

## CS 378 - Spring 2017 - Alpha Release

**Due date:** 3/22/17 by 11:59pm, late work is not accepted  
**Points:** 150 points  
**Submit:** A zip file.  
Name your file: **Group<group-number>Alpha.zip**  
Example: Group9Alpha.zip

One team member should submit for the group.

Submission components - all should be included in the zip file:

1. The application project folder.
2. Include in your application project folder:
  1. The **GROUP<group-number>README** file - see below for contents.  
Example: Group9README
  2. Optional: A video showing how to use the application:
    1. Named: **Group<group-number>Video.<extension>**  
Example: Group9Video.mov

**Description:** The alpha version of your project.

**Note:** This version MUST include a standard login screen with username and password fields. You can store the username and password data locally in UserDefaults, and don't need to worry about encrypting the data. The login screen can also include 3rd-party login capability, such as using the Facebook login api.

Include in your **GROUP<group-number>README** file the following sections (see example below):

1. **Implementation Contributions:** A list of the team members that shows each members overall contribution to this release and a list of what features/functions they worked on.
2. **Grading Level:** If the team agrees that all members should receive the same grade for this release, then say "*Same grade for all members*". If not, identify the grade level for each member - 100% for a full grade, 90% for 90% of the full grade, etc.
3. **Differences:** Explanations of any differences between what is submitted for this release and what the App Idea Paper defined as being included in this release, and why that difference exists. If there are no deviances write *None*.
4. **Special Instructions:** Any special instructions needed to make sure your app can be built and run. For example, if you use CocoaPods - the minimum version to use.

Things to keep in mind:

- The app should not crash attempting to use any of its current functionality.
- Wherever data entry is needed, make sure to include code to dismiss the keyboard.
- The UI should be visually consistent - background colors, themes, layout, etc. - all UI screens should look like it was designed/created by the same team and are part of the same app. Step back and ask "would I enjoy working with and looking at this UI?". Ask friends what they think.
- Make using the app as obvious/intuitive as possible. If users don't know what to do next, they'll start trying anything/everything. One way to make it clear(er) is by using movement (animating/throbbing a UI element) or color - to draw the users attention to a specific area of the UI.

Grading criteria:

- Correct file name. (5 points)
- Project builds. (5 points)
- App does not crash. (10 points)
- How much of the defined release deliverables are included. (60 points)
- Quality of the user interface. (40 points)
  - How clean, consistent and visually compelling the user interface is.
  - How reasonably close the user interface matches the mockups.
- Quality of the code implementation. (30 points)
  - How well designed and structured the code is. From simple stuff like good and consistent code indentation and spacing (starting with what's defined in the coding standard), to well thought out and implemented classes, etc.
  - The keyboard should dismiss in all usage scenarios.

GROUP9README example:

***Implementation Contributions:***

Jennifer Lawrence: 10%

- Some of the Login screen

Jonah Hill: 55%

- Majority of the Login screen
- Settings screen
- Learned the Parse API and set up the backend database

Joe Pesci: 35%

- Some of the Table view that shows the list of items
- The detail screen that shows the person details

***Grading Level:***

Jennifer Lawrence: 80%

Jonah Hill: 100%

Joe Pesci: 90%

***Differences:***

None

***Special Instructions:***

None