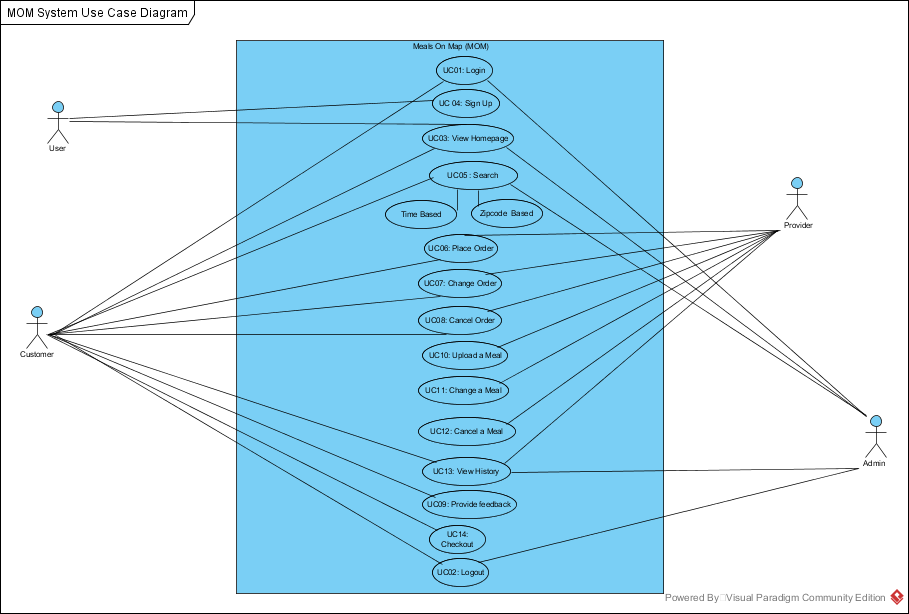
|  |  |
| --- | --- |
| **Use Case Id** | UC-07 |
| **Use Case Name** | Change Order |
| **Actors** | Customer, Provider |
| **Description** | This use case describes the interaction between the user and our MOM website.  The user wishes to change an order that is already placed. |
| **Trigger** | The user shall click “change order” button provided, strictly not less than 3 days before the date of order. |
| **Pre-Condition** | The user should have already placed an order and the order should appear in his/her order history. |
| **Post-Condition** | The user shall be able to change an order and an email confirmation sent to his registered mail ID. |
| **Main Flow** | * The user selects the order he/she wishes to change. * The user selects his/her preference from the new list of options available. * The User clicks the “Confirm” button provided. * The user receives an e-mail confirmation once the order is changed successfully. |

|  |  |
| --- | --- |
| **Use Case Id** | UC-08 |
| **Use Case Name** | Cancel Order |
| **Actors** | Customer, Provider |
| **Description** | This use case describes the interaction between the user and our MOM website.  The user wishes to cancel an order that is already placed. |
| **Trigger** | The user shall click “cancel order” button provided, strictly not less than 3 days before the date of order. |
| **Pre-Condition** | The user should have already placed an order and the order should appear in his/her order history. |
| **Post-Condition** | The user shall be able to cancel an order and an email confirmation sent to his registered mail ID. The payment is returned to the customer’s card in the next 3 to 4 business days. |
| **Main Flow** | * The user selects the order he/she wishes to cancel. * The User clicks the “Confirm” button provided. * The date is checked, so that he doesn’t have to pay a fine. * If the date is less than three days before the actual order date, then he/she will have to pay a fine amount and if it is the exact day of the order, he/she ends up paying a fine of half the amount. * The user receives a confirmation e-mail once the order is cancelled. * If the cancellation is done 3 days prior to the day of order, the entire amount is returned to the customer’s card in the next 3 to 4 business days. |

