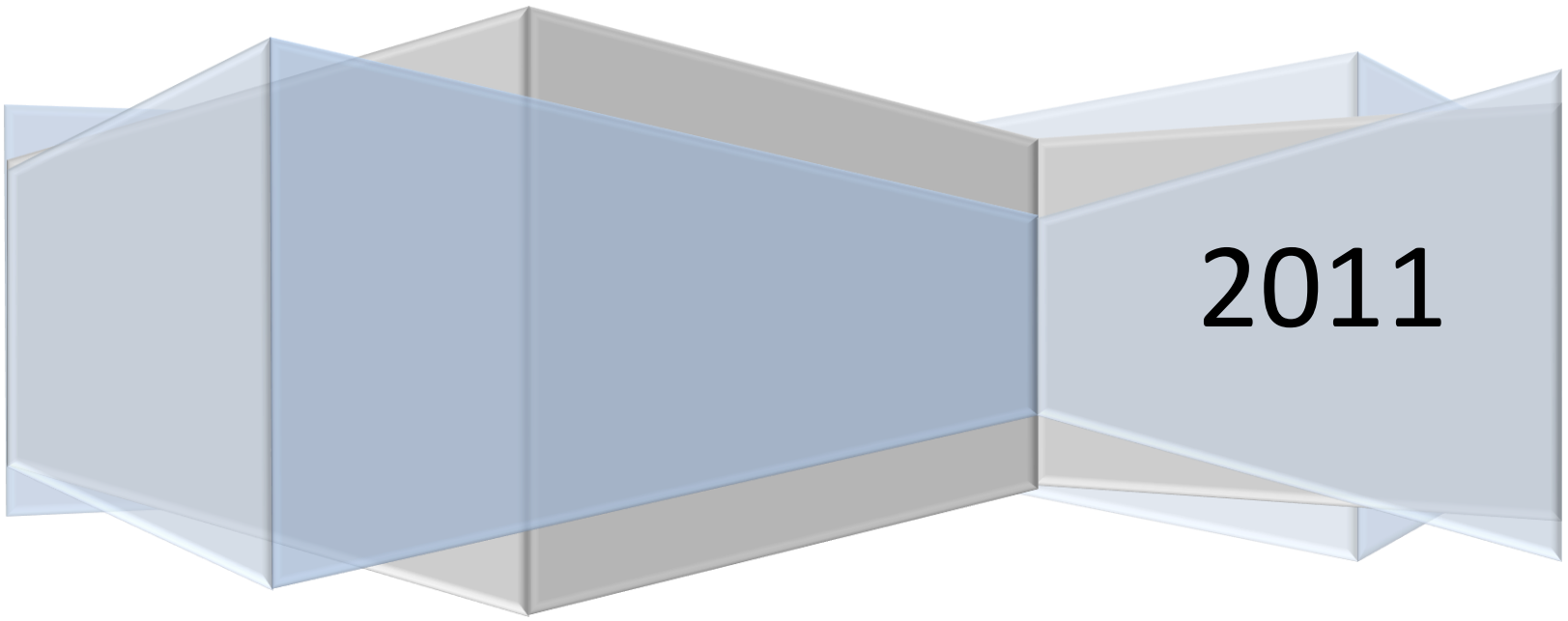
# knoWhere Design Document

## Team 8

Ankita Jain
Dharmil Shah
Sameer Abdul
Sree Harsha Uddandam
Pratik Mathran
Pranav Kothare
Srivardhan Jalan



# Contents


Purpose of Design Document ..... 2

Summary of Requirements ..... 2

General Priorities ..... 2

Outline ..... 3

Design Issues ..... 6

Design Details ..... 9

Presentation Layer ..... 9

Business Layer ..... 12

Data Layer ..... 15

Appendix ..... 17

Sequence Diagrams ..... 17

## Purpose of Design Document

The Purpose of the Design Document is to summarize the requirement for the Project and also specify the design issues encountered while designing the system. The work flow of the system is specified and the different aspects of the system that are utilized to perform user requested from the UI are highlighted.

The purpose of the software is to help in social discovery. Many people find it hard to discover events that are happening around them and getting involved. This software helps a user to find events around him that cater to his interests and hence get involved. This application is an easy to use system which enables the user to find his interests without any kind of extensive search.

## Summary of Requirements

1. An Anonymous user can Sign up for the service.

2. Any user can view the map and events that are happening in the area.

3. Any User can search, sort, filter events by name, category, location, date and time.

4. Logged-in users can RSVP, rate and review Events.

5. When a user responds to the invitation (RSVP), the event gets added to his calendar.

6. Locations Managers can create/edit/delete Events, create Event Managers for that Event and create other Location Managers.

7. Event Managers can edit and update event information and also create new Event Managers.

8. System Administrator can do everything.

9. The client can only interact with the server with an API request to which the server provides a response and the client parses.

## General Priorities

**1. Extensibility** - The server and the client have been decoupled in our project in order to have the ability to extend it to other platforms easily in future. In future, functionality can be easily modified and added without having to make major changes in our current system architecture.

**2. Maintainability** - The system has been designed in a way that every module can be individually taken and tested without the hassle of taking other modules into consideration. This provides us with the capability of providing maintenance in case of future amendments. Since most people know the Web Forms pattern, the use of Web Forms pattern will make it easy for programmers to expand on our work easily in future.

**3. Usability** - The system has been designed in the most user friendly way so that the user can access all the functionality of the web service with minimum hassle. Users of all experience levels can use the web service intuitively.

**4. Performance** - The system has been designed in n-tier architecture to enhance performance. The architecture was chosen keeping in mind that the system would need to perform heavy data exchanges. The time delay for accessing the various functionalities in the website has been minimized.

**5. Security** - Security issues like the assignment of location managers, event managers and creation of events have been kept in mind and the system has been designed to prevent any malicious user from adding deleting or modifying user sensitive data.

## Outline

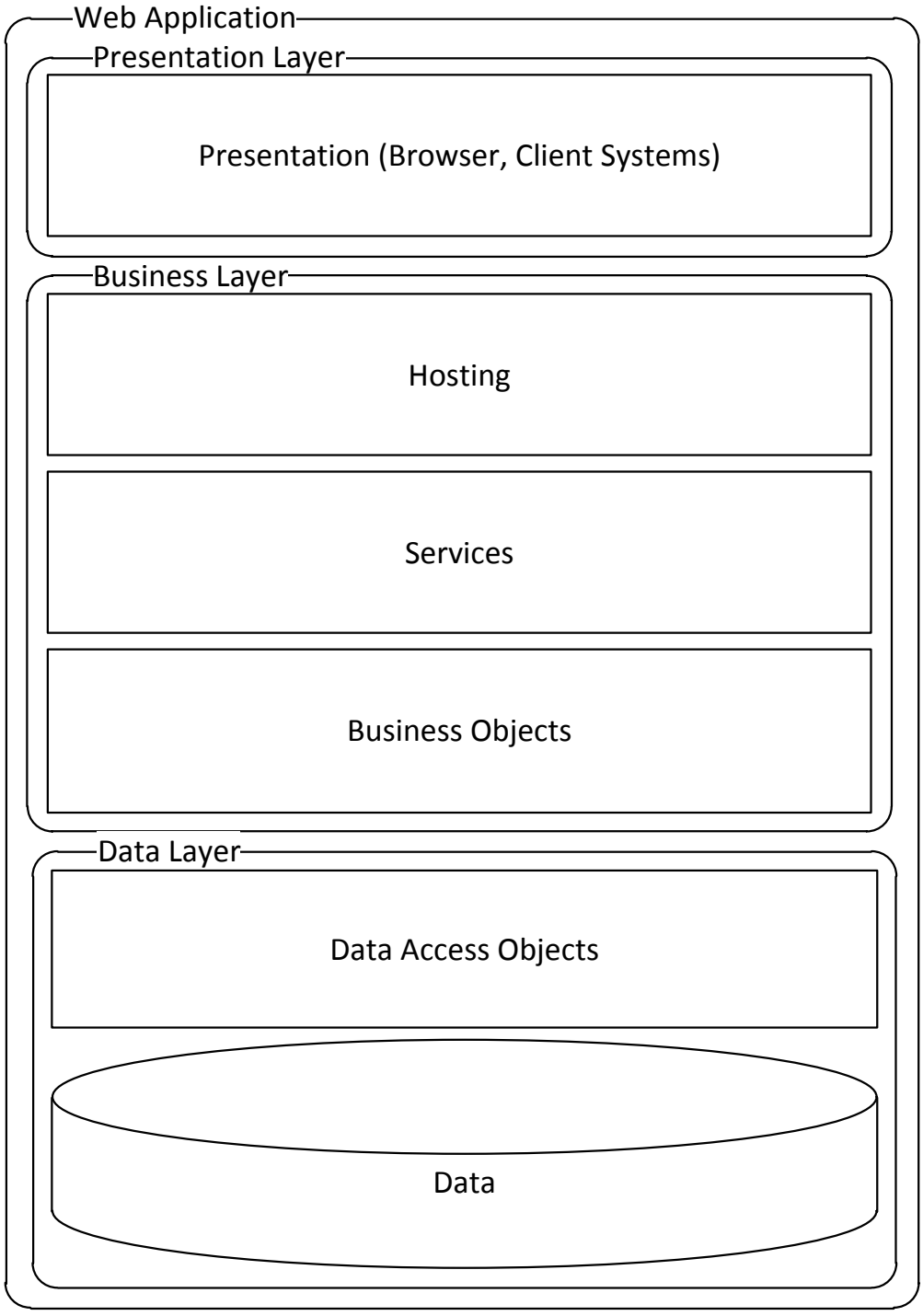



Figure 1 Web Application Architecture

Figure 1 illustrates the Architectural Design of the Application. The data layer provides access to the database and maps database tables to business objects and vice versa. This is the only place in the application where data access occurs, including the execution of dynamic SQL and/or stored procedures in the database. Data access objects represent the data access functionality required to run the application. The interfaces will be implemented by database specific data access classes, see Figures 7 and 8.

