Concordia University
Department of Computer Science
and Software Engineering

Touch For Food

Architecture

SOEN 490
Capstone Project
Fall 2012 – Winter 2013

CloudNine

Josh Hum (Team Leader)	9583157
Katrina Anderson	9106251
Cristian Asenjo	9280014
Christian Daher	9599673
Cynthia Donato	9353852
Mikhail Levkovsky	9583165
Patrick Modafferi	9401377
Ryan Nasr	9605614
Matthew Tam	9675701

Table of Contents

List of Figures

List of Tables

Touch For Food

Architecture

Version 11.48

Revision History

Date	Rev.	Description	Author(s)
2012-09-17	0.00	Document Creation	Katrina Anderson
2012-09-26	0.01	Contributed to Section 3 Architectural Goals and Constraints	Mikhail Levkovsky
2012-09-26	0.02	Contributed to Section 10 Quality	Mikhail Levkovsky
2012-09-28	0.03	Contributed to Section 1 and 2	Patrick Modafferi
2012-09-29	0.04	Contributed to Section 8 (Physical View) and 9 (Size & Performance)	Cristian Asenjo
2012-10-01	0.05	Contributed to Section 5.2 and 5.3. added the figures to Table of Figures	Christian Daher/ Ryan Nasr
2012-10-14	0.06	Contributed to Section 7.1	Matthew Tam
2012-10-20	1.07	Updated to sprint 1Contributed section 4.3	Patrick Modafferi/ Mikhail Levkovsky
2012-10-26	1.08	Added Sequence Diagrams Populate Menu and updated actor goal list	Mikhail Levkovsky
2012-10-27	1.09	Added Communication Diagrams Populate Menu	Patrick Modafferi
2012-11-04	2.10	Reviewed and made minor corrections	Josh Hum
2012-12-16	3.11	Added clarification for load balancing solution.	Cristian Asenjo
2012-12-19	4.12	Removed outdated use cases, updated use case models, included use case 2.1	Katrina Anderson
2012-12-20	4.13	Updated class diagram & domain model	Josh Hum & Matthew Tam
2012-12-21	4.14	Reviewed document and reduced redundancies	Katrina Anderson
2013-01-10	5.15	Added Appendix B Prototypes	Katrina Anderson
2013-01-14	5.16	Updated Section 3	Mikhail Levkovsky
2013-01-16	5.17	Added Prototypes to the Appendix	Patrick Modafferi
2013-01-27	6.18	Added sequence diagram 5.3.2 and state diagram 6.4.2	Cynthia Donato
2013-01-27	6.19	Review and update sections	Patrick Modafferi
2013-01-28	6.20	Added logging in activity diagram	Cristian Asenjo
2013-01-28	6.21	Added logging in sequence diagram	Cristian Asenjo
2013-01-28	6.22	Added Review prototype	Patrick Modafferi
2013-02-02	7.23	Added Communication Diagram and Sequence Diagram	Mikhail Levkovsky
2013-02-02	7.24	Added ERD diagram	Mikhail Levkovsky
2013-02-04	7.25	Added Sequence Diagram	Mikhail Levkovsky
2013-02-07	7.26	Added a diagram for MVVM	Patrick Modafferi
2013-02-08	7.27	Updated Class diagram	Christian Daher
2013-02-10	7.28	Added Activity Diagram 6-5	Matthew Tam
2013-02-11	7.29	Added Sequence diagrams 5-6 and 5-7	Cynthia Donato

2013-02-12	7.30	Added System Sequence diagram – Create Menu	Josh Hum
2013-02-12	7.31	Added Placing Order activity diagram	Cristian Asenjo
2013-02-12	7.32	Updated Domain Model	Cynthia Donato
2013-02-12	7.33	Reviewed SAD and added Create Review Prototype	Mikhail Levkovsky
2013-02-12	7.34	Reviewed SAD for formatting mistakes	Josh Hum
2013-02-28	9.35	Removed some sections and put them in Requirements doc and Analysis doc. Made corrections to reflect that some information is no longer here. Updated references. Fixed formatting issues.	Josh Hum
2013-03-02	9.36	Added SD27.1 Create Menu sequence diagram	Josh Hum
2013-03-02	9.37	Updated Review Sequence Diagram	Mikhail Levkovksy
2013-03-02	9.38	Repasted Cynthia's SD34.1 into document.	Katrina Anderson
2013-03-02	9.39	Added Login to Personal Profile state diagram	Cristian Asenjo
2013-03-02	9.40	Added Login to Personal Profile sequence diagram	Cristian Asenjo
2013-03-03	9.41	Added Place order Sequence Diagram	Patrick Modafferi
2013-03-05	9.42	Updated section 2	Josh Hum
2013-03-05	9.43	Added section on concurrency	Cynthia Donato
2013-03-05	9.44	Fixed state diagrams	Ryan Nasr
2013-03-06	9.45	Split up the Class Diagram more	Christian Daher
2013-03-06	9.46	Removed and Updated Activity Diagram	Matthew Tam
2013-03-06	9.47	Reviewed document	Josh Hum
2013-03-30	11.48	Reviewed document	Matthew Tam

1 Introduction

1.1 Purpose

The purpose of this document is to determine, on an architectural level, how each part of the TFF application will work. These architectural diagrams represent the inner workings of the system. When it comes to implementing, developers should refer to this document to save time and maintain consistency.

There are many types of diagrams some help determine the flow of the system and some explain how components interact with one another. The class diagram shows how objects are related. Finally, some diagrams show more physical elements like the deployment diagram.

Another reason for this documentation is scalability, maintainability and reusability. In order to allow for someone to potentially expand on this project, reuse it for something else or maintain the finished product, the documentation must be able to serve as a road map or a guide for any future developer.

1.2 Scope

The scope of the architecture artifacts cover the main components of the system. As TFF is developed, the components planned for each sprint will be designed. The resulting architecture and design diagrams will be added to the document. In this way, the document will incrementally be built up to a final document containing diagrams for the essential components in the system.

1.3 Definitions, Acronyms and Abbreviations

Refer to the Requirements document - Appendix B Glossary and Appendix C Acronyms for a complete list of terms and definitions.

1.4 References

Please see Appendix A, References, of this document.

1.5 Overview

The document is organized into four main views to show four of the 4+1 architectural views: Logical, Development, Process and Physical (See section 2 for more details). The Use Case view can be found in the Analysis document. As needed, other items will be discussed such as Quality, Size and Performance. This document will also discuss architecture styles and constraints as well as a comparative analysis to justify some decisions that were taken. As a whole, the document should completely represent all the architecture of the system.

2 Architectural Representation

Section 2 of this document provides a summary of the architecture that represents TFF. The remaining sections represent the four of the 4+1 views of the TFF system. This section of the document summarises what architectural artifacts are found in each view.

Table 2-1 Diagram Types Vs. Views

Views	Use Case	Logical	Process	Physical
Diagram Types	Actor-Goal List	Domain Model	Activity Diagrams	Deployment Diagram
	Use Case Diagrams	Class Diagram	State Diagrams	
	Use Cases	Sequence Diagrams		

2.1 Use Case View

The use case view represents important requirements through fully detailed use cases. Use case designations are also represented through actor goal lists and use case models.

2.2 Logical View

The logical view determines how the layers, packages, classes and other software elements are organized in the system.

2.3 Process View

The process view illustrates processes and threads in the system. Could show how some elements interact and collaborate throughout a process.

2.4 Physical/Deployment View

The physical view represents the components, processes, communications and important structures of the system.

3 Architectural Goals and Constraints

TFF will be designed using an MVC architecture. Reasons for choosing an MVC architecture are the following:

- Decouples presentation, data and domain logic
- Allows multiple people to work on different parts of the same project
- Promotes low coupling between different components
- Promotes organization and code reuse
- Promotes consistent and well defined interfaces between each layer

Well-structured MVC architecture will also allow us to swap out or update any component independently of the others as long as the interfaces are respected. The Model-View-View-Model architecture will also be employed, which is an extra component to the classic MVC architecture.

MVC consists of:

- Model – The models will be database driven, that will be used to represent the state of the application
- View – Determines how to present the model data to the user
- Controller – Passes the information between the user requests and the model, and vice-versa.
- View Model – Will be used when extra logic is required for the view to properly render. (see Figure 3-1)

Some constraints of the MVC architecture are:

- Tracing end-to-end flows of data can be challenging as the system grows

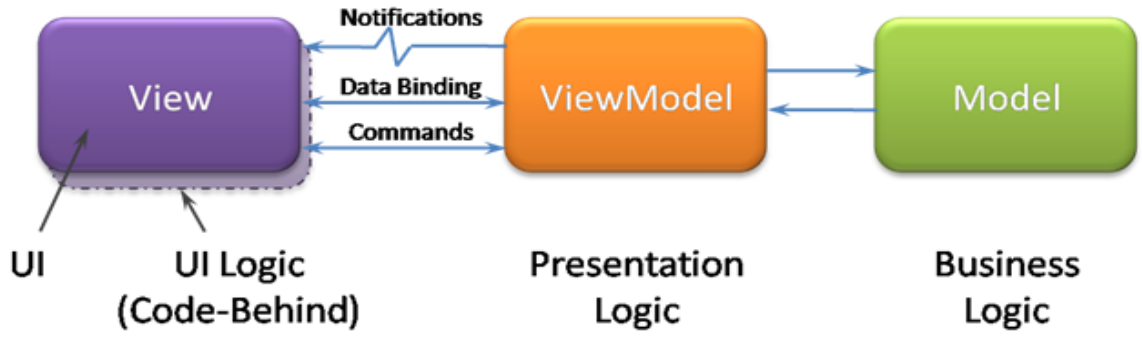


Figure 3-1 Diagram for a typical MVVM Pattern that goes with MVC [1]

3.1 MVC vs. 3-Tier Architecture

Since 3-Tier Architecture is used for large scale enterprise solutions which require physical separation, MVC has been chosen to develop TFF. MVC allows us to represent each component of the system as its own encapsulated part. If needed, MVC can be converted to a 3-Tier Architecture.

