

Retirement Planning Tool

CS 787 - 001

Saarika, Vidya, Bruce, Shravani, Shilpa and Gargi



Roadmap

- Concept
- System features
- Approach to building system and model
- The Model
- Tools Used
- Demo
- Challenges and Limitations
- Next Steps

Imagine Retirement

When do you want to retire?

What do you want to do when you retire?

How much money will you need?



Challenges

Reliable financial or retirement planning advice

Inflation

Rising and unknown cost of healthcare

Market conditions



Considering Risk

Individual Stocks

Options

Crypto

ForEx

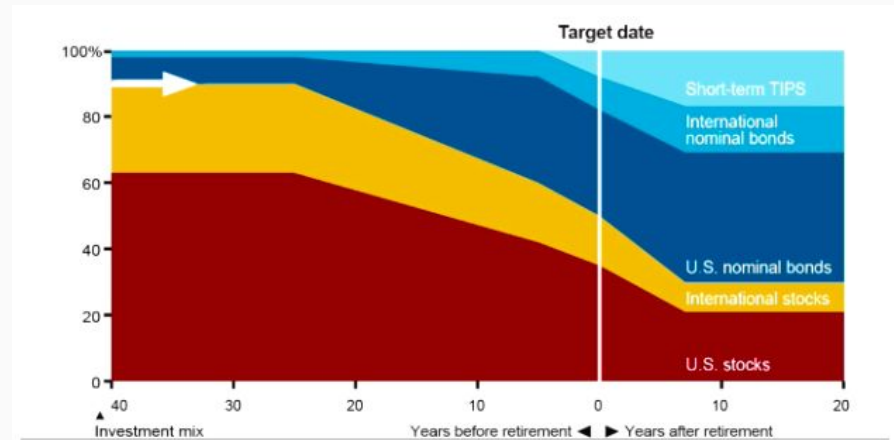


source : <https://www.coinbase.com/price/bitcoin>

Asset Allocation

Target date funds:

Ratio of stocks to bonds decreases as retirement date approaches, as indicated by the “glide path”



Vanguard Target Retirement Fund Glide Path

Source: <https://www.whitecoatinvestor.com/designing-your-glide-path/>

Asset Allocation and Risk

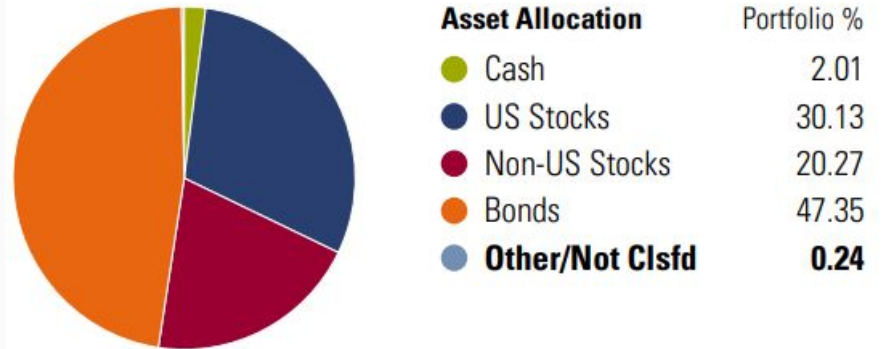
Considerations:

Personal risk tolerance

Time to Retirement

Spending/Asset Ratio

Asset Allocation, year before retirement



Source:

<https://seekingalpha.com/article/4320465-retirement-target-date-allocation-glide-path-in-depth-view>

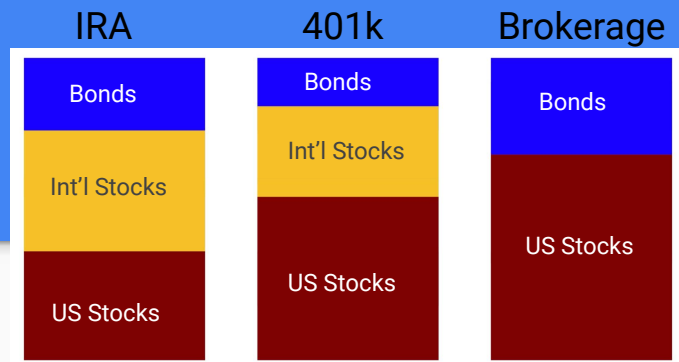
Concept

Retirement Calculator

Three fund portfolio ([38] in "CS 787 Project Citations" doc)

Main Retirement Income Sources ([1]): IRAs ([4]), 401ks ([10,11,13]), SS ([5,6]), Pensions ([9],[10]), equity in home ([2,3,7,8])

Compound interest - the earlier you make the changes, the better



Approach and System Features

Other calculators we modeled our system after:

- Brighthouse Financial Annuity Income Calculator([54])
- Personal Finance Club Investment Growth Retirement Calculator ([70])
- cFIREsim ([71])

What we are contributing:

Simple diagnosis if person will be ok financially through retirement

Optimize Contributions - Ensuring maximum benefits from tax advantaged retirement account options

Optimize Asset Allocations

Alternatives like Annuities ([12])

System Inputs

- Pre/Post Retirement Salary
- Current Annual Spending
- Taxes
- Current age
- Expected retirement age
- Expected life expectancy
- User expectations around inflation and salary increases

Let's look at your *present status*

* Annual
Income ?

* How old
are you?