The business layer contains business objects. Business objects represent the domain logic that is relevant to the application, i.e., the data and the business rules that are important to the functioning of the application. Each business object will have the ability to validate its data using a simple business rule engine that is built into the base class for each business see Figure 6.

The services section implements a single point of entry throughout which all communication with the layers below must occur. It is a busy place because it manages business objects and coordinates several common operational tasks, including data validation, user authorization, and transaction management. It also coordinates the interaction between business objects and data access objects by saving and retrieving business objects using data access objects to and from the database

The interface (or contract) between the service and the presentation layer is built around a messaging model in which messages contain Data Transfer Objects. Data Transfer Objects is an enterprise design pattern in which you have objects that contain business data only and no behavior (methods or properties), see Figure 5.

The hosting section exposes the service to the outside world. Setting up this section is primarily a configuration exercise in which the hosting environment is established, such as IIS, Windows Services, WAS, or others. The choice of selecting the optimal hosting model and configuration is largely driven by performance requirements, the operating platform, and the clients (i.e. the presentation layer), that are going to consume the services. We intend to provide a generic service that uses Hypertext transfer protocols. However, it is unclear at this point if that will be completely possible and more research will be required.

The presentation layer will consist of different UI platforms that are unique with respect to deployment, user experience, coding methods, etc. However, with the 3-tier architecture we have clearly demonstrated the ability to build a single foundation that is accessible by any kind of Technology with several different hosting options. Whatever UI platform or hosting options, no code changes will be required for the business and data layers. To see an example UI of a web application see Figures 3 and 4. Figure 2 illustrates how client systems will communicate with our service and Google's web services on a high level.

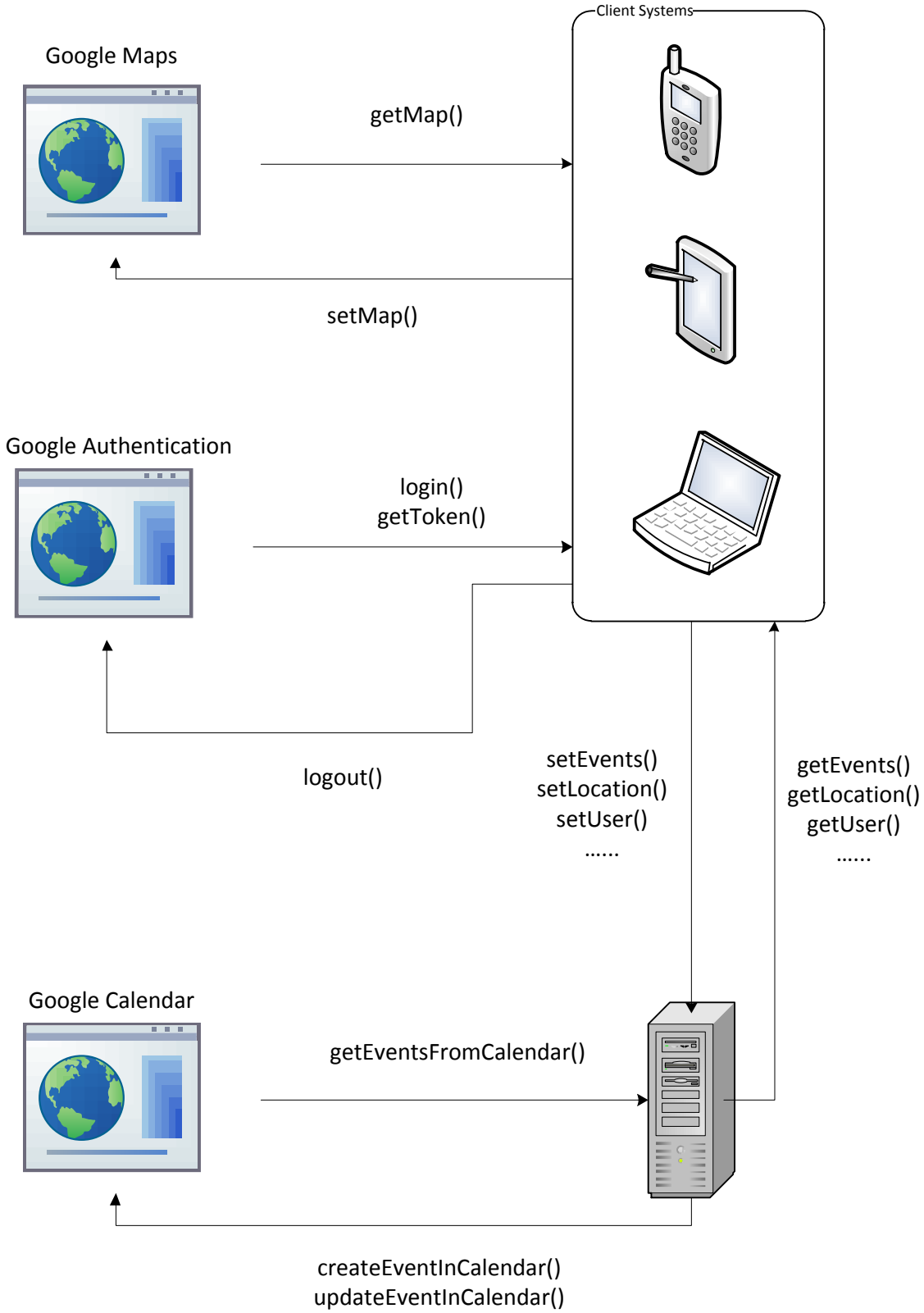



Figure 2 Communication between Services

## Design Issues

**1. Agreeing On the Area of Application Usage**

Option 1: Only Purdue.

Option 2: West Lafayette and Lafayette.

Option 3: Major cities in US.

Decision: Option 1 was selected.

Reason: We had thought about major cities but we scraped the idea because the extent of work required was too great for the duration of this course. Also we wanted to make this software exclusively for Purdue University as we are Purdue students and we are familiar with the wants of the other students. Purdue also does not have such a system in place, and the systems that are in place such as eBoard and eFlyer, have not been developed to their full potential. This software would also help the students immensely.

**2. Login Credentials**

Option 1: Allow only Gmail user to login.

Option 2: Allow all Email ID for login.

Option 3: Make Own registration system where the user has to key in the username and password he wants.

Decision: Option 1 was selected.

Reason: We used only Google account for login and validating credentials as Gmail is one of the most popular Emailing websites. Our application uses the Google API for accessing the map, navigation system and calendar. Using Gmail accounts would enable us to import/ export events to the users calendar when the user RSVP's events. This would also enable the user to have email reminders about events he has RSVP'd.

**3. Privileges for Adding and Editing Event**

Option 1: Having 1 Manager who adds event and manages the events as well

Option 2: Having 2 Managers- 1 for adding events and 1 for managing events

Decision: Option 2 was selected.

Reason: We have 2 managers in our system as a manager for a location i.e. the manager who adds events, would not necessarily want to manage the event as well. He could want some other person/ group to manage the event. This is where the event managers come in, as they would be able to edit the details for the events and provide the users with addition details for the event. This reduces the load for a location manager as he might have many events taking place simultaneously in the same location.

**4. Designing the Server and Client**

Option 1: Having the server and client closely integrated

Option 2: Having the server and client decoupled

Decision: Option 2 was selected.

Reason: We decided that we would provide the server in the form of an API so that it could easily be adopted to create applications in different platforms. Creating applications for other platforms in a closely integrated system

would be a problem and could not be done easily. We also intend to provide the API for the general public so that they can use it for further application development.

**5. System Architecture**

Option 1: N-Tier Architecture

Option 2: Service Oriented Architecture

Decision: Option 1 was selected.

