Assignment #1

Dr. Fayad

CSCE 866

Team X:
Anees Dhala
Sanhitha Seerapu
Vandana Sunkara

Abbott's Approach:

This approach is used to identify classes, their attributes, operations and relationships (associations, aggregations, inheritance). Objects and classes are identified by nouns, pronouns and noun phrases. Individual objects are identified by noun phrases. Operations associated with objects are identified by verb and verb phrases.

List of Nouns – Potential Classes, Attributes, Relationships:

(Selected classes and Attributes are marked in Bold)

Problem	Irrelevant
Platform	Irrelevant
Family	Redundant (used as 'Group')
Friends	Redundant (used as 'Group')
Group	**Class**
Application	irrelevant
Members	**Class**
Messages	**Sub-Class of 'Information'**
Photos	**Sub-Class of 'Information'**
Group calendars	redundant (used as Calendar)
Address books	redundant
Group emails	redundant
Logins	**Attribute of 'Member'**
System	vague
Family page	implementation specific
Privilege	irrelevant
Family name	**Attribute of 'Group'** (adapted as 'group name')
Username	**Attribute of 'Member'**
Password	**Attribute of 'Member'**
MyFamilyReunion.com	Implementation specific
Option	irrelevant
Registration page	implementation specific
Registration form	implementation specific
Login name	redundant (username)
Events	**Attribute of Calendar**
Contacts	redundant (attributes of 'Member')
Names	**Attribute of 'Member'**
Email addresses	**Attribute of 'Member'**
Emails	**Class**
Owner	**Sub-class of 'Member', Aggregation of Group**
Birthday	**Attribute of 'Member'**
Calendar	**Class**
Features	vague
Details	irrelevant
Section	irrelevant

Date	**Attribute of ‘Calendar’**
Time	vague
List	vague
Words	vague
Bulletin board	vague
Webpage	implementation specific
Service	irrelevant
Resources	irrelevant
Name	**Attribute of ‘Member’**
Address	**Attribute of ‘Member’**
Telephone number	**Attribute of ‘Member’**
Information	**Class** (generalization of Message and Photo)
Day	Redundant (included in Date)
Occasion	Redundant (event)
Chat	**Class**
Conference	Redundant (chat)
People	vague
Email servers	irrelevant
Photo sharing servers	irrelevant
Chat servers	irrelevant
Loved ones	irrelevant
Effort	irrelevant
Button	implementation specific
Personal details	redundant
Group name	redundant
Fields	irrelevant
Link	implementation specific
Page	implementation specific
Invitation emails	redundant (emails)
Person	vague
Login page	implementation specific
Formalities	irrelevant
Post message link	implementation specific
Window	implementation specific
Text area	implementation specific
‘Post’ button	implementation specific
Post-It note	implementation specific
‘Next’ button	implementation specific
‘Re’ button	implementation specific
Threads	irrelevant
‘Photos’ link	implementation specific
thumbnail	implementation specific
images	vague
‘Upload’ link	implementation specific
path	vague
file name	**Attribute of ‘Photo’**

'Addresses' link	implementation specific
'Edit' button	implementation specific
'Add New' button	implementation specific
contacts form	implementation specific
Microsoft Outlook	irrelevant
Outlook Express	irrelevant
Month	redundant (included in Date)
Shortcut	irrelevant
Description	vague
Message board	vague
Miniature calendar	implementation specific
'Previous' button	implementation specific
Year	redundant (included in Date)
'To all members' option	implementation specific
column	irrelevant
online	vague
chat window	implementation specific
conversation	vague
submenu	implementation specific

List of Verbs and Verb Phrases– Potential Operations and Relationships

(Selected Relationships and Operations are marked in Bold)

To Design	Irrelevant
To implement	Irrelevant
To use	Irrelevant
To enable	Irrelevant
To post	**Relationship between 'Member' and 'Message'**
To share	vague
To maintain	irrelevant
To send	**Relationship between 'Member' and 'Email'**
To view	vague
To chat	**Relationship between 'Member' and 'Chat'**
To log-in	**Operation of Member**
To visit	irrelevant
To start	irrelevant
To give	irrelevant
To add	**Operation of 'Owner'**
To select	vague
To specify	irrelevant
To go	irrelevant

To create	vague
To choose	irrelevant
To take	irrelevant
To prompt	irrelevant
To fill	irrelevant
To store	vague
To ask	irrelevant
To input	irrelevant
To insert	vague
To invite	redundant (To add)
To join	vague
To change	**Operation of 'Member'**
To include	vague
To take	irrelevant
To provide	irrelevant
To explain	irrelevant
To display	irrelevant
To see	Vague
To upload	**Relationship between 'Member' and 'Photo'**
To contain	irrelevant
To enter	irrelevant
To wish	irrelevant
To communicate	vague
To decide	irrelevant
To study	irrelevant
To work	irrelevant
To increase	irrelevant
To keep	irrelevant
To reduce	irrelevant
To click	implementation specific
To approve	irrelevant
To assign	irrelevant
To pop-up	implementation specific
To appear	irrelevant
To save	vague
To access	vague
To reply	**Operation of 'Email', 'Message'**
To type	Irrelevant
To browse	irrelevant
To modify	vague
To enlarge	irrelevant
To want	irrelevant
To compose	**Operation of Email**
To label	irrelevant

List of High Level Classes with Corresponding Sub-classes:

- Group
- Member
  - Owner
- Information
  - Message
  - Photo
- Email
- Calendar
- Chat

Testing Classes:

Uniformity Test:

Each instance must have the same set of characteristics and be subject to the same rules.

e. g) Each group has the same set of characteristics such as a group name, owner etc.

More Than a Name Test:

Every object has attributes or else it is an attribute of another object.

e. g) Member has attributes such as name, address, user name, password etc..

Or Test:

Inclusion criteria should not use "OR" in any significant way- driver's license number or learners permit number.

e. g) Each group has a unique group name.

More Than a List Test:

Inclusion criteria is only a list of instances.

Class	Uniformity Test	More than a Name Test	Or Test	More than a List Test
Group	Pass	Pass	Pass	Pass
Member	Pass	Pass	Pass	Pass
- Owner	Pass	Pass	Pass	Pass
Information	Pass	Pass	Pass	Pass
- Messages	Pass	Pass	Pass	Pass
- Photo	Pass	Pass	Pass	Pass
E-mail	Pass	Pass	Pass	Pass
Calendar	Pass	Pass	Pass	Pass
Chat	Pass	Pass	Pass	Pass

List of Classes after the Tests:

- Group
- Member
  - Owner
- Information
  - Message
  - Photo
- Email
- Calendar
- Chat

CRC Cards: Class-Responsibilities-Collaborators cards

CRC cards are filling cards on which the class, its responsibilities and other collaborators are noted. A class is responsible for the knowledge its objects should have (attributes) and for the operations its objects must carry out to fulfill their tasks and reach their goals. Collaborators are other classes for which relationships must exist to enable the class to fulfill its tasks.

Group (Role: Family)		
Responsibility	**Collaboration**	
	Client	**Server**
Allow Members to share information	▪ Members ▪ Owner ▪ Information ▪ Calendar ▪ Email ▪ Chat	▪ Provide details of all the members ▪ Act as a platform for members to share information and keep in touch ▪ Keep track of the most recent logins of all members

Member (Role: Member of Family/Group of Friends)		
Responsibility	**Collaboration**	
	Client	**Server**
Keep in touch with other members of the group	▪ Group ▪ Information ▪ Calendar ▪ Email ▪ Chat	▪ Post Messages, Photos ▪ Provide personal details ▪ Add events to calendar ▪ Chat with other Members ▪ Send Emails ▪ Reply to Posted Messages

Owner (Role: Moderate the Group)		
Responsibility	**Collaboration**	
	Client	**Server**
Creating and Maintaining the Group	▪ Members ▪ Group	▪ Create the Group ▪ Add members ▪ Delete members

Information (Role: Messages and Photos)		
Responsibility	**Collaboration**	
	Client	**Server**
Share messages and photos among the members of the group	▪ Member	▪ Members can upload photos ▪ Members can post messages ▪ Members can reply to messages

Email (Role: Send and Receive Mails)		
Responsibility	**Collaboration**	
	Client	**Server**
Allow Family Members to communicate by exchanging text messages	▪ Members	▪ Allow Members to keep in touch ▪ Can be sent to multiple members ▪ Attached files can be sent

Calendar (Role: Common calendar of a Group)		
Responsibility	**Collaboration**	
	Client	**Server**
Store and intimate members of events	▪ Members ▪ Email ▪ Information (Messages)	▪ Store events input by members ▪ Store birthdays of members ▪ Post automatic messages on events ▪ Send email to member on special occasions

Chat (Role: Online Chat)		
Responsibility	**Collaboration**	
	Client	**Server**
Allow Family Members to send and receive instant messages	▪ Members	▪ List members currently online Send and receive instant messages among 2 people or a group of people

Reworked Problem Statement:

The problem is to design a system that enables communication of various kinds between members of a group. The group can be a group of friends or a family. The members of the group can share information in the form of short text messages, photos, emails and instant messaging.

The group is created by a person, designated the owner. The owner can assigns a name to the group, referred to as Family name. This name is used to distinguish groups. The owner can add and delete members belonging to the group.

A person added by the owner can access the group using his user name and password. Each member provides information about himself, like his contact details, his birthday and other events important to him when he logs in for the first time.

A calendar is maintained, which is accessible to all members of a particular group. The calendar contains birthdays of all the members (specified at first log-in) and other events input by members at any other time.

Short textual messages can be posted by a member and viewed by all members. Members can also reply to a posted message. All messages are time stamped along with the name of the member who posted it. Members can also upload photos, to be viewed by other members of that group.

A member has access to the email addresses of all members of the group. Using this, he can email selected members or all the members at once. The system keeps track of the most recent log-ins of all members of the group. Using this feature, members can view a list of other members currently online and send instant messages to each other.

Value of OO Design Heuristics in MyFamilyReunion:

Heuristic: ***Go beyond the problem domain***
In certain cases, the problem statement might be very specific to a particular group of persons, when in the real world, the same problem may be experienced by different people, but stated in different terms. In this sense, our problem was stated from a family's point of view. The same problem may be experienced by any group that wants to keep in touch. (For example, a group of friends, colleagues, etc.). To overcome this, and to generalize the solution, we have used the word 'Group' to represent users of the system.

Heuristic: ***Separate General Functionality from Specific Policy***
The aim of this heuristic is to design a re-usable system. On analysis of our problem, we found that the two entities, messages and photos share the same functionality, with a few differences. For example, both of them are posted by members and are deleted after a point of time. To avoid redundancy, we grouped them as subclasses of a larger class, namely 'Information'.

Heuristic: ***Avoid redundant and irrelevant classes which add no value in the problem domain***

The design of the system should be kept concise and precise. This makes the system more maintainable. In our problem, we have used the class ‘chat’ and so the class ‘conference’ is redundant.

Heuristic: ***Do not turn an operation into a class.***

A class should not be added to the system just to introduce an operation. In our system, the owner performs the operations of adding and deleting members from the group. Rather than introducing these operations as classes by themselves, we have added them as functions of the class ‘owner’.

Heuristic: ***Remove classes which have no attributes***

The class ‘owner’ performs the operations of adding and deleting members from the group. In all other respects, it shares attributes of any other member. Therefore, instead of introducing ‘Owner’ as a separate class without attributes, we introduced it as a subclass of ‘member’.

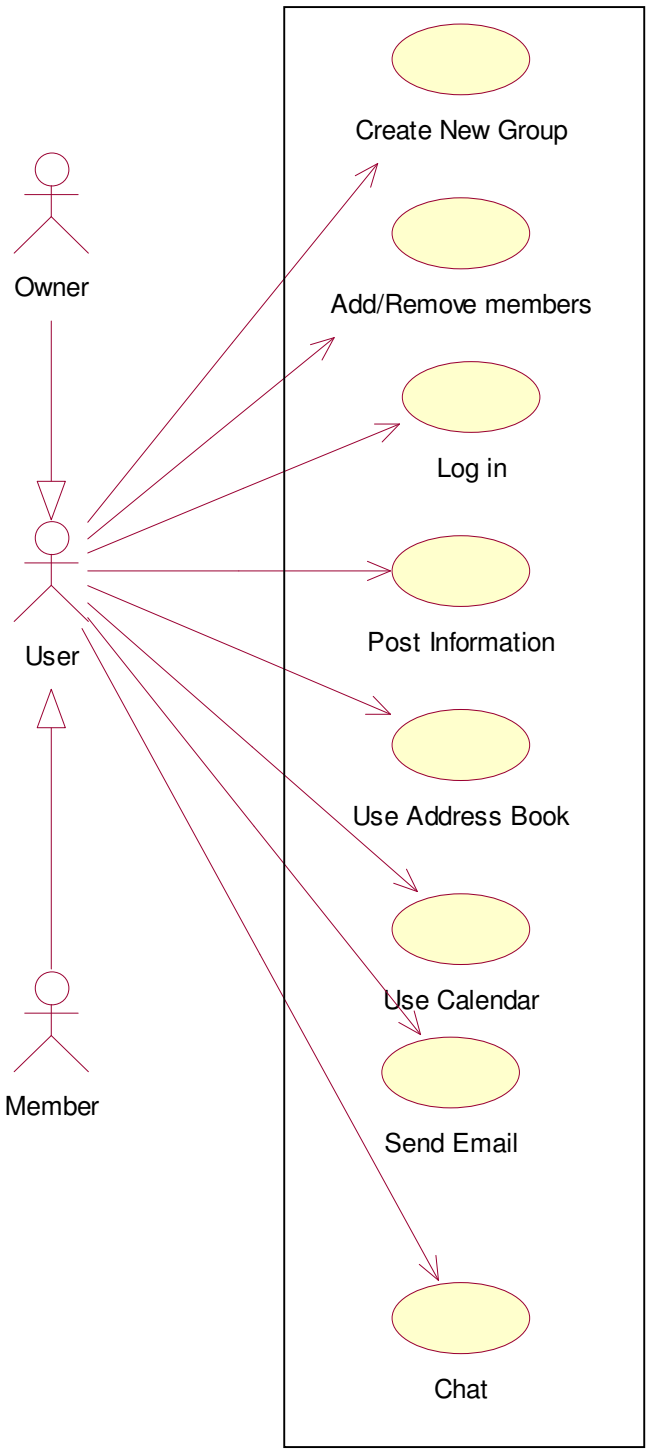
Assignment #2

Dr. Fayad

CSCE 866

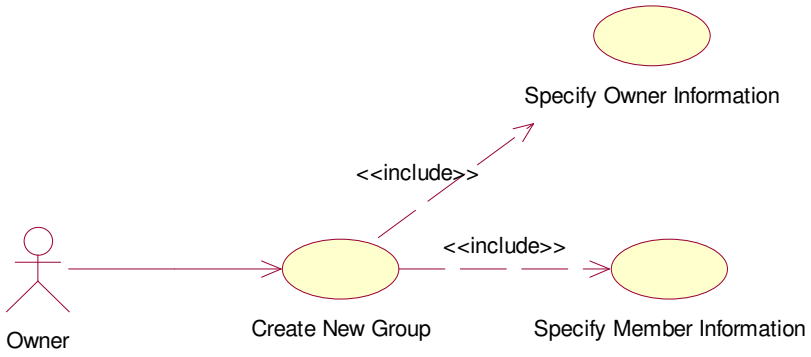
Team X:
Anees Dhala
Sanhitha Seerapu
Vandana Sunkara

Use Case Model:

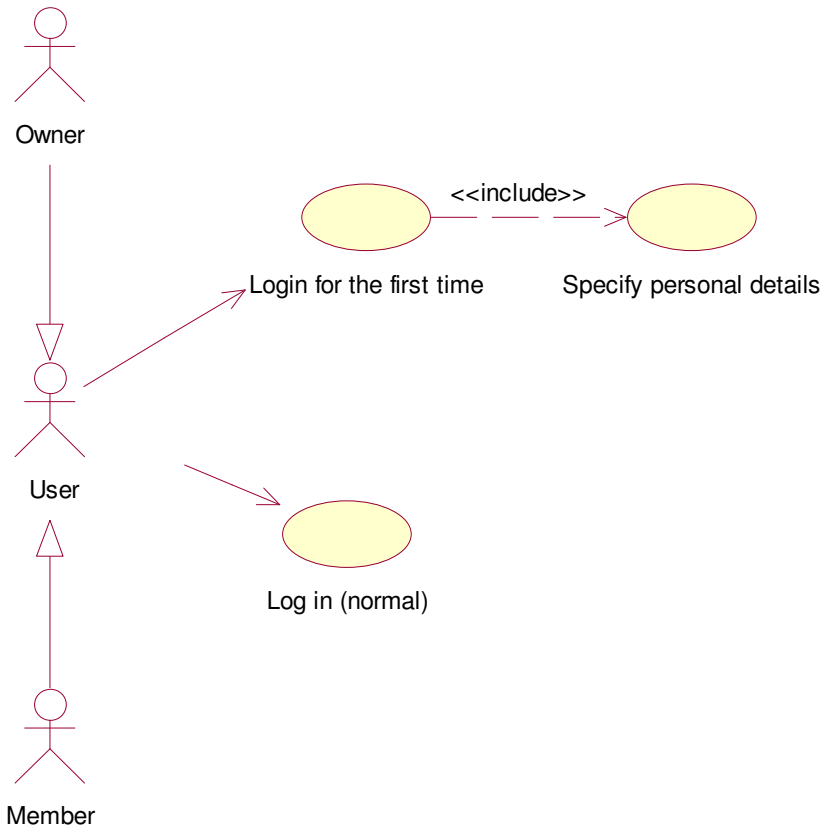


Use Case Diagrams:

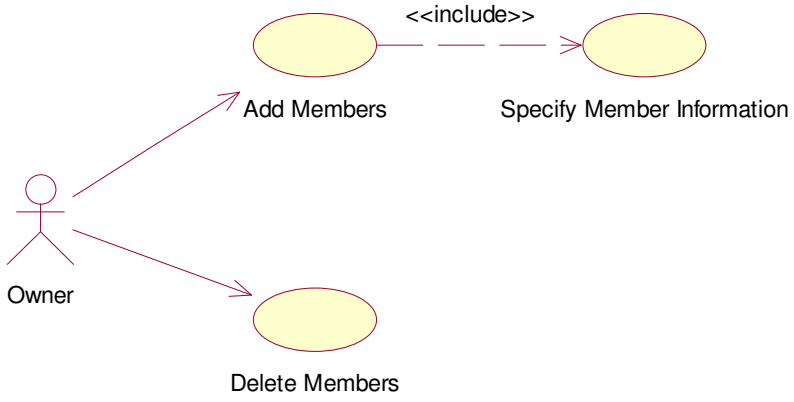
1. Create New Group:



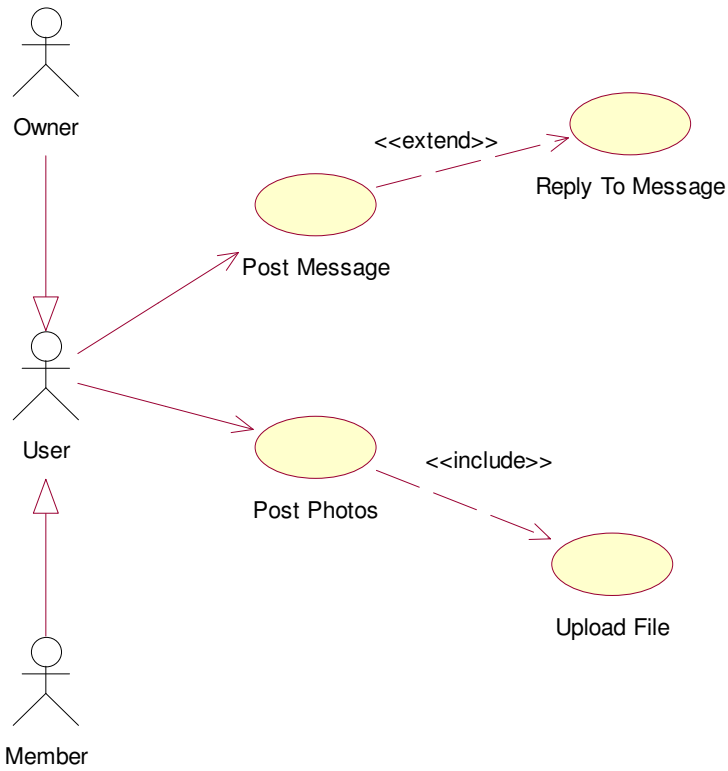
2. Log-in:



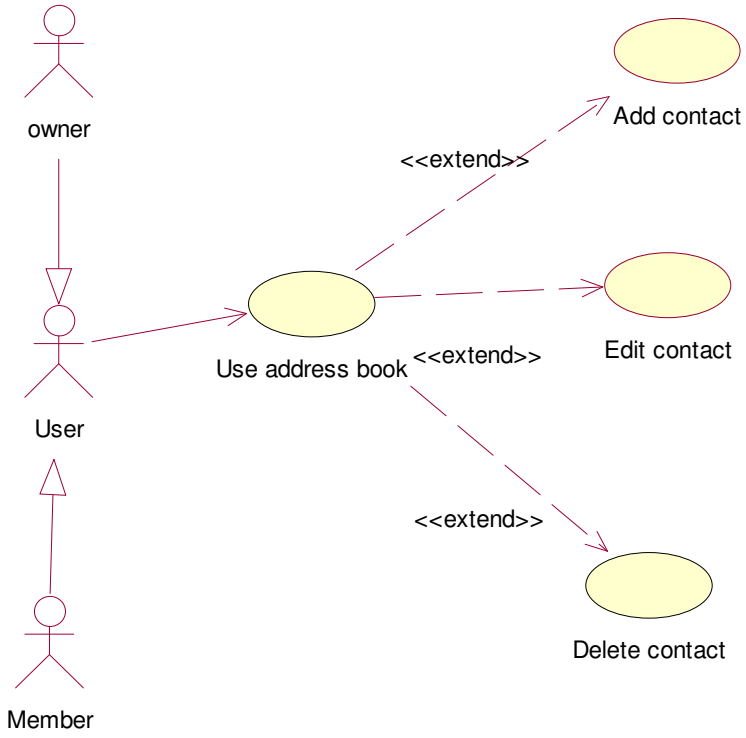
3. Add/Remove Members:



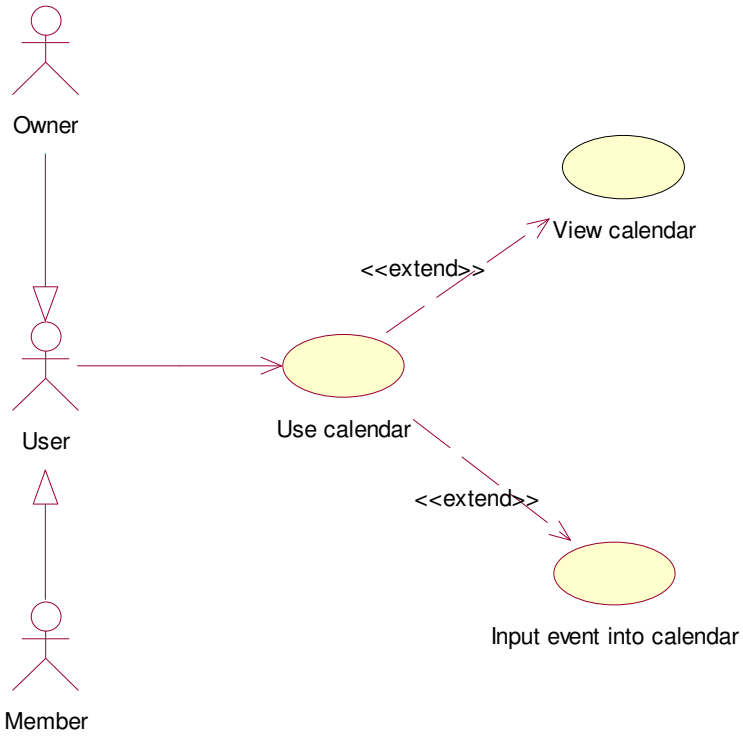
4. Post Information:



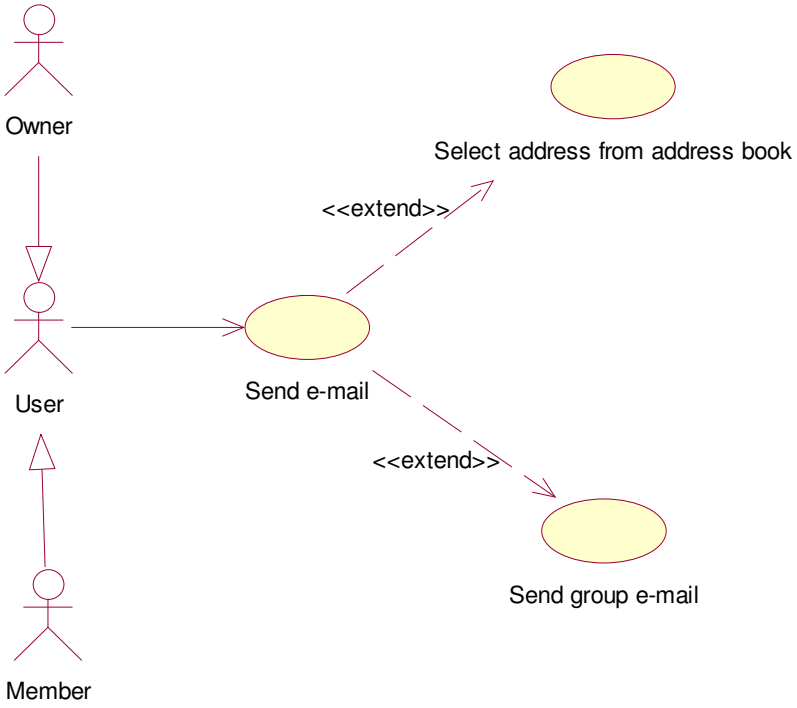
5. Use Address Book:



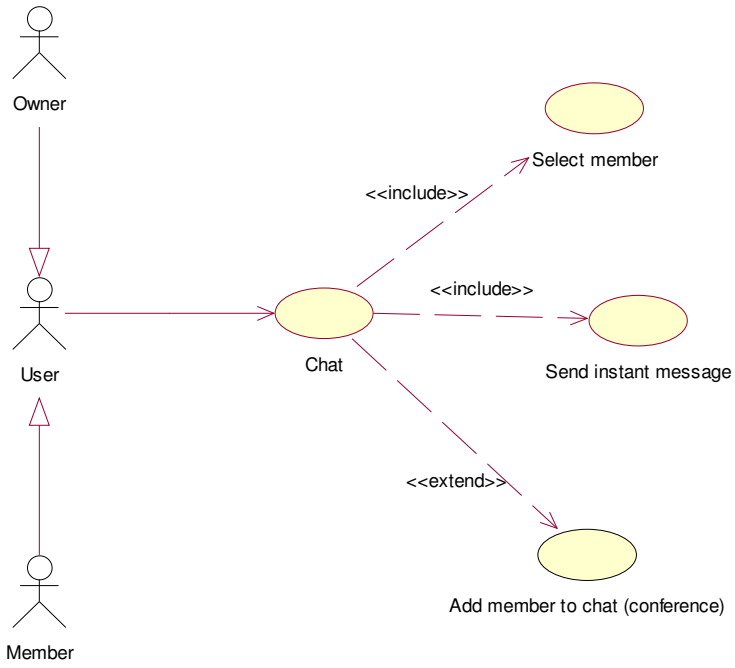
6.0 Use Calendar:



7. Send Email:


Owner
Select address from address book
<<extend>>
User
Send e-mail
<<extend>>
Send group e-mail
Member

8.0 Chat:


Owner
Select member
<<include>>
<<include>>
User
Chat
Send instant message
<<extend>>
Add member to chat (conference)
Member

Use Case Documentation:

Use Case ID	UC-01
Use Case Title	Create new group
Actors and Roles	Actor: User Role: Owner
Types	-
Corresponding classes	Group, Member (subclass: Owner)
Corresponding attributes	**Attributes of member:** Username, Password, Name, Address, Telephone number Email address, Birthday, Latest Login, Family name. **Attributes of Group:** Group name, Numbers of members, Name of owner, Date created.
Corresponding interfaces	-
Use case description	1. The user goes to www.MyFamilyReunion.com and is taken to the homepage. 2. He clicks on the 'Create New Group' button and is taken to the registration page. 3. He enters a group name and his personal details. 4. Once all the fields have been entered, he is asked to choose a user name (also called login name) and a password. 5. Once a unique user name has been approved, he can log on using that
Alternate flows	1. If the home page does not appear, the user should try again later 4. If the user clicks 'Enter' without filling all the information he is asked to fill in all the fields. 5. If he doesn't choose a unique user name, he is asked to choose another user name.

Use Case ID	UC-02
Use Case Title	Add/Remove members
Actors and Roles	Actor: User Role: Owner
Types	-
Corresponding classes	Member (subclass: Owner), Group
Corresponding attributes	**Attributes of Member:** Username, Password, Name, Address, Telephone number Email address, Birthday, Latest Login, Family name. **Attributes of Group:** Group name, Numbers of members, Name of owner, Date created.

Corresponding interfaces	-
Use case description	1. The owner clicks on the 'Members' link 2. If he wants to add members, he clicks on the 'Add Member' link. a. He inputs names and email ids of the people he wants to add. 3. If he wants to delete members, he right clicks on the members name and clicks on 'Delete'
Alternate flows	2. If the owner does not fill in all required information, he is prompted to do so.

Use Case ID	UC-03
Use Case Title	Log in
Actors and Roles	Actor: User, Role: Member
Types	-
Corresponding classes	Group, Member
Corresponding attributes	**Attributes of Member:** Username, Password, Name, Address, Telephone number Email address, Birthday, Latest Login, Family name. **Attributes of Group:** Group name, Numbers of members, Name of owner, Date created.
Corresponding interfaces	-
Use case description	1. The member goes to www.MyFamilyReunion.com 2. He selects the name of the group that he wishes to log on to. 3. In the log in page, he enters his user name and password. 4. If this is the first time that the user is logging on, he is prompted to enter his personal details like address, telephone number, birthday etc. 5. Henceforth, he can log on without these formalities.
Alternate flows	1. If the home page does not appear, the user should try again later 2. If his group name is not found, he can email the customer service representative. 3. If the user name or password is wrong, he is prompted to re-enter.

Use Case ID	UC-04
Use Case Title	Post Information
Actors and Roles	Actor: User, Role: Member
Types	-
Corresponding classes	Group, Member, Information
Corresponding attributes	**Attributes of Member:** Username, Password, Name, Address, Telephone number Email address, Birthday, Latest Login, Family name. **Attributes of Group:** Group name, Numbers of members, Name of owner, Date created. **Attributes of Information**: Posted By, Date, **Message:** Subject, Text Message **Photo:** Filename, Size
Corresponding interfaces	-
Use case description	1. To post a message, the member clicks on the 'Post message' link. a. A small window pops up with the date, time and the member's name on it. b. The user can enter his message into the text area present. c. The user clicks on the 'Post' button. d. The message appears on the family homepage as a Post-It note. 2. To reply to a message, the user clicks on the 'Re' button below the Posted message. a. A text box to type in the reply, the name of the member posting the reply and the time appears in a pop-up window. b. After typing the reply, the member clicks on 'Post' for the message to be posted. 3. The member clicks on the 'Photos' link a. He is taken to a page where there are thumbnail images of all the photos that have been uploaded thus far. b. Clicking on the thumbnail image takes the user to the full size image. c. The member can upload pictures by clicking on the 'Upload' link on the photos page. i. In the pop-up window, he may either

	type the name of the file with the path, or browse and select the file name. ii. Once the picture is uploaded, it may be viewed by any member.
Alternate flows	1. If there are more than 30 messages, the user is informed that the earliest message posted will be deleted. a. If the member attempts to send an empty message, he is prompted to enter a message. 2. If there are more than 30 messages, the user is informed that the earliest message posted will be deleted. 3. If the file uploaded is not a known image format, the user is prompted to upload another file.

Use Case ID	UC-05
Use Case Title	Use Address Book
Actors and Roles	Actor: User, Role: Member
Types	-
Corresponding classes	Group, Member, Contact
Corresponding attributes	**Attributes of Member:** Username, Password, Name, Address, Telephone number Email address, Birthday, Latest Login, Family name. **Attributes of Group:** Group name, Numbers of members, Name of owner, Date created. **Attributes of Contact:** Name, Address, Email id, telephone number
Corresponding interfaces	-
Use case description	1. From the family homepage, the user clicks on the 'Addresses' link and is taken to the address book. 2. The member can edit contacts by clicking the 'Edit' button that appears next to the name 3. The member can add a contact by clicking on the 'Add New' button. a. A contact form appears. b. The member fills out the form and click on 'Save' c. The new contact is added to the address book or the details of an existing contact are modified.
Alternate flows	3b. If the user clicks on 'Save' without inputting all

	relevant fields, he is prompted to complete the 'contact' form. c. If the user inputs exact details of an existing contact, he is prompted that the contact already exists.

Use Case ID	UC-06
Use Case Title	Use Calendar
Actors and Roles	Actor: User, Role: Member
Types	-
Corresponding classes	Group, Member, Calendar
Corresponding attributes	**Attributes of Member:** Username, Password, Name, Address, Telephone number Email address, Birthday, Latest Login, Family name. **Attributes of Group:** Group name, Numbers of members, Name of owner, Date created. **Attributes of Calendar:** Date, Event type, Event Description
Corresponding interfaces	-
Use case description	1. The user clicks on the miniature calendar on the homepage to enlarge it. 2. He can view the calendar a. To view the calendar for the previous or next month, he clicks on the 'Previous' and 'Next links below the miniature calendar. b. Similarly the entire year's calendar can be viewed by clicking on the year name (Example: 2001, 2002…) 3. To enter events into the calendar, the user clicks on the miniature calendar and is taken to a page with a larger calendar. a. In this page, the user clicks on the particular date and types a description of the event next to the date. b. On that date, a message is posted in the message board with the description just typed.
Alternate flows	3.a. If the user types a description of more than 20 words, he is prompted about the 20 page limit. b. If there is more than one event on a particular date,

	all events are posted.

Use Case ID	UC-07
Use Case Title	Send Email
Actors and Roles	Actor: User, Role: Member
Types	-
Corresponding classes	Group, Members, Email, Contact
Corresponding attributes	**Attributes of Member:** Username, Password, Name, Address, Telephone number Email address, Birthday, Latest Login, Family name. **Attributes of Group:** Group name, Numbers of members, Name of owner, Date created. **Attributes of Contact:** Name, Address, Email id, telephone number **Attribute of Email:** From, To, Subject, Date, Message Body
Corresponding interfaces	-
Use case description	1. Users can send mails to the other members in the group by selecting their mail addresses from the Address Book. 2. He selects the ids by just clicking on their names in the Address Book. a. The user can choose the 'To All Members' option on the Address Book to send an email to all the members of the group. 3. He composes the mail. 4. He may attach a file to the email. 5. When he clicks 'Send', his mail is sent to all the selected members.
Alternate flows	4. If the size of the file attached is larger than 500Kb, he is informed that the file is too large. 5. If the user does not enter a subject, he is informed of this.

Use Case ID	UC-08
Use Case Title	Chat
Actors and Roles	Actor: User, Role: Member
Types	-
Corresponding classes	Group, Member, Chat
Corresponding attributes	**Attributes of Member:** Username, Password, Name, Address, Telephone number Email address, Birthday, Latest Login, Family name. **Attributes of Group:** Group name, Numbers of members, Name of owner, Date created. **Attributes of Chat:** Members in Chat, Time started, History
Corresponding interfaces	-
Use case description	1. The user can then double click on an 'online' member's name with whom he wants to start chatting. 2. A chat window pops up. 3. In the chat window, he can send and receive messages. 4. If the user wants to add other members to the conversation, he clicks on 'File', 'Add member to Chat' in the chat window. a. From the submenu, he clicks on the particular member's name and that person is added to the Conference.
Alternate flows	-

Updated Classes:

- Group
- Member
  - Owner
- Information
  - Message
  - Photo
- Email
- Calendar
- Chat
- Contact

Updated CRC Cards:

Group (Role: Family)		
Responsibility	**Collaboration**	
	Client	**Server**
Allow Members to share information	▪ Members ▪ Owner ▪ Information ▪ Calendar ▪ Email ▪ Chat	▪ Provide details of all the members ▪ Act as a platform for members to share information and keep in touch ▪ Keep track of the most recent logins of all members ▪ Display list of members

Member (Role: Member of Family/Group of Friends)		
Responsibility	**Collaboration**	
	Client	**Server**

Keep in touch with other members of the group	▪ Group ▪ Information ▪ Calendar ▪ Email ▪ Chat	▪ Post Messages, Photos ▪ Provide personal details ▪ Add events to calendar ▪ Chat with other Members ▪ Send Emails ▪ Reply to Posted Messages ▪ Change password

Owner (Role: Moderate the Group)		
Responsibility	**Collaboration**	
	Client	**Server**
Creating and Maintaining the Group	▪ Members ▪ Group	▪ Create the Group ▪ Add members ▪ Delete members

Information (Role: Messages and Photos)		
Responsibility	**Collaboration**	
	Client	**Server**
Share messages and photos among the members of the group	▪ Member	▪ Members can upload photos ▪ Members can post messages ▪ Members can reply to messages ▪ Display the number of messages and photos.