* What's
your Annual
Spending?

* Annual Tax
Amount ?

* Account
Type ?

What would
be your total
yearly
investment
contribution?

System Inputs Continued

- Typical Retirement Accounts
 - IRAs, 401k, HSA([14-16]), Brokerage
 - Pre Retirement Account Info: Yearly contribution, rate of return, Current Balance
 - Portfolio Choices: US, International, Bond ETF/mutual funds
 - Defaults Vanguard: VOO, VXUS, BND
 - Why? Diverse ([42, 44], [47-49]), low expense ([43])

What is your current balance in your investment account?

What would be the rate of return of your investment?

Please select your ETF and mutual funds:

US Stocks

International Stocks

Bonds

Now, let's analyze your expectations..

* What would be your retirement age?

* What would be your yearly retirement spending?

* Monthly Social Security Benefits in retirement?

What are ETFs/Mutual Funds

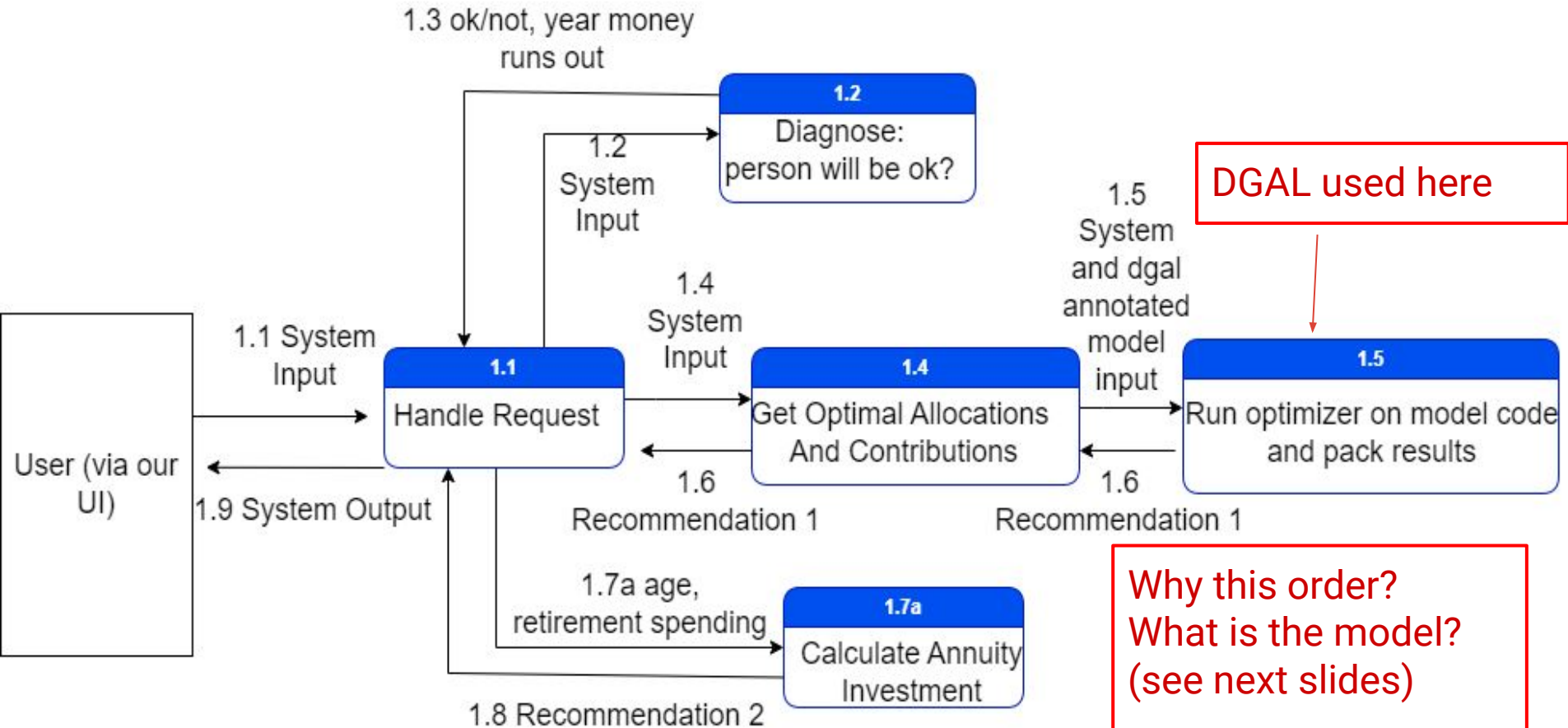
Mutual Funds and Exchange-Traded Funds

- A group of many individual investments
- Passively or Actively Managed
- Diversifies risk by investing in many different securities
- ETFs typically charge less in fees than Mutual Funds ([32,35,40])

System Outputs

- **Diagnosis:** person's income lasts through retirement?
- **Recommendation 1:**
 - per account/total **optimal allocations, contributions** (preretirement)
 - Combined metrics: total cost, balance from all accounts
- **Recommendation 2:**
 - **Annuity investment amount to** meet postretirement spending (as in [54])
 - Appears if: user > 45 yrs and retirement balance > amount and retirement income < retirement spending (aligns with [51,52] recommendations)