Reason: In an N-Tier Architecture the focus is on building monolithic applications that can be distributed across multiple processors to increase performance and scalability. This architecture is useful when handing large amount of data exchange. Service oriented architecture is useful when handing small amount of data exchange as it uses a large amount of bandwidth. The N-Tier Architecture is used for our application as we are continuously interacting with a map which involves large data exchanges.

**6. Details of an event that an Event manager can edit**

Option 1: All details including location and time.

Option 2: All details besides location and time.

Decision: Option 2 was selected.

Reason: We decided not to give the event managers the power to edit an event location and time as the event managers are assigned by location managers and might not always know the availability of a particular location on a particular time. Editing event location and times are left only to Location managers who know all the events that take place in that particular location. Also an event manager I specifically assigned under a location manager and if he changed the location it would create conflicts in the system.

**7. Event managers allowed adding other event managers**

Option 1: Event managers having to go to location managers to add other event managers for a particular event.

Option 2: Event managers having the power to create other event managers for a particular event.

Decision: Option 2 was selected.

Reason: We decided to give event managers the power to add other event managers for a particular event as by doing this event managers could reduce the work load on themselves by assigning others to help them. By not having to ask location managers to add them instead they save the location managers for unnecessary extra work and also makes the process that much faster in the real world.

**8. Implementing many to many relationship in the database**

Option 1: We can implement the relationship as a column in one table and serialize the multiple objects into one single object

Option 2: We can implement the relationship as a mapping table and both object being foreign keys in that table

Decision: Option 2

Reason: We decided on the mapping table as it's very flexible and provides a decentralized approach to storing in database. We can delete one object easily in the mapping table without affecting other objects.

**9. System Architecture of the Presentation Layer**

Option 1: Follow MVC pattern

Option 2: Follow Web Forms pattern

Decision: Option 2 was selected

Reason: We started by using the MVC pattern and created some Models, Controllers and Repositories. Our home page had the entire Purdue campus map along with an AJAX button with a lot of AJAX functionality. But the AJAX commands required the server to propagate the entire DOM and we faced a lot of difficulties fixing it. This could be easily solved if we shifted to Web Forms pattern.

**10. Location Managers allowed adding other location managers**

Option 1: Location managers having to go to the system admin to add other location manager for a particular location.

Option 2: Location managers have the power to create other location managers for a particular location.

Decision: Option 2 was selected.

Reason: We decided to give location managers the power to add other location managers for a particular location as by doing this location managers could reduce the work load on themselves by assigning others to help them. By not having to ask the system admin to add them instead they save the system admin for unnecessary extra work and also make the process that much faster in the real world.

**11. Layout of the web application**

Option 1: Have a separate dashboard for all the users and the admins which contained all their functionality and privileges and a fixed Navigation Bar.

Option 2: Have a Navigation Bar with tabs to all the functionality and privileges which changed according to the user.

Decision: Option 2 was selected.

Reason: Initially the home page of our web application just had the log in functionality with a description of all the different functionality for a user and an admin. The application had a fixed Navigation bar and a separate page for the dashboard which contained all the functionality and privileges of a user. But this was really inconvenient as the user had to go back to the dash board each and every time to perform an action. Having a Navigation Bar which would change according to the user with all the tabs to his privileges made much more sense and would be much more convenient. Also the home page was changed to the Purdue campus map because the main objective of our service is social discovery.

