1.12.2013

CENG 490

SOFTWARE DESIGN DESCRIPTIONS DOCUMENT

Group 10 | The Cereal Killers

Members and Signatures

Member	**Signature**	**Date**
Yaşar Barış ULU		1.12.2013
Kemal Çağın GÜLŞEN		1.12.2013
Mert ERGUN		1.12.2013
Kerem GÖKHAN		1.12.2013

Change History

Date	Revision	Responsible Party	Comment
13.11.2013	0.2	K.Çağın GÜLŞEN	Initial Version
24.11.2013	0.7	Y. Barış ULU E. Kerem GÖKHAN	Detailed Version
1.12.2013	1.0	Mert ERGUN	Final Version

Preface

This document contains the software design descriptions of Event-Based Social Network Project. This document is prepared according to the "IEEE Standart for Information Technology Software Design Descriptions – IEEE Std 1016 - 2009" document.

This Software Design Documentation provides a complete description of all the system design and views of the Event-Based Social Network Project.

The first and second sections of this document include the purpose, scope of the document and the references used though out the document.

The third section includes the conceptual model for software design descriptions.

The fourth section contains design description information content and lastly, the fifth section includes design viewpoints of the Event-Based Social Network Project.

Table of Contents

Table of Figures

1. Introduction

This Software Design Descriptions document is written by Kemal Çağın Gülşen, Mert Ergun, Yaşar Barış Ulu and Kerem Gökhan for the Event-Based Social Network Project. This document provides the details for how the Event-Based Social Network software should be built. The details are represented by using graphical notations such as use case models, sequence diagrams, class diagrams, object behavior models, and other supporting requirement information. All models and diagrams are built with UML Tools and represented as UML diagrams.

1.1. Scope

The software that is going to be implemented is a cross-platform mobile application. Users will be able to create, share, promote and join events. Furthermore users will be able to see events nearby, trending events and promoted events. Aim of this project is to provide an application that helps people with all kinds of events. The end product will serve users with various types of events such as academic, entertainment, commercial, technological, leisure activities. This project will be implemented by four people between October 2013 and June 2014.

1.2. Purpose

The purpose of this software design descriptions document is to describe how the software will be structured to satisfy the requirements of the Event-Based Social Network. This document will explain the design details of the system.

1.3. Intended audience

The intended audience of this software design descriptions document is both the developers who will build the system and the stakeholders.

1.4. References

[1] IEEE. IEEE Std 1016-2009 IEEE Standard for Information Technology – System Design – Software Design Descriptions. IEEE Computer Society, 2009.

[2] StarUML 5.0 User Guide. (2005). Retrieved from http://staruml.sourceforge.net/docs/userguide(en)/toc.html

2. Definitions

Terms	**Definitions**
GUI	Graphical User Interface
IEEE	Institute of Electrical and Electronics Engineers
SDD	Software Design Description
SRS	Software Requirements Specification
UML	Unified Modeling Language
Args	Arguments
Use Case Diagram	It represents user's interaction with the system.
Deployment Diagram	It models the physical deployment of artifacts on nodes.
Component Diagram	It is used to illustrate the structure of arbitrarily complex systems.
State Diagram	It is used to define the behavior of the system
Activity Diagram	Activity diagrams are graphical representations of workflows of stepwise activities and actions.
Class Diagram	It describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

This document has five sections to state the detailed design of the project. In the following section, which is the third section, conceptual model for software design descriptions will be given. At fourth section, design description information content will be described and finally design viewpoints will be explained.

3. Conceptual model for software design descriptions

Basic terms, concepts and context of SDD will be given in this part of the document.

3.1. Software design in context

In Event-Based Social Network project, object oriented approach will be used as a design method. Hence, it will be easier to implement the project and add possible future features. Since this project is a social network application, this property is critically important. Furthermore multi-layered system architecture will be used. Database, business and client layers will help modularity and adaptability of the software. With object oriented design and multi-layered architecture, portability and integrability between components will be improved.

3.2. Software design descriptions within the life cycle

3.2.1. Influences on SDD preparation

The key software life cycle product that drives a software design is typically the software requirements specification (SRS) document. SRS document captures the software requirements (product perspective, functional and non-functional requirements and interface requirements) that will drive the design and design constraints to be considered or observed.

3.2.2. Influences on software life cycle products

The Software Design Descriptions document influences the content of several major software life cycle work products. During the preparation of SDD and during the implementation stage of the project requirements may change. Hence SDD influences will lead to requirements changes. Besides, SDD will influence test plans and test documentation of the Event-Based Social Network project.

3.2.3. Design verification and design role in validation

Test cases will be prepared after the development phase. Verification of the software will be tested with these test cases and all parts will be evaluated. Success of the software system will be determined with test cases. After the test results, Validation of the software will be checked if requirements of the system are fulfilled or not.

4. Design description information content

4.1. Introduction

Software Design Description of Event-Based Social Network project identifies how this system will be implemented and designed. Event-Based Social Network project has a modular object-oriented structure. Furthermore a qualified interface will be designed in a way that system will represent events of the simulation successfully.

4.2. SDD identification

This SDD contains these features:

-Summary
-Glossary
-Change history
-Date of issue and status
-Scope
-Issuing organization
-Authorship (responsibility or copyright information)
-References
-Context
-One or more design languages for each design viewpoint used
-Body

4.3. Design stakeholders and their concerns

Design stakeholders of Event-Based Social Network project is the developer team of the system and their advisors. Design concern of the stakeholders is implementing the project in a modular structure according to SDD document. End product has to contain design features that are described at SDD document. Modular approach is essential for the project since there will be multi types of clients; namely, web client and mobile clients.

4.4. Design views

UML is used for representing diagrams of views. In order to have required UML knowledge, one can use this website http://tutorialspoint.com/uml/ . It is highly recommended.

4.5. Design viewpoints

In this Software Design Descriptions document; context, composition, logical, dependency, state dynamics and interaction viewpoints are given. These viewpoints are explained in details with required UML diagrams.

4.6. Design rationale

In this project, design choices are made according to performance concerns and integrability of the system. System has to be designed in a way that future models and features can be added and current models can be changed and updated independently. Stakeholders may have and request further requirements, therefore system parts have to be modular. Developers of the system has to document development process and use comments in their code frequently, so that in the future other developers may understand code and the structure of the system.

4.7. Design languages

In this project, Unified Modeling Language (UML) is selected as a part of design viewpoint and it will be used for clarifying design viewpoints.

5. Design Viewpoints

5.1. Introduction

In this part, six main design viewpoints will be explained in detail.

- Context Viewpoint
- Composition Viewpoint
- Logical Viewpoint
- Dependency Viewpoint
- Interaction Viewpoint
- State Dynamics Viewpoint

During this section, UML diagrams will be used to increase understandability.

5.2. Context Viewpoint

Our software context viewpoint shows the functionalities between user and system provided by a design. There are two main functionalities which are user and system functionalities in the software. There are also sub functions of these main functions. The context is defined by users which interact with the software.