System Flow

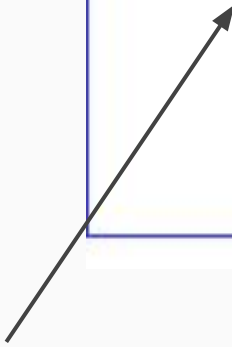


Diagnosis Relation To Model


Diagnosis (Pseudocode on right):

- Purpose: Evaluates user preparation for retirement
- **Requires Invocation of Optimizer (on Model)**

Implementing
compound interest
formula



```
for j in [current, death yr]:  
    income = salary + (retired? retirement account balances: 0)  
    if (income - expenses < 0):  
        return false  
    else:  
        for acnt in retirement accounts:  
            growth = balance * return rate  
            balance += growth + yearly contribution
```



Increasing these = improved results =>
indicate model factors

Model: Initial Formulation

Operations Research Perspective:

- **Mathematical model: “mathematical representation of an actual situation”** [55, Ch. 1, p. 1]
- **Understand goals and impacting factors,** then make mathematical model [55, Ch. 1, p. 5]
- **How: find decision variables, objective function, and constraints,** construct optimization model [55, Ch.3, pp. 49-53]

Application to Our Project:

- Goal: Max retirement account balance
- Factors: **Gains**(contribution, rate of return), **Losses** (volatility (fluctuation of returns [61]), expenses)

2 models: (Inaccurate System Representation):

Model 1 :
Maximize
contributions

Model 2 :
Maximize
balances

Model: Revised Formulation

After Professor Brodsky's Feedback, HA2, and HA3:

- To represent complex system: form inputs/outputs, metrics for **individual components** of the system **then combine**
- **Revised models**: one Atomic, one Composite

CS 787 Perspective: Gives steps to transform system idea to model to program

Composite Model: represents combination of accounts (All of User's Retirement Accounts)

Atomic Model:
represents single retirement account (ie IRA)

(401k) single account

HSA single account

Single Account Model: Inputs

```
1 {  
2   "user_information": {  
3     "total_current_income": 60000,  
4     "current_spending": 25000,  
5     "current_amount_paying_in_taxes_annually": 12000,  
6     "retirement_age": 73,  
7     "year_range_running_model_on": {  
8       "start_age": 30,  
9       "end_age": 65  
10    }  
11  },  
12  "account_information": {  
31  }
```

```
    },  
    "account_information": {  
      "account_type": "IRA",  
      "account_asset_info": {  
        "VVO": {  
          "weight": 0.5  
        },  
        "BND": {  
          "weight": 0.5  
        },  
        "VXUS": {  
          "weight": 0  
        }  
      },  
      "total_yearly_investment_contribution": 10000,  
      "current_balance_investment_account": 0  
    }  
  }
```

Decision Variables:
weights and
contribution

Single Account Model Outputs

Main Outputs:

- **Metrics: expected return, variance** (eventually standard deviation), **cost, growth**
- **Constraints**
Evaluation: (True/False)

```
return {  
  "account_information" : {  
    "account_type": account_type,  
    "account_expected_return" : portfolio_expected_return,  
    "account_variance": portfolio_var,  
    "total_growth_of_account_over_range": total_growth_of_  
    "total_account_balance_at_end_of_range": total_account_  
    "total_cost_incurred_on_account_by_end_of_range": tota  
    "total_yearly_cost_on_account": total_yearly_cost,  
    "constraints": constraints,  
    "total_account_balance_each_year": total_account_balanc  
    "debug": { },  
    "account_asset_info": newAssetInfo,  
    "total_yearly_investment_contributions_to_account": ye  
    "legal_limits_on_C_for_year_range": accountLegalLimits  
  },  
  "user_information": { }  
}
```

Single Account Model: Metrics (Our Implementation)

```
portfolio_expected_return += asset_info[securityTicker]["weight"] * geom_mean
```

Expected Return formula implementing (below from [56]):

$E_p = w_1 E_1 + w_2 E_2 + w_3 E_3$, **$w_i$ = weight of security, E_i = security's annualized return rate**

Calculating E_i :

- Sources calculate simple annualized return rate: (mean daily return in adjusted close prices (scraped from Yahoo Finance via Pandas DataReader * 250/252 trading days/yr) ([57,58])
- Geometric Mean: more accurate (accounts for growth on reinvestments) [59,60]
- Currently: we calculate geometric mean with yearly returns (calculated as in [57,58]) from year of fund inception to current year

$$\mu_{\text{geometric}} = [(1 + R_1)(1 + R_2) \dots (1 + R_n)]^{1/n} - 1$$

where:

- $R_1 \dots R_n$ are the returns of an asset (or other observations for averaging).

Picture Source: [59]

<https://www.investopedia.com/terms/g/geometricmean.asp>

Single Account Model: Metrics (Continued)

Standard Deviation:

Why use? Frequently used to measure asset volatility^[61], intuitive to user

Other volatility measures (listed from ^[63]):

Beta value: measures slope between security and market index returns (^[64],^[65])

Sharpe Ratio: formula simplifies to return/risk (or standard deviation) (^[66],^[67])

```
portfolio_var = sum([
    (asset_info[r]["weight"] * asset_info[c]["weight"] *
     asset_std_devs[r] * asset_std_devs[c] *
     asset_correlation_matrix.loc[r,c])
    for r in asset_correlation_matrix.index
    for c in asset_correlation_matrix.columns
])

# optimizer cannot handle math.sqrt so just take sqrt of variance returned
portfolio_std_dev = math.sqrt(portfolio_var)
```

