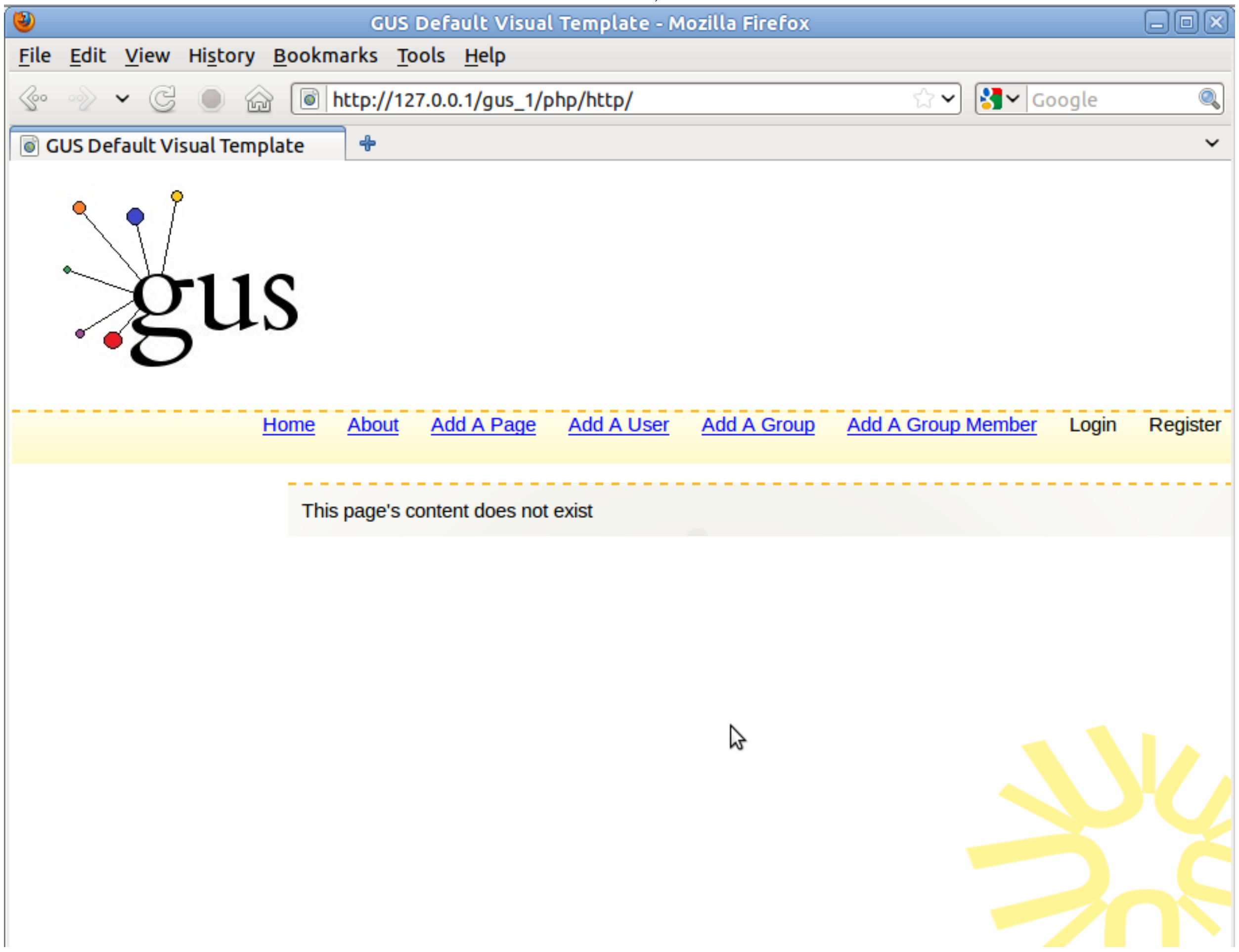
SYSTEM AND SOFTWARE DESIGN DESCRIPTION (SSDD): Incorporating Architectural Views and Detailed Design Criteria
FOR
Groups in a University Setting
Version 0.004
December 2, 2010



This page's content does not exist

Prepared for:
The Associated Students of the University of Idaho and Dr. Clinton Jeffery
Prepared by:
The PHP Gus Development Group
University of Idaho
Moscow, ID 83844-1010
CS383 SSDD

Contents

1 Introduction

1.1 Identification

The software under development is referred to as Groups in a University Setting (GUS). The customers providing specifications for the PHP team are the student groups at the University of Idaho. The end users will be of GUS will be the students and student groups at the University of Idaho. This is a new product effort, so the version under development is version 0.004.

1.2 Document Purpose, Scope, and Intended Audience

1.2.1 Document Purpose

The purpose of this document is to describe the design of GUS. This document will serve as a guide for the PHP team as they move forward in development. After the product is completed, this document will serve as a reference in maintaining and updating the product.

1.2.2 Document Scope

This document aims to contain the documentation needed to develop GUS. It will include UML diagrams, use case descriptions and other important details with regards to the design of the product.

1.2.3 Intended Audience for Document

This document is intended to be read by the development team, the team's sponsor Dr. Jeffery, and any personnel responisble for performing maintenance on or updating GUS. While the system will be used by university personnel, this document is intended to be read and understood by UICS software designers and coders. This document will also be approved by Dr. Clinton Jeffery.

1.3 System Software Purpose, Scope, and Intended Users

1.3.1 System and Software Purpose

The purpose of the system under development is to provide a utility to groups at the University of Idaho for administration and communication purposes. GUS will allow students to connect with groups that will meet their involement needs. The system will be used by clubs, sports teams, and other university groups to increase student involvement by connecting and recognizing the involvement of members.

1.3.2 System and Software Scope/or Context

The finished product will run on University of Idaho servers and will be accessible from any computer with a web browser and Internet access. A user will be able to log in and receive updates and communications from their groups anytime it is convenient for the user.

1.3.3 Intended Users for the System and Software

The intended users of the system are students and student groups at the University of Idaho.

1.4 Definitions, Acronyms, and Abbreviations

Term or Acronym	**Definition**
Acquirer	The person, team, or organization that pursues a system or software product or service from a supplier. The acquirer may be a buyer, customer, owner, purchaser, or user. ISO/IEC 42010:2007 (§3.1).
AD	Architectural Description: A collection of products to document an architecture ISO/IEC 42010:2007 (§3.4).
Alpha test	Limited release(s) to selected, outside testers
Architect	The person, team, or organization responsible for systems architecture ISO/IEC 42010:2007 (§3.2).
Architectural Description	(AD) A collection of products to document an architecture ISO/IEC 42010:2007 (§3.4).
Architectural View	A representation of a whole system from the perspective of a related set of concerns ISO/IEC 42010:2007 (§3.9).
Architecture	The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution ISO/IEC 42010:2007 (§3.5).
Beta test	Limited release(s) to cooperating customers wanting early access to developing systems
Design Entity	An element (component) of a design that is structurally and functionally distinct from other elements and that is separately named and referenced IEEE STD 1016-1998 (§3.1).
Design View	A subset of design entity attribute information that is specifically suited to the needs of a software project activity IEEE STD 1016-1998 (§3.2).
Final test	aka, Acceptance test, release of full functionality to customer for approval
DFD	Data Flow Diagram
SDD	Software Design Document, aka SDS, Software Design Specification
Software Design Description	A representation of a software system created to facilitate analysis, planning, implementation, and decision making, A blueprint or model of a software system. The SDD is used as the primary medium for communicating software design information IEEE STD 1016-1998 (§3.4).
SRS	Software Requirements Specification
SSDD	System and Software Design Document
SSRS	System and Software Requirements Specification
System	A collection of components organized to accomplish a specific function or set of functions ISO/IEC 42010:2007 (§3.7).
System and Software Architecture and Design Description	An architectural and detailed design description that includes a software system within the context of its enclosing system and describes the enclosing system, the enclosed software, and their relationship and interfaces.
System Stakeholder	An individual, team, or organization (or classes thereof) with interests in, or concerns, relative to, a system ISO/IEC 42010:2007 (§3.8).