# Design Details

## Presentation Layer

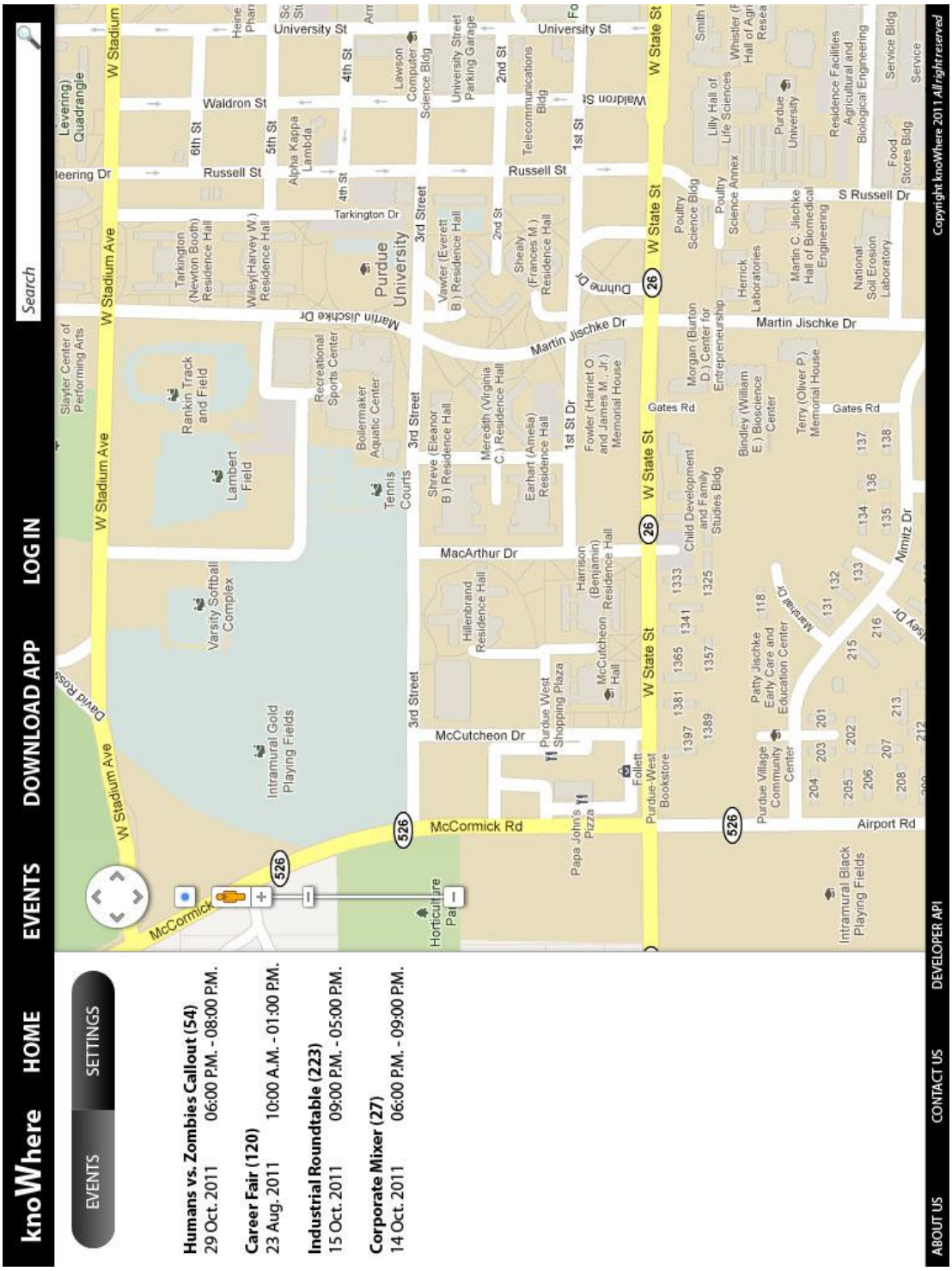



Figure 3 Homepage for Anonymous User

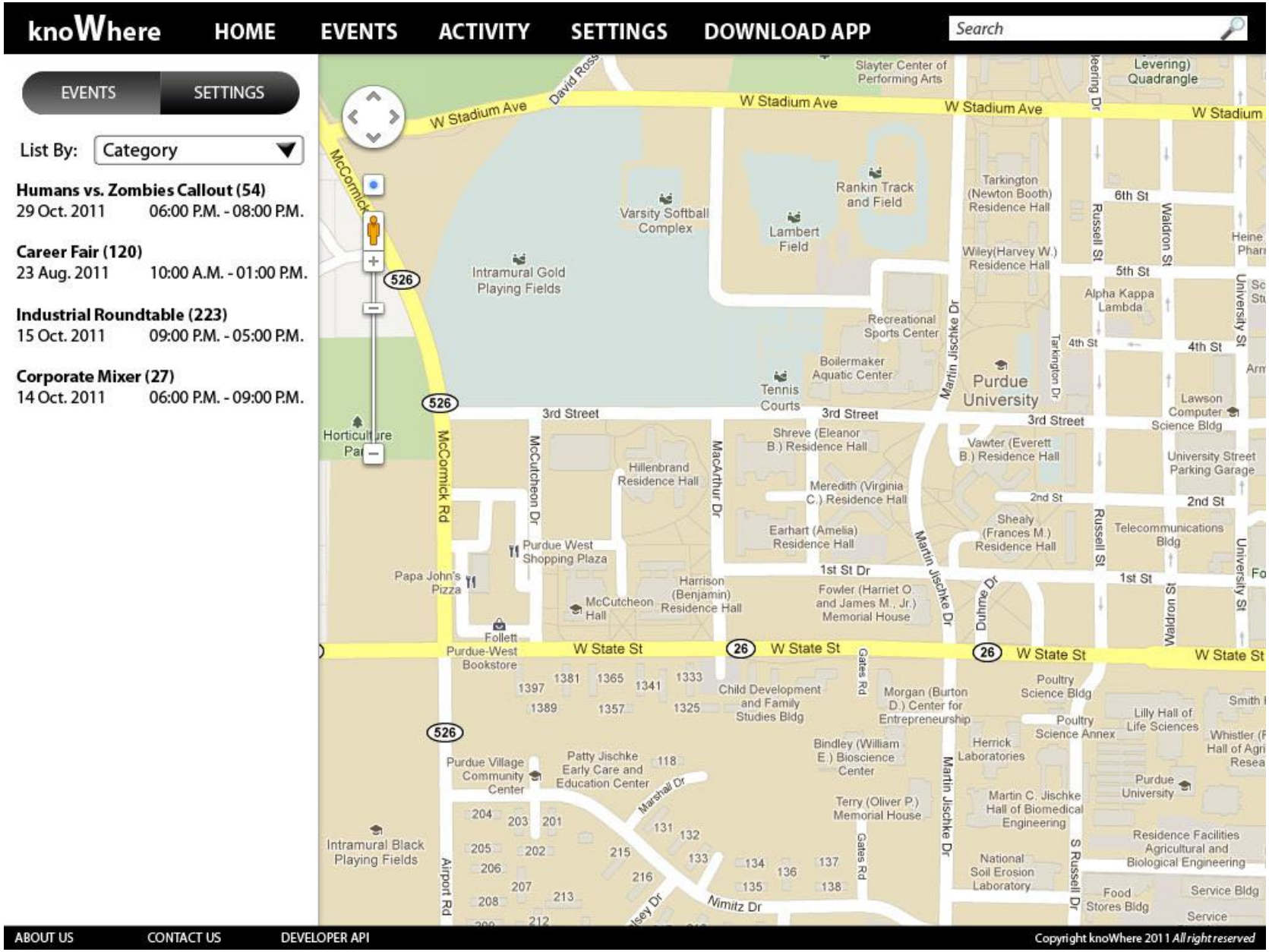



Figure 4 Homepage for Logged In User

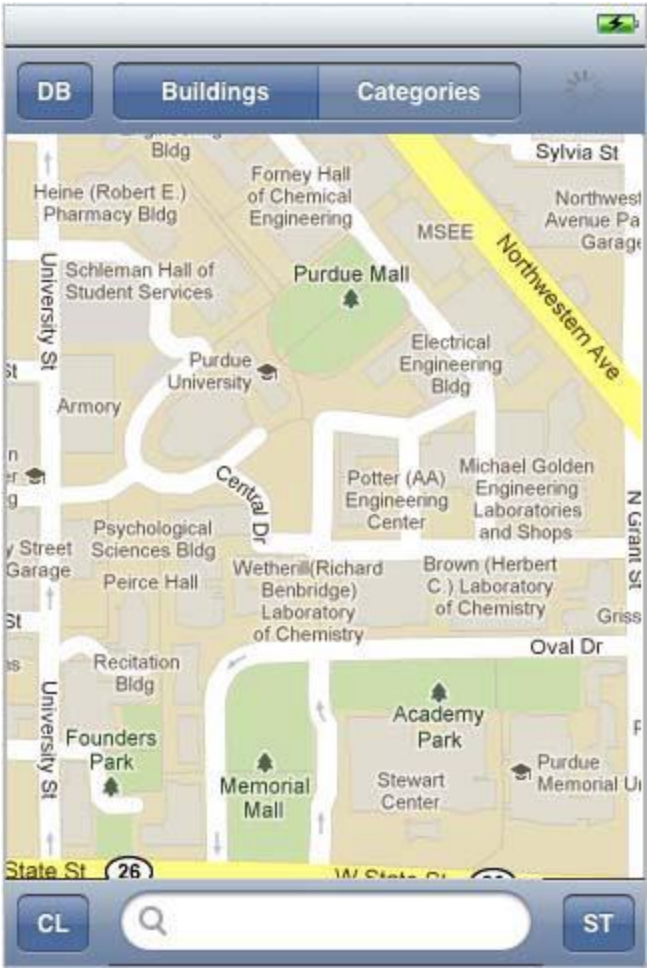



Figure 5 Homepage for Anonymous User on an iPhone

## Business Layer

### *Service Section*

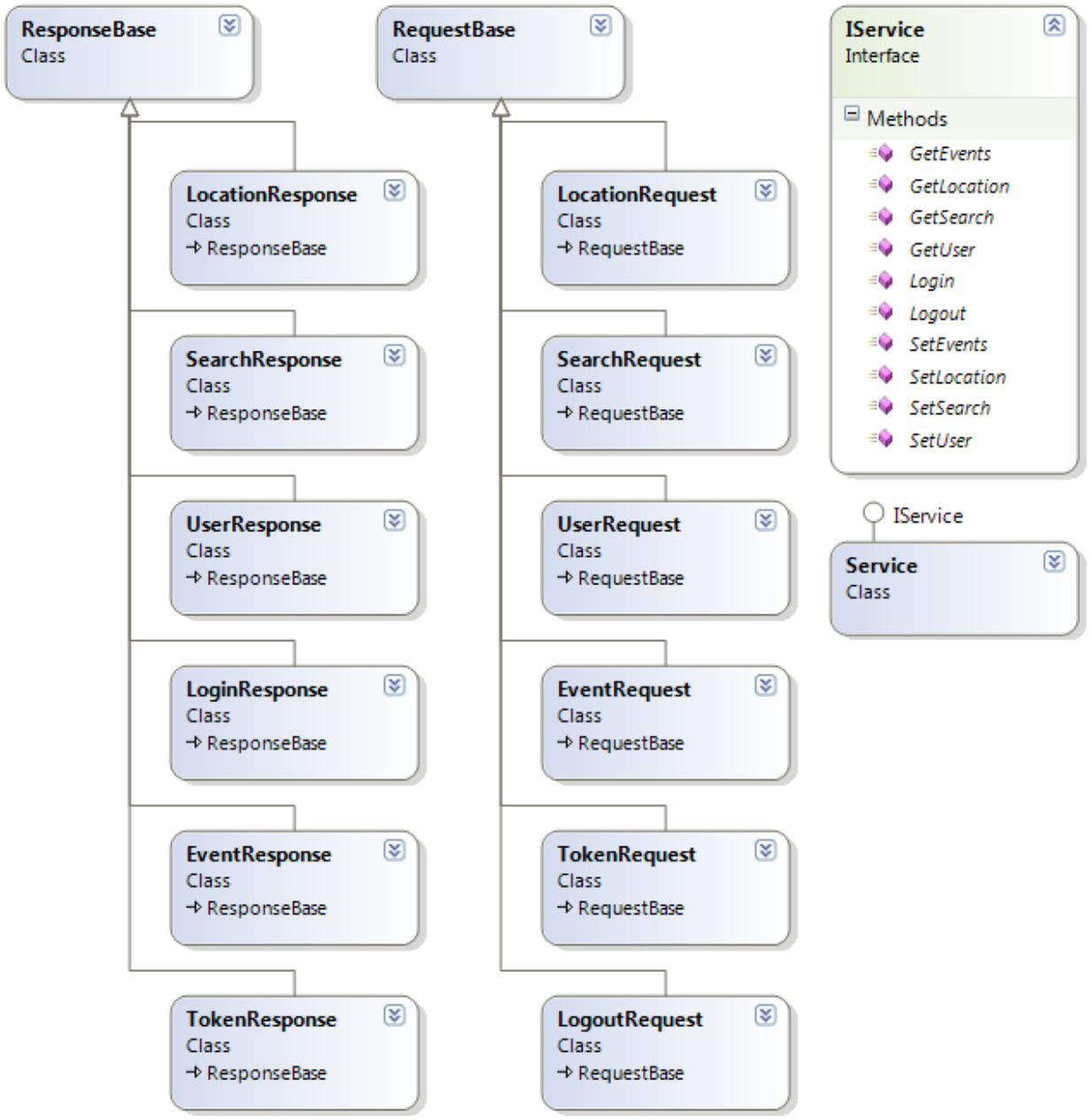



Figure 6 Services with Corresponding Messages

## *Business Objects Section*

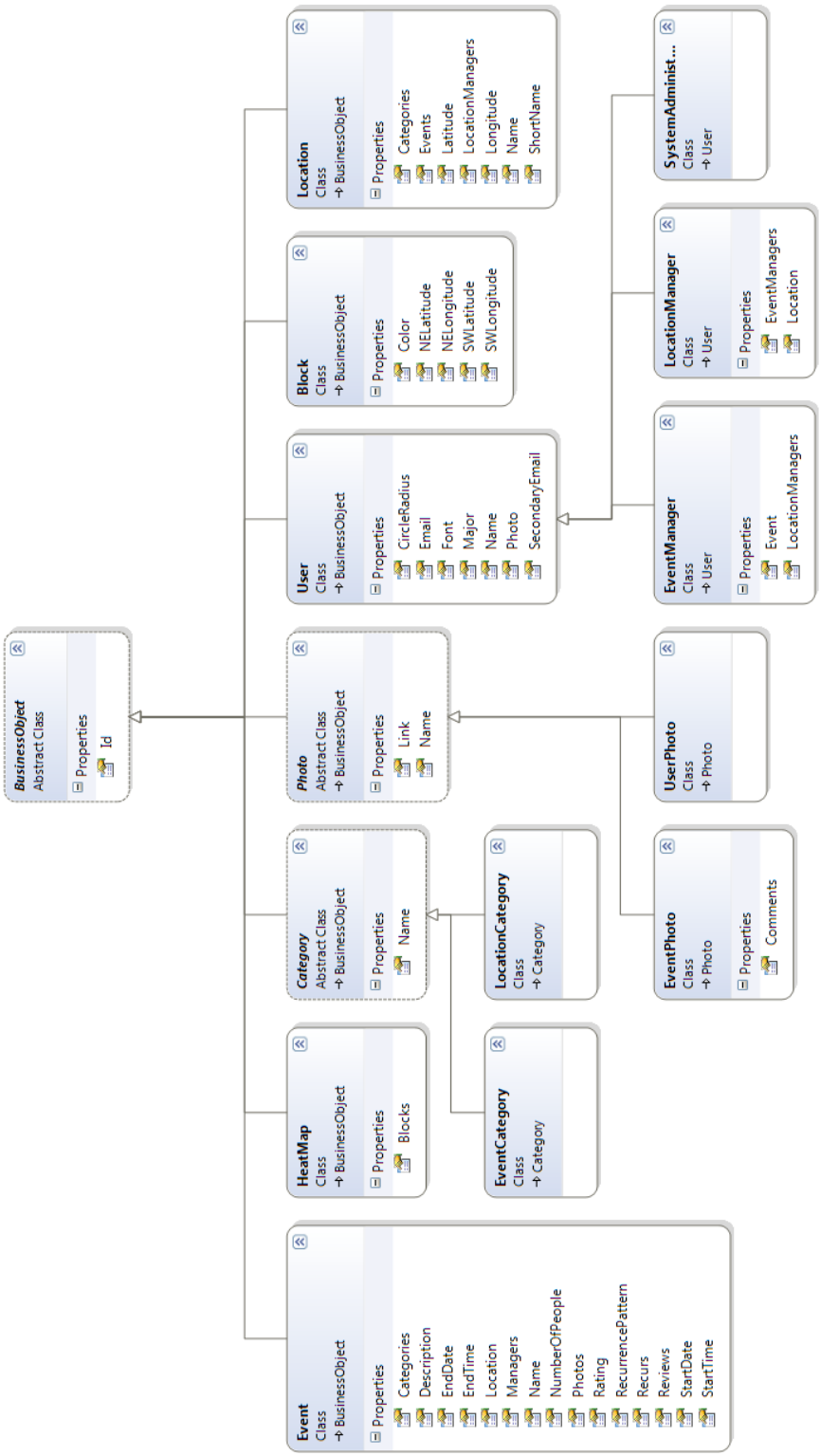



Figure 7 Business Layer Objects

Event

It is used to access the event details for a particular event by the service section.

Location

It is used to access the location details for a particular event by service section.

User

It is used to access the user details for a particular user by the service section

- LocationManager
  It is used to access location manager details by the service section. A location manager extends a User
- EventManager
  It is used to access event manager details by the service section. An event manager extends a User.
- SystemAdministrator
