Initial Design Report
For
Cloud-SOMS

Prepared by

Erkin Yavuz
Gökçe Çözen
Kazım Buğra Tombul
Sercan Pehlivan

Middle East Technical University
Ankara

27 December 2010

1. Introduction

This document is designed to serve readers to understand the concepts of Cloud-SOMS and to use this tool easily and effectively. The Cloud-SOMS is designed so that individuals and companies can reduce the time and effort to collaborate.

1.1. Problem Definition

Organizations including student clubs, non-governmental organizations, and small and medium size enterprises (SME) have the need for distributed collaboration tool to work together. Up to now, several tools and platforms, such as online collaboration tools, forums, social network groups/events, have been used to overcome this problem. However, there are major missing features in these tools/platforms, some of which can be listed as follows:

1. Role and available time assignment for individuals so that the optimal team can be formed in an event.

2. To-Do list item assignment from Owner/Leader/Chair

3. Keeping track of the state of the assigned items

4. Mobile access

5. Event notification

6. Organization recommendation to users having possible interest to it.

Currently, there is no such a platform, which satisfies all the needs of users mentioned above.

1.2. Purpose

The purpose of this document is to describe requirements for the Cloud-based Scalable Organization Management System with Semantic Web, Mobile Interface and Social Integration (Cloud-SOMS) software that will serve as a foundation for the final product. It's planned to satisfy everyone's expectations. Written descriptions and types of modeling diagrams were used in order to demonstrate the high level structure of the application. Some similar diagrams are used in this document to provide an alternative point of view so that all stakeholders may have a suitable view to their area of responsibility.

Cloud-SOMS is intended to provide a quick, easy and user-friendly cloud-based organization management system. For instance, one can easily manage organizational tasks, which will be operated automatically by the system. The system should be designed so that management-time is minimized. These and many other features of the system will be described in a greater detail.

1.3. Scope

The Cloud-SOMS is a complex, scalable and safe system, which is on web. The purpose of this user-friendly, safe and secure system is to provide organizations including student clubs, non-governmental organizations, and small and medium size enterprises a distributed collaboration tool. It is designed so that organization creation, participation to an organization via search or recommendation of a member, role sharing in an organization according to member's interests and skills, accepting or rejecting an event, integrated personal and organizational calendar, organization/event recommendation according to users' interest, social network connection (Facebook, Twitter etc.), notification of upcoming events via e-mail, showing upcoming events and hot topics for every member and mobile access can be done via Cloud-SOMS. It is assured that none of the information clashes or get harmed. Since all the data are kept in cloud, it is accessible from everywhere in the world. In case it is lost or damaged, it can be easily recovered. The cloud keeps the necessary information of both organizations and individuals. In order to keep the data secure, organized and up to date, Cloud-SOMS provides several functionalities. Those are analyzed detailed in the following pages.

1.4. Overview

This document is organized into several sections.

In section 2, system overview of Cloud-SOMS, which provides a general description of the software system including its functionality and matters, related to the overall system and its design will be given. Then, the goals, objectives and benefits of Cloud-SOMS will be explained briefly. This will provide the basis for the brief description of the final product.

In section 3, special design issues, which need to be addressed or resolved before attempting to devise a complete design solution, will be explained.

Section 4 is reserved for the basic Data Structure Design, which for this project is a No-SQL datastore. While it is separated out here for emphasis, it is really the lowest level of the Architectural Design.

Section 5 is the Architectural Design. This is the heart of the document. It specifies the design entities that collaborate to perform the functionality of the system. Each of these entities has an Abstract Specification and an Interface that expresses the services that it provides to the rest of the system. In turn each design entity is expanded into a set of lower-level design units that collaborate to perform its services.

In Section 6, the functionality of the system from the user's perspective will be described. Then, how the user will be able to use Cloud-SOMS to complete all the expected features and the feedback information that will be displayed for the user will be explained.

In Section 7, libraries and tools, which constitute Cloud-SOMS, will be given in detail.

Time schedule of Cloud-SOMS will be given in a Gantt chart in section 8.

1.5. Definitions, Acronyms and Abbreviations

Cloud-SOMS: Cloud-based Scalable Organization Management System with Social Integration

GUI: Graphical User Interface

SaaS: Software as a service

SME: Small and Medium Enterprise

API: Application Programming Interface

SDK: Software Development Kit

No-SQL: Not only Structured Query Language

KISS: Keep It Short and Simple

DRY: Don't Repeat Yourself

SOC: Separation of Concern

ACL: Access Control List

GQL: Google Query Language

XMPP: Extensible Messaging and Presence Protocol

Python: An interpreted scripting programming language.

Google App Engine Framework: it is a cloud-based development environment.

Google BigTable: It is a Not Only SQL (noSQL) database.

Query Builder and GQL: There are two alternative ways to create queries for BigTable, which are query builder and GQL. They provide easy to express queries similar to natural language and SQL respectively.

HTML/CSS/JavaScript: The basic ingredients for web applications.

JQuery: It is a fast and concise JavaScript Library that simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development.

Google Analytics: This will be used to provide the organizations statistical information about their users.

Facebook Python SDK: This client library is designed to support the Facebook Graph API and the official Facebook JavaScript SDK, which is the canonical way to implement Facebook authentication

Facebook JavaScript SDK: The JavaScript SDK enables you to access all of the features of the Graph API and Dialogs via JavaScript

Twitter API: Twitter exposes its data via an Application Programming Interface (API).

1.6. References

Business IT Online: http://www.businessitonline.com

Zoho: http://www.zoho.com

Qtask: http://www.qtask.com

tinyPM: http://www.tinypm.com/

Google App Engine: http://code.google.com/App Engine/

Gliffy: http://www.gliffy.com

Creately: http://www.creately.com

Software Requirements Specification Template. (n.d.). Retrieved from Process Impact: http://www.processimpact.com/process_assests/srs_template.do
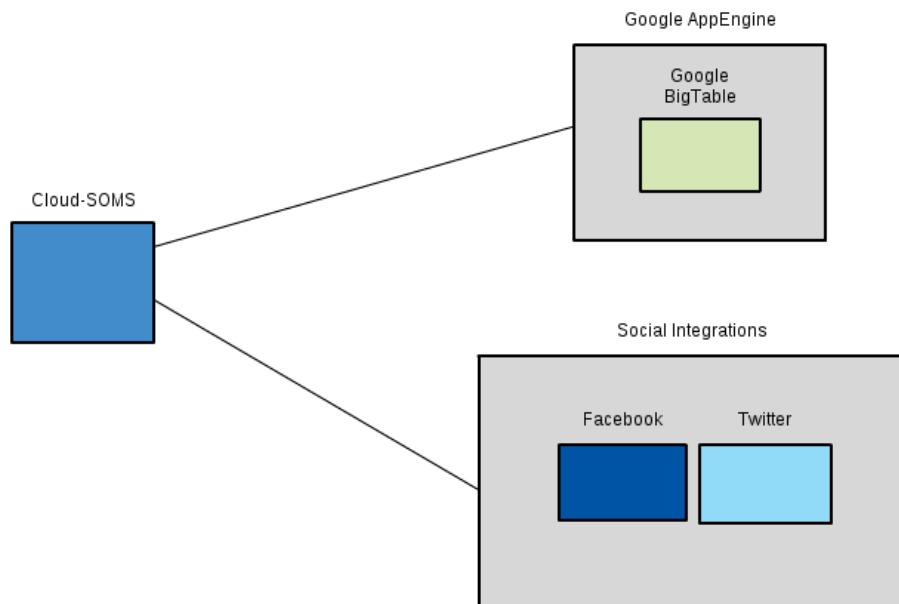
2. System Overview

Figure 1 - Basic Product Structure Diagram

Cloud-SOMS is a self-contained product, is not a component of a larger system.

It will be built on Google App Engine and use Google BigTable as data storage. Google App Engine will provide a cloud-based framework and Google BigTable will provide a cloud

5

based NoSQL data storage. It will also be integrated to social networks such as Facebook and Twitter using their public API's and SDK's.

3. Design Considerations

There are a few things that should be highly considered during design and implementation process. These are explained in the following sections:

3.1. Design Constraints

Many constraints have affected the design decisions for the Cloud-SOMS project. To be able to complete balanced working software in less than a year with the requirements met, these have to be considered.

Time: It is planned to finish the project by the end of May. To complete the project in this interval, a detailed schedule was planned. Each member in Cloud-SOMS will try to follow this schedule.

Performance: Since the project will work on the Internet as a web page the only thing we should consider is reducing response time. To do so, several numbers of tests will be applied frequently to detect problems.

The key issues that must be in our software are:

Reliability: Our project will run without any problems or bugs. There will be no consistency problems. Therefore several tests will be applied in order to provide reliability.