5.2.1. User Use Case Diagram

The user has the following two main functionality sets.

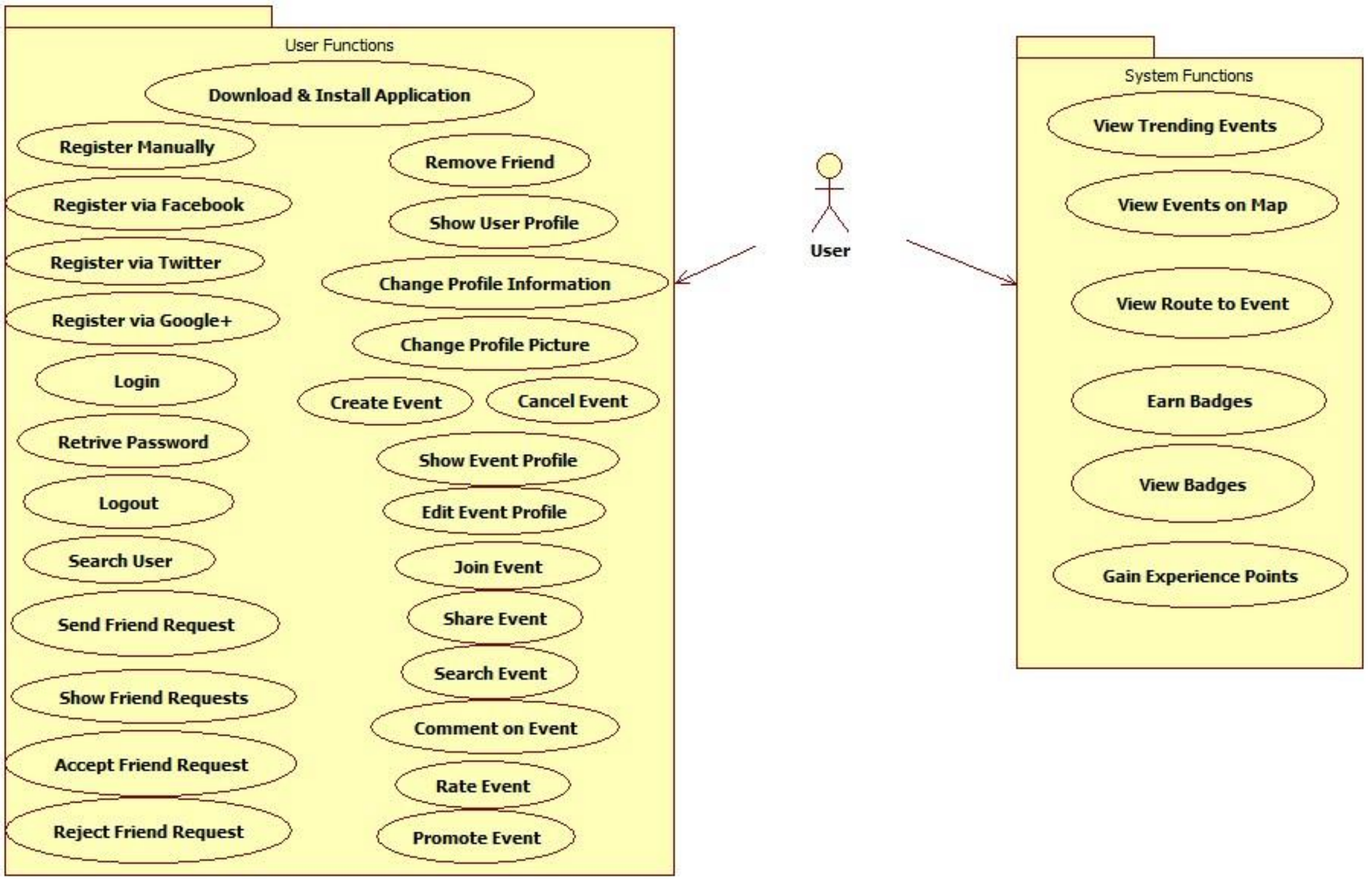


Figure 1: Use Case Diagram

5.3. Composition Viewpoint

Our system is composed of five components which are Database Server, Web Service, Computer Client, Android Client, Windows Phone Client and iOS Client. Database Server is placed at lower level of system. Web Server is connected between database server and client side. Client components basically consist of computers and smartphones. Also smartphones are divided into different types according to their operating systems.

5.3.1. Deployment Diagram

Deployment diagram of our system is shown below.

