

# Piggy \$aver

Managing Your Money Made Ea\$ier

7<sup>th</sup> November 2020

## Members: WalletSaver

1. Thang Le ([thangle2@bu.edu](mailto:thangle2@bu.edu))
2. Zining Ye ([ziningye@bu.edu](mailto:ziningye@bu.edu))
3. Radhey Patel ([rexu6@bu.edu](mailto:rexu6@bu.edu))
4. Bryan Jaimes ([bjaimes@bu.edu](mailto:bjaimes@bu.edu))

## Links

1. [Github](#)
2. [Youtube](#)



## Overview

Every January, a new survey presents the same data as the last one, stating "[x% of Americans Living Paycheck-to-Paycheck](#)." In 2019-20, the numbers reached a high of 49-59%, with [62% of households reporting no savings/emergency funds](#). These seemingly inconspicuous numbers turned into reality for Federal workers during the government shutdown of 2019. Meanwhile, the impact has stringently grown for workers in all sectors during the pandemic. The best solution is to make more money, save more, live below your means, but that's easier said than done. With the increasing uncertainty of financial security, managing money has become a necessity. However, starting the habit of budgeting early on has been troublesome for young-adults due to limited to no personal finance education; this is true for even recent graduates joining the workforce.

Our App makes it easier for its users to take control of your finances and gets the user started with a simple plan for the users to manage their cash flow. Piggy \$aver resolves this by letting the users take a survey, instead of the hassle for one to set up a complicated spreadsheet. The App assesses and allocates funds for the necessities (like emergency fund/savings) over the superfluous expenditure, focusing on subscriptions. Our aim with this App is to reduce the barrier of entry for anyone attempting to habitually budget. In the day and age of 10-second TikTok clips and 10-minute Youtube videos, the [Minimum Viable Effort](#) for Piggy \$aver is a 5-minute survey, making it easier for the user to assess their financial health -- "You can fix what you can see." Users will be warned of overspending on subscriptions and would be presented with the percentage of overspending in that category. The aim to focus on subscriptions is to counteract the strategy of subscriptions directly- a reminder/ visual access to it makes an active attempt for the users to not avoid renewals. Rather than letting another year's resolution fail, the User can now strive for a new habit with a little assistance!

## Marketability

### Target Market Demographics

Age Range: 18-26;

Gender: Irrelevant;

Location: North America (Currently, in the U.S.A.);

Geographical Positioning: Urban Market;

Level of Education(Current): High School (Upper Class), College (Undergraduate)

Marital Status: Single, In-relationship without current legal affiliation (unmarried);

Occupation: Urban White / Pink / Blue / Grey / Green Collar Jobs;

Religious Affiliation: N/A;

Ethnic Background/Race: N/A;

Lifestyle, Social Class: Middle-to-Upper Middle Income.

## Why Piggy \$aver over the competition?

In the current marketplace, there are three competitors: Mint, Personal Capital, and, the classic, Spreadsheets. Our App focuses on streamlining setting up a budget into a habit by simplifying the user inputs to its bare minimum- whereas, Mint requires mandatory connections with a specified bank account (For limited banks) available in their network and does not assess the subscription spending or other additional expenses. Meanwhile, Personal Capital assesses and organizes the retirement and savings capital of individuals, but the target market for our App is avoidant of retirement savings at the age range. Spreadsheets are the reason people are afraid to start to budget. Though it would be customizable, Spreadsheets have an inherently higher threshold to form a habit, let alone be visually accessible weekly.

## Description: Processing

When the user opens the app for the first time, they will be given a survey to complete that asks for their basic personal information (such as their name, age, and state they live in), and information about their monthly income and expenses. All of the personal information is initialized in the "User" class, which contains a default constructor, and get and set functions for each variable. In the "Financials" class, which inherits from the "User" class, there are variables for monthly income, basic monthly as well as weekly expenses, and HashMaps for subscriptions, investments, and bills. HashMaps are applied to store an organized list of items that include name and cost/amount. A TreeMap is implemented to track the user's weekly spending because it is a more convenient way to store this data in chronological order.

```

18  /serial/
19  public class Financials extends User implements Serializable {
20      protected float monthlyIncome;           //monthly Income post taxation (payroll)
21      protected float savings;                 //Personal Savings
22      protected float weeklyGroceries;         //Groceries Expenditure (Weekly/not Monthly)
23      protected float transportation;          //Transportation Cost
24      protected float monthlybudget;           //Monthly Budget for calculation
25      protected float weeklybudget;            //Weekly Budget for calculation
26      protected Date startday;                  //
27      protected Date currentday;               //
28      protected long howmanydays;              //
29      protected int spendingtoomuch;           //
30      protected float weeklysavings;           //
31      HashMap<String, Float> subscription = new HashMap<String, Float>(); //Repeating Subscription costs
32      HashMap<String, Float> investment = new HashMap<String, Float>(); //Individuals' Investment (not savings) monthly
33      HashMap<String, Float> bills = new HashMap<String, Float>();
34      TreeMap<String, Float> weeklySpending = new TreeMap<String, Float>();
35      List<TreeMap<String, Float>> allWeeklySpending = new ArrayList<TreeMap<String, Float>>();