  It is used to access system administrator details by the service section. A system administrator extends a User.

Category

It is used to access category information by the service section.

- EventCategory
  It is used to access event category information by the service section. EventCategory extends Category.
- LocationCategory
  It is used to access location category information by the service section. LocationCategory extends Category.

Photo

It is used to access photos by the service section.

- UserPhoto
  It is used to access User photo i.e. profile picture by the service section. UserPhoto extends Photo.
- EventPhoto
  It is used to access event photos by the service section. EventPhoto extends Photo.

Block
It is used to access information in a given block of area by the service section.

HeatMap
It is used to access heat map details by the service section. Heat map implements Block and is a collection of block

## Data Layer

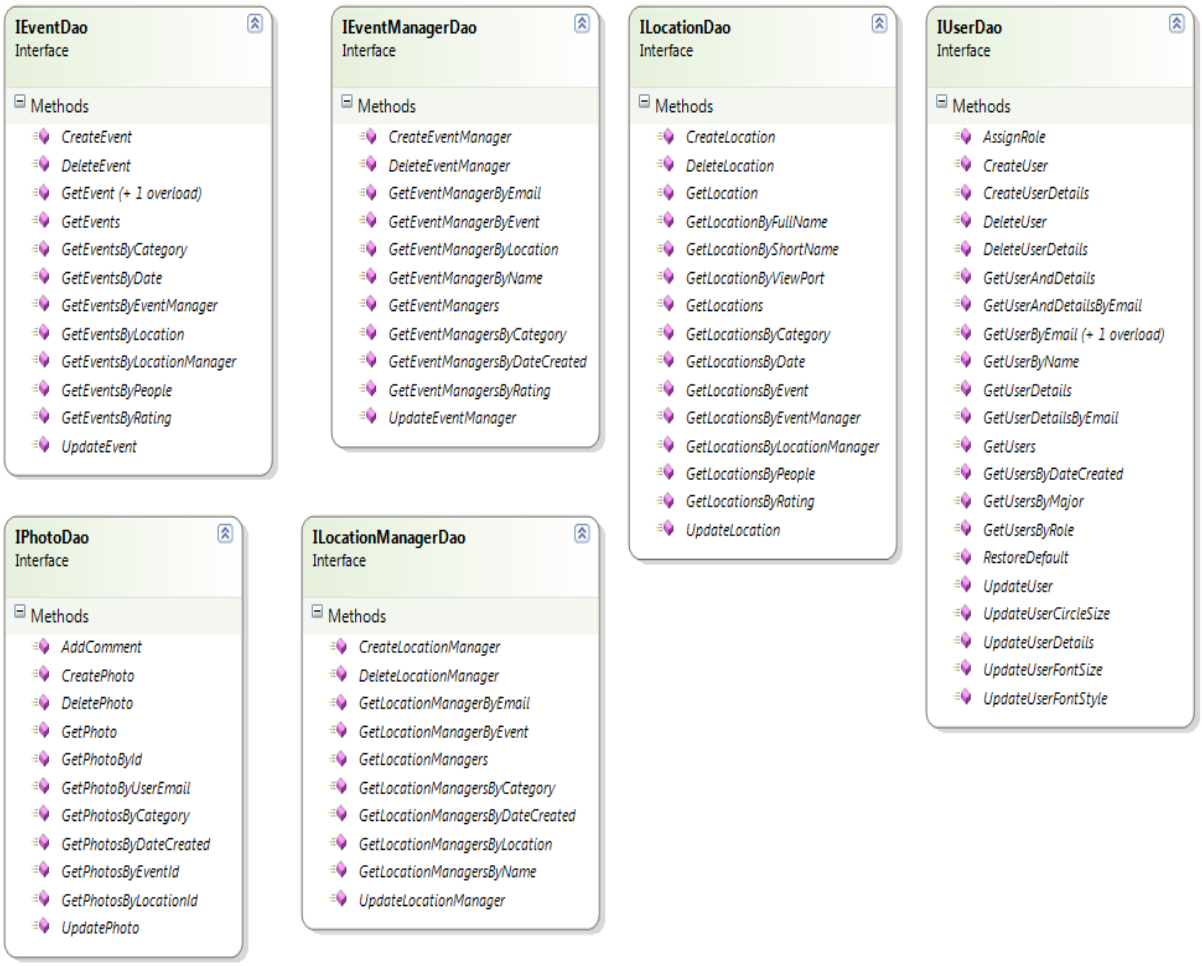



Figure 8 Data Access Interface

### *Data Access Section*

The Service section will call the Data Layer methods for access to any objects in the database.