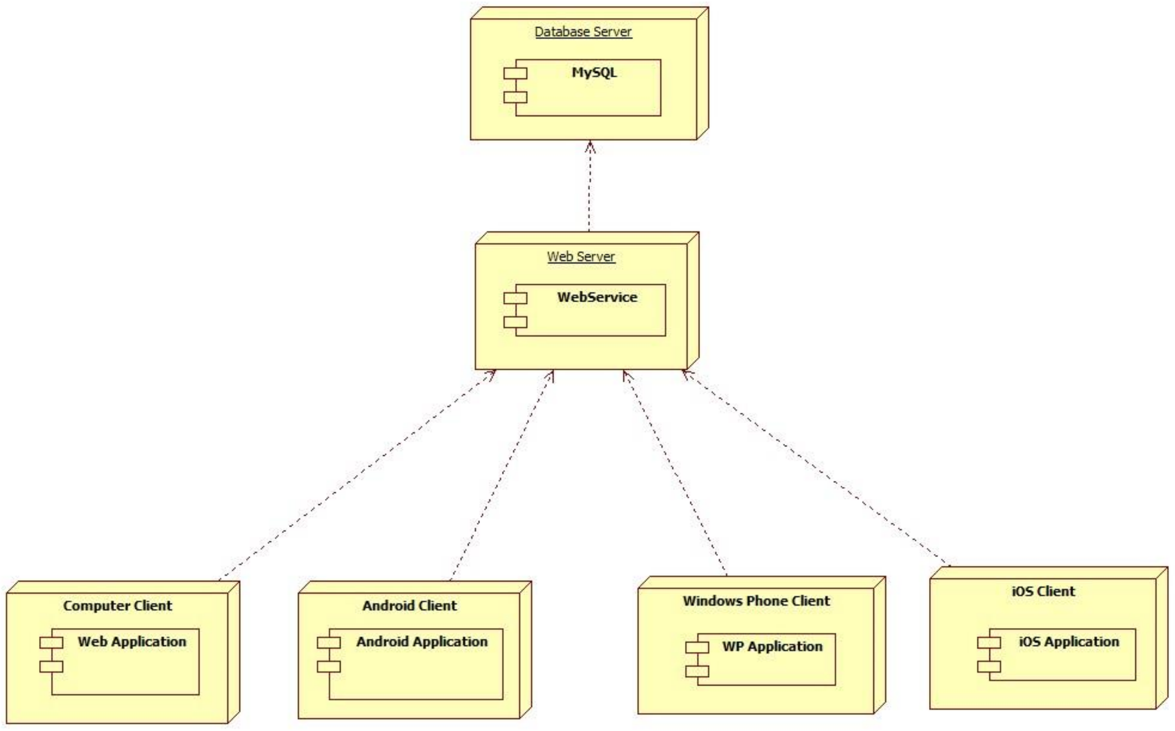


Figure 2: Deployment Diagram

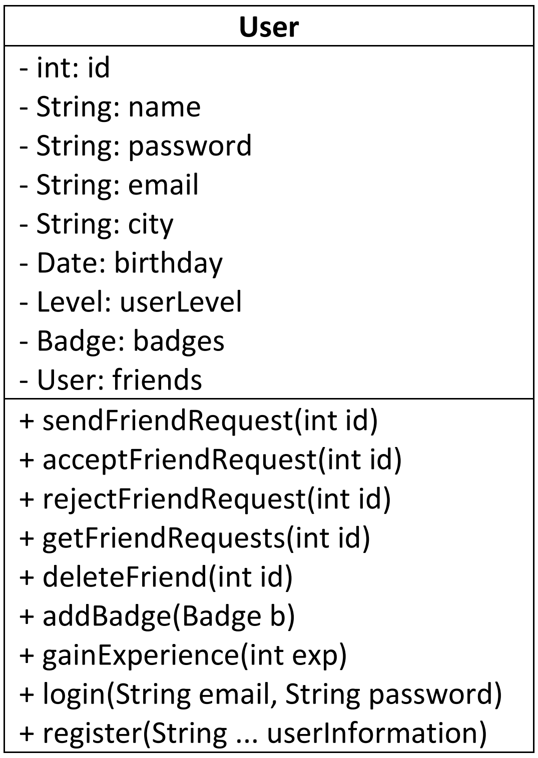
5.4. Logical Viewpoint

In this part of SDD, the classes which we will use in our project and the relationship between classes are explained in detail. Firstly, all classes are explained with class diagram separately. After that the relationship between these classes are shown.

There are five classes which are User Class, Event Class, Location Class, Level Class and Badge class. In this section these classes will be explained separately.

5.4.1. User Class

User class will handle user related data operations of the system and contain all user information with badge and level information.



User

- int: id
- String: name
- String: password
- String: email
- String: city
- Date: birthday
- Level: userLevel
- Badge: badges
- User: friends

+ sendFriendRequest(int id)
+ acceptFriendRequest(int id)
+ rejectFriendRequest(int id)
+ getFriendRequests(int id)
+ deleteFriend(int id)
+ addBadge(Badge b)
+ gainExperience(int exp)
+ login(String email, String password)
+ register(String ... userInformation)

Name	Type/Return Value Type	Visibility	Definition
id	int	Private	Defines the unique id value of the user.
name	String	Private	Defines the name string of the user.
password	String	Private	Defines the encoded password string of the user.
email	String	Private	Defines the email address of the user as a string.
city	String	Private	Defines the city name of the user as a string.
birthday	Date	Private	Defines the birthdate of the user as a Date object.
userLevel	Level	Private	Defines the current level data of the user as a Level object.
badges	Badge	Private	Defines the badges the user has as a list of Badge object.
friends	User	Private	Defines the friends of the user as a list of User object.
sendFriendRequest()	bool	Public	Sends a friend request to a user.
acceptFriendRequest()	bool	Public	Accepts a friend request.
rejectFriendRequest()	bool	Public	Rejects a friend request.
getFriendRequests()	User[]	Public	Gets the list of friend requests.
deleteFriend()	bool	Public	Deletes a friend from friends list.
addBadge()	bool	Public	Adds a badge to the badges list.
gainExperience()	bool	Public	Increase the experience of the user.
login()	bool	Public	Sign-in the user into the system.
register()	bool	Public	Sign-up the user into the system.

5.4.2. Event Class

Event class will handle event related data operations of the system and contain all event information with location, owners and attendees information. This class will be updated at the later stages of the project.

Event
- int: id - Location: eventLocation - User: owners - String: description - bool: isPublic - User: usersGoing

+ createEvent(User u, String ... eventInformation)
+ addAttendee(User u)
+ getAttendees()
+ shareEvent(User u)
+ addOwner(User u)
+ promoteEvent(User u)
+ commentOnEvent(User u, String message)
+ rateEvent(User u, int rate)
+ InviteFriends(User[] friends)

Name	**Type/Return Value Type**	**Visibility**	**Definition**
id	int	Private	Defines the unique id value of the event.
eventLocation	Location	Private	Defines the location of the event as a Location object.
owners	User[]	Private	Defines the owner users of the event as a list of User object.
description	String	Private	Defines the description text of the event as a string.
isPublic	bool	Private	Defines the privacy level of the event: public or private.
usersGoing	User	Private	Defines the list of users which are attendees.
createEvent()	bool	Public	Creates a new event to the system using given information.
addAttendee()	bool	Public	Adds a new attendee to the event.
getAttendees()	User[]	Public	Returns the list of users which are attending the event.
shareEvent()	bool	Public	Shares the event with a user.
addOwner()	bool	Public	Adds a new owner to the event.
promoteEvent()	bool	Public	Promotes the event.
commentOnEvent ()	bool	Public	Adds a comment text to the event from given user.
rateEvent()	bool	Public	Calculates the new rate value of the event after a new rate from given user.
InviteFriends()	bool	Public	Invites the event with given list of users.

5.4.3. Location Class

Location class will handle location related data operations of the system and contain all location information.

Location
- int: id - LatLong: place

+ getLatLong(int id)
+ getCity()
+ getNearbyEvents(double range)

Name	**Type/Return Value Type**	**Visibility**	**Definition**
id	int	Private	Defines the unique id value of the location.
place	LatLong	Private	Defines the location coordinate as a LatLong object.
getLatLong()	LatLong	Public	Returns the LatLong object.
getCity()	String	Public	Returns the city using LatLong data.
getNearbyEvents()	Event[]	Public	Returns the list of events within a given range.

5.4.4. Level Class

Level class will handle level related data operations of the system and contain all level information.

Level
- int: levelValue - int: experience - String: name - int: experienceForNextLevel

+ addExperience(int exp)
+ getExperience()

Name	Type/Return Value Type	Visibility	Definition
levelValue	int	Private	Defines the current level.
experience	int	Private	Defines the current experience.
name	String	Private	Defines the title of level.
experienceForNextLevel	int	Private	Defines the experience needed for the next level.
addExperience()	bool	Public	Adds experience to the current experience.
getExperience()	int	Public	Returns the current experience.

5.4.5. Badge Class

Badge class will handle level related data operations of the system and contain all badge information.

Badge
- int: id - String: name - Image: icon - int: experienceValue
+ getIcon(int id)

Name	Type/Return Value Type	Visibility	Definition
id	int	Private	Defines the unique id value of the badge.
name	String	Private	Defines the name of the badge as a string.
icon	Image	Private	Defines the icon of the badge as an image.
experienceValue	int	Private	Defines the experience value of the badge.
getIcon()	Image	Public	Returns the icon of the badge as an image.