4 Logical View

4.1 Sequence Diagrams

SD27.1 Order Item Management

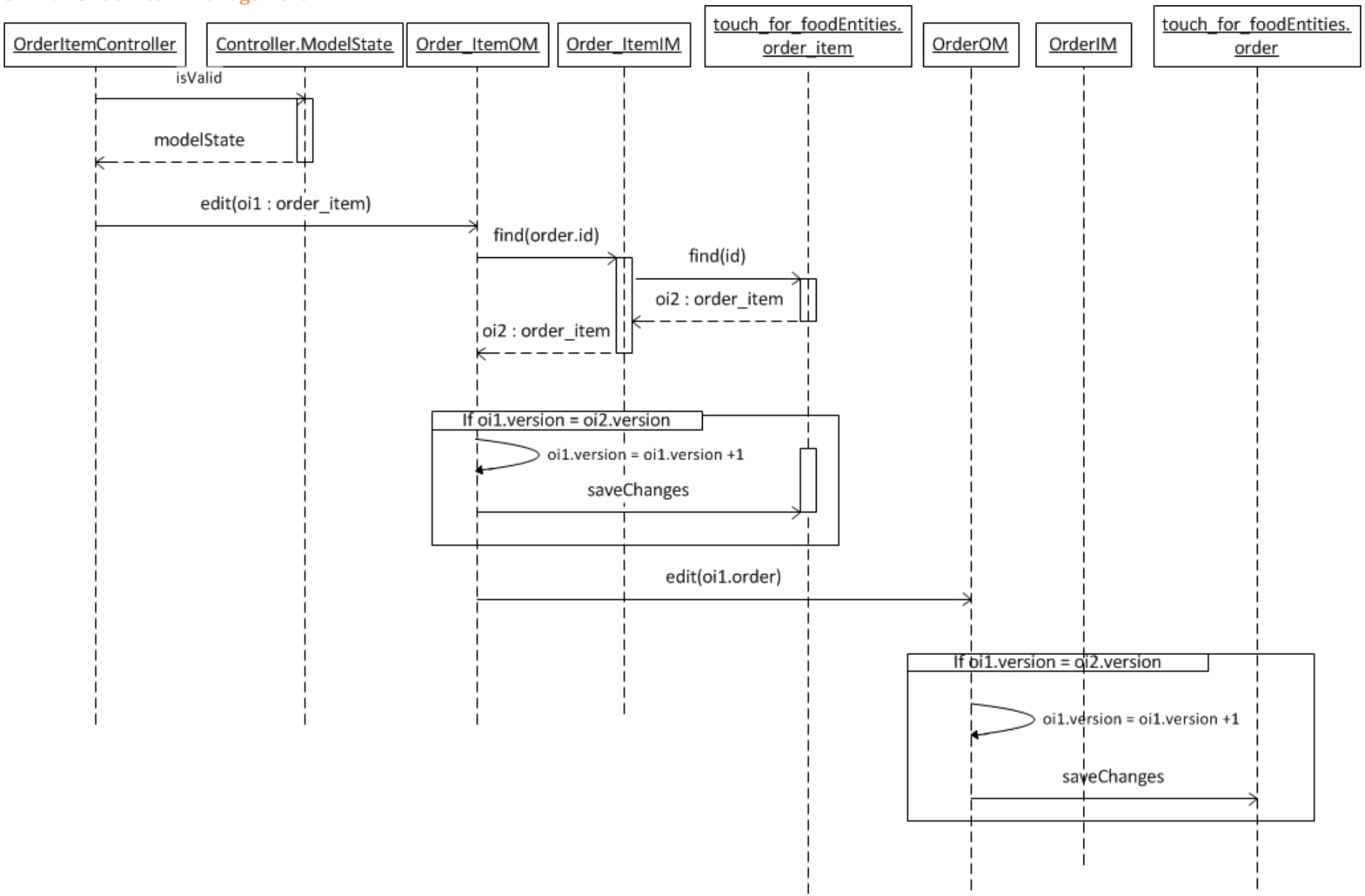


Figure 4-1 Order Item Management Sequence Diagram

SD27.2 Create Menu

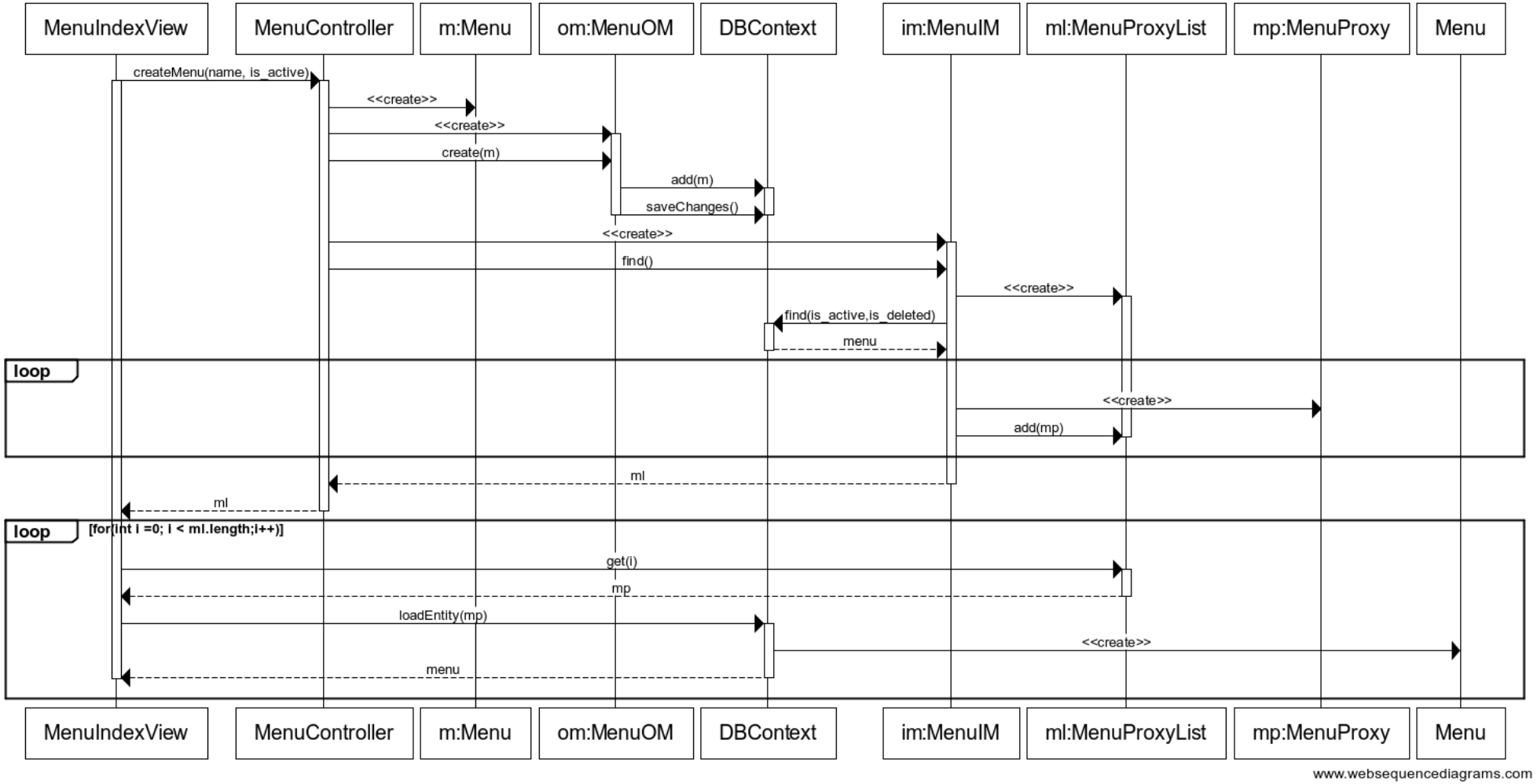


Figure 4-2 Create Menu Sequence Diagram

SD27.3 Create Category

To view a larger version of this diagram in a browser, follow this link: http://goo.gl/atX2l