```

After completing the survey, the user gets taken to their home page, where they can see their calculated monthly budget, which is their monthly income minus a summation of all of their expenses in every category. The user has the option to change/add any expenses on the settings page. On the home screen, the user can add any expense of any kind, and the amount they spend per week will be subtracted from their calculated weekly budget. There will be messages to warn the user that they are getting close to, or have surpassed their monthly budget. Once an expense is added, the home screen will be updated with a list of the last 15 expenses made. To show this, a for-loop loops through the user's weekly spending TreeMap, and creates a string for each item.

```

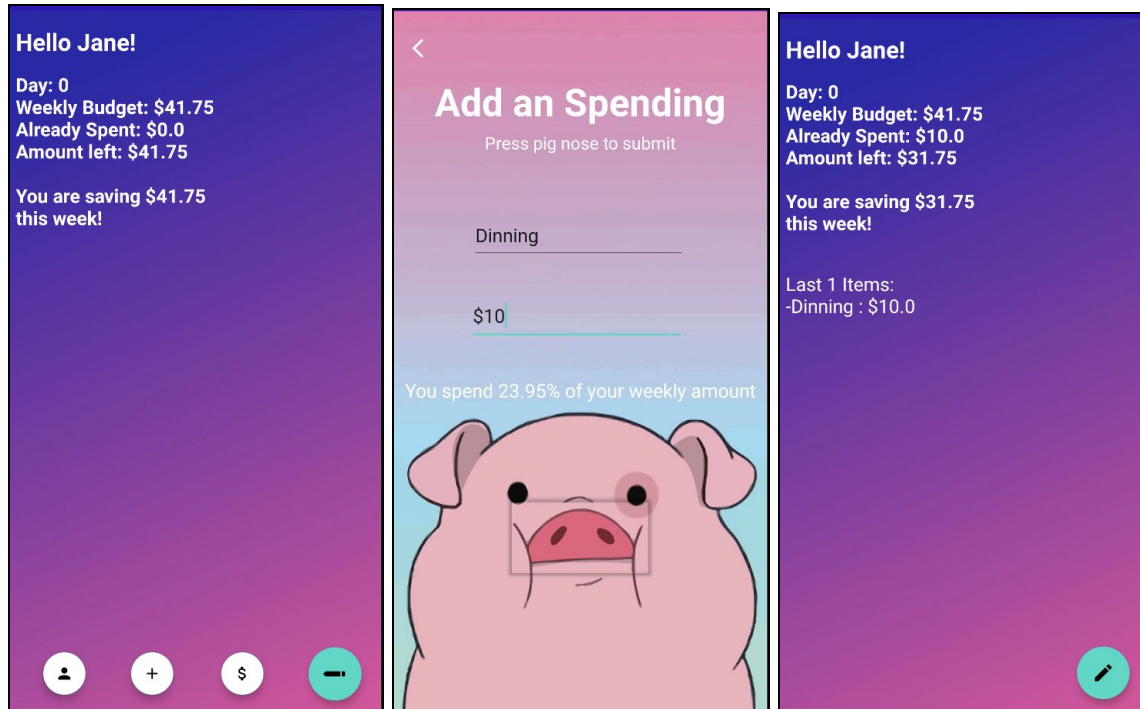
96  @Override
97  public void run() {
98      if (originaluser.weeklySpending.size() > 0) {
99          if (originaluser.weeklySpending.size() > 15) {
100             int counter = 0;
101             result = "\nLast 15 Items:\n";
102             for (TreeMap.Entry<String, Float> entry : originaluser.weeklySpending.entrySet()) {
103                 if (counter >= originaluser.weeklySpending.size() - 15) {
104                     String name = (String) entry.getKey();
105                     Float value = (Float) entry.getValue();
106                     result = result + "-" + name + " : $" + Float.toString(value) + "\n";
107                 }
108                 counter++;
109             }
110         }
111         totalamount.setCharacterDelay(35);
112         totalamount.setText("");
113         totalamount.setText(result);
114     }
115     else {
116         result = "\nLast " + Integer.toString(originaluser.weeklySpending.size()) + " Items:\n";
117         for (TreeMap.Entry<String, Float> entry : originaluser.weeklySpending.entrySet()) {
118             String name = (String) entry.getKey();
119             Float value = (Float) entry.getValue();
120             result = result + "-" + name + " : $" + Float.toString(round(value, decimalPlace: 2)) + "\n";
121         }
122     }
123 }

```

## Description: GUI

The survey for the new user is made up of seven different screens. After the user has entered input, they can click on "Next", to get taken to the next screen. If the user does not enter a valid input, they will be notified and asked to re-enter an input. For the screens where the user is able to input multiple entries, they will be able to see the name and amount for the last expense they input. The user can keep adding entries while remaining on the same screen, and when they are finished can click on "Next" to continue.