Security: Cloud-SOMS aims to reach millions of people; security is one of the most important issues that must be considered. To achieve this goal, no one can see another user's information without his or her permission. ACL (Access Control List) is used to construct a secure structure.

Availability: Cloud-SOMS working on Google App Engine, this makes our system 99.9% available. If a machine in the cloud is broken another machine in cloud will respond to incoming requests. Availability is one of the major concerns while creating software. As Cloud-SOMS is based on a web page, it will be available as long as Facebook, Twitter is available.

Data, which includes information regarding of users and organizations, should be secured against malicious deformations. Data should not be corrupted in case of system crash or power failure. Google Big Table is used for data storage. Although Cloud-SOMS is highly scalable, secure and maintainable software, it has some limitations due to Google App Engine's daily limitations and quotas (free version), which can be seen from tables below. The software requires a modern web browser such as Mozilla Firefox, Google Chrome for connection between users and Cloud-SOMS.

6

CPU Time	6.50 CPU hours
Requests	43,200,000
Outgoing Bandwidth	1.00 GBytes
Incoming Bandwidth	1.00 GBytes
Secure Requests	43,200,000
Secure Outgoing Bandwidth	1.00 GBytes
Secure Incoming Bandwidth	1.00 GBytes

Table 1 - Google App Engine Request Quotas

Datastore API Calls	141,241,791
Datastore Queries	417,311,168
Blobstore API Calls	141,177,600
Total Stored Data	1.00 GBytes
Blobstore Stored Data	1.00 GBytes
Data Sent to Datastore API	72.00 GBytes
Data Received from Datastore API	696.00 GBytes
Datastore CPU Time	2,487.70 CPU hours
Number of Indexes	200

Table 2 - Google App Engine Storage Quotas

Mail API Calls	7,000
Recipients Emailed	2,000
Admins Emailed	5,000
Message Body Data Sent	0.06 GBytes
Attachments Sent	2,000
Attachment Data Sent	0.10 GBytes

Table 3 - Google App Engine Mail Quotas

UrlFetch API Calls	657,084
UrlFetch Data Sent	4.00 GBytes

UrlFetch Data Received	4.00 GBytes

Table 4 - Google App Engine UrlFetch Quotas

Image Manipulation API Calls	45,273,600
Data Sent to API	562.00 GBytes
Data Received from API	427.00 GBytes
Transformations executed	47,001,600

Table 5 - Google App Engine Image Manipulation Quotas

Memcache API Calls	0 of 192,672,000
Data Sent to API	0.00 of 558.00 GBytes
Data Received from API	0.00 of 640.00 GBytes

Table 6 - Google App Engine Memcache Quotas

XMPP API Calls	46,310,400
XMPP Data Sent	1,046.00 GBytes
Recipients Messaged	46,310,400
Invitations Sent	100,000

Table 7 - Google App Engine XMPP Quotas

Channel API Calls	46,310,400
Channels Created	8,640
Channel Data Sent	1,046.00 GBytes

Table 8 - Google App Engine Channel Quotas

Task Queue API Calls	100,000
Task Queue Stored Task Count	1,000,000
Task Queue Stored Task Bytes	104,857,600

Table 9 - Google App Engine Task Queue Quotas

Deployments	1,000

Table 10 - Google App Engine Deployment Quotas

3.2. Design Goals and Guidelines

The following goals will be achieved for the end product:

Portability: Our project is designed to be platform-independent. Therefore, appropriate libraries are chosen to achieve this goal.

Usability: Usability is the most important issue in our project. Since this project is web-based, users of this project do not have to be experienced using similar programs. To ease use of Cloud-SOMS user-friendly, simple and usable GUI will be implemented.

Principles that are adopted for Cloud-SOMS can be listed as follows:

Don't repeat yourself: This is a principle of software development aimed at reducing repetition of information of all kinds, especially useful in multi-tier architectures. The DRY principle is stated, as every piece of knowledge must have a single, unambiguous, authoritative representation within a system.

KISS: Kiss is an acronym for the design principle Keep it simple, Stupid. Other variations include "keep it short and simple" or "keep it simple and straightforward". The KISS principle states that simplicity should be a key goal in design, and that unnecessary complexity should be avoided.

Separation of Concerns: separation of concerns (SoC) is the process of separating a computer program into distinct features that overlap in functionality as little as possible. A concern is any piece of interest or focus in a program. Typically, concerns are synonymous with features or behaviors. Progress towards SoC is traditionally achieved through modularity of programming and encapsulation (or "transparency" of operation), with the help of information hiding. Layered designs in information systems are also often based on separation of concerns (e.g., presentation layer, business logic layer, data access layer, database layer).

4. Data Design

4.1. Data Description

Users, organization, event, task objects will be managed and manipulated by Cloud-SOMS.

4.1.1. Data object

Cloud-SOMS system has:

status: state of the system (active or passive) is kept in this field.

systemAdministrators: the list of users who manage the Cloud-SOMS system are kept in this field.

User has:

id: a unique identifier of the user.

username: each user has a unique username.

name: each user has a real name of the user.

invitations: the list of invitation that comes to a user is kept in this field.

organizations: organizations that a user belongs to is kept in this field.

organizationRoles: the role of a user in an organization is kept in this field.

Organization has:

id: unique identifier of an organization.

name: each organization has a unique organization name.

members: each organization has a member list and this list is kept in this field.

admins: each organization has at least one admin.

requests: requests that come to an organization is kept in this field.

Event has:

id: unique identifier of an event

name: each event has a name

dateTime: each event has a time interval.

place: each event has a place.

guests: the list of users who attend an event is kept in this field.

Task has:

id: unique identifier of a task.

name: each event has a name.

percent: each task has a percent that shows how proportion of work has done so far.

owner: each task is created by an owner.

dueDate: each task has a due-date.

4.1.2. Complete Data Model and Relationships

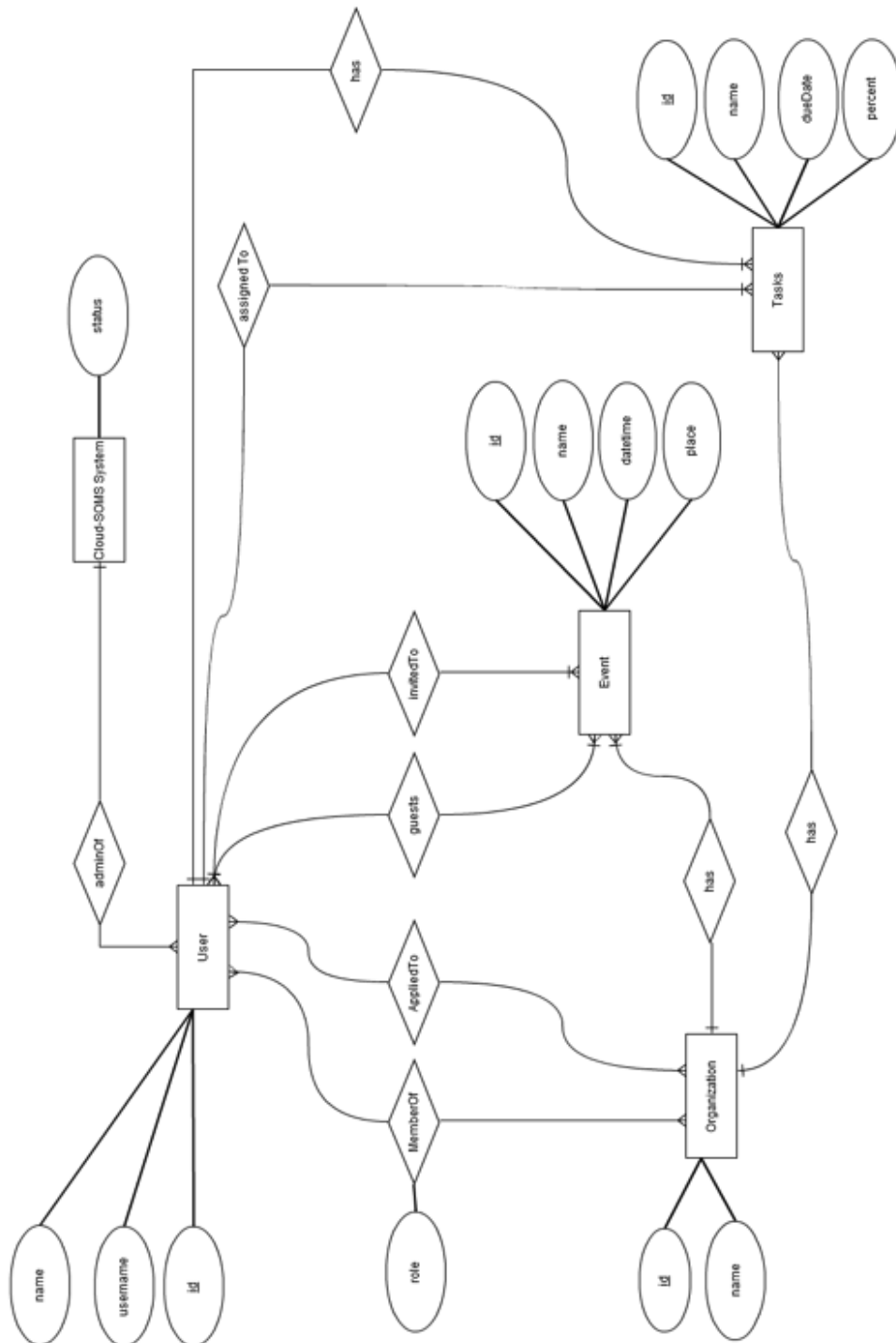Complete data model and relationship of Cloud-SOMS is shown in the figure below.