We are calculating **portfolio standard deviation** (formula from ^[62] implemented above) with standard deviation of daily return rates since inception of the fund to current year

Single Account Model: Metrics (Continued)

Total growth of account over range (commented) = cumulative growth in account

Total cost incurred = cumulative maintenance cost

Total account balance at end of range: accounts for growth and cost

Total account balance each year: for graphing

```
for y in range(startyr, endyr):
    growth = total_account_balance_at_end_of_range * portfolio_expected_return
    growth += yearly_C
    # total_growth_of_account_over_range += growth
    total_account_balance_at_end_of_range += growth
    # calculation for cost: in terms of expense ratios (leave out taxes for now)
    # do we want to include inflation?
    # interesting: optimizer gives this issue since multiplying objective (total_c
    # RuntimeError: Selected solver is unable to handle objective functions with qu
    total_yearly_cost = sum([(newAssetInfo[security]["expense_ratio"] *
                             newAssetInfo[security]["weight"] *
                             total_account_balance_at_end_of_range)
                             for security in newAssetInfo])
    total_cost_incurred_by_end_of_range += total_yearly_cost
    total_account_balance_at_end_of_range -= total_yearly_cost
    total_account_balance_each_year.append(total_account_balance_at_end_of_range)
```

Metrics nonlinear
cannot use LP
optimizer

Single Account Model: Constraints

```
364 constraints = dgal.all([
365     allAssetWeightsGreaterThanOrEqualToZero,
366     assetWeightsSumToOne,
367     yearlyContributionGreaterThanOrEqualToZero,
368     legalYearlyLimitsOnContributionSatisfied,
369     ageLegalLimitsOnContributionSatisfied,
370     allocateMoreAccordingToTraits,
371     yearlyContributionWithInBudget
372 ])
```

yearly contribution < IRS 2022/23 limits

401k contribution = 0 after retired or
HSA contribution = 0 after 65 (Medicare)

Contribution < (salary - spending - taxes)

Constraints encode domain (US Legal and Financial) rules

Implementing 4% rule from Trinity Study [72]

Allocate highest weight to security based on user spending and time to retirement

```
if (years_left_til_retirement < 20 or
    (current_spending > (0.04 * investment_starting_balance))):
    # make weight of least risky security is highest
else:
    # make weight of highest return security highest
```

Combined Accounts Model: Inputs

HA2 and
3 format

```
3   "account_information": {
4     "account_type": "combined",
5     "subAccounts": [
6       "IRA",
7       "401k",
8       "HSA"
9     ],
10    "sub_account_inputs": { }
11  },
12  "user_information": {
13    "total_current_income": 60000,
14    "current_spending": 25000,
15    "current_amount_paying_in_taxes_annually": 12000,
16    "retirement_age": 73,
17    "year_range_running_model_on": {
18      "start_age": 30,
19      "end_age": 65
20    }
21  },
22  }
```

Holds each single account model's account information input as a separate object (types from subAccounts)

For each account:
invokes atomic
model with user and
account info

Combined Accounts Model: Outputs, Metrics, and Constraints

Outputs:

- **Constraints:** every account constraints satisfied and total contributions < salary - spending - taxes
- **Metrics:** all total_X = sum of X from single account model output from all accounts
- Other output for reference/debugging

```
constraints = dgal.all([
    yearlyContributionsToAllAccountsInBudget,
    subAccountConstraints])

return {
    # "utility": utilityFunction,
    "constraints": constraints,
    "total_balance_from_all_investments_at_end_of_year_range" : total_
    "total_expected_return" : total_expected_return,
    "total_variance": total_variance,
    "total_growth_of_all_accounts": total_growth_of_all_accounts,
    # "all_investment_accounts_avg_expected_return" : all_investment_c
    # "all_investment_accounts_avg_std_dev": all_investment_accounts_c
    "account_information": {
        "account_type": "combined",
        "subAccounts": subAccounts,
        "sub_account_outputs": allSubAccountModelOutputs
    },
    "user_information": userInfo
}
```

Current Optimization (Same Format as in HA2 and HA3)

Linear Objective function:

- total_growth_of_all_accounts (metric in combined accounts model output o)
- Inaccurate, needed for LP solver (in progress)
- **Why this metric?**
Incorporates all decision variables (solver finds nonzero values)

Note: we reused the code from Homework 2 and 3 in CS 787 (given by the Prof)

```
def runOptimizer(annotated_input):  
    def constraints(o):  
        return (dgal.all([ o["constraints"]]))  
  
    optAnswer = dgal.max({  
        "model": combiningAccountsMod,  
        "input": annotated_input,  
        "obj": lambda o: o["total_growth_of_all_accounts"],  
        "constraints": constraints,  
        "options": {"problemType": "mip", "solver": "glpk", "debug": True}  
    })  
    optOutput = combiningAccountsMod(optAnswer["solution"])
```

=
sum(total_growth_of_account
over_range) (from each single
account model output)

```
length_run = endyr - startyr  
total_growth_of_account_over_range =  
    (length_run * (total_account_balance_at_end_of_range * portfolio_expected_return))  
    + (length_run * (yearly_C))
```

Current investment balance

Annuity Relation to Model

- PO = Principal
- r = Annual interest rate
- n = Number of payments per year
- t = Number of years of payments