Term or Acronym	Definition

1.5 DOCUMENT REFERENCES

1) CSDS,*System and Software Requirements Specification Template* , Version 1.0, July 31, 2008, Center for Secure and Dependable Systems, University of Idaho, Moscow, ID, 83844.

2) ISO/IEC/IEEE,*IEEE Std 1471-2000 Systems and software engineering – Recommended practice for architectural description of software intensive systems,*First edition 2007-07-15, International Organization for Standardization and International Electro-technical Commission, (ISO/IEC), Case postale 56, CH-1211 Genève 20, Switzerland, and The Institute of Electrical and Electronics Engineers, Inc., (IEEE), 445 Hoes Lane, Piscataway, NJ 08854, USA.

3) IEEE, *IEEE Std 1016-1998 Recommended Practice for Software Design Descriptions*, 1998-09-23, The Institute of Electrical and Electronics Engineers, Inc., (IEEE) 445 Hoes Lane, Piscataway, NJ 08854, USA.

4) ISO/IEC/IEEE,*IEEE Std. 15288-2008 Systems and Software Engineering – System life cycle processes,* Second edition 2008-02-01, International Organization for Standardization and International Electro-technical Commission, (ISO/IEC), Case postale 56, CH-1211 Genève 20, Switzerland, and The Institute of Electrical and Electronics Engineers, Inc., (IEEE), 445 Hoes Lane, Piscataway, NJ 08854, USA.

5) ISO/IEC/IEEE, IEEE Std. 12207-2008,*Systems and software engineering – Software life cycle processes,* Second edition 2008-02-01, International Organization for Standardization and International Electro-technical Commission, (ISO/IEC), Case postale 56, CH-1211 Genève 20, Switzerland, and The Institute of Electrical and Electronics Engineers, Inc., (IEEE), 445 Hoes Lane, Piscataway, NJ 08854, USA.

1.6 Document Overview

1.6.1 Section 2

This section of the document describes the system and software constraints imposed by the operational environment, system requirements and user characteristics, and then identifies the system stakeholders and lists and describes their concerns and mitigations to those concerns.

1.6.2 Section 3

This section of the document describes the system and software architecture from several viewpoints, including, but not limited to,the developer's view and the user's view.

1.6.3 Section 4

This section of the document provides detailed design descriptions for every component defined in the architectural view(s).

1.6.4 Sections 5

This section of the document provides traceability information connecting the original specifications (referenced above) to the architectural components and design entities identified in this document.

1.6.5 Section 6

Section 6 and beyond are appendices including original information and communications used to create this document.

1.7 Document Restrictions

This document is for LIMITED RELEASE ONLY to University of Idaho CS personnel working on the project and Dr. Jeffery.

2 Constraints and Stakeholder Concerns

2.1 Constraints

2.1.1 Environmental Constraints

GUS must be maintained in a secure office to prevent unauthorized physical access to the server. The office should be located in an area that is unlikely to suffer from natural disasters, such as flooding, earthquakes, etc. The system should also be able to withstand frequent power outages in case of wind storms.

2.1.2 System Requirement Constraints.

Since users will access GUS through the Internet, GUS needs to be able to maintain a constant connection to the Internet. GUS will also need to be able to interface with any data source it needs information from, such as databases, web authentication servers, and users. It will also need to keep this information secure and confidential when appropriate.

2.1.3 User Characteristic Constraints.

GUS will be easy for any user to understand after a brief explanation. Even without an explanation, users will be able to determine the usage and functionality of the system by looking through the options. Basic computer skills and a simple conceptual explananation should be enough for everyday usage.

2.2 Stakeholder Concerns

Stakeholder x Concern x Mitigation Table			
Stakeholder Concern	**List of Stakeholders**	**Stated Concern**	**Mitigation Mechanism or Design Criteria Reference Number**
Appropriateness of the system and	Acquirer	Will this project develop the software engineering skills required to work in the industry?	Limit the size of the teams, require developers to use server-side scripting and extensive development documentation
software in fulfilling its mission	User	Will this system be user friendly enough that a computer illiterate person can use it?	Provide visual and textual cues to make it easy for non-technical users to understand
Feasibility of constructing, testing,	Developer	Will this project be interesting and completable within one semester?	Encourage members to work on the parts of the project they are interested in, and share the workload.
verifying and deploying the system and software.	Tester	Will the testing utilities be user-friendly enough to learn without spending too much time?	Collectively identify appropriate testing utilities for each platform the testers will be using, and identify helpful tutorials
Risks of constructing, deploying, and	Acquirer	Will this system make the University of Idaho legally liable if the system is hacked?	Educate users of the risks of posting private information on the Internet, and limit the information GUS tracks
using the system and software object of this SSDD.	User	Will my information be kept private from stalkers, telemarketers, and other creeps?	Use security measures, such as keeping GUS physically secure, encrypting information, and using modular **mutually suspicious programming**
Concerns with respect to the maintainability and	Acquirer	Will there be a maintainer?	Include Dr. Jefferey in every step of the design process, and try to fill an important niche for the University of Idaho
evolvability of the system and software.	Maintainer	Will GUS be easy to port to another system that would maintain it?	Keep all aspects of GUS extremely simple and practice good coding to enable portability

3 System and Software Architecture

3.1 Developer's Architectural View

The basic architecture of GUS is a database driven, client-server web application. The components of GUS are: graphical user interfaces (websites), web-interfaces to a database, and a database. The graphical user interfaces are the only part the user directly sees and manipulates. They allow users to input and view information in the database and they communicate with the web-interfaces to access the database and present the data in a human-readable form. The web interfaces ensure that the input from the users is valid and safe to put in the database, and they also ensure that data pulled from the database is sent only to the appropriate users.

3.2 User's Architectural View

GUS is a website with different levels of privelege for users, officers, and administrators. When logged into GUS, a user will have member-level access to groups that they are a member of, officer-level access for groups they are an officer in, and administrative privileges for groups they own.

3.2.1 User's View Identification

From the user's point of view, GUS is a tool that enables users to connect with one another through groups. As such, communication between users is a very important component and is fulfilled by several different modules including e-mail and forums. One of the major distinctions between users in a group is their level of privilege, and there are specific modules for each of the three levels of privelege.

3.2.2 User's View Representation and Description

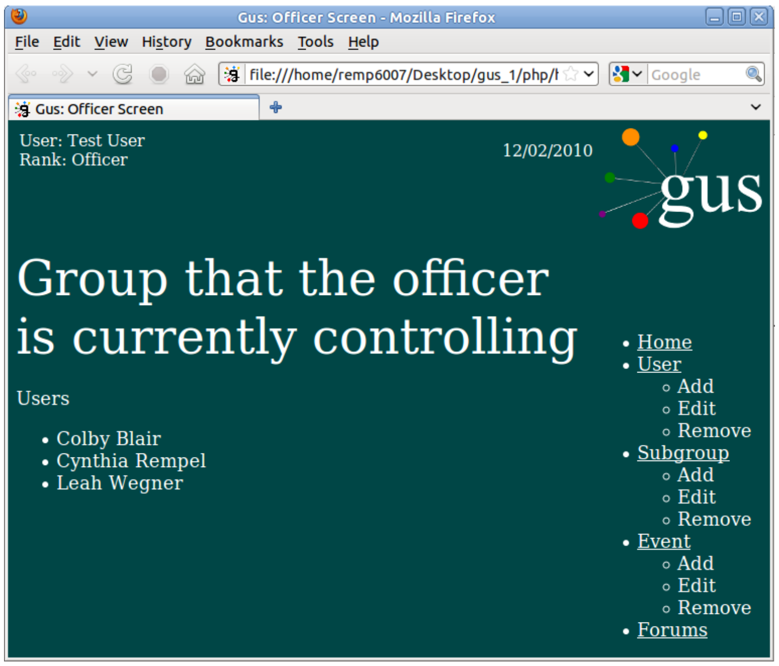
3.2.2.1 Group Administrator Use Cases

3.2.2.2 Add Officer

3.2.2.3 Modify Officer Access

3.2.2.4 Remove Officer

3.2.2.5 Officer Use Cases The purpose of the officer module is to enable officers to manage the group events, membership, and website. Officers can: add, modify, or remove events; add, modify, or remove group websites, and add or remove members. Because officers can just be a member of a different group, the officer's available actions would be listed in a pop-out menu connected to the group's site. To improve useability, the pop-out menu should look like an extention of the user menu. For example: the officer navigates to the group website, his user menu becomes longer, he hovers over "events", the pop-out would display "add event, modify event, and remove event", the officer would then select an option, such as, "remove event" and the remove event