Email (Role: Send and Receive Mails)		
Responsibility	**Collaboration**	
	Client	**Server**
Allow Family Members to communicate by exchanging text messages	▪ Members	▪ Allow Members to keep in touch ▪ Can be sent to multiple members ▪ Attached files can be sent

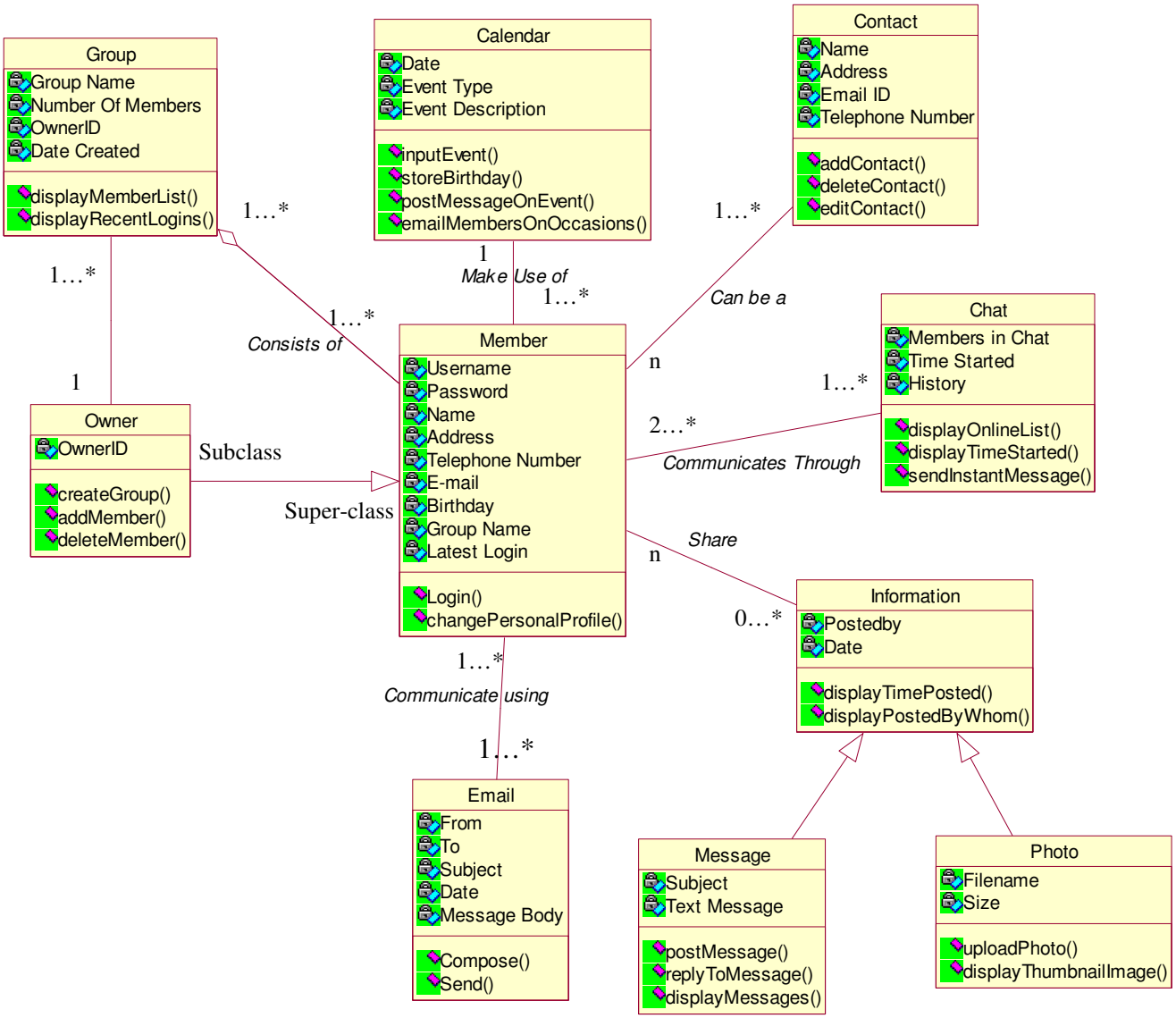
Calendar (Role: Common calendar of a Group)

Responsibility	Collaboration	
	Client	**Server**
Store and intimate members of events	▪ Members ▪ Email ▪ Information (Messages)	▪ Store events input by members ▪ Store birthdays of members ▪ Post automatic messages on events ▪ Send email to member on special occasions

Chat (Role: Online Chat)		
Responsibility	**Collaboration**	
	Client	**Server**
Allow Family Members to send and receive instant messages	▪ Members	▪ List members currently online Send and receive instant messages among 2 people or a group of people

Contact (Role: Address Book)		
Responsibility	**Collaboration**	
	Client	**Server**
Keep track of contact information	▪ Group ▪ Member	▪ Stores names, addresses, telephone numbers, email ids, etc. of members and their contacts.

UML Class Diagram:



Description of Class Diagram

The Class diagram is a graphical representation of all the classes and the relations they have with the other classes in the model.

In our model, the various Classes interact and are related in the following way:
(The Classes are italicized)
Our diagram consists of a *Group,* which consists of many *Member*s. An 'Owner', who creates the *Group,* is a subclass of *Member*. Each *Member* can be a *Contact* .Two or more *Member*s communicates through *Chat* .They also can communicate using *E-mail*. All *Member* shares *Information*, which has 'Photo' and 'Message' as its subclasses.

The *Member* has, the name of the user, his password, his name, his contact details which include his phone number, address, mail id, his birthday and his last login as its attributes. The operation, login(), allows the user to enter his user name and password with which he logs on to the system. Change personal profile() enables him to update or change his existing details in the database .

The Owner which is a subclass of *Member* has the owner id has its attribute. It creates a new group using the operation createGroup(). AddMember() is used to add a new member into the group while delateMember() is used to delete an existing member from the group.

Group has the name of the group, the number of members, the id of the owner and the date the group was created as its attributes .The operation displayMemberList() displays the list of members and displayRecentLogins() displays the logins of the recent users.

Information consists of the date the information was posted and the name of the person the information was posted by as its attributes. The operation displayTimePosted() displays the time a message was posted and displayPostedByWhom() shows the name of the person who posted the message. It has two subclasses: Message and Photo.A Message has a subject & the text. postMessage() posts a message,displayMessage() displays the message and replyToMessage() enables the user to reply to a message that has been posted by clicking on 'Reply' button in the message text window. The filename and the size of the photo are the attributes of the subclass Photo. The operation uploadPhoto() uploads the photo from a location in the computer and displayTumbnailImage() displays a small image which when clicked on enlarges the image for a better and complete view.

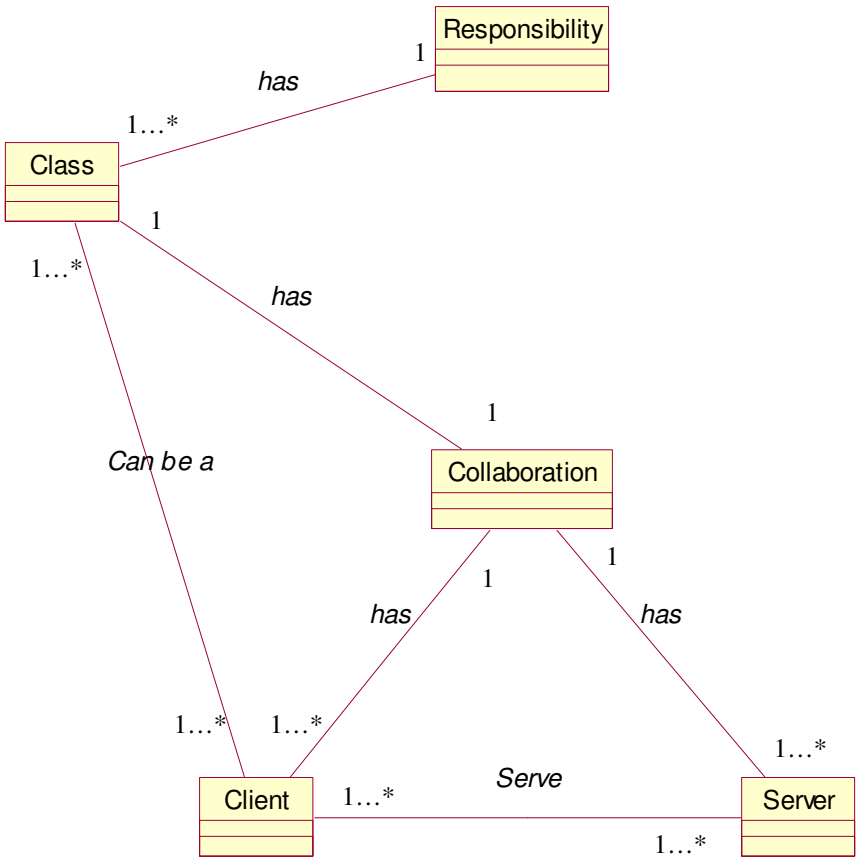
Contact contains the name of the user, his address, his e-mail id and his telephone number as its attributes. The operation addContact(),deleteContact(),editContact() enables the user to update the Address Book or make changes to it.

We have a *Calendar* which as a date, the type of the event and the description of the event as its attributes. InputEvent() inputs the event type into the calendar, storeBirthday() stores a date of the event, postMessageon Event() posts a message on an event and emailMembersOn Occasion() sends an individual e-mail to the person wishing him on the occasion.

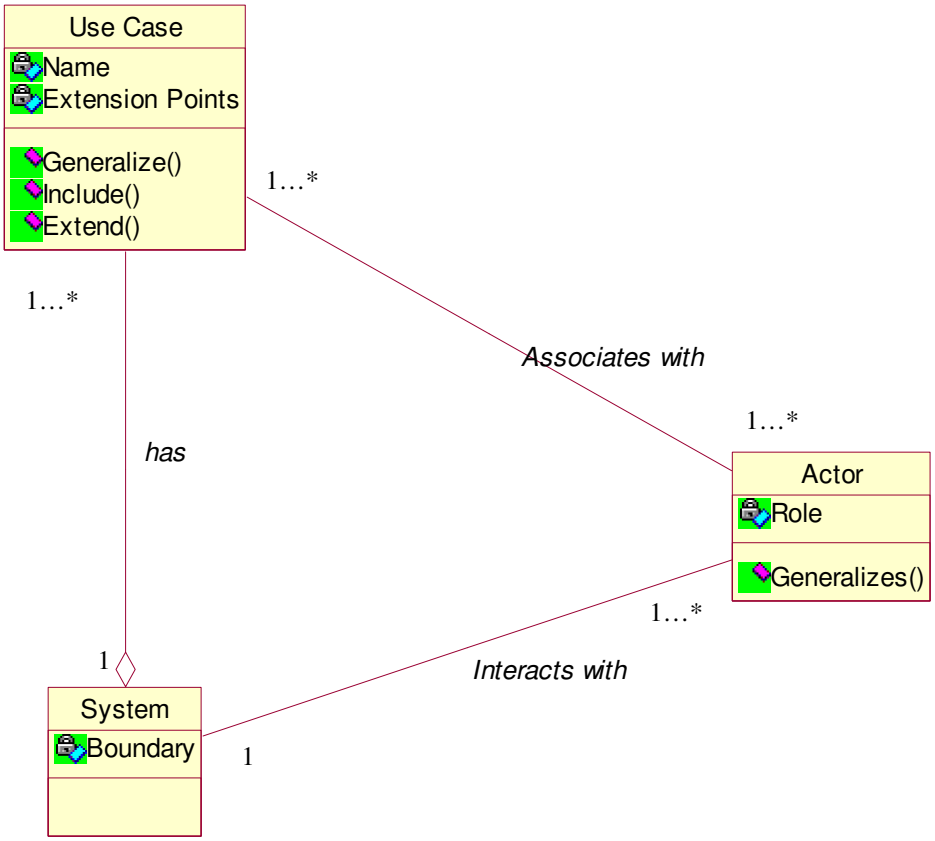
E-mail consists of who the message is from, whom it is being sent to, the date it is sent, the subject & message body. The operation compose() is used while the user is filling in the mail template. Send() uses mail protocol to send the e-mail to the recipients.

Meta Models:

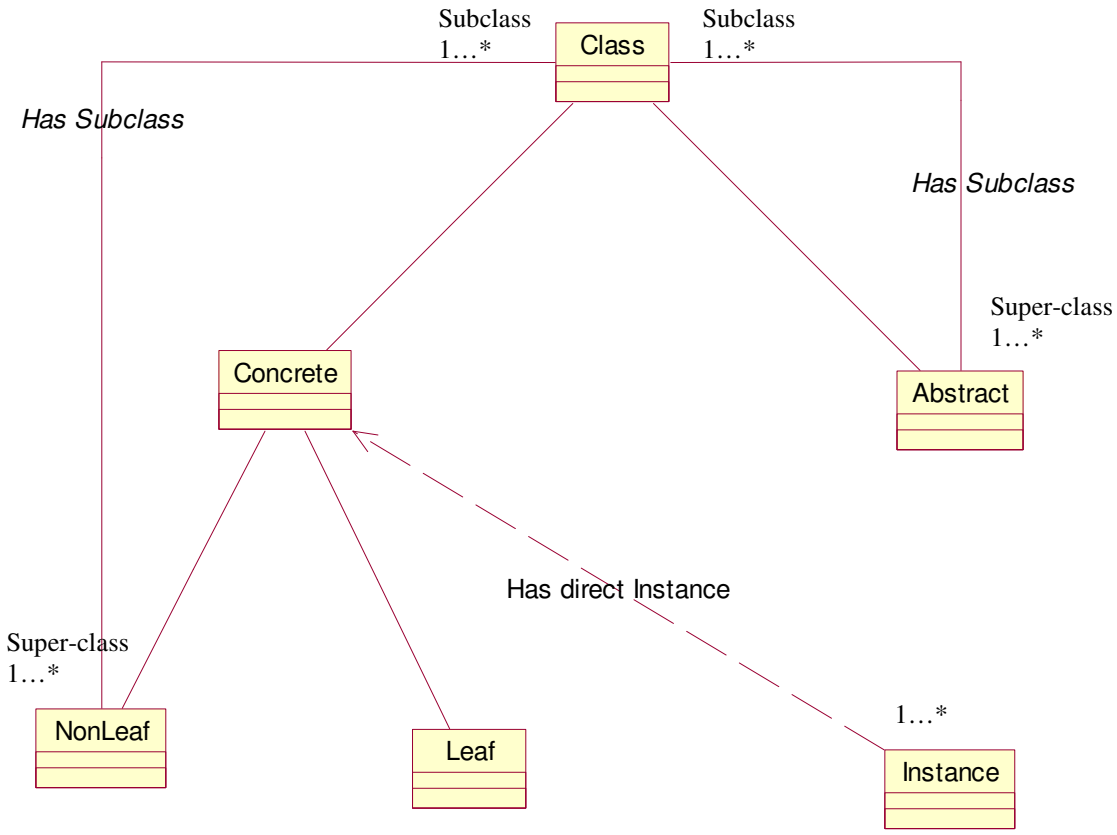
CRC Card:



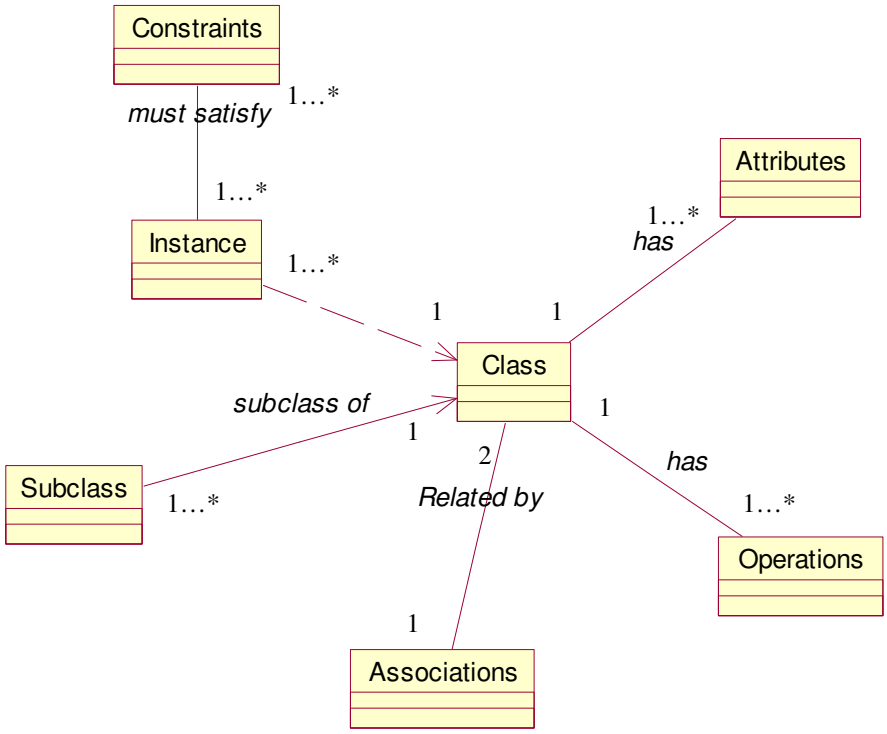
Use Case Diagram



Abstract Class



Class Diagram



Value of OO Design Heuristics

Heuristic #1*: Derived classes must have knowledge of their base class by definition, but base classes should not know about their classes.*

Derived classes have knowledge of their base classes as they inherit from base classes. But if base classes have knowledge of their derived classes, then if a new derived class is added to a base class, the code of the base class needs to be modified which is not desirable.
Example: Owner class is derived from Member class. Owner class knows all the data and members in 'Members' class. But, the methods such as 'add member', 'delete member' are private to 'Owner' class and are not known to 'Member' class.

Heuristic # 2*: Most of the methods of a class should use most of the data most of the time.*

Methods with in a class should use most of the data within the class.
Example: In all classes most methods use most data. For example, in 'Email' class data such as 'to', 'from' are used by methods such as 'send()'.

Heuristic # 3:*Keep related data and behavior in one place.*

Data for a particular class and its related methods should be kept in one particular place, so that most methods use most data most of the time.
Example: all classes satisfy this heuristic.

Heuristic #4: *Do not turn an operation into a class. Be suspicious of any class which has only one piece of meaningful behavior, especially if its name is a verb or derived from a verb.*
In our problem, for the 'Message' subclass, for example, there is an operation 'postMessage()'. Even though this is the main operation of this class, it has not been made a class by itself. There are some attributes related to the class and some variations of the same operation.
On the other hand, even though the 'chat' is a verb or a verb derivative, it is used as a class name in this context and has other attributes and operations other than just 'chat'.

Evaluation of Models using Model Essentials:

A model is a description of something. It is a well-established human process and should allow us to answer questions about a real thing before we build it. It should be simple and capture only those features deemed essential by model building for their goals. After a model is built, it has to be validated and evaluated.

We now evaluate our model based on a few parameters.

Simple	Yes
Complete	Yes
Stable to technological changes	Yes
Testable	To an extent
Easy to understand	Yes
Visual or graphical	Yes

1*) Simple*: Our model basically deals with the design and implementation of a platform for communication of family members and friends. The whole idea behind the design of our model is to achieve an application, which is user-friendly as we have a varied range of users of our model (anyone from a software developer to a simple man who needn't even be computer literate). There are not many interactions with sources in the model.
We designed our model in a way that is easy to understand and use. There are instructions at each and every step, which lead a user on how to use a specific module in the application making the use of model highly simple.
The Class diagrams given specify how the model functions in a clear and simple way.

2) *Complete:* The purpose of our model is to achieve a platform, which helps the users to keep in touch.
All the modules introduced are depicted clearly in the Class diagrams. The Case diagrams list the corresponding classes and attributes of each Class in the application. The Actor and the Role are specified.
The CRC cards (old and updated) specify the responsibilities and the collaborations.
The responsibilities are stated as generally as possible. The behavior is kept with related information and the information about one thing are kept in a place. The responsibilities are shared among related objects.
In Collaboration, each responsibility identified for each class is the class capable of fulfilling this responsibility itself.
Our CRC cards clearly & completely specify the responsibility of each class and its collaboration in terms of the client and the server thereby making our model complete.

However, since ours is a small model, it had to be scaled down to a certain extent. All the possible attributes have not been specified.

3) *Stable to technological changes:* When the data in the model is updated or a new module is to be added to our model, it can be easily done. A new CRC card is developed to update the new change. The Class diagrams can be modified so as to fit it the new addition or to delete a class. Changes in terms of addition of a new module or may be a change in the existing module can always be added to the model evaluation.

4) *Testable:* Each feature in our model is traceable to its requirements. However, each module cannot be tested unless a few more additional features are added to the model. Although Testable to an extent, we cannot claim that our model is completely Testable under present circumstances.

5)*Easy to understand:* The basic idea behind the design of our model is to achieve an application, which is user-friendly as we have a varied range of users of our .We designed our model in a way that is easy to understand and use. There are instructions at each and every step, which lead a user on how to use a specific module in the application making the use of model highly simple.
The use of CRC cards clearly specifies the Actor and the Role and the relationships of each class. Our case diagrams are completed in a simple manner, which is easy to understand.

6) *Visual or graphical:* The use of all the Class diagrams and the Use Case diagrams make our model graphical. The CRC cards are easy to understand and are depicted in a simple way. The entire evaluation of the model is highly graphical.