5.5. Dependency Viewpoint

In this viewpoint the relationships of interconnections and access among packages are explained in detail and with UML component diagram.

In our project we will use the three-tier system architecture will be used. The three-tier architecture provides the following benefits.

- Scalability: Each tier can scale horizontally. For example, we can load-balance the Presentation tier among three servers to satisfy more Web requests without adding servers to the Application and Data tiers.
- Performance: Because the Presentation tier can cache requests, network utilization is minimized; the load is reduced of Application and Data tiers. If needed, we can load-balance any tier.
- Availability: If the Application tier server is down and caching is sufficient, the Presentation tier can process Web requests using the cache.

Three-tier architecture is composed of three main packages. These are presentation tier, middle tier and data management tier.

- Presentation Tier: A layer that users can access directly, such as web page and client application.
- Middle Tier: This layer encapsulates the business such as business rules and data validation, domain concept, data access logic.
- Data Management Tier: The external data source to store the application data such as database server, mainframe or other legacy systems. The one we meet often today is database server.

In our system, we can divide into three basic packages. These are User Interface, Business Logic and Database. The explanation of these packages is below:

User Interface: It is top most module of our system. Basically its function is translating tasks and showing the results to the user. In User Interface package we have mainly two modules which are View and Controller.

- **View:** The view can be separated into two main modules according to usage. Actually, all of these have almost same features. The View renders a presentation of modeled data. We will use HTML5 technology for client application.
- **Controller:** The Controller handles requests from user. It is responsible for rendering back a response with the aid of Model module. Controller can be seen as managers taking care that all needed resources for completing a task are delegated to the correct workers. We will use Java programming language for this module.

Business Logic: It is the transition module of our system. The Business Logic coordinates the application, processes commands, makes logical decision and evaluation and performs calculations. It also moves and processes data between the modules, User Interface and Database in our system.

- **Model:** The model represents the part of our application that implements the business logic. It is responsible for the retrieving data and converting it into meaningful data for our application. This includes processing, validating, association or other tasks relate to handling data. We will use Java programming language for this module.

Database: It is the lowest module in our system. In this module the information is stored and retrieved from a database. The information is then passed back to the Business Logic for processing and then eventually back to the user. We will use MySQL for database package.

5.5.1. Component Diagram

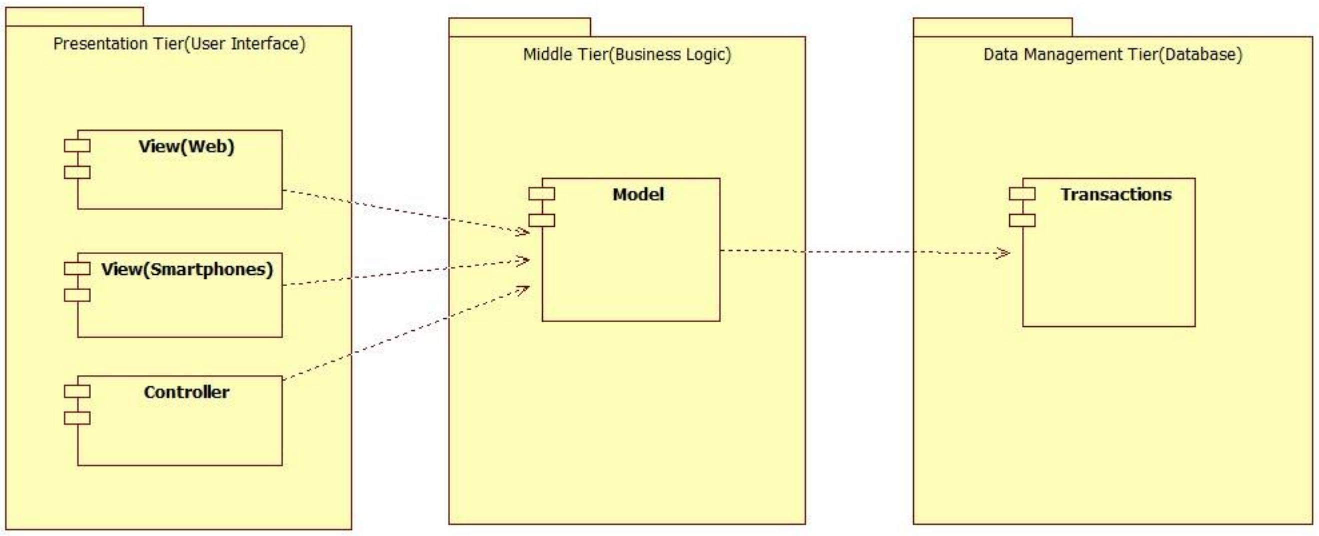

Presentation Tier(User Interface)
View(Web)
View(Smartphones)
Controller
Middle Tier(Business Logic)
Model
Data Management Tier(Database)
Transactions

Figure 3: Component Diagram

5.6. State Dynamics Viewpoint

State dynamics viewpoint explains system and user functionalities step by step. There are two state diagrams in our project. Although these state diagrams are similar to each other, they represent different systems. The prior corresponds to mobile application lifecycle and the latter is represents the web page client.