|  |  |
| --- | --- |
| **Use Case Id** | UC-10 |
| **Use Case Name** | Upload a meal |
| **Actors** | Provider |
| **Description** | This use case describes the interaction between the provider and our MOM website.  The provider wishes to upload a meal/meals in the system. |
| **Trigger** | The provider clicks the “Upload a meal/meals” button provided after login to the provider account. |
| **Pre-Condition** | The user logs into MOM with the provider role. |
| **Post-Condition** | The provider shall be able to upload a meal for a particular day or meals for an entire week. |
| **Main Flow** | * The user logs in to the system with the role of a “Provider”. * The MOM takes us to the provider homepage showing the list of meals he/she has already provided. * The provider can then upload the new menu onto the system where ingredients and quantity(headcount) and price will be a mandatory fields to fill. |
| **Use Case Id** | UC-09 |
| **Use Case Name** | Provide Feedback |
| **Actors** | Customer |
| **Description** | This use case describes the interaction between the user and our MOM website.  The user wishes to provide feedback for the order he has placed. |
| **Trigger** | The user wishes to provide feedback for an order that he/she has placed by clicking on the stars or by entering his/her remarks in the box provided and clicking on “SAVE/ POST REMARKS”. |
| **Pre-Condition** | The user should have been registered in the system, should have placed at least one order with the provider and the order should appear in his/her order history. |
| **Post-Condition** | The user shall be able to provide feedback. |
| **Main Flow** | * The user selects the order he/she wishes to provide feedback for. * The User clicks the stars from the five stars given/ gives rating for the his/her order. * The user can also fill the feedback box with any kind of feedback he would like to provide about the cook, the dish or the service. * The user clicks on the “SAVE/POST REMARKS” button provided for them to appear on the website for the particular provider. |
| **Use Case Id** | UC-11 |
| **Use Case Name** | Change a meal |
| **Actors** | Provider |
| **Description** | This use case describes the interaction between the provider and our MOM website.  The provider wishes to change a meal that has already been entered into the system. |
| **Trigger** | The provider clicks the “Change the menu” button provided after logging in to the system. |
| **Pre-Condition** | The user logs into MOM with the provider role. |
| **Post-Condition** | The provider shall be able to change a meal for a particular day or meals for an entire week. |
| **Main Flow** | * The user logs in to the system with the role of a “Provider”. * The MOM takes us to the provider homepage showing the list of meals he/she has already provided. * The provider can then change the menu and upload onto the system where ingredients and quantity(headcount) and price will be a mandatory fields to fill. * An e-mail notification will be sent to all the customers who ordered this meal, mentioning the change. * Customers who do not like the change, need to cancel this order and place another order according to their choice. * As this is a provider change, when a customer cancels the order, he will get a full refund back irrespective of number of days before the order. |
| **Use Case Id** | UC-12 |
| **Use Case Name** | Cancel a meal |
| **Actors** | Provider |
| **Description** | This use case describes the interaction between the provider and our MOM website.  The provider wishes to cancel a meal that has already been entered into the system. |
| **Trigger** | The provider clicks the “Cancel the menu” button provided after logging in to the system. |
| **Pre-Condition** | The user logs into MOM with the provider role. |
| **Post-Condition** | The provider shall be able to cancel a meal for a particular day or meals for an entire week, provided it is less than 3 days of the day of meal. |
| **Main Flow** | * The user logs in to the system with the role of a “Provider”. * The MOM takes us to the provider homepage showing the list of meals he/she has already provided. * The provider can cancel the menu and will have to pay a fine amount if the cancellation is done less than 3 days of the meal date. * An e-mail notification will be sent to all the customers who ordered this meal and the amount credited back to their credit card within 3 to 4 business days. |
| **Use Case Id** | UC-13 |
| **Use Case Name** | View History |
| **Actors** | Customer, Provider, Admin |
| **Description** | This use case describes the interaction between the User’s and our MOM website. The User wishes to view his/her history of orders in MOM. |
| **Trigger** | The user clicks the “View History” button provided after logging in to the system. |
| **Pre-Condition** | The user logs into MOM with the appropriate role. |
| **Post-Condition** | The user shall be able to view the order history. |
| **Main Flow** | * The user logs in to the system with the appropriate role. * The MOM takes us to the provider their respective homepages. * The user can click on the “View History” button provided and can see the list of all his/her orders. * If the user is an Admin of the system, he still has the rights to view the order history of customers using the unique number generated and maintained at the backend and known only to the admin in case he needs information to resolve any issue. |

|  |  |
| --- | --- |
| **Use Case Id** | UC-14 |
| **Use Case Name** | Checkout |
| **Actors** | Customer |
| **Description** | This use case describes the interaction between the MOM website and payment module. The payment module is out of scope of our system. Our MOM system captures only the customer’s Credit card details and provider’s bank details and shares it with the payment module. |
| **Trigger** | The user clicks the “Checkout” button after placing an order in MOM. |
| **Pre-Condition** | The user places an order and is ready to checkout. |
| **Post-Condition** | The user shall be able to pay for his/her order successfully and then an e-mail confirmation is sent. |
| **Main Flow** | * The user logs in to the system with the role of a “Customer”. * The MOM takes us to the provider customer home page. * The customer places his/her order and finally clicks the “Checkout” button provided. * If it is a registered customer, the site is redirected to the payment module, as the CC details are already available. * If unregistered, the site is redirected to the registration page and then goes to ask CC details which are then proceeded to the checkout. * Once the payment goes through successfully, confirmation mails are sent to the customers that their orders have been placed successfully. |