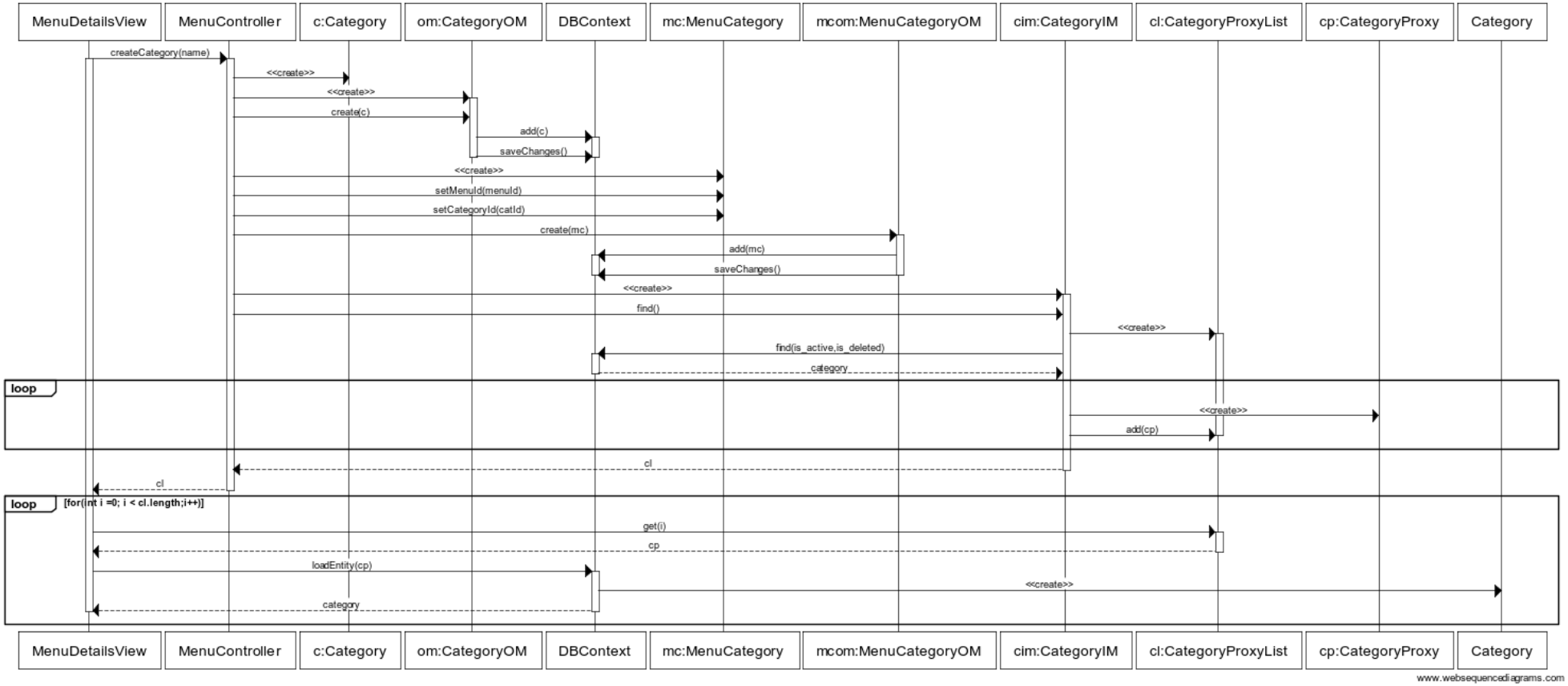


Figure 4-3 Create Category Sequence Diagram

SD27.4 Create Item

To view a larger version of this diagram in a browser, follow this link: http://goo.gl/ZdqO4

