CENG 491
SENIOR PROJECT

Detailed Design Report

Macro-Hard

PROJECT TITLE: Fizan

COMPANY STAFF: Uğur Altun 1448356
Kürşat Aksakallı 1448315
Kemal Beşkardeşler 1448455
Ömer Faruk Çelik 1448547

Index

1. INTRODUCTION

Purpose of the document:

The purpose of this document is to show initial design of our project “FIZAN”, and introduce our project with all needed diagrams for you to learn about the project and for us to plan our works we will do in the future.

2. PROJECT DESCRIPTION

2.1 Problem Definition

Nowadays usage of mobile phones is widespread and almost everybody use mobile phones in daily life. Therefore mobile phones need to provide most of the required features a person may need in daily life. It is needed to make mobile phones something that makes people life easier in daily life. We can consider such situations:

When a person wants to meet with a friend in some place at some time, it is difficult to arrange this. Many calls or many message sending is needed, for this simple thing. If it is a meeting with many people it is harder and more expensive. Therefore we can imagine such a feature that makes a place visual to all mobile phones. Then we can put an event on this map which can be seen from our friends, which has meeting time and exact position on the map will be useful to arrange these kinds of meetings.

For parents watching their children is a big matter. We can add a feature to mobile phones which permits parents to see their children position on a map, and it will give a warning when children go somewhere else parents don’t want.

A person may need to learn some information around such as what is where, where to find a good shopping center, a cinema, etc. With the developing technologies after now, providing environment knowledge of a person is a requirement. This information may be about advertisements and campaigns of shopping places and social places, and other places where a person may need such as hospitals, police station, taxi rank, etc.

With seeing these problems we conclude such a feature is essential for mobile phones, that is seeing other people on a real world map, and communicate with other people by using this map featured environment.

2.2 Project Features

Our project will answer the problem we have defined previously. FIZAN will provide a map based environment to mobile phones.

We will do our project for Android OS based mobile phones. Why we have chosen Android is that, Android is a Google product and we think it will be more popular and will be used most of the mobile phones in the future. Since we want our product to be used for a long time Android will be good for us. Also developing programs in Android is simple, and it enables to use Java as programming language. And also some features of Android makes it better for us. First of all it is free and easy to use with Eclipse as a Java ME. Android provide good interfaces and enables to use Google products such as GTalk, Google Maps, etc. Beside all of these, since it is a new technology and a Google product, we think it will be widespread and a product done with Android will be valuable.

Some similar projects are done in foreign countries but not in our country. Our project will have very useful and enjoyable features for all people.

Our project is consisting of two parts: Client side and server side.

At client side there will be user interface, a database and classes to connect server (send and get data, call methods).

At server side we will keep our database and all information. Also we will keep a web page for to use all these features without a mobile phone on the internet.

In our user interface, there will be a map of surroundings and on it social places, advertisements, events and friends will appear as pop-ups. On the user interface, user will be able to set event, answer to an event, put and change photos in the facebook account of that user, look at his/her friends, and start instant messaging with them. When user clicks on a friend's pop-up on the interface, then he/she will be able to start instant messaging, see friend's information, facebook photo album. When a friend comes near system will send a message to warn us about this. Therefore FIZAN will make people really social.

Social places and shopping centers will be able to set advertisements and give information of their campaigns by contacting the GSM Company. But our program will be able to filter this information according to user wishes not to confuse user.

Project will have some main features:

Visualizing: We will show the real world on a map. We are thinking to take the map from Google Maps.

Showing friends: People which user has the permission to see will be visualized on the map by taking their location information (It will be taken from GSM company or phone cells GPS). When user chooses a person several actions will be ready to be chosen (Start instant messaging, see photos, etc.) .

Instant messaging: User will be able to start instantly messaging with another user on clicking the pop-up on the map. We will do this by GTalk.

Giving advertisement: Some social places and shopping centers will give advertisements and information about their campaigns which will be shown on the map.

Modes: There will be modes for users to switch which will increase the functionality and range of people that product able to reach.

Client & server side: Program will communicate with database and other mobile phones via this feature.

Social Networks: Product will communicate the social networks such as Facebook. User can directly communicate with these kinds of services, send and get information.

Web part: Almost all of the features that cell-phones side has in the project will be added to web.

Lost phone: User will login to his/her own account and will be able to see location of the phone.

Warning: According to settings a friend's coming near will warn user with a simple message.

2.3 Design Goals & Objectives

2.3.1 Usability

Our project will be very easy to use because it is a mobile phone technology which we want all people with a mobile phone can use this product. Therefore it has to have a usage which is very clear for everyone. Also we will provide some modes which will enable or disable some features of the product that will make usage simpler for those who cannot use all features.

2.3.2 Security and Privacy

Since if everybody sees all other people on the map and all other people's information such as where he/she is at the moment, it will be bad for everyone. Therefore we will protect users' information from other users although that user gives some permission to see some kind of information. That is a user can see another user if only other user has given permission that user to see him/her. Also user will be able to hide himself/herself on the map at any time; therefore our product secures users' private life in daily life. However we think to give permission parents to see their children at any time, but it may be optional too.

2.3.3 Availability

Our product will be easy to get. People will be able to get it from the internet by downloading to the personal mobile phone, that simple.

2.4 Design Problems

2.4.1 Time Problems

Our project's finish time is determined by ceng490 syllabus. Detailed schedule is given in the schedule part.

2.4.2 Software Constraints & Requirements

We will write our project mainly in Java, to the mobile phones with Android OS, using Eclipse. These are our main software requirements:

Web Server (JSP): It is Java technology that allows software developers to dynamically generate HTML, XML or other types of documents in response to a Web client request.

DBMS (MySQL): MySQL is a powerful database management system which we can be used to manage and store our data in the server side.

Eclipse: It is an open source IDE which makes possible and easier to develop codes for Android.

Android OS: An operating system for mobile phones which enables to produce programs for mobile phones.

GTalk: It is a Google product for instant messaging and voice over internet protocol (VOIP).

SQLite: It will be used for database on mobile phones.

3 User Interface Design

3.1 GUI Design Principles

We have designed our GUI, according to the following 7 basic principle

- Don't be different
- Keep it simple
- Look professional
- Be direct
- Be consistent
- Give cues and feedback
- Forgive mistakes

3.2 ScreenShots

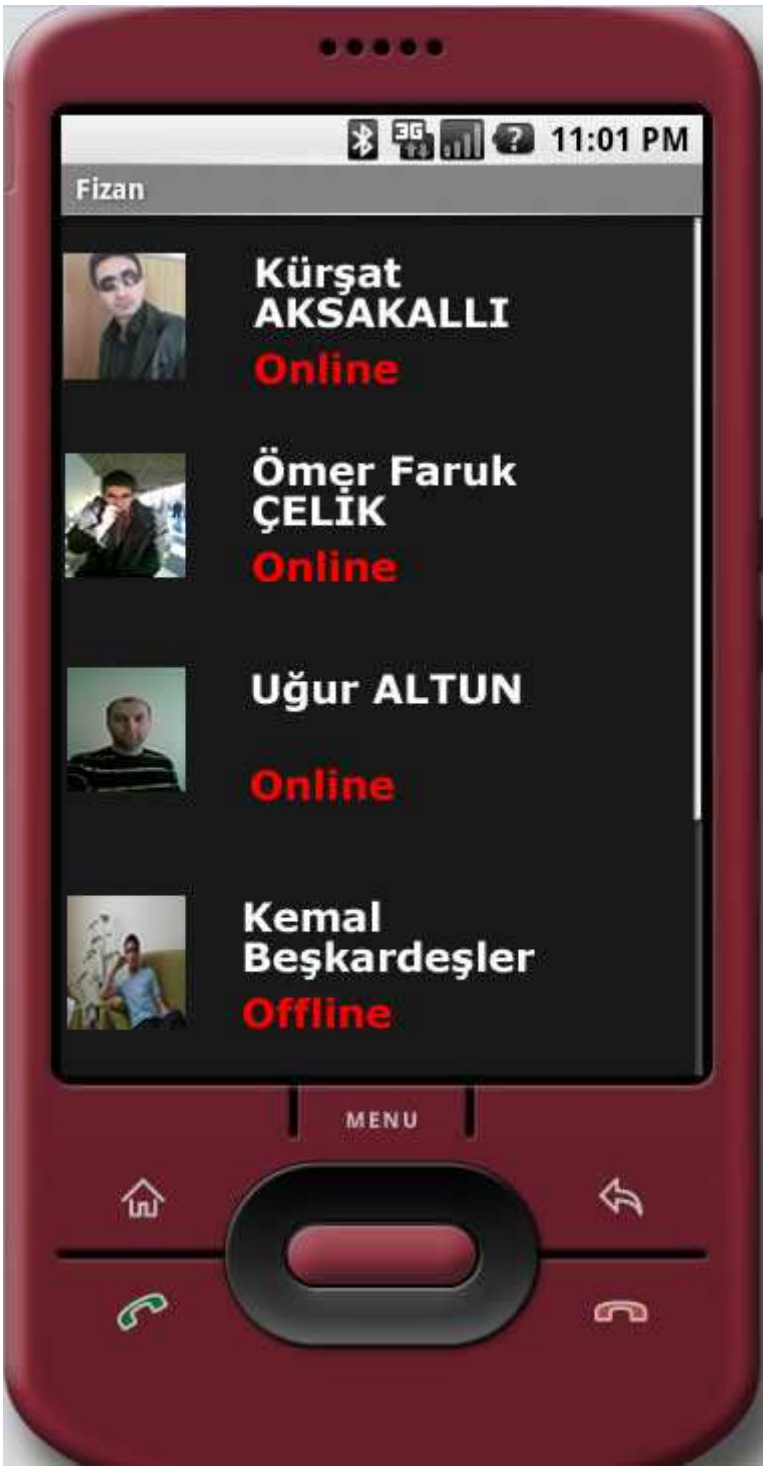
Some of the screenshots below is from the application and some of them are not created yet but design for only showing purposes. They will be created further phases.

3.2.1 Opening

3.2.2 Menu

These are opening screen and menu screen in the application. Opening works for login issues and Menu works for navigation.

3.2.3 List



3.2.4 Map

3.2.5 Friends

Here is a map in the application we should see friends icon on them. Friend interface will be opened when we choose one of the friends and gives information about the friend or able start instant messaging. And also list interface will show our friends as a list and give information about status.

4. Data Design

In the project Fizan, Users, Friends, Events should be kept in database. But tables in database will be required in both Server side and Client side. Then, according to efficiency issues, we kept whole data in server side database and required parts in cell-phone side.

4.1 Server Database Tables

Users:

In server side, all the users and user's data will be hold in the database in the name of "Users" table. This table includes all the information that belongs to user.