Figure 2 - Complete data model and relationships

5. System Architecture

The system architecture of Cloud-SOMS is described in detail in this section.
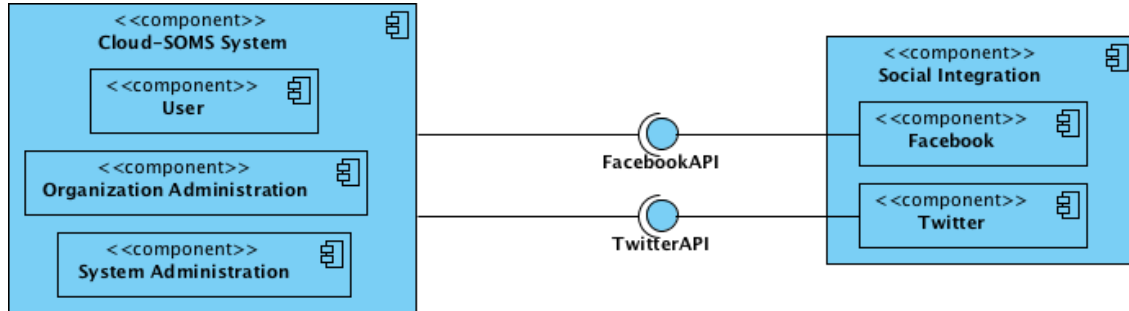
5.1. Architectural Design

Figure 3 - Component Diagram

5.2. Description of Components

5.2.1. Cloud-SOMS System Component

The Cloud-SOMS System Component consists of three sub-components.

5.2.1.1. User Component

This component provides user management interface.

5.2.1.1.1. Processing narrative for User Component

Users interact with users using this component. To activate this component a user should be logged in to the system. This component will be deactivated when the user logs out or when the user interacts with organizations or the system.

5.2.1.1.2. User Component Interface Description

This component provides user management interface. Creating, viewing, editing and deleting users, user organizations, user events and user tasks are done via this interface.

5.2.1.1.3. User Component Processing Detail

This component interacts with System Administration component since all operations that User component requires datastore communication. It also interacts with Organization Administration component.

5.2.1.1.4. Dynamic Behavior of User Component

The dynamic behavior of User Component is shown in Figure 4, Figure 13 and Table 12.

5.2.1.2. *Organization Administration Component*

This component provides organization management interface.

5.2.1.2.1. Processing narrative for Organization Administration Component

Users interact with organizations using this component. To activate this component a user should be logged in to the system. This component will be deactivated when the user logs out or when the user interacts with users or the system.

5.2.1.2.2. Organization Administration Component Interface Description

This component provides organization management interface. Creating, viewing, editing and deleting organizations, organization members, organization events and organization tasks are done via this interface.

5.2.1.2.3. Organization Administration Component Processing Detail

This component interacts with System Administration component since all operations that Organization Administration component requires datastore communication. It also interacts with User component.

5.2.1.2.4. Dynamic Behavior of Organization Administration Component

The dynamic behavior of Organization Administration Component is shown in Figure 4, Figure 14 and Table 12.

5.2.1.3. *System Administration Component*

This component provides system administration interface.

5.2.1.3.1. Processing narrative for System Administration Component

Users interact with the system using this component. To activate this component a user should be logged in to the system. This component will be deactivated when the user logs out or when the user interacts with organizations or the system.

5.2.1.3.2. System Administration Component Interface Description

This component provides system administration interface. Creating, viewing, editing and deleting organizations and users, activating and deactivating the system, and datastore operations are done via this interface.

5.2.1.3.3. System Administration Component Processing Detail

Since this component includes operations on system users and organizations, it interacts with User and Organization Administration components.

5.2.1.3.4. Dynamic Behavior of System Administration Component

The dynamic behavior of Organization Administration Component is shown in Figure 6, Figure 12 and Table 12.

5.2.2. Social Integration Component

The Social Integration Component consists of two sub-components.

5.2.2.1. Facebook Component

This component provides interface for Facebook integration.

5.2.2.1.1. Processing narrative for Facebook Component

Users interact with Facebook using this component.

5.2.2.1.2. Facebook Component Interface Description

This component provides interface for Facebook integration. Signup, sharing information and promoting organizations and their activities are done via this interface.

5.2.2.1.3. Facebook Component Processing Detail

Since this component requires access to information of users and organizations, it interacts with User and Organization Administration components.

5.2.2.2. Twitter Component

This component provides interface for Twitter integration.

5.2.2.2.1. Processing narrative for Twitter Component

Users interact with Twitter using this component.

5.2.2.2.2. Twitter Component Interface Description

This component provides interface for Twitter integration. Sharing information and promoting organizations and their activities are done via this interface.

5.2.2.2.3. Twitter Component Processing Detail

Since this component requires access to information of users and organizations, it interacts with User and Organization Administration components.

5.3. Class Diagrams

User −id : int −username : string −name : string −password : string −email : string −organizations : Organization[] −organizationRoles : HashMap<Organization, enum> −invitation : Organization[] −events : Event[] −tasks : Task[] +getId() : int +getUsername() : string +getName() : string +getPassword() : string +getEmail() : string +getOrganizations() : Organization [] +getOrganizationRoles() : HashMap<Organization, enum> +getOrganizationRole(Organization) : enum +getInvitations() : Organization [] +getEvents() : Event [] +getTasks() : Task [] +setId(id) : void +setUsername(username : string) : void +setName(name : string) : void +setPassword(password : string) : void +setEmail(email : string) : void +addOrganization(Organization) : void +removeOrganization(Organization) : void +addOrganizationRole(Organization, enum) : void +removeOrganizationRole(Organization, enum) : void +addInvitation(Organization) : void +removeInvitation(Organization) : void +addEvent(Event) : void +removeEvent(Event) : void +addTask(Task) : void +removeTask(Task) : void +viewUser() : void +editUser() : void +viewInvitations() : void +viewCalendar() : void	This class contains information about users of Cloud-SOMS. Users can view and edit their information, events, tasks, calendar and respond to invitations.
Organization −id : int −name : String −members : User[] −administrators : User[] −requests : User[] −events : Event[] −tasks : Task[] +getId() : int +getName() : String +getMembers() : User [] +getAdministrators() : User[] +getRequests() : User[] +getEvents() : Event[] +getTasks() : Task[] +setId(id : int) : void +setName(name : String) : void +addMember(User) : void +removeMember(User) : void +addAdministrator(User) : void +removeAdministrator(User) : void +addRequest(User) : void +removeRequest(User) : void +addEvent(Event) : void +removeEvent(Event) : void +addTask(Task) : void +removeTask(Task) : void +respondToRequest(User) : void +editProfile() : void +assignRole(User, enum) : void +viewOrganization() : void +applyToOrganization(User) : void +respondToInvitation() : void	This class contains information about organizations in Cloud-SOMS. Users can view and edit organizational information, events, tasks, calendar and respond to invitations.

Event −id : int −name : String −dateTime : dateTime −place : String −guests : User[] +getId() : int +getName() : String +getDateTime() : dateTime +getPlace() : String +getGuests() : User[] +setId(id : int) : void +setName(name : String) : void +setDateTime(dateTime : dateTime) : void +setPlace(place : String) : void +addGuest(User) : void +removeGuest(User) : void +updateEvent() : void +respondToInvitation(User) : void	This class contains information about events and guests. A user can create an event and invite other users to that event.
Task −id : int −name : String −percent : float −owner : User −dueDate : DateTime +getId() : int +setId(id : int) : void +getName() : String +setName(name : String) : void +getPercent() : float +setPercent(percent : float) : void +getOwner() : User +setOwner(owner : User) : void +getDueDate() : DateTime +setDueDate(dueDate : DateTime) : void +updateTask() : void +assignTask(User) : void	This class contains information about tasks. A user can create a task and can assign it to other users.
DatabaseManager +createOrganization() : void +createEvent() : void +createTask() : void +saveUser(User) : void +saveOrganization(Organization) : void +saveEvent(Event) : void +saveTask(Task) : void +saveSystemState(enum) : void +deleteUser(User) : void +deleteOrganization(Organization) : void +deleteEvent(Event) : void +deleteTask(Task) : Task +searchOrganizations(string) : Organization [] +getOrganizationEvents(Organization) : Event [] +getOrganizationTasks(Organization) : Task [] +getUserEvents(User) : Event [] +getUserTasks(User) : Task [] +getUpcomingEvents(User) : Event []	All datastore operations are managed by this class.
SocialIntegration +recommend(Organization) : void	Social network integration is handled by using this class.

