WeCalEvent

Design Document

Software Engineering 2

Authors:

Mert Ergun

Mohamed Mehdi Kaabi

Nazrin Javadova

Milano – 2014

LaTeX

Contents

1 Overview

1.1 Scope

This document is prepared considering requirements analysis and specification document of MeteoCal Project which is published before. This document covers software design stage of MeteoCal Project and it also contains important information relevant to software implementation. System structure and the components of the project is also described in this document.

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 MeteoCal. 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.

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

3 Conceptual model for software design descriptions

3.1 Software design in context

In MeteoCal 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 DD preparation

The key software life cycle product that drives a software design is typically the requirements analysis and specification (RASD) document. RASD 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 document influences the content of several major software life cycle work products. During the preparation of DD and during the implementation stage of the project requirements may change. Hence DD influences will lead to requirements changes. Besides, DD will influence test plans and test documentation of the MeteoCal 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 Document of MeteoCAl project identifies how this system will be implemented and designed. MeteoCal 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 DD identification

This DD 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 MeteoCal 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 Design Document. End product has to contain design features that are described at Design 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

In this document contextual, composition, interface, logical, interaction and state dynamics view will be explained in next sections. Detailed description and diagrams about these views will clarify them. Each view is given with its corresponding viewpoint.

4.5 Design Viewpoints

Software design description identifies context, composition, interface, logical, interaction and state dynamics viewpoints. Context viewpoint specifies the system boundaries and actors interacting with the system. Composition viewpoint identifies the system modules, components, frameworks and system repositories. Interface viewpoint explains the interaction between different software interfaces. Logical viewpoint includes the detailed description of data design and class diagrams. Interaction viewpoint gives the sequence of events in the system and the state dynamics viewpoint models the system as a state machine and shows the state transitions and conditions on these transitions.

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

<u>User functions:</u>

Register to the system: The user will be able to register into system manually to access opportunities of the application.

Login: The user will be able to login our application after successful registration.

Create event: The user will be able to create an event.

Invite users: The user will be able to invite other users to his events.

View calendars: The user will be able to view other userвЂ™s public calendars.

Answer event invitation: The user will be able to accept or decline invitations for events.

Modify calendar visibility: The user will be able to update the visibility of his calendar to public/private.

Import/export calendar: the user will be able to import or/and export calendars.

Search user: the user will be able to search user from users of the system.

Search event: The user will be able to search an event from all available event in the system.

Modify profile: The user will be able to update his profile data.

Modify event: The user will be able to update his events.

Delate event: The user will be able to delate any for his events.

System functions:

Notify users: the system will notify user in case of weather forecast updates.

Notify by email: The system will notify user by email in case of weather forecast updates.

Notify every 12h: The system will notify users when they login every 12 hours in case weather forecast updates.

Notify every 3 days: The system will notify users when they login every 3 days in case weather forecast updates.

Avoid conflicts event: The system warn the user when he create an event in the same time as another created event.

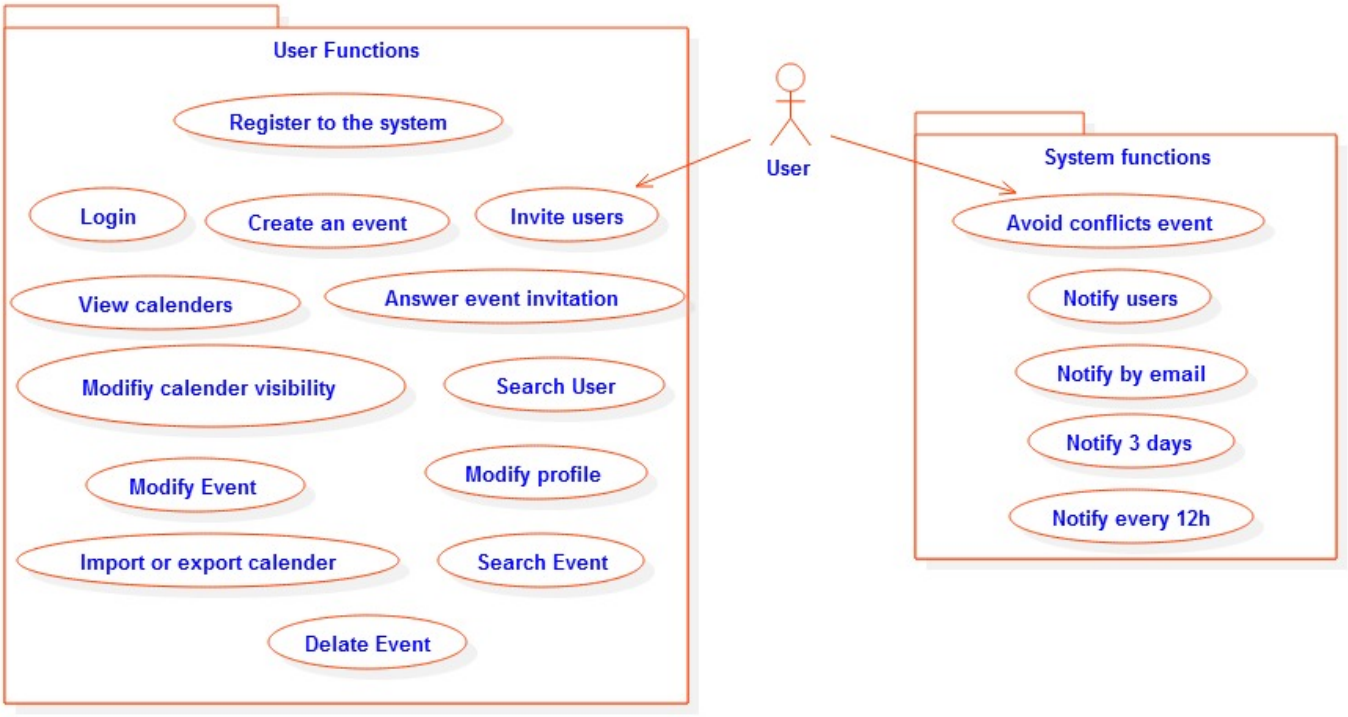


Figure 1: UseCases

5.3 Composition Viewpoint

Our system is composed of following components: HTML Client, HTTP Services (Servlet and JSF) EJB Container and MySQL Database. Database is placed at the lowest level of the system. HTML Client is basically the internet browser navigated to our application. Clients are connected to Enterprise Java Beans via Java Server Faces. And finally database transactions are handled by Eclipse Link framework though a JDBC connection.