5.6.1. State Diagram Mobile

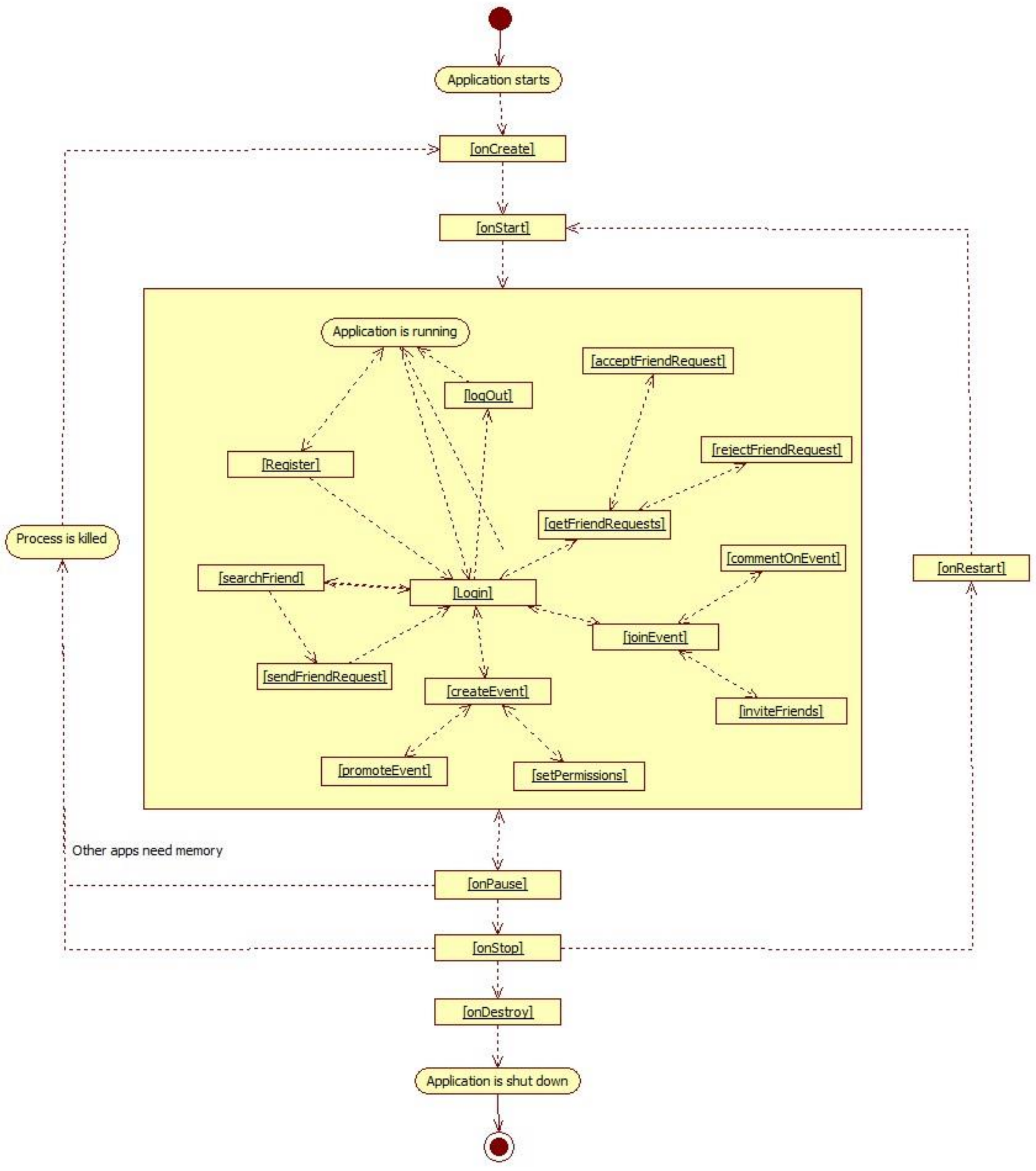


Figure 4: State Diagram Mobile

5.6.2. State Diagram Web

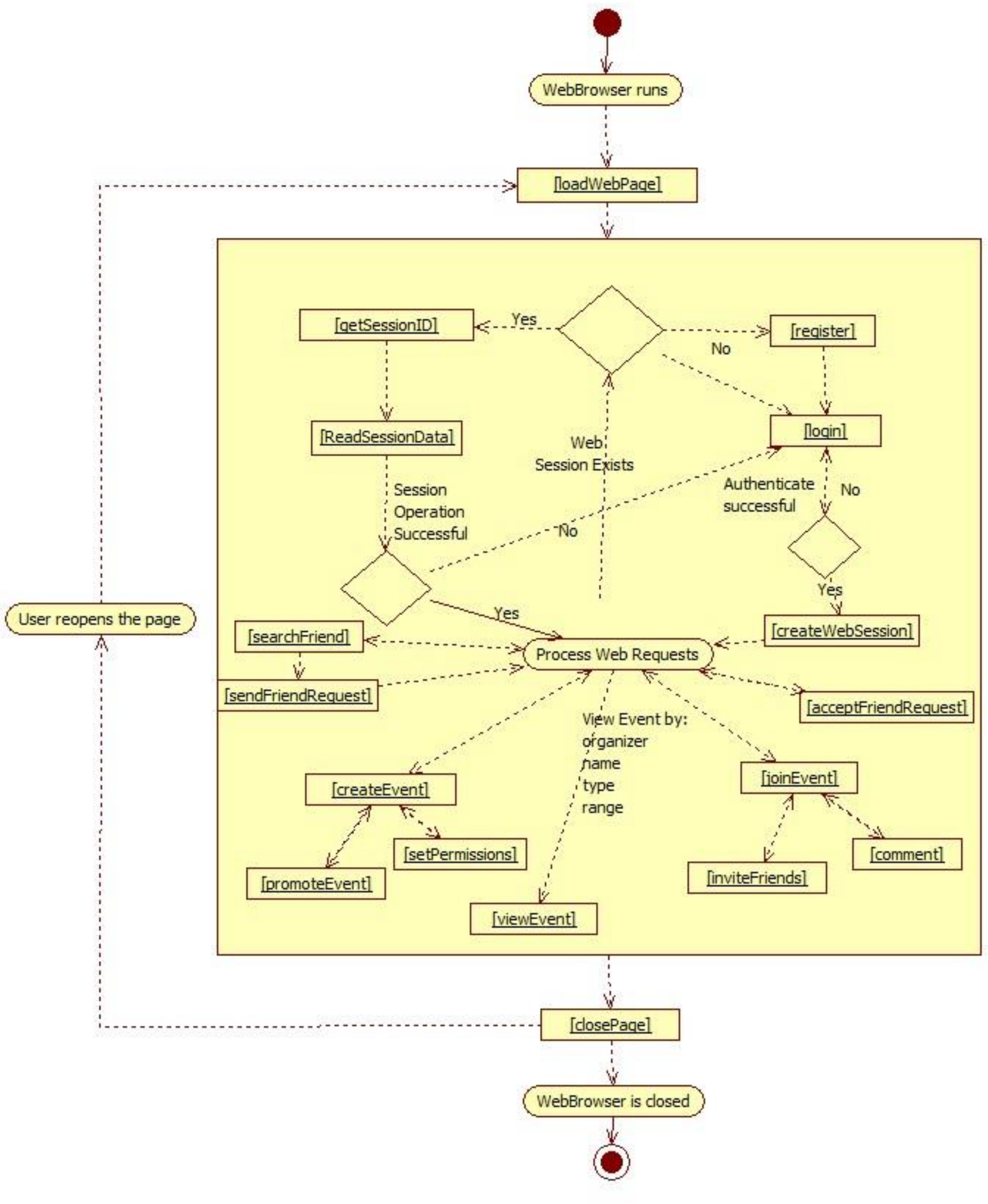


Figure 5: State Diagram Web

5.7. Interaction Viewpoint

Interaction viewpoint is provided through sequence diagrams. The purpose is to explain the main functionalities step by step. There exist seven sequence diagram to represent whole system.

5.7.1. User – Server Sequence Diagram 1

This sequence diagram represents not only represents the register and login functionalities but also represents main interaction between user and server.