Assignment #3

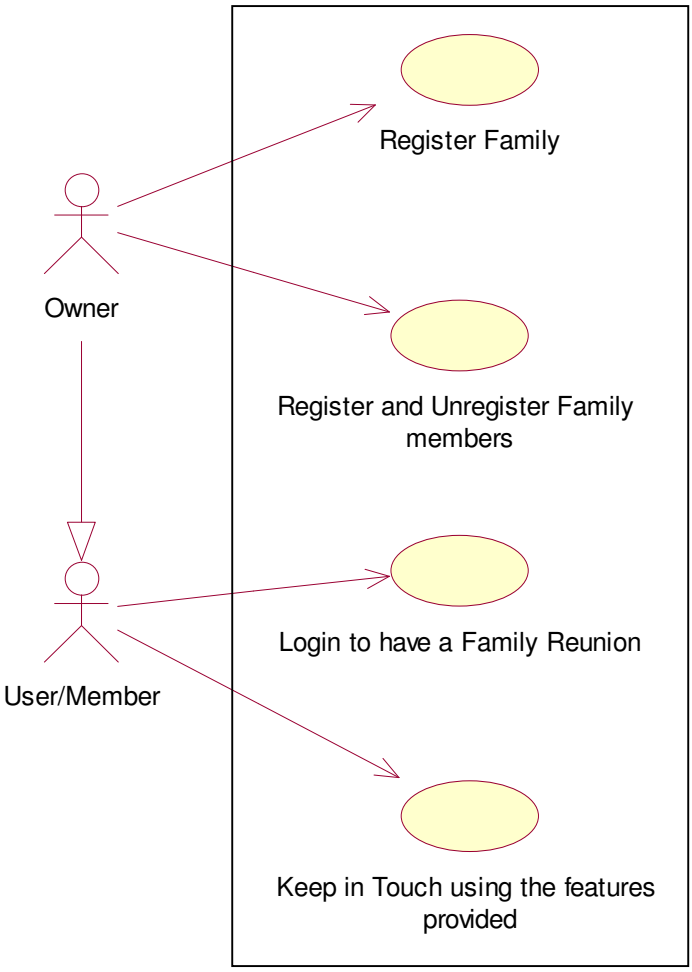
Dr. Fayad

CSCE 866

March 28, 2002

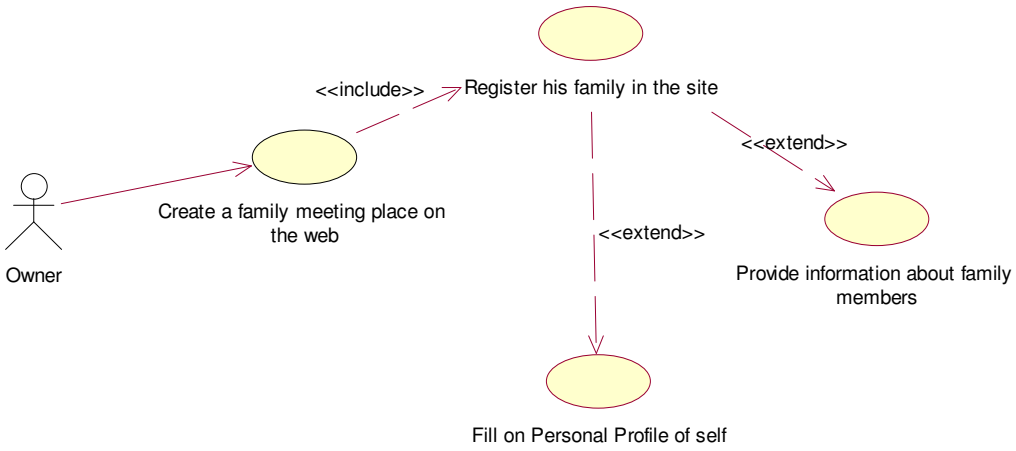
Team X:
Anees Dhala
Sanhitha Seerapu
Vandana Sunkara

Use Case Model:

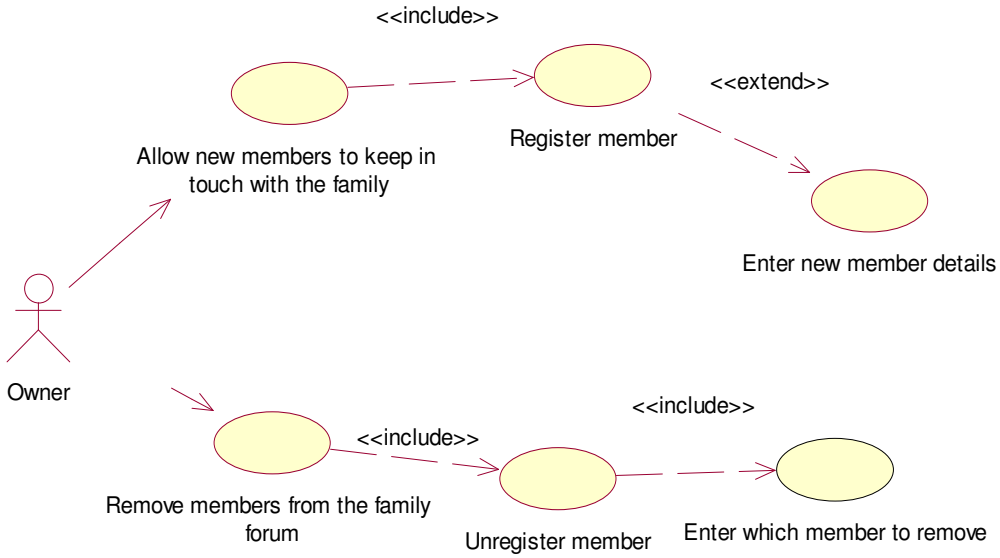


Use Case Diagrams:

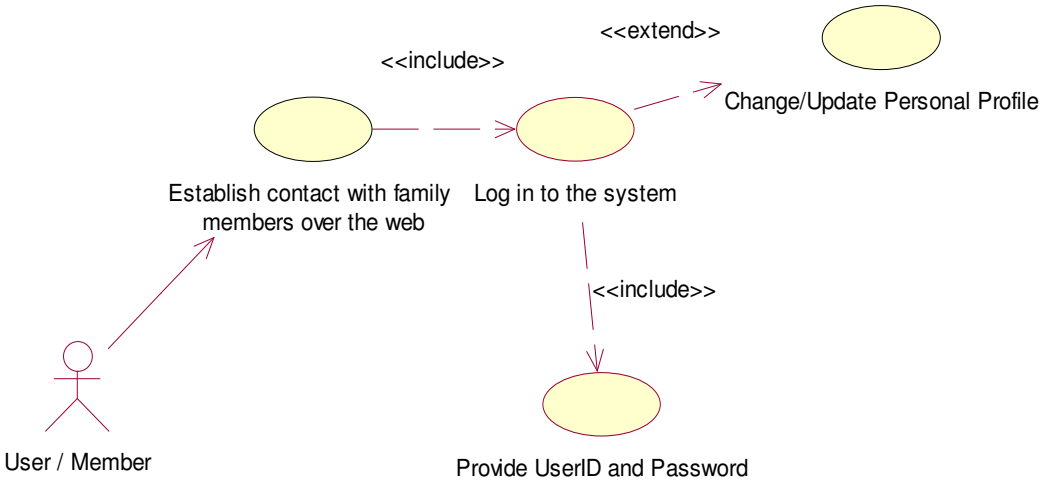
Register Family



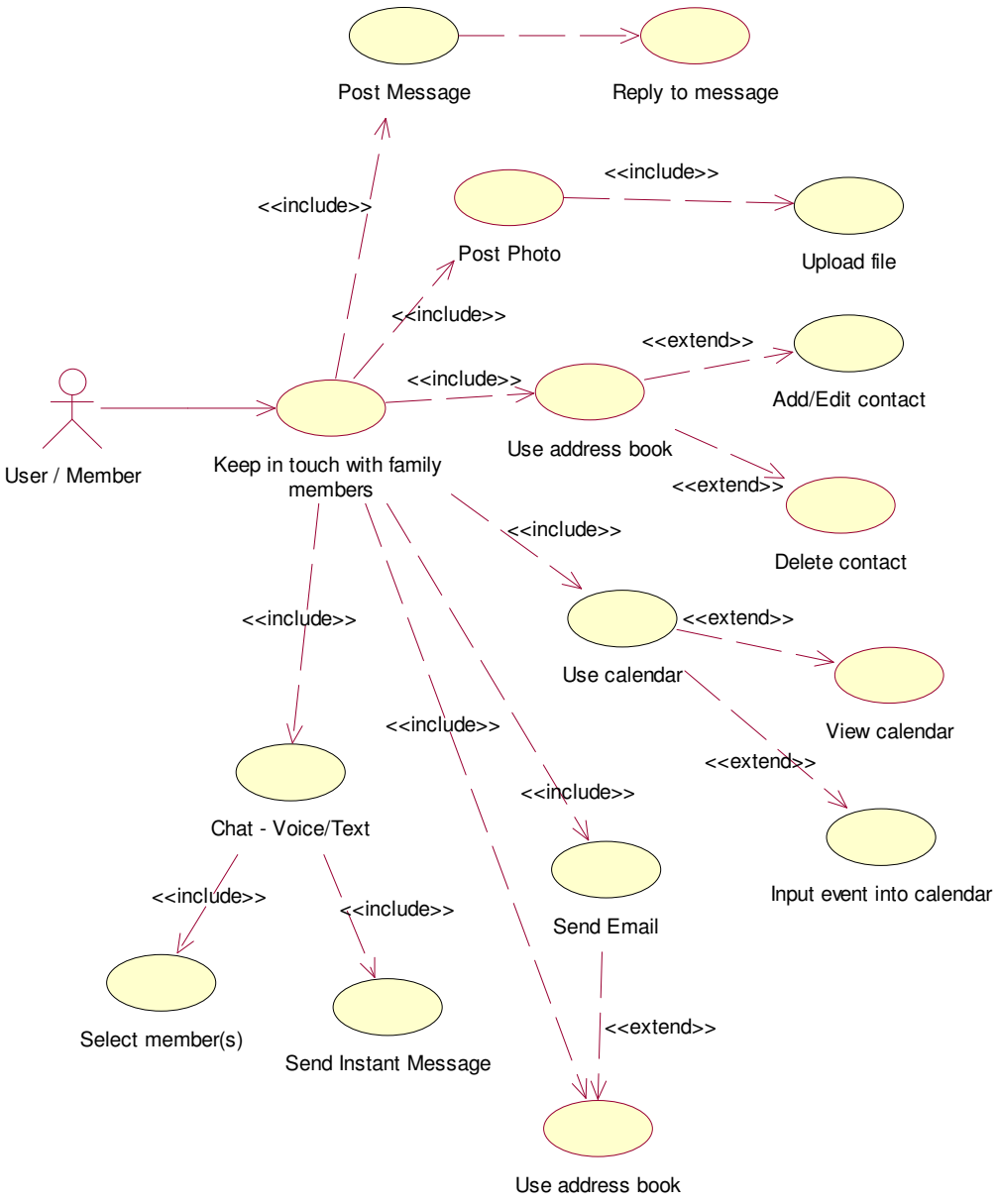
Register/Unregister Members:



Login to the system:



Keep in touch with family:



Use Case Documentation:

Use Case ID	UC-01
Use Case Title	Register family
Actors and Roles	Actor: User Role: Owner
Types	-
Corresponding classes	User (role), group, share feelings, keep in touch, family reunion.
Corresponding attributes	**Attributes of user:** Username, Password, Name, Address, Telephone number Email address, Birthday, Latest Login, group name, role. **Attributes of Group:** Group name, Numbers of members, Owner Id, Date created.
Corresponding interfaces	Log-in, view members and latest log-ins.
EBTs	Family Reunion Keeping in touch Share feelings.
BOs	User (role)
IOs	Owner, group
Use case description	6. The user goes to www.MyFamilyReunion.com and is taken to the homepage. 7. He clicks on the 'Create New Group' button and is taken to the registration page. 8. He enters a group name and his personal details. 9. Once all the fields have been entered, he is asked to choose a user name (also called login name) and a password. 10. Once a unique user name has been approved, he can log on using that
Alternate flows	6. If the user clicks 'Enter' without filling all the information he is asked to fill in all the fields. 7. If he doesn't choose a unique user name, he is asked to choose another user name.

Use Case ID	UC-02
Use Case Title	Register/ Unregister members
Actors and Roles	Actor: User Role: Owner
Types	-
Corresponding classes	User (subclass: Owner), Group
Corresponding attributes	**Attributes of User:** Username, Password, Name, Address, Telephone number

	Email address, Birthday, Latest Login, Group name, role. **Attributes of Group:** Group name, Numbers of members, Owner Id, Date created.
Corresponding interfaces	Adding and deleting members.
EBTs	Family Reunion Keeping in touch Share feelings.
BOs	User (role)
IOs	Owner, group
Use case description	4. The owner clicks on the 'Members' link 5. If he wants to add members, he clicks on the 'Add Member' link. a. He inputs names and email ids of the people he wants to add. 6. If he wants to delete members, he right clicks on the members name and clicks on 'Delete'
Alternate flows	2. If the owner does not fill in all required information, he is prompted to do so.

Use Case ID	UC-03
Use Case Title	Log in
Actors and Roles	Actor: User, Role: Member
Types	-
Corresponding classes	Group, User, Family Reunion, Keeping in touch, Share feelings.
Corresponding attributes	**Attributes of User:** Username, Password, Name, Address, Telephone number Email address, Birthday, Latest Login, Group name, role. **Attributes of Group:** Group name, Numbers of members, Owner Id, Date created.
Corresponding interfaces	Log-in manager.
EBTs	Family Reunion Keeping in touch Share feelings.
BOs	User (role)
IOs	Owner, group
Use case description	6. The member goes to www.MyFamilyReunion.com 7. He selects the name of the group that he wishes to log on to. 8. In the log in page, he enters his user name and password. 9. If this is the first time that the user is logging on, he

	is prompted to enter his personal details like address, telephone number, birthday etc. 10. Henceforth, he can log on without these formalities.
Alternate flows	4. If his group name is not found, he can email the customer service representative. 5. If the user name or password is wrong, he is prompted to re-enter.

Use Case ID	UC-04
Use Case Title	Post Messages, Photos
Actors and Roles	Actor: User, Role: Member
Types	-
Corresponding classes	Group, User, Photo, Message, Pictorial, Textual, Conversation.
Corresponding attributes	**Attributes of User:** Username, Password, Name, Address, Telephone number Email address, Birthday, Latest Login, Group name, role. **Attributes of Group:** Group name, Numbers of members, OwnerID, Date created. **Attributes of Photo:** File name, Size, DatePosted, Posted by. **Attributes of Message:** Subject, Text, Sender Date.
Corresponding interfaces	-
EBTs	Keeping in Touch, Share Feelings
BOs	Textual, Pictorial, User
IOs	Photo, Message, Group
Use case description	4. To post a message, the member clicks on the 'Post message' link. a. A small window pops up with the date, time and the member's name on it. b. The user can enter his message into the text area present. c. The user clicks on the 'Post' button. d. The message appears on the family homepage as a Post-It note. 5. To reply to a message, the user clicks on the 'Re' button below the Posted message. a. A text box to type in the reply, the name of the member posting the reply and the time appears in a pop-up window.

	b. After typing the reply, the member clicks on 'Post' for the message to be posted. 6. The member clicks on the 'Photos' link a. He is taken to a page where there are thumbnail images of all the photos that have been uploaded thus far. b. Clicking on the thumbnail image takes the user to the full size image. c. The member can upload pictures by clicking on the 'Upload' link on the photos page. i. In the pop-up window, he may either type the name of the file with the path, or browse and select the file name. ii. Once the picture is uploaded, it may be viewed by any member.
Alternate flows	4. If there are more than 30 messages, the user is informed that the earliest message posted will be deleted. a. If the member attempts to send an empty message, he is prompted to enter a message. 5. If there are more than 30 messages, the user is informed that the earliest message posted will be deleted. 6. If the file uploaded is not a known image format, the user is prompted to upload another file.

Use Case ID	UC-05
Use Case Title	Use Address Book
Actors and Roles	Actor: User, Role: Member
Types	-
Corresponding classes	Group, User, Contact, Textual
Corresponding attributes	**Attributes of User:** Username, Password, Name, Address, Telephone number Email address, Birthday, Latest Login, Group name, role. **Attributes of Group:** Group name, Numbers of members, OwnerID, Date created. **Attributes of Contact:** Name, Address, Email id, telephone number
Corresponding interfaces	Address Book.
EBTs	Keep in Touch, Family Reunion
BOs	Textual, User
IOs	Contact, Group
Use case description	4. From the family homepage, the user clicks on the

	'Addresses' link and is taken to the address book. 5. The member can edit contacts by clicking the 'Edit' button that appears next to the name 6. The member can add a contact by clicking on the 'Add New' button. a. A contact form appears. b. The member fills out the form and click on 'Save' c. The new contact is added to the address book or the details of an existing contact are modified.
Alternate flows	3b. If the user clicks on 'Save' without inputting all relevant fields, he is prompted to complete the 'contact' form. c. If the user inputs exact details of an existing contact, he is prompted that the contact already exists.

Use Case ID	UC-06
Use Case Title	Use Calendar
Actors and Roles	Actor: User, Role: Member
Types	-
Corresponding classes	Group, User, Calendar, Textual, Keeping in Touch, Message
Corresponding attributes	**Attributes of User:** Username, Password, Name, Address, Telephone number Email address, Birthday, Latest Login, Group name, role. **Attributes of Group:** Group name, Numbers of members, OwnerID, Date created. **Attributes of Calendar:** Date, Event type, Event Description
Corresponding interfaces	Calendar
EBTs	Keeping in touch
BOs	User, Textual
IOs	Calendar, Message, Group
Use case description	6. The user clicks on the miniature calendar on the homepage to enlarge it. 7. He can view the calendar a. To view the calendar for the previous or next month, he clicks on the 'Previous' and 'Next links below the miniature calendar. b. Similarly the entire year's calendar can be

	viewed by clicking on the year name (Example: 2001, 2002…) 8. To enter events into the calendar, the user clicks on the miniature calendar and is taken to a page with a larger calendar. a. In this page, the user clicks on the particular date and types a description of the event next to the date. b. On that date, a message is posted in the message board with the description just typed.
Alternate flows	3.a. If the user types a description of more than 20 words, he is prompted about the 20 page limit. b. If there is more than one event on a particular date, all events are posted.

Use Case ID	UC-07
Use Case Title	Send Email
Actors and Roles	Actor: User, Role: Member
Types	-
Corresponding classes	Group, User, Email, Contact, Keeping in touch, Share Feelings, Textual
Corresponding attributes	**Attributes of User:** Username, Password, Name, Address, Telephone number Email address, Birthday, Latest Login, Group name, role. **Attributes of Group:** Group name, Numbers of members, OwnerID, Date created. **Attributes of Contact:** Name, Address, Email id, telephone number **Attribute of Email:** From, To, Subject, Date, Message Body
Corresponding interfaces	-Email sending and receiving interface
EBTs	Share Feelings, Keeping in touch
BOs	Textual, User
IOs:	Group, Email, Contact
Use case description	6. Users can send mails to the other members in the group by selecting their mail addresses from the Address Book. 7. He selects the ids by just clicking on their names in the Address Book. a. The user can choose the 'To All Members' option on the Address Book to send an email to all the members of the group. 8. He composes the mail.

	9. He may attach a file to the email. 10. When he clicks 'Send', his mail is sent to all the selected members.
Alternate flows	9. If the size of the file attached is larger than 500Kb, he is informed that the file is too large. 10. If the user does not enter a subject, he is informed of this.

Use Case ID	UC-08
Use Case Title	Chat
Actors and Roles	Actor: User, Role: Member
Types	-
Corresponding classes	Group, User, Chat, Oral, Conversation, Textual, Keeping in touch, Share feeling, Family Reunion
Corresponding attributes	**Attributes of User:** Username, Password, Name, Address, Telephone number Email address, Birthday, Latest Login, Group name, role. **Attributes of Group:** Group name, Numbers of members, Owner ID, Date created. **Attributes of Chat:** Members in Chat, Time started, History, Message File
Corresponding interfaces	Textual and Oral Chatting
EBTs	Share Feelings, Family Reunion, Keeping in touch, Conversation
BOs	User, Textual, Oral
IOs	Group, Chat
Use case description	5. The user can then double click on an 'online' member's name with whom he wants to start chatting. 6. A chat window pops up. 7. In the chat window, he can send and receive messages. 8. If the user wants to add other members to the conversation, he clicks on 'File', 'Add member to Chat' in the chat window. a. From the submenu, he clicks on the particular member's name and that person is added to the Conference.
Alternate flows	-

Updated Classes:

#	Class Name	EBT	BO	IO
1	Family Reunion	X		
2	Share Feelings	X		
3	Keep in touch	X		
4	Conversation	X		
5	Pictorial		X	
6	Textual		X	
7	Oral		X	
8	Photo			X
9	Email			X
10	Message			X
11	Calendar			X
12	Contact			X
13	Chat			X
14	User/Member (Role)		X	
15	Owner			X
16	Group			X

Updated CRC Cards:

Share feelings(role: express emotion)		
Responsibility	**Collaboration**	
	Client	**Server**
Allow Members to share feelings, keep in touch and motivate family reunion.	▪ Family Reunion ▪ Keeping in touch ▪ Group ▪ User ▪ Owner	▪ Members of a family share feelings through communicating and keeping in touch by means of mails, messages, chatting, photos etc. ▪ Members send and receive information.