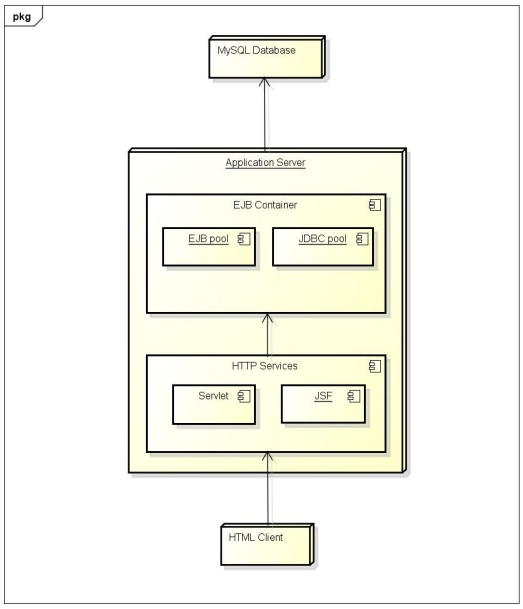


Figure 2: Deployment Diagram

MySQL Database: Database server provides database services to clients. There are some advantages of database server, these are:

- The database offers a single repository of data
- Having a single stored database makes it easier to maintain the accuracy and currency of the data
- A single database makes it easier to protect against loss of data due to hardware failure

For database server we have selected MySQL which is an open source system and from everyone to anyone can use it. MySQL is the most opted form of database because of cross platform operability.

Application Server: As an application server, Glassfish v4.1 and JAVA EE capabilities with Enterprise Java Beans will be used.

EJB comes with the following advantages: "The EJB specification intends to provide a standard way to implement the back-end 'business' code typically found in enterprise applications (as opposed to 'front-end' interface code). Such code addresses the same types of problems, and solutions to these

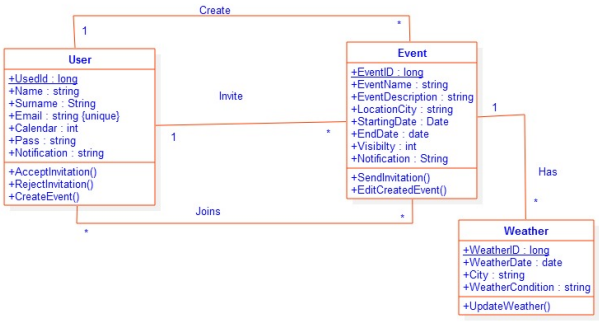
problems are often repeatedly re-implemented by programmers. Enterprise JavaBeans are intended to handle such common concerns as persistence, transactional integrity, and security in a standard way, leaving programmers free to concentrate on the particular problem at hand."

Client: The only client side in our project is HTML Client. HTML codes are generated by JSF framework included in the EJB service. In order to make User Interface development easier, PrimeFaces v5.1 library is added as a dependency to our project.

5.4 Logical Viewpoint

In this part of design document, 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 entity relationship diagram, which defines the database structure of our project, will be provided.

There are only three classes which are User, Event and weather. In the following sections, each class 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 invited participated and created events.

Name	Type/Return Value Type	Visibility	Definition
User_id	Long	Private	Unique id of the user
Name	String	Private	Name of the user
Surname	String	Private	Surname of the user
Email	String	Private	Unique email address of the user
Pass	String	Private	Password of the user account
Calendar	Integer	Private	Calendar visibility of the user
Notification	String	Private	Notifications specific to this user
acceptInvitation()	Boolean	Public	Accept invitation from an event
rejectInvitaion()	Boolean	Public	Reject invitation from an event
createEvent()	Boolean	Public	Create new event

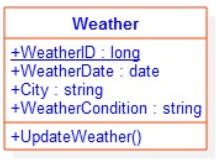
5.4.2 Event Class

Event class will handle event related data operations of the system and contains information about the event, invited or participating users and owner of the event.

Name	Type/Return Value Type	Visibility	Definition
Event_id	Long	Private	Unique id of the event
Event_name	String	Private	Name of the event
Event_description	String	Private	Description of the event
Location_city	String	Private	Location of the event
Starting_date	DateTime	Private	Starting date of the event
Ending_date	DateTime	Private	Ending date of the event
Visibility	Integer	Private	Visibility option of the event
Notification	String	Private	Notifications specific to this event
sendInvitation()	Boolean	Public	Send invitation to another user from created event
editCreatedEvent()	Boolean	Public	Edit the event previously created

5.4.3 Weather Class

Weather class will handle event related data operations of the system and contains information about the weather conditions in a specific location.



Name	Type/Return Value Type	Visibility	Definition
Weather_id	Long	Private	Unique id of the Weather
Weather_date	Date	Private	Date of the weather forecast
City	String	Private	Location of the weather forecast
Weather_condition	String	Private	Weather forecast
updateWeather()	Bool	Private	Updates the weather forecast

5.5 Dependency Viewpoint

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

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 only one module:

- **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.

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.

- **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.

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.