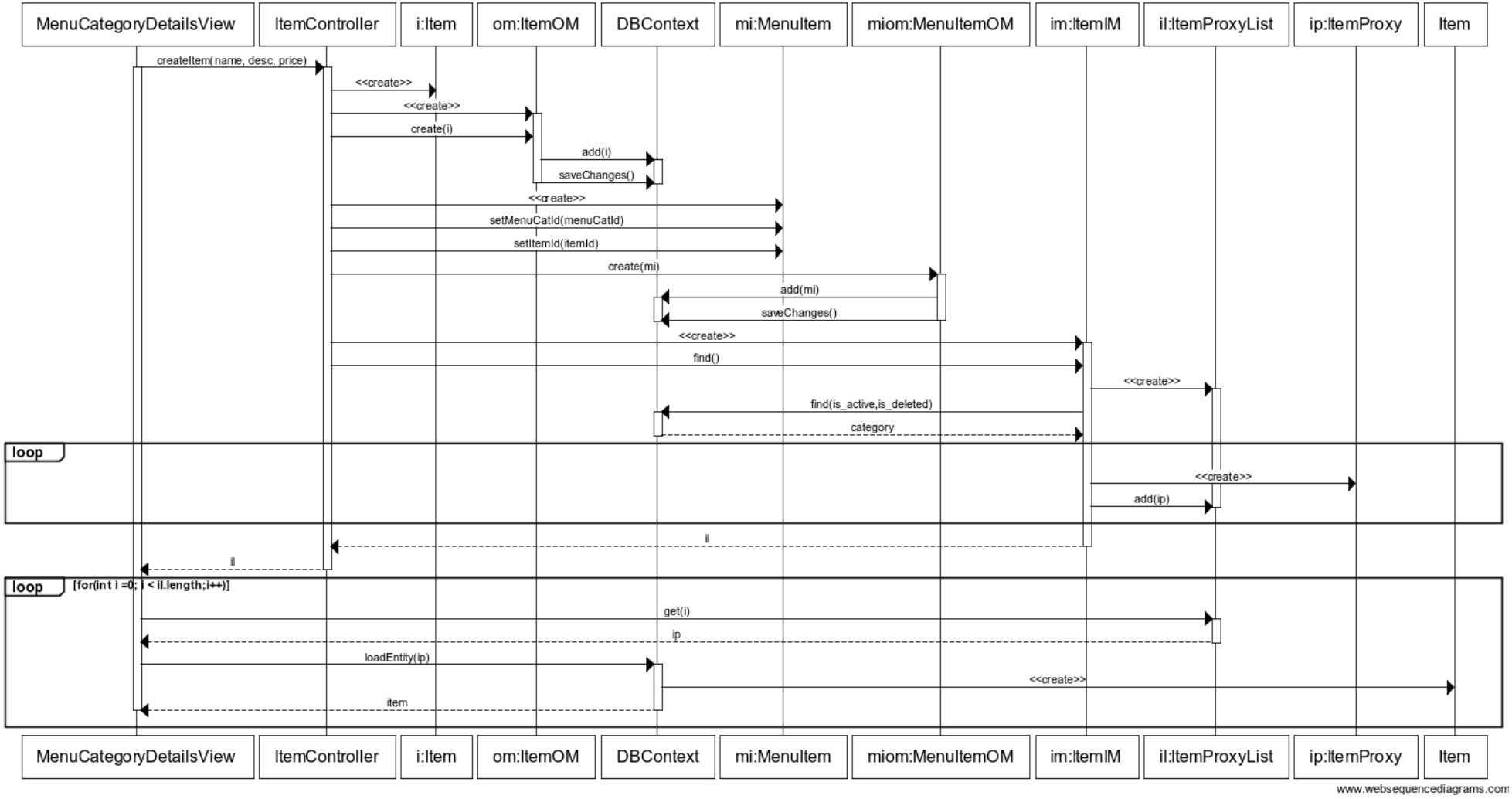


Figure 4-4 Create Item Sequence Diagram

SD34.1 Service Request

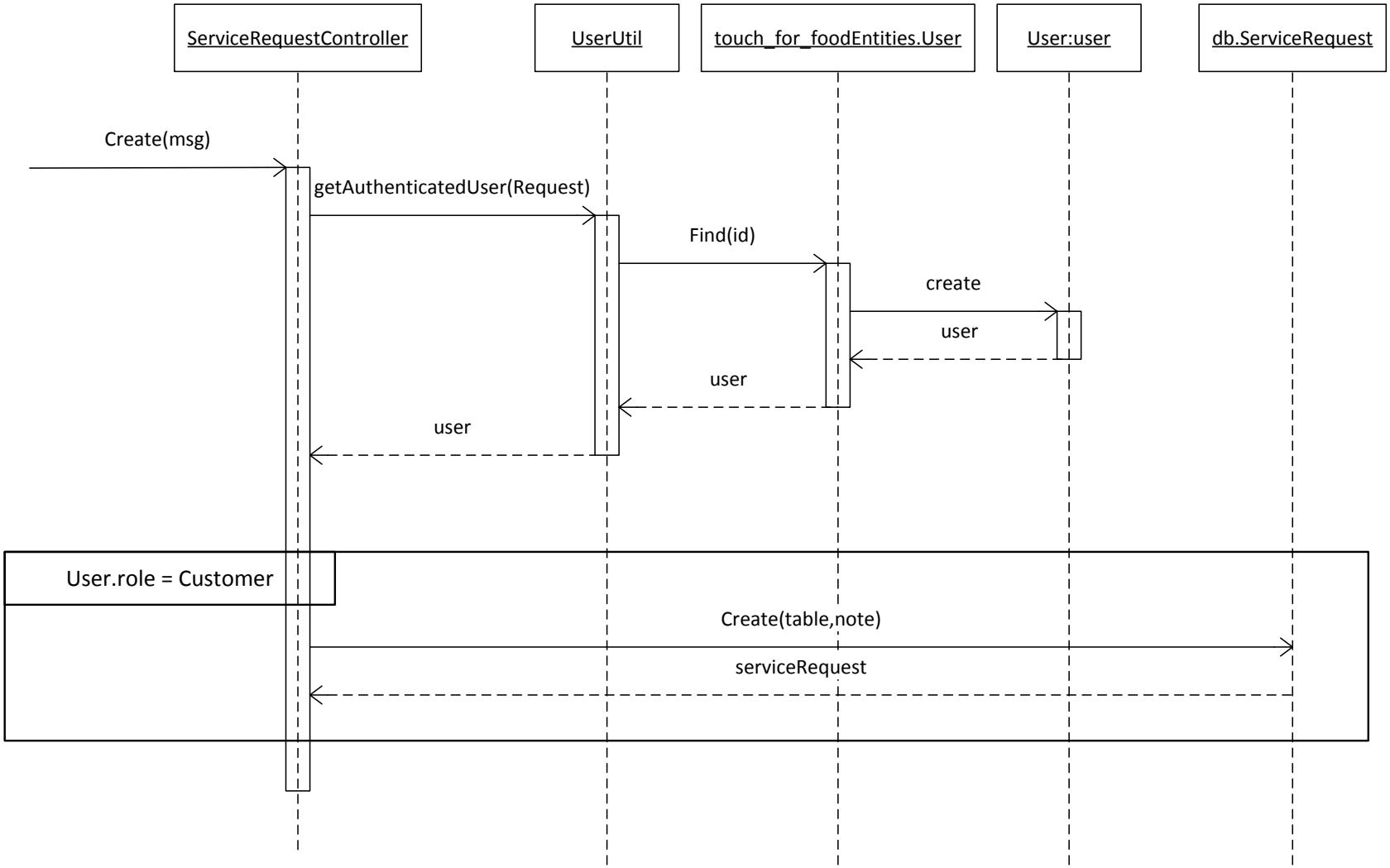


Figure 4-5 Service Request Sequence Diagram

SD35.2 Login to Personal Profile

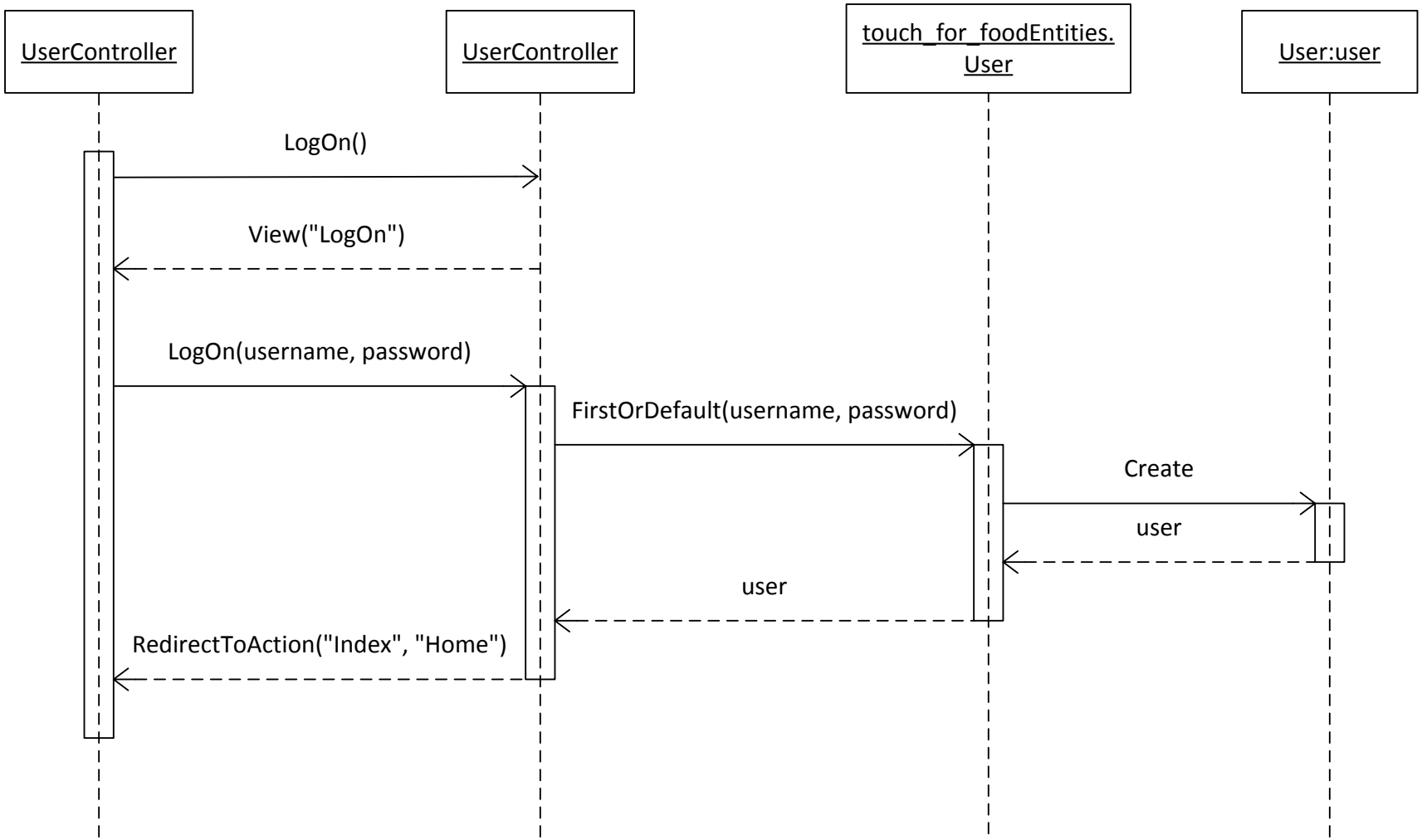


Figure 4-6 Login to Personal Profile Sequence Diagram

SD39.1 Submit a Review

To view a larger version of this diagram in a browser, follow this link: http://goo.gl/RKL6n