Keeping in touch (Role: Maintain Contact)		
Responsibility	**Collaboration**	
	Client	**Server**
Allow Members to share feelings, keep in touch .	▪ Family Reunion ▪ Share feelings ▪ Conversation ▪ Pictorial ▪ Group ▪ User ▪ Owner	▪ Members of a family share feelings through communicating and keeping in touch by using textual messages, oral communication. ▪ Brings family members together. ▪ Uses user(member) data to keep in touch.

Group (Role: Family)		
Responsibility	**Collaboration**	
	Client	**Server**
Allow Members to share feelings and keep in touch.	▪ Family Reunion ▪ Keeping in touch ▪ Share feelings ▪ User ▪ Owner	▪ Provide details of all the members ▪ Act as a platform for members to share information and keep in touch ▪ Keep track of the most recent logins of all members ▪ Display list of members ▪ Members of the group communicate through e-mails, messages, chat etc.

User (Role: Member of Family)		
Responsibility	**Collaboration**	
	Client	**Server**
Keep in touch with other members of the group	▪ Group ▪ Keeping in touch ▪ Share feelings	▪ Post Messages, Photos ▪ Provide personal details ▪ Add events to calendar ▪ Chat with other Members ▪ Send Emails ▪ Reply to Posted Messages

Owner (Role: Moderate the Group)		
Responsibility	**Collaboration**	
	Client	**Server**
Creating and Maintaining the Group	▪ User ▪ Group	▪ Create the Group ▪ Add members ▪ Delete members

Conversation (Role: casual dialog)		
Responsibility	**Collaboration**	
	Client	**Server**
To bring together members of a family.	▪ User ▪ Group ▪ Share feelings ▪ Keeping in touch ▪ Oral ▪ Textual	▪ Members express feelings and get together ▪ Members communicate through textual and oral messages. ▪ Textual messages are sent via e-mails and messages. ▪ Oral communication is done by chat.

Pictorial (Role: Visual Communication)		
Responsibility	**Collaboration**	
	Client	**Server**
Share photos among the members of the group	▪ User ▪ Keeping in touch ▪ Group	▪ Members can upload photos ▪ Members can display photos. ▪ Display the number of photos.

Textual(Role: interface for exchanging textual data)		
Responsibility	**Collaboration**	
	Client	**Server**
Allows family members share feelings by communicating via textual messages.	▪ Member ▪ Chat ▪ Group ▪ Message ▪ Contact	▪ Send textual messages via e-mails ▪ Send messages for special events listed in the calendar. ▪ On-line members are listed for chating..

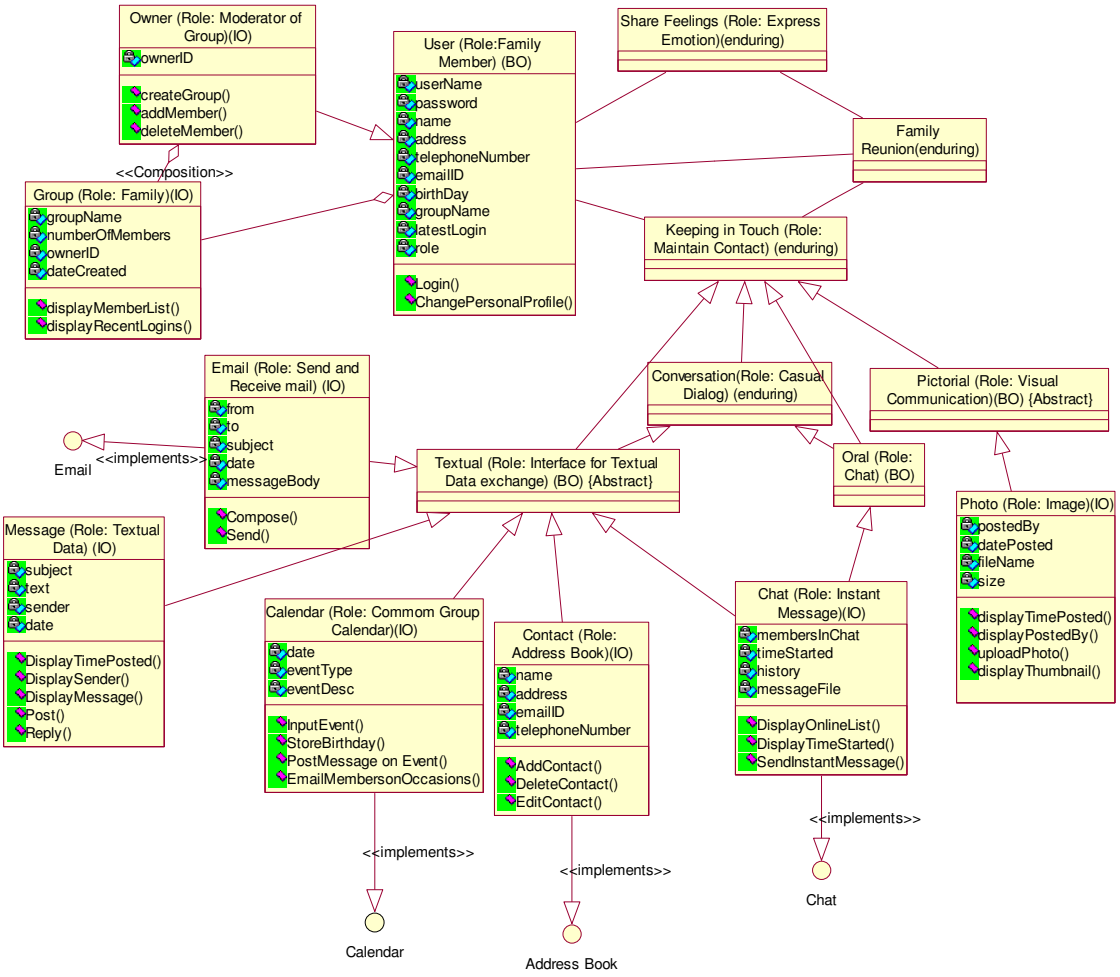
Email (Role: Send and Receive Mails)		
Responsibility	**Collaboration**	
	Client	**Server**
Allow Family Members to communicate by exchanging text messages	▪ Members	▪ Allow Members to keep in touch ▪ Can be sent to multiple members ▪ Attached files can be sent

Calendar (Role: Common calendar of a Group)		
Responsibility	**Collaboration**	
	Client	**Server**
Store and intimate members of events	▪ Members ▪ Email ▪ Information (Messages)	▪ Store events input by members ▪ Store birthdays of members ▪ Post automatic messages on events ▪ Send email to member on special occasions

Chat (Role: Instant Message)		
Responsibility	**Collaboration**	
	Client	**Server**
Allow Family Members to send and receive instant messages	▪ Members	▪ List members currently online Send and receive instant messages among 2 people or a group of people

Contact (Role: Address Book)		
Responsibility	**Collaboration**	
	Client	**Server**
Keep track of contact information	▪ Group ▪ Member	▪ Stores names, addresses, telephone numbers, email ids, etc. of members and their contacts.

UML Class Diagram:



Description of Class Diagram

The Class diagram is a graphical representation of all the classes and the relations they have with the other classes in the model.

In our stable model, the various Classes(EBTs,IOs &Bos) interact and are related in the following way:
(The Classes are italicized below)
Our diagram consists of a *Group.* It is an IO. It consists of many *User* which although is a subclass of an IO is itself a BO. This BO is considerably stable to technological changes.The subclass of this BO is another IO, An 'Owner', who creates the *Group.* The Owner is not a stable class and is subjected to changes.Two or more *Users* communicates through *Chat,* which is an IO. They also can communicate using *E-mail(*an IO) and the *User* shares *Message*, which is also an IO .
The *User* has, the name of the user, his password, his name, his contact details which include his phone number, address, mail id, his birthday and his last login as its attributes. The operation, login(), allows the user to enter his user name and password with which he logs on to the system.
There are a few IOs which are a subclass of an abstract BO *Textual.E-mail,Message,Calendar,*
Contact & Chat which are all IO and can be subjected to technological changes are subclasses of an abstract BO *Textual.* The BO here is considerable stable when compared to its subclass IOs which are highly subjected to changes.The *Chat* is also a subclass of *Oral* which is also a BO. The BOs are subclasses of an EBT *Conversation.* It is highly stable & remains contant even on technological changes.This enduring business theme *Conversation* is a subclass of another EBT *Keeping in Touch.* The EBTs *Keeping in touch and Conversation* are highly stable over a period of time. They are the purpose of the whole model.We intend to *converse* with our family members and thus *keep in touch.*
The BO *Photo* is an abstract business class which as *Photo* ,an IO as its subclass. Photos which are pictorial images can be shared among family members.The abstract BO *Oral* has *Chat* .
Family Reunion which is an EBT is main purpose of our stability model.It achieves a purpose of *Keeping in Touch* and *Share Feelings.* They are the associations of the BO *User* and the EBT *Keeping in Touch.*
Owner which is a subclass of *User* (the BO)has the owner id has its attribute. It creates a new group using the operation createGroup(). AddMember() is used to add a new member into the group while delateMember() is used to delete an existing member from the group.

*Group,*a subclass of the BO *User* has the name of the group, the number of members, the id of the owner and the date the group was created as its attributes .The operation displayMemberList() displays the list of members and displayRecentLogins() displays the logins of the recent users.

Photo (an IO) consists of the date the information was posted and the name of the person the information was posted by as its attributes. The operation displayTimePosted() displays the time a message was posted and displayPostedByWhom() shows the name of the person who posted it.
the message. It has two subclasses: Message and Photo.A Message has a subject & the text. postMessage() posts a message,displayMessage() displays the message and replyToMessage() enables the user to reply to a message that has been posted by clicking on 'Reply' button in the message text window. The filename and the size of the photo are the attributes of the subclass Photo. The operation uploadPhoto() uploads the photo from a location in the computer and displayTumbnailImage() displays a small image which when clicked on enlarges the image for a better and complete view.

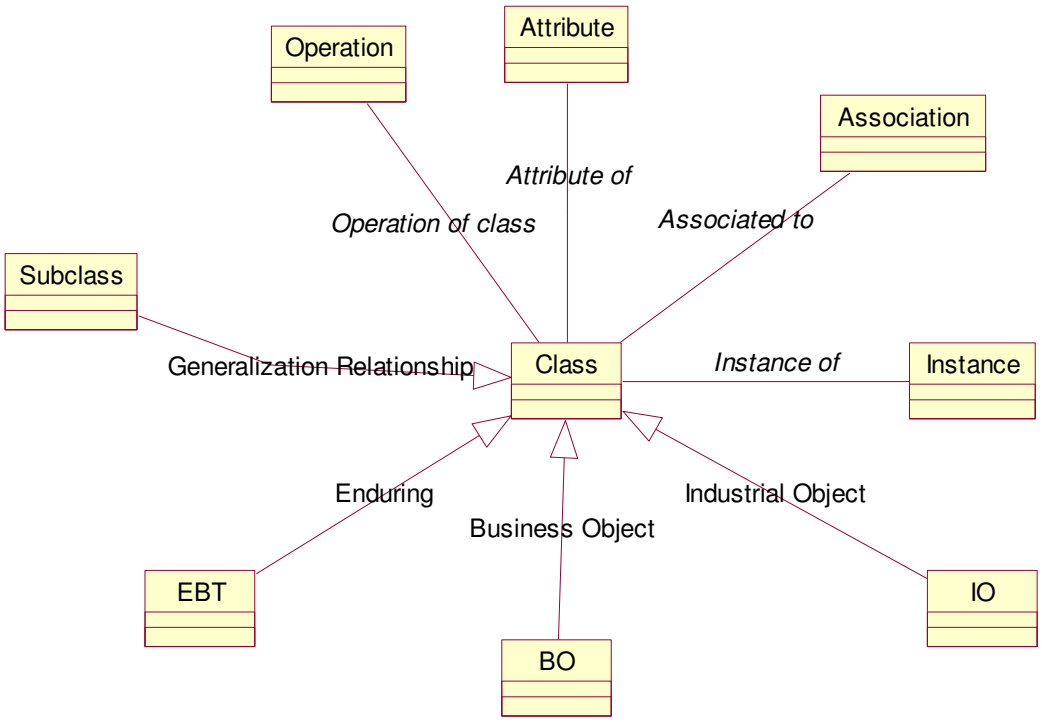
Contact contains the name of the user, his address, his e-mail id and his telephone number as its attributes. The operation addContact(),deleteContact(),editContact() enables the user to update the Address Book or make changes to it.

We have a *Calendar* which as a date, the type of the event and the description of the event as its attributes. InputEvent() inputs the event type into the calendar, storeBirthday() stores a date of the event, postMessageon Event() posts a message on an event and emailMembersOn Occasion() sends an individual e-mail to the person wishing him on the occasion.

E-mail consists of who the message is from, whom it is being sent to, the date it is sent, the subject & message body. The operation compose() is used while the user is filling in the mail template. Send() uses mail protocol to send the e-mail to the recipients.There are three Interfaces to the model namely the Calender,the Address book and Chat.They IO *Calendar* uses the implemantation of the interfact calender,*Contact* an IO uses the implemenation of the Adress Book interfact and *Chat* which is also an IO implements the Chat interface.

Meta Models:

Software Stability Model:



Evaluation of Models using Model Essentials:

A model is a description of something. It is a well-established human process and should allow us to answer questions about a real thing before we build it. It should be simple and capture only those features deemed essential by model building for their goals. After a model is built, it has to be validated and evaluated.

We now evaluate our model based on a few parameters.

Simple	Yes
Complete	No
Stable to technological changes	Yes
Testable	No
Easy to understand	Yes
Visual or graphical	Yes

1*) Simple*: Our stability model basically deals with the design and implementation of a platform for communication of family members and friends. The whole idea behind the design of our model is to achieve an application, which is user-friendly as we have a varied range of users of our model (anyone from a software developer to a simple man who needn't even be computer literate).The use of EBTs and BOs make our model more focused and clear.The analysis of the model now is precise and accurate and there are not many interactions with sources in the model.
We designed our model in a way that is easy to understand and use. There are instructions at each and every step, which lead a user on how to use a specific module in the application making the use of model highly simple.
The Class diagrams given specify how the model functions in a clear and simple way.

2) *Complete:* The purpose of our model is to achieve a platform, which helps the users to keep in touch.
All the modules introduced are depicted clearly in the Class diagrams. The Case diagrams list the corresponding classes and attributes of each Class in the application. The Actor and the Role are specified.
The CRC cards (old and updated) specify the responsibilities and the collaborations.
The responsibilities are stated as generally as possible. The behavior is kept with related information and the information about one thing are kept in a place. The responsibilities are shared among related objects

However, since ours is a small model, it had to be scaled down to a certain extent. All the possible attributes have not been specified. There can be many modes of communication which can use a webcam and many other modules. We have not been able to include all the possible means of communication & the sharing of news and information can only be achieved to a certain extent. As of now,our model is still incomplete.

3) *Stable to technological changes:* When the data in the model is updated or a new module is to be added to our model, it can be easily done. A new CRC card is developed to update the new change. The Class diagrams can be modified so as to fit it the new addition or to delete a class. The use of EBTs now used are adaptable without a change. The Bos are also adaptable through internal change essential. Changes in terms of addition of a new module or may be a change in the existing module can always be added to the model evaluation. The use of EBTs and Bos make out model stable over time.

4) *Testable:* Each feature in our model is traceable to its requirements. However, each module cannot be tested unless a few more additional features are added to the model. We have not included Test Patterns in our model as of now and so we cannot claim that our model is completely Testable under present circumstances.

5)*Easy to understand:* The basic idea behind the design of our model is to achieve an application, which is user-friendly as we have a varied range of users of our .We designed our model in a way that is easy to understand and use. The use of EBTs and BOs make the model analysis complete and precise.It now has become more specific and clear.There are instructions at each and every step, which lead a user on how to use a specific module in the application making the use of model highly simple.
The use of CRC cards clearly specifies the Actor and the Role and the relationships of each class. Our case diagrams are completed in a simple manner, which is easy to understand.

6) *Visual or graphical:* The use of all the Class diagrams and the Use Case diagrams make our model graphical. The CRC cards are easy to understand and are depicted in a simple way. The entire evaluation of the model is highly graphical.

Comparison of the Traditional and Stability Models

Our Traditional model was based on basically on Industrial Objects (IOs).Although simple in its own way, it was not precise and specific. The analysis was not completely focused .
The Stability model includes Enduring Business Themes(EBTs) and Business Objects(BOs).The analysis is now complete in a way.It is more focused and clear. The EBTs are extremely stable and their use in our model makes the model stable to technological changes when compared to the Traditional model which used the IOs which are unstable. The EBTs and BOs are essentials while the IOs in the traditional model are replaceable.
The Stability model is conceptual when compared to the tangible Traditional model.

