Feasibility Study: FundSave

Feasibility Study

FundSave

Alok Mantri, Jordan Lin, Yuriy Neyra Alvarado

Date: August 19th, 2025

Problem Definition

Problem Definition

FundSave is a Java application that aims to combine three key capabilities: displaying and searching for an individual's current investment options, such as stocks, bonds, cryptocurrencies, and GICs. It lets users track budgets and savings, and it shows their overall financial status. User data, expenses, and investments can be stored using the Java concepts we learned in class, like file I/O, so that information can be saved between sessions. The program also utilises a Swing GUI, which aids in keeping the Java Application user-friendly and error-free.

The main purpose of *FundSave* is to make financial literacy easier for young adults. Most finance apps are too simple or too complex for many who need to practice their learnt skills, so this Java application aims to give a tool to young adults and students alike that is accessible, realistic, and most importantly, educational.

Names: Alok Mantri, Jordan Lin, Yuriy Neyra Alvarado **Date**: 8/19/2025

Problem Analysis

Problem Analysis

When creating the idea of *FundSave*, several challenges and limitations are essential to consider. The Java app will require a computer that can run Java programs smoothly and efficiently. Since this is a high school project, therefore considered a public use application, and there is no major financial cost involved. If this weren't the case, however, expenses such as hosting or external services required to manage the Java application full-time could serve as the costs. The program will use local file storage for this high school project, which will avoid the need for costly databases or expensive servers.

Some research is required, such as Java concepts like Swing GUI, inheritance, polymorphism, file I/O concepts, and searching/sorting algorithms. Time management will also be an issue as this Java application is due within 3 days of work, so the features must be realistic and also divided fairly among our team. We will focus on completing essential features and tasks first, in order to ensure the program is functional for the final day of testing and evaluations.

Software Project Plan

Statement Of Work :

Our team will build *FundSave* by dividing the project into smaller tasks:

1. Creating classes for investments, stocks, crypto, and ETFs
2. Building a budget and saving tracker.
3. Displaying the user's financial information.
4. Search functionality for investments
5. Swing GUI with input validation

Algorithms and some pseudocode will be created to show how searching, sorting, and saving will work. A UML diagram will be designed to plan out the program’s classes and attributes.

Resource List :

- Team members with computers that can efficiently run Java applications.
- Time and skills required to code and create various documents.
- Online tutorials for Java Swing GUI, file I/O, and sorting/searching.
- Brightspace lessons for previously learnt Java lessons.

Names: Alok Mantri, Jordan Lin, Yuriy Neyra Alvarado **Date**: 8/19/2025

Work breakdown and time estimates :

- Investment classes and inheritance structure: 2 hours
- Budget/savings tracker with file I/O: 2-3 hours
- Search feature with binary/recursion search: 2 hours
- GUI screens + error checking: 5 hours
- Testing and final debugging: 4-5 hours

Project Schedule :

The tasks will be scheduled in order using a Gantt chart to show when each of the tasks will be worked on. The critical task is finishing the data storage and search features before moving on to the GUI, as we can simply call the classes. In order to mitigate the event that we run out of time, our team will ensure that the basic features are complete and functional. These include the budget tracker, search, and display. With hard work and dedication, our team remains positive about completing this project!

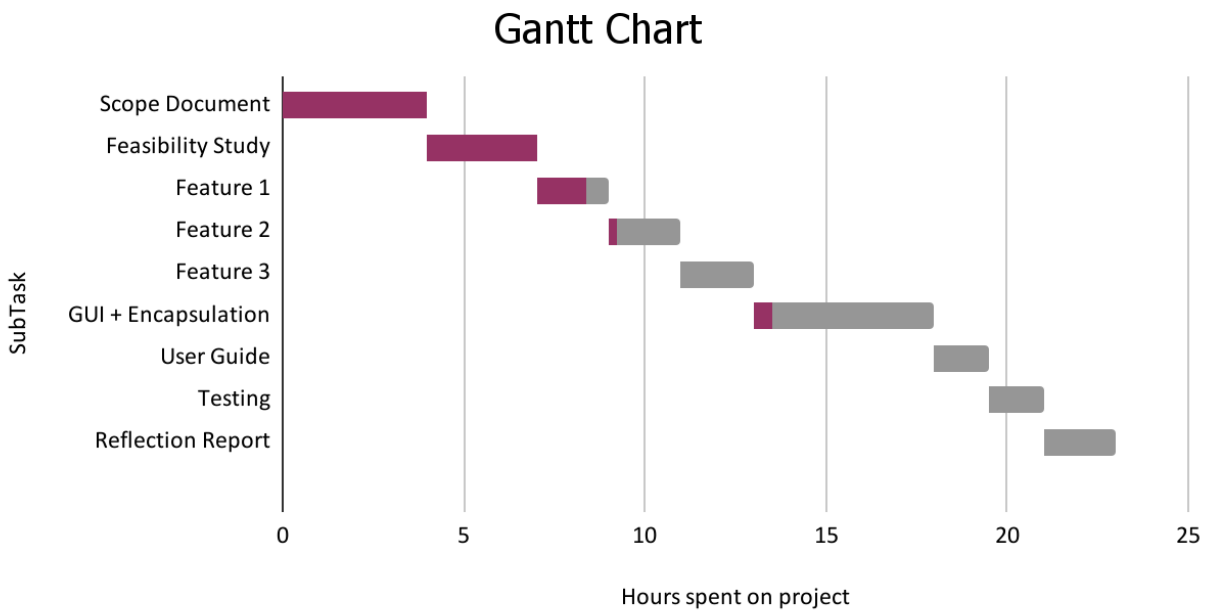

Chart 1: Gantt Chart at the time of submitting this Feasibility Study

Names: Alok Mantri, Jordan Lin, Yuriy Neyra Alvarado **Date**: 8/19/2025

SWOT Analysis for *FundSave*

1. **Strengths**

FundSave has several strengths that make it stand apart from its competitors. The program is designed specifically for young adults, which makes it extremely practical and reliable for students learning about budgeting and investments. The team (Alok, Jordan, Yuriy) has excellent coding skills in Java, including file I/O, arrays, recursion, and object-oriented programming. These highlight our great technical skills. Another strength is that *FundSave* utilises a user-friendly Swing GUI with input validation, so users can interact with the program easily without making errors.