Gus: Officer Screen - Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///home/remp6007/Desktop/gus_1/php/l
Google
Gus: Officer Screen
gus
Home
User
Add
Edit
Remove
Subgroup
Add
Edit
Remove
Event
Add
Edit
Remove
Forums

User: Test User
Rank: Officer

12/02/2010

Group that the officer is currently controlling

Users

- Colby Blair
- Cynthia Rempel
- Leah Wegner

Figure 1: Officer Panel Mockup by Leah Wegner

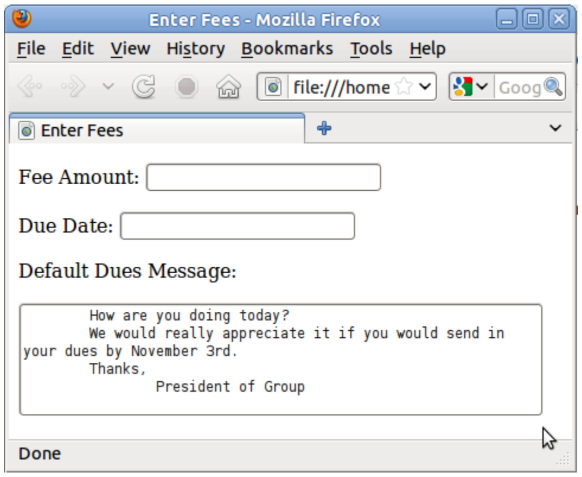
page would be displayed. For example, the president of the UI Tech club could post a plaque engraving event for 2:30 PM, and after 10 people signed up, there could be a snowstorm (2 ft in one hour!) at 12:30 he cancels the event, so he decides to send a group text (which includes not only UI Tech club members likely to attend but didn't sign up, but everyone who RSVPed to include John (who's just taking the class and is not in club), followed up by an email, and finally the president prints out a roster of the likely attenders who only have a land-line with no internet to call manually. John later decides to join the Tech club, and the treasurer uses GUS to add him to the Tech club.

An officer may want to change the contents of a group's web page. For example: this year Idaho's Business Professionals of America State Leadership Conference is holding a C++ event, and BPA may want to add that back to their group page.

3.2.2.6 Add Activity

3.2.2.7 Add User

3.2.2.8 Add Website


Enter Fees - Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///home
Enter Fees
Done

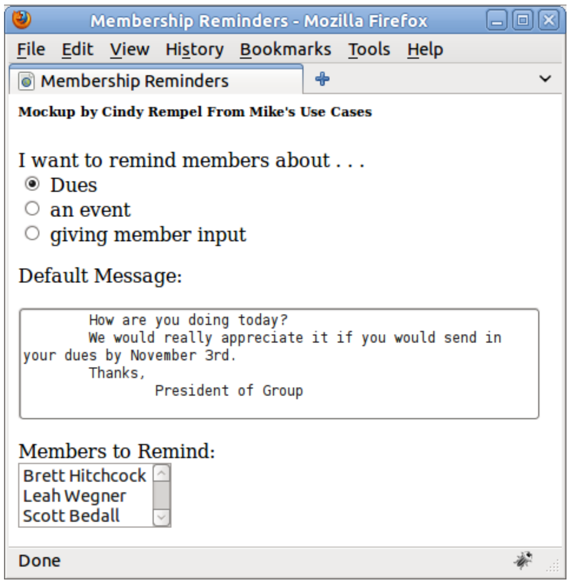
Fee Amount:

Due Date:

Default Dues Message:

```
        How are you doing today?
        We would really appreciate it if you would send in
your dues by November 3rd.
        Thanks,
                President of Group
```

Figure 2: Enter Fees Mockup by Cindy


Membership Reminders - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Membership Reminders
Done

Mockup by Cindy Rempel From Mike's Use Cases

I want to remind members about . . .
- (●) Dues
- ( ) an event
- ( ) giving member input

Default Message:

```
        How are you doing today?
        We would really appreciate it if you would send in
your dues by November 3rd.
        Thanks,
                President of Group
```

Members to Remind:
- Brett Hitchcock
- Leah Wegner
- Scott Bedall

Figure 3: Membership Reminders Mockup by Cindy

3.2.2.9 Enter Fee Information

Actor: officer

Goal: Store group dues in GUS database

Precondtitions: Officer is logged on and at the group's officer page

Related use cases: Remind users to pay dues, enter membership criteria

Steps:

1. Click 'enter fee information'
2. Type in fee amount
3. Select due date from a calendar
4. Click 'save'

Alternatives:

Postconditions: the dues information is saved in the database

This use-case can be expanded by using the same screen for an officer to specify all the membership criteria: waivers, dues, service project, class standing, GPA, major, etc.

3.2.2.10 Modify Activity

3.2.2.11 Modify Website

3.2.2.12 Use case: notify members concerning dues/fees from Mike's use cases.

Actor: group leader

Goal: notify members of outstanding expenses

Precondtitions: expense information is already entered into GUS

Related use cases: enter fee information, send message, notify members to fill out waivers

Steps:

1. Click 'money'
2. Click 'send notifications'
3. Write a reminder message or use the default dues reminder message
4. Choose the users who need to be reminded (perhaps GUS could be used to store who has paid dues and who hasn't)

Alternatives: the officer cancells out of the usecase by clicking elsewhere, perhaps this use-case could be expanded to allow the officer to remind members about multiple membership criteria they need to meet, but personalized so members are only reminded of what they are deficient in.

Postconditions: a notification is emailed to each applicable member

3.2.2.13 Remove Activity

3.2.2.14 Remove User

3.2.2.15 Remove Website

3.2.2.16 Text and E-mail Module

User-view class diagram which this entry is based on was designed by Brett Hitchcock, this entry was written by Cynthia Rempel.

The purpose of the e-mail module is to enable users to send e-mails to groups and other users. Two advantages of using GUS to communcate are: texting and predefined social networks. The texting feature enables users or groups to communicate about events at the last minute via SMS text messages. Instead of each user manually updating their e-mail lists when people leave the group, GUS tracks the members for each group. For example: the president of the Business Professionals of America might want to invite the Information Systems club to a meeting about their information systems competitive events. Another example: the president of the Tech club cancels a meeting at the last minute because the advisor is ill, and he texts all of the Tech club members. A user can access their e-mail and text features through their e-mail page, and they can only text people (or groups) that have permission to e-mail or text.

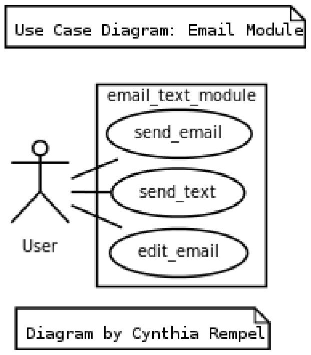


Figure 4: Replace with screenshot of email mock-up

3.2.2.17 Send Group Email

3.2.2.18 Send Text

3.2.2.19 User Use Cases

3.2.2.20 Create Email

3.2.2.21 Edit Email

3.2.2.22 Edit Personal Page

The user-view class diagram which this entry is based on was designed by Brett Hitchcock, this entry was written by Cynthia Rempel.

The purpose of a profile is to enable the user or group to present a picture of who she or they are, so other users or groups can make an informed decision of whether to associate with her or them.