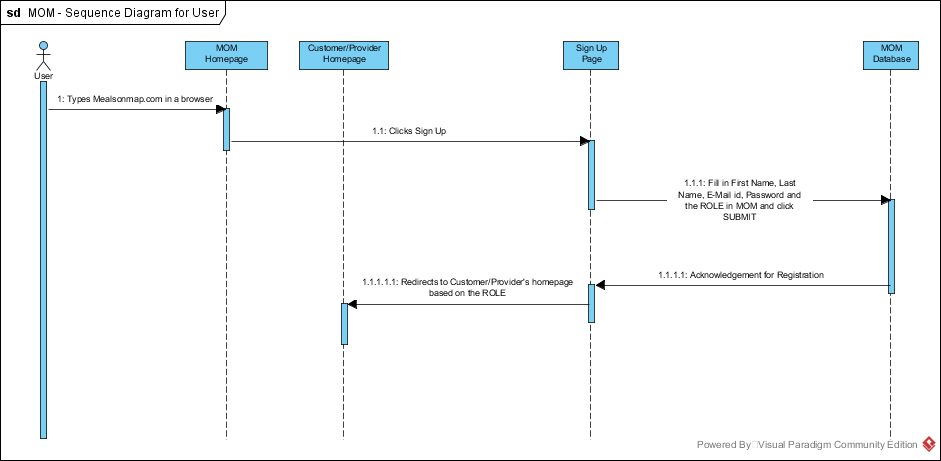
4.1.2 System Use Case Diagram



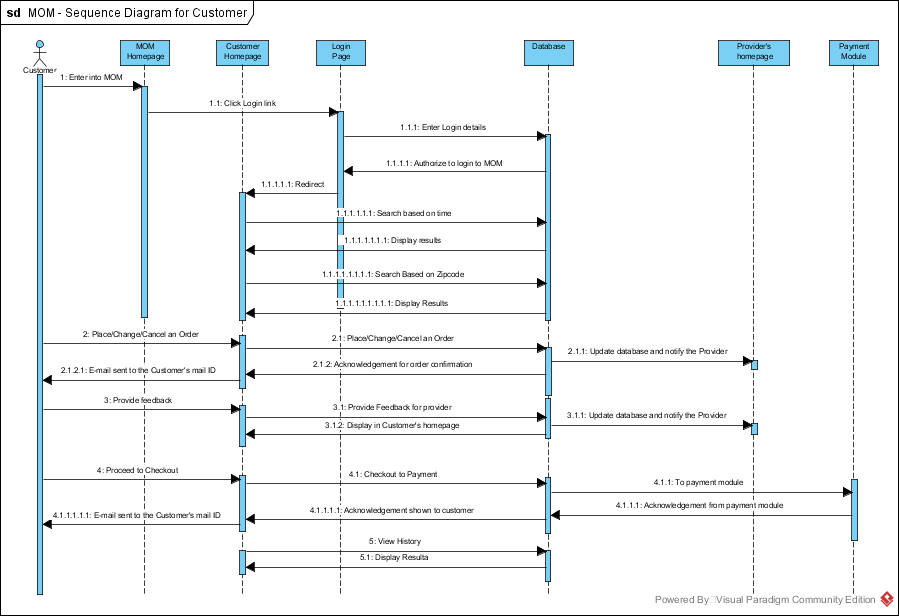
#### 4.1.2.1 Logical View

##### Sequence Diagrams

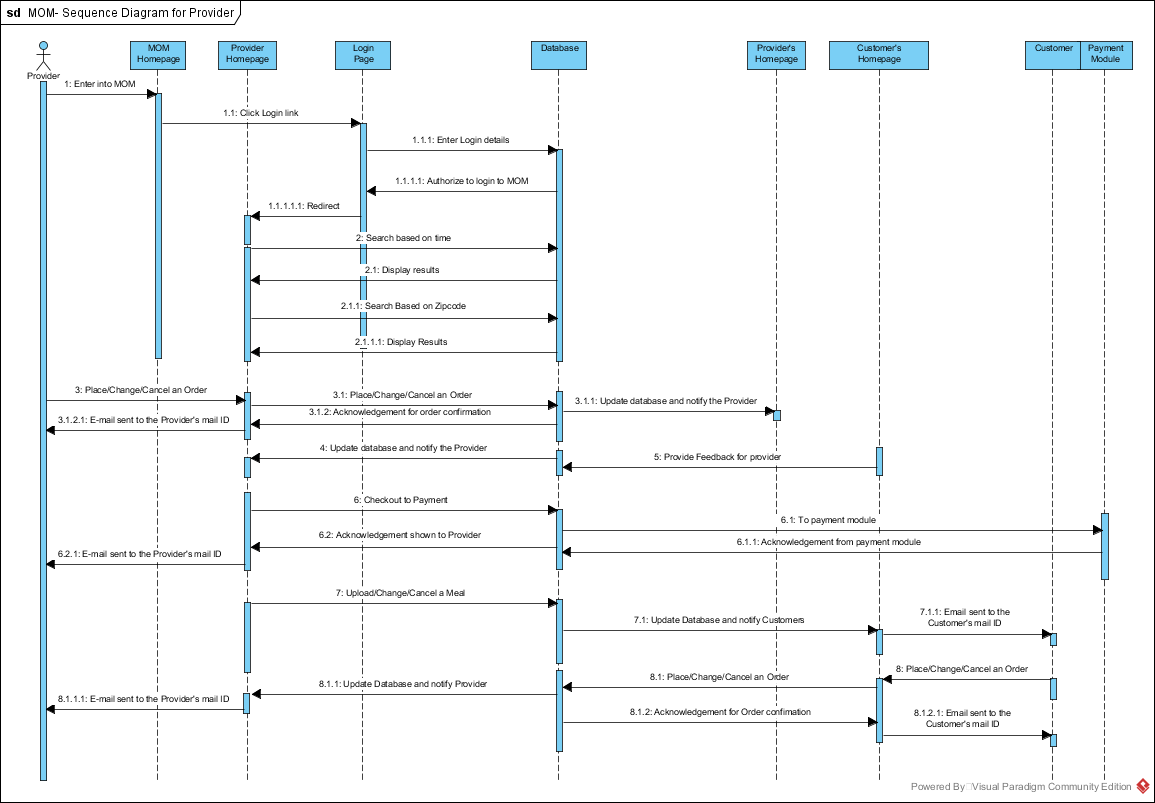
###### Anonymous User



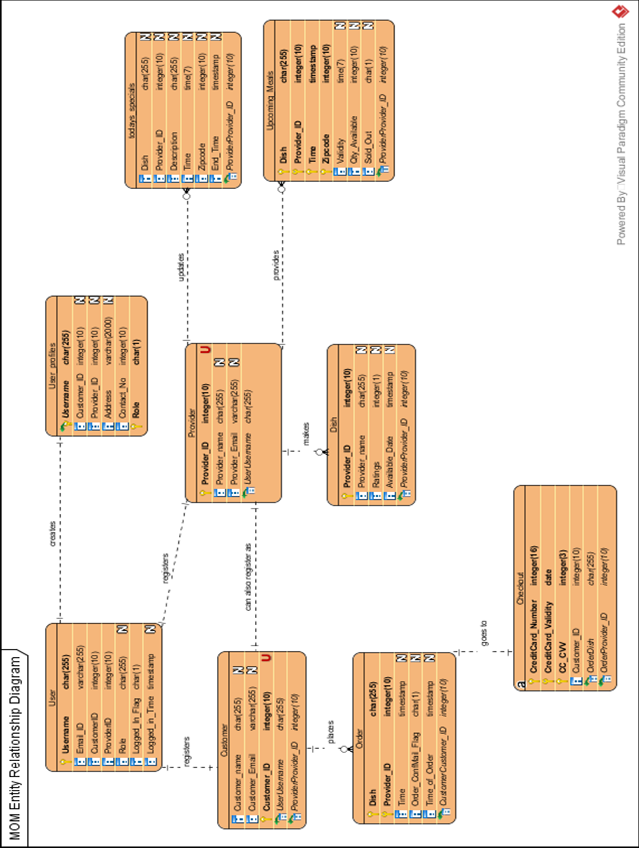
###### Customers



###### Provider



##### b) Entity Relationship Diagram



# **7. Design Overview**

## **7.1. Design Rationale**

Currently, we are building a web application for this application to be successful we think it is necessary that the application holds below mentioned qualities. It should:

* Work as per desired during peak hours with multiple users accessing the website simultaneously.
* Model the data access through a centralized data store to have high security enforced through a central server.
* Support high volume usage by having load balancing and fault tolerance mechanisms.
* Support accessing the application from different environments (different devices, browsers etc.)
* Be scalable and each tier should scale horizontally.
* Have high performance. Presentation tier should be able to cache requests, so network utilization is minimized, and the load is reduced on the Application and Data tiers.
* Be highly available. We should be able to make each layer fault-tolerant independently. If the Application tier server is down and caching is sufficient, the Presentation tier should be able to process web requests using the cache. We should be able to add redundant database servers to have fault-tolerance for DB tier.

Given the above needs, we have finalized to design our architecture following **client-server**

**Architectural style**, and to use **4-tier architectural pattern** model.

Along with above architectural needs, it is necessary that application development follows certain good practices as highlighted below.

* Separation of concerns - business logic should be independent and should be able to be used from different presentation tiers (mobile, tablets, desktops).
* Parallel development by separate team’s - developers of UI and business logic should be able to focus exclusively on their part.
* Test driven - should allow easy unit/ functional/ integration testing for large scale applications.

We have decided to go ahead with **MVC pattern** to take into consideration the above developmental needs.

Overall, we have planned to develop our application using **Django framework (Python)** which satisfies our design need of using client-server architectural style and MVC pattern. Django uses a “shared-nothing” architecture, which means you can add hardware at any level – database servers, caching servers or Web/application servers. We have planned to use below tools and technologies for development of our application.

Table 7.1.1

|  |  |
| --- | --- |
| Server side programming | Python 2.7.11 |
| Framework | Django 1.8.12 |
| Client side programming | HTML, CSS, JavaScript |
| Database | MySql |
| Document sharing | Google Docs |
| Cloud Servers | AWS |
| Version Control | Git |

## **7.2. Software Architecture**

### **7.2.1 High-level Architecture:**

The high-level architecture (HLA) is a general purpose architecture for distributed computer simulation systems. Below is the high-level architecture for our project.

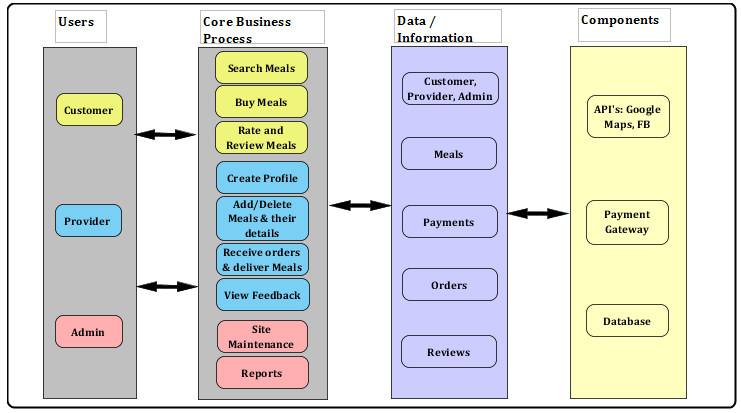


Figure 7.2.1

### **7.2.2 4-tier Architecture:**

In software engineering, multitier (n-tier) architecture is a client-server architecture. Basically here the presentation, application processing (business logic) and data management functions are physically separated. Below diagram indicates various layers in our application.