- **User_id:** All users have unique id,
- **User_name:** Name of the user**,** has default value " ".
- **User_passwd:** Password for the user, has default value "1234"
- **User_gmail_account name:** User's gmail account
- **User_gmail_pass:** User's gmail password
- **User_facebook_mail_address:** mail address used in Facebook.
- **User_facebook_ pass:** password used in Facebook
- **User_tel_num:** telephone number of the user
- **User_address:** Open address of the user
- **User_country:** Name of the Country
- **User_data_frequency:** time gap between the location data retrievals.
- **User_city:** Name of the city
- **User_status:** Status for Fizan (1: busy, 2: available etc.)
- **User_parents_tel_no:** Phone number of parents which can be used in parent Mode
- **User_log_file_name:** Name of the log file which track of the user kept
- **User_border_point1_x:** Latitude of first point which will be used in defining borders of the parent mode.
- **User_border_point1_y:** Longitude of first point which will be used in defining borders of the parent mode.
- **User_border_point2_x:** Latitude of second point which will be used in defining borders of the parent mode.
- **User_border_point2_y:** Longitude of second point which will be used in defining borders of the parent mode.
- **User_avatar:** File name of the avatar picture.
- **User_last_online_time:** Date of the last online time

User_id	INTEGER
User_name	STRING
User_passwd	VARCHAR(20)
User_gmail_account name	VARCHAR(100)
User_gmail_pass	VARCHAR(20)
User_facebook_mail_a dddress	VARCHAR(100)
User_facebook_ pass	VARCHAR(10)
User_tel_num	VARCHAR(20)
User_address	VARCHAR(500)
User_country	VARCHAR(30)
User_status	INTEGER
User_city	VARCHAR(30)
User_data_frequency	INTEGER
User_parents_tel_no	VARCHAR(20)
User_border_point1_x	INTEGER
User_border_point1_y	INTEGER
User_border_point2_x	INTEGER
User_border_point2_y	INTEGER
User_avatar	VARCHAR(50)
User_last_online_time	Date
User_log_file_name	VARCHAR(50)
PrimaryKey(User_id)	
Constraint NOT NULL (User_passwd), NOT NULL (User_tel_num)	
NOT NULL (User_name)	

Friends:

Friend list will be hold in this table. For each of the users all of the friends will be kept here.

- **Fri_id1:** Who has the friends
- **Fri_id2:** User with id1's friend
- **Fri_type:** `How user whom has id1 wants to appear to user with id2. (1 for Online, 2 for busy, 3 for offline)
- **Fri_dist:** What is the alert distance for id1 and id2
- **Fri_online:** Is id2 online ?

Fri_id1	INTEGER
Fri_id2	INTEGER
Fri_type	INTEGER
Fri_dist	INTEGER
Fri_online	INTEGER
PrimaryKey(Fri_id1, Fri_id2)	
ForeignKey(Fri_id1), ForeignKey(Fri_id2) references Users	

Events:

This table holds the detailed data for events. Therefore events can be specified and classified.

- **event_id:** Unique id of the event
- **event_cat:** Category of the event (i.e. 1 for entartainment, 2 for shopping…)
- **event_start:** Day of starting
- **event_finish:** Day of finishing
- **event_time:** Time of the event if any (i.e. Concert time)
- **event_fare:** Cost of the event if any (i.e. Concert fee)
- **event_owner:** Name of the owner of the event (maybe company name)
- **event_loc1:** Latitude of the location
- **event_loc2:** Longitude of the location
- **event_region:** Local name of the event (i.e. Kizilay)
- **event_city:** Name of the City
- **event_off_addr:** Formal address
- **event_unoff_addr:** Informal address
- **event_tel:** phone number related to event
- **event_link:** Web link for detailed info
- **event_desc:** General info for event or slogan for advertisement
- **event_foto_name:** Name of the photo which will be displayed
- **event_view_type:** numbered type of the event view (like 1 for just photo or 2 for photo with description)

event_id	INTEGER
event_cat	INTEGER
event_start	Date
event_finish	Date

event_time	Date
event_fare	INTEGER
event_owner	VARCHAR(100)
event_loc1	INTEGER
event_loc2	INTEGER
event_region	VARCHAR(50)
event_city	VARCHAR(50)
event_off_addr	VARCHAR(300)
event_unoff_addr	VARCHAR(300)
event_tel	VARCHAR(20)
event_link	VARCHAR(100)
event_desc	VARCHAR(300)
event_foto_name	VARCHAR(30)
event_view_type	INTEGER
PrimaryKey(event_id)	
Constraints NOTNULL (event_cat), NOTNULL(event_start),	
NOTNULL(event_finish), NOTNULL(event_loc1), NOT NULL(event_loc2)	

User-Event Settings:

This table holds the setting information for user about events. User-Event Settings table can be combined with Users table but for the performance issues it is separated.

Setting_user_id: Id of the user
Setting_event_type: Category of the wanted event
Setting_event_dist: Maximum distance of the event.
Setting_event_interv: Time gap between re-controlling the event. (in minutes)

Setting_user_id	INTEGER
Setting_event_type	INTEGER
Setting_event_dist	INTEGER
Setting_event_interv	INTEGER
PrimaryKey(user_id, even_type)	
ForeignKey(user_id) references Users	

Event-Attendance:

In this table user's events are kept and also attendance information.

- **User_id:** Id of the user
- **Event_id:** Id of the Event
- **Attend_status:** Attending status (1 for Yes, 2 for Maybe, 3 for No)

User_id	**INTEGER**
Event_id	**INTEGER**
Attend_status	**INTEGER**
PrimaryKey(User_id, Event_id)	
ForeignKey(User_id) references Users, ForeignKey(Event_id) refererences Events	

4.2 Client Database Tables

Android has SQLite for database tools. Since SQLite is a weak database tool, database tables in Cell-Phone side should be lighter. Moreover log records will be kept in a file not database.

Friends:

- **Fri_id:** Id of the friends.
- **Fri_type:** `How user wants to appear to his/her friend. (1 for Online, 2 for busy, 3 for offline)
- **Fri_dist:** What is the alert distance for user and friend
- **Fri_online:** Friends status
- **Fri_gmail:** Gmail account of the friend.
- **Fri_facebook_account:** Facebook account name of the friend

Fri_id	INTEGER
Fri_type	INTEGER
Fri_dist	INTEGER
Fri_online	INTEGER
Fri_gmail	VARCHAR(100)
Fri_facebook_account	VARCHAR(100)
PrimaryKey(fri_id)	

Events:

This table holds the detailed data for events, which user wants to display on the screen. Therefore events can be specified and classified.

- **event_id:** unique id for event.
- **event_cat:** Category for the event.
- **event_str:** Starting time.
- **event_fin:** Finishing time.
- **event_time:** Occurring time of the event (if exists).
- **event_owner:** Name of the owner of the event. If it is commercial event, it can be name of the company.
- **event_loc1:** latitude of the event.
- **event_loc2:** longitude of the event.
- **event_addr:** official address of the event.
- **event_tel:** telephone no of the event owner.
- **event_link:** related web link of the event.
- **event_desc:** General description of the event.
- **event_photo:** Photo name of the event.
- **event_view_type:** numbered type of the event view (like 1 for just photo or 2 for photo with description).

event_id	INTEGER
event_cat	INTEGER
event_str	DATE
event_fin	DATE
event_time	DATE
event_owner	VARCHAR(100)
event_loc1	INTEGER
event_loc2	INTEGER
event_addr	VARCHAR(300)
event_tel	VARCHAR(15)
event_link	VARCHAR(300)
event_desc	VARCHAR(50)
event_photo	VARCHAR(50)
event_view_type	INTEGER
PrimaryKey(event_id)	

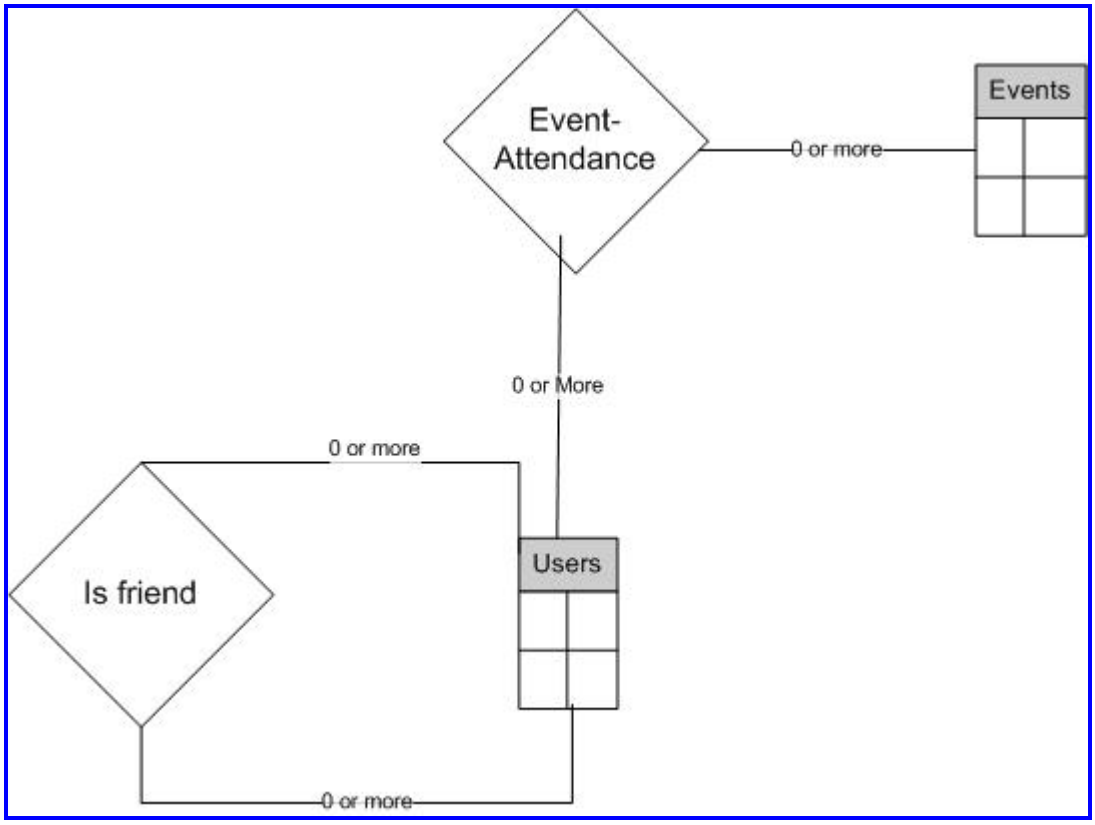
Photos:

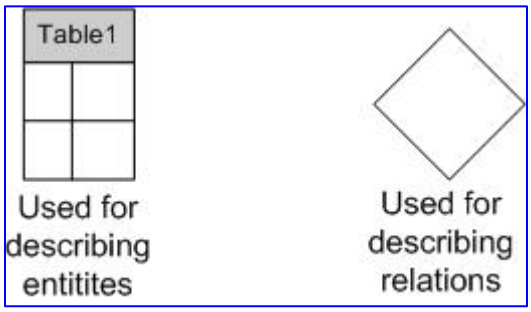
Keeps the record of the photos that added to the application.