For example: the Business Professionals of America (BPA) club might appeal to the Virtual Technology and Design (VTD) majors, because BPA displays their video animation competive event for which they want to recuit judges at the local high-school level and participants at the college-level. Also, the Lutheran Campus Ministry might want to attract music majors, because of their proximity to the Music building. Another example of a profile feature might be an incoming freshmen might be really interested in video games, and the video game club is looking for members.

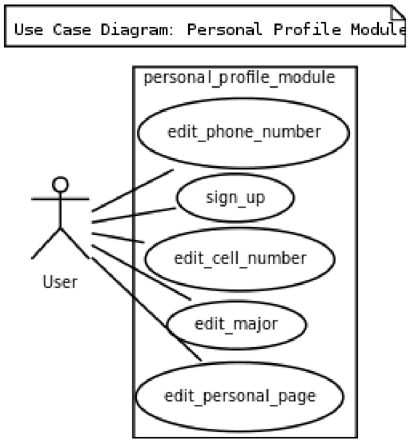


Figure 5: replace with screenshot of profile mock-up

Users may want the amount of information shared limited to only a subset of groups they are members of. For example: Joan2833 may want to share her schedule with members of the National Youth Accreditation Association (NYACC), but not with ASUI.

The personal profile is also a way for users to keep their information up-to-date. For example: if the head coach needs to contact the Tech club president for last minute changes for the awards to be engraved (the underdog pulled through at the last minute) and so his name is to be engraved now. Another example would be, every fall the ASUI emails all the student groups with the times for mandatory officer orientation, and up-to-date emails facilitate this process.

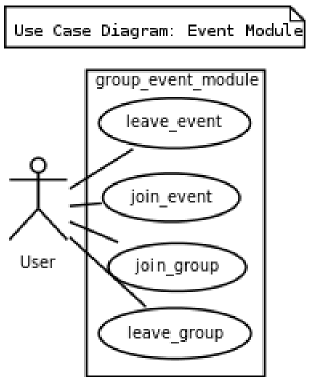


Figure 6: Replace with mock-up of a page inviting the user to join a group and/or an event

3.2.2.23 Event Group Module

The user-view class diagram which this entry is based on was designed by Brett Hitchcock, this entry was written by Cynthia Rempel.

The purpose of the event group module is to enable users to join groups and events. Users can use GUS to look for groups that interest them, or they might want to join events that interest them. For example: a softmore interested in becoming a Methodist minister may decide joining Religion and Ethics may look really good on an application to go to seminary. A marketing major may want to sign up to judge at a DECA event that Business Professionals of America is hosting to encourage high-schoolers to learn business skills.

When a user makes a request to join a group or event, their request is checked against the database as to whether they meet the criteria to perform the add group or event action. If the user is qualified, then the change is made in the database, and a confirmation is displayed on the screen and sent to the user's email, otherwise, the user is notified they do not meet the criteria.

Some groups and events are open to any GUS user, but other groups and events may not be open to the public, so there are three levels of approval to be built in: open, auto-approved if the criteria is met, and approved by officers only.

3.2.2.24 Join Activity

3.2.2.25 Join Group

3.2.2.26 Leave Group

3.2.2.27 Send Email

3.2.2.28 Sign Up

3.2.2.29 Use case: register for gus

Actor: prospective user

Goal: user will be registered for GUS

Preconditions: user is at GUS registration page

Related Use Cases: edit personal information

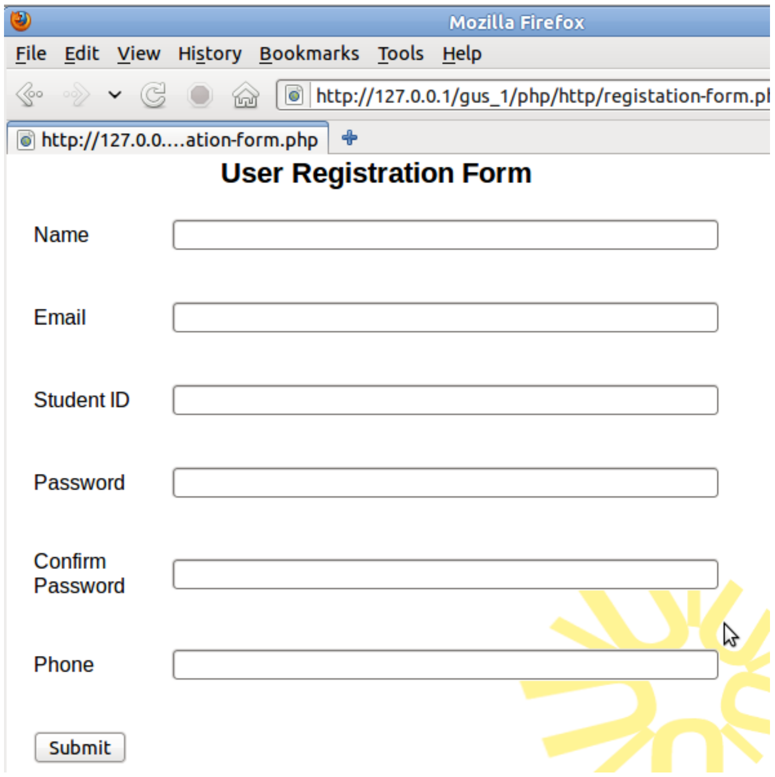
Steps:

1. User enters their name
2. User enters their email
3. User enters their user name
4. User enters their password
5. User enters their password again
6. User enters their phone number
7. User clicks submit

Alternatives: user navigates away from page, officer or administrator may desire to add a user who is less technically literate

Post Conditions: The user's name, email, user name, password, and phone number are validated, and then sent to GUS to be put in the database

Once the user submits the registration form, a javascript is executed, which then checks the user's input against regular expressions. Finally, once the form is validated it is checked again by GUS for valid information, then finally the information is sent to the database, then a series of confirmations are sent, and the user is then given a confirmation message that displays the information added to the database.


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://127.0.0.1/gus_1/php/http/registation-form.pl
http://127.0.0.....ation-form.php

User Registration Form

Name	
Email	
Student ID	
Password	
Confirm Password	
Phone	

Figure 7: registration form validated with regular expressions by Abhay Patil

3.2.2.30 Forums Use Cases

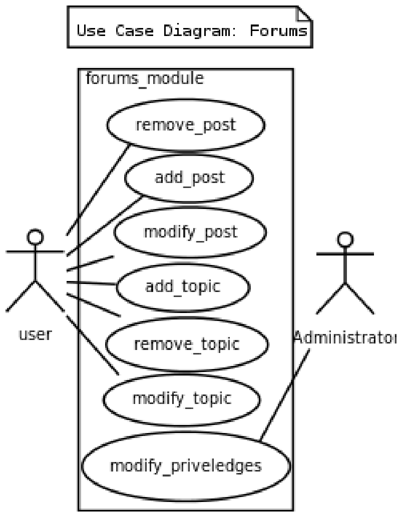


Figure 8: replace with screenshots of forums mock-ups

3.2.2.31 Forums Module — diagram which this entry is based on was designed by John Sandmeyer, this entry was written by Cynthia Rempel.

The forums module is designed to enable groups to collaborate on group decisions and consolidate member input. For example: the president of the tech-club may start a forum on brainstorming and developing informal business plans to raise money. From there, the vice president might post a topic of using the club to start computer repair shop, the secretary may line out possible services, the treasurer may suggest a fee schedule, and members may volunteer what services, and how much time they could commit to the idea.

To post on the forums page a user would have to be granted permissions by the groups administrator, and the user would have to be on the groups forum page. Types of permissions a user would expect to see are: add post, modify own post, modify other members posts, remove own post, and remove other members posts; there would be a similar (but possibly more restrictive) set of permissions for topics.

