Project Globus

Senior Design I

David Crane

Jesse Miller

Kelsey Crea

Taylor Olson

**ABSTRACT**

Project Globus is a group management system designed to work for a variety of groups, both casual and professional. Whether it’s a group of friends planning a weekly board gaming session, student’s working on a school project or a business team working on a company’s next biggest product, project Globus will provide an innovative, simplistic and versatile experience to keep the group organized via document sharing, group notifications, group calendars and group messaging.

1. **Introduction**

Project Globus is our senior design project that is a group management system. We decided to implement an Android application for senior design I and then make the application cross platform for senior design II on IOS and as a web page. We have implemented the Android platform for senior design I. Throughout this paper, we discuss how the project started and how we planned the project through the waterfall approach with a dedicated timeline, as well as the design of the user interface and how the primary application features were developed. Finally, we conclude with issues and problems that we encountered while we were working on this project in senior design I as well as our use cases for the application that we have developed. Project Globus is a group management system designed to work for a variety of groups, both casual and professional.

1. **Origins, Preplanning and Planning**

This project idea started in early September of 2014 with the needs of one of our group member’s business fraternities, Alpha Kappa Psi, needs for a better group collaboration application. As a group, we decided to stem the project from Android to IOS and web page and that we would tackle the Android application during senior design I and the IOS application and web page in senior design II. We decided on the name “Project Globus” because it incorporates both group and globe into one word, Globus, which is how far our application can span.

As a team, we sat down and discussed how to strategically plan what aspects we were looking for in the project. We decided that we wanted a versatile application that could be aimed at both professional and causal applications (see use case’s citation or something here). From this broad target audience, we then discussed the individual features that we wanted the application to be capable of having. We broke the project up into multiple sections in order to adhere to a waterfall approach of the project. The planning process was the first step that we took as a group in the project.

The first planning meeting that we had, discussed the advantages and disadvantages of different aspects of the application. We decided that we wanted a central calendar that the application would revolve around. The application would start off with a typical login page that most other applications have, leading into a group selection page, and from there, a centralized master group calendar, Google drive link, attendance check, blanket messages, message feed that we deemed ‘The Whiteboard’, and a permissions section would be included in the application. We then decided that the central database would be the backbone of the project and started discussing how we would structure the database. This first communication about the database was to develop table structures and relationships. Once the database development was underway, we started writing the Android application in eclipse. We developed the Android application almost until completion and will link the application with the database over winter break and complete the app. The Android application will be completely done by January 1st, 2015.

1. **Database flow chart and description**

The first definitions of our database were designed such that we could keep as many objects as separate as possible, to ease creation of our classes on the application side. In our design, there were five major tables that controlled the flow of information: the tables for *Users, Groups, Messages, Calendar,* and *Calendar\_Events*. These tables held the primary keys that would help validate data stored in the other tables using foreign keys. Those tables were *Group\_Members*, which tied users to groups, and *Event\_Members*, which is used for attendance and organization of events. See Image 1 for a complete database implementation table.

Setting up the database on a SQL server proved quite easy using Amazon’s Relational Database Service (RDS). Creation of the database, necessary tables, and granting remote access proved to be an easy task. The next step was writing the initial code for our application to interact with the database. We used a SQL driver issued by Oracle, and coded access functions that worked directly with the database, manipulating it as necessary to populate and control information within the tables. We knew this was a naïve implementation, but this first step was necessary for proof-of-concept connection with the database. Eventually, we would need to create a server-side application to handle requests from the clients for us, allowing for greater security and faster access times.