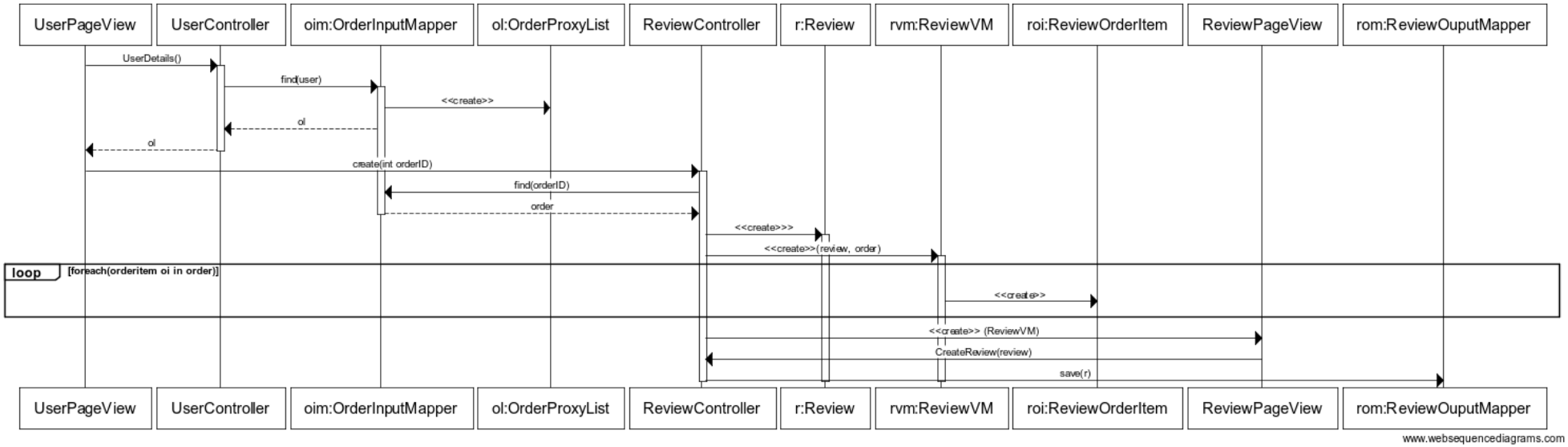
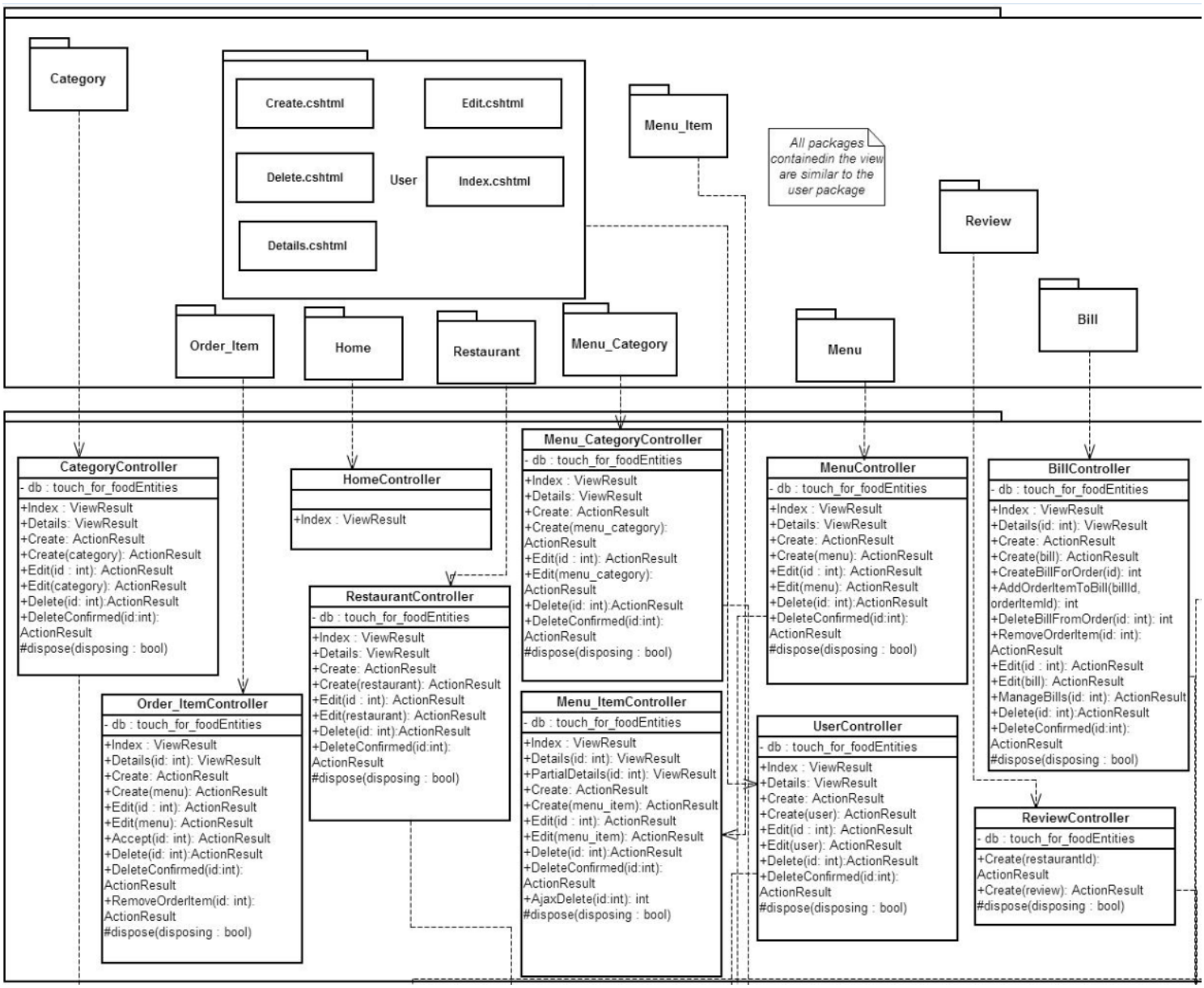