Annuity Information (from [12,50-53]):

- Equal to CDs and pensions
- Less Risky **Alternative to Optimizer's Recommendations**
- **Average 2% return** (small return rate)
- Invest P_0 and get paid through retirement

Payout Annuity Formula

$$P_0 = \frac{\text{PMT} \left(1 - \left(1 + \frac{r}{n} \right)^{-nt} \right)}{\frac{r}{n}}$$

Both Pictures' Source: ([69])

<https://www.annuity.org/annuities/immediate/annuity-calculator/>

Tech Stack

- Python
- Pyomo
- Flask
- HTML/CSS/JAVASCRIPT
- Chart.js
- Heroku
- EC2

FLASK

- Flask is a micro web application framework written in Python and based on
 - WSGI toolkit and
 - Jinja template engine
- Flask is extensible while keeping the core simple. Examples: Flask-SQLAlchemy, Flask-Mail, Flask-Admin, Flask-Cache
- The applications that make use of the Flask framework are Pinterest, LinkedIn as well as the community web page for Flask itself.

Heroku

- Heroku is an elastic multi-language, multi-framework platform as a service.
- Heroku is built on a managed container system and runs apps in smart containers called DINOS.
- Developers can deploy their application's code in any of the languages Heroku supports as dinos work in a fully managed runtime environment.
- Its supported languages include nodeJS, ruby, Java, PHP, python, go, scala, and closure, or you can use another language through a third-party build pack as long as it runs on Linux.
- Heroku takes care of monitoring and patching the system and language stacks to keep the platform up-to-date, reliable and secure.

Steps to deploy Python Web API to Heroku (followed from [68])

- Before deploying we need to generate two files in the root folder.

❏ Procfile

❏ requirements.txt

- Install a python server called gunicorn → `pip install gunicorn`
- After installing, generate a new file called Procfile → `echo > Procfile`
- Add this line `web: gunicorn main : app` in Procfile
- Now we will generate a requirements file by using the command
→ `pip freeze > requirements.txt`
- Now, make use of GitHub to deploy the code to Heroku by linking your GitHub account to Heroku.

contd.

- GLPK was installed via the use of an Aptfile and Heroku's build packs for python applications ([80] explains this is needed for files not in Python package).
- Followed the steps (from [80]) below to create the Heroku app using GIT
\$ git init
\$ git add .
\$ git commit -m "first commit"
\$ heroku create cs787-proj-app
\$ heroku buildpacks:set heroku/python
\$ heroku buildpacks:add --index 1 heroku-community/apt
\$ git push heroku master

```
requireme...  main.py  modelco
aniso8601==9.0.1
asgiref==3.5.2
autopep8==1.5.4
click==7.1.2
Flask==1.1.2
Flask-Cors==3.0.10
Flask-RESTful==0.3.9
gunicorn==20.1.0
itsdangerous==1.1.0
Jinja2==2.11.2
MarkupSafe==1.1.1
numexpr
numpy
numpydoc
pandas
pandas-datareader==0.10.0
pycodestyle==2.6.0
Pyomo==6.4.2
python-dateutil
python-lsp-jsonrpc==1.0.0
pytz==2020.5
scikit-image
scikit-learn
scipy==1.9.1
six==1.15.0
toml==0.10.2
Werkzeug==2.0.3
xlrd
```

Contains packages from [80],
[68] requirements.txt

Aptfile (same as in [80])

2 lines (2 sloc) | 20 Bytes

```
1  libglpk40
2  glpk-utils
```


Bootstrap

- Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development.
- We used bootstrap's HTML and CSS based design templates for screen size, cards, buttons, modals, tables and tooltips.

Chart.js

- Chart.js is a free, open-source JavaScript library for data visualization and it provides a selection of ready to go charts which can be styled and configured.
- Chart.js renders chart elements on an HTML5 canvas unlike several other charting libraries that render as SVG making it very performant, especially for large datasets.
- We are using line and pie charts to visualize growth of accounts and % contribution to each account.

Chart.js (contd.)

- Include a link to the providing CDN
`"https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.9.4/Chart.js"`
- add a `<canvas>` to where you want to draw the chart: `<canvas id="myChart" style="width:100%;max-width:700px"></canvas>`
- Example:

```
var myChart = new Chart("myChart", {  
  type: "line",  
  data: {},  
  options: {}  
});
```

AWS Elastic Compute Cloud (EC2)

- We will be using AWS EC2 for cloud hosting of our 'Retirement Planning Tool'.
- It allows to obtain and configure virtual compute capacity in the cloud.
- EC2 also ensures support for security, ports, processors and networking facilities.