Table 11 - Class Diagrams

16

5.4.   Sequence Diagrams

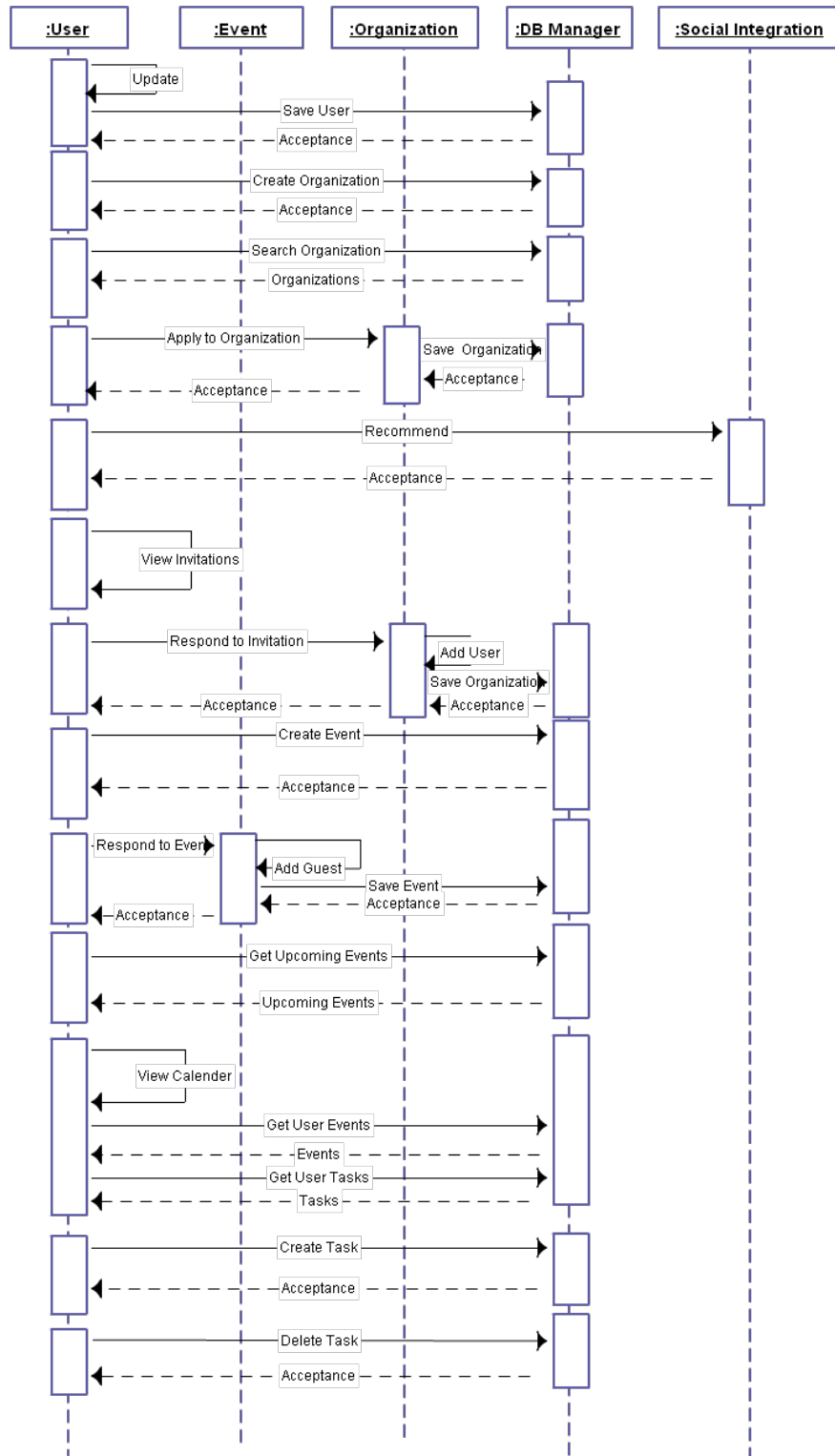Sequence diagrams of the Cloud-SOMS software are shown below.
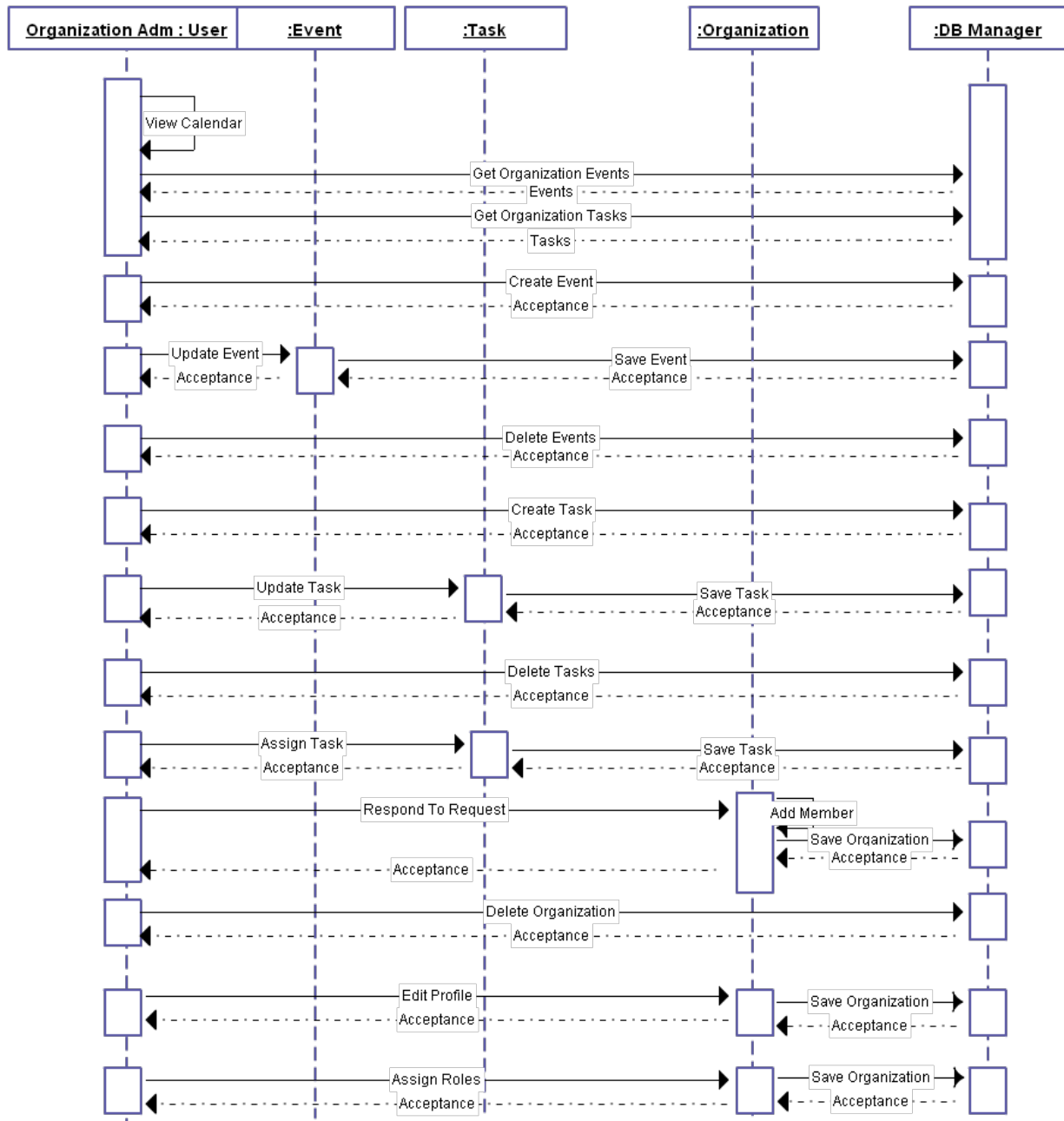

