Final Report

Analysis of the Team WPWPWP Project

An Hu

Tianjao Mo

Zhihao Xue

05/20/2015

Contents

Executive Summary

This final report document begins with a description of the initial problem that created the basis for this project. The specific solution implemented to address that problem, the major challenges of he proposed solution, and the overall design of the database used as part of the solution are discussed following the introduction of the problem. Finally an analysis of the result of implementing the particular solution is performed, citing both the strengths of the particular approach and its weaknesses. An appendix is included at the end of the document containing the final entity relationship diagram and relational schema diagram for the project database. Further document references are pe3ovdied through a glossary of key terms, a external references list and an index.

Introduction

This document is the final report of Project WPWPWP (what happened, what's happening, what will happen) implemented by An Hu, Tianjiao Mo and Zhihao Xue. The purpose of this report is to take the problem outlined in previous documents and discuss the effectiveness of the solution implemented by Team WPWPWP. In examining the effectiveness of the solution this report will list feedback from stakeholders on user experience and previous documents.

Problem Description

The on-campus activities and events are a large portion of college life. However, currently, all those kinds of events are announced via SharePoint ®, such as Academic Seminar or posted calendar on webpages, like IM Field activities. In this way, students, faculties and staffs barely have interactions either with these events and activities, or with other friends. So, it is exigent to find a new way to keep all members be involved in. To solve this, we develop this new system, which generate a new, easy, efficient and convenient way, to tack events and schedules. Also, this this system will give students a new solution to create, join and share activities. In this way, we hope everyone can interact with campus more.

Below, it is the list of features of the end system as listed in the final version of Team WPWPWP's problem statement.

Feature #	Feature Name	Feature Explanation
1	Web-based application	Our project will run over the internet
2	User account	User can create account in our system and use their own account to login.
3	Schedule lookup	User can track current status of events
4	Creation	User can create public or private event
5	Invitation	User can invite their friends to join events
6	Join	User can join any public events

7	Search	User can search events by giving specific condition.
8	Friendship	User can become friends with other users.

Solution Description

The solution decided upon by Team WPWPW was to develop the web-based application using HTML, CSS, JavaScript for our front-end and a Microsoft® SQL Server 2012 to maintain all the information related to our database. Front-end and back-end are connected using PHP.

Front-End Discussion

On the front-side, we use HTML, CSS and JavaScript to build our web page and GUI(graphical-user-interface). The web page layout is designed as easy to use as possible. Each operation has its own button, and when the button is clicked, a new dialog box will pop up to tell user what to do. Also, few drop-down menus added to help user to make decision, such as mark an event public or private. When user logs into the system, he can create event, view event, and edit it. User can send invitation to his friends to attend a particular event together. User are able to accept invitation from his friends as well. When user creates or edits an event, EventName, start time and end time are required, other field are optional. Meanwhile, to main database consistency, start time must be earlier than end time. If you want to make friend, you have to know username whom you want to add to. To maintain our database secure, only limited access are provided to users.

Back-End Discussion

The GUI runs through the use of the WPWPWP database and the information contained with that. Tables containing information of users, events, organization and friendship. PHP is used to communicate between front-end and back-end. Currently, our database exist on the Rose-Hulman Titan server of Computer Science and Software Engineering Department. A more in depth description of database can be found in the Database Design section.

