Team Cronus - Process Description

* Karthik Palaniappan(karth295)
* Ki Ming (Keith) Yeung(kyeungk)
* Linsen Wu(linsenwu)
* Tiffany Lin(lintiff)
* Micaela Tolliver(mictol42)
* Panji Wisesa(panjiw)
* Tsz Fung Cheung(tfcheung)

**Software Toolset**

**What programming languages, data sources, version control, bug tracking, and other tools will you use? Explain why you chose these tools and languages, as well as why they are suitable for use on this project.**

We decided to make our project a web-based application, because web is cross-platform (i.e. we do not need to worry about iOS vs Android). We also acknowledged that building well-designed UIs (which will be a large factor over our competitors), is easiest with web languages. Finally, these languages generally match our skillset. Here are the tools and the explanations on why they were chosen:

-HTML/CSS/JavaScript: The backbone of the frontend of any web application. We will obviously

be using these heavily.

-JQuery: Is a JavaScript library that will simplify our frontend code—it provides helpful functions

on Ajax requests, basic animation, selecting elements in the DOM, etc.

-AngularJS: Is a MV\* framework for frontend heavy applications. We will use it as both a “frontend

PHP” (e.g. HTML with control flow), and to easily inject the data from the backend.

-Ruby/Rails: We will use Ruby on Rails as our backend framework. The backend will provide the

overall architecture of the site, as well as handle querying the database, and returning

the formatted data to the front end.

-SQL/MySQL: Our data will be very relational, so it makes sense to use a relational database like

MySQL. We will then obviously query it using SQL. The data will solely come from user input, as the application is driven around group task management.

-Git: This will function both as our version control and bug tracking system. As noted in lecture,

Git is a fairly intuitive and popular version control system. Its bug tracking system is visual

and clear.

**What, if any, software components will you attempt to use “off the shelf” versus implementing them from scratch?**

The main UI challenge will likely be implementing the calendar view. We first discussed Google Calendar, but unfortunately, that would allow users to modify their calendar without going through our system. That would immediately render our backend inconsistent with their view, which is a problem we do not want to have. We then came across a JavaScript calendar implementation (http://arshaw.com/fullcalendar/) that allowed more control over user access. We probably will use this as the base of our calendar, but adding in the variety of features we discussed.

Other than that, our frameworks (Angular and Rails) and libraries (JQuery), will help prevent us from needing to reinvent the wheel. After getting past the learning curve, these should dramatically reduce the amount of code we need to write. This will be especially important in synchronizing the backend and frontend. From scratch, we will likely need to design/code the frontend views (login, manage groups, calendar, TODOs, and bills). On the backend, we will need to design the database architecture, and how to query it.

**Group Dynamics**

We have decided on the following roles:

Project Manager: Karthik

Front End: Linsen, Keith, Tiffany

Back End: Micaela, Panji, Fung

Our project manager was selected because Karthik was interested in understanding the integration of front end and back end, he also has some experience with other frameworks and APIs. Also, with his knowledge in a little bit of both sides, we felt that he would be able to help with future decisions. Other roles were chosen based on the previous experiences and knowledge each of our members have. Those in the back end have taken database classes and are familiar with more development languages. Those in the front end have taken web programming classes and have some experience with HTML, CSS, JS and web related skills.

Roles are flexible since members will be expected to support each other. Those who have knowledge in areas can help with coding. And those who end up having less to do for a week can help with researching, teaching, and learning new tools to help with project development.

Based on how our group has currently been cooperating, each of us are very flexible to new ideas and learning. At the beginning of this project, we had critical discussions over what project our group would work on. Each of us were not afraid to make claims for why we felt a project would be better than another. Our problem was deciding on what we would all move forward with, which we have learned from by voting. If a disagreement arises, we all have a chance to discuss our thoughts about the problem. Members are asked for their ideas and are free to input where they see a problem with a current situation, and can make suggestions. We will decide by consensus - we are each flexible and just want to do what’s best for our team. If we absolutely cannot agree on something, we will have a vote and majority wins.

**Schedule / Timeline**

Below is a rough schedule we have put together based on class deadlines and our core features: group management, login, bill/task creation, calendar display, to do list display, and bill display. In blue, are deadlines for our CSE 403 class, and in green are the TeamPlayer deadlines we hope to meet.



We first estimated the amount of time it might take to finish these features, some of which could be grouped together. We realize that we need to organize the front and back end simultaneously, since it is essential that they will need to communicate with each other. We want to know immediately that both ends tie together nicely.

- Group Management/Login = 1 week

This merely stores user ID, email, and group names in a database. The system needs to handle group invitations, which can be as simple as sending out emails. This should take 2-3 days but the setup of a working database may require 3-4 days more.

- Bill/Task Creation = 2 days

Creating bills and task are very similar, we need to create a simple UI for inserting data values into the database. Simple forms and validations are easy to implement so only takes about 2 days.

- Calendar Display = 1-2 weeks

Displaying calendar takes longer because of the complex ways we can put them on the web app with possible features like monthly, weekly, day views and color coding events and tasks. More importantly, we need to figure out how the data will be stored on database to allow scalability.

- To Do List/Bills Display = 1 week?

Displaying to-do list and bills should take about 1 week because it is similar to calendar displaying, retrieving data from the database, displaying it in a simple UI, then filtering by selecting items.

**Risk Summary**

We feel that major risks we will face while completing this project include integration between code, like connecting the front end UI and backend database, managing time to finish all requirements before deadlines, and overestimating our ability to implement certain features. The front end and the back end are being developed separately so we will have to make sure to communicate carefully so the two parts will be easy to combine. The front end javascript has to be able to take into account the SQL code from the beginning so they can be joined.

We are most worried about time management because it is hard to accurately estimate the size of a project which can lead to not having enough time to complete all the features. This is a serious risk because we have a given amount and if we do not finish by then we can’t just extend our deadline. Features relates somewhat to timing in that we are not sure how long each feature will take, and so we need to make good decisions about which features are core and which ones are stretch features. Or else we could make the project too complicated and unfeasible and not finish on time.

To try and reduce risks, we could attempt to build a small sample of the project, like the home screen to see how long it takes. Then with that information we can have an estimate to how long we will need for the whole project. If we cannot overcome a problem, like we are running out of time, we will organize more times for the group to meet so we can meet the deadline. We can also move people around to work on different sections, so if one part is falling behind other people can work on that part instead. If nothing works and we have no time left in the end, we would probably have to end up cutting features.

The points in our process where external user feedback would be most helpful would be after we have our UI and features figured out, and after we have a useable website. At this point feedback would be helpful for finding bugs, flaws, or anything missing that people may find useful.