Figure 4 - User Sequence Diagram

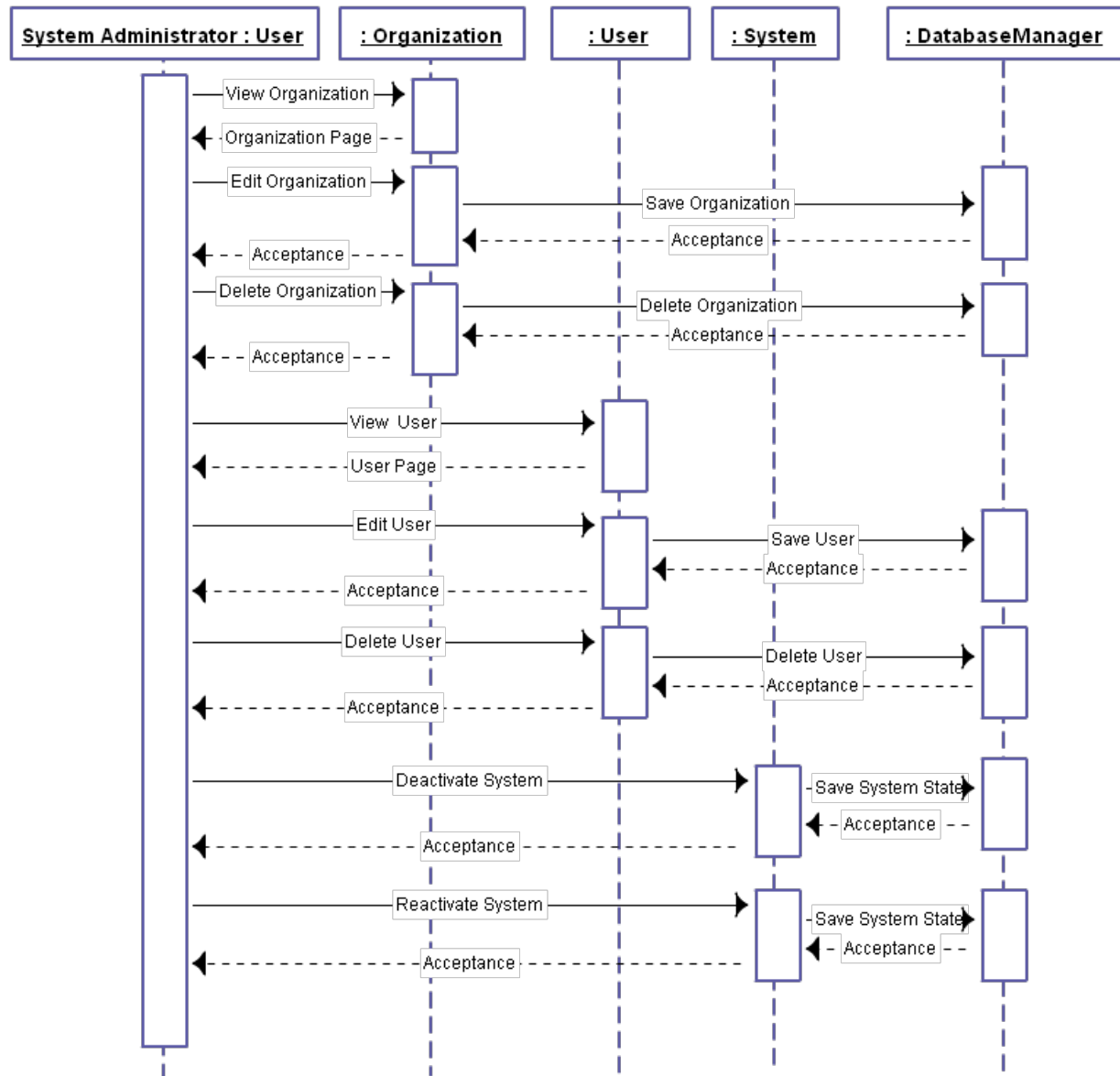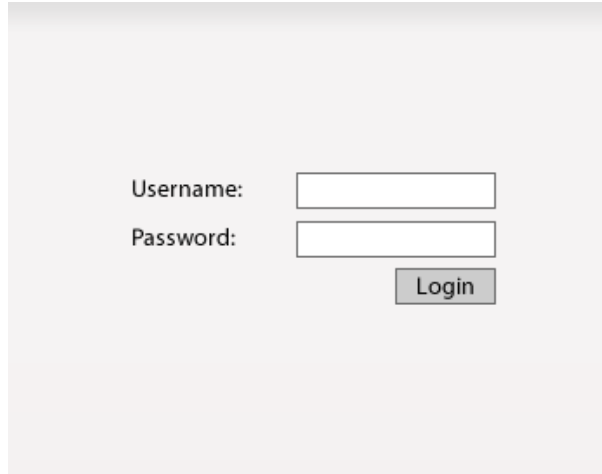**Figure 5 - Organization Administrator Sequence Diagram**

Figure 6 - System Administrator Sequence Diagram

19

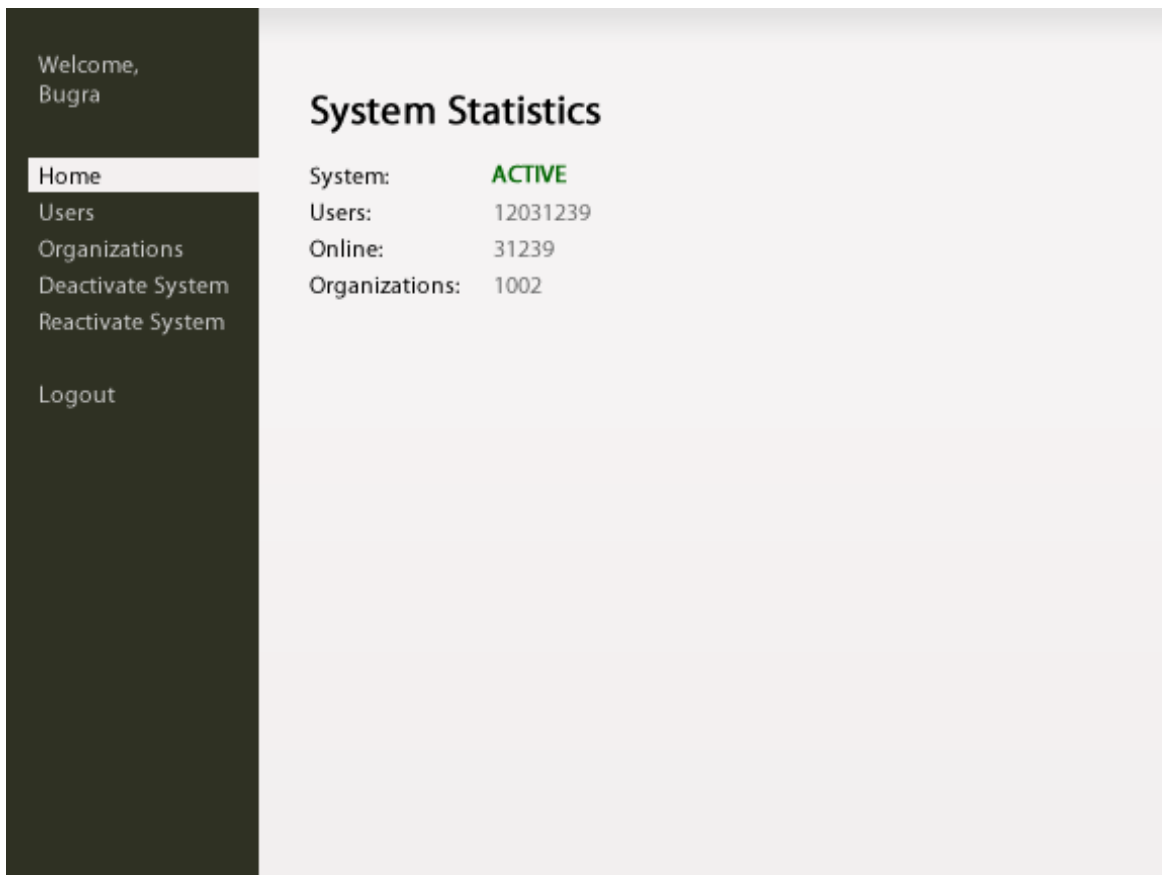# 6. User Interface Design

6.1. Overview of User Interface

6.2. Screen Images

These are screen images and user interfaces of Cloud-SOMS.