1. **Application design and flow**

The design of the application tried to follow some very basic, but overlooked ideas of user interface design. When designing the user interface, we wanted to follow several fundamentals that would be strictly enforced throughout the entire product. Those fundamentals are: know the audience, consistency, keep it simple, and empower the user. Knowing the audience involved us sitting down and analyzing our target audience, which turned out to be very large. This meant that we didn’t have the luxury of catering the design to one specific type of user. Rather, we had to make an interface that wouldn’t be “too dumbed down” and “too simple,” while at the same time it could not be too complicated that non-technical users would be able to use it without having to spend much time learning. Consistency was a no-brainer for us. We wanted to make sure the user was familiar with all aspects of Globus and no one area of the system was extremely different than another. Keeping it simple goes back to knowing the audience. We decided to go with a simple color scheme of blue and gray. This would allow us to have a nice contrast while still keeping the user engaged on what’s important – the content. Finally, empowering the user was very important. We wanted the user to feel as if they were running a group system, not a group system running them. Therefore, we focus the interface of Globus to be very non-protruding to the group’s productivity. We are also keeping a very open mind and are going to get feedback from users when we are in the beta testing phase.

Once the user interface was planned and designed, we decided to work on the flow of the application itself. The application was originally split into two activities but then later a third activity was added in order to make the android programming simpler and easier to understand.

We broke the first activity into several android development fragments that includes the ‘Welcome fragment’, the Login fragment’, and the ‘Create a new account fragment’. This activity has database interaction to validate the credentials of what the user enters. Once validity of the user entered data is confirmed the first activity is closed and the username is passed from activity one to activity two and the second activity is started. As a group we decided to move from activity one to two because once the user was logged in, they would not need to log in over and over again while the application was open.

When the second activity starts, the database is queried to grab the users’ group information before the ‘User Page fragment’ loads. The ‘User Page fragment’ loads the groups that the user is already a member of and has the options for the user to join a new group or create a new group. All of these options lead to the third activity. If the user chooses the option to create a new group, then the ‘Create a new group fragment’ will load on the screen. The user will enter the appropriate group information and then that information is check for validity against the database. If the information is valid then the third activity is started. If the user chooses the option to join an existing group, the ‘Join Group fragment’ will load. The user will enter the groupID and password and then the database checks for validity of the information. Once the user entered information is designated as valid, the third activity will start. The user also has the option to click on one of the groups that they are already a member of. When this happens, the third activity loads.

Upon the third activity starting, the group information is passed from activity two to three. Unlike the first activity, when the third activity starts, activity two is still running in the background. This gives the user the ability to go back and select a different group if desired.

Once the third activity loads the main application page is displayed to the user. The activity contains a multi pane fragment, meaning that multiple fragments are displayed within the window at the same time. The user defined group name is displayed across the top of the screen, a fragment that contains buttons is displayed on the left and a fragment that contains a message feed, that we have deemed ‘The Whiteboard’ is displayed on the right hand side of the screen. The two fragments have a 1 to 3 weight relationship for the buttons and whiteboard respectively. The user has the ability to click on the buttons within the left fragment and then the corresponding fragment will load display where the whiteboard fragment is located (for more information see section VII).

The user can click on the calendar button and the database is queried for all calendar and calendar event information. Then the calendar is displayed on the right side of the window. The user can add a new event by clicking the ‘New Event’ button and the ‘Create a new event fragment’ is loaded. The user will then enter information and it will be stored as a new entry in the database and displayed upon the main calendar. The ‘Create a new event fragment’ will then close, and only the calendar fragment will be displayed.

The user can click on the attendance button which will display the ‘Attendance fragment’ in the right side of the screen.

The whiteboard button can be clicked on and the whiteboard will be displayed on the right side of the screen. This is a message feed that displays events, messages, and other information that is important to the group. This information is pulled from the database calendar as well as the appropriate database tables (for more information see section III).

The user can click on the google drive button which will load the Google Drive web page within the right half side of the window. This is to keep the user inside of the application but still giving them the ability to access a group google drive page. The database is also queried to get the stored google drive login information and display it to the group at the top of the fragment.

The user can click on the settings buttons which will display the ‘Settings fragment’. This fragment will give the users the ability to set permissions for group members.

At all points during the application the user can select the back button. The back buttons will bring the user back to all current pages in the activities that are open. This means that the user can easier navigate forward and backward through the login pages, and then through the second and third activities.

