MOB - Final project Prompt
**to be given on week 1*

1) Project prompt
In a EITHER a self-assigned group of 2-3 OR working solo, work with our resident iOS expert to scope and build an app of your own design. Use the tools we've covered in class to scope, design, build and present your app to the class**.**

2) Goal of project
- Scope a technical project that is doable in a 4-5 week time frame
- Practice agile techniques in a group
- Create an app that works, does what is advertised, doesn't crash, is performant, follows best practices for apps in the App Store
- Communicate an app idea/technical architecture to a technical audience

3) Project requirements
- Submit a final project proposal for instructor approval prior to start of app development
    - Format: One-page proposal should describe and illustrate your app and include:
        - Title
        - App description
        - Rough app requirements
        - Rough wireframes of primary app flow
- Track your progress using stand-ups and an agile task management system
    - Format: Up-to-date Trello board, shared with instructors and TAs, for use by group in tracking progress
- App betas must be distributed weekly to the entire class via TestFlight/Crashlytics
    - Format: Apps distributed and available on TestFlight/Crashlytics website
- Use persistence with files OR core data/sqlite OR interaction with a remote web server/API using AFNetworking
    - Format: Inclusion and use of the above in the final project
- Gather performance and usage data for your app using Mixpanel OR Google Analytics to gain key insights into your apps behavior
    - Format: Functional Google Analytics/Mixpanel dashboard tracking beta users, number of how many times your app has crashed and been opened during betas
- Track crashes with Crashlytics
    - Format: Functional Crashlytics dashboard tracking beta user crashes
- Use Cocoapods to track dependencies
    - Format: Use of Cocoapods in final project
- Use Autolayout either in code or using Interface Builder
    - Format: Use of Autolayout (or Masonry) via APIs or IB
- Use two major non-UIKit iOS frameworks, including:

    - CoreAudio (possible uses: audio apps)
    - CoreMotion (possible uses: fitness apps)
    - CoreText (possible uses: reading/written content apps)
    - EventKit (possible uses: scheduling apps)
    - GameKit (possible uses: game/entertainment/leaderboard apps)
    - MapKit (possible uses: location apps)
    - PassKit (possible uses: ticketing apps)
    - StoreKit (possible uses: e-commerce/digital goods apps)
    - SpriteKit (possible uses: animation/game apps)
    - Or any other non-UIKit iOS framework
    - Format: Inclusion and use of two of the above frameworks in final project

- App must be 'App Store-ready', meaning it should have a title, app store description and screenshots
    - Format: Text file & png files included in final submission

Bonus opportunities:

- App must be accessible
    - Format: Final app must be accessible when iOS accessibility options are enabled
- App must be localized in one real or fake language
    - Format: Final app must be localized when iOS device changed to different language
- App must have a 'debug mode' which assists in logging/debugging
    - Format: App must have debug mode in settings, which outputs other log statements

- App must take advantage of concurrency APIs
    - Format: Use of GCD APIs in final project

4) Deliverables

- One-page proposal of app to be built
- Rough wireframes
- 4 weekly betas delivered via TestFlight/Crashlytics
- Final app project (code, resources, project file, app description, app screenshots) posted on Github
- Presentation of app, app purpose, app architecture to class

5) Timeline:

Week 7: Project groups assembled, one page proposal and rough wireframes complete

Week 8: Designs for app fleshed out, first beta sent out, Crashlytics integrated, home screen UX (first-pass) complete

Week 9: Second beta sent out, home screen prototype plus one additional screen prototype

Week 10: Third beta sent out, home screen plus one additional screen complete, remaining screens prototyped

Week 11: Fourth beta sent out, all screens mocked out, persistence/networking complete

Week 12: App complete, Analytics added

6) Suggested ways to get started

- Idea generation – There are hundreds of thousands of apps out there, so there may already be 'an app for' many of your ideas already. That's certainly OK! Feel free to use existing apps as inspiration for your own, or simply focus on a problem you have that you'd like to see solved with an app. Remember: you're not limited to consumer-facing products; businesses need apps too.
    - Example ideas:
        - To do-list manager
        - Weather
        - Flight tracker
        - News/RSS reader
        - Simple game
        - Alarm clock
        - Reminder
- Product – Start with the problem you're solving, then the people you're solving it for, then the product you're building. What kind of action(s) do you want your users to take? What metrics would make your product 'successful'?
- Wireframes – Start with how people will navigate around your app, and go from there. Will your app use a tab bar? A hamburger menu? A navigation controller? Modal dialogs? Think of the high-level flow of the screens in your app before diving into design details. Make sure to have fairly detailed wireframes for what you want to build ***before*** you start coding.

7) Resources

- Trello (task tracking)
- TestFlight (beta distribution)
- Google Analytics/Mixpanel (app usage tracking)
- Crashlytics (crash tracking)
- Public APIs (Google/4sq/twilio/etc) (frameworks)
- iOS frameworks (frameworks)
- AFNetworking (framework)
- Cocoapods (package management)
- Masonry (layout toolkit)

8) Evaluation
Your planning abilities, project, codebase and presentation will be evaluated using this rubric.

INSTRUCTOR NOTE:

- App Readiness, Execution & Scope are to be graded while ***using the submitted app***
- Presentation & Communication are to be graded during ***student presentation of their app***
- All Technical items are to be graded while ***reviewing the submitted code for the app***