Figure 7 - Login Page
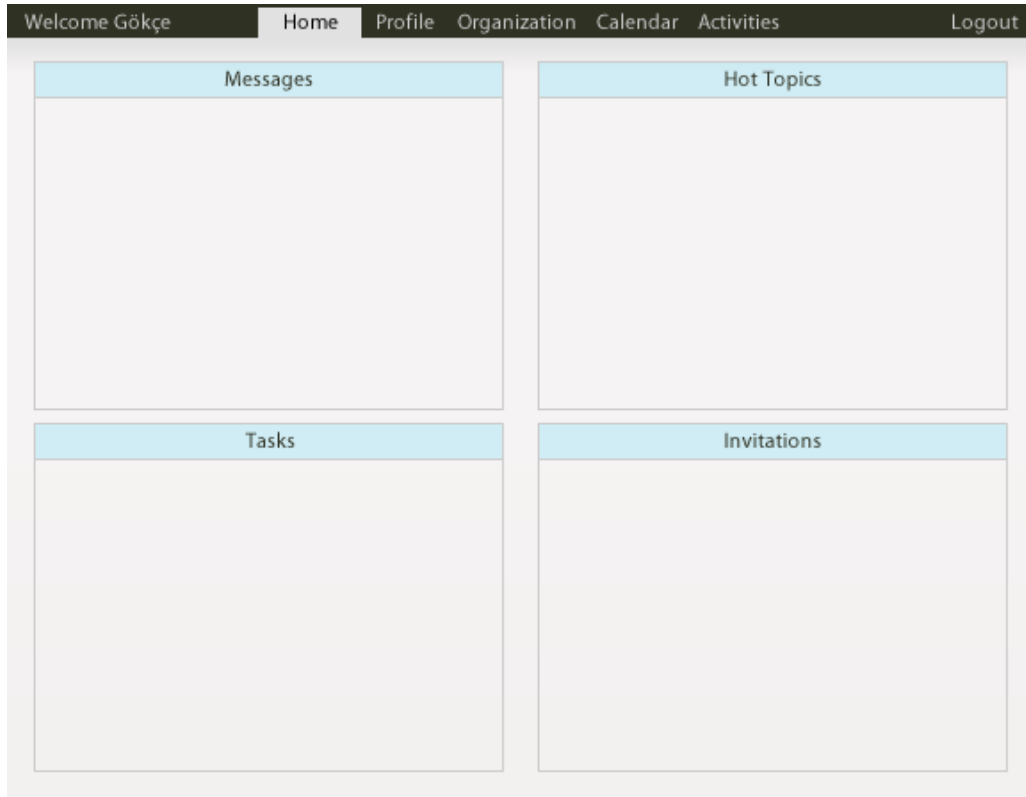

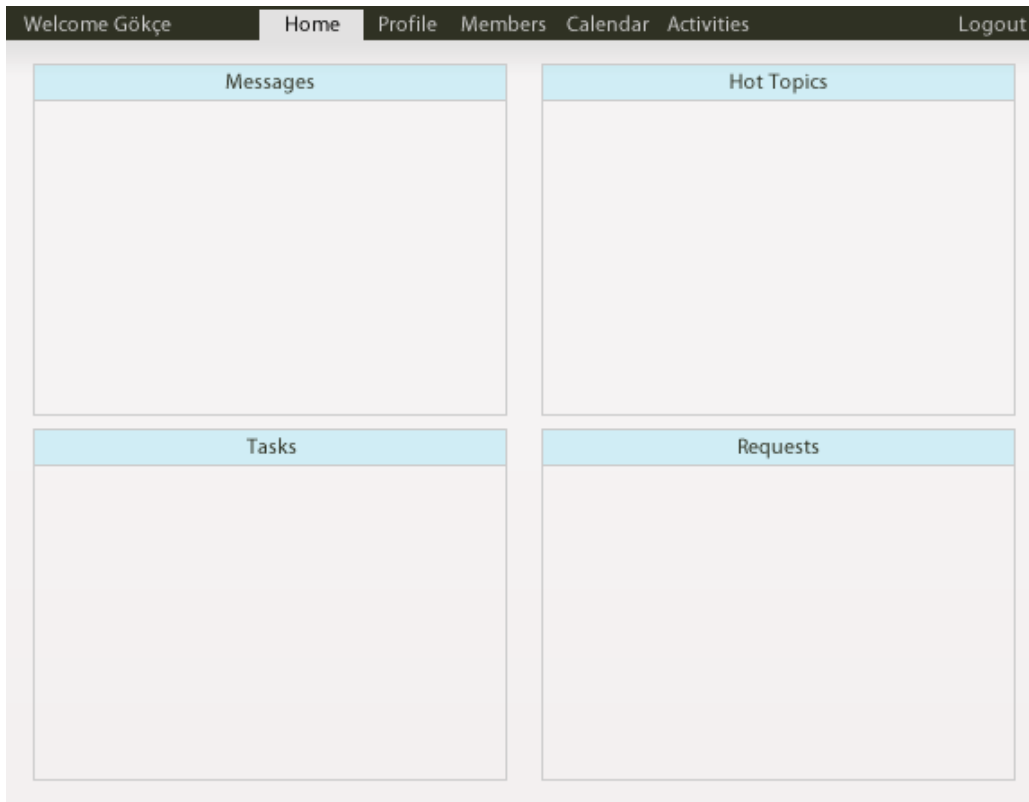Figure 8 - System Administrator Page

Figure 9 - User Page

Figure 10 - Organization Administrator Page

6.3. Screen Objects and Actions

All screen objects and actions are show in the table below:

Login Page	System Administrator Page	System Administrator logs in the system
Login Page	User Page	User logs in the system
Login Page	Organization Page	Organization Administrator log in the system
System Administrator Page	User List Page	System Administrator views user list
System Administrator Page	Organization List Page	System Administrator views organization list
System Administrator Page	System Administrator Page	System Administrator Deactivates the system
System Administrator Page	System Administrator Page	System Administrator Reactivates the system
User Page	Edit Profile Page	User edits his/her profile
User Page	Organization List Page	User views organization list
User Page	Calendar Page	User views calendar page
User Page	Activities Page	User views activities page
Organization Page	Edit Profile Page	Organization Administrator edits his/her profile
Organization Page	Member List Page	Organization Administrator views member list
Organization Page	Activities Page	Organization Administrator views activities page
Organization Page	Calendar Page	Organization Administrator views calendar page
User List Page	User Edit Page	System Administrator edits a user
User List Page	User List Page	System Administrator deletes a user
Organization List Page	Organization Edit Page	System Administrator edits an organization
Organization List Page	Organization List Page	System Administrator deletes an organization
Edit Profile Page	User Page	User saves his/her profile
Edit Profile Page	User Page	User cancels editing his/her profile
Organization List Page	Organization List Page	User applies to an organization
Activities Page	Tasks Page	User views tasks
Activities Page	Events Page	User views events
Edit Profile Page	Organization Page	Organization Administrator saves organization profile
Edit Profile Page	Organization Page	Organization Administrator cancels editing organization profile
Edit Profile Page	User Page	Organization Administrator deletes the organization
Member List Page	Role Assignment Page	Organization Administrator assigns a role to a member
Activities Page	Tasks Page	Organization Administrator views tasks
Activities Page	Events Page	Organization Administrator views events
User Edit Page	User List Page	System Administrator saves user
User Edit Page	User List Page	System Administrator cancels editing user
Organization Edit Page	Organization List Page	System Administrator saves organization
Organization Edit Page	Organization List Page	System Administrator cancels editing organization
Tasks Page	Create Task Page	User creates task
Tasks Page	Edit Task Page	User edits task
Tasks Page	Tasks Page	User deletes task
Events Page	Create Event Page	User creates event
Events Page	Edit Event Page	User edits event
Events Page	Events Page	User deletes event
Tasks Page	Create Task Page	Organization Administrator creates task
Tasks Page	Edit Task Page	Organization Administrator edits task
Tasks Page	Tasks Page	Organization Administrator deletes task
Events Page	Create Event Page	Organization Administrator creates event
Events Page	Edit Event Page	Organization Administrator edits event
Events Page	Events Page	Organization Administrator deletes event
Create Task Page	Tasks Page	User saves task
Create Task Page	Tasks Page	User cancels creating task
Edit Task Page	Tasks Page	User saves task
Edit Task Page	Tasks Page	User cancels editing task
Create Event Page	Events Page	User saves event
Create Event Page	Events Page	User cancels creating event

Edit Event Page	Events Page	User saves event
Edit Event Page	Events Page	User cancels editing event
Create Task Page	Tasks Page	Organization Administrator saves task
Create Task Page	Tasks Page	Organization Administrator cancels creating task
Create Task Page	Tasks Page	Organization Administrator assigns task to a member
Edit Task Page	Tasks Page	Organization Administrator saves task
Edit Task Page	Tasks Page	Organization Administrator cancels editing task
Create Event Page	Events Page	Organization Administrator saves event
Create Event Page	Events Page	Organization Administrator cancels creating event
Create Event Page	Events Page	Organization Administrator invites users to event
Edit Event Page	Events Page	Organization Administrator saves event
Edit Event Page	Events Page	Organization Administrator cancels editing event