Figure 4-7 Submit Review Sequence Diagram

4.2 Class Diagram



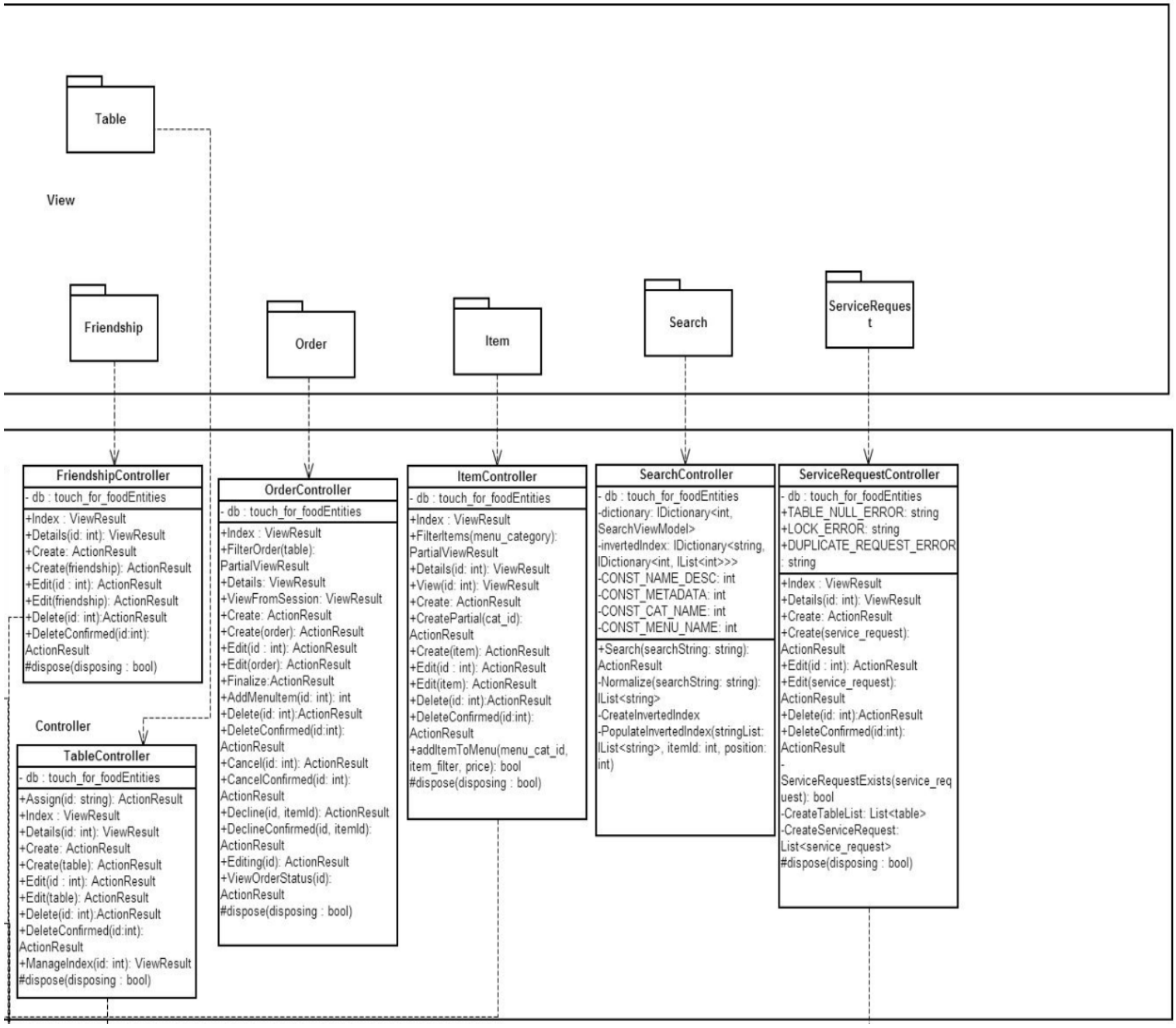


Figure 4-8 Class Diagram

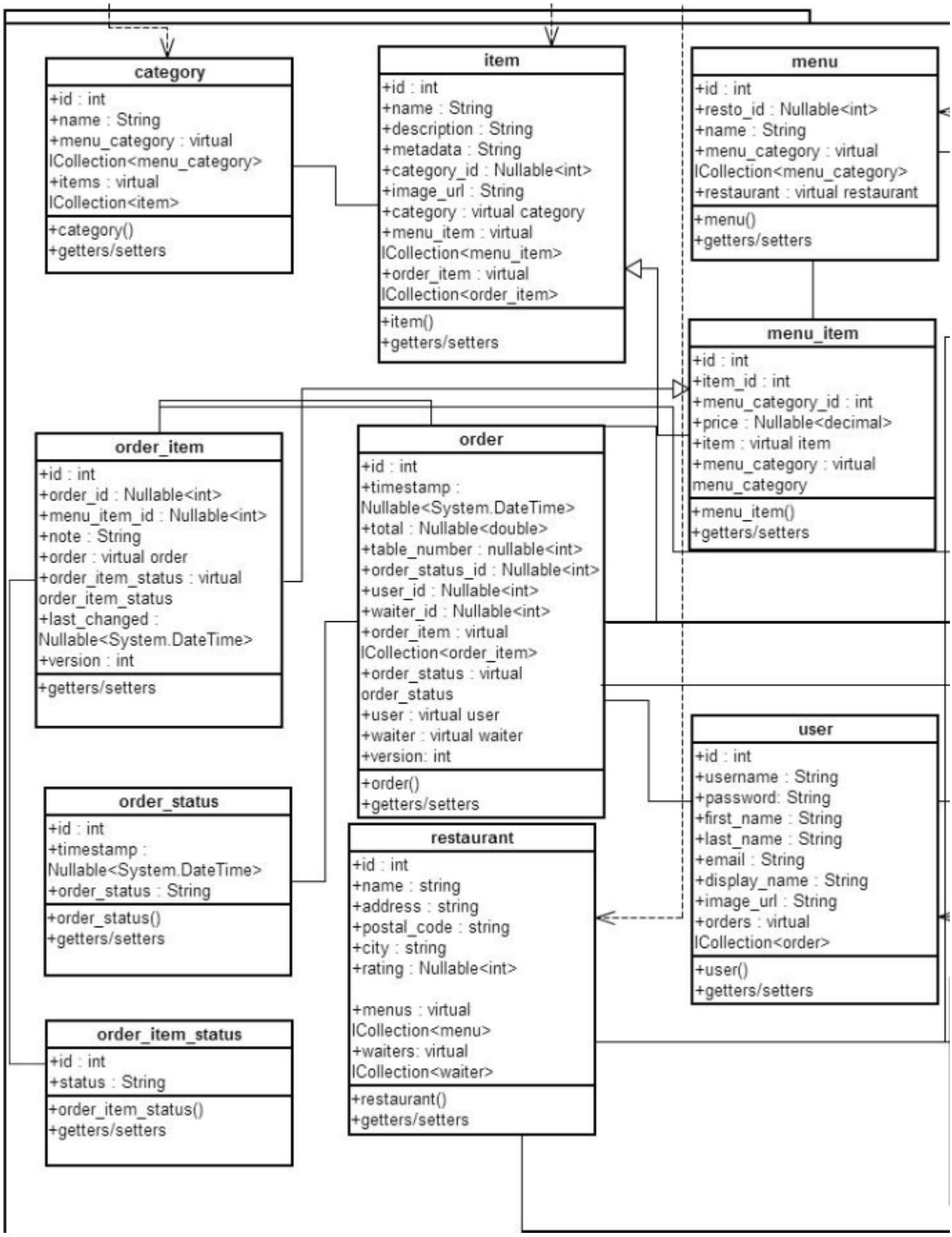


Figure 4-9 Class Diagram (Model)