For the administrator to modify privaledges, she should be on the forums page of the group, ideally, there should be some minimal guidance on the security aspects of giving users permissions right on the forums page, perhaps a **permissions wizard**

3.2.2.32 Add Post

3.2.2.33 Add Topic

3.2.2.34 Edit Post

3.2.2.35 Edit Topic

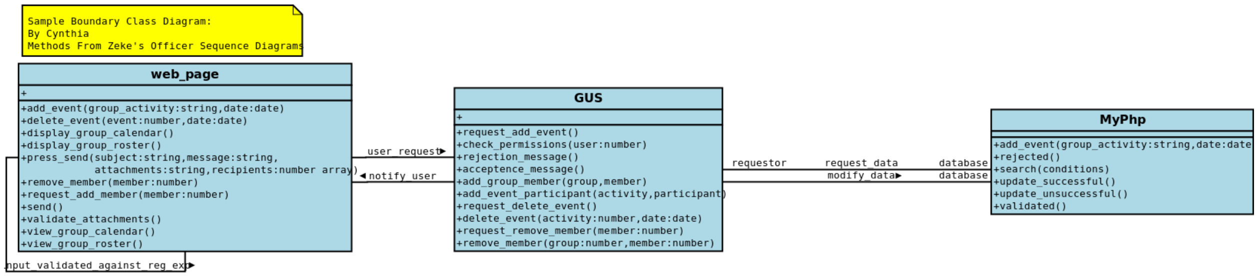
3.2.2.36 Modify Privaledges

3.2.2.37 Remove Post

3.2.2.38 Remove Topic

3.2.3 Developer's View Representation and Description

3.2.3.1 GUS is made of three basic systems, the user interface (forms class), the interface between the user interface and the database (GUS class), and the calls to the database (MySql class). The forms class collects data from users, then hands the data to GUS, which in turn, transmits the data to the database. When a user makes a request for data, GUS queries the database, prepares the information to be displayed, and the forms class displays the information. The forms class can be extended to have additional text fields for user input. The GUS class establishes a connection to the database for the transfer of data, and uses that data to set up page content for the user. The MySql class connects and disconnects from GUS, implements the database design, queries the database, and enables GUS to make changes to the database.



Sample Boundary Class Diagram:
By Cynthia
Methods From Zeke's Officer Sequence Diagrams
web_page
+
+add_event(group_activity:string,date:date)
+delete_event(event:number,date:date)
+display_group_calendar()
+display_group_roster()
+press_send(subject:string,message:string, attachments:string,recipients:number array)
+remove_member(member:number)
+request_add_member(member:number)
+send()
+validate_attachments()
+view_group_calendar()
+view_group_roster()
nput_validated_against_reg_ex
user_request
notify_user
GUS
+
+request_add_event()
+check_permissions(user:number)
+rejection_message()
+acceptence_message()
+add_group_member(group,member)
+add_event_participant(activity,participant)
+request_delete_event()
+delete_event(activity:number,date:date)
+request_remove_member(member:number)
+remove_member(group:number,member:number)
requestor
request_data
modify_data
database
database
MyPhp
+add_event(group_activity:string,date:date
+rejected()
+search(conditions)
+update_successful()
+update_unsuccessful()
+validated()

Figure 9: Sample Boundary Class Diagram by Cynthia

3.2.3.2 Add Officer

3.2.3.3 Modify Officer Access

3.2.3.4 Remove Officer

3.2.3.5 Change Group password – implement access through roles The level of access a person has to the group's website is implemented through the use of roles: when a user navigates to a group's website, the system checks with the database to see if the user is an officer, and if so, displays the additonal officer menu underneath or next to their user menu.

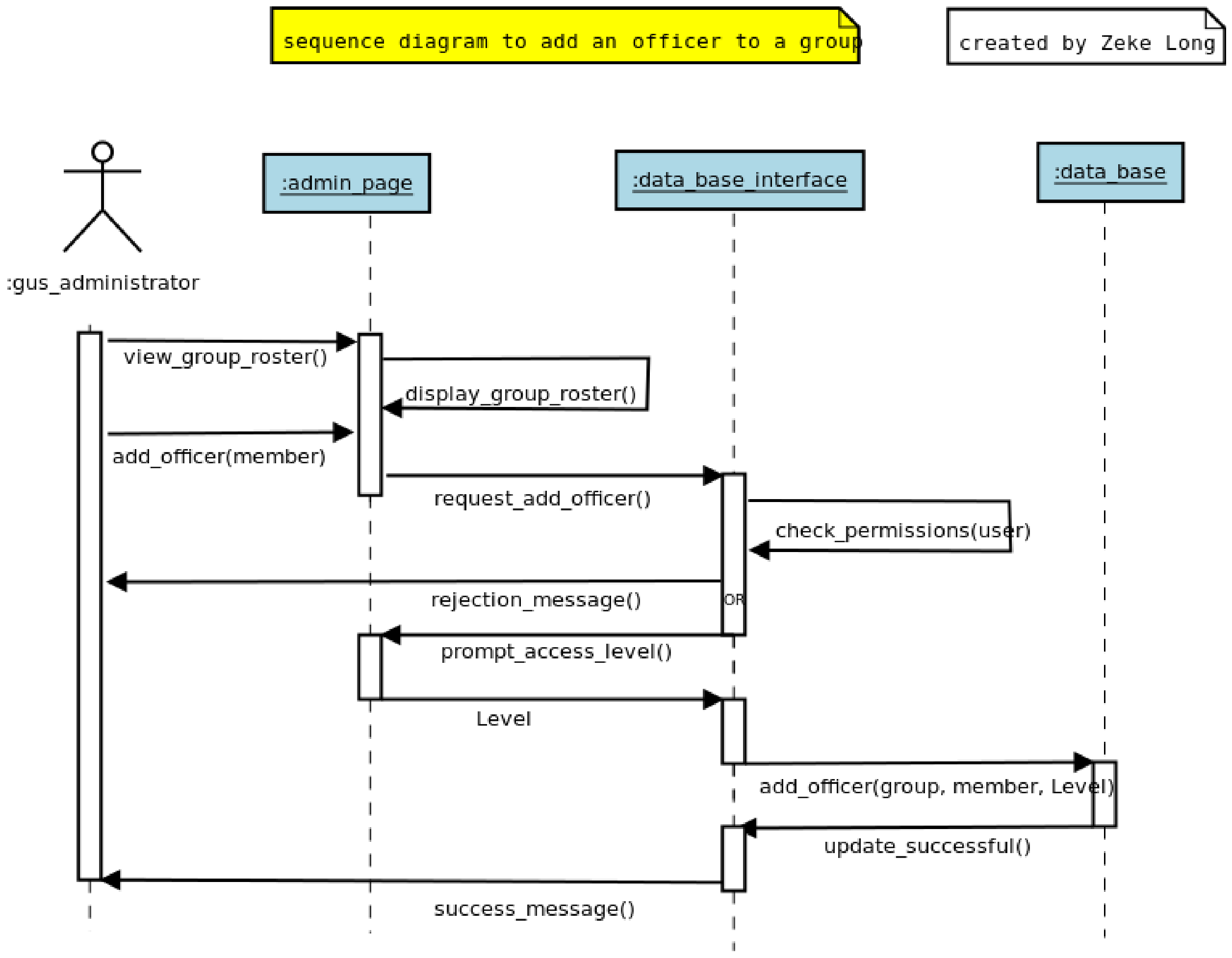


Figure 10: Add Officer Sequence Diagram by Zeke, image linked by Cindy, textual description by ...