- **photo_id:** There is a unique Id for every photo.
- **Owner_name:** Name of the owner
- **photo_type:** Where is it belongs to (Enumerated with Personal, Friends, Facebook, Events…)
- **photo_name:** Name of the photo

photo_id	INTEGER
owner_name	VARCHAR(100)
photo_type	INTEGER
photo_data	VARCHAR(100)
PrimaryKey(photo_id)	

4.3 ER DIAGRAMS

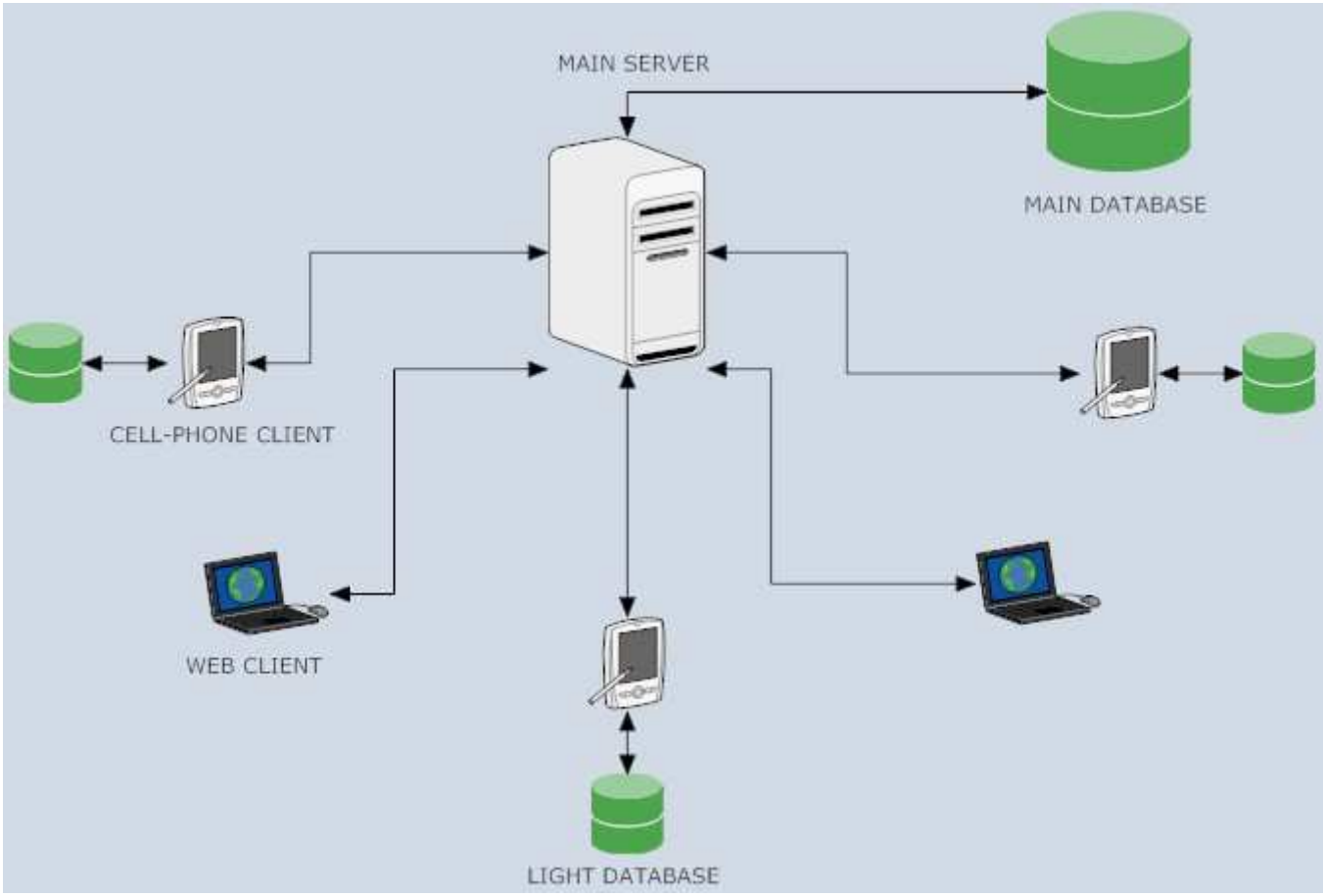




In The ER (Entity relationship) diagram above is for server side, there are two entities and 2 relations. First relation is “Is Friend” relation. This shows the relation between two users. And give the answer to the question “is users are friends or not”.

As can be seen clearly, second one shows the relation between users and events and gives idea about whether user attends the event or not.

5. Architectural Design



The current idea is, server does almost all the operations, fetching data from database, updating database, doing necessary calculation, getting data from clients and sending data to clients.

For the client side, Client sends request to the server and gets data from the server. So, Client side is not bothered with calculations. Generally, client shows the data coming from server on its user interface. For example, client sends friend list to server and wants to take location data from friends once an hour.

When user opens FIZAN the map of surrounding will come to his/her user interface. On the map his/her friends will appear as pop-ups. Client side will request peoples' locations from the server and after getting data, it will show on the map. According to settings of that mobile phone some other features will be shown such as advertisements and events.

When user clicks on a friends pop-up menu there will be some options such as view facebook photo album, start instant messaging and etc. When user wants to see photos of that friend, (if it is permitted) main server will connect to facebook and photos will be shown. And user also will be able to access his/her own facebook

account and put on or delete from photos his/her own album. If user wants to start instant messaging GTalk will be run on the server and this will be shown on the user interface.

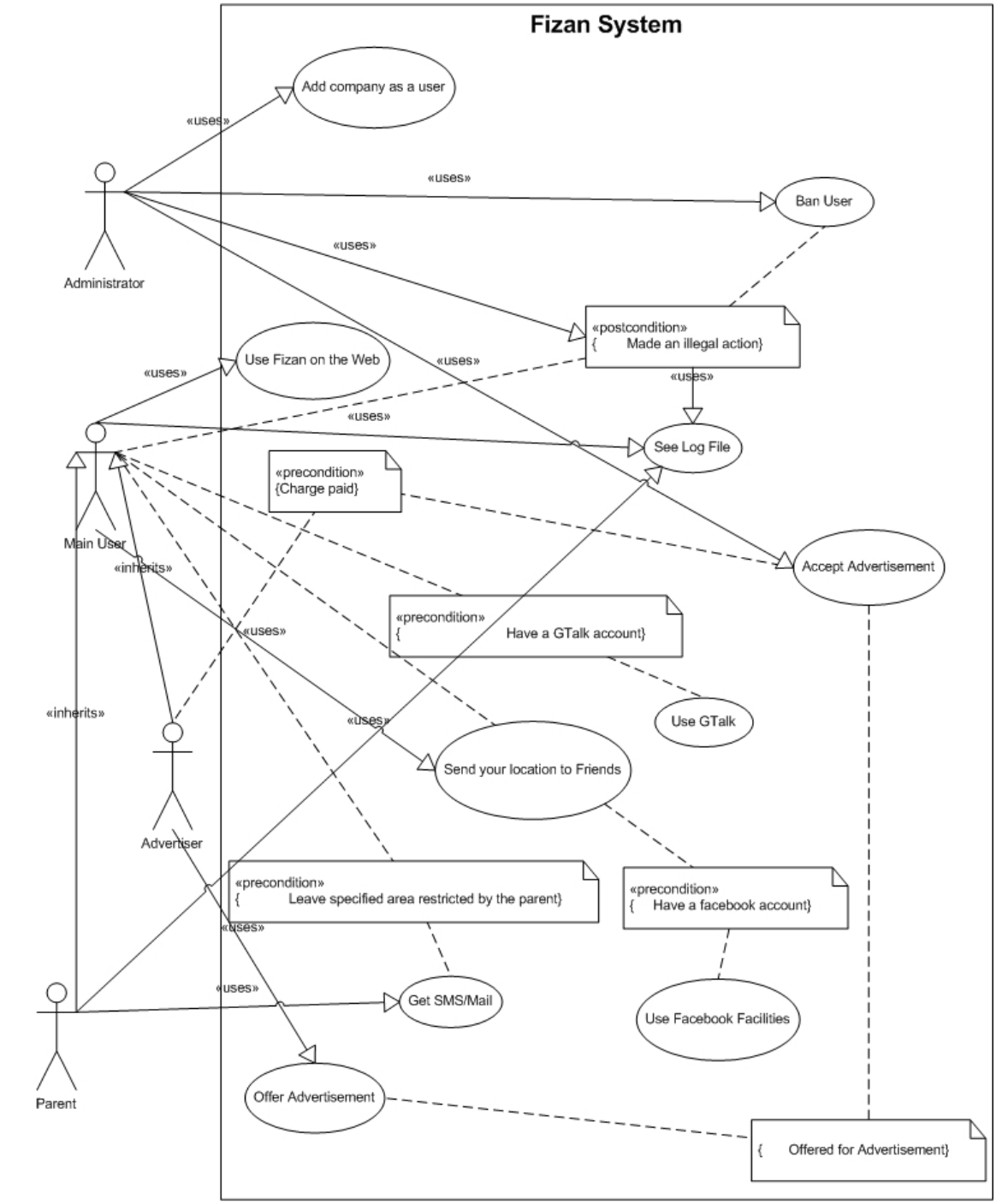
User will be able to activate some events on the map and information of content, to information and location information will be send to server, which server will send this information to related peoples clients databases. Related people of this event will see time, location and information about the event and also who declared this event and who participated to this event. Since information of mobile phones databases will be updated some time when a change occurs, user will be able to see it.

Client side will be responsible to visualize every required thing on the map and request data from server. Server is going to store data in global database and answer to client and update information according to client's wishes and responses. Server will control authorization, save data, send data, and will do most of the work.

6. SYSTEM DESIGN

6.1. Use Case Diagram

Administrator is a employee at the Fizan Company, controlling server and accounts.



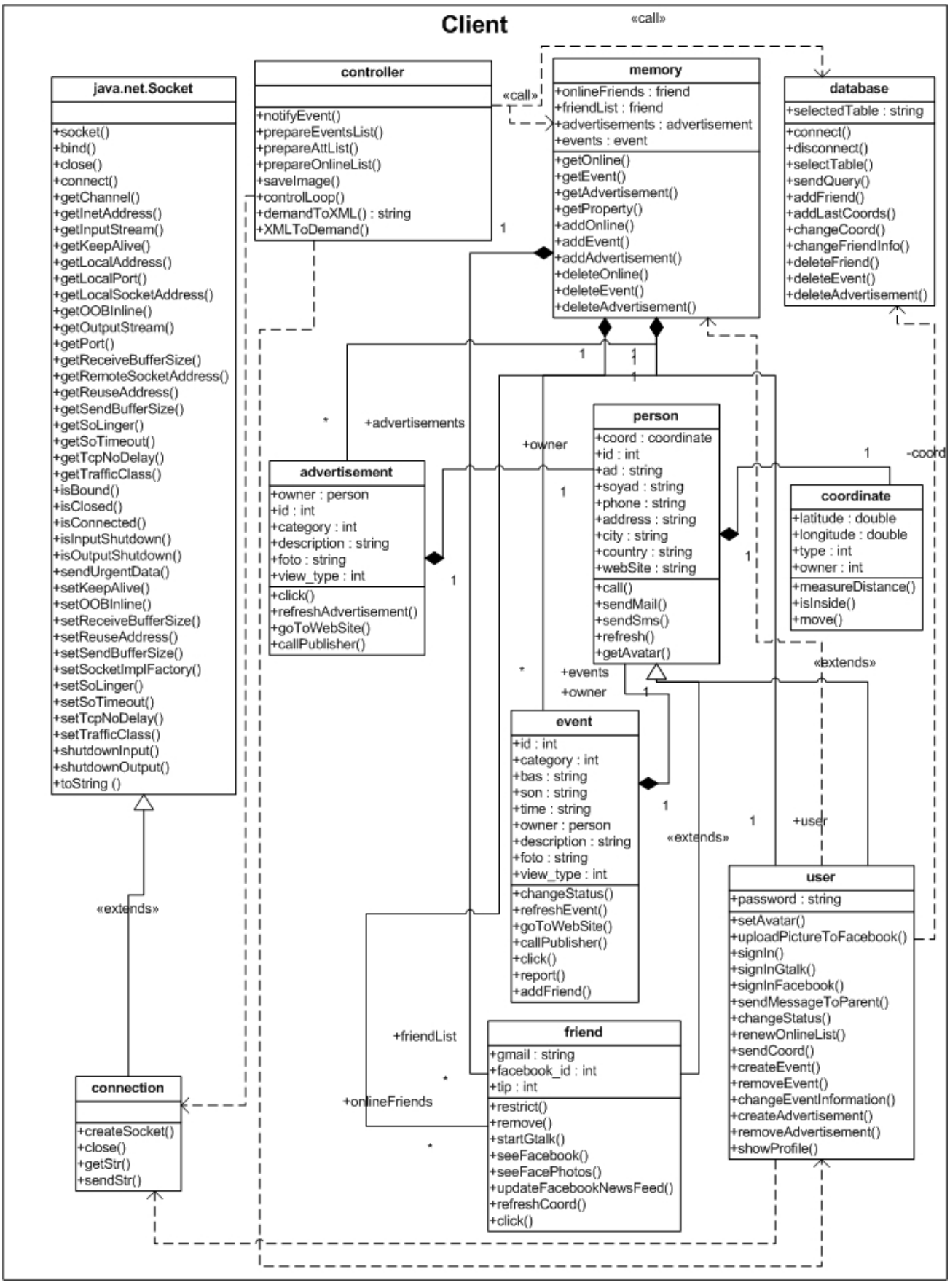
User may not have any parent, in this case user can use parent operations.