Next Steps

Needed Steps:

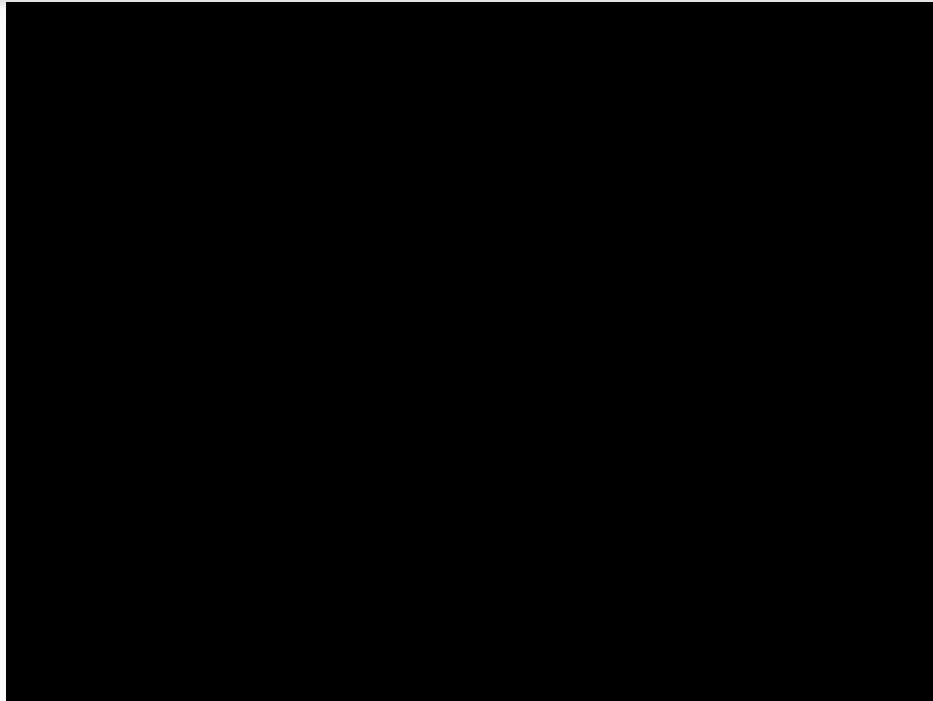
- Fix Diagnosis Code - incorporate more than one account
- Integrate our backend code with data (from JSON files)
- Integrate front end form and result screen, also make them more dynamic

Potential Steps/Challenges:

- Make linear functions to represent our KPIs to incorporate **utility function**
- **Extract user's risk level** using hypothetical questions (as in calculators)
- adjust **allocateMoreAccordingToTraits** constraint
- Try to find API that gives complete data set of daily prices
- Providing other optional categories for asset allocation: REITs - Real Estate
- Account for taxes in different account types (Traditional vs. Roth)

Demo

We incorporated the
following pictures in
our product:
[73], [74], [117]



Challenges and Limitations

The following limitations reduced accuracy in our recommendations:

Data Limitations:

- **IRS changes** contribution and RMD **limits** each year (**our solution: had to manually find and update for 2023, assuming: limits remain same over time**)
- Pandas DataReader for Yahoo Finance:
 - did not return inception dates, expense ratios (**our solution: manual find**)
 - Missing some daily close prices (since inception) - causes inaccurate expected return rate calculated (**our solution: considering using expected return rates from past 10 yrs gathered manually/keep our calculations for security expected return**)
- Inflation (solution: accept as input) and expense ratios hard to account (solution: take retail)

Metrics Limitation:

- Cannot certify expected return rate (though more stable for mutual funds/ETFs than stocks)

Citations

- [1] "Sources of Retirement Income | FINRA.org."
<https://www.finra.org/investors/learn-to-invest/types-investments/retirement/managing-retirement-income/sources-retirement-income>
(accessed Sep. 01, 2022).
- [2] A. Fontinelle, "Reverse Mortgage Definition," *Investopedia*. <https://www.investopedia.com/mortgage/reverse-mortgage/> (accessed Sep. 05, 2022).
- [3] A. Twin, "Refinance: How and When It Happens," *Investopedia*. <https://www.investopedia.com/terms/r/refinance.asp> (accessed Sep. 05, 2022).
- [4] The Investopedia Team, "What Is an IRA?," *Investopedia*. <https://www.investopedia.com/terms/i/ira.asp> (accessed Sep. 05, 2022).
- [5] Liz Manning, "Federal Insurance Contributions Act (FICA)," *Investopedia*. <https://www.investopedia.com/terms/f/fica.asp> (accessed Sep. 01, 2022).
- [6] Julia Kagan, "How Social Security Benefits Work," *Investopedia*. <https://www.investopedia.com/terms/s/social-security-benefits.asp> (accessed Sep. 05, 2022).
- [7] J. Lee, "What Is Home Equity And How Does It Work?," *Bankrate*. <https://www.bankrate.com/home-equity/what-is-home-equity/> (accessed Sep. 05, 2022).
- [8] J. Brown, "What Is A Home Equity Loan?," *Bankrate*. <https://www.bankrate.com/home-equity/what-is-home-equity-loan/> (accessed Sep. 05, 2022).

Citations (Continued)

