Real Estate Portfolio Optimization Project Plan

Blake Roberts	Project Lead / Backend
Colton Goode	Meeting Scribe / Backend
Kevin Johnson	Test Engineer / Frontend
Leelabari Fulbel	Meeting Facilitator / Frontend
Nickolas Moeller	Report Manager / Backend
Client	Principal Financial Group
Faculty Advisor	Chinmay Hegde
Team Website	https://sdmay19-07.sd.ece.iastate.edu
Team Email	sdmay19-07@iastate.edu

Table of Contents

Figures

Tables

Symbols

Definitions

API	Application Program Interface
GUI	Graphical User Interface
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
JS	JavaScript
MPT	Modern Portfolio Theory
MVC	Model View Controller
MVP	Minimum Viable Product
PDF	Portable Document Format
PM	Portfolio Manager
REST	Representational State Transfer
SPA	Single Page Application
SQL	Structured Query Language
SSL	Secure Socket Layer
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UI	User Interface
UX	User Experience

VPN	Virtual Private Network
Web App	Website Application

Introductory Materials

Acknowledgment

We would like to thank our advisor Chinmay Hegde. Also, Benjamin Harlander, Q Mabasa, Arthur Jones, Jonathan Ling, and Jonathan Frank of Principal Financial Group.

Problem Statement

Principal lacks an in-house (or not-3rd-party) tool capable of assessing the potential return and risk of real estate investments. Without such a tool, analysis of assets comes down to a per-property basis which lacks a high, market, level view of the investment space. Principal attempts to mitigate the lack of tooling through the use of a third party, utilizing Costar. Costar compiles quarterly reports including market and property level analysis; however, the reports are both costly and slow in the making.

An application is needed which analysis market data and user constraints to advise portfolio managers in purchase and sale decisions. The application will demonstrate principles of Modern Portfolio Theory (MPT), namely, the Markowitz portfolio optimization model. The application must be flexible to support many modes of portfolio analysis by handling a multitude of constraints over factors such as: market, property type, and time period.

Intended Project

Our software application will be used in a native application that would not be vulnerable to network or power dropouts.For data processing, we are going to use Python and SQL Database. The user interface will most likely be written in Dash which is an open-source Python framework. That will allow the Principal data science team to continue to modify the application after we are done.

Intended Users and Intended Uses

Primarily, the intended users of the application are portfolio managers and analysts. The users will load portfolio data and constraints and then run the Markowitz portfolio optimization algorithm. The PM's will utilize the optimized portfolio to adjust their current investment holdings as they see fit. In addition, users will define their own constraints to feed our optimization algorithm to meet their personal portfolio requirements.

The application will perform Markowitz and Black-Litterman portfolio optimization on a well-defined set of configurable constraints. The efficiency frontier graph will be accompanied by further market data analysis. Recommendations will be provided for the best course of action for the portfolio. The generated information and analysis will be available for export as a PDF report.

Assumptions and Limitations

Assumptions

1. The users will have an understanding of portfolio optimization and the concepts of MPT.
   a. Markowitz Model
   b. The effect of constraints on a optimization model
   c. What the results mean
2. User's input data follows a standard format of real estate properties.

Limitations

1. A closed set of portfolio analysis and optimization constraints will be available.
2. Purchasing and selling real estate is a complicated process which may render certain portfolio optimizations impossible.

Expected End Product and Other Deliverables

Our team is expected to deliver software capable of performing real estate portfolio optimization. As such, the application will be split into three project milestones: prototype, minimum viable product (MVP), and final product. The application code base and software documentation will be included in the delivery of the final product.

Prototype - December 2018

The prototype will consist of an application capable of performing Markowitz portfolio optimization on a limited set of constraints and displaying the model's efficiency frontier. The prototype will be demonstrated to the client, but will not be made readily available or distributed to stakeholders.

Minimum Viable Product - March 2019

The MVP will be a minimally complete application capable of optimization of a real estate portfolio on a logical subset of constraints. The MVP will be made available and distributed to stakeholders for quality assurance with the intention being iteration and improvement.

