Updated 6/2 - all changes are highlighted

Jukebox (formerly SharedMusic)

Eric Nhan - ericnhan
Adam Stephenson - jasteph
Keith Yang - keith619
Kevin Fan - kpyfan
Gunnar Onarheim - onarhg
Reggie Jones - reggiej7
Svetlana Grabar - sgrabar
Tanner Coval - tcoval

Updated Schedule / Timeline:

(Note: In the repository, some of the group did not commit under the same user account as GitHub, so please look at the commit history individually instead of on the overviews provided by GitHub)

- A lot of time was spent on getting to learn the languages. (1 week)
- Handled by server
    - User gets identification in a room (3 days)
    - Create and share a room (3 days)
    - Joining a group (2 days)
    - Vote to skip a song (2 days)
- Client
    - User gets identification in a room (2 days)
    - Create and share a room (2 days)
    - Joining a group (1 day)
    - Vote to skip a song (1 day)
    - Search for and add a song to be played (4 days)
    - Play a song (5 days)
    - Music tiles (5 days)
    - Chat with other users (2 days)
    - Invite to room via url link (1 day)
- We probably have one to two more week of testing and bug fixes

Adam and Tanner worked on the initial server architecture (1 week)
Adam completed most of the server (1 week)
Adam and Gunnar worked on testing (1 week)
Kevin worked on the client side communication with the server (2 weeks)
Keith worked on the search functionality (1 week)
Eric worked on the music player module (1 week)

Svetlana worked on the initial UI (1 weeks)
Adam added a music tiles feature (5 days)
Reggie worked on a lot of the testing and management (3 days)
Tanner worked on the bulk of the new UI (1 week)

Process Description

Software Toolset:

We will be using a combination of Javascript libraries, HTML, and CSS for the front end of our product. These are the most commonly used (and arguably the best) languages for the job. We are very likely to be using AngularJS as well as Express. We are also considering using ASP.NET and building CSHTML pages for our front end.

For our backend, we will be using Node.js if we are using Javascript libraries for our front end. Otherwise, we will be using more C# and ASP.NET. There is also a small possibility that we will choose another backend language such as PHP or Ruby on Rails, but it is unlikely.

For our database, we will be using either PostgreSQL if we want a SQL based language, or MongoDB for a NoSQL style language. These were chosen based on experience and the willingness to learn of our group members.

At this point, we are considering two types of stacks - a MEAN stack, which is MongoDB, Express, AngularJS, and Node.js and an ASP.NET based stack, which would use C# and ASP.NET for frontend and backend alongside either a PostgreSQL or MongoDB database. These technologies were chosen primarily based off of the existing experience of group members as well as what people were interested in learning during the course of the project. Either way, we will be utilizing the web APIs for both Facebook and Soundcloud.

For version control, we will be using a public Github repo. This will also host our project website. You can view the website (and see our repository) at http://sharedmusic.github.io/. Github made sense because it is commonly used and most people in our group have at least some experience with Git. It also made hosting web pages incredibly easy for us because of the Github Pages feature.

For tracking tasks and bugs, we will be using Trello. We have chosen this system because it allows us to be very flexible in assigning tasks to people. This is extremely important to our group because of the way we plan on organizing ourselves on each portion of the project. Because the project and the team are both fairly small, we will be using Trello to track bugs as well. We considered using a bug tracking system such as Fogbugz, but we feel that doing so would lead to too much clutter and that it would be too much of a hassle to have to keep track of both systems.

Group Dynamics:

Our overall project manager will be Keith. He was the one who came up with the original idea, and as such, we feel that he has a strong vision of what our product will be. He also has been core in getting meetings and other coordination technologies set up. We feel that he will fit well in a role that helps to drive the group as a whole.

Other members will have flexible roles based on their interest on specific features and roles. Everyone will share in design, development, and testing. We want everyone to be able to work on what interests them and what they think they can contribute the most to. That being said, we can and will assign roles if nobody is willing to pick up a specific slot, though we'd prefer not to. We have chosen these roles because we want everyone to be able to experience each part of developing software. We also want people to work on what they are interested in learning and not feel pigeonholed into something because someone told them to do it.

Roles will vary from part to part in the project. One person will be the team lead for each portion of the project. The team they lead may or may not be the entire project team depending on which portion they are working on. In the case that a part of the project does not require the entire group to be working on it, people will be chosen based on their interest levels.

We generally will resolve any group disputes by bringing it up for discussion at a group meeting. We feel this gives all members a chance to express their views and sway others. If we cannot reach a decision after discussing, we will vote.

We have chosen this structure because it gives people the chance to work on what they are interested in learning and working with while still allowing us to spread work evenly. Because we have such a large group, it will be difficult to communicate and resolve issues individually which is why we want to mainly resolve disputes at group meetings.

Schedule / Timeline:

The first week of the project was getting a sense of what we wanted our project to focus on. In the second week we will make design decisions including which toolsets to use. No sub-groups have been formed yet, but one of the first major components we will focus on is UI. We made some design decisions in a group meeting already but getting a clean and intuitive UI should take around 1 to 2 weeks.

As for development, our current goal is to get all four of our main features completed by week 8 (5 weeks from now). We will add some testing before we start on each feature and add more as we go. Testing will accumulate to around 2 weeks over the quarter.

Our predicted time schedule for each feature is as follows:

- Create and share a room with friends: 2 weeks
- Joining a group: 1 week
- Search for and add a song to be played: 1-2 weeks
- Vote to skip a song: 1 week

Completed use cases

- Search for and add a song to be played
- Handled by server
    - User gets identification in a room
    - Create and share a room
    - Joining a group
    - Vote to skip a song
- Client
    - User gets identification in a room
    - Create and share a room
    - Joining a group
    - Vote to skip a song
    - Chat with other users
- Invite to room via url link

Risk Summary:

One major risk is being able to correctly synchronize the music that a group is listening to. Another risk is making the application simple to use while still keeping it robust and useful. A third risk is the fact that we have to learn new technologies that most of us have very little experience with.

The most serious risks is the learning of new languages. From the research we have done, we have found that one or two of the libraries we are working with are not the easiest to pick up. We have budgeted time for learning the technologies and are confident that the team can pick up the technologies in the allotted time. If we are unable to learn the languages well, it will be difficult to get things done.

One possible experiment we can do is perform usability tests to see which functions and UI designs are best. We can also perform tests to make sure our randomization algorithm is deterministic (makes sure everyone stays synced even though we’re shuffling).

If we realize we are unable to overcome specific challenges, we will try to mitigate the impact as much as possible. If necessary, we will cut or scale down the feature that we cannot complete. To make this decision, we will discuss the individual situation at a group meeting.

The most useful time for feedback would be after our UI design phase. Being able to get external feedback on how we have organized our product and how easy or hard it is to use is extremely important. While our basic feature set is fairly limited (we'd rather have to add to the list than to cut features), we think that it encompasses everything the user needs for the core idea of the project. Hearing from the user about what we've done well and what needs improvement in this will make our product much cleaner.