1. **Login group pages and database interaction**

When a uses logs in to Globus, we will need to connect to the Globus database and pass it the username and password that the user provided. If the user has been listed in the database and their password matches the stored password, than the application will immediately log him/her in and send the device he/she is using the id associated with the user. If the user creates a new account, then the user will have to fill out there information and pass it to the app. At this point, the app will query the database to determine if there is already an account with the username/email address given by the user. If there is, the app will not create the account and instead it will inform the user that the username is already in use. If it isn’t, then a new entry will be added to the users table in the database containing the given information and the user will be informed that his/her account has been created. The user will also be logged in to his/her new account.

1. **Group selection pages and database interaction**

After logging in, the user will be sent to a fragment containing a list of groups he/she is currently a member of. This list will be obtained through the database, querying it for all unique groups listed with this user’s id in our group members table (the table contains a list of group ids attached to user ids, showing what users are members of specific groups. The group names, ids and descriptions will all be pulled from the database, and fed into the list for the user to see. At this point, the user can either select one of his/her current groups, create his/her own group or join a new group. If the user chooses to create a new group, they will have to input the name, description and password of the group they wish to create. The app will then query the database to make sure the group name has not already been used. If it has, the app informs the user that he/she needs to change the name. If it hasn’t, the app will add a new entry into the groups table of the database, including not only the information provided, but also the user’s id which will be used in the future to give the creator administrator privileges. The user will then be sent to the Whiteboard If the user wishes to join a group, they have to enter the group’s id number into the app. The app will then query the database to see if there is a group with the given id, and ask the user if they wish to join the group if there is. Otherwise, the user will be prompted that no groups could be found, and they will be asked to enter in a new id. After the user either creates, joins or selects a group he/she is already a member of, the app will send the user id and group id to the main activity of the application which will initially take them to the Whiteboard display.

1. **Main application portions that are completed**

In its current state, our Android implementation of the Globus application can be considered a good concept and (or) rendition of the final product. In the app, the login, create account, group selection, create group, join group, Google Drive and Whiteboard fragment have been completed in most every aspect except for database integration. That is to say, as soon as we are able to connect the app to the database, the majority of code we will need to add will be interacting with the database. Aside from database work, the only code that will be written for these fragments will be for fixing bugs and updating the user interface for its initial launch.

Along with these sections of the application, we also have the beginnings of our calendar fragment (the fragment that will allow users to see all group events and add new events to the group). The reason we consider this fragment more of a mock-up than a fully implemented fragment is because the majority of this code relies on the database, whether we need to pull a list of calendars to find the correct calendar to display, find all the events listed under a certain group or add new events to the group’s calendar. Until the database can be connected to the application, it will be very difficult to test the functions of this screen.

1. **Database problems**

Once the database code was written, the next step was to use the code with the android application. This proved to be a very significant issue. At first, the entire android application would crashing, returning many errors, both android-based and SQL-based. After many hours of debugging, a trip to Jim Ward was necessary. He pointed us in the right direction by pointing out that our database adapter was compiled for regular Java rather than Android Java. Once a proper adapter was found on Oracle, only one error was given. That error was a connection issue that was given by our own database code. This turned out to be an issue with Android security not allowing the connection over the port. Because of the issue, we are developing server-side code to run all the functions that will feed to the mobile applications.

It was a serious setback to learn that Android could not maintain persistent SQL connections—one that made a server-side handler for database queries immediately necessary. However, there is no template or default code in Java to handle such tasks readily available. As such, we’re currently working to design our own client-server code to create and process requests efficiently.

The current code, still in progress, runs as an intermediary that creates a new thread for each client. Each thread then receives, validates, and executes the SQL commands on behalf of each client, before returning resulting information (if any). At the moment, we’re suffering from some sever concurrency issues, mostly involving correct thread termination after connections time out.