After completing the survey, they will see a summary of their subscriptions, and then get taken to their home page. On the home page, they can click on the pencil icon in the lower right corner of the screen to open new animated icons and navigate to the other pages: user profile, add a spending, and weekly summary. When adding a new expense, the user will be taken to a new page similar to the one from the survey, and be able to input the name and amount of their new expense.



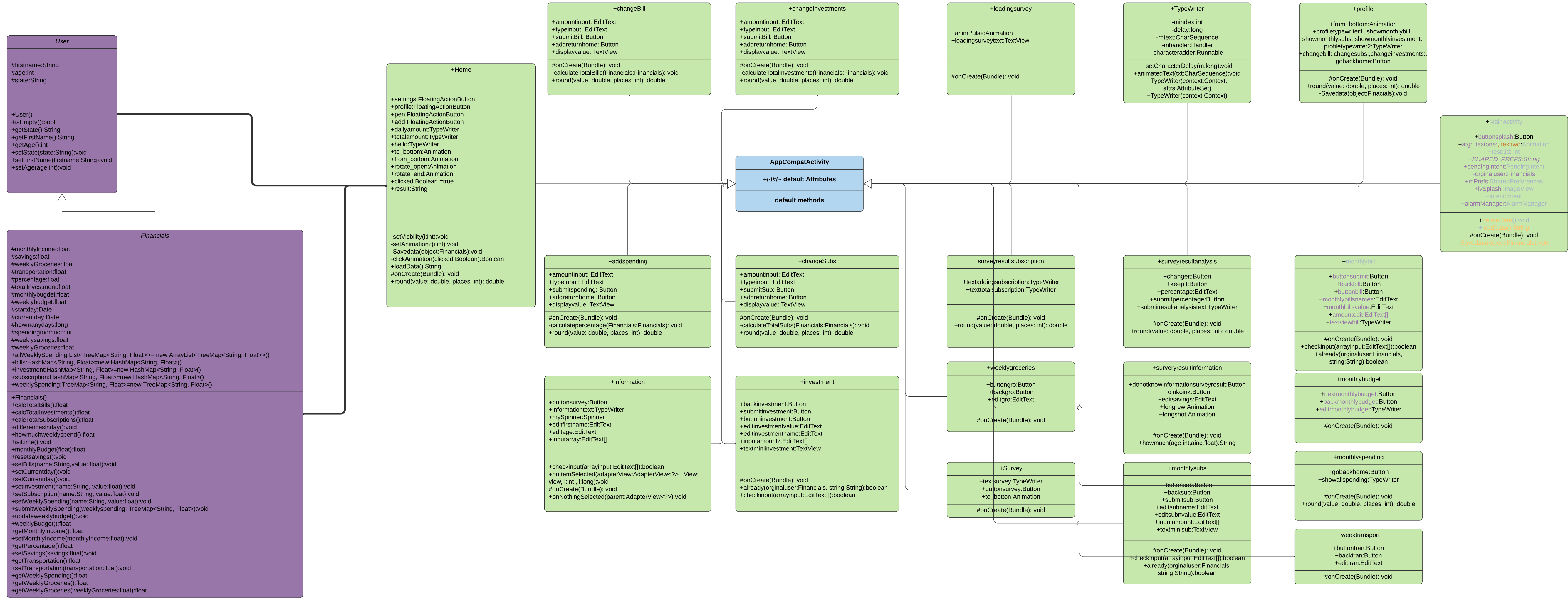
## Architecture

The App is constructed based on Object- Oriented Programming Paradigm; opting for such structure enables the development team to control modularity of its functionality as well as encapsulates data from being altered in any state or form directly. Following is the attached UML Class architecture for our current application:



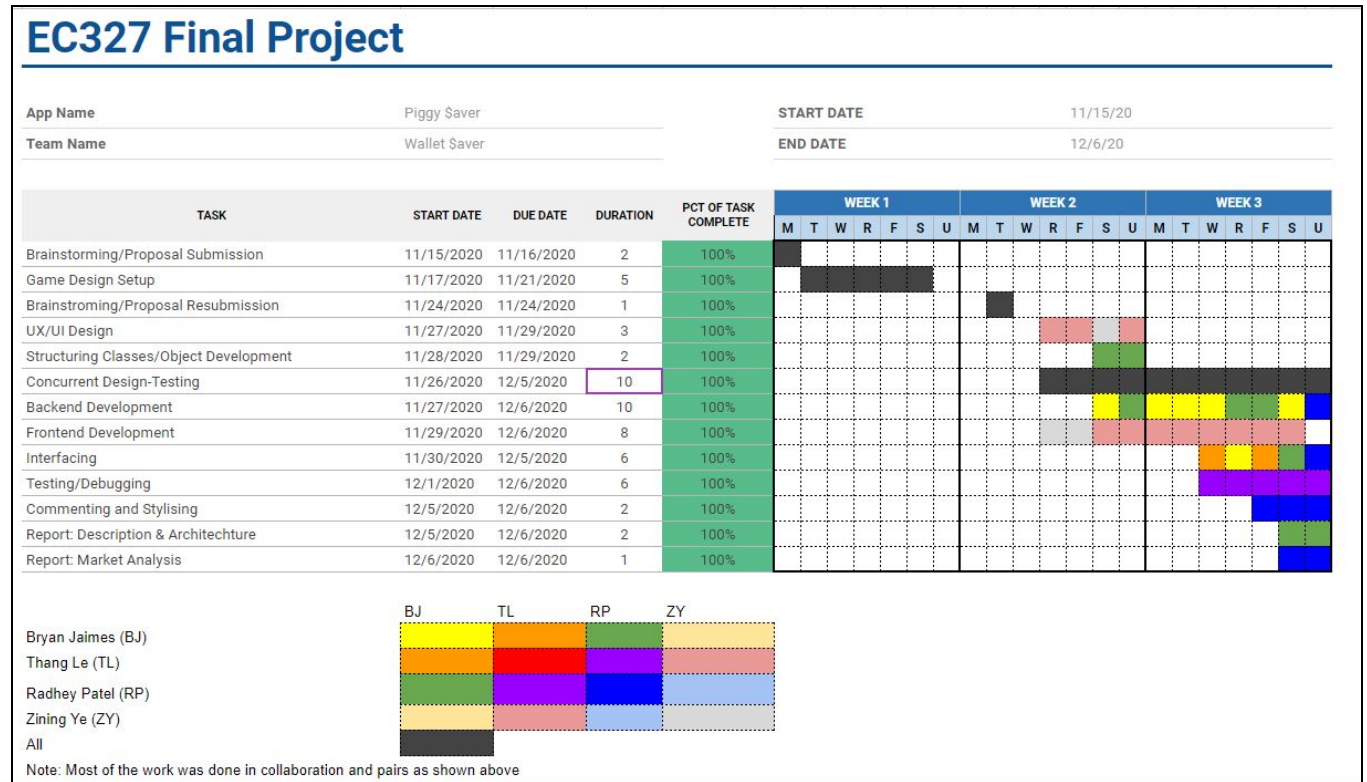
## Class diagram with UML Formating

Radhey Patel | December 7, 2020



## Timeline

## Gantt Chart



### Team Work-Distribution:

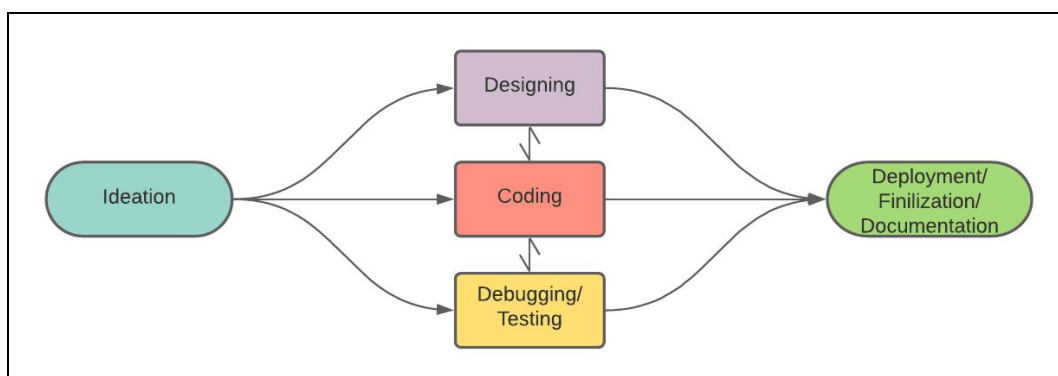
Thang Le: 40%


Bryan Jaimes: 20%

Zining Ye: 20%

Radhey Patel: 20%

## Workflow Method & Implementation





Our Workflow was based on the Concurrent Design Flow to manage our expected outcome within the time constraints and scope of the project. This was practiced as to prevent overburdening the workload of the team while simultaneously staying focused on the core functionality of the Minimum Viable Product. Though not using Agile Development methods completely, we incorporated some elements of iterative development throughout the process.