3.2.3.6 Add User

3.2.3.7 Add Activity

3.2.3.8 Add Website

3.2.3.9 Modify Activity

3.2.3.10 Modify Website

3.2.3.11 Remove User

3.2.3.12 Remove Website

3.2.3.13 Remove Activity

3.2.3.14 Send Group Email

3.2.3.15 Send Text

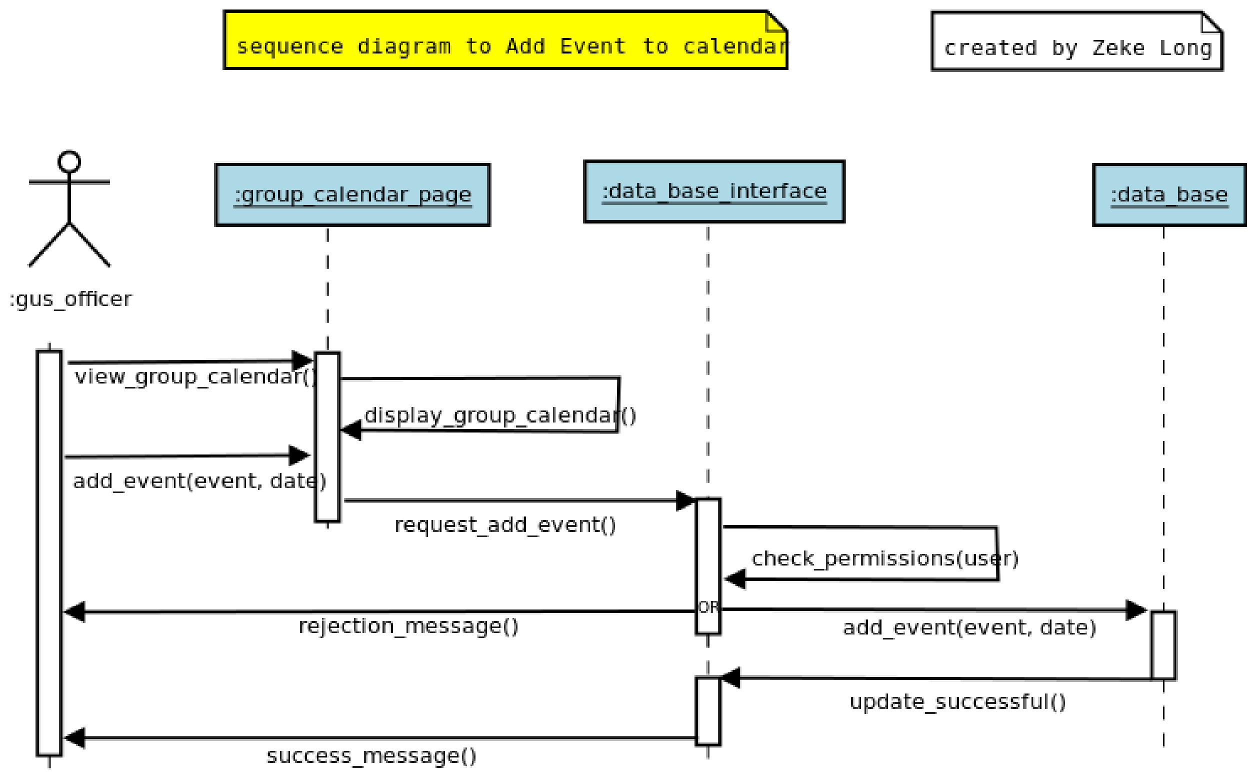


Figure 11: Add Activity Sequence Diagram by Zeke, image linked by Cindy, textual description by ...

3.2.3.16 Create Email

3.2.3.17 Edit Email

3.2.3.18 Edit Personal Page

3.2.3.19 Join Activity

3.2.3.20 Join Group

3.2.3.21 Leave Group

3.2.3.22 Send Email

3.2.3.23 Sign Up

3.2.3.24 Add Post

3.2.3.25 Add Topic

3.2.3.26 Edit Post

3.2.3.27 Edit Topic

3.2.3.28 Remove Post

3.2.3.29 Remove Topic

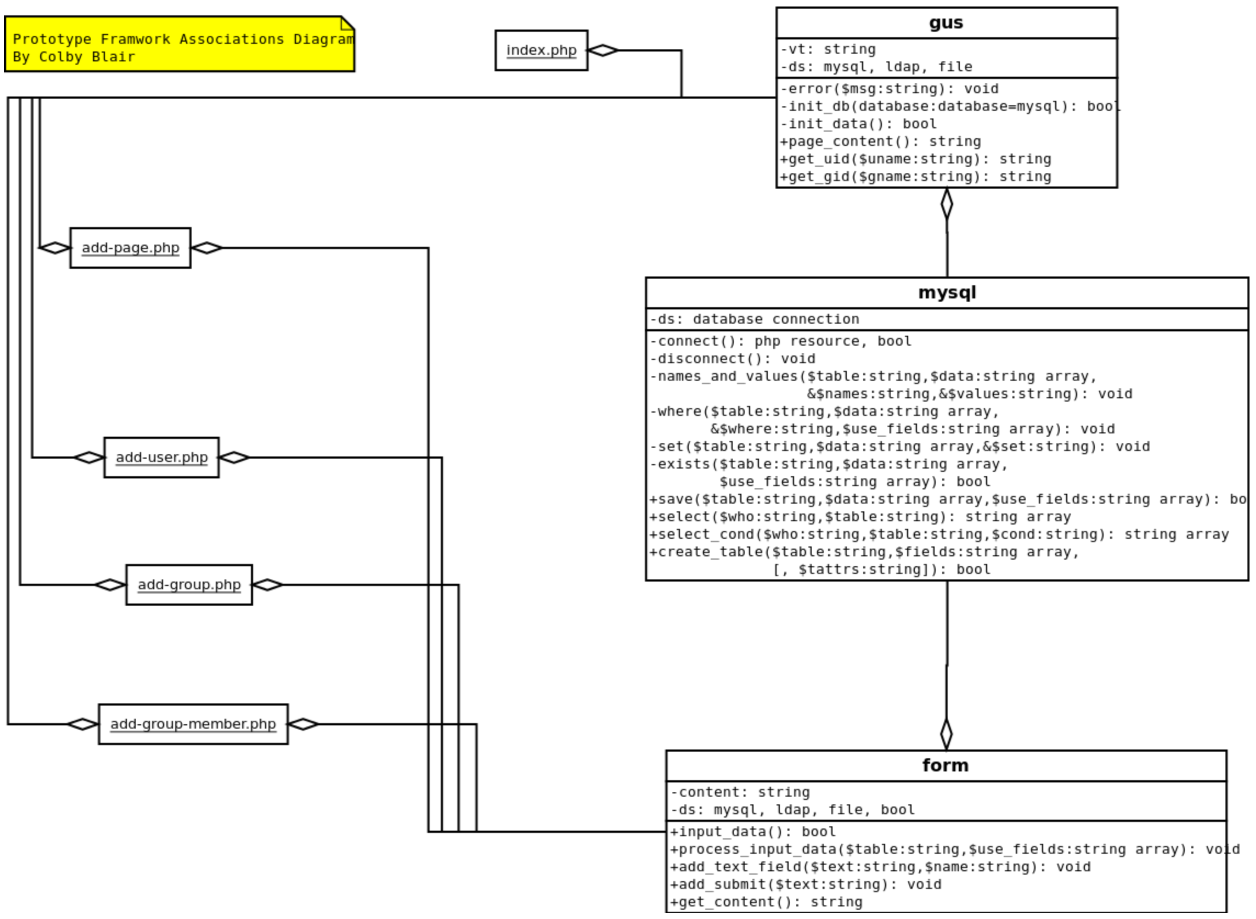

Prototype Framwork Associations Diagram
By Colby Blair
index.php
gus
-vt: string
-ds: mysql, ldap, file
-error($msg:string): void
-init_db(database:database=mysql): bool
-init_data(): bool
+page_content(): string
+get_uid($uname:string): string
+get_gid($gname:string): string
add-page.php
mysql
-ds: database connection
-connect(): php resource, bool
-disconnect(): void
-names_and_values($table:string,$data:string array, &$names:string,&$values:string): void
-where($table:string,$data:string array, &$where:string,$use_fields:string array): void
-set($table:string,$data:string array,&$set:string): void
-exists($table:string,$data:string array, $use_fields:string array): bool
+save($table:string,$data:string array,$use_fields:string array): bo
+select($who:string,$table:string): string array
+select_cond($who:string,$table:string,$cond:string): string array
+create_table($table:string,$fields:string array, [, $tattrs:string]): bool
add-user.php
add-group.php
add-group-member.php
form
-content: string
-ds: mysql, ldap, file, bool
+input_data(): bool
+process_input_data($table:string,$use_fields:string array): void
+add_text_field($text:string,$name:string): void
+add_submit($text:string): void
+get_content(): string

Figure 12: Prototype Framwork Associations by Colby Blair

3.2.3.30 Modify Privaledges

3.2.4 Developer's Architectural Rationale

3.3 [insert name of viewpoint] ARCHITECTURAL VIEW

This subsection contains the descriptions of a system and all of its major components, using the methods, techniques, and languages from other than the developer's or user's viewpoint. Each viewpoint description includes the viewpoint identification, description, and diagrammatic representation.