6.2 Classes and Class Diagram

Roughcast of the Classes and the Work Design

Fizan's code is divided into two, some to the server side and some to the client side. At the server side, majority of the process is database processes. At the server, there are some tables about situation which is being updated. These tables keep online people, informations about people, friend linked list, events, advertisements, settings, event attendance, photo informations, coordinates. When any of these has changed, client send information to the server and updated here. When there is any demand from clients, server sends that information. These demands will be periodically or manually. Client program takes data from server and inform the user. Briefly, data of the users are in the server database, and the client program takes other data related to it from the server and after design, serves to user. Controller classes at the both side works and uses other classes. Pool's main loop method control aal of the things not belong to the clients. At server side, for every connection with the clients, there is a contoller class, their controlLoop works in threads.

6.2.1 Client Class Diagram



Client Classes:

connection: This class communicates with the server side communication class. When any demand occurs in the program activated by the user, this class gets instruction from other client classes, and sends information to the server. This class serves as an interface to the client classes. This class listens server's connection class to take new informations immediately.

Methods:

createSocket: Creates socket.
close: Closes socket.
getStr: Gets string from other side.
sendStr: Sends data to other side.

controller: Understands what is received from server and makes some file operations and provides display.

Methods:

notifyEvent: Notifies the user about an event of his/her friend and asks to read/ignore.
prepareEventsList: Prepares events/advertisements list that the server responded to user's demand.
prepareAttList: Prepares event.friendsAttList/friendsMaybeAttList.
prepareOnlineList: Prepares online friends list.
saveImage: Saves received image to a directory/file.
controlLoop: This method is available when fizan is processing in the phone. It waits for responses coming from the user or server, when a response comes it calls methods accordingly to control requests.
demandToXML: Converts demands(i.e what information is wanted, like online friends list) to XML strings.
XMLToDemand: Converts XML strings to objects.
changeScreen: Changes the screen layout to desired one according to the demands.

person: It keeps general personal information. id, name, avatar, phone number, address, mail address, web site, coordinate are data kept in this class.

Methods:

call: Calls the person.
sendMail: Sends mail to the person.
sendSms: Sends sms to the person.
refresh: Takes new information about friend's account from server.
getAvatar: Takes avatar picture of the person.

friend: This class extends person class. It has some additional data about friends and additional methods. Gmail(keeps gmail account),facebook.

Methods:

restrict: Change status and restrict friend.

remove: Remove friend from friends list.
startGtalk: Starts a conversation between user and friend using gtalk.
seeFacebook: open web gadget to see friend's facebook page.
seeFacePhotos: Go to the web page, which shows photos of the friend.
updateFacebookNewsFeed: Gets unread news of friend from facebook.
refreshCoord: Gets new coordinate values of the friend.
click: Displays information when friend clicked on the screen.

user: This class extends person class.

Methods:

setAvatar: Sends avatar picture to the server to set avatar.
uploadPictureToFacebook: Uploads a picture to specific facebook album.
signIn: Sign in to fizan.
signInGtalk: Sign in to the Gtalk.
signInFacebook: Sign in to the facebook and goto facebook home page.
sendMessageToParent: Sends a message to the parent.
changeStatus: Change user status and sends information to the server.
renewOnlineList: Renew online list.
sendCoord: Sends current coordinate to the server.
createEvent: Displays a window, in which you can adjust event's information, select friends to be notified,etc. And after completing final processes, necessary operations will be completed on the phone and on the server.
removeEvent: Remove any created event. (as create event)
changeEventInformation: Opens event's information dialog to be changed by the user.
createAdvertisement: Create an advertisement in a similar way of creating an event.
removeAdvertisement: Remove any created advertisement.
changeAdvertisementInformation: Opens advertisement's information dialog to be changed by the user.
showProfile: Displays the user's own profile.

event: This class keeps event informations. friendsAttList(attending event), friendsMaybeAttList(maybe attending event)

Methods:

changeStatus: Changes attendance status of the user for this event.
refreshEvent: Gets event information.
goToWebSite: If website of that event is declared, goto web site via web browser.
callPublisher: Call event's owner.
click: Displays event information on the screen.
report: Prepares report about the event and sends it to the admins.

addFriend: Adds friend to the list of friends who will attend the event (discrete list is used for people whose attending status is set as "maybe").

advertisement: This class keeps information about advertisements and when client want to make any process about advertisement, this class will be used.

Methods:

click: Displays advertisement information on the screen.
refreshAdvertisement: Gets advertisement informations.
goToWebSite: If website of that advertisement exists, goto website via web browser.
callPublisher: Call advertisement's owner.

coordinate: Keeps information of the coordinates. Latitude and longitude are kept in this class.

Methods:

measureDistance: Measures distance between other objects.
isInside: Returns whether current coordinate is inside the rectangle, provided as a parameters, or not.
move: Change coordinates references to saved coordinates.

memory: This class is used to fetch and write back to memory (into created objects) about anything received from the server. It arranges memory.

Methods:

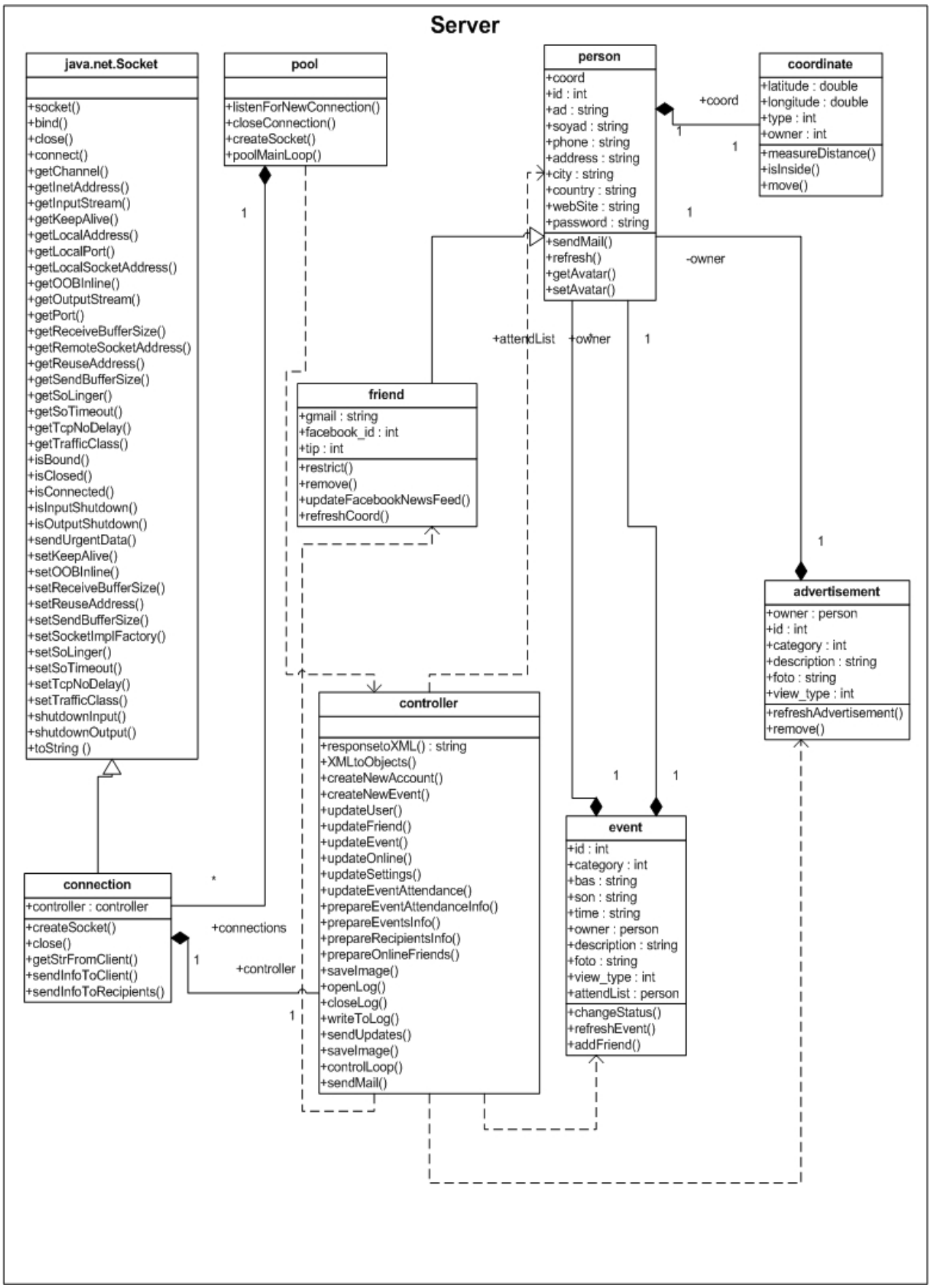
getOnline: Get online person from online list and call "addOnline" method.
getEvent: Get the event (clicked on the screen) from events vector.
getAdvertisement: Get the advertisement (clicked on the screen) from advertisements vector.
getProperty: Gets a specified property from vector, according to parameters.
addOnline: Adds a person to online list (Vector of person objects).
addEvent: Adds an event to event list (Vector of event objects).
addAdvertisement: Adds an advertisement to the list (Vector of advertisement objects).
deleteOnline: Deletes person from online list.
deleteEvent: Deletes event from events list.
deleteAdvertisement: Deletes advertisement from advertisements list.

database: This class used to make database usage easier(In Android, SQLite is our database).

connect: Connect to database.
disconnect: Disconnect from the database.
selectTable: Select a specific table.
sendQuery: Sends specific query which cannot have any method in this class.

addFriend: Adds a friend to friends table.
addLastCoords: Adds a friend coordinate to the table who is recently became online.
changeCoord: Changes the coordinate of a person.
changeFriendInfo: Changes personal informations of a friend.
deleteFriend: Deletes a friend from the friends table.
deleteEvent: Deletes an event from events table.
deleteAdvertisement: Deletes an advertisement from events table.

6.2.2 Server Class Diagram



Server Classes:

pool: This class keeps a vector of connection objects and has methods to manage network traffic. It waits a connection, if occurs, create a new socket with it, and add it to the list.