1. **Additional problems**

The decision to move from two activities to three activities came from attempting to display a multi pane fragment within an already created activity. This causes a memory leak within the application because a multi pane fragment is best suited to be loaded from the main activity. The solution that we decided would be best to fix the problem, was to create a third activity that initially loads the multi pane fragment. This way we were able to use the advanced android layout with multiple fragments we had already created, while at the same time created a small division in the application that still matched the flow that we envisioned for the program.

The navigation of the primary fragment of the application turned out to be a little less than trivial. This buttons ended up causing a memory leak within the app. It turned out that this was because the images were sized too large. Resizing them makes the memory leak cease, but the images are not as crisp as we would like. Further research into this will take place later on.

While working with the calendar fragment, many problems arose. First, we had to start learn about how Android systems store there user calendars and how to add new calendars. Aside from this, we also discovered that Android does not include a calendar object capable of showing events tied to certain days on the object itself. In order to allow users to see all group events in their corresponding days and times on the calendar object, we will either have to make our own custom calendar or download a third-party calendar. In the meantime, we have kept the current calendar object inside the fragment to give users an idea of what the completed fragment will look like.

1. **Work for winter break**

While initial development of the Android app comes to a close, winter break is going to be the time when primary development of the iOS app is going to take place. The difficult part of the mobile applications was designing the initial application and deciding where everything would go and how we wanted the user interface to be designed. This means that writing the iOS app will not be as planning-intensive because the design aspects of the application have already been decided. The primary goal of the iOS app is to match the Android app as closely as possible. This is to keep a consistent interface to for users within a group, no matter what operating system is used on their mobile device.

1. Time line (KELSEY)
2. **Mobile project vs Senior Design I** 
   1. Database  
      Implementation of the database for Mobile programming ended up not being a part of the equation. While attempting to work on database implementation, as we mentioned in section VIII of this paper, we ran into some issues. This in turn ended up being completely removed from the scope of our Mobile project. Rather we used hard-coded credentials, etc, for example purposes. While using this skeleton, we plan to implement our server-based database that would allow for the group management system to be available everywhere.
   2. Globus Features

In the Mobile Programming submission, the app is a very under developed version of the final product. It will not contain, or even discuss user privileges or attendance tracking/mandatory attendance, or the ability to send blanket messages to each member of the group. These are purely for the Senior Design version of our app. It will also not grow beyond the current state of the Globus app (this means it won’t ever have a fully functioning group calendar, or the ability to communicate with other users). The Mobile Programming application resides on one mobile device and no information is shared between a database or other users. The submission for Mobile Programming is essentially the fragment interaction and activity interaction that takes place within the application. The actual fragment and activity buttons and intricate code was not turned in for Mobile Programming.

1. **Use Cases** 
   1. **Use Case: Alpha Kappa Psi**

**CHARACTERISTIC INFORMATION**

**Goal in Context:** The fraternity uses Globus to manage members and pledges.

**Scope:** Fraternity

**Level:** Summary

**Preconditions:** The Alpha Kappa Psi chapter is already formed.

**Success End Condition:** The fraternity will have the ability to maintain communication in a central location and keep a centralized schedule keeper and message system.

**Failed End Condition:** The fraternity will not have the ability to maintain communication within a central location and will require the use of several different non-cohesive application methods.

**Primary Actor:** Members (especially officers), pledges, advisors.

**Trigger:** The chapter is initiated, pledges or members join.

**MAIN SUCCESS SCENARIO**

1. An Alpha Kappa Psi chapter is initiated.
2. One of the founding members creates a group on Globus.
3. The other members join the group on Globus.
4. The members create calendar events for the fraternity.
5. The members send group messages to the fraternity through Globus.
6. The secretary takes attendance through Globus at events.

**EXTENSIONS**

2a. This will require the user to create an account, then create a group.

3a. This will require users to create an account, then join a group.

**SUB-VARIATIONS**

5a. Members may also require official emails or written messages.