Behavior Models - MyFamilyReunion
Assignment #4

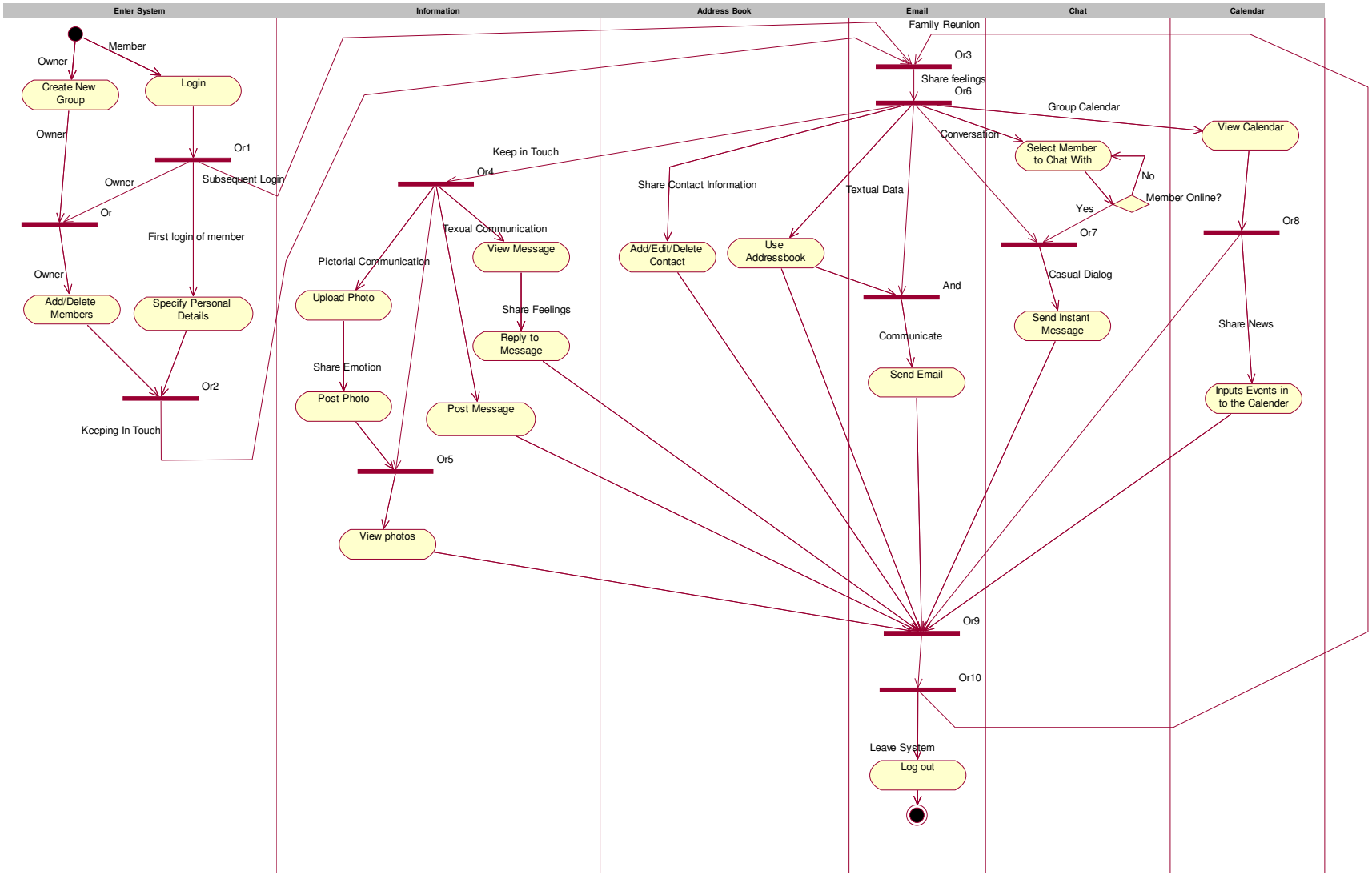
Dr. Fayad

CSCE 866

April 25, 2002

Team X:
Anees Dhala
Sanhitha Seerapu
Vandana Sunkara

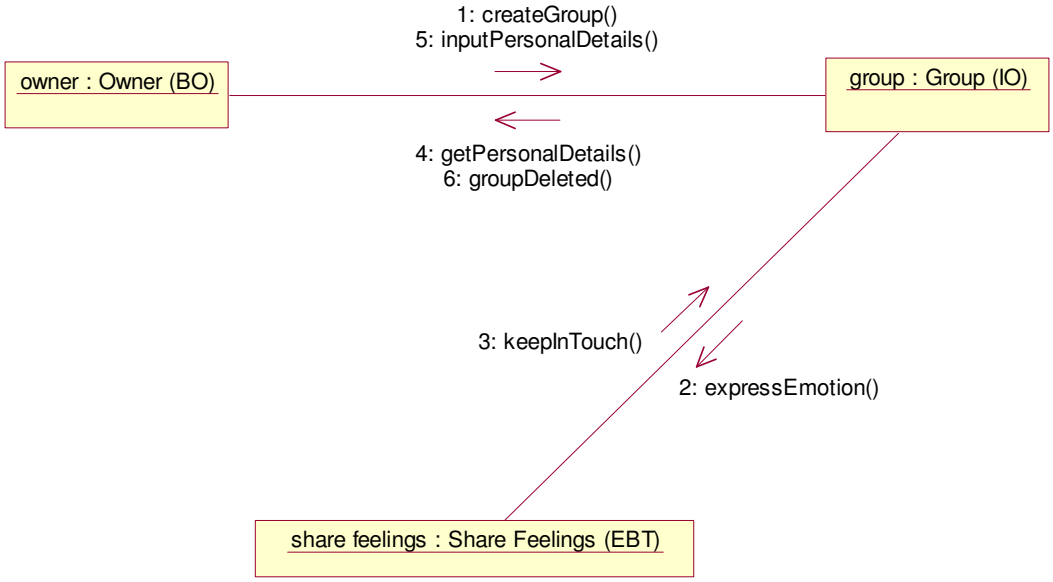
Activity Diagram



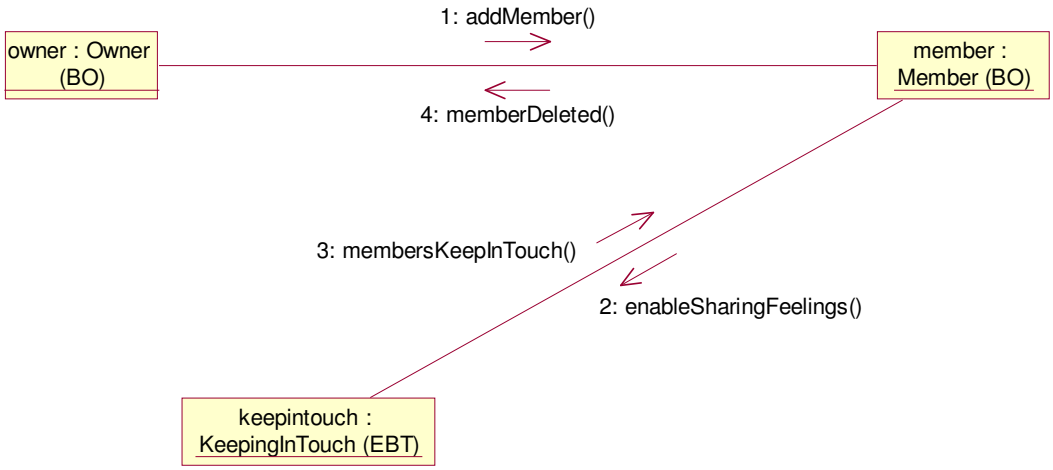
Brief Description:
In the Activity Diagram, we describe the procedural possibilities of our system with the aid of activities (an activity is a single step in a processing procedure).
We subdivided our diagram into responsibility domains called *swimlanes*. We have 6 swimlanes namely, Enter System, Information, Address Book, Email, Chat, Calendar.
The owner can create a new group. He has the ability to add and delete members. When a member logs into the system for the first time, he is required to specify his personal details. From then on, he can start using the features of the system directly after logon. The purpose of our system is to aid in 'keeping touch' and 'sharing feelings'. The members 'keep in touch' by posting messages directly or by viewing and replying to a posted message. The members 'share emotions' by communicating pictorially, in the form of photos. They can 'share contact information' by add/delete/edit contact information. He can use the address book and 'communicate' via e-mail or directly use e-mail. He can 'share feelings' by 'conversing' with a select member of the group who is online. He checks if a member is online and starts a 'casual dialog' by sending an instant message. He can view the calendar and 'share news' by inputting events into the calendar. He can even exit the system after he views the calendar.
A member can exit the system at any point of the time after using one or more of the services. He logs out and leaves the system.

Collaboration Diagrams:

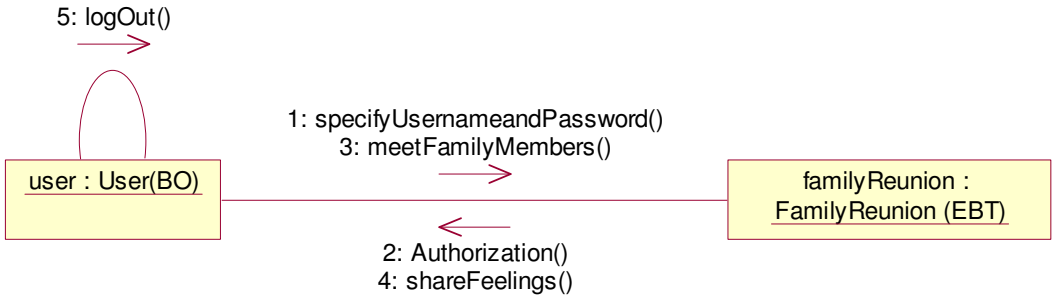
Creating Group:



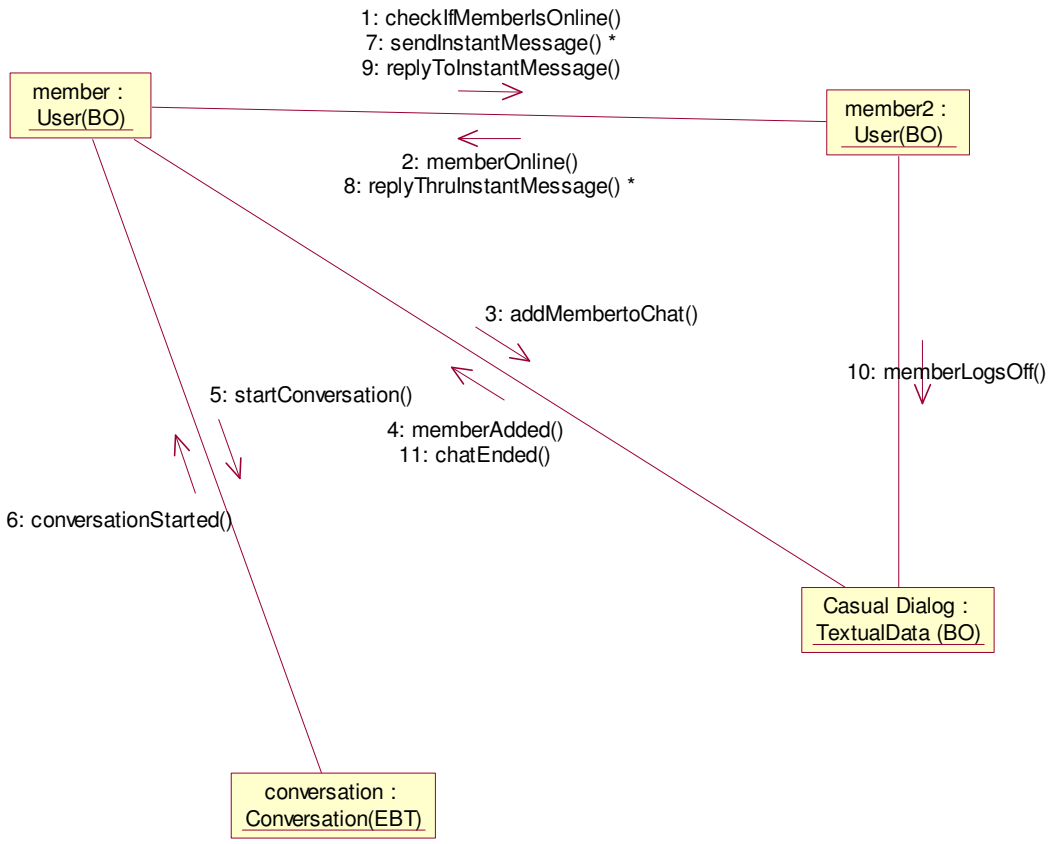
Adding Members:



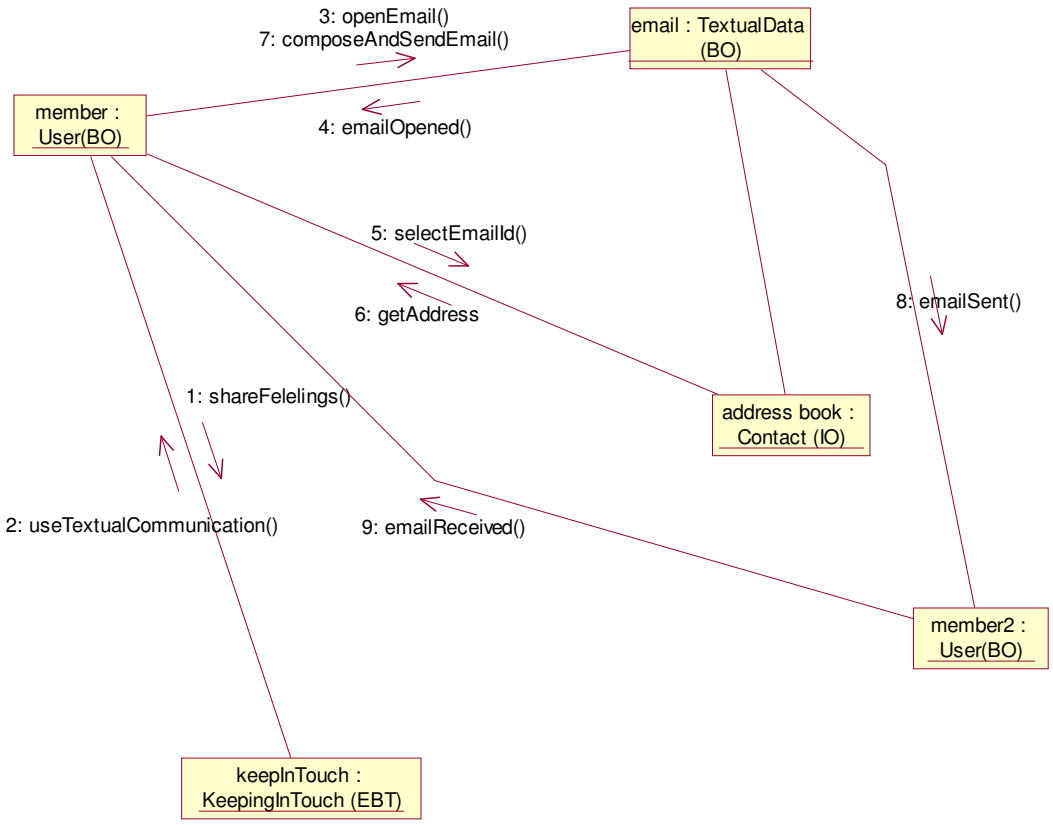
Family Members Logging into System:



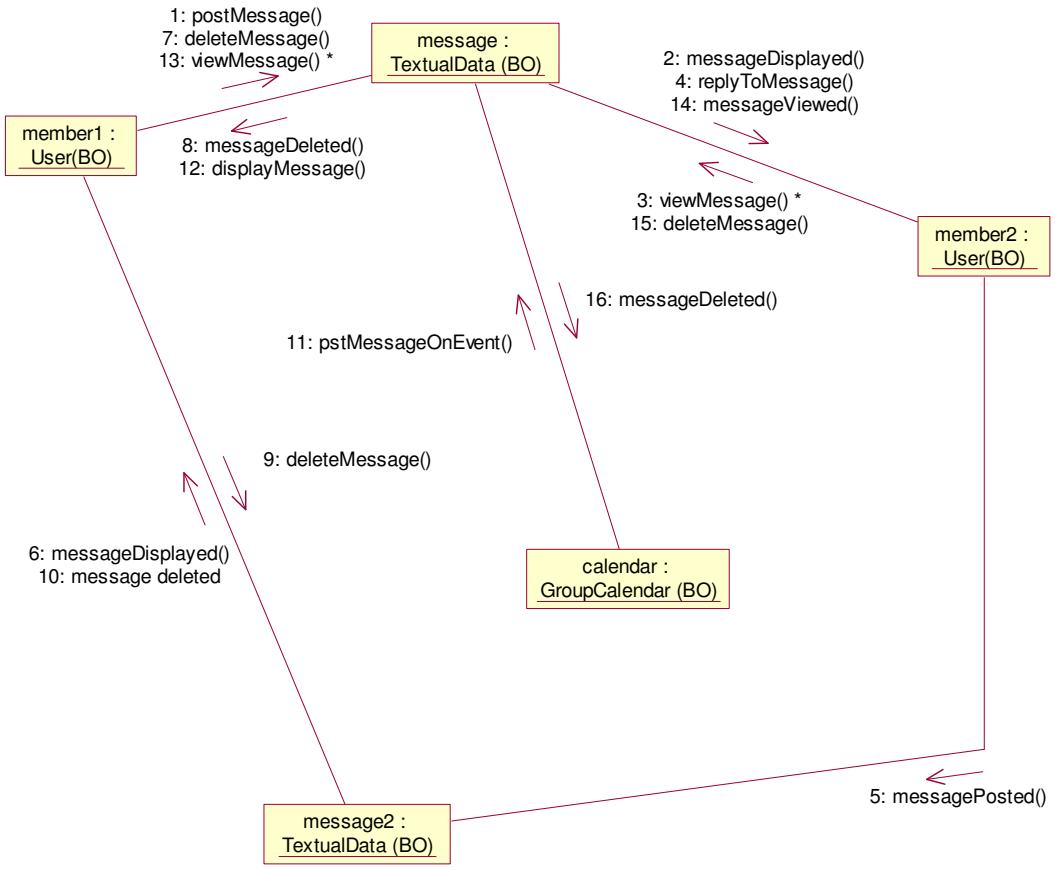
Chatting:


1: checkIfMemberIsOnline()
7: sendInstantMessage() *
9: replyToInstantMessage()
member : User(BO)
member2 : User(BO)
2: memberOnline()
8: replyThruInstantMessage() *
3: addMembertoChat()
10: memberLogsOff()
5: startConversation()
4: memberAdded()
11: chatEnded()
6: conversationStarted()
Casual Dialog : TextualData (BO)
conversation : Conversation(EBT)

Sending Email using the Address Book:



Posting and Deleting Messages:



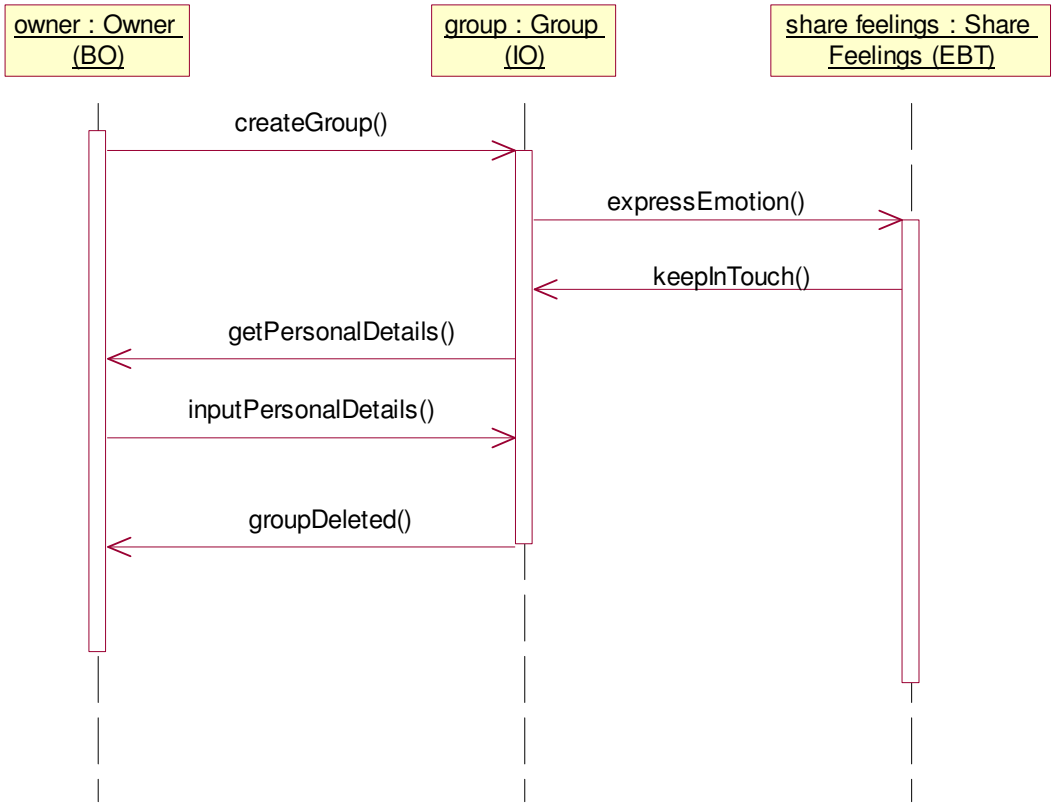
Brief Description:

A Collaboration diagram shows a set of interactions between selected objects in a specific, limited situation, focusing on the relations between the objects and their topology.
The objects interact with one another using messages. The owner creates a group using createGroup( ) . A group interacts to share feelings and express emotions. All members in the group keep in touch. A message can have a response in the form of another message. The owner can delete a group. An owner can add a member using addmember( ), he can also delete a member. The members keep in touch and share emotions while they interact with other objects in the system.
A member can keep in touch by posting messages. He can view, delete and reply to a message. He can post message on Event using calendar. The members online can converse with one another and share feelings. They can exchange casual dialog by sending messages to each other. When a member is online and wishes to converse with another member, he sends him a message

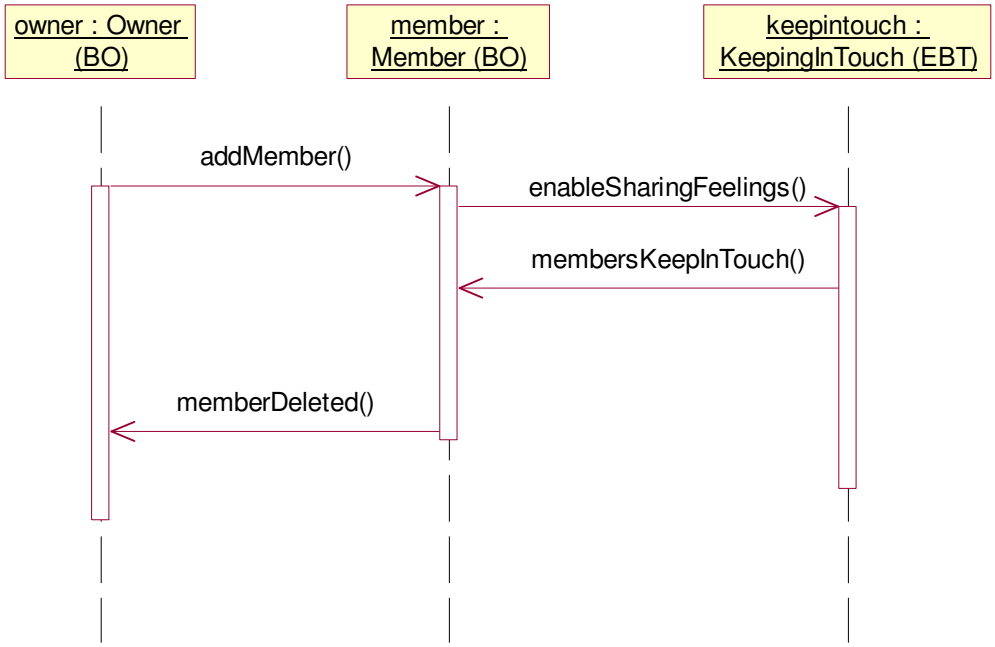
and starts conversation. When they wish to add another member into their conversation, they can do it. They send and receive messages, share feeling and thus keep in touch.

