Motivational Model
Features of App

1. Basic run-through of the Business Model portion of Nick Malik's Motivational Model.
    1. http://motivationmodel.com/ebmm/
2. Each page has a Menu linking to HOME, GOTO, SHARE, PRINT, and ABOUT
    1. HOME sends to the Intro page, where a user can change the business name.
    2. GOTO allows the user to auto jump to any part of the business model.
        1. Each link title will have a a green Checkmark if you've completed that section, or a red Exclamation mark if you have not.
    3. SHARE pops up an ActivityViewController to bring open access to AirDrop, LinkedIn and E-mail.
    4. PRINT utilizes AirPrint to connect to an AirPrint-enabled printer and print out the document.
        1. Sharing or Printing with unfinished sections in the Business Model pops up a warning, confirming your decision.
            1. Some of these are not optional, like Value Proposition.
    5. ABOUT switches to a new page that talks about the model, my dad and his business, as well as me and the app.
3. INTRO Page provides an intro (my dad will create that content) and then prompts the user for a business name, and if the Business Name exists, it allows them to press "Go To View Proposition"
4. Adaptable View Template will share the same layout to be applied for 10 different views. (Pending, as new ideas may arise for specific views)
    1. A large field for user input
    2. A description of the section at hand (Value Proposition, Required Competencies, etc)
    3. A "Continue" button, which displays an action sheet (or ActivityViewController if I can customize it) with links to the possible outlets for that specific view.
        1. Each link title will have a green Checkmark if you've completed that section, or a red Exclamation mark if you have not.
    4. A navigation bar displaying a Title, the Menu button, and a Back button, with an abbreviation of the previous view's title.
5. Persistence
    1. Once you leave a scene, minimize the app, or quit the app, it saves the data for the current section using NSKeyedArchiver
    2. If performance is too slow for NSKeyedArchiver, then the data will be saved to a Singleton in memory, and Archived on the background.
        1. The singleton shouldn't remember it's data if the user quits the app. On first load after quitting the app, the app displays an ActivityIndicator to load the NSKeyedArchive data to the Singleton, then from the Singleton to the user. From this point, all data will be manipulated on the Singleton while any new inputs are being stored on the Singleton and queued for NSKeyedArchiver.

3. *[STRETCH GOAL] Use CloudKit to securely backup any information to the Cloud. This allows users to load their data on a different device*

I think this is all of it. I could easily be forgetting some things, however, so bear with me. 😛