Methods:

listenForNewConnection: Creates a connection object, adds it to the vector of connections. It will be a Java listener for coming connections (to be changed). When new connection created, its controller started to serve in a thread.
closeConnection: If connection between server and client breaks in any way, this method will be called.
createSocket: Creates socket, add it to the vector.
poolMainLoop: Processes instructions about server, not about clients' demands.

connection: This class communicates with the communicate class of the client side. It constructs a network connection as a server with the other side. According to the demands coming from the clients, this class communicates with other server classes.It fetches information from the database or files, and sends desired information to the connection class of the client; or it updates information in tables or files. It serves as an interface to the server classes.

Methods:

createController: Creates a controller object.
sendInfoToClient: Sends prepared information to the client.
sendInfoToRecipients: Informs determined recipients about an event.
close: Closes socket
getStrFromClient: Gets string from other side.

controller: Understands what the client wants and organizes data, received from client to connection object, for database and file operations.

Methods:

responsetoXML: Converts demands to XML strings.
XMLtoObjects: Converts XML strings to objects.
createNewAccount: Creates a new account, assigns a user id, adds user to "Kullanicilar" table.
createNewEvent: Creates a new event/advertisement, assignes an event/advertisement id, adds event/advertisement to "Events" table.
updateUser: Update user information in "Kullanicilar" table.
updateFriend: Update friend information in "Arkadaslar" table.
updateEvent: Update event information in "Events" table.

updateOnline: Update being online/offline information in “Online” table and takes coordinates.
updateSettings: Update settings in “Settings” table.
updateEventAttendance: Update event attendance status in “Event_Attendance” table.
prepareEventAttendanceInfo: Prepares friends list attending an event.
prepareEventsInfo: Prepares filtered events/advertisement package according to the desired properties to be sent to the client.
prepareRecipientsInfo: Prepares event information with recipients declared.
prepareOnlineFriends: Prepares online friends information within declared range.
saveImage: Saves received image to the related directory and assigns a photo id.
openLog: Opens log file to write.
closeLog: Close Log file.
writeToLog: Writes data to the log file.
sendUpdates: Sends updated informations to the user.
saveImage: Saves image to a appropriate directory, names it and change settings for it.
controlLoop: This method is available when this class is in memory. It waits for responses, when it comes it uses its own classes methods’ to control requests.
sendMail: Sends mail.

person: This class keeps only datas about the person, and manages these data.

Methods:

sendMail: Sends mail to specified address using mail server.
refresh: Refreshes personal informations and send it to the client.
getAvatar: Gets address of the avatar picture’s location.
setAvatar: Loads picture and update database and memory for the new avatar information.

friend: This class is child of the person class. It keeps extra data about friends, which user can reach.

Methods:

restrict: Restricts any friend.
remove: Removes some friend from friend list.
updateFacebookNewsFeed: Updates facebook news for the friend, to learn is there any new activity for the user.
refreshCoord: Refreshes coordinate of the friend, and sends it to the client.

event: This class keeps only datas about the event, and manages these data.

Methods:

changeStatus: Changes attendance status for specified user.
refreshEvent: Refreshes event information, and send it to the specified client.
addFriend: After coming any information of attending to this event, sends new friend to the owner of the event.

advertisement: This class keeps only data about the advertisement, and manages these data.

Methods:

refreshAdvertisement: Refreshes advertisement information, and send them to the specified user.
remove: Removes advertisement from the server.

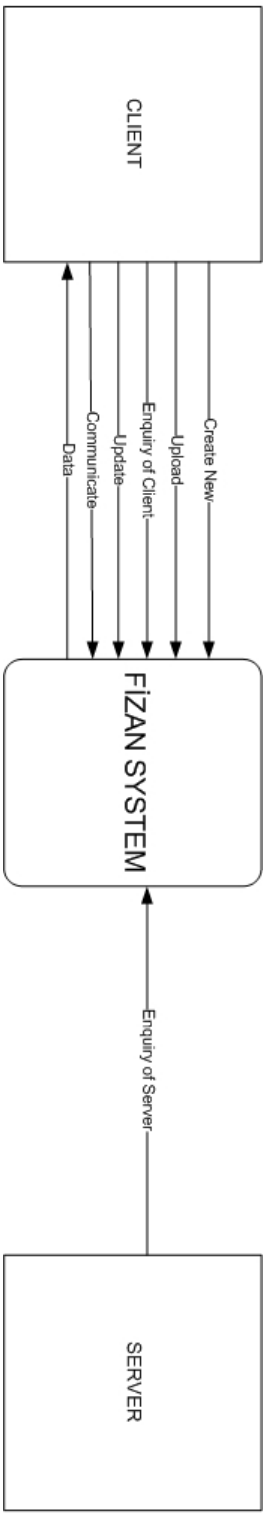
coordinate: This class keeps only datas about the coordinates, and manages these data.

Methods:

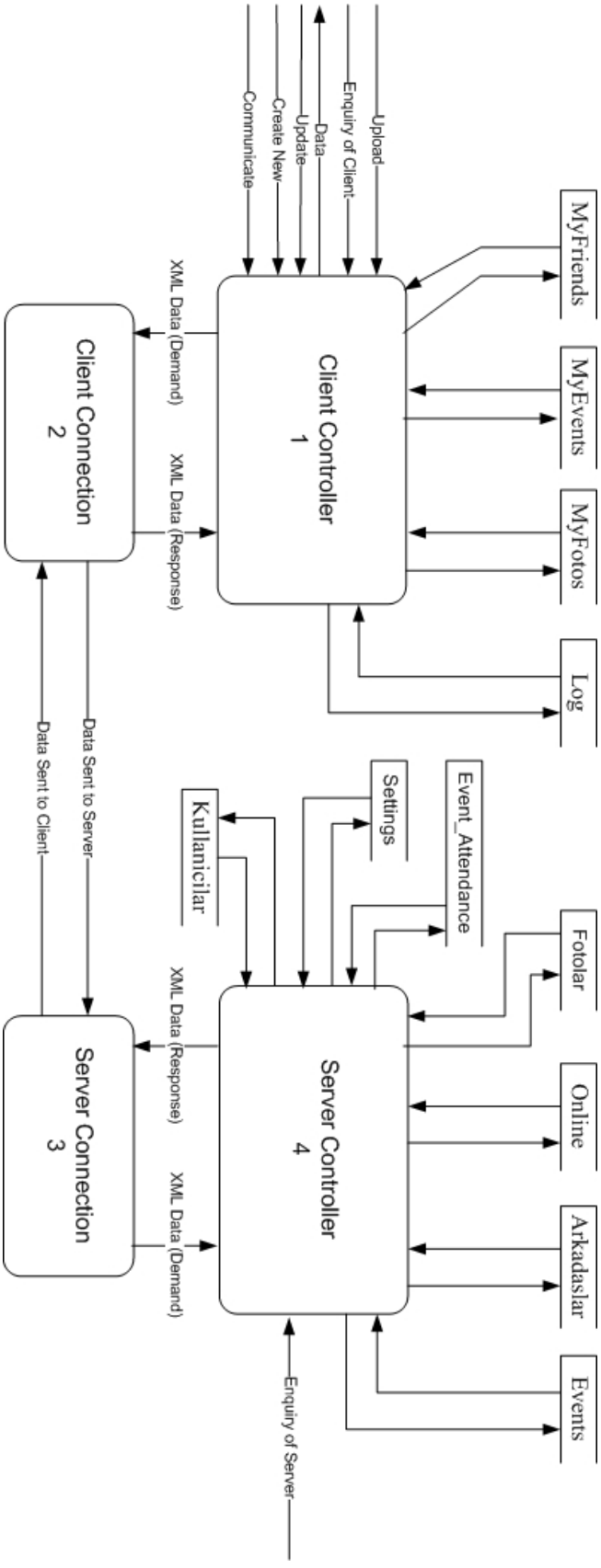
measureDistance: Measures distance between other objects.
isInside: Is our coordinate inside the rectangle coming as a parameter.
move: Change coordinates referenced to saved coordinates.