Database Design

Entity Relationship Diagram and Relational Schema

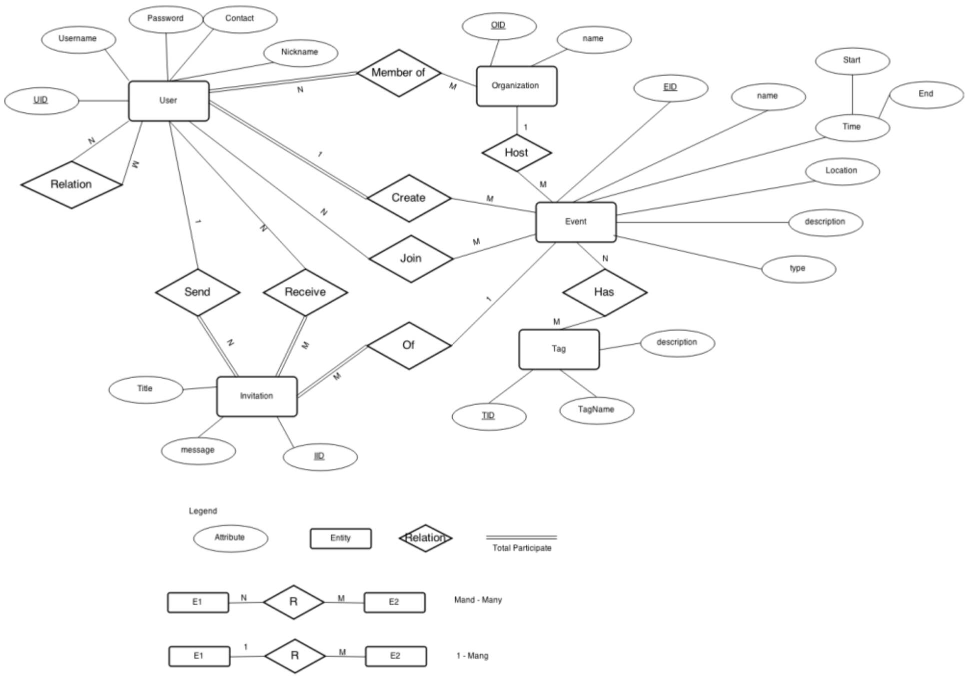

Fig 1.1 Entity Relationship Diagram

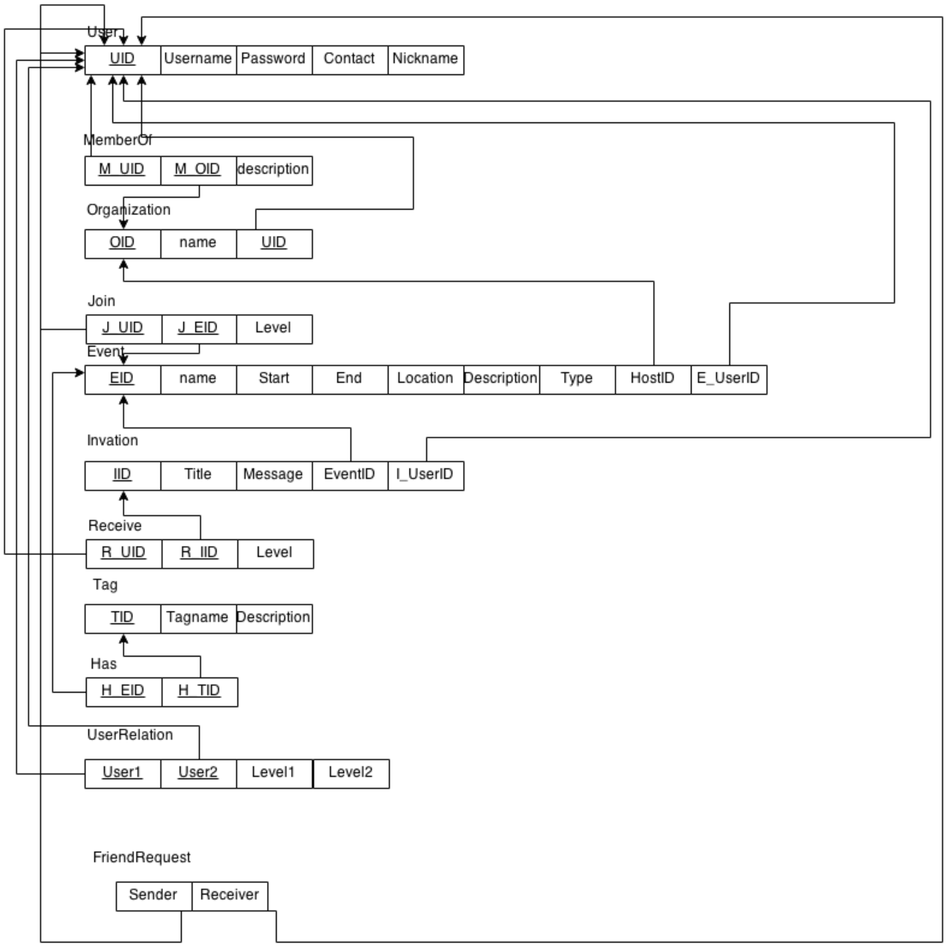

Fig 1.2 Relational Schema

User information store in User table. For each user, he can send friend request to other users. Once the receiver confirms the request, they two become friends and their relationship will store in UserRelatioh table. User also can create an organization. Once an organization is created, other user can join in this organization. And an event that host by this organization can be created by this creator.

User is capable of creating events. One user only create one event at one time. And the owner has the authorization to edit and delete this event.

Event has only two types: public and private. When an event is created, user must choose event type by select from a drop-down menu.

If an event is public, then, everyone has access to join this event. Otherwise, user must be invited to join a private event. User who already joined an event and has authorization to send invitation is able to invite other user no matter public or private.

Each event has it own tag which indicated its category, such as academic, personal, sale, ride etc. Tag is used for user searching and identifying.

Security Measure

The web page accessed the database through a user account associated with the password. All direct interaction between the front-end and database is limited only to executing stored procedures. This provides added security by first off eliminating the need for the user to build his own queries. Besides, we disabled ';' and few other characters to prevent SQL injection. Also, Also, at front-end, all data passed to stored procedures are handled as parameters and are not made part of query through concatenation. Besides SQL injection, user in our system has different levels to indicate different authorization. User with lower level grade can have more access than user with higher lever grade.

Integrity Constraints

Various referential integrity constraints exist within the database and are represented as arrows on the Relational Schema shown in Fig 1.2.

