Business Plan Guide

Your final app business plan will have ten sections:

1. Product definition statement
2. User definition
3. Market analysis
4. Budget
5. User scenarios
6. App data requirements
7. Milestone list and tasks
8. PERT chart of tasks, milestones, dependences. Development schedule.
9. Future work
10. Fallback strategy

Product definition statement: This is the mantra for your app. It is a single sentence that you can easily recite. It captures the main purpose of the app and who it serves. Successful apps are always designed around benefitting a user. Whenever you are thinking of adding a feature to your app, recalling the product definition statement should help you decide.

User definition: Who is your user? What factors affect whether a user will find your app useful? Age? Gender? Ethnicity? Native language? Religion? Political affiliation? Income level? Occupation? Hobby? Geographic location? Season of the year?

Market analysis: Based on the user definition, research census and other data to develop a numerical estimate of the number of potential users. Look up statistics for app sales, including with various pricing models (free, free with in-app purchases, different price tiers). Check for competing apps. Estimate the likely number of sales (this will be very imprecise, and may be expressed as a range). Cite your data sources and make a reasonable case for your estimate.

Budget: Estimated income is number of sales times price times 0.7 (Apple takes 30%). Expenses include things like equipment (may just be depreciation of your laptop, could be more if you need to host a server), labor (development hours times par rate), advertising (affects sales numbers), taxes (if you strike it rich, assume 35%). Your personal development time is a labor cost. Don't sell yourself cheap -- if your app takes off, you want to pay yourself at a reasonable rate. If you're seeking investment, you have to negotiate this but your proposal should be a starting point for those discussions.

User scenarios: As a result of brainstorming how the app will serve the user, you should have a complete set of scenarios that illustrate its various capabilities. These should be described with a series of screen shots of UI mockups that include a narrative of what they each represent. You can use XCode to do the mockups, or look for one of many prototyping tools, such as BluePrint or AppCooker (which also has a market estimator).

App data requirements: This is an incredibly important element of planning an app. Many app ideas crash and burn when they run into the problem of populating the app with data, so it's worthwhile to think about it early. It can take a lot of work to build a useful data set (Google spent millions creating the data for Maps) and that can actually be the main value provided by an app. The data requirements could also include a complex mathematical model (e.g., a satellite orbit). Another aspect is how the data set will be maintained. If you have to manually enter the training class schedule for the fitness center each week from a printed handout, it's not a sustainable service. You should identify the data the app will use, where it will come from, and how it will be stored and maintained.

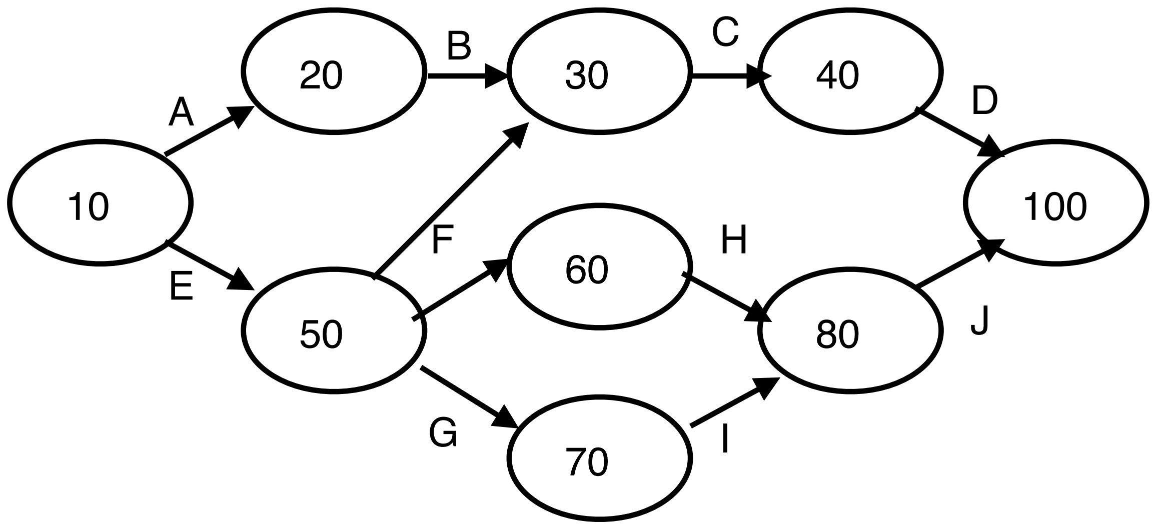
Milestone list and tasks: This is where you plan the process of building the app. A milestone is the completion of a key component or the demonstration of a significant capability. For example: UI implemented, backend server connection working, primary user scenario functional, etc. Each milestone has a set of tasks that need to be done. For example, getting the backend server connection working could include setting up the host services, testing them from a laptop, building the network interface for the app, pushing test packets through the network interface to and from the server, and having a set of app test stubs that exercise all server communication operations. Some tasks may support multiple milestones. You should create a numbered list of milestones, and for each one indicate the task(s) it depends on. You should have a separate lettered list of tasks so that you can just list task letters next to milestones.

PERT chart: This is a graphical representation of your milestone and task lists. Each milestone is a circle with the milestone number inside it. The circles are connected by arrows that are labeled with task letters. The connections indicate dependences. For example, starting from Milestone 10, task A needs to be completed to achieve Milestone 30. In addition to graphically representing the milestone/task relationship, each arrow is further labeled with a time estimate for the task (in this case, programmer hours). From this representation you should be able to identify critical paths (the chains of arrows with the longest total time). That will help you focus your efforts most effectively. Based on the time estimates, you should develop a schedule of deliverable dates for each milestone so that you can track your progress and see if you are in danger of missing deadlines.

Future work: You should always have some features that you are looking forward to incorporating in a future release. They help set a direction for initial work, and if you find yourself ahead of schedule you can add them in. List these in this section.

Fallback strategy: Sometimes things go wrong and you need to sacrifice certain aspects of functionality to make a release deadline. It is best if you plan this in advance so that you can shed workload in a timely manner and avoid wasted work. You can also strategize the levels of functionality that will maintain a useful and coherent app. Describe a series of fallback positions that you can resort to depending on progress.

Example PERT Chart



Milestone	Date	Description
10	9/23	Business plan complete, UI mockup done, tasks assigned
20	10/2	Top level view working with stubs for lower views
30	10/15	User login/account creation view works, data entry works
40	11/1	All views work with test database
50	9/30	User login/account portion of database working
60	10/20	Database data entry operational for test database
70	10/25	Data editing working
80	11/15	Backend management of database working
100	11/20	UI and live database integrated and tested

Task	Time (Hours)	Description
A	2	Create top level view and test framework
B	5	Build account creation view and JSON connection to DB
C	10	Create data entry, editing, browsing views and tests
D	6	Link UI to live database, begin populating it
E	5	Create database with all necessary fields, test with driver
F	7	Build login/account creation scripts, JSON interface
G	5	Build editing and search scripts for database
H	6	Create secure link for admin access to account data
I	6	Set up remote admin view/management for database
J	8	Build out test database, test with UI, set up live DB