**OPEN ISSUES**

Any custom design requirements required specifically by Alpha Kappa Psi

**SCHEDULE:**

Due Date: release 1.0

* 1. **Use Case: 4 Student Project**

**CHARACTERISTIC INFORMATION**

**Goal in Context:** Four students uses Globus to manage meetings and communication for a group project.

**Scope:** Student Group

**Level:** Summary

**Preconditions:** The student group is already formed.

**Success End Condition:** The student group will have an effective system that will allow them to have a central location for communication and planning for the group project.

**Failed End Condition:** The student group will be required to rely on separate non-cohesive systems to manage scheduling and messaging within the group.

**Primary Actor:** Group members.

**Trigger:** The group has been established, the members join.

**MAIN SUCCESS SCENARIO**

1. A student group has been established for a group project.
2. One of the members of the group will create a group on Globus.
3. The other members will join the group on Globus.
4. The members will be able to plan collaborative meetings.
5. The members will be able to send group-wide messages.
6. The group will have access to their collaborative Google Drive account.
7. The group will have a central location to communicate in regards to that specific group project, which is not lost in the mix of other communication forms within their device.

**EXTENSIONS**

2a. This will require the user to create an account, then create a group.

3a. This will require users to create an account, then join a group.

**SUB-VARIATIONS**

6a. Documents may not all be digital, and shared physically.

**OPEN ISSUES**

If a group requires another form of communication, perhaps if one member uses Windows phone. To be discussed later.

**SCHEDULE:**

Due Date: release 1.0

* 1. **Use Case: Sigma Nu Fraternity**

**CHARACTERISTIC INFORMATION**

**Goal in Context:** The fraternity uses Project Globus to communicate through multiple groups, manage members and events.

**Scope:** Sigma Nu Fraternity

**Level:** Summary

**Preconditions:** Sigma Nu Fraternity's members have access to one of Android, iOS, or web application.

**Success End Condition:** Sigma Nu will have a single tool that can provide effective inter-member communication, maintain calendars and events, as well as member participation.

**Failed End Condition:** Sigma Nu will remain to using multiple applications under varied interfaces in order to maintain chapter operations inefficiently.

**Primary Actor:** All members, with particular use by chapter officers.

**Trigger:** Sigma Nu Fraternity is chartered, or fraternity membership changes.

**MAIN SUCCESS SCENARIO**

1. Sigma Nu Fraternity is chartered.

2. Chapter Commander creates a group for the chapter.

3. Members are invited to join (or join via password method) and are assigned roles and privilege levels by Commander.

4. Members use communication tool provided by Project Globus using the project's wide array of message types, importance levels and notifications.

5. Members use the project's calendar and scheduling tools to maintain chapter events and accurately record event leadership and participation.

**EXTENSIONS**

2. a. The user must register with our software and create a group.

3. a. Members must register with our software and be invited or join the group.

**SUB-VARIATIONS**

2. a. Registration with our software can be done on all published platforms: Android, iOS, web.

3. a. Registration with our software can be done on all published platforms: Android, iOS, web.

4. a. Interaction with our software can be done on all published platforms: Android, iOS, web.

5. a. Interaction with our software can be done on all published platforms: Android, iOS, web.

**RELATED INFORMATION (optional)**

**Priority**: Moderate

**Performance** Target: Minimal, in conjunction with AKP Use Case.

**Frequency**: Many times, daily.

**Channel to primary actor**: Interactive Android, iOS and web applications.

**OPEN ISSUES (optional)**

1. Custom design options.

2. File sharing extensibility (use of and possible editing of shared files).

**SCHEDULE**

Due Date: Release 1.0, January 1, 2015.

* 1. **Use Case: 4 Company Project Dev. Team**

**CHARACTERISTIC INFORMATION**

**Goal in Context:** For the Dev. Team to utilize Globus in-order to keep all members up to date and organized on the teams actions

**Scope**: Company Development Team

**Level**: Summary

**Preconditions**: Team leader is aware of the Globus app.

**Success End Condition:** The team utilizes Globus to help them stay organized and complete their project on time.

**Failed End Condition**: The team does not use Globus and falls into disrepair.

**Primary Actor**: A development team tasked with creating/implementing something for a company.

**Trigger**: The team begins planning how they will communicate between each other.

**MAIN SUCCESS SCENARIO**

1. Team leader suggests to the rest of the team that they all download and use Globus to keep themselves on track and up to date.

