Teamwork Plan 4.2
Juan Diego Castrillon, Austin Edward Freel, Joshua Daniel Haimson, Ryan Masami Lau

Stakeholders
As this application is intended for use during small gatherings, the stakeholders include the type of people that frequently host or attend small gatherings with friends. We recognize that a gathering can potentially take many forms, and thus that our app might have several use cases, however we are choosing to focus on the experience of a typical college kickback. With this defined, our stakeholders are more specifically college students who host or attend kickbacks, which are generally five to twenty people gatherings.

Resources
We will be using MongoDB for our database and will be hosting on OpenShift to start, so we will be computationally limited by the respective limits of these services. We will also be using the YouTube API to play our music, and so any computational issues involved with streaming videos from YouTube will be unavoidable.

Because these services are free, and since we do not intend to scale our product, we expect there to be no cost involved.

We recognize the time constraint of this project as a restraint on the amount of features that we can implement, but not as a restraint on creating a usable application. We will have all core functionality and a handful of extra features implemented by the time this project is intended to be finished. Because our application has the potential for a lot of extra features, the time constraint will force us to choose the most useful subset of these to implement.

Tasks
Task List
- Host (JuanD)
  a. Create data model; login, logout
  b. Create front end
c. Create skipping ability
d. Update UI/make mobile responsive

- Room/Player (Josh)
  a. Create data model; integrate player
  b. Create front end
  c. Create overlay/private rooms
  d. Update UI/make mobile responsive

- Playlist/Queue (Austin)
  a. Create data model; add/delete songs
  b. Create front end
  c. Create reordering feature
  d. Update UI/make mobile responsive

- Song (Ryan)
  a. Create data model; integrate with Youtube API (search)
  b. Create front end
  c. Create search autocomplete
  d. Update UI/make mobile responsive

Calendar
- Each (a) will be done by 11/16
- Each (b) will be done by 11/18 (MVP deliverable)
- Each (c) will be done 11/25
- Each (d) will be done by 12/2 (Final code deliverable)
- Revisions, updates, etc by 12/7

Risks

Implementation Risks
Using an external API like that of YouTube, we face the risk that the API does not have all of the functionality that we are looking for, or that it has unexpected performance issues. Although there is not much we can do to avoid performance issues, we can mitigate the functionality gap by building more complex abstract methods on top of the basic features available.

The project lends itself to a lot of potentially cool and useful features, so there is the risk that members of our team, or our team as a whole, get sidetracked from the core application and spend too much time on extraneous features. Not only would this be unfortunate because of the time constraint, it would also lead to unnecessary
complexity. We will mitigate this by clearly defining our central goals and, when meeting, talking about what each person has been working on and evaluating what is essential at that time.

**User Experience Risks**

One risk stems from the fact that a specific room will likely be accessed via a code or URL, which means that anyone who attains this piece of information will be able to join the room and add songs. Therefore, there may be internet trolls who access random rooms and ruin the party. Our mitigation will be to allow for rooms to have passwords, so that one needs both the URL and the password in order to access the room, thus enhancing security against outsider access attempts.

Another risk is that there are actually party trolls--people at the party who want to mess with the playlist, either by adding bad songs or an abundance of songs. A mitigation of this is to give the host the power to manage songs, deleting the ones he or she finds unfitting or inappropriate to the occasion.

Another risk stems from the fact that we will be streaming YouTube videos, and thus that there is likely to be advertisements at the beginning of each song. This would be quite disruptive to the flow of the party as it interrupts the music periodically. A mitigation for this is to encourage users to install Ad-Block, specifically the host as the music will be streamed from his or her device.

**Minimum Viable Product**

Our minimal viable product will run the basic core features for our app to be useful. The following subsets outline the components of our MVP at different levels.

- **Concepts**
  - Host → A Host is able to create a Room, manage the Room’s Playlist and Queue, and add songs to both the Playlist and the Queue. The Host is able to manage the Playlist and Queue by deleting any song request. The Host is also able to pause/resume playback of a Room.
  - Guest → A Guest is able to add songs only to the Queue. Does not have to be a user. Simply goes to the Room’s page and adds songs.
  - Room → A Room will contain both a Playlist and a Queue. It is identified by a unique code (String) the Host creates. The Host can add to both lists. The Guests can only add to the Queue. At any point, the Queue
takes priority over the Playlist. The Room’s player will play songs in the Queue if it is not empty. If the Queue is empty, playback will resume from the songs in the Playlist.

In practice, the Host would have selected a list of songs that he/she wants to play throughout the entirety of the party contained in the Room’s Playlist. The Guests, on the other hand, will hear the songs they request relatively quickly because they are only adding to the Queue.

- Queue → The stream of songs added to by the party Guests.
- Playlist → The stream of songs added to by the Host.

- Implementation
  - YouTube → For our MVP we will implement the song choices by YouTube videos. YouTube has an API we can use to play music videos as a free service.

- Authentication
  - Host → A Host must be a user of PandaPlay. They should be able to return to previously created Rooms assigned to their profile
  - Guest → A Guest will not have to have an account. They should be able to add songs to any Room if they have the Room code with the least amount of hurdles.

Issues Postponed (Features we will implement for our final version)

- Security
  - Private Rooms → A Host will be able to create a private Room that will have a Room password. Passwords will keep out trolls who are not in the Room and adding bogus song requests. Increases security.

- Additional Features
  - “Next” Button → Host is able to skip playback. This would be implemented as a “Next” button on playback features.
  - Mobile responsive design → Guest or Host view will be made for mobile. We foresee the Guest and Host to be on his/her mobile phone to access PandaPlay
  - Overlay QR Code → When the Host desktop view is in-active for five minutes, there will be an overlayed message that discourages any Guest from modifying the Host’s setup. This message will contain a quick message encouraging the Guest to access the Room via a QR code on his phone.
  - Host can re-order songs → The Host will be able to drag a song up or down to a different position in the Queue or Playlist. The Host is also
able to move a song request to the top of the Playlist or Queue by selecting a “Move to front” button.

○ Autocomplete Search → After the user types in a few letters, the real-time search pulls up the youtube search of videos.