6.3 Data Flow Diagrams

6.3.1 Context Level Diagram

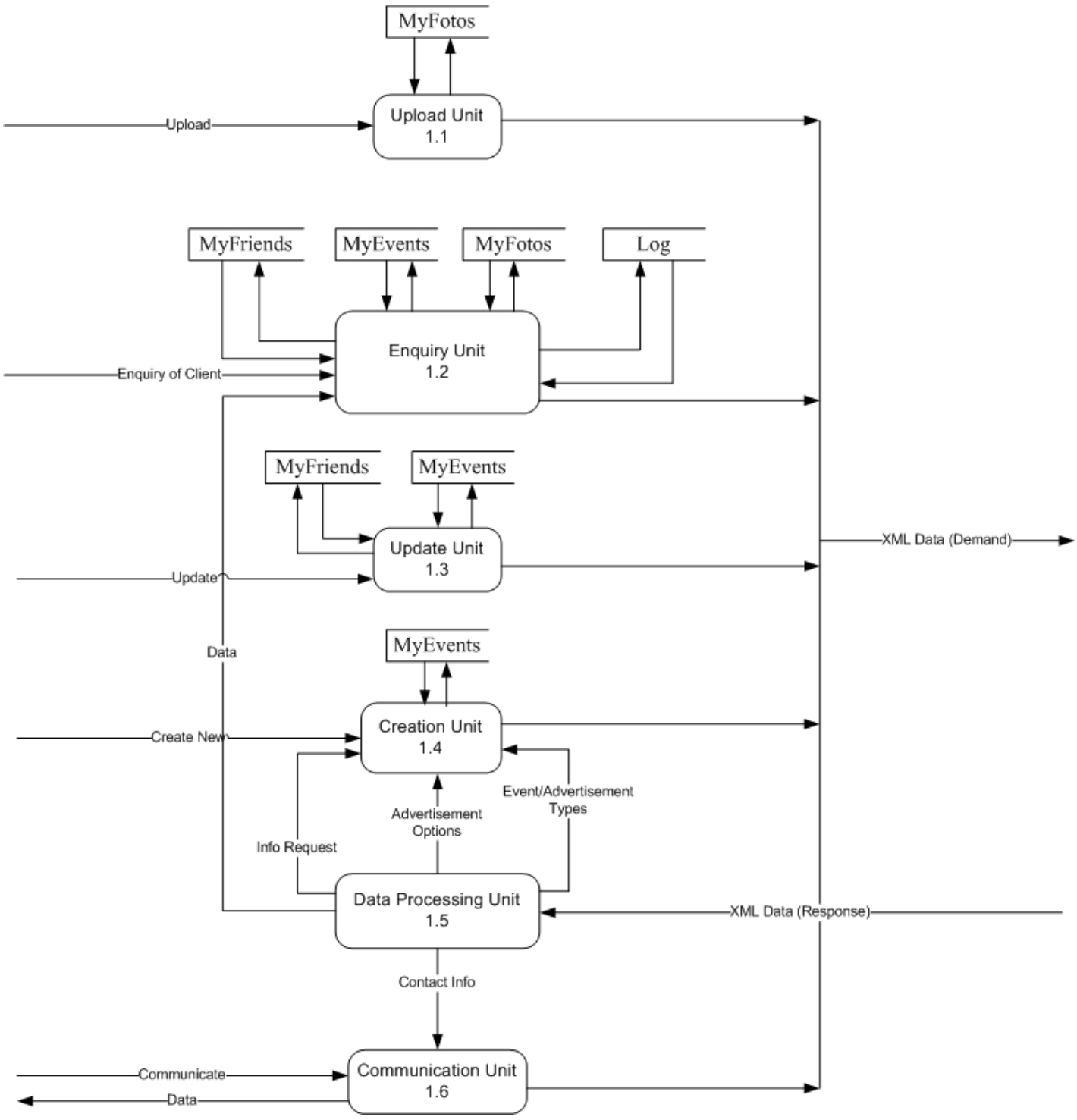


6.3.2 Level0 DFD

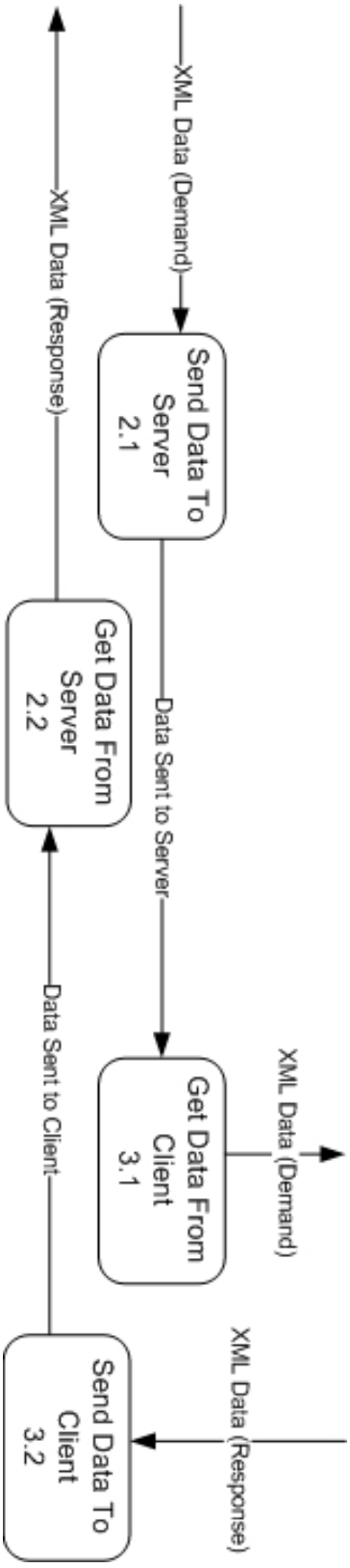


6.3.3 Level1 DFD

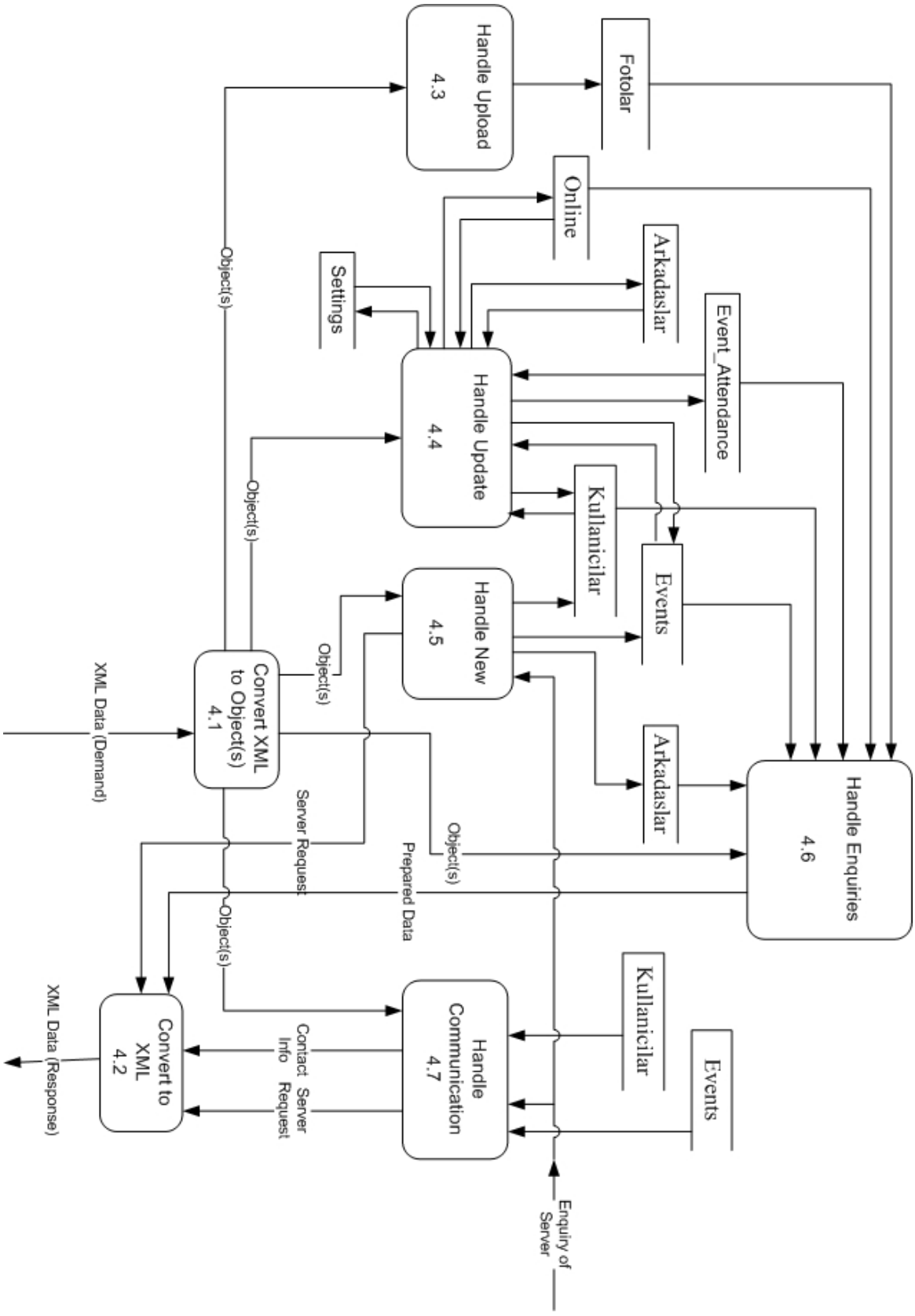
1. Client Controller



2. Client Connection & Server Connection

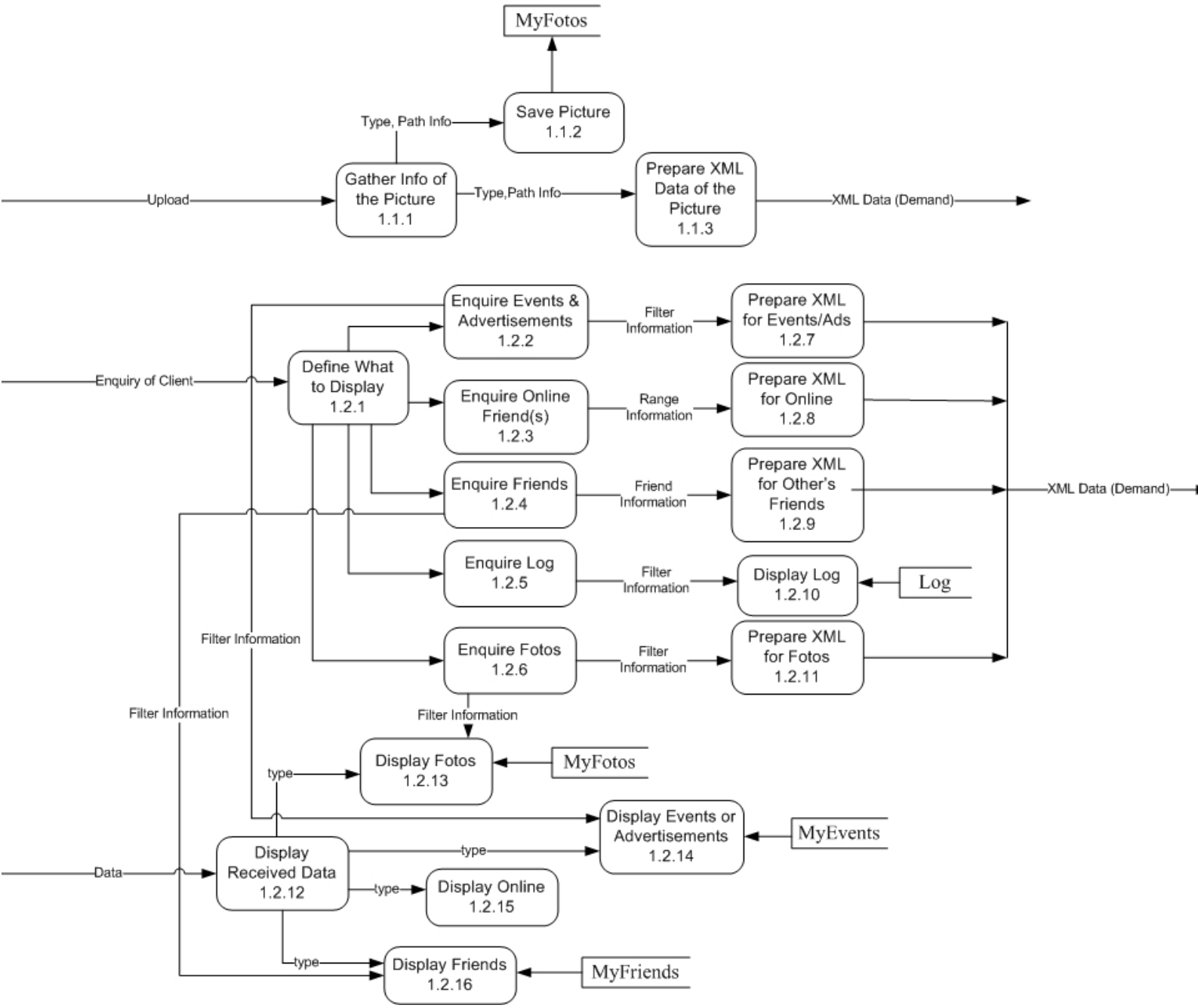


3. Server Controller



6.3.4 Level2 DFD

1. Client's Upload Unit & Enquiry Unit



Name	Type, Path Info
From	Gather Info of the Picture 1.1.1
To	Save Picture 1.1.2 & Prepare XML Data of the Picture 1.1.3
Description	Includes type and path information of the picture

Name	Filter Information
From	Enquire Events & Advertisements 1.2.2
To	Prepare XML for Events/Ads 1.2.7 & Display Events or Advertisements 1.2.14
Description	Includes information of what to filter from Events or MyEvents table

Name	Range Information
From	Enquire Online Friend(s) 1.2.3
To	Prepare XML for Online 1.2.8
Description	Includes range information to be filtered from Online Friends

Name	Friend Information
From	Enquire Friend(s) 1.2.4
To	Prepare XML for Other's Friends 1.2.9
Description	Includes friend information (i.e. name, id) to be filtered from Arkadaslar table

