**Phase 0**

These are the five ideas that I brainstormed for my Android Application.

1. Budgeting/personal finance organization app
2. Fitness/nutrition management app
3. Graphing calculator/matrix calculation environment
4. Genera task scheduler/organizer; planner
5. IMDb-based movie recommender

**Phase 1**

My final app idea that I am choosing to pursue is that of the budgeting/personal finance organization application. Upon brainstorming ideas, I asked myself, “what is it that I personally need the most in my life that an Android app could help fix?” I eventually concluded that I need a personal budget and an easy way to manage my finances. I deemed this the most practical of all my app ideas because it is something that I (and therefore possibly many other college-aged students) do not have but could very well use in my day-to-day life. The void I find in my personal life will be the driving force and my source of motivation of the development of this application for both myself and other users.

The intended audience of my budgeting app is college students and recent college graduates, though I’m sure high school students as well as older audiences would easily be able to use it (though not necessarily need to utilize all of its features). I plan to implement features that college-aged people will generally have a use for, such as weekly bill-planning and loan management.

Here is an overall list of features that I’d like to implement:

* Bill planning
* Weekly spending tracker
* Loan management
* Investment future mapping

The bill planning and weekly spending tracking would be integrated together to make a week-by-week budgeting tool. This budgeting tool will be the main feature/focus of the app. By inputting an amount of money that is counted as a paycheck, adding in bills that are due that week, and adding in discretionary purchases, a budget is generated. The week’s bills and paychecks can be added at any time during the week, but adding them at the beginning of the week is advantageous because it shows how much money is left over for the later-added discretionary purchases. Also, anything that is not specifically classified as a “bill” or a “paycheck” will either be an “additional money in” or a “discretionary purchase” (depending on if the “transaction” *raises* or *reduces* the amount of money after bills and paychecks, respectively).

Loan management will be built into the weekly budget by classifying loan bills as standard bills with a “loan” option checkbox checked (this would be shown in the area where a purchase is added). When the loan checkbox is ticked, the app will ask for an input of a number indicating the loan’s APR interest rate, and then it will produce a table indicating how long it would take to completely pay off that particular loan with that particular interest rate (continuously compounded), if one pays that same payment every month until finishing off the loan. The table will have another part where it shows the minimum monthly payment needed to finish off the loan in a year of monthly payments.

Investment future mapping will be implemented by showing the power of compound interest over time. It will show, in a table (or similar structure), how much any investment will grow after some time. It will have three input fields: principal investment, weekly addition, and anticipated annual growth rate. From this, the app will crunch some numbers to show how much the investment will be worth after a single month, 6 months, 1 year, 2 years, 5 years, 10 years, and 20 years (given that the weekly contributions and growth rate remain completely constant, and that no money is pulled out of it).

The general layout of the app will be similar to this: the main activity that would open upon a fresh launch of the app would be the weekly budget. From the top left menu-navigation button, a menu can be accessed where the user will be able to choose to either manage their budget for the week, use the investment mapper, or alter settings for the app. The budgeting tool will allow users to change which week is being viewed by use of arrows on the left or right (to go backward or forward in time, respectively) near the top of the screen. Also near the top of the screen will be a button to add a “money in” or a “money out”, from each of which one can press a button determining the type of such transaction, and each transaction then will have a small form to fill out regarding the transaction’s information (amount, if it’s a loan, label, any user-generated tags). Below this area there will be a list-esque view showing some basic calculations with the money in and money out (with the transactions’ labels appearing directly next to each transaction), with a total “money remaining” quantity at the bottom (this amount will also be displayed near the top of the screen, for visibility). Near this final “money remaining” amount, there will be a few other useful tips: an amount of transactions at $20 each that the user can perform before going broke, and how much the user can spend on each of 10 individual purchases (on average) before going broke. Basically, these two numbers are supposed to make it easy to see what freedoms the user has with their money; the smaller those numbers are, the less they can spend throughout the rest of the week.

I think that this app would be interesting and useful because it would give college-aged students a concrete method of managing their finances, which many of them do not currently do (or at least, do not currently do *well*). By managing their finances responsibly, users can take control of their monetary lives in order to benefit their immediate and far futures immensely.

I am confident that I can complete this project in the 2.5 weeks allotted because I am decently skilled at designing finance-oriented mathematical functions based on initial conditions and goals. My biggest challenge will be learning more about the Java programming language and how to implement it in the application itself, but that should be no problem: I am generally familiar with the logic of computer programming which (more or less) transfers between mutually exclusive coding languages (syntax and some core ideas, however, do *not* always transfer). Because of this, my app will be done on time, fully-featured.