The following is a list of the domain integrity constraints that exist in the database:

- All primary keys are integers and increment automatically within database;
- All primary keys are greater than 0;
- Event.start time must be earlier than Event.end time;
- All names are unique except Invitation.title;
- Event.type only be one of public or private;
- Each user join an event must associate with a user level;
- Tag table only can be modified by database administrator, not user;

Stored Procedures

The following list contains all stored procedures in alphabetical order

Name	Function
addTag	Administrator uses this stored procedure to modify Tag table
check_event	List all events that current user has joined
check_event_tag	List all the tags of the given event

check_invitation	List all invitations that current user have
check_joined_organization	List all organizations that current user has joined
check_my_current_event	List all user joined events which happening now
check_my_organization	List the organization the current user created
check_people_in_event	List all people who joined a particular event
check_request	List all the friend requests received by the current user
check_tags	List all the tags stored in the database
confirm_invitation	User accept the invitation and join the corresponding event
confirm_request	User accept the friend request and then the receiver and sender will become friends.
create_event	User uses this stored procedure to create an event and specify the attributes of the event.
create_invitation	User uses this stored procedure to create an invitation and writes the content, then the invitation will be ready to send
create_organization	User uses this stored procedure to create an organization and then other users can join this organization
create_Account	User uses this stored procedure to register
delete_event	User uses this stored procedure to delete the events which created by himself
delete_request	Delete an request sent by the current user
delete_invitation	Delete an invitation created by the current user
delete_organization	Delete the organization created by the current user
edit_event	User uses this stored procedure to edit the attributes of an event which he or she has the right to edit
edit_user	User uses this stored procedure to edit his or her own information
event_can_invite	List all the events which the current user has right to invite others to join
friends_and_event	List all the basic information of the friends of the current user including the events they are attending
get_all_organization	List all the organizations in this database
join_event	User uses this stored procedure to join an public event, request will be rejected if the event is private
join_organization	User uses this stored procedure to join an organization
Login	User uses this stored procedure to log in
member_in_organization	User uses this stored procedure to check the members of an organization
my_event	List all the events created by the current user
my_friends	List all the basic information of friends of the current user
my_invitation	List all the invitations received by the current user

saltedHash	The procedure used to hash the user's password in order to secure the account
search_by_tag	User uses this procedure to search event by tag
search_event	User uses this procedure to search event by specifying attributes
search_user	User uses this procedure to search other users
send	User uses this procedure to send invitation to other users
send_request	User uses this procedure to send a friend request to other users
set_level	User uses this procedure to set other users level in a friendship, users who has a level higher than the level setting of this user will be able to see the event that he or she is attending.
update_current_event	User uses this procedure to update the event that he or she is attending currently.

Indices

Additional indices are add into the database to help improve searching function.

Non-clustered index has been added onto Event.Name, Event.Location and User.username.

Triggers

EventUserTrigger: If any event lose both HostID(organization) and E_UserID(creator), that event will be deleted automatically.

EventOrgTrigger: If any event lose both HostID(organization) and E_UserID(creator), that event will be deleted automatically.

Key Challenges

- Challenge: Little experience with SQL Server and front-end
  - The one who familiar with front-end is in charge of front-end and communication with back-end. The rest of the team divided up to build and test the database to establish a convenient environment for front-end.
- Challenge: Hard to alter table with foreign key
  - When we want to modify a table or column with foreign key, the query is usually rejected. We have to delete the foreign keys first, make the changes, and then re-establish the foreign keys
- IS NOT NULL and <>
  - A little bit confusing between data structure check and reference check.

Design Analysis

Strengths

- The system seems to have a high level of security since all the deletion to any entity can only be done by the creator and edition could only be done by user who has high level privilege.
- Passwords are encrypted with a saltedhash procedure thus to enhance the security of user information
- All user actions are supported with stored procedures thus to enhance the security of the database.
- All of the stored procedures do not take any ID as argument but take some other unique values to locate certain rows thus to enhance the security of database.
- All stored procedures contain explanations to errors.
- Different reactions to delete/update with foreign keys to keep the consistency of database.
- Attribute names are relatively clear and descriptive of what they contain. Only a few are ambiguous abbreviations.
- Non-clustered indices are created for columns that are highly possible to be used in any search action.

Weaknesses

- Passwords are hashed on the database end which means during the transmission if the information is captured by others, it could be seen directly.
- Two procedures used to search event are better to be combined into one.
- Repeated checks exist, although they won't cause any error but reduce the efficiency of code.

Glossary

- GUI: graphical user interface refers to images displayed on the screen that user interacts with through the keyboard and mouse
- Index: in the sense of databases, this is a mapping that allows the database system to quickly locate desired data
- Integrity Constraints: limitations imposed on the database to preserve certain relationships among the data and permit only certain values to exist.
- PHP: a server-side scripting language designed for web development but also used as a general-purpose programming language.
- Query: a specific instance of SQL code to accomplish a certain task once it has been sent to the database
- SQL: a language used for specifying actions to perform on a database system

- Stored procedures: a stored procedure is a piece of coded functionality that performs a specific task each time it is called and is maintained on the database itself.
- Trigger: similar to a stored procedure in that it is a piece of functionality that performs a specific task each time it is executed the main difference is that triggers will execute automatically whenever information in a table is altered in some way