Figure 3: Component Diagram

5.6 State Dynamics Viewpoint

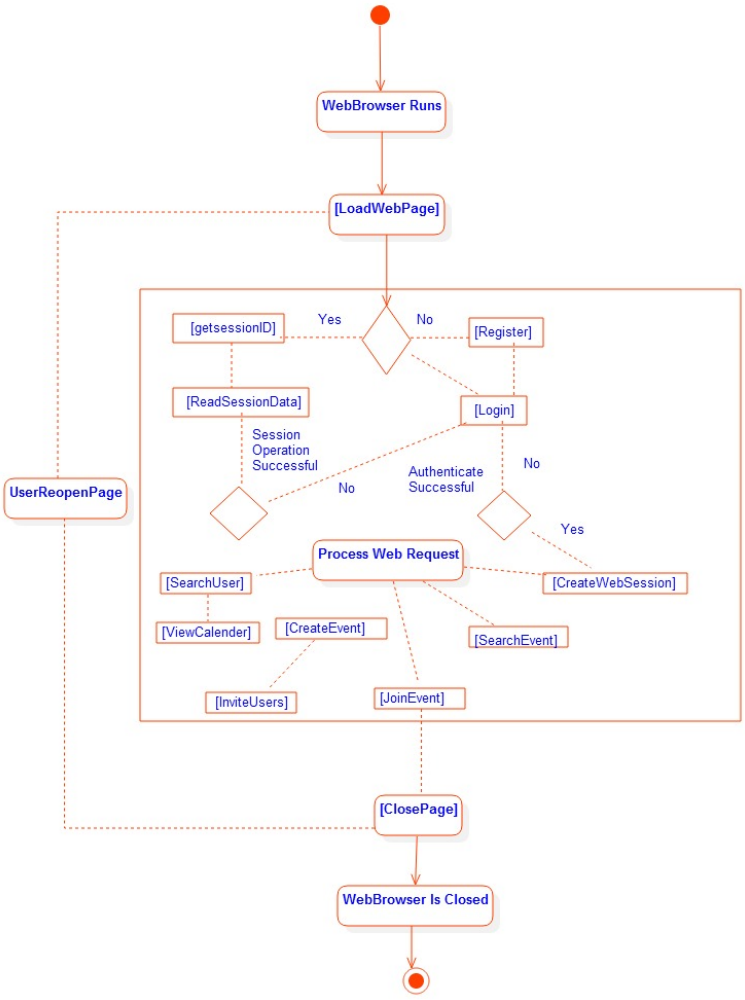


Figure 4: Activity Diagram

5.7 Interaction Viewpoint

5.7.1 Register to system

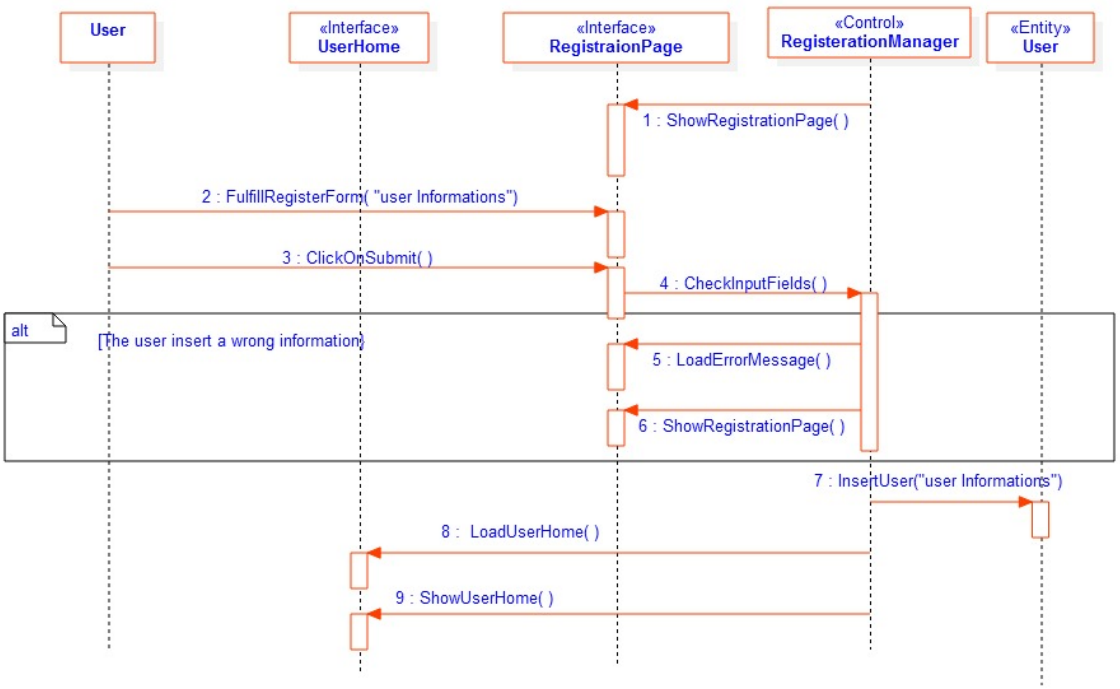


Figure 5: Register to system

5.7.2 Modify calendar visibility

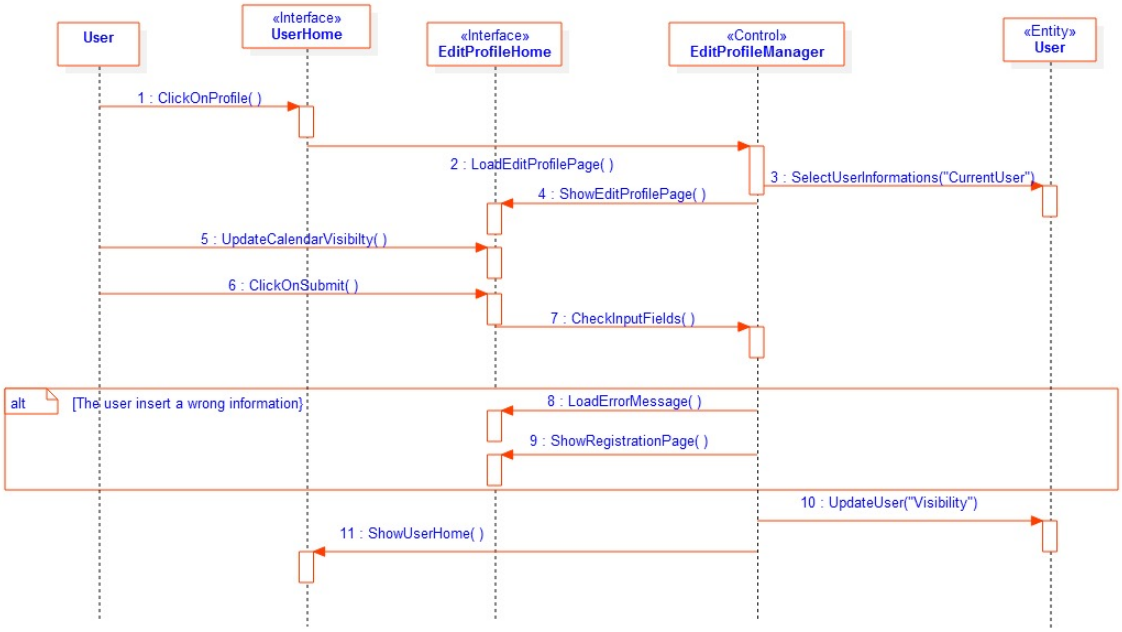