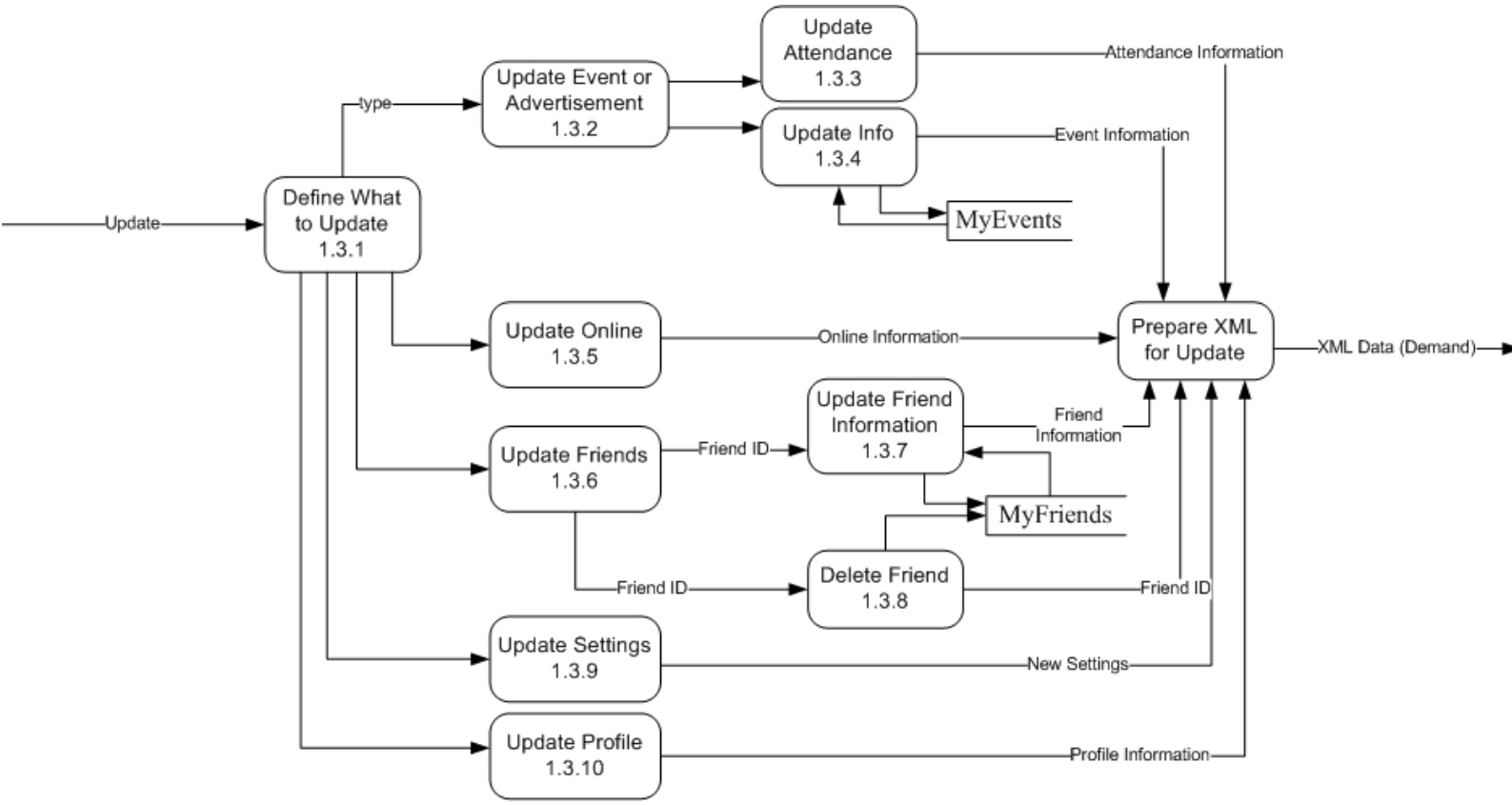
Name	Filter Information
From	Enquire Friend(s) 1.2.4
To	Display Friends 1.2.16
Description	Includes friend information (i.e. name, id) to be filtered from MyFriends table

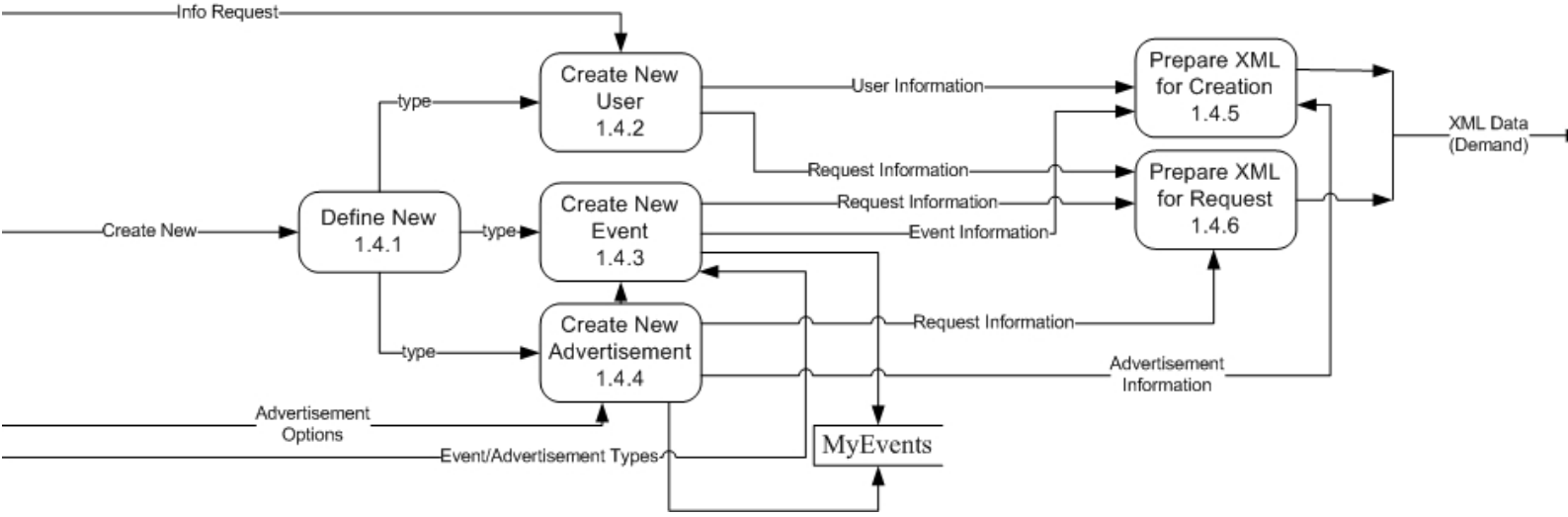
Name	Filter Information
From	Enquire Log 1.2.5
To	Display Log 1.2.10
Description	Includes log information (i.e. time) to be filtered from Log table

Name	Filter Information
From	Enquire Fotos 1.2.6 & Display Fotos 1.2.13
To	Prepare XML for Fotos 1.2.11
Description	Includes foto information (i.e. type) to be filtered from Fotolar or MyFotos table

Name	type
From	Display Received Data 1.2.12
To	Display Fotos 1.2.13 & Display Events or Advertisement 1.2.14 & Display Online 1.2.15 & Display Friends 1.2.16
Description	Includes type of received data

2. Client's Update Unit & Creation Unit





Name	type
From	Define What to Update 1.3.1
To	Update Event or Advertisement 1.3.2 & Update Online 1.3.5 & Update Friends 1.3.6 & Update Settings 1.3.9 & Update Profile 1.3.10
Description	Includes type of data to be updated

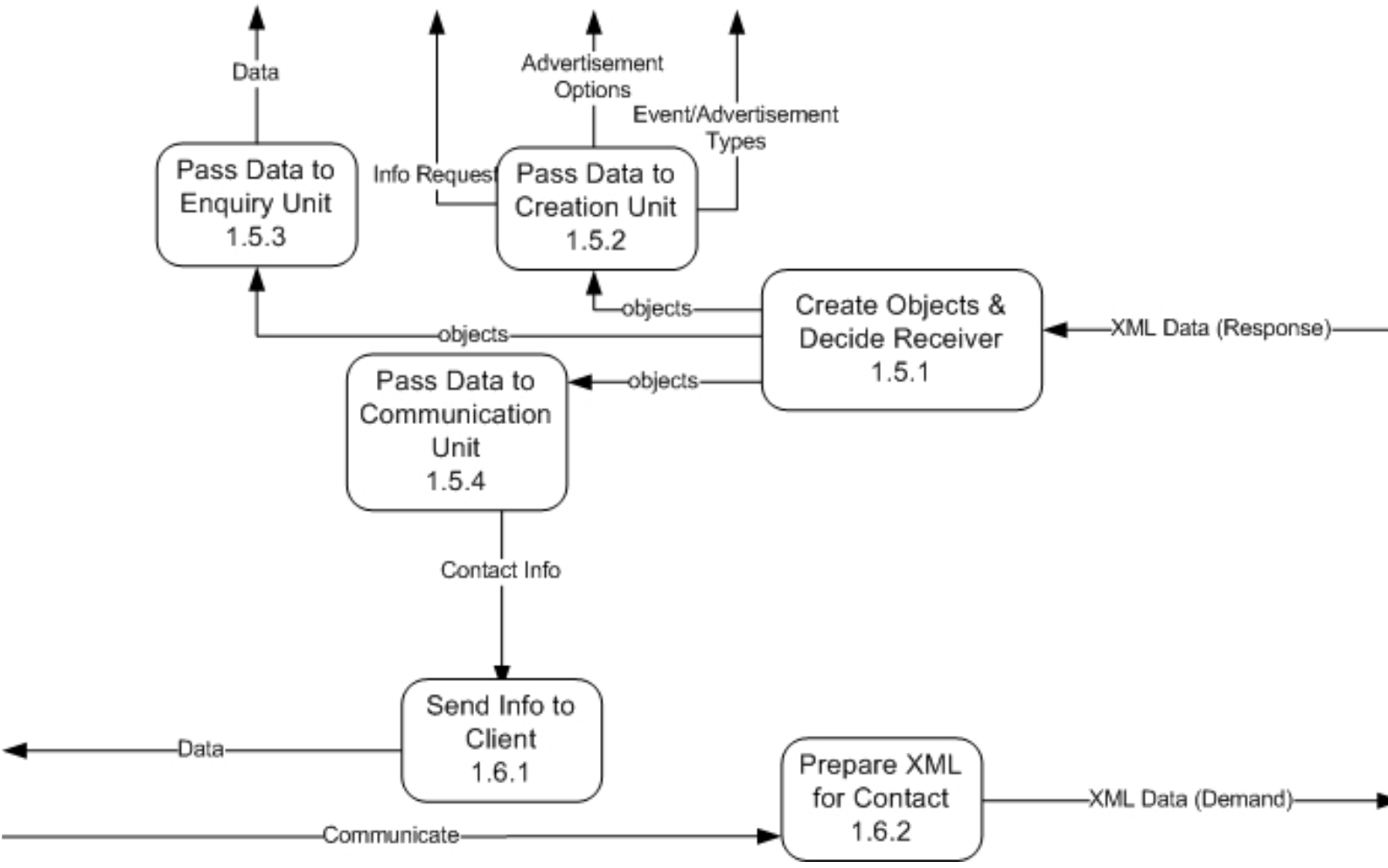
Name	type
From	Define New 1.4.1
To	Create New User 1.4.2 & Create New Event 1.4.3 & Create New Advertisement 1.4.4 &
Description	Includes type of data to be created

Name	User Information
From	Create New User 1.4.2
To	Prepare XML for Creation 1.4.5
Description	Includes full information of the user for creating new user

Name	Request Information
From	Create New User 1.4.2
To	Prepare XML for Request 1.4.6
Description	Includes basic information of the user for creating new user to be confirmed by the server

- These last two tables are similar for create new event/advertisement tools.