- [9] The Investopedia Team, "What Is a Pension?," *Investopedia*. <https://www.investopedia.com/terms/p/pensionplan.asp> (accessed Sep. 05, 2022).
- [10] The Investopedia Team, "Defined-Benefit vs. Defined-Contribution Plan: What's the Difference?," *Investopedia*. <https://www.investopedia.com/ask/answers/032415/how-does-defined-benefit-pension-plan-differ-defined-contribution-plan.asp> (accessed Sep. 05, 2022).
- [11] Adam Hayes, "What Is a Lump-Sum Payment?," *Investopedia*, Dec. 29, 2021. <https://www.investopedia.com/terms/l/lump-sum-payment.asp> (accessed Sep. 05, 2022).
- [12] Julia Kagan, "What Is an Annuity?," *Investopedia*. <https://www.investopedia.com/terms/a/annuity.asp> (accessed Sep. 08, 2022).
- [13] Claire Boyte-White, "How Does a 401(k) Work After Retirement?," *Investopedia*. <https://www.investopedia.com/articles/personal-finance/111615/how-401k-works-after-retirement.asp> (accessed Sep. 16, 2022).
- [14] Kelley C. Long, "Medicare's tricky rules on HSAs after age 65," *Journal of Accountancy*, Jul. 01, 2021. <https://www.journalofaccountancy.com/issues/2021/jul/medicare-rules-on-hsa-after-age-65.html> (accessed Sep. 16, 2022).
- [15] Amy Fontinelle, "Retirement Uses for Your Health Savings Account (HSA)," *Investopedia*. <https://www.investopedia.com/articles/personal-finance/091615/how-use-your-hsa-retirement.asp> (accessed Sep. 16, 2022).
- [16] "5 ways HSAs can help with your retirement | Fidelity," Aug. 22, 2022. <https://www.fidelity.com/viewpoints/wealth-management/hsas-and-your-retirement> (accessed Sep. 05, 2022).

Citations (Continued)

- [32] "Types of ETFs," *Schwab Brokerage*. <https://www.schwab.com/etfs/types> (accessed Sep. 27, 2022).
- [33] "Vanguard ETF/fund ratios - Bogleheads." https://www.bogleheads.org/wiki/Vanguard ETF/fund_ratios (accessed Oct. 04, 2022).
- [34] "Types of mutual funds," *Schwab Brokerage*. <https://www.schwab.com/mutual-funds/types> (accessed Sep. 27, 2022).
- [35] "ETFs vs Mutual Funds," *Schwab Brokerage*. <https://www.schwab.com/etfs/mutual-funds-vs-etfs> (accessed Sep. 27, 2022).
- [38] Bogleheads, "Three-fund portfolio," *Bogleheads Wiki*, Nov. 06, 2010. [Online]. Available: https://www.bogleheads.org/wiki/Three-fund_portfolio [Accessed Oct. 4, 2022]
- [40] "What Is an Exchange Traded Fund (ETF)?," *Investopedia*. <https://www.investopedia.com/terms/e/etf.asp> (accessed Sep. 22, 2022).
- [42] "VOO vs. VTI - Vanguard's S&P 500 and Total Stock Market ETFs," *Optimized Portfolio*. <https://www.optimizedportfolio.com/voo-vs-vti/> (accessed Sep. 22, 2022).
- [43] "The 5 Best Index Funds with Low Expense Ratios You Can Invest in Right Now," *Time*, Oct. 27, 2021. Accessed: Sep. 27, 2022. [Online]. Available: <https://time.com/nextadvisor/investing/5-best-index-funds-low-expense-ratio/>
- [44] "Charles Schwab vs. Vanguard," *Investopedia*. <https://www.investopedia.com/charles-schwab-vs-vanguard-4587941> (accessed Sep. 27, 2022).
- [47] "VOO-Vanguard S&P 500 ETF | Vanguard." <https://investor.vanguard.com/investment-products/etfs/profile/voo> (accessed Oct. 04, 2022).
- [48] "BND-Vanguard Total Bond Market ETF | Vanguard." <https://investor.vanguard.com/investment-products/etfs/profile/bnd> (accessed Oct. 04, 2022).
- [49] "VXUS-Vanguard Total International Stock ETF | Vanguard." <https://investor.vanguard.com/investment-products/etfs/profile/vxus> (accessed Oct. 04, 2022).
- [50] "Retirement Annuities | Annuity Solutions to Consider | Fidelity." <https://www.fidelity.com/annuities/overview> (accessed Oct. 04, 2022).
- [51] "What is the Best Age to Buy An Annuity?," *Annuity.org*. <https://www.annuity.org/annuities/buy/best-age-to-buy-an-annuity/> (accessed Oct. 03, 2022).
- [52] "Alternatives to Annuities for Income in Retirement | What Are My Options?," *Annuity.org*. <https://www.annuity.org/annuities/buy/alternatives/> (accessed Oct. 03, 2022).
- [53] "Annuities | Investor.gov." <https://www.investor.gov/introduction-investing/investing-basics/investment-products/insurance-products/annuities> (accessed Oct. 04, 2022).

Citations (Continued)

- [54] "Annuity Income Calculator | Brighthouse Financial." [Online]. Available: <https://www.brighthousefinancial.com/products/annuities/annuity-income-calculator/>. [Accessed: Nov. 12, 2022]
- [55] W. L. Winston and J. B. Goldberg, Operations Research: Applications and Algorithms, 4th ed. Belmont, CA: Thomson/Brooks/Cole, 2004.
- [56] "How to Calculate Expected Portfolio Return," Investopedia. [Online]. Available: <https://www.investopedia.com/ask/answers/061215/how-can-i-calculate-expected-return-my-portfolio.asp>. [Accessed: Nov. 14, 2022]
- [57] randerson112358, "Calculate Annualized Expected Stock Returns Using Python," Medium, Mar. 23, 2022. [Online]. Available: <https://python.plainenglish.io/calculating-annualized-expected-stock-returns-using-python-aaba430ca8a9>. [Accessed: Nov. 14, 2022]
- [58] B. Brenyah, "Calculating Expected Rates of Returns for a Portfolio of stocks with Python," DS Biz, Sep. 18, 2017. [Online]. Available: <https://medium.com/python-data/calculating-expected-rates-of-returns-for-a-portfolio-of-stocks-with-python-e3afbd4eeba5>. [Accessed: Nov. 14, 2022]
- [59] "What Is a Geometric Mean? How to Calculate and Example," Investopedia. [Online]. Available: <https://www.investopedia.com/terms/g/geometricmean.asp>. [Accessed: Nov. 14, 2022]
- [60] "How to Calculate Your Investment Return," Investopedia. [Online]. Available: <https://www.investopedia.com/articles/08/annualized-returns.asp>. [Accessed: Nov. 14, 2022]
- [61] "Volatility: Meaning In Finance and How it Works with Stocks," Investopedia. [Online]. Available: <https://www.investopedia.com/terms/v/volatility.asp>. [Accessed: Nov. 14, 2022]
- [62] A. K. Srivastav, "Portfolio Standard Deviation," WallStreetMojo, Oct. 10, 2018. [Online]. Available: <https://www.wallstreetmojo.com/portfolio-standard-deviation/>. [Accessed: Nov. 14, 2022]