Figure 6: Modify calendar visibility

5.7.3 Import/Export

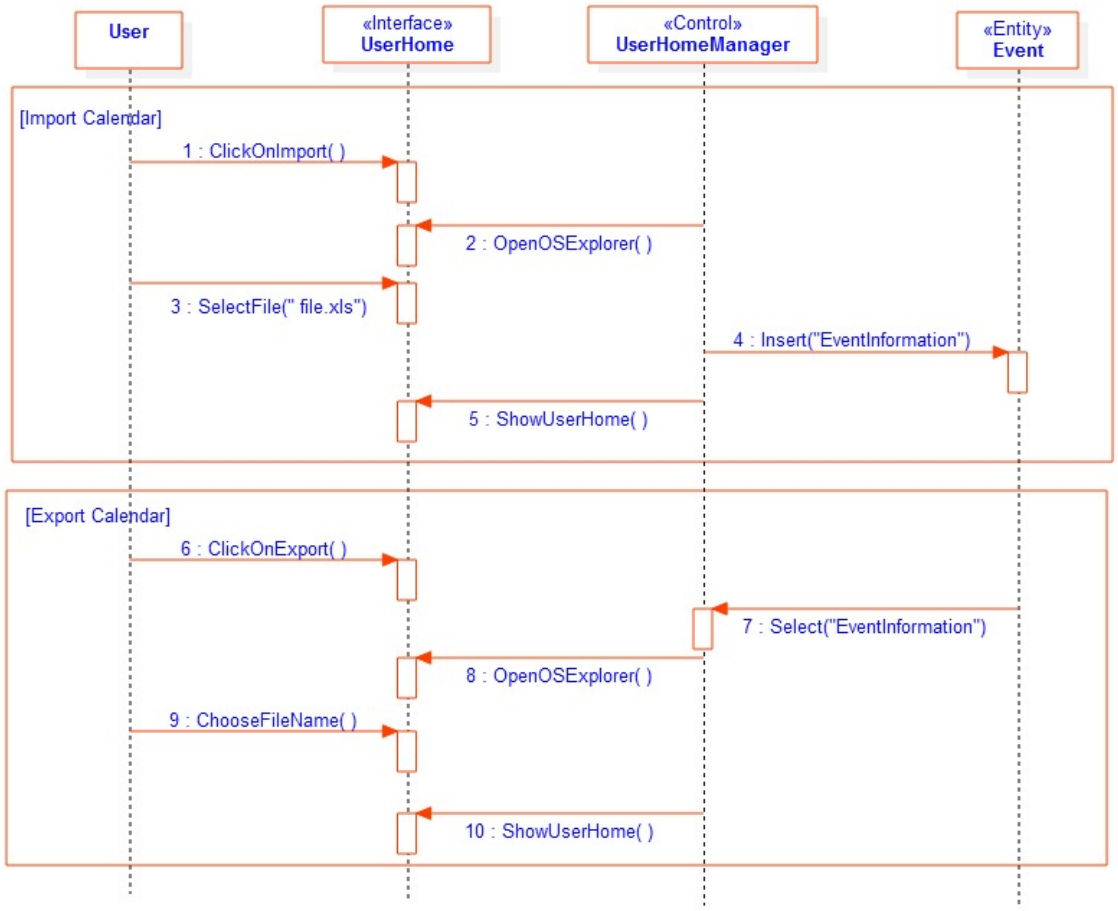


Figure 7: Import/Export

5.7.4 Search user

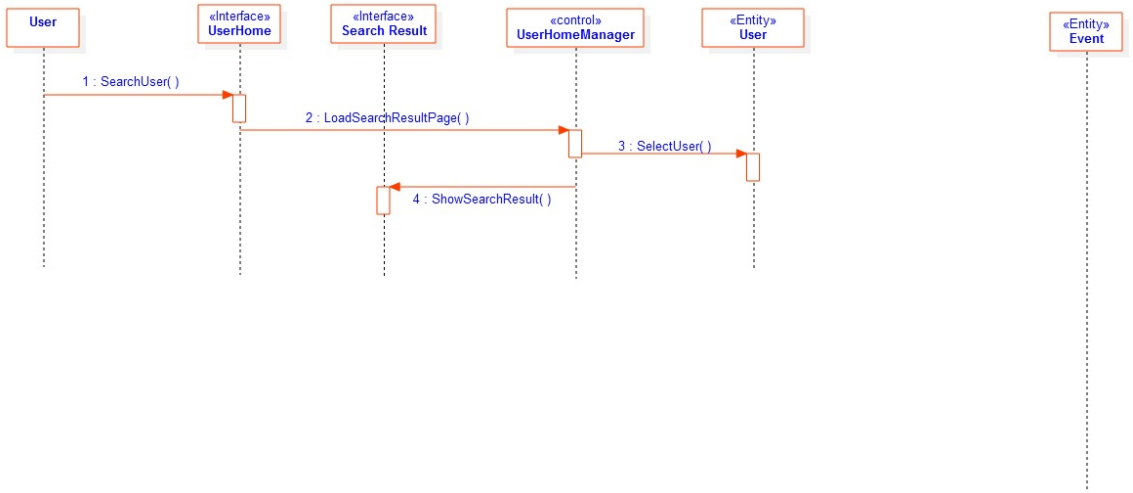


Figure 8: Search user

5.7.5 Search event

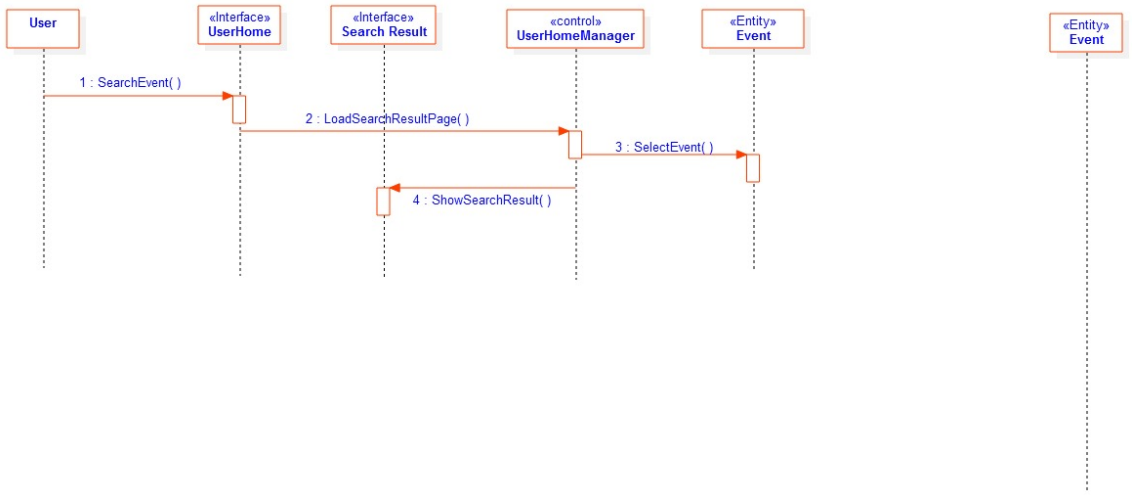


Figure 9: Search event