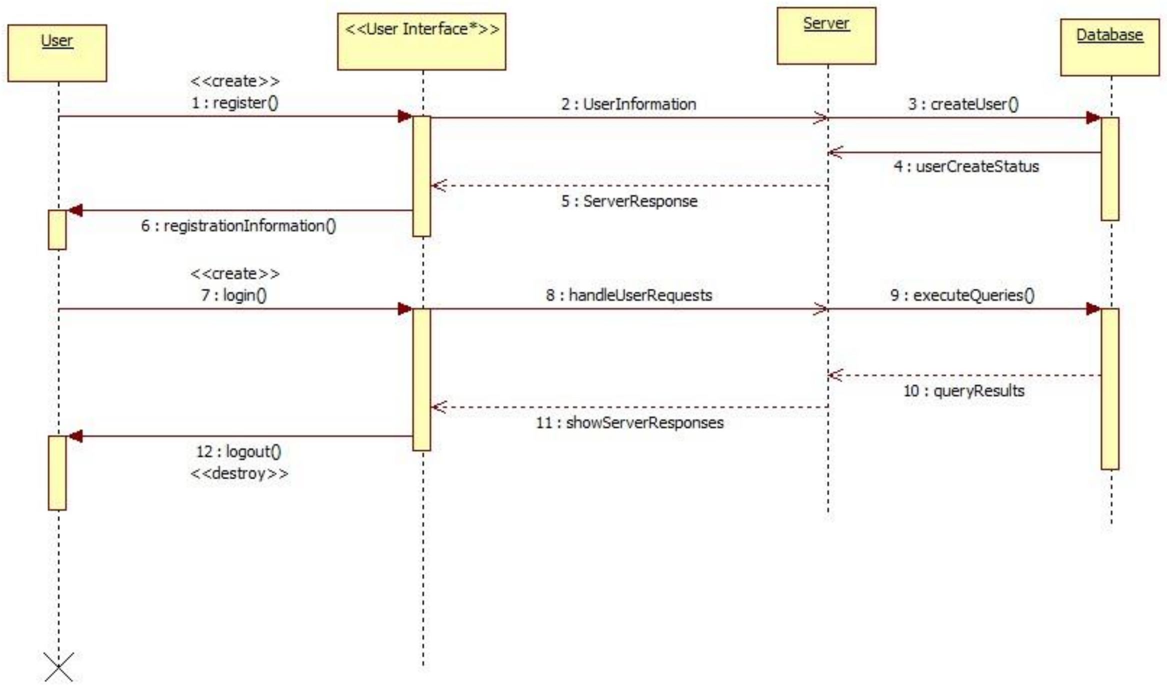


Figure 6: User - Server Sequence Diagram 1

5.7.2. User – Server Sequence Diagram 2

This diagram represents other functionalities between user and server.

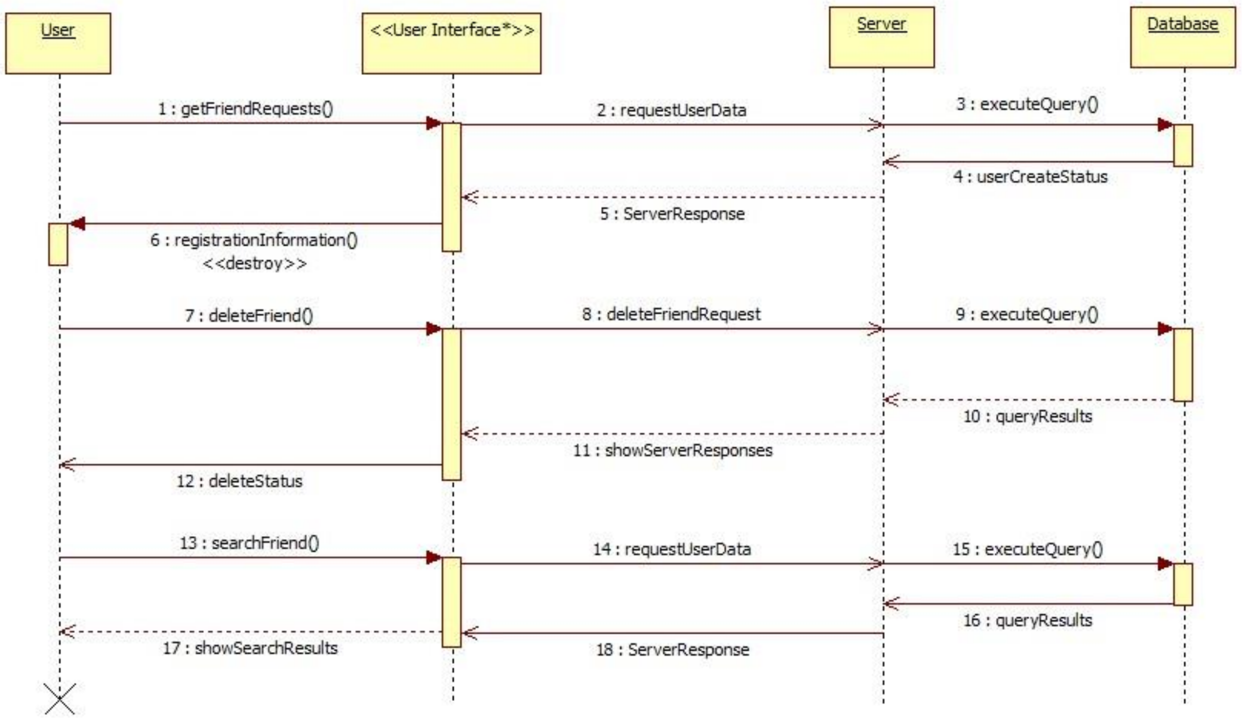


Figure 7: User - Server Sequence Diagram 2

5.7.3. Event – Server Sequence Diagram 1

This diagram represents interaction between Event and Server. Sequence of actions stated below will be triggered by the server with a certain frequency.