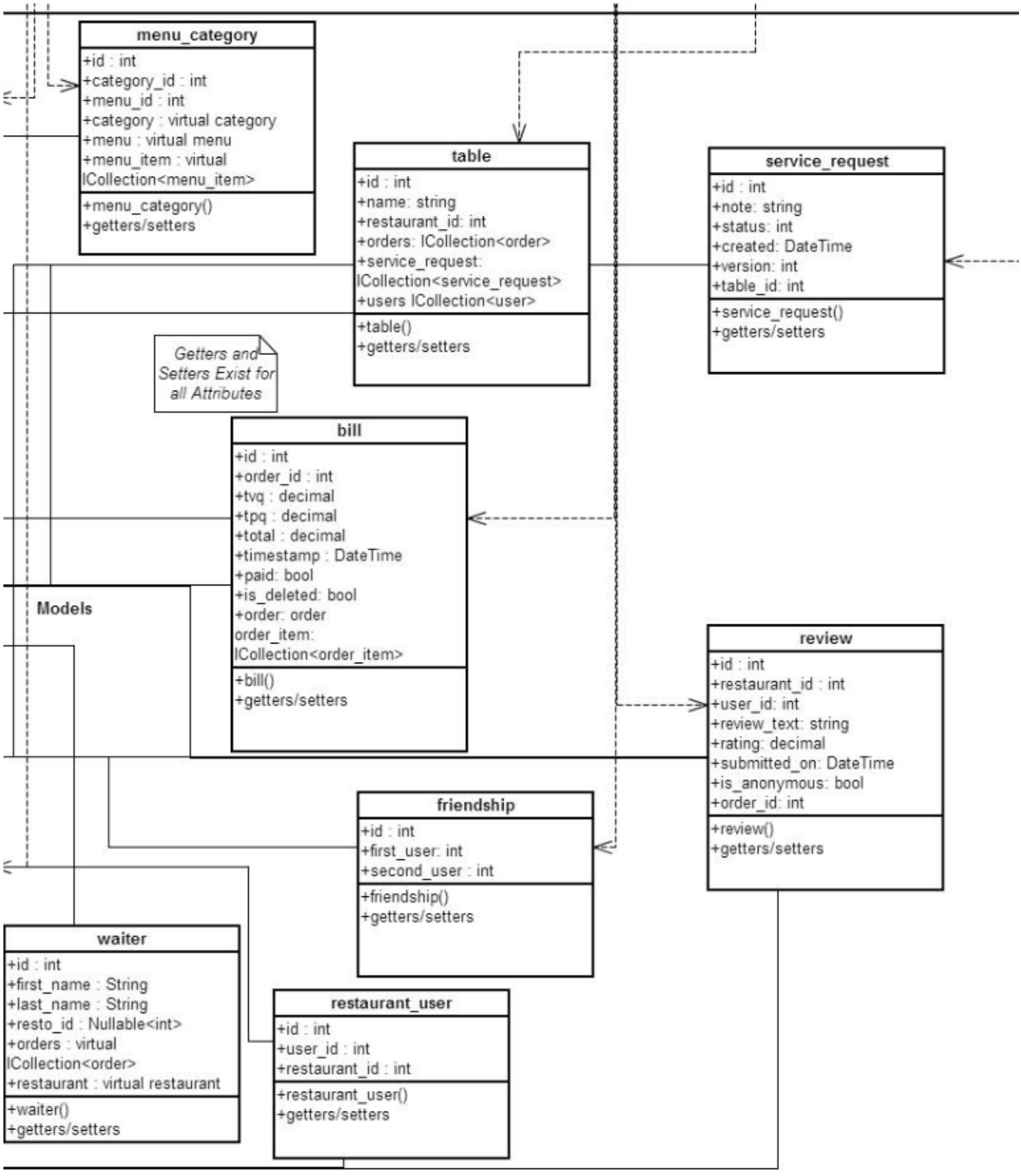


Figure 4-10 Class Diagram (Model) 2

4.3 ERD Diagram

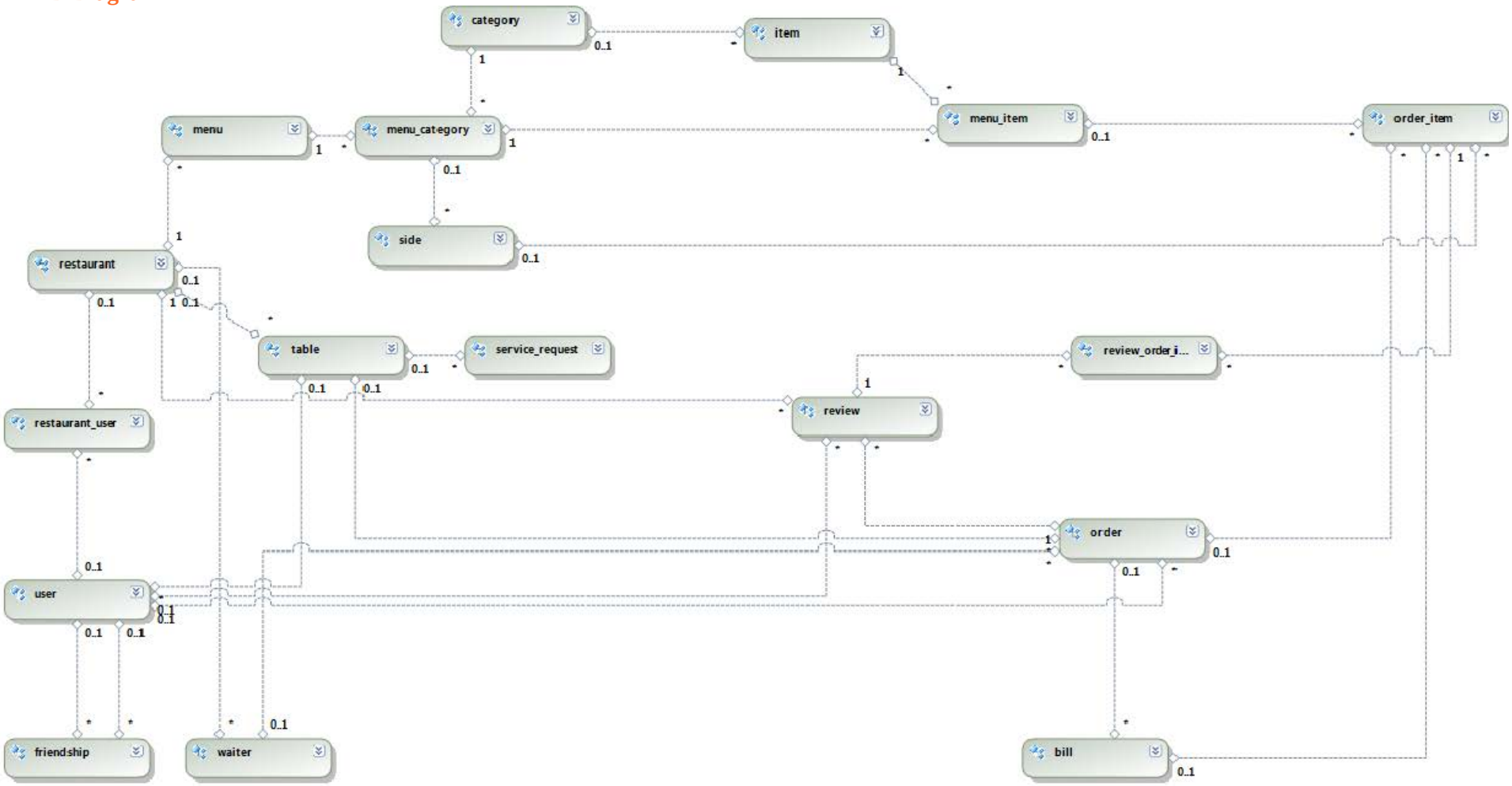


Figure 4-11 ERD Diagram

5 Process View

5.1 Activity Diagrams

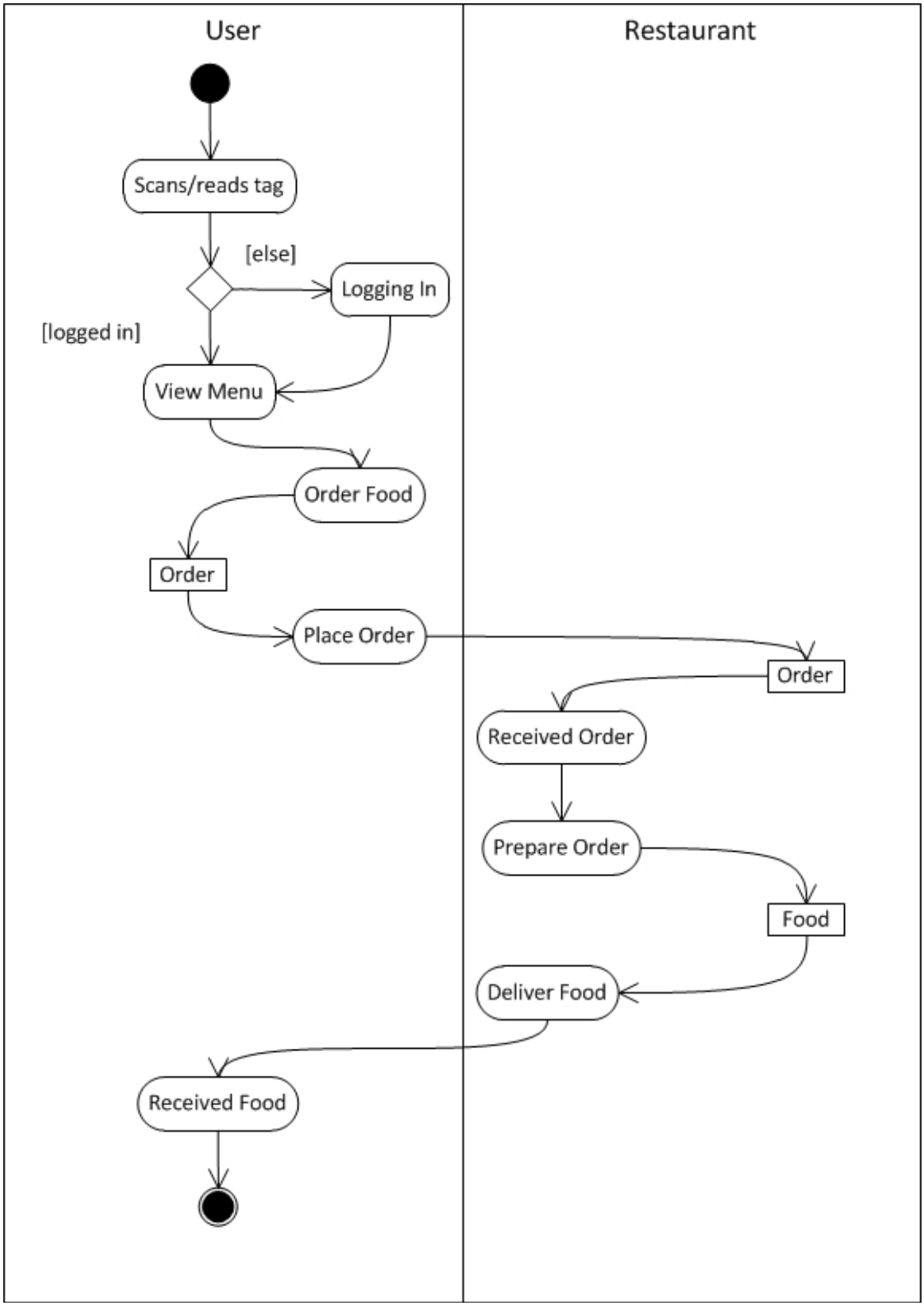


Figure 5-1 Process Order

5.2 State Diagrams

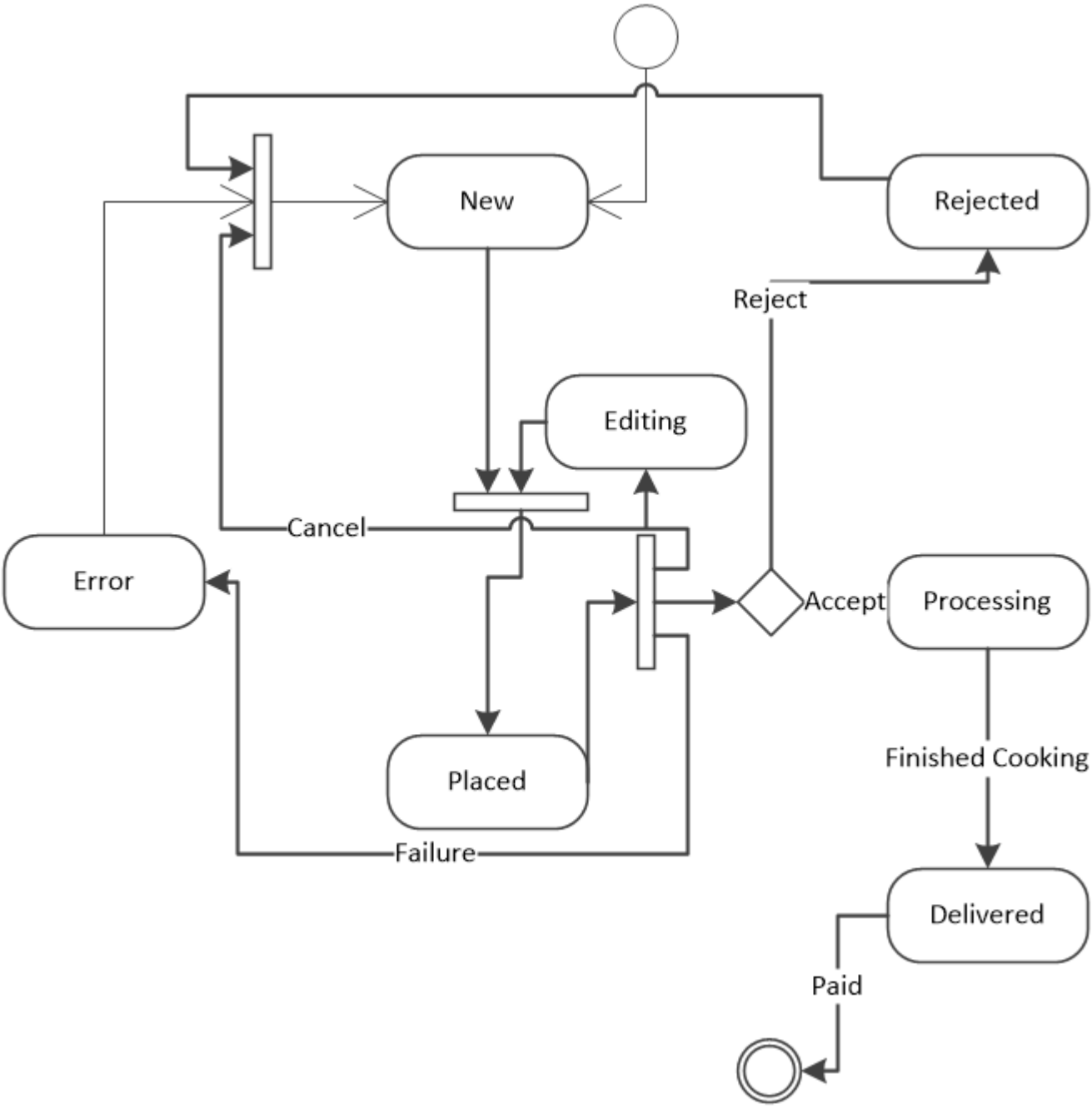


Figure 5-2 State Diagram for Order Food

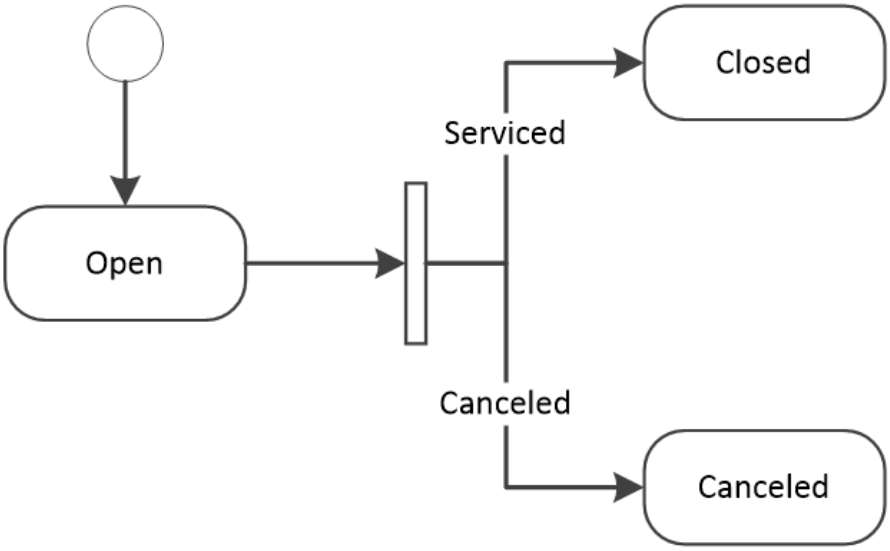


Figure 5-3 Call Waiter State Diagram

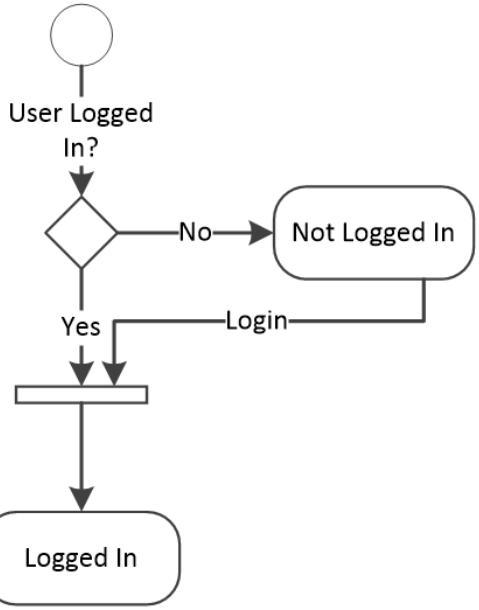


Figure 5-4 Login to Personal Profile State Diagram

6 Physical View

6.1 Deployment Diagram

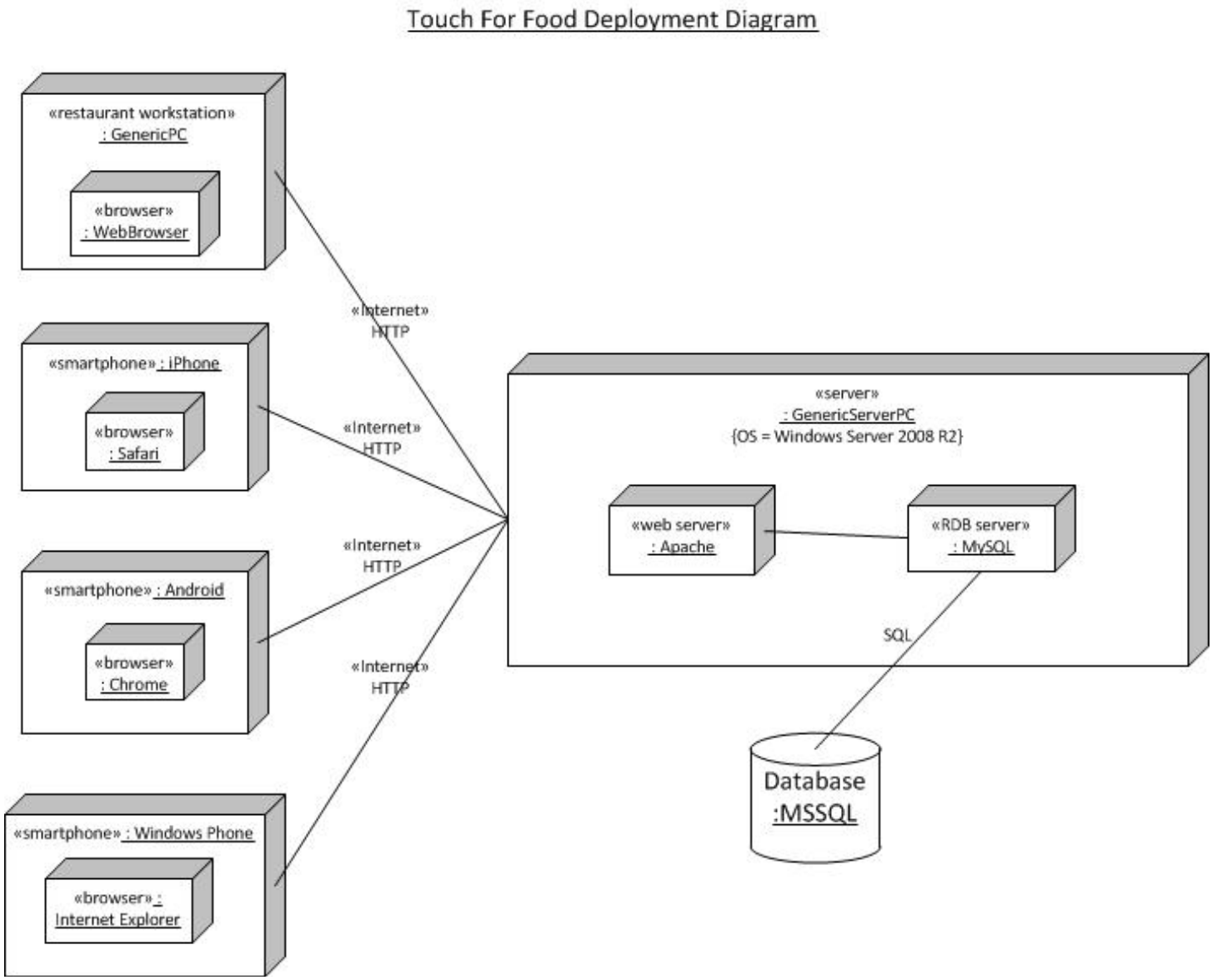


Figure 6-1 Deployment Diagram

7 Size & Performance

From a backend point of view, the element that will impact the performance the most will be the .Net framework along with the MSSQL database server. Many functions used in TFF are already included in the framework and as such are optimized to work better with the MSSQL database server, as opposed to a third party solution like MySQL. Additionally, the .Net framework includes functions and tools that can be used to form relations between different elements in the most efficient way possible.

In terms of the front end, the performance will greatly depend on the power the device accessing the mobile application (or website) actually has. In order to mitigate this potential performance hit, a JavaScript library such as JQuery or Dojo will be used. Additionally, if the user is accessing the system through the mobile application, the use of native (i.e. compiled) code will also help the overall performance of it.

The size of the system will initially be small, with one server to provide the order services. When/if the system begins to grow, then more servers will have to be deployed in order to maintain a balanced load across all of them. Windows Server 2008 R2 has an optional Network Load Balancing (NLB) component which would be used in order to configure a NLB cluster. This NLB cluster would be seen by the outside world as a single virtual server which would distribute the traffic equally between each server that makes up the cluster.

8 Quality