5.7.6 Login

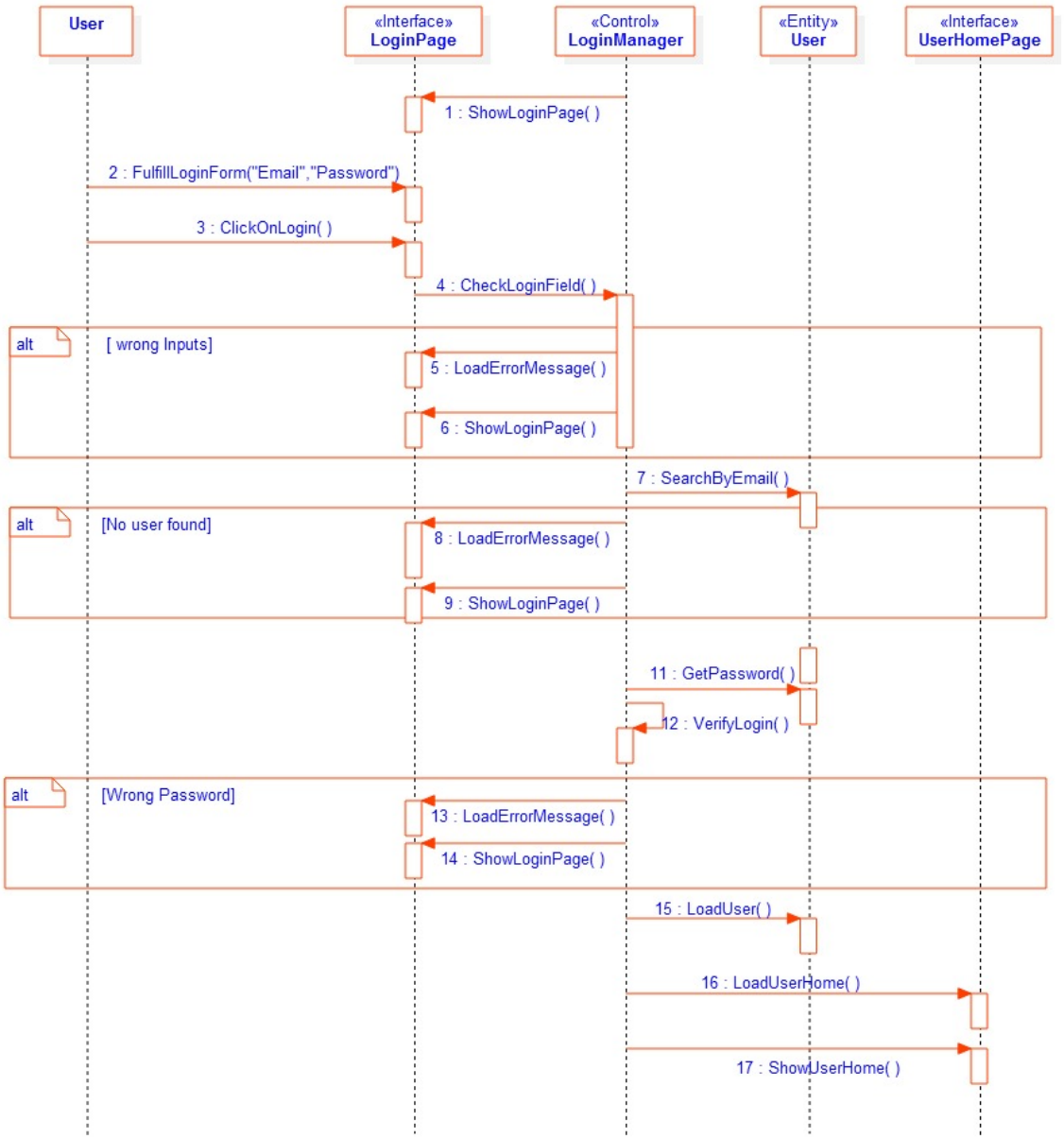


Figure 10: Login

5.7.7 Create event

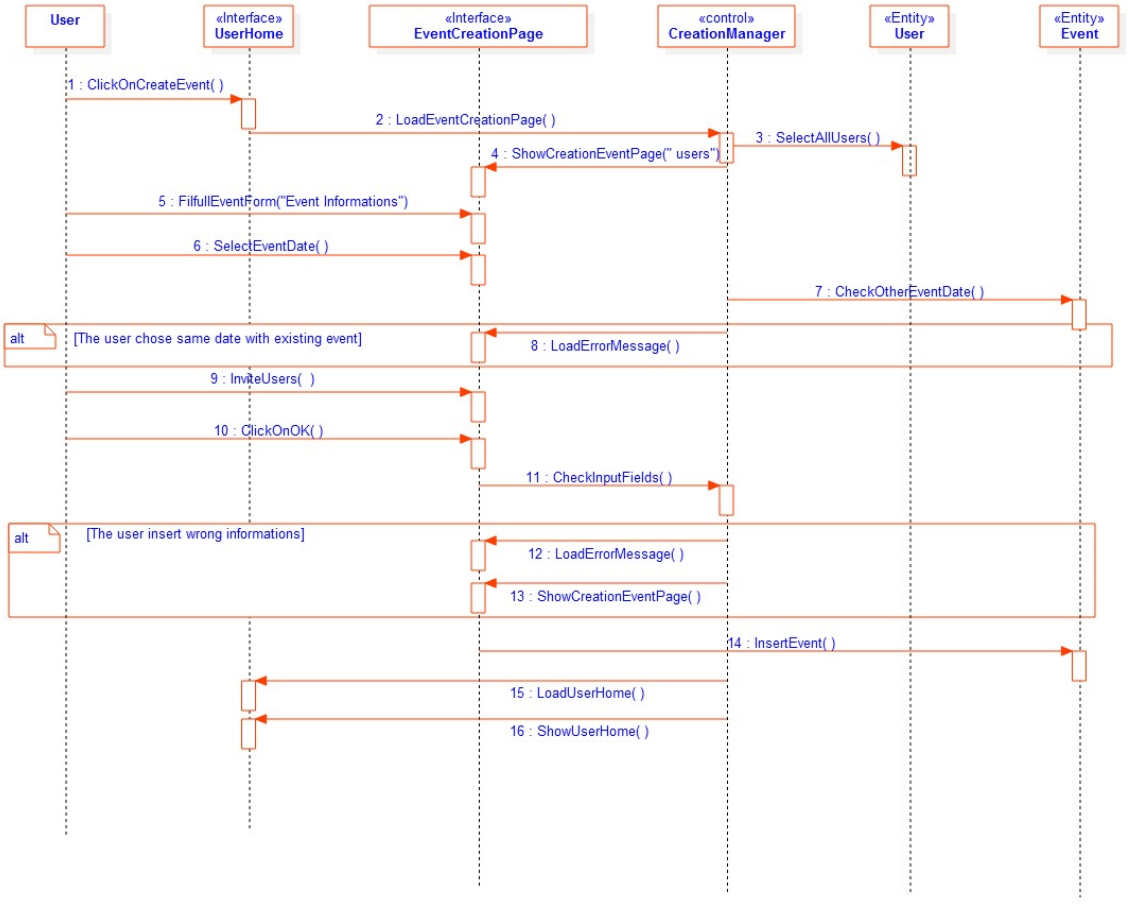


Figure 11: Create event

5.7.8 Modify event

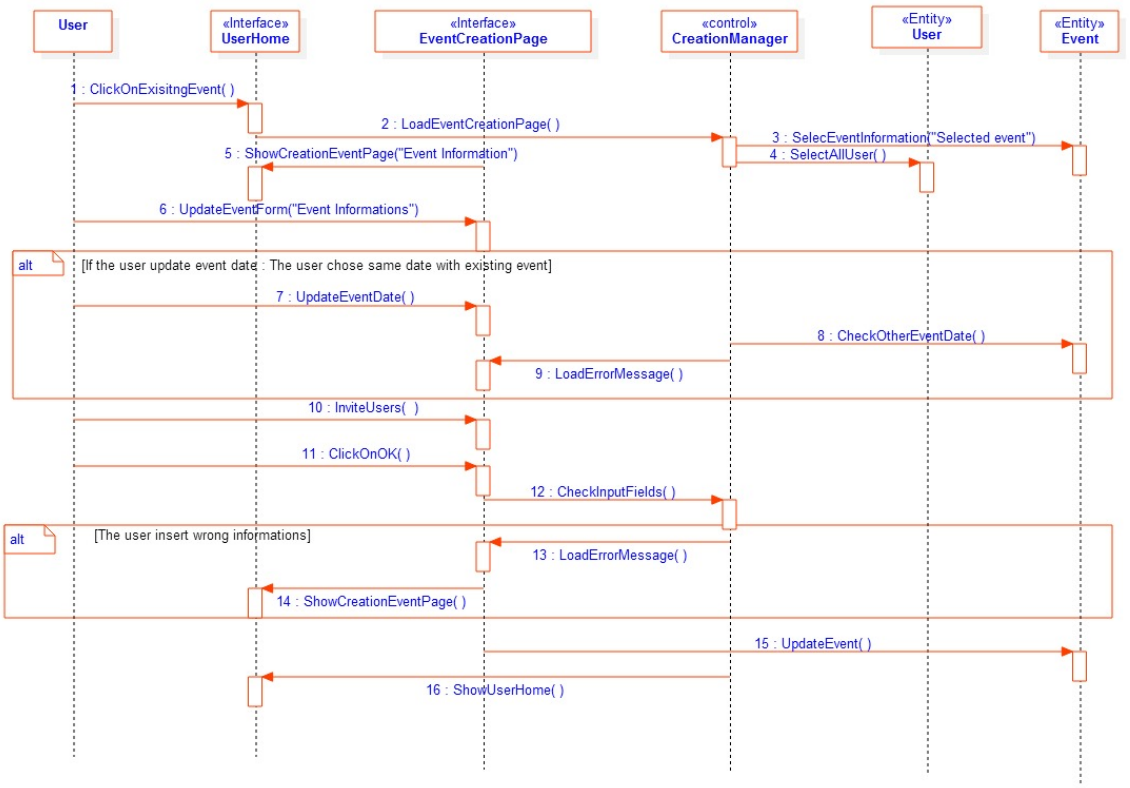


Figure 12: Modify event

5.7.9 Delete Event

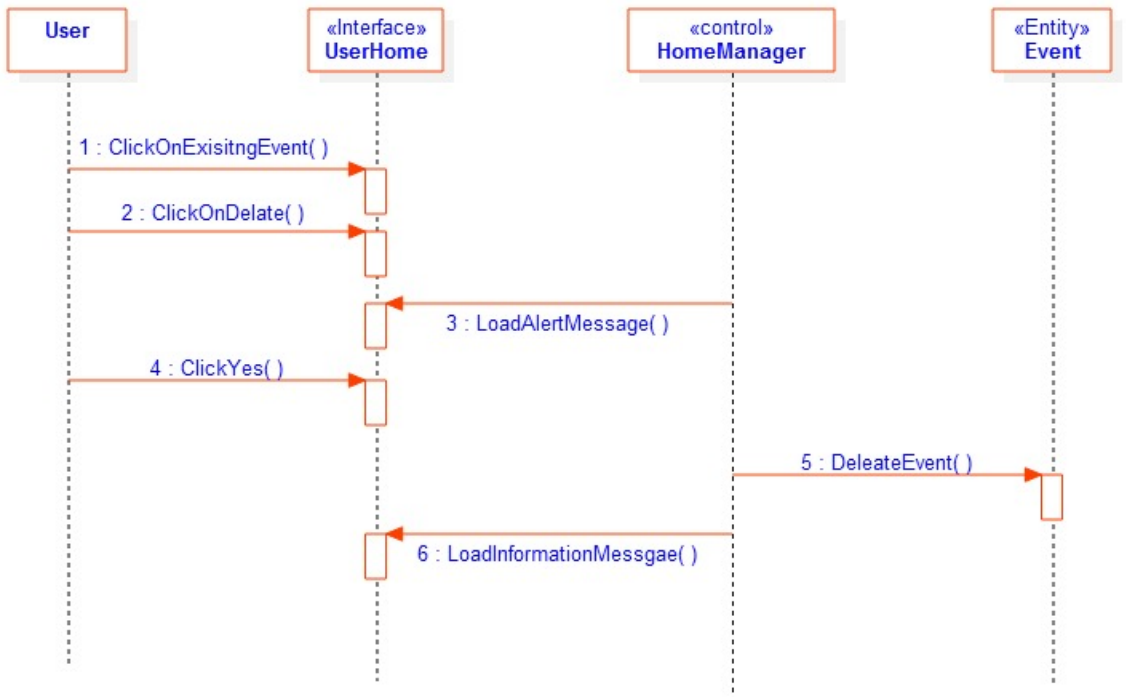