Repeat this subsection for each viewpoint identified.

3.3.1 [insert name of viewpoint]'s View Identification

Identify the view, state the purpose of the view, and identify major components or processes of the architecture.

[Insert text here.]

3.3.2 [insert name of viewpoint]'s View Representation and Description

[Insert diagram and descriptions here.]

3.4 CONSISTENCY OF ARCHITECTURAL VIEWS

For compliance with ISO/IEC 42010:2007 (§5.5) an Architectural Description (AD) shall include a list of all known inconsistencies between the architectural views and an analysis of consistency across all the architectural views.

3.4.1 Detail of Inconsistencies between Architectural Views

[Insert text and graphics here.]

3.4.2 Consistency Analysis and Inconsistency Mitigations

For each inconsistency identified above, provide solutions or mitigations that resolve potential conflicts between the stakeholder viewpoints.

[Insert text or table here.]

4 Software Detailed Design

4.1 Developers Viewpoint Detailed Software Design

4.2 Component Dictionary

Component Dictionary				
Name	**Type/Range**	**Purpose/Function**	**Dependencies**	Subordinates
Form	web-page	graphical user interface	requires GUS to transmit data back and forth to database	NA
GUS	web-server	interface between users and database management system	requires users to enter data and requires the MySql database to manage data	NA
MySql	database management system	maintain information, such as, user details, and student group organizations	requires GUS to transmit data back and forth from users	NA

4.3 Component Detailed Design

4.4 Developer's View Identification

4.4.1 Detailed Design for Component: Forms

The forms.php was written by Colby Blair, the section was reverse-engineered by Cynthia Rempel

4.4.1.1 Introduction to Forms Users can use forms to tell GUS what is needs to know; their personal information, such as names, majors, phone numbers, etc.; what groups they want to join; allow users to post to forums; and any other data a user might want to input into GUS. Forms also display data organized by the MySql class in a way that is meaningful for example: a forum organized by topic and then chronologically, or an organization chart for a group.

4.4.1.2 Input for Forms

1) textual user input

2) content from the GUS class

4.4.1.3 Output for Forms

1) content for GUS class

2) information displayed on a web page

4.4.1.4 Forms Process to Convert Input to Output Forms collects user input by using **text-fields**, into an attribute called **content**, which is then collected by GUS. Forms also takes **content** from GUS and displays it to users through **text-field**s.

4.4.1.5 Design constraints and performance requirements for Forms Forms must encrypt passwords, and only transmit sensitive data through secure protocols. Forms must collect the minimum amount of data about a user to meet the mission of student group organization and recruitment to minimize the desirability of GUS as a target for maliciousness.

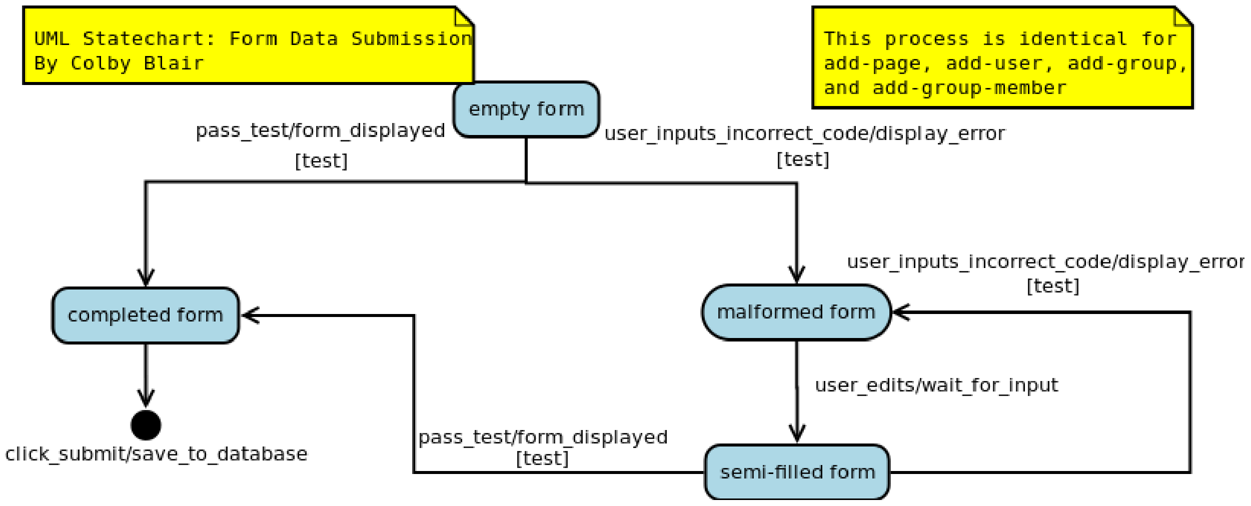

UML Statechart: Form Data Submission
By Colby Blair
This process is identical for add-page, add-user, add-group, and add-group-member
empty form
pass_test/form_displayed [test]
user_inputs_incorrect_code/display_error [test]
completed form
malformed form
user_inputs_incorrect_code/display_error [test]
user_edits/wait_for_input
click_submit/save_to_database
pass_test/form_displayed [test]
semi-filled form

Figure 13: Form Submission Statechart by Colby Blair, transition descriptions by Cynthia Rempel

4.4.2 Detailed Design for Component: GUS

The gus.php was written by Colby Blair, the section was reverse-engineered by Cynthia Rempel

4.4.2.1 Introduction to GUS

GUS uses a **visual template** and a **database connection** to initiate a database connection, and set-up page content. The visual template attribute is used to translate information obtained through the database connection into a user friendly format.

4.4.2.2 Input for GUS

1) content from a form
2) content from the database connection

4.4.2.3 Output for GUS

1) a visual template for a web-page (if it's going to the user)
2) database entries (if it's going to the database connection)

4.4.2.4 GUS Process to Convert Input to Output

1) Initialize Database Connection (**init-db**) this private function establishes a connection with the database by creating a new MySql object that has the correct data base host, admin, and uses the correct password.
2) Initialize Data (**init-data**): this private function queries the database through the database connection to set-up a visual template. It returns the first result, or if there is no result, it returns the original visual template.
3) Page Content (**page-content**): this public function gets a visual template (or makes one), and queries the database to find a web-page. If the page doesn't exist it returns an error message.

4.4.2.5 Design constraints and performance requirements of GUS

GUS must be able to handle incorrect user input, and deal with corrupt data from the database.

4.4.3 Detailed Design for Component: MySql

The my.php was written by Colby Blair, the section was reverse-engineered by Cynthia Rempel

4.4.3.1 Introduction to MySql

The MySql class uses a database connection (**db**) to: manage and query a database, and connect to GUS.

4.4.3.2 Input for MySql

[insert your text here]

4.4.3.3 Output for MySql

[insert your text here]

4.4.3.4 Component/Entity Process to Convert Input to Output

1) Connect: this private function uses a **hostname**, a user-name (**un**), and a password (**pw**), and returns a connection to GUS.
2) Disconnect: this private function closes the connection to GUS
3) Names and Values: this private function fills a **table** with **data** and assigning that data **names** and **values**

4) Where: this private function queries **data** in certain fields (**use-fields**) in a **table** and returns a string (**where**) with all the desired values.