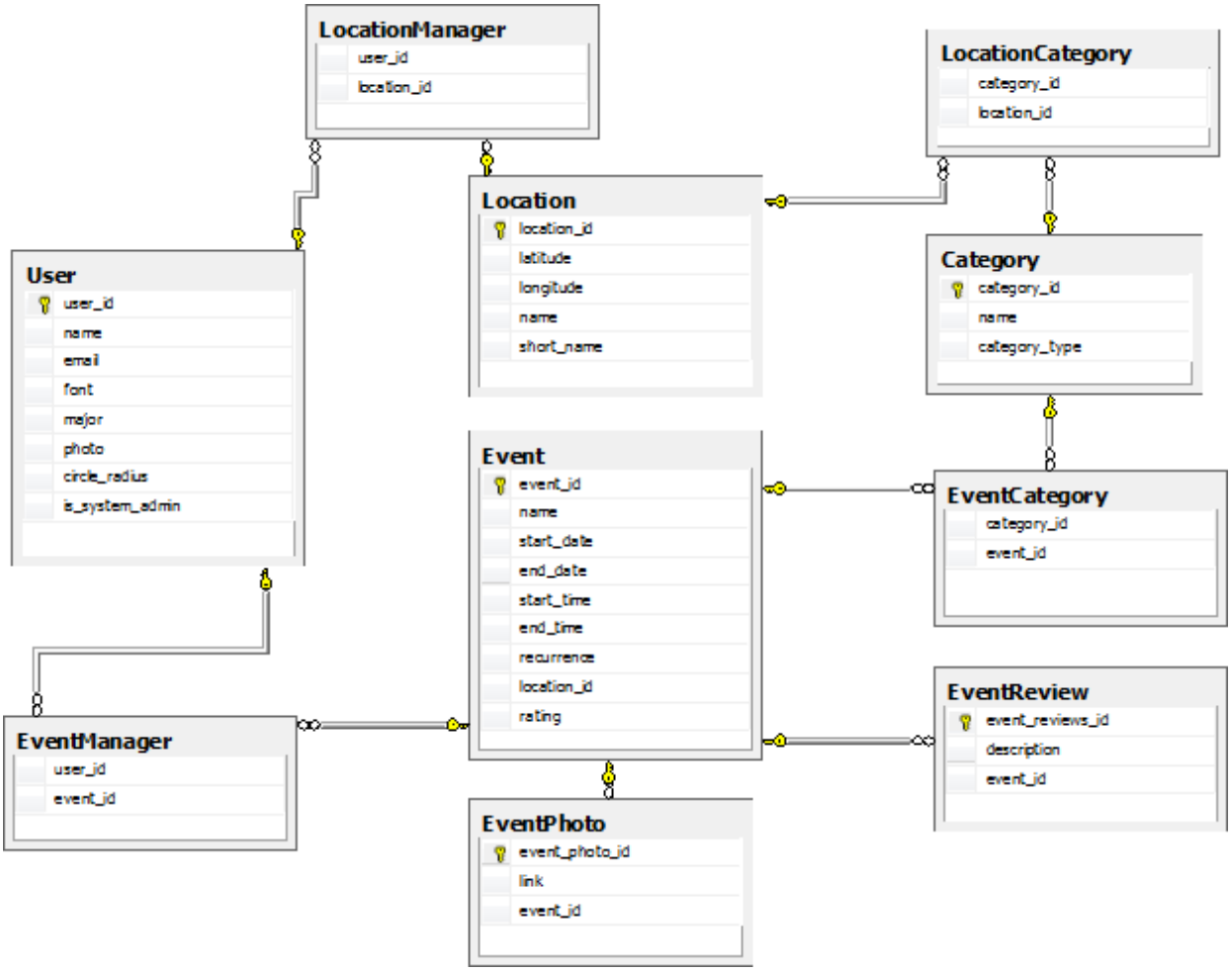



Figure 9 Data Schema

# Appendix

## Sequence Diagrams

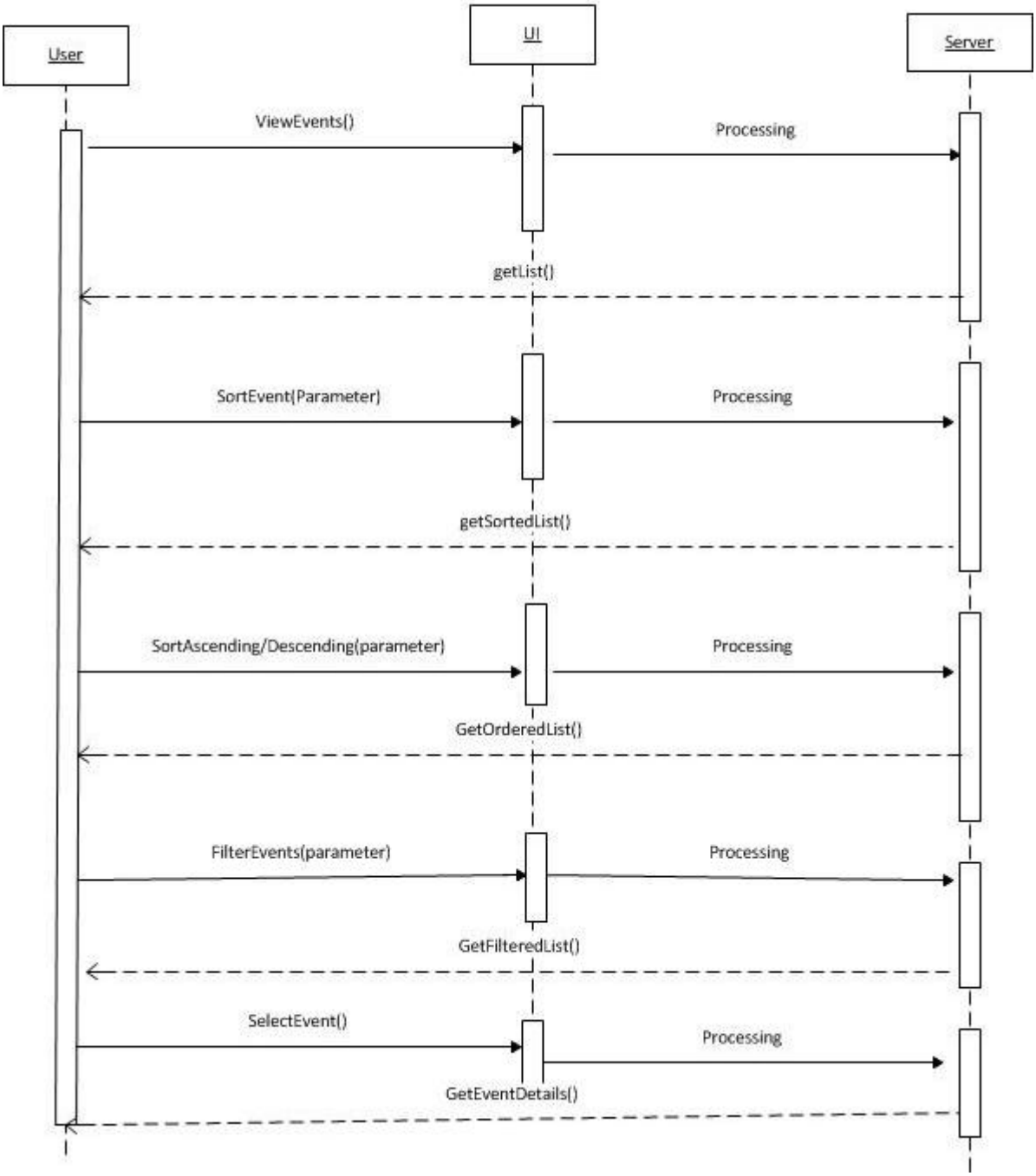



Figure 10 View List According to Order and Category