Real Estate Portfolio Optimization - May 2019

The final application will be handed off to Principal for future support and iteration. The application, in its final state, will be hosted and supported by Principal IT. Support of the application's availability will be handed off to Principal IT. The completed project will include:

- CodeBase - the git repository of the application's source code.

- Software Documentation - documentation on how to run the application locally for development purposes, as well as the software architecture and design.

Proposed Approach and Statement of Work

Objective of the Task

The goal of this project is to develop an application capable of performing real estate market analysis. Specifically, the application will implore ideas of MPT including Markowitz portfolio optimization. The application will also display the results of such analysis in common forms such as the portfolio efficiency frontier. The application will be a hosted website application available internally to Principal.

Functional requirements

Read current portfolio holdings. Basic: Upload portfolio holdings from csv files. This would require standardizing the input format. Ideal: Read returns and covariance from a database.

Read expected returns and covariance matrix for all assets. Basic: Upload returns and covariance matrix from csv files. This would require standardizing the input format. Ideal: Read returns and covariance from a database.

Update expected returns. User should have the ability to update the expected returns for specific asset groups by location, property type, or both.

Define optimization constraints. User can add or edit constraints prior to optimization. See section on Optimization Constraints for desired functionality.

Generate optimal portfolios. Based on the user's defined constraints, launch a backend process to determine the optimal portfolio holdings.

Generate and visualize efficient frontiers. Generate an efficient frontier from a range of risk & return values. Show individual markets, property types, and the current portfolio compared to the frontier.

Visualize current holdings. Show current portfolio holdings by geography, property, etc. Show top N holdings. Summarize expected returns and risk.

Visualize optimal portfolio results. Show optimal portfolio holdings by geography, property, etc. Show top N holdings. Summarize expected returns and risk.

Visualize differences between optimal and current holdings. Summarize differences between the optimal and current portfolios and display in maps, charts, plots, etc.

Recommend actions based on current holdings. Recommend buy and sell decisions given the current portfolio holdings. Justify decisions based on increasing expected returns, reducing risk, or both.

Share results. Functionality to export results to a report or share via email.

Constraints Considerations

Permanent Constraints

1. Portfolio must be fully invested - The assets weights in the portfolio must sum to 1.0.
2. Long positions only - All asset weights must be greater than or equal to zero.

User-defined Constraints

1. Property type constraint - The user can specify the minimum and maximum allocation of the portfolio to a specific property type (i.e. Sector as shown on the image below.)
    a. Example: Total Office weight < 30%
2. Geographic constraints - The user can specify the minimum and maximum allocation to a specific region, state, or metro.
    a. Example: New York City weight > 25%
3. Geography + Property type constraints - The user can specify a constraint for a property type in a specific geographical area.

Previous Work and Literature

In order to accurately design a system to perform portfolio optimization and real estate market analysis, research into the concepts of MPT and forms of portfolio optimization models was required.

Technology Considerations

At the moment we have two options for the frontend. The first would be running the frontend as a web app. This option for the frontend will be written in JavaScript that will visualize the data provided from the backend. The frontend will interface via HTTP and/or TCP with a backend application written in Python. Another option for the frontend is the Python framework Dash to generate User Interface elements from the data. The backend will use Python with various open-source libraries. For both frontend options, data will be taken in from the user via an SQL database or a csv file, then be stored and queried via an SQL database.

Security Considerations

The application will be hosted internally by principal. This application will be accessed via client web browser. The application will not need public network access; however, data transfer will be encrypted via HTTPS and/or SSL. This will remove the risk of internal man-in-the-middle attacks during a security breach. Additionally, this functionality will remove the risk of said attack if the application is made available outside Principal's office without a VPN connection.

Safety Considerations

Due to the nature of the project, there is no risk of harm or detriment to health in the development of our project or use of our product.

Task Approach

There are two approaches that we have ideated. Our first approach is a Server and Client paradigm to segment the front and backend using HTML/JS and Python. Our second approach utilizes the Dash framework to encompass the whole application in Python.