TFF is being developed with quality in mind. The architectural designs have been discussed and made to ensure that the quality of TFF is high.

Maintainability, scalability, security and portability are the 4 focuses of TFF. The code base needs to be flexible and easily maintained. Bugs should be easy to locate within TFF's code base. New and different components must be easy to add/modify. This will be ensured by keeping a consistent architecture as well as deploying necessary GRASP patterns.

Since TFF will be exposed to the internet, and it collects user data, security will also be a main focus. Steps will be taken to ensure that malicious users and hackers do not get access to the system and are not able to abuse it.

For more information on NFR related quality measures, please refer to the Requirements document, Supplementary Specifications section.

9 Concurrency

The TFF application allows multiple users to access and update shared data. For this reason it was imperative that concurrency be addressed by the TFF system. The TFF system manages concurrency using an optimistic offline lock. The lock is implemented through the use of a version field in the database. All data that is stored and can be updated concurrently in the database has such a field. This field is represented by a simple integer that is incremented with each change to the particular line item in the database.

Each time an update is made to the particular line item in the database the version field is first verified to make sure that the in memory object and the stored object are the same. If they are the same, the necessary updates to that field are performed as well as incrementing the value of the version.

This ensures that if two users are accessing the same page and updating it at the same time, the user who saves first will be granted the rights to save their changes. When the second user then goes to save their changes, their save will have failed. They will be informed that someone else has updated that particular item and that they should refresh and try again.

Appendix A References

[1] David Hill's WebLog. (2013) http://blogs.msdn.com/b/dphill/archive/2009/01/31/the-viewmodel-pattern.aspx. [Online]. http://blogs.msdn.com/b/dphill/archive/2009/01/31/the-viewmodel-pattern.aspx