<div align="center">**Table 12 - Screen Objects and Actions**</div>

6.4.   State Transition Diagrams

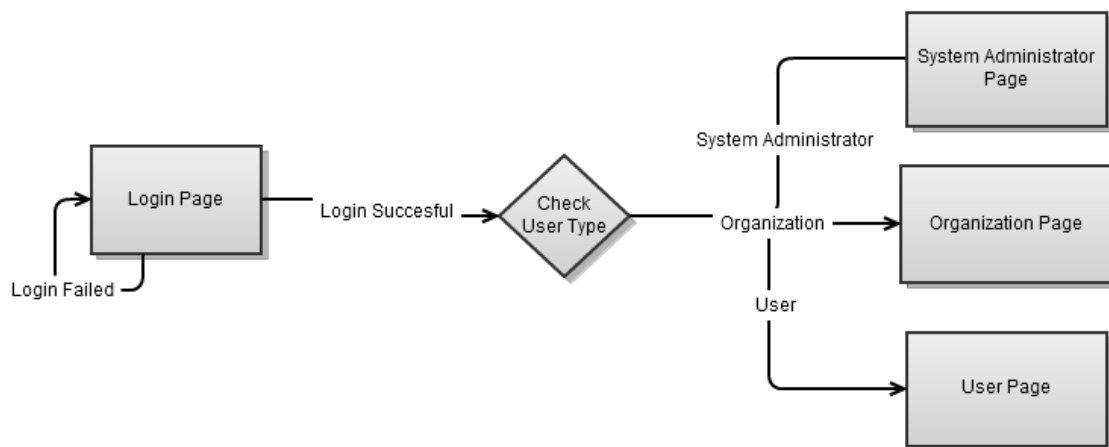<div align="center">**Figure 11 - State Transition Diagram for Login Process**</div>

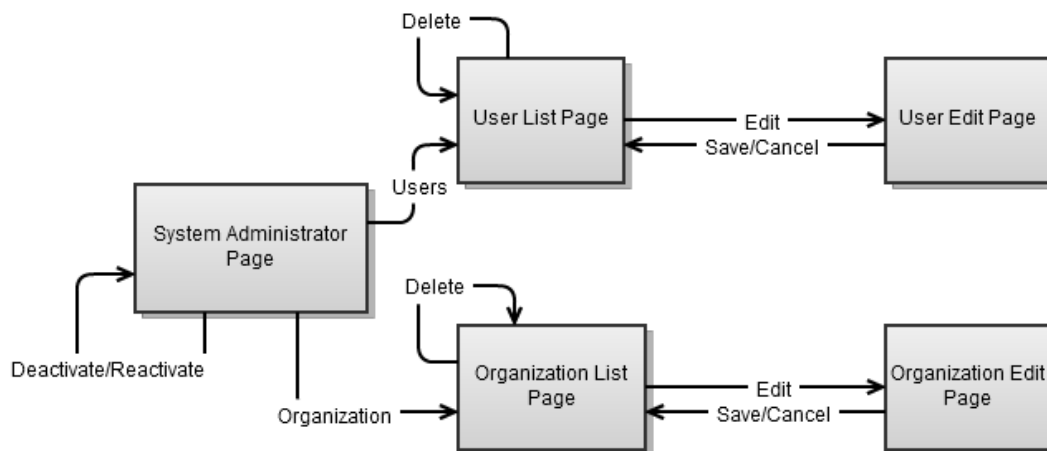<div align="center">**Figure 12 - State Transition Diagram for System Administrators**</div>
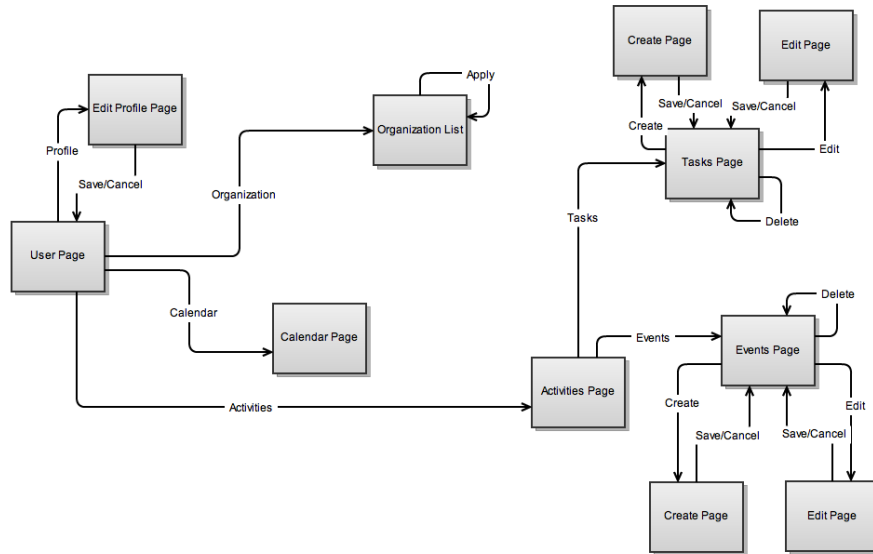
23

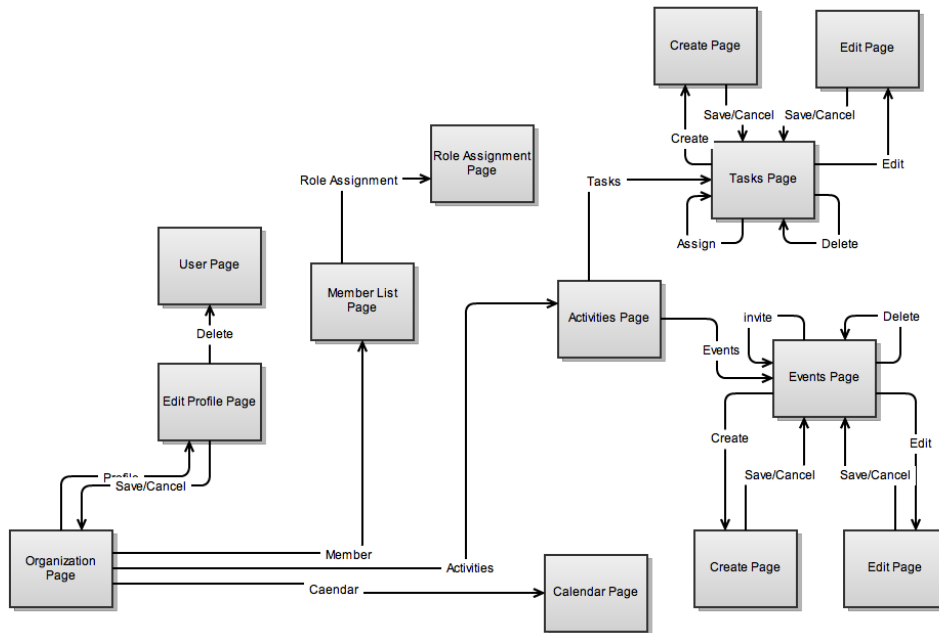Figure 13 - State Transition Diagram for Users

Figure 14 - State Diagram for Organizations

7. Libraries and Tools

Programming languages and frameworks that are needed to create Cloud-SOMS can be listed as follows:

Eclipse IDE: Eclipse is a multi-language software development environment comprising an integrated development environment (IDE) and an extensible plug-in system. It is written mostly in Java and can be used to develop applications in Java and, by means of various plug-ins, other programming languages including Ada, C, C++, COBOL, Perl, PHP, Python, Ruby

(including Ruby on Rails framework), Scala, and Scheme. The IDE is often called Eclipse ADT for Ada, Eclipse CDT for C/C++, Eclipse JDT for Java, and Eclipse PDT for PHP.

Facebook Python/JavaScript API: The application will be integrated into Facebook. In order to achieve this, these two APIs of Facebook will be utilized.

Google App Engine Framework: It is based on two choices: Python and Java. Being a part of Google infrastructure, it is a cloud-based development environment. Considering its performance/price ratio is highly competitive to its rivals such as Amazon E2C.

Google BigTable: It is a Not Only SQL (noSQL) database. It is easy to learn and use; it also provides consistent interfaces to developers. In addition to these, management of queries is quite efficient and this makes the applications highly scalable.

HTML/CSS/JavaScript: The basic ingredients for web applications. Latest improvements on HTML5 and CSS3 will be used.

JQuery: It is a JavaScript library. In its web page it is defined as "a fast and concise JavaScript Library that simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development". Besides its power, having lots of plug-in makes it a desirable choice.

MySQL: MySQL is a relational database management system (RDBMS) that runs as a server providing multi-user access to a number of databases.