Approach 1: Server/Client

The initial approach, utilizing the server/client paradigm, creates a clear separation of the user interface and experience (UI/UX) and the data analysis and portfolio optimization code. Specifically, the frontend client code, written in JavaScript, will handle user input and displaying data. The backend server will be required to take in a standard set of user data and perform aggregation and analysis on portfolio market data. It will output a standard data model for the frontend to consume and display the data.

The added complexity of a frontend system runs the risk of slowing future feature development by the client's data science team. Extensive documentation on how to contribute to the frontend system will be required. However, the backend system will be very simple in its responsibility. This will allow for fast feature development of additional optimization strategies and constraints. This system also has the added benefit of an HTTP API. This API could be consumed by systems other than the frontend. One such example being a Python script capable of leveraging the data analysis of the server without the necessity of interaction with the user interface.

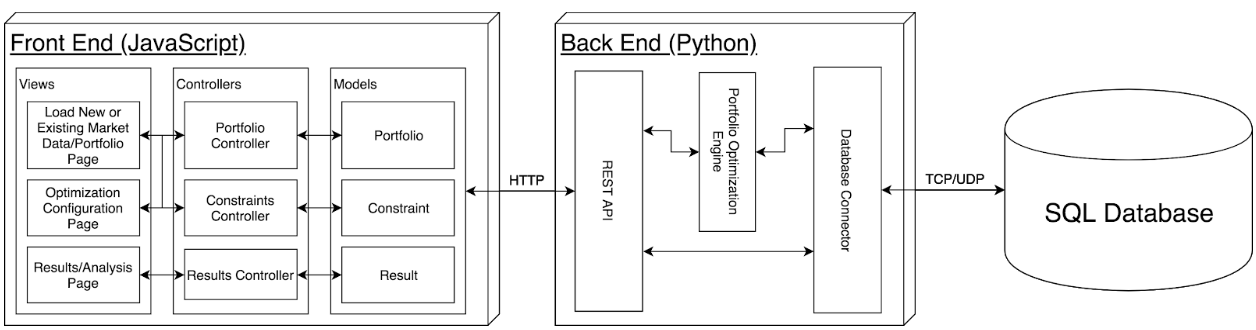

Figure 1: Application Block Diagram Utilizing a Server/Client Paradigm

Approach 2: Dash Framework

Utilizing the Dash framework will simplify the overall design of the application. This will enable future iterations to be made by Principal with little software experience in languages such as JavaScript. The worry with this design is its maintainability. Due to the large amount of abstraction by the Dash framework, development may become cluttered. This will lead to increased feature development time and the necessity of repeated code refactoring.

Nonetheless, the devised software architecture attempts to circumvent this risk by the use of the Model, View, Controller (MVC) design paradigm. By distinct, logical separation of concerns, the code will maintain clear organization. Including documentation surrounding the organization of the code will lead to ease of future feature development by Principal's data science team.