Citations (Continued)

- [63] “5 Ways to Measure Mutual Fund Risk,” Investopedia. [Online]. Available: <https://www.investopedia.com/investing/measure-mutual-fund-risk/>. [Accessed: Nov. 14, 2022]
- [64] “Beta: Definition, Calculation, and Explanation for Investors,” Investopedia. [Online]. Available: <https://www.investopedia.com/terms/b/beta.asp>. [Accessed: Nov. 15, 2022]
- [65] “Portfolio Beta Calculator.” [Online]. Available: <https://www.omnicalculator.com/finance/portfolio-beta>. [Accessed: Nov. 15, 2022]
- [66] “Sharpe Ratio Formula and Definition With Examples,” Investopedia. [Online]. Available: <https://www.investopedia.com/terms/s/sharperatio.asp>. [Accessed: Nov. 15, 2022]
- [67] “Sharpe Ratio,” Corporate Finance Institute. [Online]. Available: <https://corporatefinanceinstitute.com/resources/risk-management/sharpe-ratio-definition-formula/>. [Accessed: Nov. 15, 2022]
- [68] khalan, samir. “Flask-RESTful API With Heroku.” *Analytics Vidhya*, 19 Jan. 2021, <https://medium.com/analytics-vidhya/flask-restful-api-with-heroku-da1ecf3e04b>.
- [69] “Annuity Calculator | Calculate Your Payout,” Annuity.org. [Online]. Available: <https://www.annuity.org/annuities/immediate/annuity-calculator/>. [Accessed: Nov. 16, 2022]
- [70] “Investment Growth Retirement Calculator,” Personal Finance Club, Jul. 07, 2018. [Online]. Available: <https://www.personalfinanceclub.com/investment-growth-retirement-calculator/>. [Accessed: Nov. 16, 2022]
- [71] “cFIREsim.” [Online]. Available: <https://www.cfiresim.com/>. [Accessed: Nov. 16, 2022]
- [72] “Trinity study,” Wikipedia. Jan. 15, 2022 [Online]. Available: https://en.wikipedia.org/w/index.php?title=Trinity_study&oldid=1065774281. [Accessed: Nov. 16, 2022]

Citations (Continued)

- [73] What is asset allocation? Best asset allocation strategies! (n.d.). Napkin Finance. Retrieved November 16, 2022, from <https://napkinfinance.com/napkin/what-is-asset-allocation/>
- [74] What are Annuities? (n.d.). Napkin Finance. Retrieved November 16, 2022, from <https://napkinfinance.com/napkin/annuities/>
- [75] W3Schools online HTML editor. (n.d.). Retrieved November 16, 2022, from https://www.w3schools.com/howto/tryit.asp?filename=tryhow_css_flip_card
- [76] Chart.js | Chart.js. (n.d.). Retrieved November 16, 2022, from <https://www.chartjs.org/docs/latest/>
- [77] contributors, M. O., Jacob Thornton, and Bootstrap. (n.d.). Get started with Bootstrap. Retrieved November 16, 2022, from <https://getbootstrap.com/docs/5.2/getting-started/introduction/>
- [78] Chart.js. (n.d.). Retrieved November 16, 2022, from https://www.w3schools.com/js/js_graphics_chartjs.asp
- [79] What is web hosting? - Web hosting explained - aws. (n.d.). Amazon Web Services, Inc. Retrieved November 16, 2022, from <https://aws.amazon.com/what-is/web-hosting/>
- [80] CuriousChemE, "FuelOptGLPK." [Online]. Available: <https://github.com/CuriousChemE/FuelOptGLPK#readme>

Citations (Continued)

[81] Salesforce Developers. *Introduction to Heroku*.

<https://www.slideshare.net/developerforce/df13-introduction-to-heroku>

[82]“Flask (Web Framework).” *Wikipedia*, 7 Nov. 2022. *Wikipedia*,

[https://en.wikipedia.org/w/index.php?title=Flask_\(web_framework\)&oldid=1120483147](https://en.wikipedia.org/w/index.php?title=Flask_(web_framework)&oldid=1120483147)

[83]Debbie. “How to Deploy a Python + Flask API on Heroku.” *Medium*, 17 Jan. 2022,

<https://levelup.gitconnected.com/how-to-deploy-a-python-flask-api-on-heroku-2e5ddfd943ef>.