Netbeans: NetBeans refers to both a platform framework for Java desktop applications, and an integrated development environment (IDE) for developing with Java, JavaScript, PHP, Python, Ruby, Groovy, C, C++, Scala, Clojure, and others. The NetBeans Platform allows applications to be developed from a set of modular software components called modules.
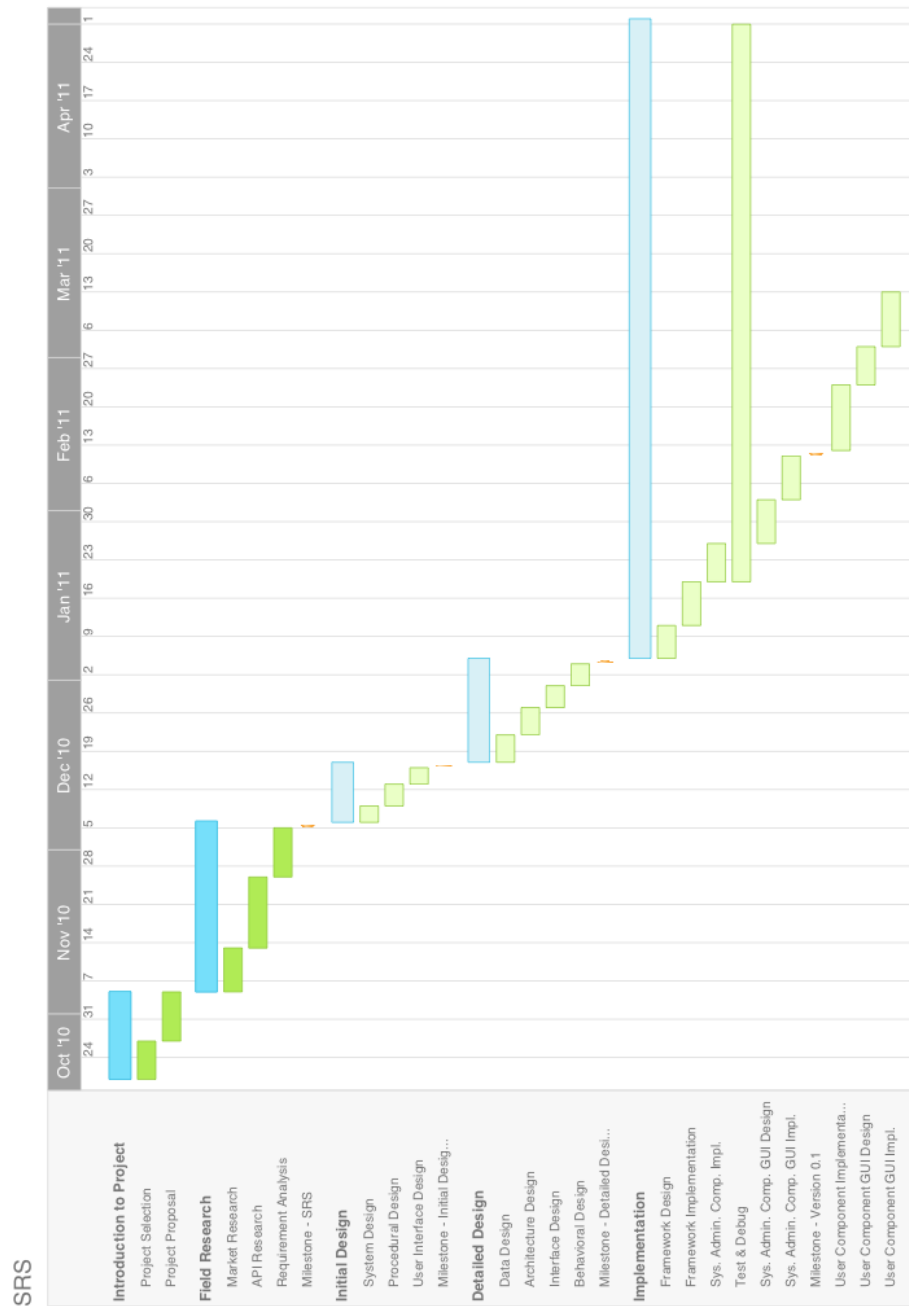
Python: An interpreted scripting programming language. It is appropriate for web application development and there are lots of web frameworks dedicated to it. Its ease of use and not having steep learning curve makes it suitable for this kind of project.

Query Builder and GQL: There are two alternative ways to create queries for BigTable, which are query builder and GQL. They provide easy to express queries similar to natural language and SQL respectively.
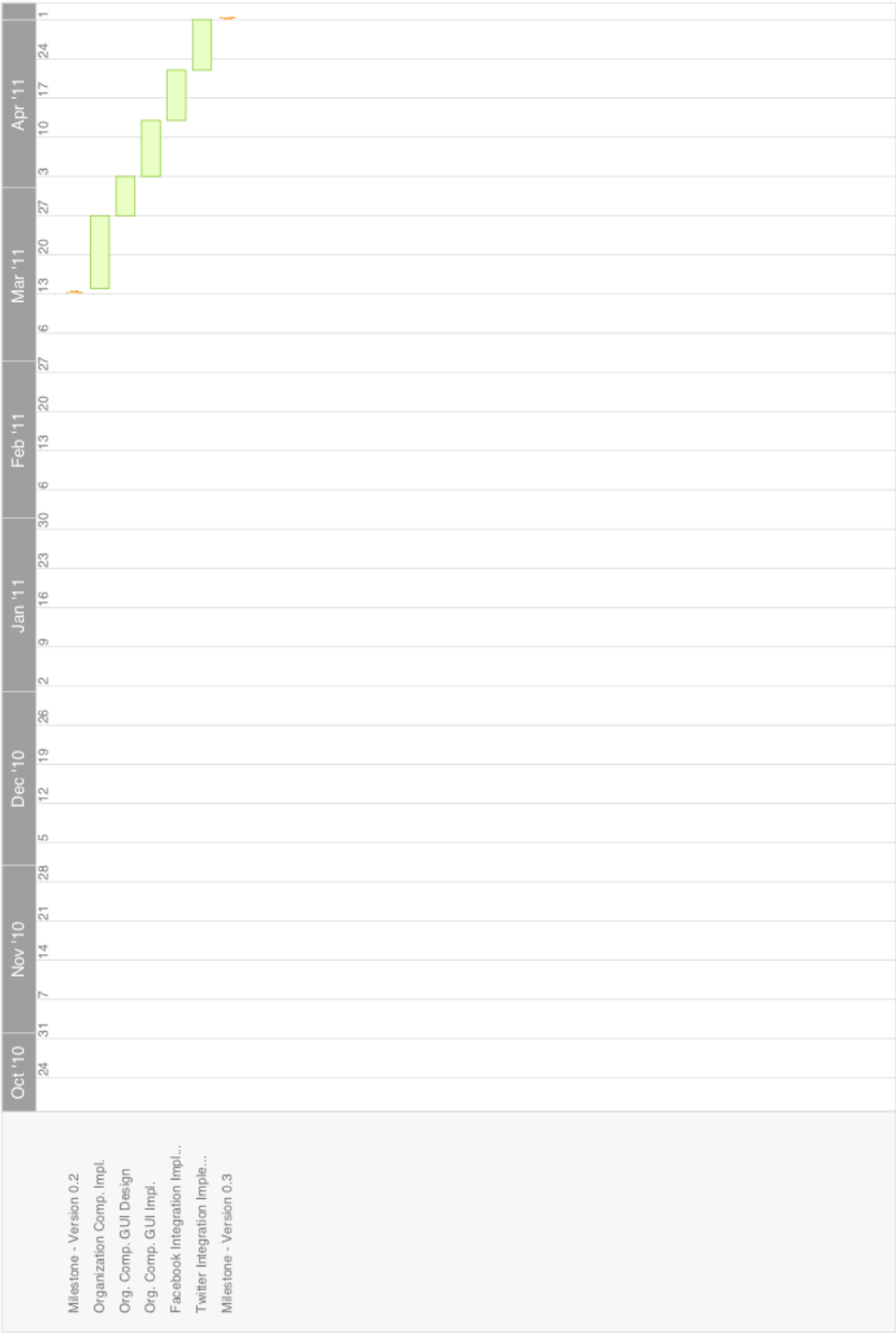
tinyPM: tinyPM is advanced, lightweight and smart tool for agile collaboration including product management, backlog, taskboard, user stories, wiki, integrations and REST API. tinyPM goes beyond software development and encourages all team members (including clients, management and stakeholders) to actively participate in all your projects.

Tomcat: Apache Tomcat is an open source software implementation of the Java Servlet and JavaServer Pages technologies. The Java Servlet and JavaServer Pages specifications are developed under the Java Community Process.

8. Cloud-SOMS Team Gantt Chart

SRS

Milestone - Version 0.2
Organization Comp. Impl.
Org. Comp. GUI Design
Org. Comp. GUI Impl.
Facebook Integration Impl…
Twitter Integration Imple…
Milestone - Version 0.3

Oct '10 Nov '10 Dec '10 Jan '11 Feb '11 Mar '11 Apr '11

9. Conclusion

This document states the design level approach adopted by Cloud-SOMS team for the Cloud-SOMS project. In this document, a fair amount of elaboration has been done on the project scenario pointing out most of the important details. The goals for the final product has become more apparent as the scenario and the desired user interface are visually explained. Additionally, this document is the first document that explains somewhat deep technical details. The architecture of the system is discussed with an overview. Further information on the technical design is given with detailed explanations of the modules, which are supported with class and sequence diagrams. Finally, the progress made by the project team is summarized.