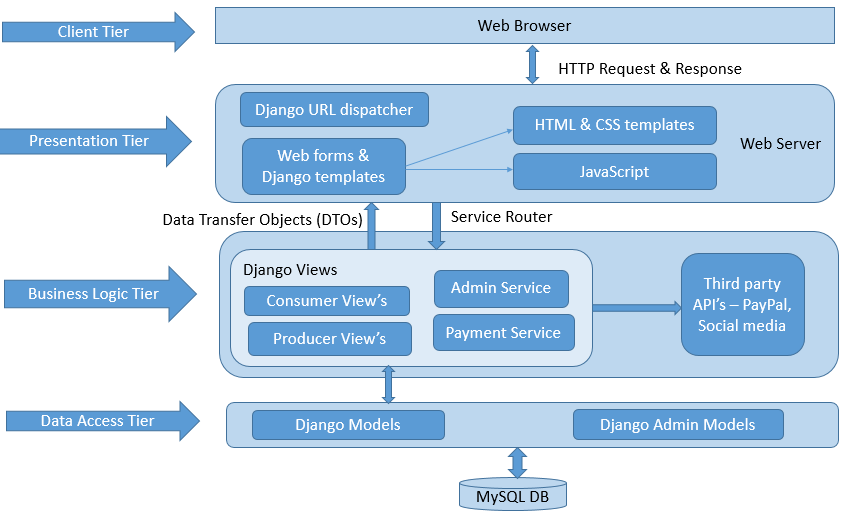


Figure 7.2.2

The first tier is the Client tier, facing the user. It is called user interface (UI), in our case it’s the web browser which our user will use to access our application from his machine (client).

Presentation tiers mainly helps to translate tasks and results to something the user can understand. It communicates with other tiers by which it puts out the results to the browser/client tier and all other tiers in the network. Dynamic HTML & CSS templates, and JavaScript are used to implement this layer in our application. The URL dispatcher (urls.py) maps the requested URL to a view function and calls it. This tier interacts with the Business logic tier through a set of services. Presentation tier holds the look and feel of the web application.

Business logic tier contains the business logic and algorithms (basically all the functionality that the user is expecting), and is comprised of several components. The Django views module (usually views.py) shown in the above figure performs the requested action, typically involves reading or writing to the database. It may include other tasks as well. They handle the functionality related to meals, consumers and providers etc. All these views interact with other third party components like Facebook and PayPal for user authentication and payment processing respectively.

Data Access tier holds the Django models and Django Admin models. The model (usually models.py) defines the data in Python code and interacts with it. We will be capturing all the necessary data required by the website in database tables. We are using the MySQL database here. Business logic tier interacts with this layer to perform CRUD (Create, Read, Update, Delete) operations on objects/records.

### 

### **7.2.3 MVC Architecture:**

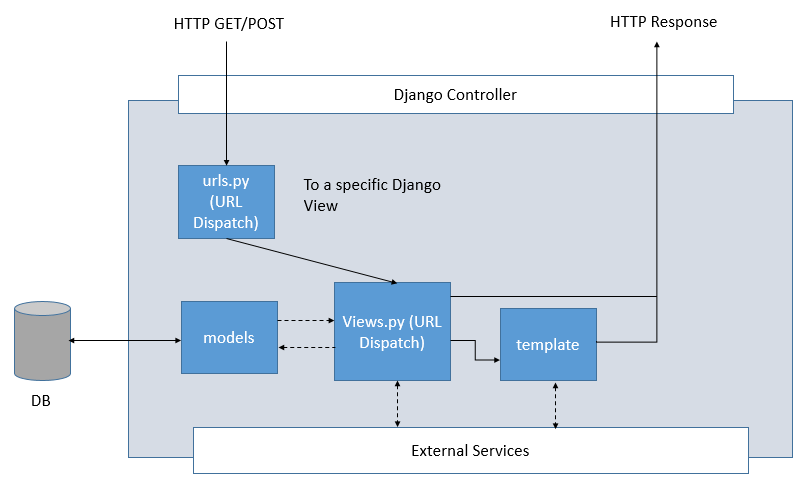


Figure 7.2.3

Django is a free and open source web application framework, written in Python, which follows the Model-View-Controller architectural pattern.

The data access logic, business logic, and presentation logic – comprise a concept that’s sometimes called the Model-View-Controller (MVC) pattern of software architecture.

Django controller handles HTTP requests and responses. Django reads a settings file so that it knows what to load and set up. Django reads the URL config file that tells it what to do with the incoming requests from the browser. The urls.py resolves the incoming request (by matching it with regex) and redirects it to the appropriate view.

In Django, a “view” describes which data is presented, but a view normally delegates to a template, which describes how the data is presented. The Views actually hold the HTML pages and the python code. It also has an automatic web admin interface for editing the models.

Django models holds all the data that a web application needs. Models basically describes your data.

To summarize,

Models – Describes your data

Views – Controls what user’s see

Templates – How user’s see it

Controller – URL dispatcher

# **REFERENCES:**

1. <http://www.projecttimes.com/articles/project-risk-identification-for-new-project-manager.html>
2. <http://www.brighthubpm.com/project-planning/60264-techniques-used-in-business-requirements-gathering/>
3. <https://en.wikipedia.org/wiki/High-level_architecture>
4. <https://en.wikipedia.org/wiki/Multitier_architecture>
5. <https://djangosites.org/>