Figure 13: Delete Event

5.7.10 Modify profile

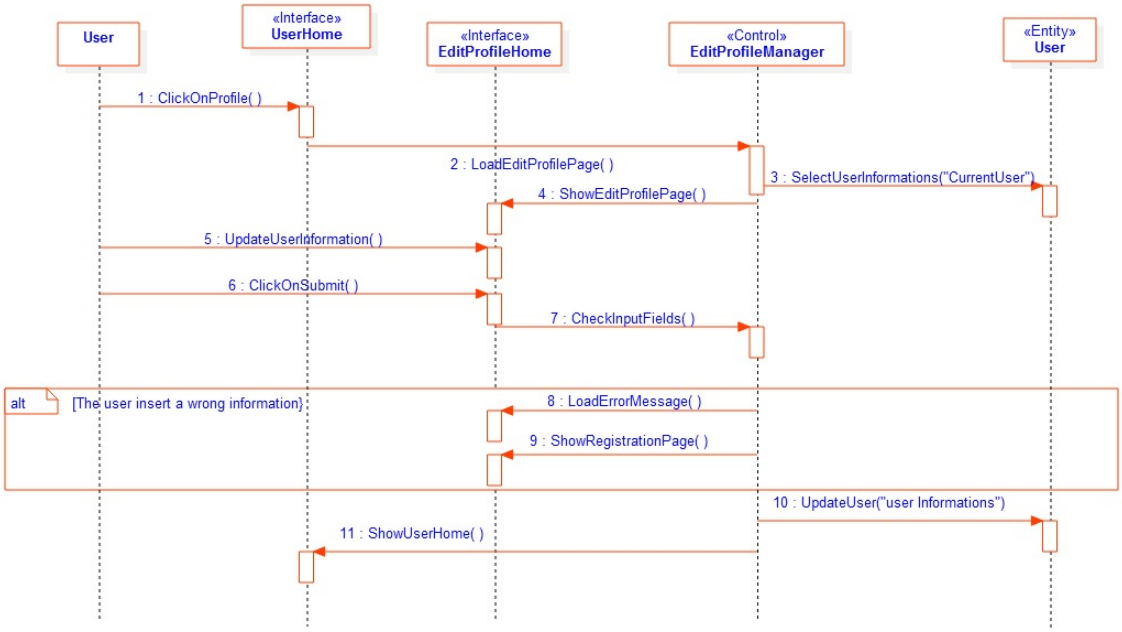


Figure 14: Modify profile

5.7.11 Notify users

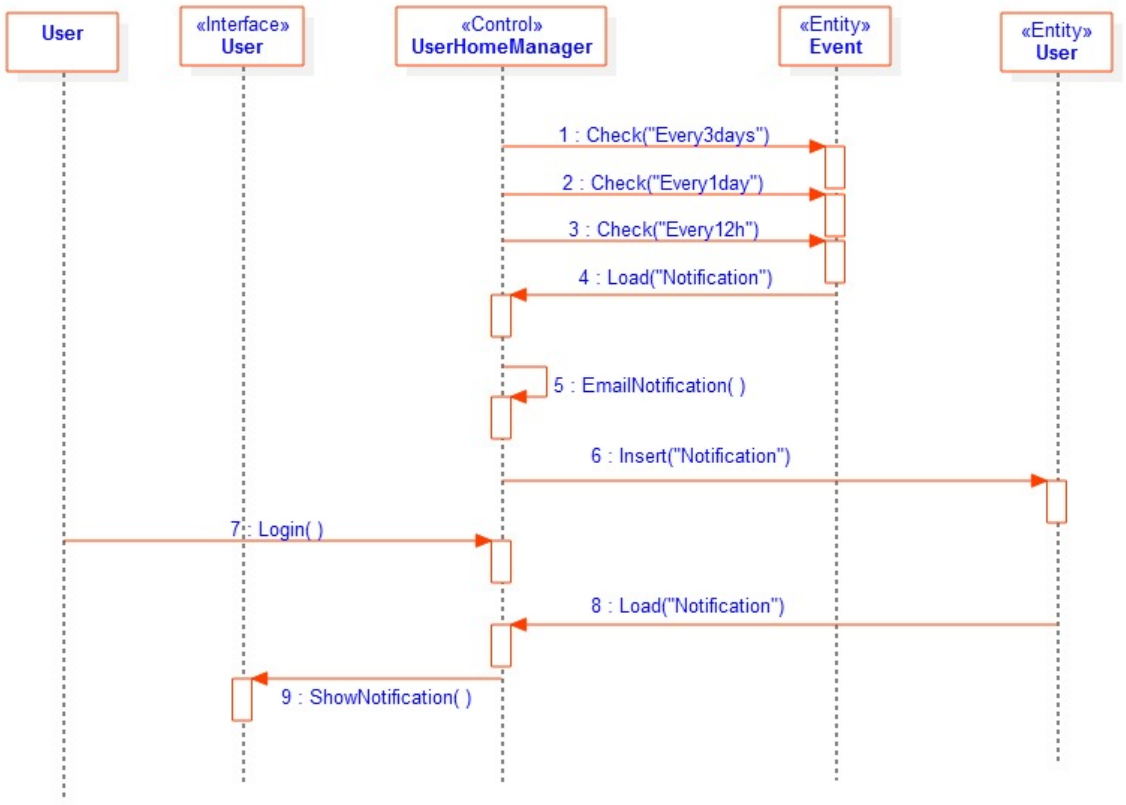


Figure 15: Notify users

5.7.12 View calendars

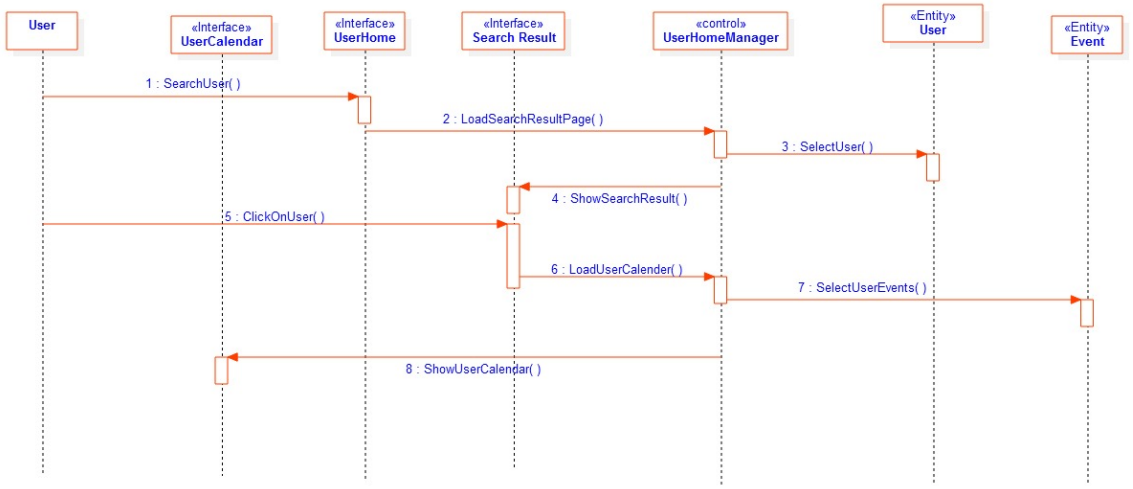


Figure 16: View calendars