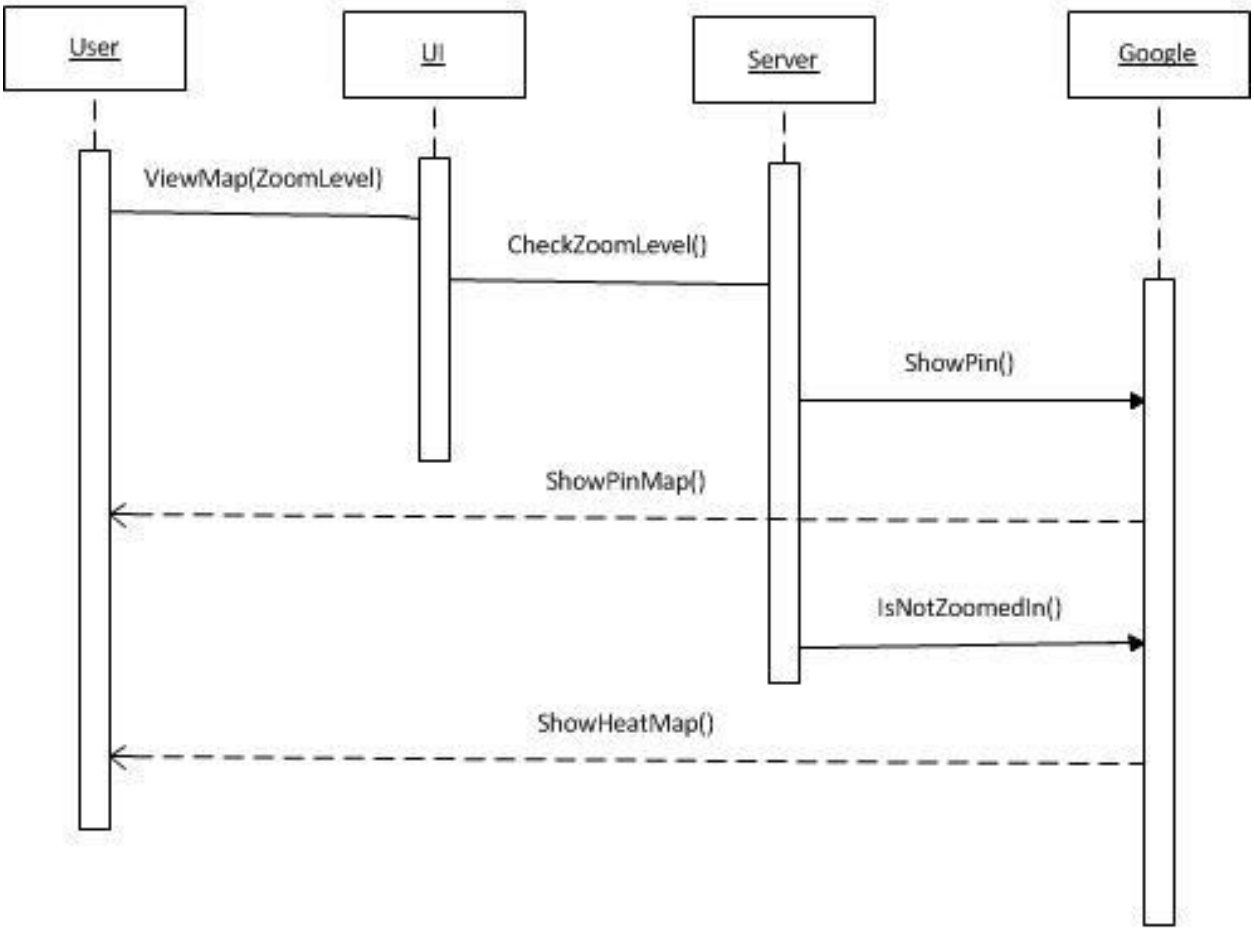



Figure 11 View Map for a given Zoom Level

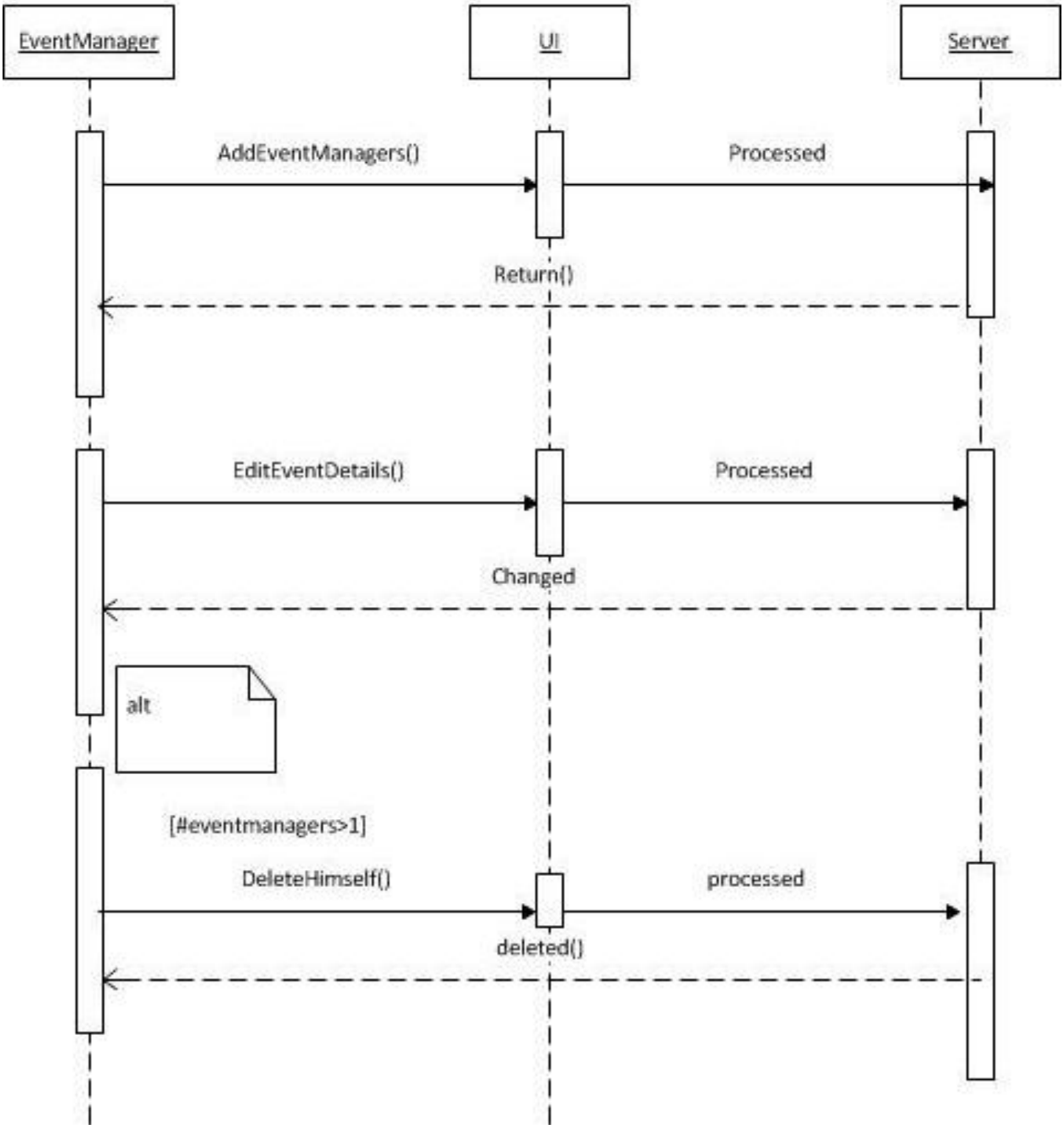



Figure 12 Functionality for Event Manager

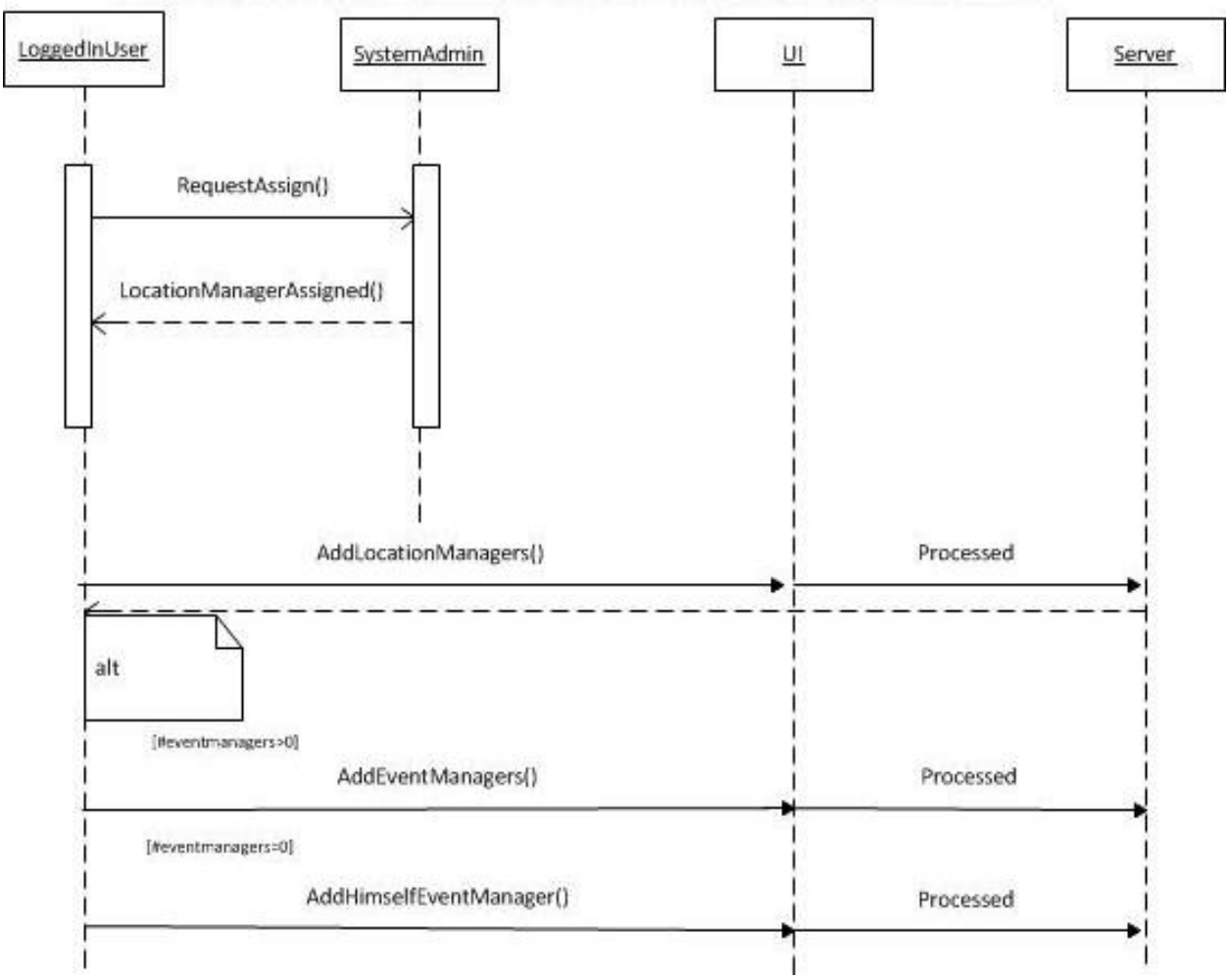