2. Team agrees to use Globus, and all members download the app and create accounts.

3. Team leader creates new group, and distributes the group id to all members of the team so they can join.

4. All dev. team members join the group.

5. Team adds deadlines for stages of the project as events in the group calendar.

6. Team sets up Google Drive account for the group for community files.

7. Team begins using the group calendar to plan meetings and to keep track of who all comes to the meetings using the attendance tracker so as to ensure that all members are in the loop and actively participating in the development process.

8. Team uses Whiteboard to inform other team mates of new files on the Google Drive account, how close they are to meeting development milestones, writing inspiring quotes to boost team morale, etc..

9. Team continues to stay on track throughout development process with help from Globus.

10. The team completes the project, and the company is satisfied.

**EXTENSIONS**

7.5 Through looking through the attendance statistics, the team leader discovers one team member has missed many important meetings.

7.5.1 The leader can talk to the team mate in question and discuss why he/she has been missing meetings, and hopefully get the team mate invested in the project again.

7.5.2 If the team mate is still missing meetings, the team leader can show the attendance statistics to higher ups in the company and have the team mate transferred to a different group within the company.

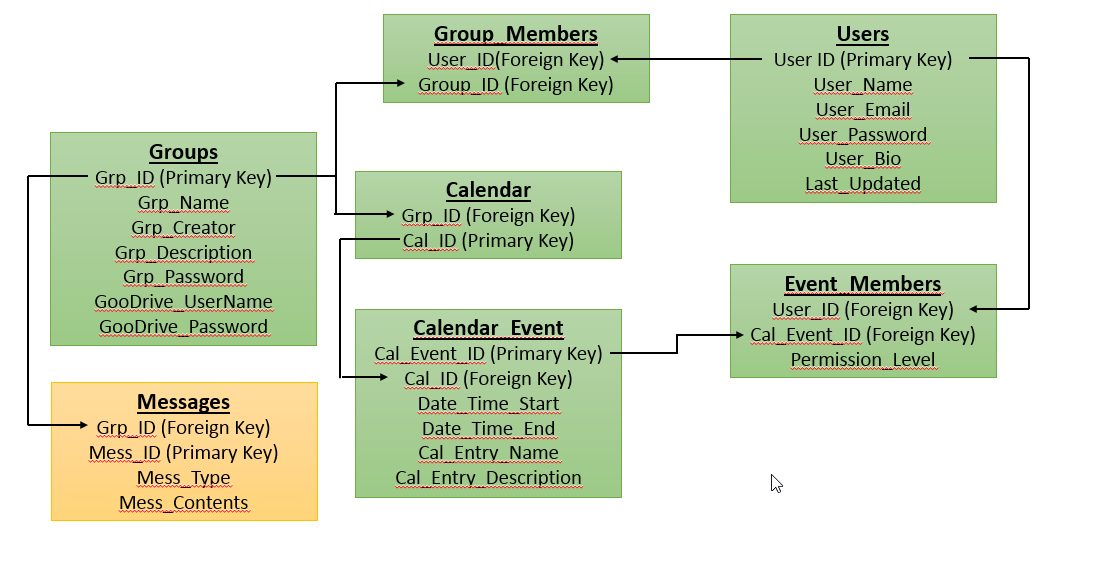
**SCHEDULE**

Due Date: Release 1.0, January 1, 2015.

1. **Plan for Senior Design II** 
   1. iOS App and Web Interface

As mentioned previously, the iOS application will be written. Globus is going to be a cross platform system, and that will allow users to use Android, iOS or a web interface. These will all focus on a really focused user interface that will allow the users to focus on their productivity and content, rather than the technology behind it.

1. **Conclusion**
2. **Images**

Image 1. Database Table Design