Sequence Diagrams:

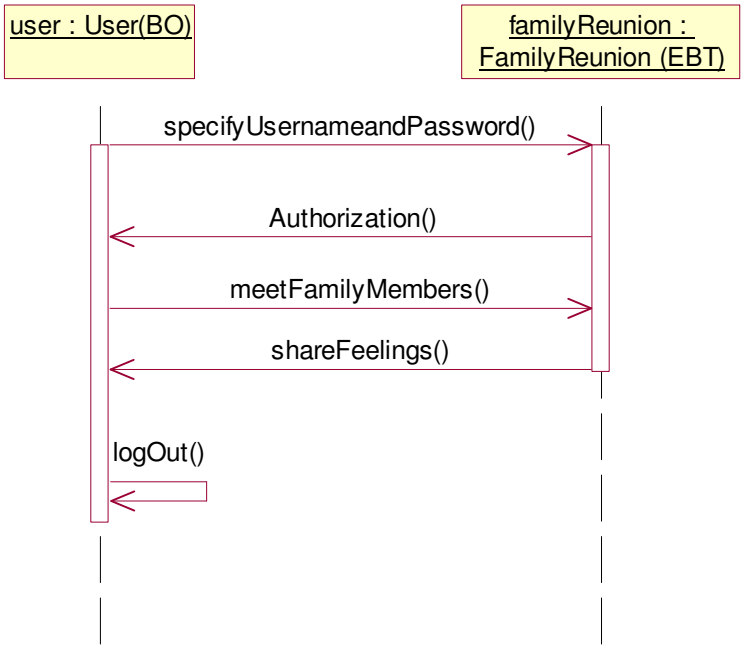
Creating a Group to Keep in Touch with Family:



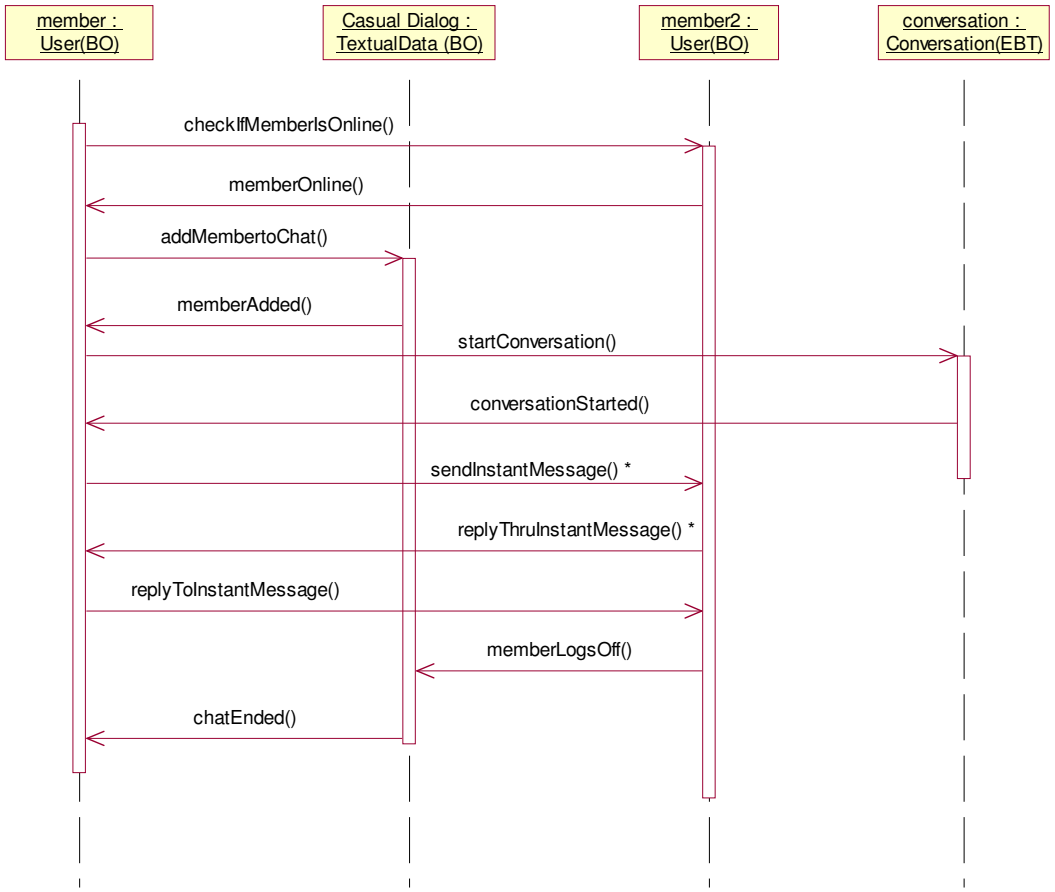
Adding Family Members to the Group:



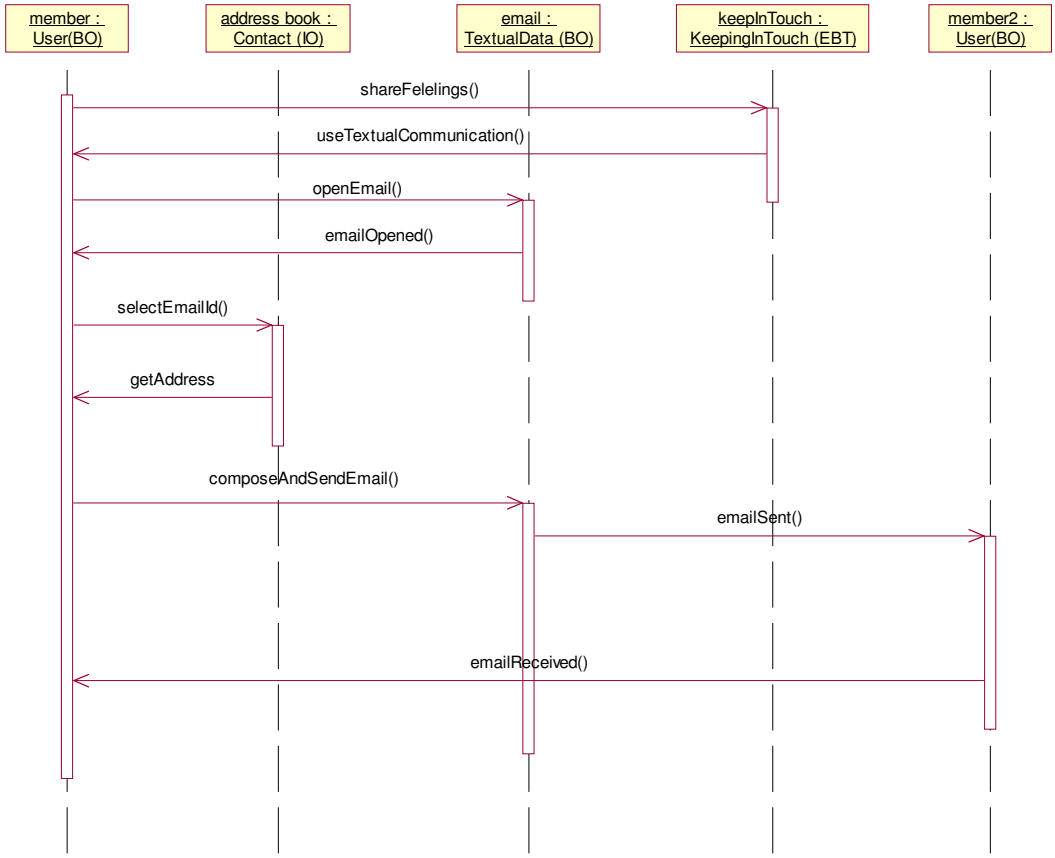
Logging in to the Group:



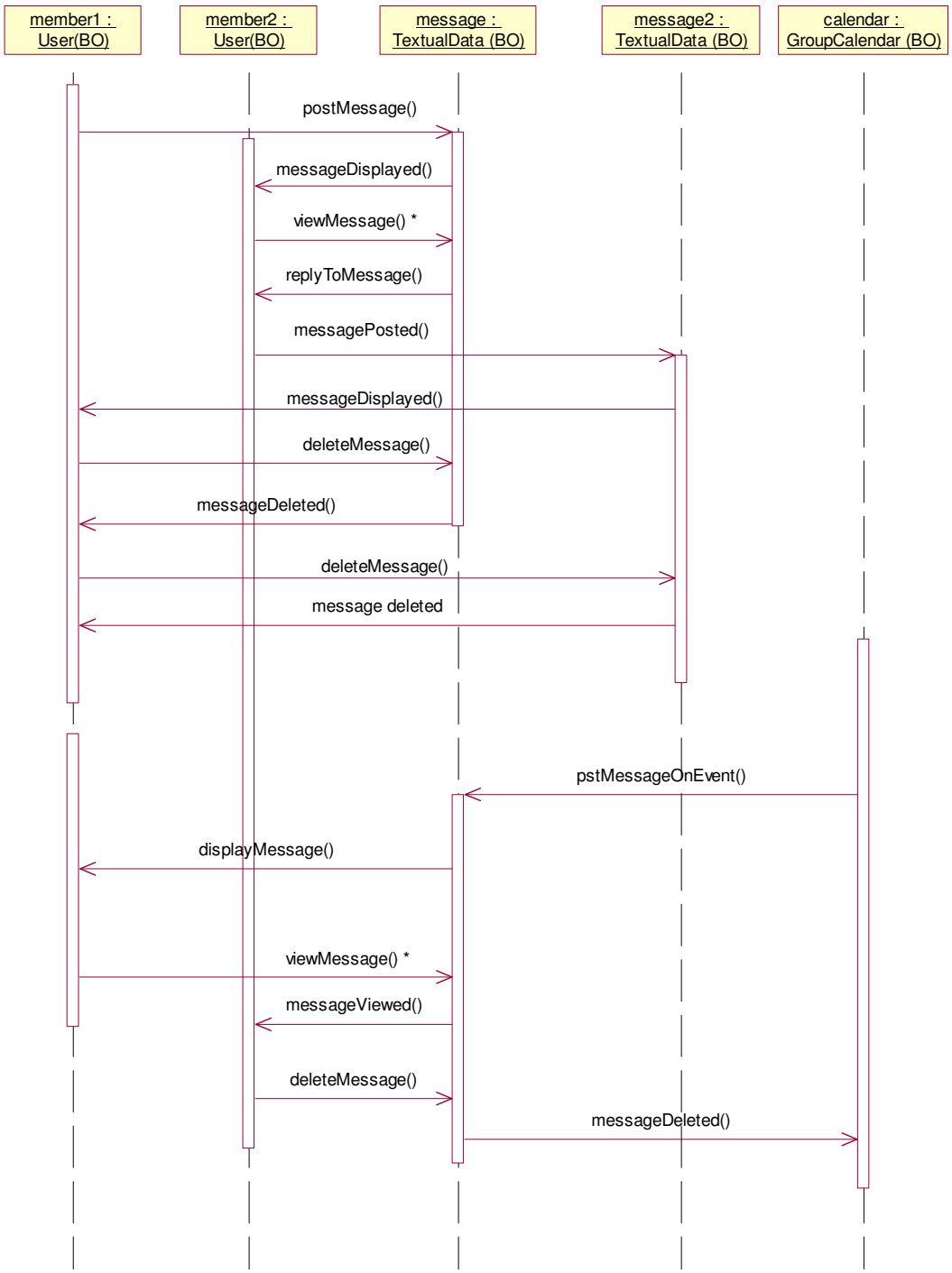
Chatting:


member : User(BO)
Casual Dialog : TextualData (BO)
member2 : User(BO)
conversation : Conversation(EBT)
checkIfMemberIsOnline()
memberOnline()
addMembertoChat()
memberAdded()
startConversation()
conversationStarted()
sendInstantMessage() *
replyThruInstantMessage() *
replyToInstantMessage()
memberLogsOff()
chatEnded()

Sending Email using Address Book:



Posting and Deleting Messages:

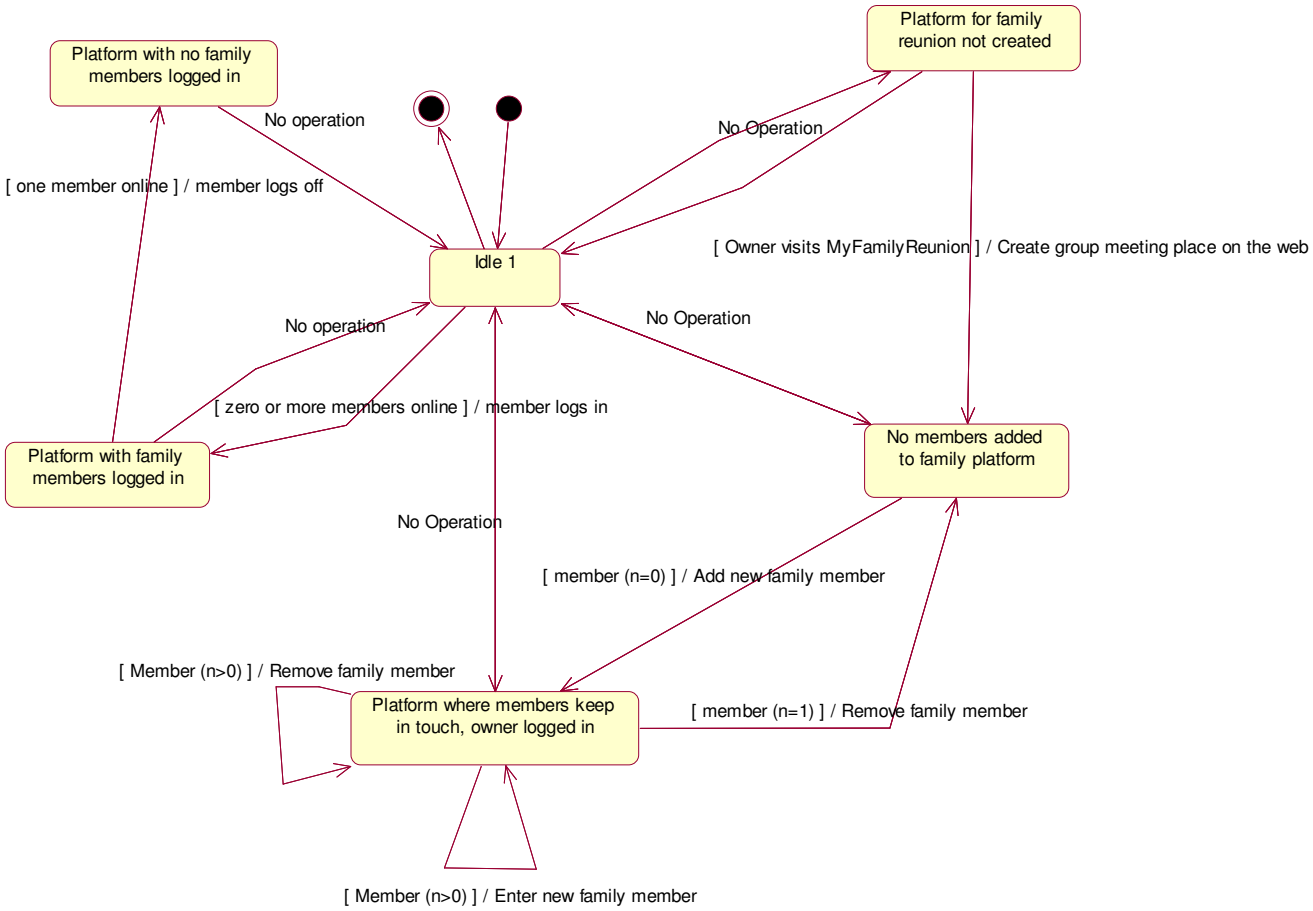


Brief Description:

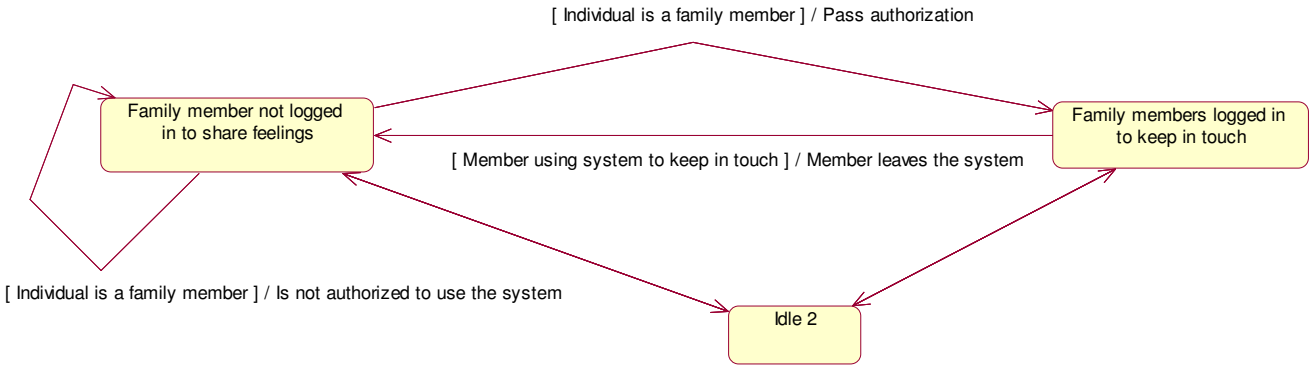
The sequence diagram shows a series of messages exchanged by a selected set of objects in a temporally limited situation, with an emphasis on the chronological course of events.
For a family to get together, a group is created first. The group consists of family members. It's for members of a family to keep in touch. The owner inputs the details of all members. Then he adds members to the group and enables family members to share feelings. The members keep in touch with each other by using various forms of 'conversation'. The owner can also delete a member. The members get username and password as they are added to the system. Members specify their username and password to log on to the system. If they are correct they are authorized to log in. After they login they meet family members and share feelings. They can log out of the system whenever they wish. Members of a family keep in touch by conversation, email, posting messages etc. they can also send instant messages. To chat online, the members check for on-line members and start conversation or send instant messages. The chat can be between two people or a number of people. The members can also send electronic mails to keep in touch. They compose textual data and send it to other member. They can use address book to select the addresses of recipient's. Posting messages is one form of keeping in touch. Members can post messages, view messages and reply to posted messages. The member of a family can use the system whenever he feels to share feelings with his family members.