5.7.13 Answer event invitation

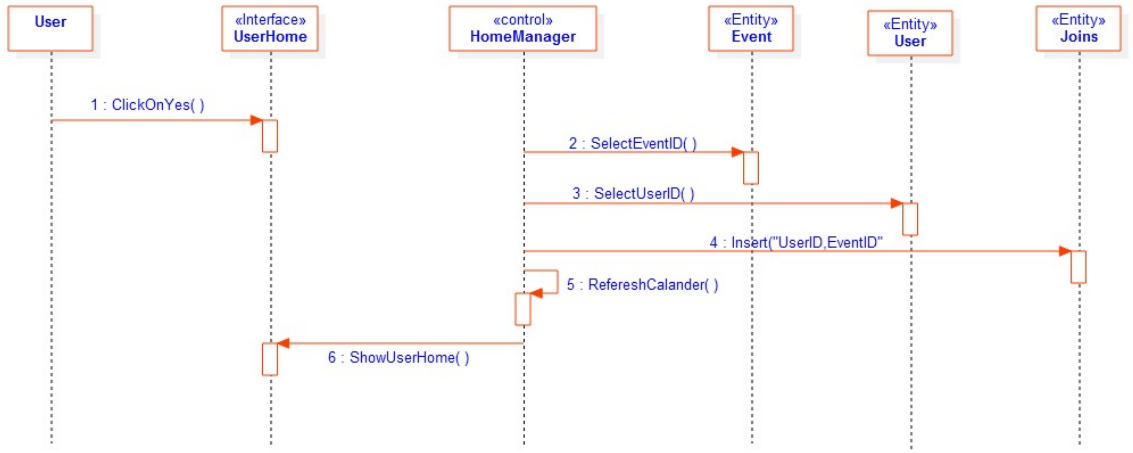


Figure 17: Answer event invitation

6 Planning

Our team consists of three students. Since it is a small team it is possible to decide and develop everything together. However, ever member has her/his responsibilities stated below with the work ing hours until the end of requirement document.

Mehdi Kaabi: Back-end development, 23h

Nazrin Javadova: Documentation, 21h

Mert Ergun: Front-end design, 22h

7 Conclusion

This software Design Document is prepared for giving detailed design information of MeteoCal 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.