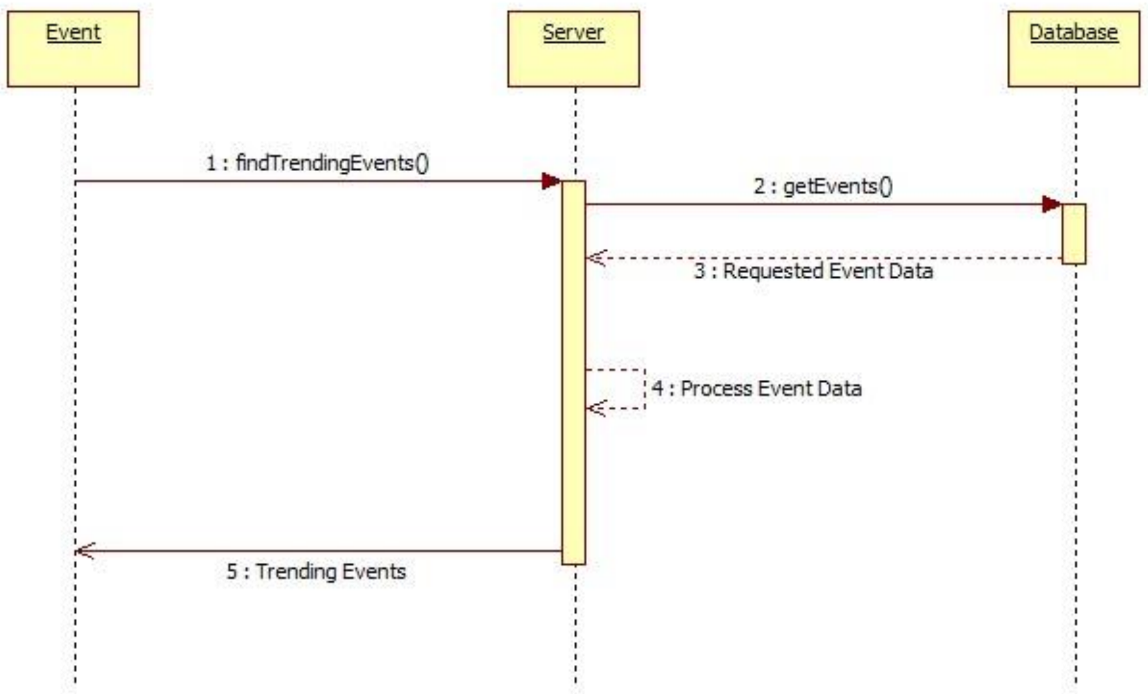


Figure 8: Event – Server Sequence Diagram 1

5.7.4. User – Event Sequence Diagram 1

This diagram represents interaction between User and Event.

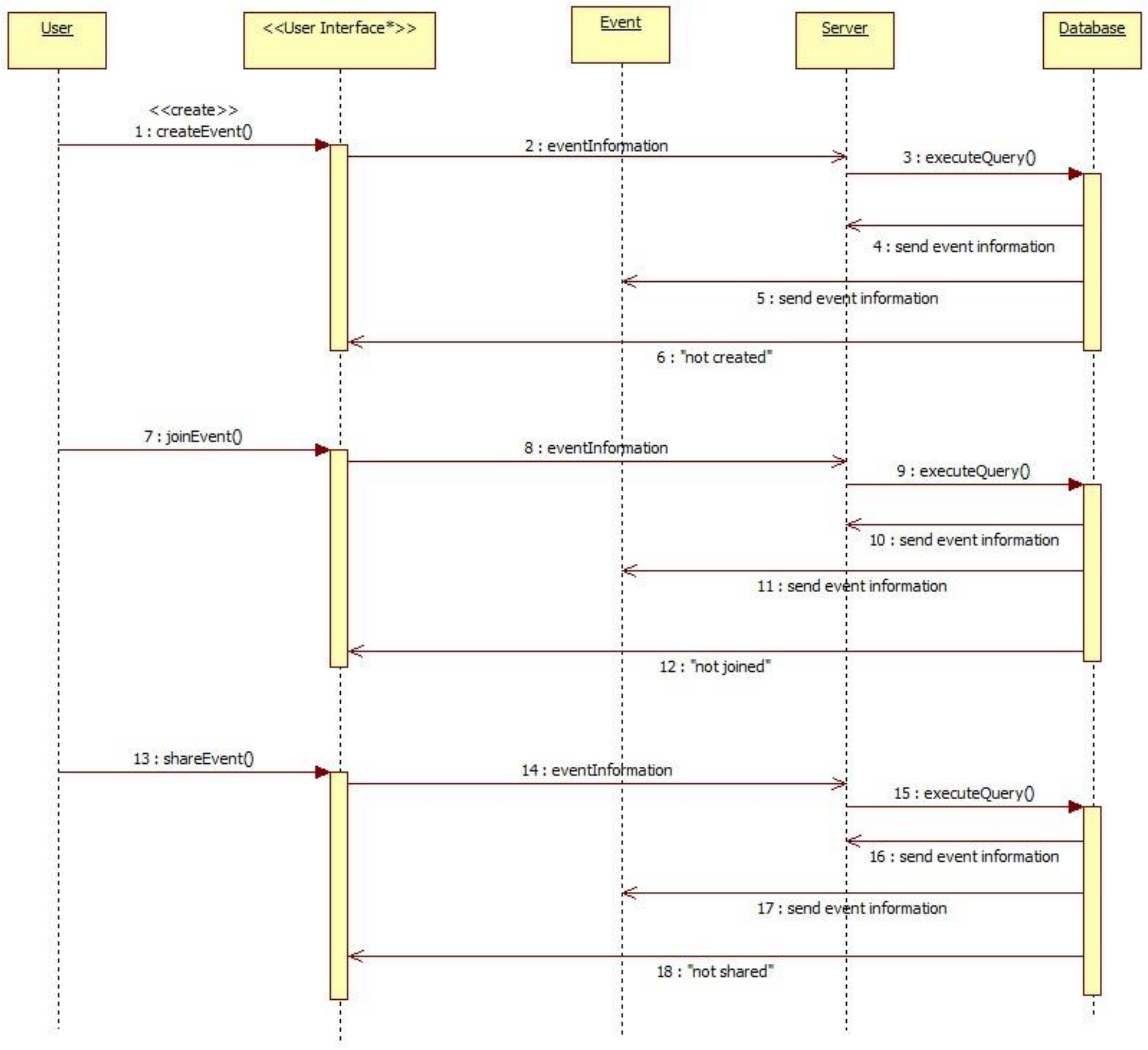


Figure 9: User – Event Sequence Diagram 1

5.7.5. User – Event Sequence Diagram 2

This diagram represents remaining interaction between user and event.