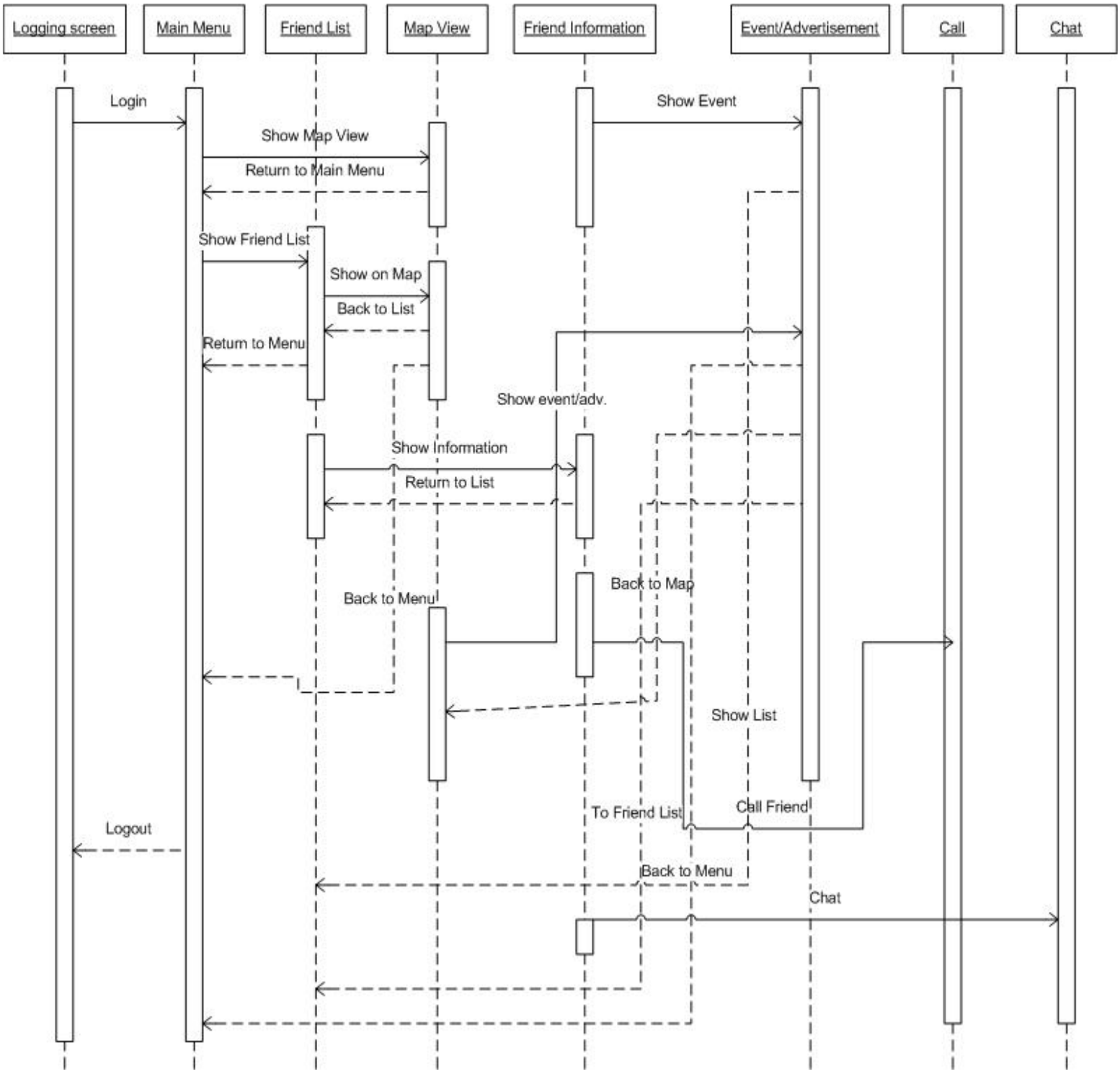
3. Client's Data Processing Unit & Communication Unit



Name	objects
From	Create Objects & Decide Receiver 1.5.1
To	Pass Data to Creation Unit 1.5.2 & Pass Data to Enquiry Unit 1.5.3 & Pass Data to Communication Unit 1.5.4
Description	Includes objects created from received XML data

Name	Contact Info
From	Pass Data to Communication Unit 1.5.4
To	Send Info to Client 1.6.1
Description	Includes contact information (i.e. phone no,e-mail) to be send to client

6.4 Sequence Diagram



We represented the time-method sequence relationship in this diagram. This is the sequence diagram for the whole cell-phone side. It shows the basic dynamic operations. It can be divided into parts and for each part we can draw a new diagram but we choose to draw it bodily.

Diagram shows the relations between the

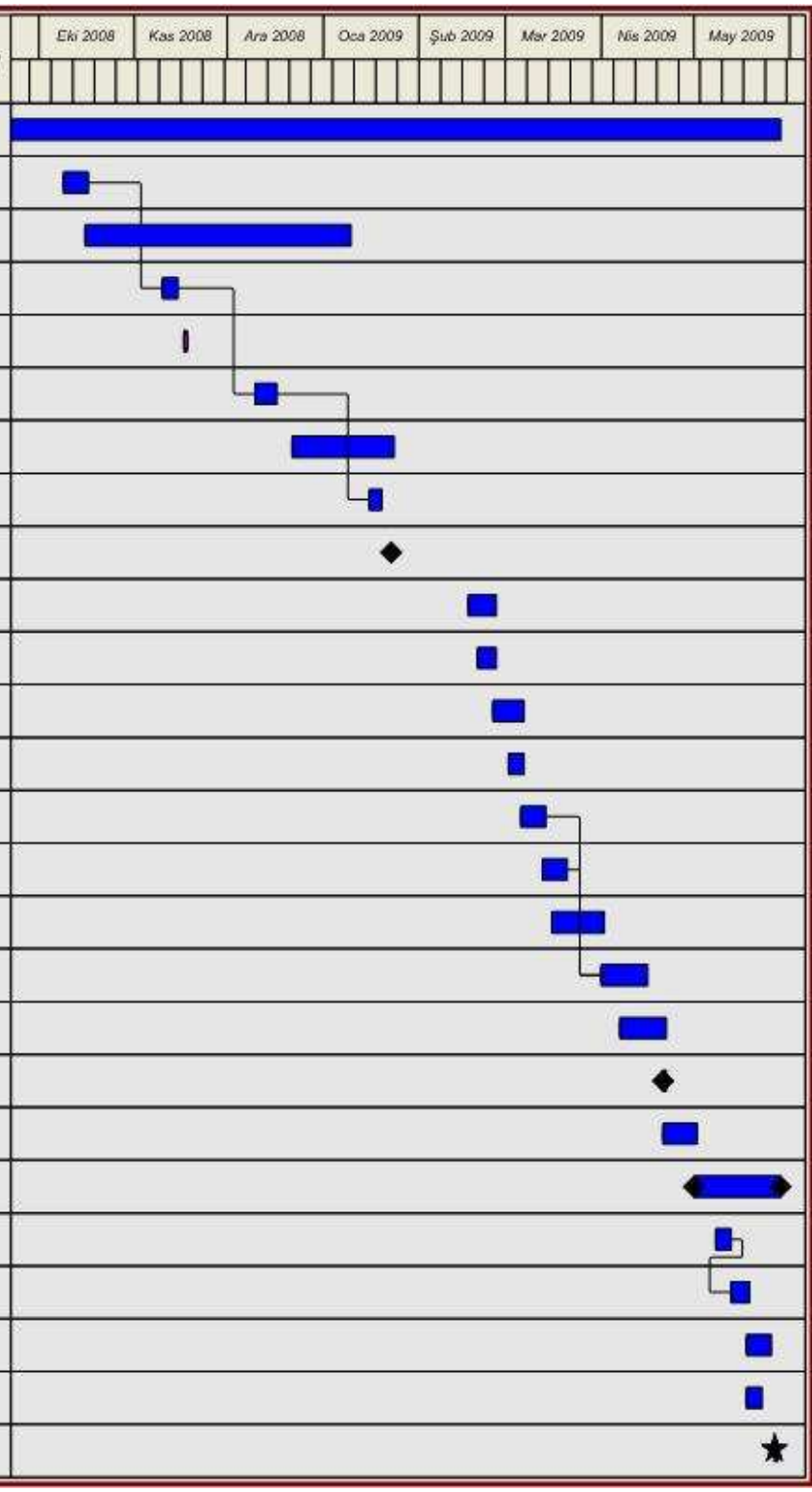
- Logging screen,
- Main menu,
- Friend List,
- Map View,
- Friend Information,
- Event/Advertisement,
- Call
- Chat.

7 Design Constraints

The schedule has milestones which are synchronized with reports and the implementations. The current stage of the project and planned future work can be viewed from the chart. Also the roadmap is discussed in the future work.

Gantt Chart

ID	Task Name	Start	Finish	Duration
1	Menaging Project	22.09.2008	28.05.2009	35,8w
2	Proposal	09.10.2008	16.10.2008	1,2w
3	Survey About Project	16.10.2008	09.01.2009	12,4w
4	Requirement Analysis Report	10.11.2008	14.11.2008	1w
5	Milestone 0	17.11.2008	17.11.2008	,2w
6	Initial Design Report	10.12.2008	16.12.2008	1w
7	Make Prototype	22.12.2008	23.01.2009	5w
8	Detailed Design Report	16.01.2009	19.01.2009	,4w
9	Milestone 1	23.01.2009	23.01.2009	0w
10	Creating Tables and Establishing DB	17.02.2009	25.02.2009	1,4w
11	Design of the Web Side	20.02.2009	25.02.2009	,8w
12	Preparing Application Interface in Android	25.02.2009	06.03.2009	1,6w
13	Establishing DB in Android	02.03.2009	06.03.2009	1w
14	Establishing Events	06.03.2009	13.03.2009	1,2w
15	Establishing Connection with Facebook	13.03.2009	20.03.2009	1,2w
16	Establishing Logic related to Modes	16.03.2009	01.04.2009	2,6w
17	Prepare Settings Part for Android and Web	01.04.2009	15.04.2009	2,2w
18	Prepare Giving Commercial Events in Server	07.04.2009	21.04.2009	2,2w
19	Milestone 2	21.04.2009	21.04.2009	,2w
20	Arrange visaulity of Application	21.04.2009	01.05.2009	1,8w
21	Final Release	01.05.2009	28.05.2009	4w
22	Alpha Testing	08.05.2009	12.05.2009	,6w
23	Beta Testing	13.05.2009	18.05.2009	,8w
24	User Manual	18.05.2009	25.05.2009	1,2w
25	Installation Manual	18.05.2009	22.05.2009	1w
26	Final Demo	27.05.2009	27.05.2009	,2w



7.2Future Work

Since the project has started, Fizan has come a long way but many things are yet to be done. The team decided to survey the subjects in an evolutionary approach.

Since team members were not familiar with mobile technology, web technology and java enterprise level technologies surveying has taken most of the time. During this period we have learnt JSP, JDBC, RMI, Swing, and Serialization in Java Technologies. In addition to that, we have learnt mobile application developing in Android sdk on java with the help of Eclipse IDE.

In the implementation part, we have learnt how to use and embed Google map with android and also GTalk. Moreover, we have started forming user interface for android applications.

Connection between sides of the project is extremely important for us and we have done this part in first semester. So our sides of the application are able to send or receive data from each other using java technologies.

For the publicity of our team and project, we have developed and deployed a web site. This site contains information about our project team, people can follow our project progress from there. We will try to update our website whenever something is added to the application.

We haven't done much implementation in first term so second term will be implementation semester; we will implement what we think in first semester in second semester

Main part of implementation, will be done in second semester, is establishing main logic in sides, exemplarily establishing logic for parental mode in both server side and cell-phone side and other modes. In addition to logic, User interface will be the one of the most challenging part because of the difficulty in developing user interface in android and dynamically changing user interface.

To conclude, there are lots of job to be done in second semester so we should divide the work equally among the team members. Synchronization among the members is another problem and we will solve it by using synchronization tools such as SVN.