State Transition Diagrams:

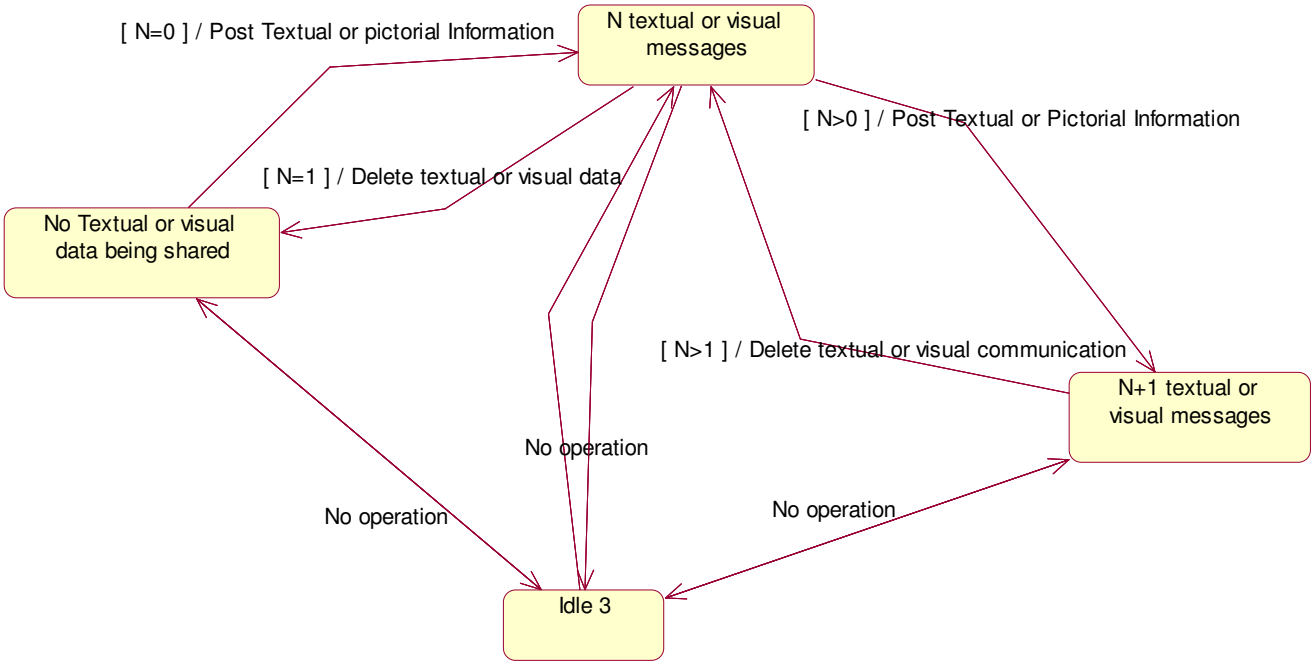
Group



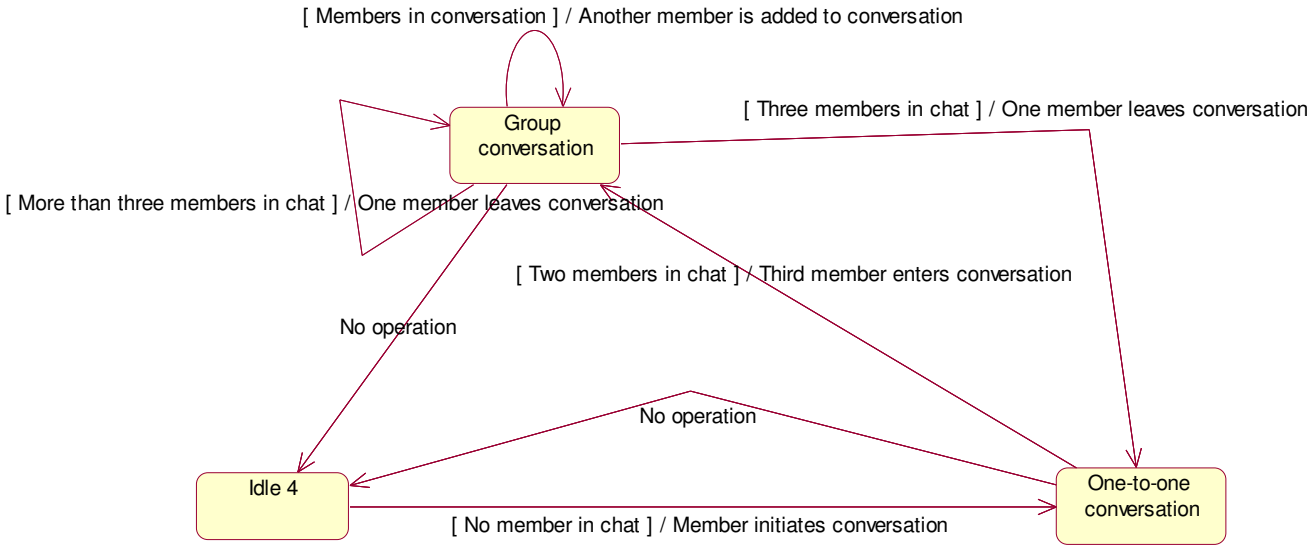
Member



Textual/Visual Information



Conversation

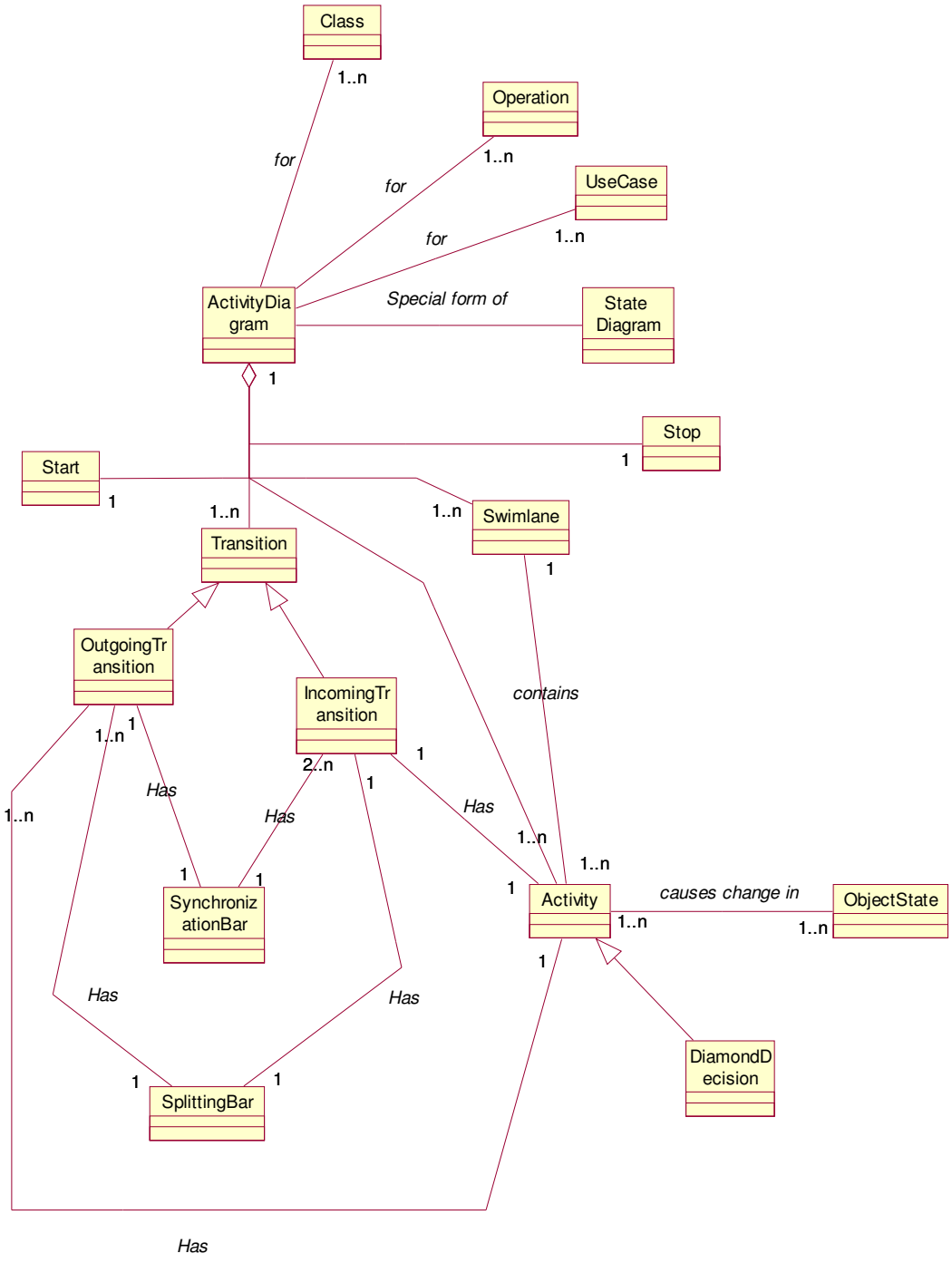


Brief Description:

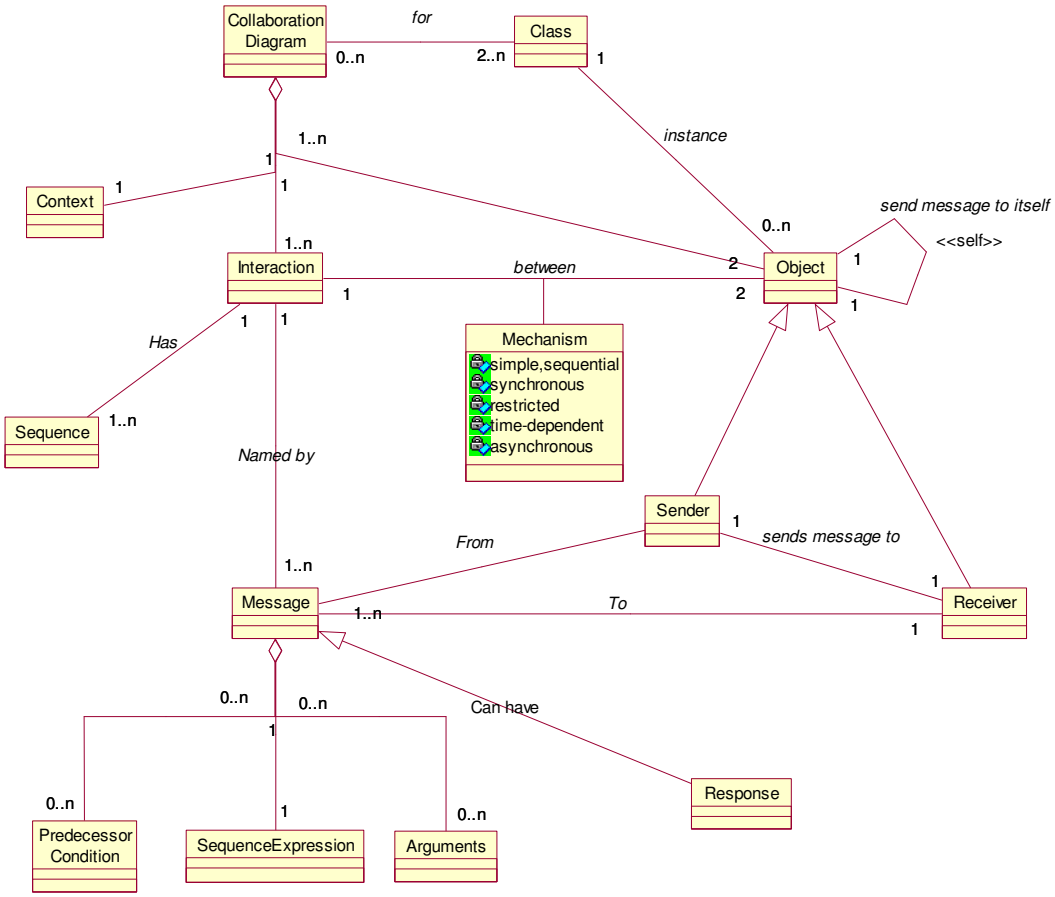
The State Transition Diagram shows a sequence of states an object can assume during its lifetime, together with the stimuli that causes changes of state. We construct state Transition Diagrams for the major classes of our problem.
In the State Diagram of the Group, we initially start by considering a platform with no family members logged in. When a member or two logs on the state changes to a platform with members logged on. A state with Member logged on, gets back to a state with no member logged on when there is 'just one member online and he logs out'. When there are members logged in, they give rise to a state where members keep in touch. When an action is performed, a new state is formed. While the members in the group converse, they give rise to two states. A state with one-one conversation is when two members have a private conversation and when there are more than one member in the conversation another new state is formed, Group Conversation. There are different states associated with how many textual messages are posted at a particular time on the system. Each action performed changes the state. When there are no messages and a message is posted, the state is changes to N textual or visual messages.
'No operation' at any point of time leads to an Idle State.

Meta Models:

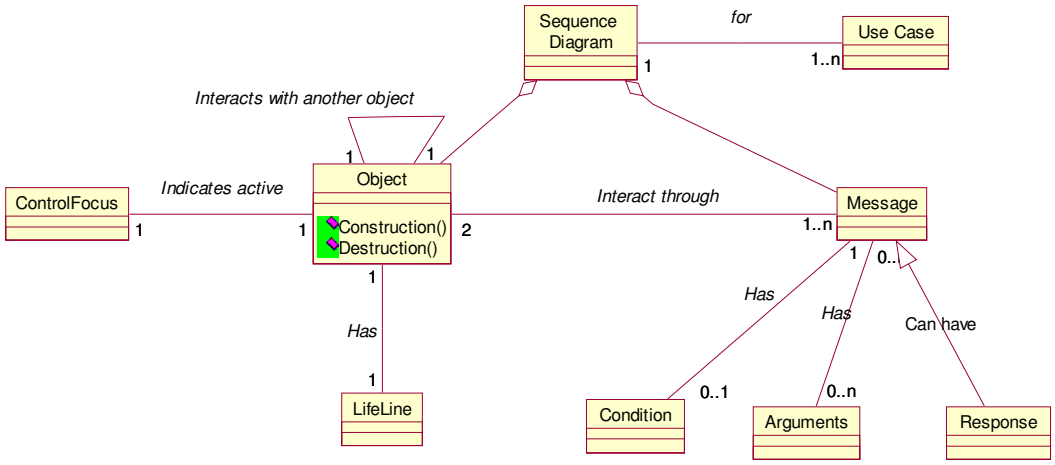
Activity Diagram



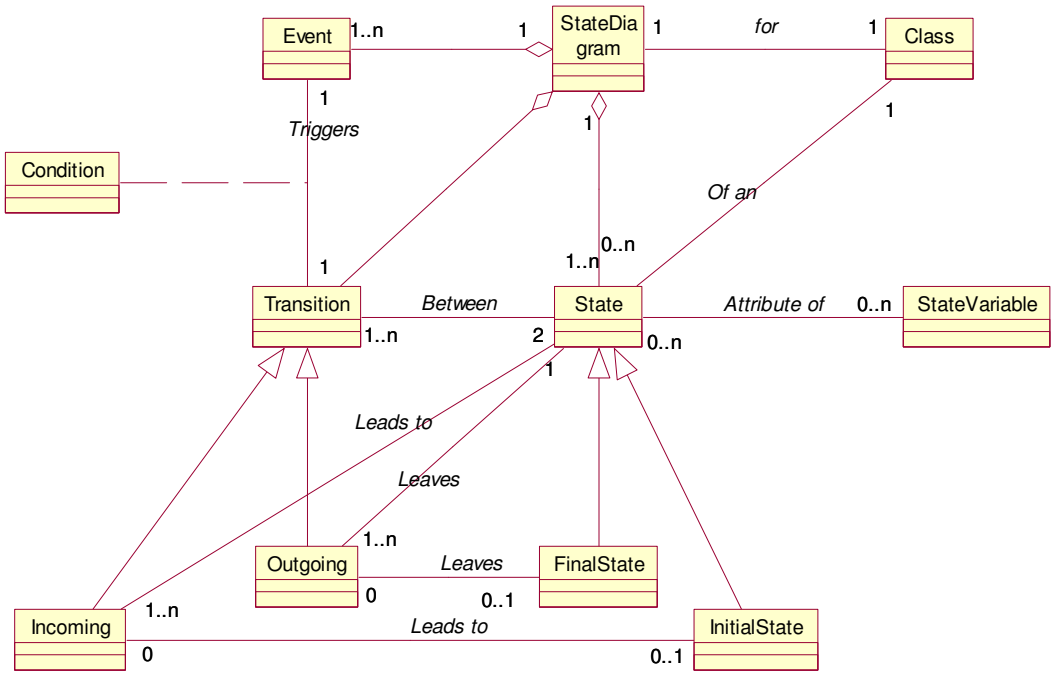
Collaboration Diagram


Collaboration Diagram
for
Class
0..n
2..n
1
instance
1..n
1
Context
1
1
send message to itself
<<self>>
0..n
1..n
Interaction
between
Object
2
1
1
2
1
1
Has
Mechanism
simple,sequential
synchronous
restricted
time-dependent
asynchronous
Sequence
1..n
Named by
Sender
1
sends message to
From
1..n
1
Message
1..n
To
Receiver
1
0..n
0..n
Can have
1
0..n
Response
0..n
Predecessor Condition
1
SequenceExpression
0..n
Arguments

Sequence Diagram



State Transition Diagram



OO Design Heuristics:

1. *Model the real world whenever possible:*

The MyFamilyReunion problem is a real world problem. The members of a family keep in touch to share feelings.

2. *Minimize the number of classes with which another class collaborates.*

In this problem, the number of classes with which another class collaborates is minimum.
For example- in the collaboration diagram of 'Adding members', 'keeping in touch' class is connected to only 'member' class.

3. *A class must know what it contains, but it should not know who contains it.*

The stability design focuses on only design. It does not focus on implementation details. The model focuses on "what a system does?", "what do people do in the system?", "what is a system for?".
MyFamilyReunion is a system, which acts as a platform for family members to keep in touch. People share feelings and keep in touch. The system is for communication, sharing feelings. So, the system knows what it contains, but it doesn't know who contains it.

4. *Derived classes must have knowledge of their base class by definition, but base classes should not know anything about their derives classes.*

In the meta model of state transition diagram, 'initial state' and 'final state' are subclasses of 'state'.
They have same properties as the base class but the base class does not know anything about their derived classes.

Reworked Problem Statement

Problem Domain

The problem is to design a system for members of a group, in particular a family, to keep in touch with each other and share feelings. The idea is to develop a platform for a family to have a reunion.

Description of the Program that is wanted:

The group is to be created by one family member who is designated the owner. This person can then authorize or remove members from using the services of the group. Once a member logs on to the system, he can communicate using textual or pictorial data. Textual data is in the form of email and short messages. Members can post photos for other members to view. There is also a provision for members who are online simultaneously to carry on casual dialog with each other through instant messaging. The

family also has a common address book, containing contact details of the members, as well as any other common contacts, and a common calendar, where group events are shared.

Use Cases:

1. Owner can create a group
2. Owner can add and remove members from the group at any time
3. Members can input and edit personal details.
4. Members can log in and out of the system
5. Members can keep in touch with each other using pictorial communication (photos)
6. Members can post short textual messages on the group bulletin board.
7. Members can reply to posted messages and create a thread of messages.
8. Members can input, edit and delete contact details of common friends in the group address book.
9. Members can use the email facility of the system to send emails. They can pull up the address of the receiver from the address book, if it has been saved there.
10. Members can view a list of other members who are online.
11. Members can send instant messages to each other.
12. Members can reply to instant messages sent to them, and continue the conversation.
13. A member can add more than one person to a conversation, making it a conference.
14. Members can input events into the calendar, and view, edit or delete existing events.
15. The system should post a message (on the bulletin board) with an event description, on the day of the event.

The system should be easy to use, stable and scalable, so that existing features can be removed and new features can be added easily.