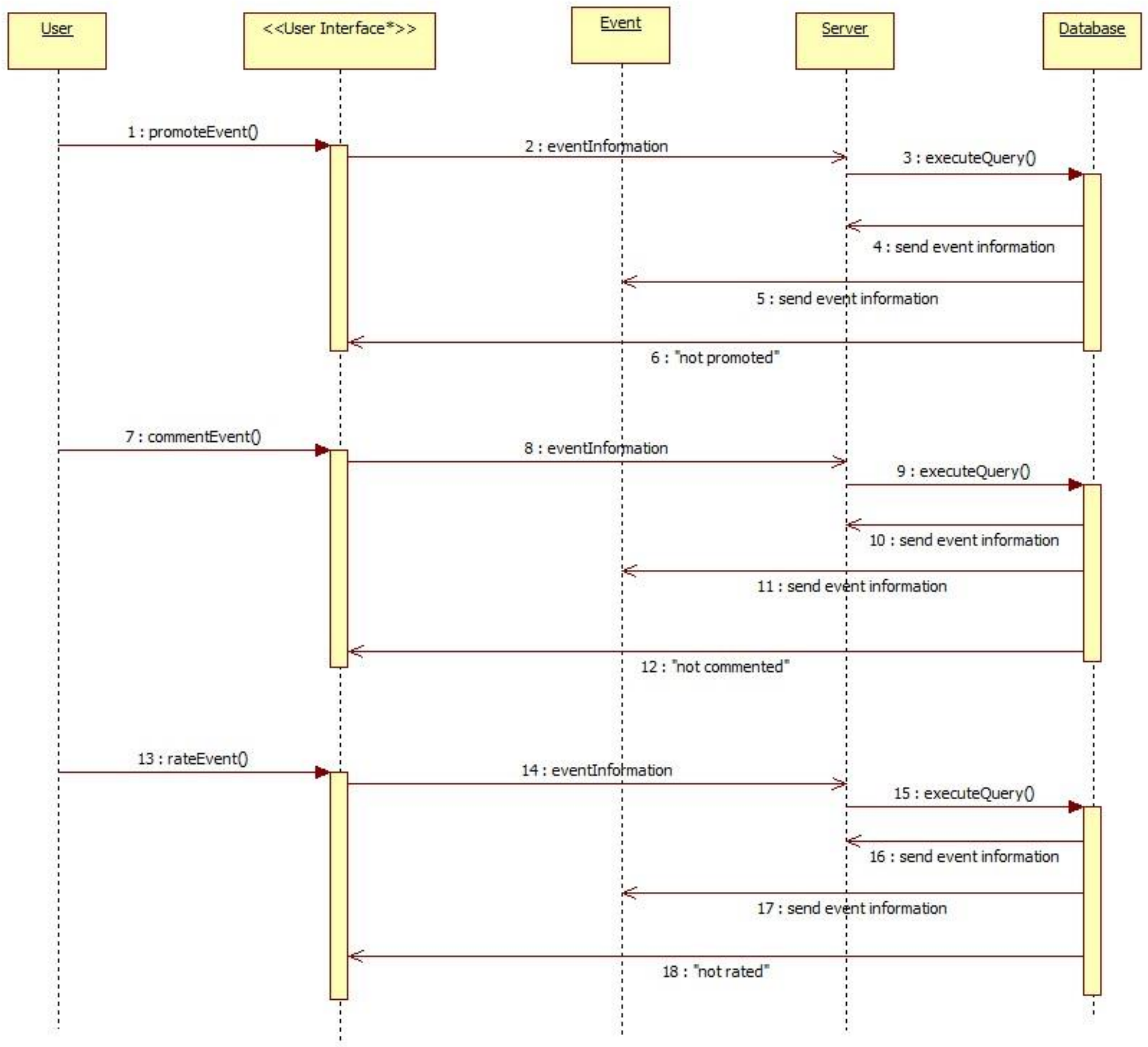


Figure 10: User - Event Sequence Diagram 2

5.7.6. User – Level Sequence Diagram

This diagram represents interaction between user and level.

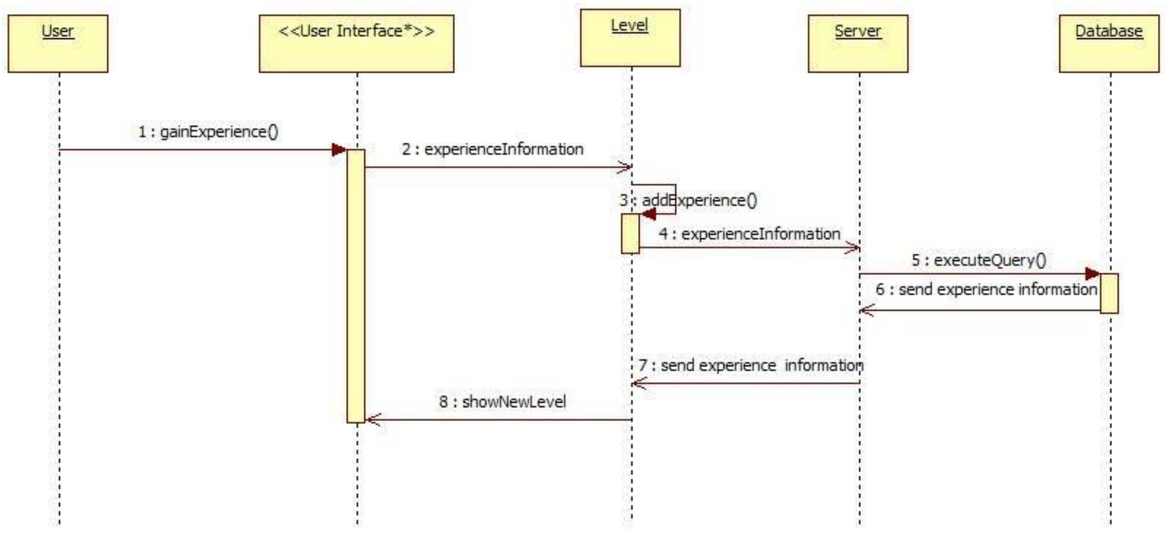


Figure 11: User - Level Sequence Diagram

5.7.7. User – Badge Sequence Diagram

This diagram represents interaction between user and badge.

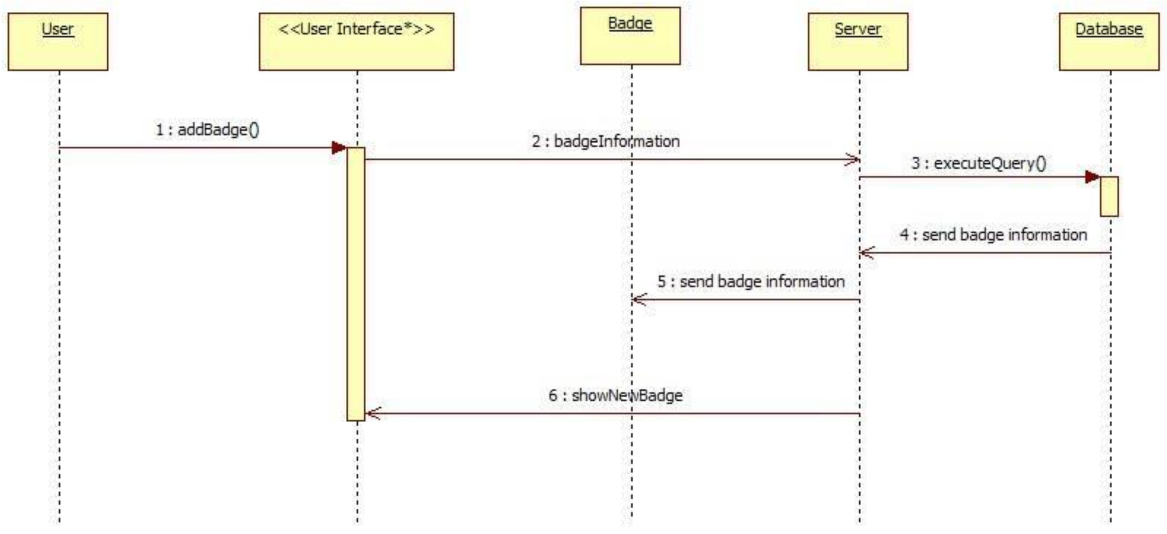


Figure 12: User - Badge Sequence Diagram

6. Conclusion

This Software Design Descriptions document is prepared for giving detailed design information of Event-Based Social Network Project. Furthermore, basics of data design, modules and design viewpoints of the system are described. The tools and libraries that will be used while designing and implementing the system are also provided.