Figure 13 Assigning of Location Manager and his Responsibilities

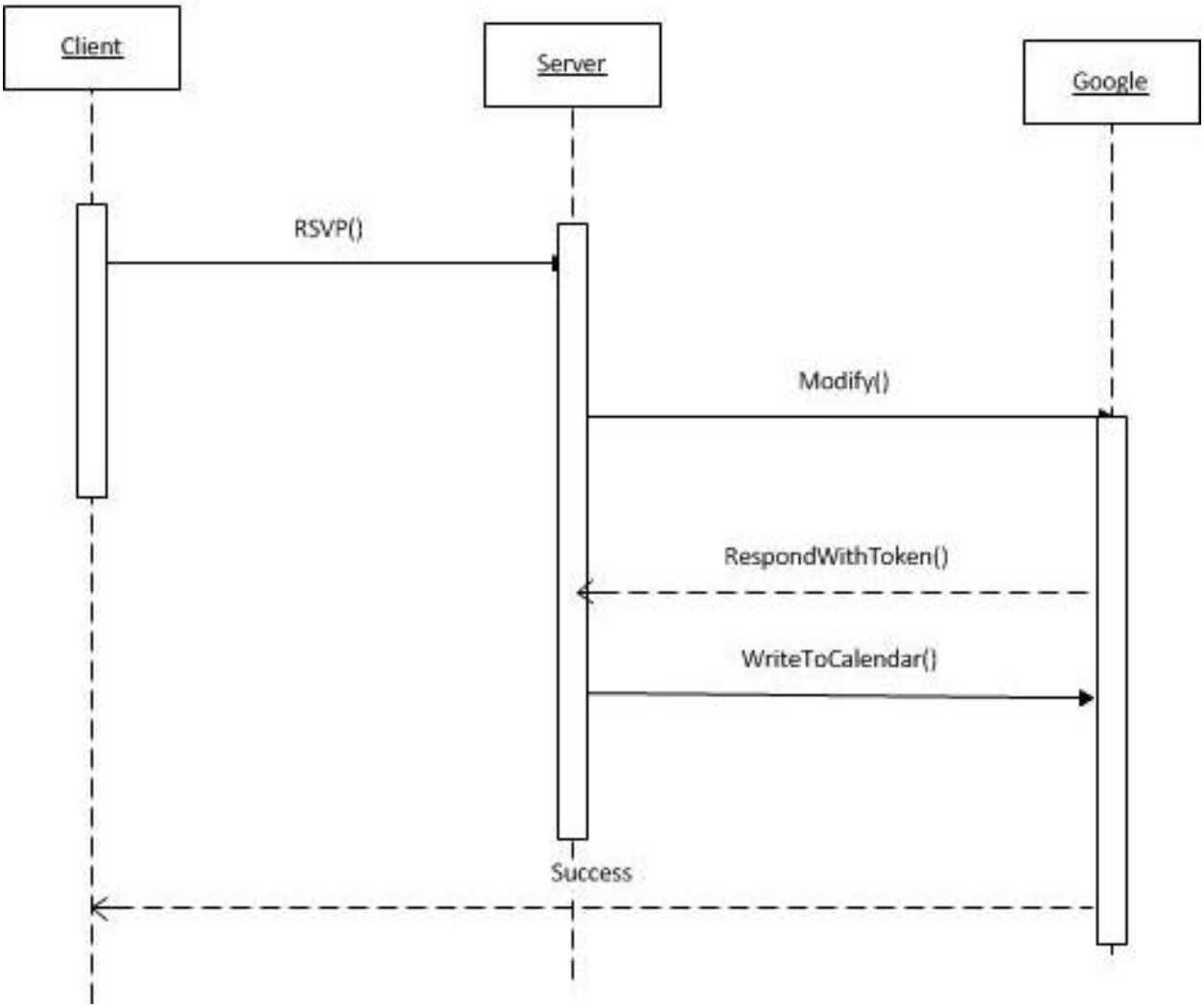



Figure 14 RSVP Process in Detail

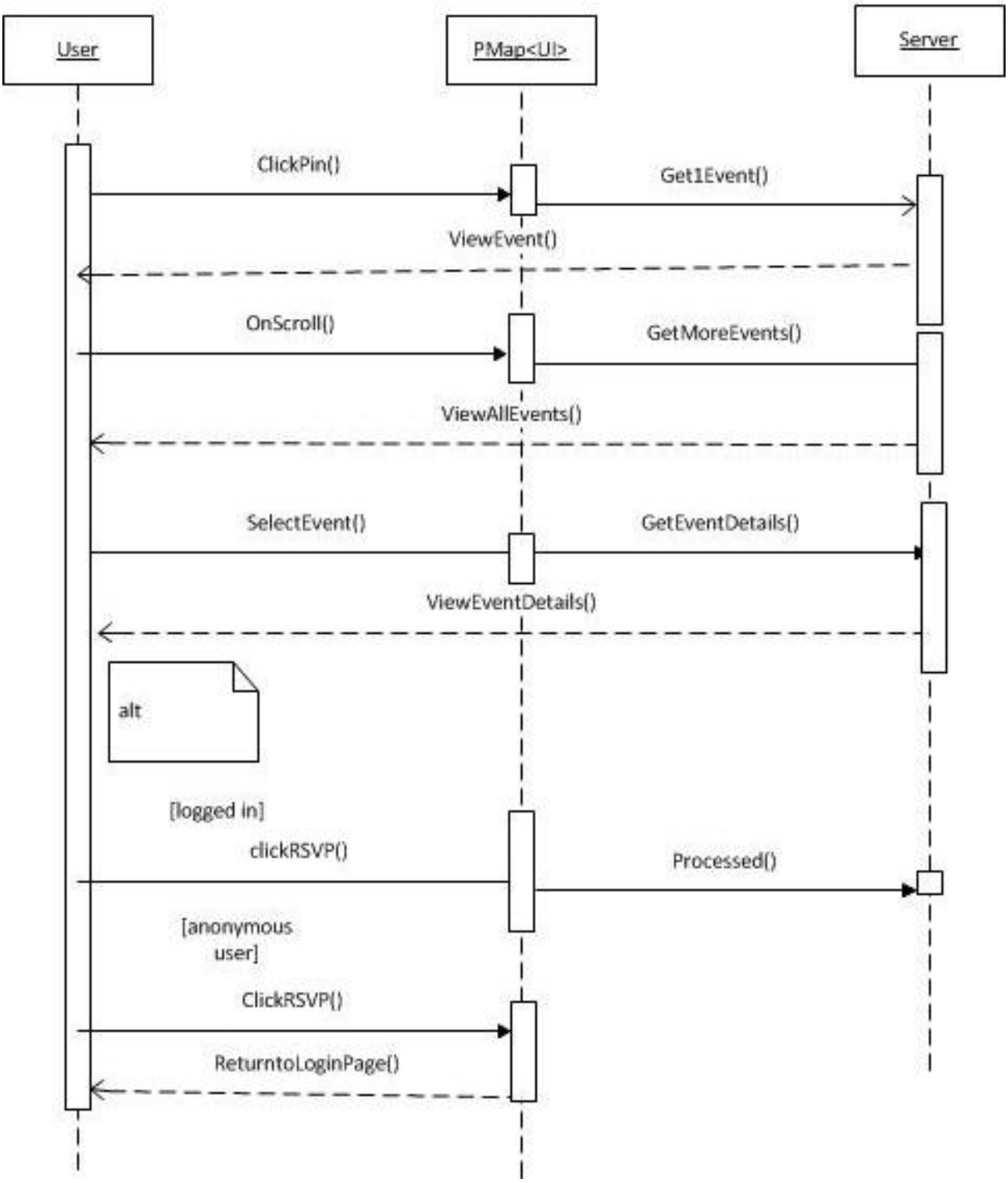



Figure 15 View Events on the Map