5) Set: this private function cycles through all the **data** in a **table** and returns a **set** of keys and values.

6) Exists: this private function takes the (names) and (values) of the **table** and **data** and looks to see if the data exists

7) Error: this is an optional public error message function

8) Save: this public function goes through a table and updates the **data**, **row** by row updating only the fields indicated. The rows that don't already exist in the table are inserted.

9) Select: this public function queries a **table** to get an array of results **who** have the indicated attribute.

10) Select Condition (**select-cond**): this public function cycles through all the rows in a table and returns all results **who** meet the **cond**ition in the **table**.

4.4.3.5 Design constraints and performance requirements of MySql The query language must be limited to the SQL dialect the database uses, and the other side of the interface must be compatible with GUS.

4.4.4 Detailed Design for the Installation Component:

4.4.4.1 Introduction to Installation Component: The installation component sets up the database, and creates the tables: page, guser (gus user), ggroup (gus group) ggroup-member (gus group member).

4.4.4.2 Input for the Installation Comnponent Although there are no explicit inputs it uses the following global parameters: the mode, the host's IP, the admin's information, password, the database's name. The choices for the mode are: MySql, LDAP, and file-driven.

4.4.4.3 Output for this Component/Entity Currently, the output for the installation module is a MySql database composed of tables.

4.4.4.4 Installation Process to Convert Input to Output The installation process connects to the host and uses a switch to identify the type of database, then the installation uses a CREATE DATABASE query, and finally a series of CREATE TABLE queries.

4.4.4.5 Design constraints and performance requirements of the installation component The component must successfully connect to the host IP securely, without destroying any data, or compromising personal information.

4.4.5 Detailed Design for Component: Add Group

4.4.5.1 Introduction to Add Group Add group is designed to use a form to add a group to the GUS database.

4.4.5.2 Input for Add Group Add group takes the name and description of a group as input.

4.4.5.3 Output for Add Group A new entry in the ggroup (gus group) table.

4.4.5.4 The Add Group Process to Convert Input to Output Takes the name and description of the group and calls on the GUS interface to add the entry to the group table.

4.4.5.5 Design constraints and performance requirements of Add Group The only people that have authorization to add university groups are: ASUI, the Graduate and Professional Student Association, and (possibly) the University of Idaho Student Bar Association. Although it probably would be advisable to include the existing graduate, professional, and legal student groups in the original installation and limit adding groups to just ASUI, because there are very few new graduate and legal groups being added, and ASUI can assist the student bar and GPSA with adding those groups (if they ever need to).

4.4.6 Detailed Design for Component: Add Group Member

4.4.6.1 Purpose of this Component Adds a member to the group membership table.

4.4.6.2 Input for this Component The input is the group ID and the user ID.

4.4.6.3 Output for this Component After this script is called an entry is made the group membership table.

4.4.6.4 Component/Entity Process to Convert Input to Output Hands the group ID and the user ID to the GUS interface to be processed and added to the group membership table.

4.4.6.5 Design constraints and performance requirements of this Component There should be a uniqueness constraint on user-ID and group-ID. For example: although john2833 can be in both Business Professionals of America (BPA) and the UI Tech Club, once he's in BPA, he should not get a second membership entry for BPA if the BPA secretary accidently tries to add him when he is already in BPA.

4.4.7 Detailed Design for the Main Component

4.4.7.1 The purpose of the main component is to identify the appropriate global parameters when installing GUS. The parameters are the database mode (DBMODE), the database host (DBHOST), the database administrator (DBADMIN), the default database user name (DBUN), the database password (DBPW), the database name (DBNAME), and the default template directory (TMPLDIR).

4.4.8 Detailed Design for Component: Uninstall

4.4.8.1 Purpose of this Component: The purpose of this component is to uninstall GUS from a computer.

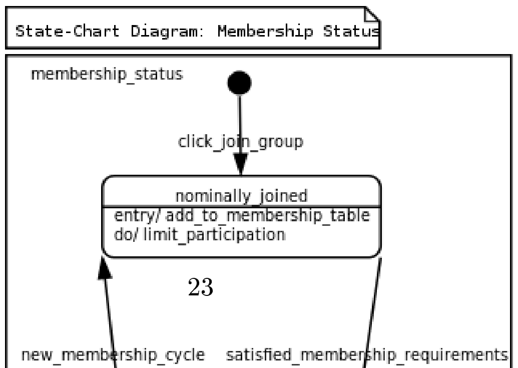
4.4.8.2 Output for Uninstall Uninstall removes the database.

4.4.8.3 Component Process to Convert Input to Output Connects to the database, and performs a DROP DATABASE command in MySql.

4.4.8.4 Design constraints and performance requirements of this Component Access to this feature should be very limited – one command can remove an entire social network.

4.4.9 Detailed Design for Entity: Membership Status

4.4.9.1 Purpose of this Entity The purpose of membership status is to enable groups to give benefits to members who have earned them. For


State-Chart Diagram: Membership Status
membership_status
click_join_group
nominally_joined
entry/ add_to_membership_table
do/ limit_participation
new_membership_cycle
satisfied_membership_requirements

example: Business Professionals of America may require members to pay $30 dues, sign a waiver, and assist with a fundraising event **before** paying for the student to go to the State Leadership Conference.

4.4.9.2 Input for this Entity

- a request from the member, officer, or administrator to change the status of the member
- a triggering action, such as, paying dues, signing a waiver, or attending a mandatory meeting
- the end of a membership cycle: such as, the end of the school-year, graduation, change in the groups staff

4.4.9.3 Output for this Entity If the conditions are met, a change in the membership table to reflect the change in the status.

4.4.9.4 Process to Convert Input to Output

4.4.9.4.1 If a user makes a request, GUS checks to see if the user is authorized to make the requested change, if not, the user's request is forwarded to the approving authority, if the group so desires. If the user is authorized to make the change, then the change is made in the table through a php call to the database.

4.4.9.4.2 If an authorized user checks off that a membership condition has been met, that information is stored in GUS. GUS then checks to see if all membership criteria for the group has been met, if so, GUS updates the members status.

4.4.9.5 Design constraints and performance requirements Membership requirements would have to be unique to each group – Lutheran Campus

4.4.10 Detailed Design for Entity: Membership Status

4.4.10.1 Introduction/Purpose of this Component/Entity [insert your text here]

4.4.10.2 Input for this Component/Entity [insert your text here]

4.4.10.3 Output for this Component/Entity [insert your text here]

4.4.10.4 Component/Entity Process to Convert Input to Output [insert your text here]

4.4.10.5 Design constraints and performance requirements of this Component/Entity [insert your text here]

4.5 DATA DICTIONARY

This subsection shall list and describe all the data and data structures defined and/or used by the components and entities specified above. For each data item or structure indicate where it is defined, referenced, and modified.

4.6 Function Dictionary

Function Dictionary				
Name	Type/Range	Defined By	Referenced By . . .	References
add-text-field		4.3.1.4		name
error	message			
init-db	GUS	4.3.2.4		ds, mysql
init-data	GUS	4.3.2.4		vt, mysql.select-cond
page-content	GUS	4.3.2.4		vt, ds.select-cond
construct	MySql			connect,

5 REQUIREMENTS TRACEABILITY

This section shall contain traceability information from each system requirement in this specification to the system (or subsystem, if applicable) requirements it addresses. A tabular form is preferred, but not mandatory. A detailed mapping between requirements and constraints in the SSRS and architectural components and detailed entities in this SSDD is required. For compliance with ISO/IEC 15288:2008 (§6.4.3.3.c) an Architectural Description (AD) shall provide roundtrip traceability between the system and software requirements and the architectural design entities. All requirements and constraints within the SSRS shall map to a set of architectural entities. All entities in all the architectural views shall be associated with either a requirement or constraint in the SSRS or an architectural constraint within this SSDD.

Priorities are: **M**andatory, **L**ow, **H**igh