2. **Weaknesses**

A major weakness that bottlenecks the development process of *FundSave* is the time allotted, since we only have a few days to finish and test it. This means advanced features like live stock market data or simulations of the stock market analysis cannot be included as mentioned within the Scope Document - FundSave. Another weakness is that the team is learning new concepts, such as Swing GUI and saving files between different instances of the Java application being run using file I/O.

3. **Opportunities**

There are many strong opportunities for *FundSave* as financial literacy continues to be a growing need and skill to master amongst many young adults. Many people struggle with budgeting and investments, so a tool like this could be valuable for students or beginner investors who want to understand different financial instruments. The project also gives us the opportunity to apply the programming skills we've learnt throughout this course, like inheritance, polymorphism, and file management, in a real-world context, strengthening our skills and preparing ourselves for university courses and future jobs. Another opportunity is that if the program works well, it could be expanded later with features like the stock simulators. This would make the program more competitive with professional financial institutions.

4. **Threats**

The main threat if this program were to be made outside of our course restrictions would be the competition of existing financial apps, which are already well established. Even if *FundSave* is great for students, it cannot match the resources and features of these professional applications. This, however, can be mitigated by various forms of funding, so our team has the tools and resources to develop and grow the application. External factors like new software requirements, lack of API access, or unexpected technical errors can affect the success of the project.

Names: Alok Mantri, Jordan Lin, Yuriy Neyra Alvarado **Date**: 8/19/2025

Software Testing Life Cycle

1. **Requirement Analysis**

For *FundSave*, the main requirements are displaying and searching investment options, tracking budgets and savings, and showing financial information. From the scope document, we know the program must use file I/O to save and load data, have a Swing GUI, and apply the Java concepts we learnt in the course, like inheritance and recursion. At this stage, we can check that each feature is testable, such as searching for a stock by CUSIP number or saving the user data correctly.

2. **Test Planning**

In test planning, we decide how to test *FundSave* within the limited time of a few days. This will cover testing the budget tracker, search feature, and financial display, since those are the core functions. Our resources include our computers, Java IDEs, and the group members' time. Risks include running out of time or encountering errors in file handling. To reduce these risks, we will test smaller parts earlier and fix issues as we make progress coding. The output will be a simple test plan that lists the objects, responsibilities, and timelines we have to finish.

3. **Test Case Design / Test Development**

Our team can create test cases for all major features. For example, we will test what happens when a user enters a valid stock CUSIP number and also when they enter an invalid one. For the budget tracker, our team can test adding, deleting, and saving budgets. Each test case will include the inputs, the expected result, and what the final result will look like.

4. **Test Environment Setup**

Our test environment setup is the Java IDE we use (CodeHS), along with the Swing GUI screens we design. The environment will be set up on our computers, which will allow us to run and debug our program. We could also prepare input files to test saving and loading data, with the goal being to understanding how the program would run from the user's perspective.

5. **Test Execution**

At this stage, we will run our test cases on *FundSave*, such as entering the stock codes (CUSIP number) to check if they match our expectations. We will also test the budget tracker by saving information, closing and reopening the Java application to see if the data loads correctly.

6. **Defect Reporting & Tracking**

If a feature does not work as planned, such as a file not saving correctly or invalid input not being handled, we will ensure it is fixed. Each defect will be mitigated as we will assign it to a team member to fix. Once fixed, we will retest the feature to make sure the problem is solved. This process continues until the program is stable and reliable.

7. **Test Closure**

When all tests are finished, we will review whether every planned feature was tested and whether the important defects were fixed. Some advanced features, like live stock prices, will not be included, so they will not be tested. This will complete the testing cycle and prepare *FundSave* so we can submit it on time.

Conclusion / Summary

Conclusion :

The *FundSave* project is a practical yet realistic Java application designed to help many young adults and students manage their finances more effectively while applying the programming skills we have learnt in class. By combining investment searching, budget tracking, and financial display features, the program addresses the issue many face when they try to practically apply the financial skills they've learnt. The project is divided into manageable tasks and is supported by a clear breakdown of the work required and how we tackle each problem that may occur. A resource list and a Gantt chart schedule are included as well, which keep the development process organised and achievable.

The feasibility study shows the program is both possible and valuable within the 3-day time frame. The SWOT analysis highlights how *FundSave's* strengths include its educational value, technical design, and user-friendly interface. We also mentioned weaknesses and threats with mitigation strategies, as well as future opportunities if this project were made outside of this course.

The Software Testing Life Cycle is about how the program is properly tested for accuracy, reliability, and user experience. Each stage, from requirement analysis to test closure, focuses on making sure that our features are functional and that any defects will be fixed before submission.

With careful time management, group coordination, and dedication, our team is extremely confident that *FundSave* will be completed successfully and meet the expectations of the culminating project.