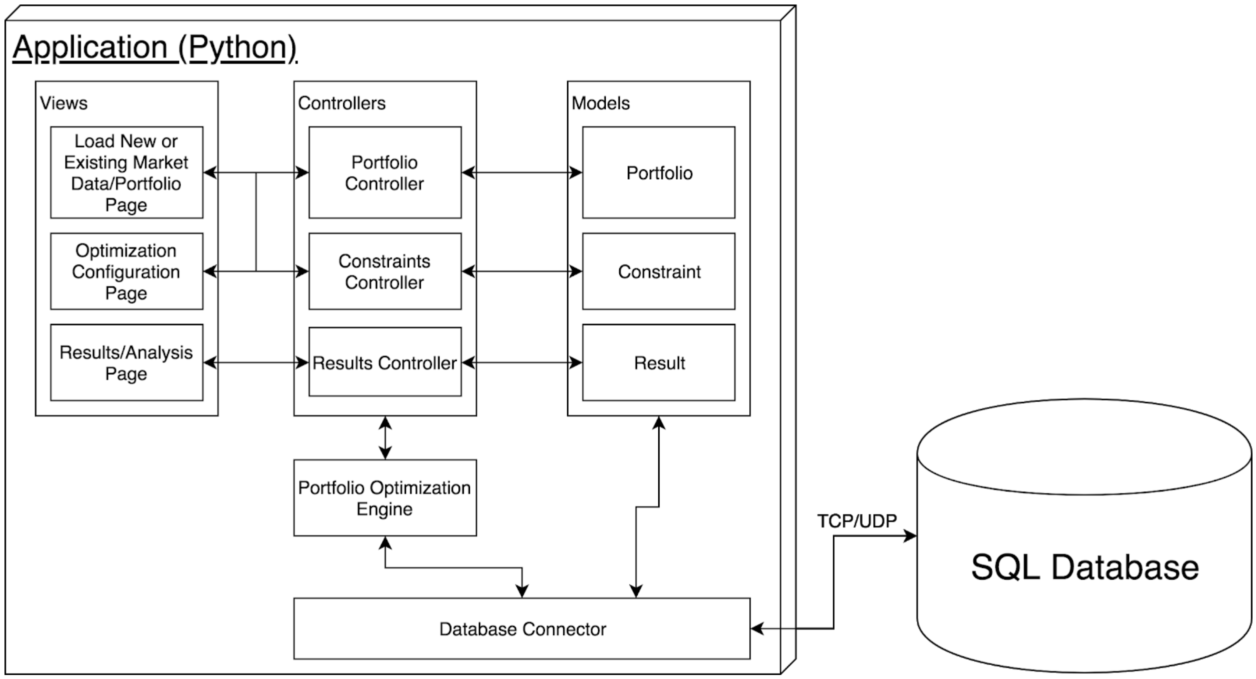

Figure 2: Application Block Diagram Utilizing Dash Framework

Possible Risks and Risk Management

The two main risks that we would expect is that the front end might not be intuitive and that the backend algorithms could be inefficient. This can be managed by starting early and making adjustments to the prototype as needed. Regular communication with the client will be crucial to creating the best product possible.

Project Proposed Milestones and Evaluation Criteria

Prototype: Software provides correct optimization with few constraints and some visual graph. The prototype will be a demoable web application.
Checkpoint: First Semester Demo December 2018

MVP: Software will be a minimally complete application capable of optimization of a real estate portfolio on a logical subset of constraints.
Checkpoint: March 2019

Testing: Software will be tested for performance and competence. User testing, with the client, followed by iteration will be carried out during this time as the final project goal is reached.
Checkpoint: April 2019

Final: The software will accomplish all functional requirements and fit the clients full intentions. The software will be available to the client as a product as well as its code base delivered with documentation.
Checkpoint: May 2019

10/26/18	First Semester Mid-Point Presentation with Principal stakeholders
12/18	Mid-Year Presentation with Advisors
12/18	Mid-Year Presentation with Principal stakeholders
03/19	Second Semester Mid-Point Presentation with Principal stakeholders
05/19	End-Year Presentation with Advisors
05/19	End-Year Presentation with Principal stakeholders

Project Tracking Procedures

Our team will participate in three weekly meetings throughout the rest of the semester. During these meeting is when a lot of larger decisions get made about the direction of the current and future and course of the project. These meetings result steps that will be taken in the future. Meeting minutes are also taken to document what was discussed.

Our team also will be filling out Principal mandated RASIC forms to document who is assigned to what task. RASIC stands for Responsible, Approves, Supports, Informed, Consulted. It is a table in which we assign the main responsibility of to a certain member and elect supporting members if need be. We also can flag who will approve something or just who will be informed and consulted. This is redone each week to document task assignments.

A project timeline will be implemented during the project. The timeline we have designed will give us a week by week picture of tasks the team needs to accomplish by a certain date. This is useful in determining what needs to be done next or if the team is behind or ahead of schedule.

Lastly, our team uses a GIT repository to keep track of the changes made to the code. GIT allows our team to develop remotely, and merge our code together in the end. It also keeps logs that allow us to keep track of any changes

Expected Results and Validation

This project will fully meet the requirements defined in Functional Requirements.
To confirm that our solution works, we will follow the Test Plan that will be describe below.

Test Plan

Program will be tested with provided sample data (imported csv files/read from database)
Program will be tested with a variety of different constraints (Geographic and/or Property Type constraints)
Algorithm will be tested to ensure the optimization follows MPT.
Testing and error handling must be done for incorrect data inputs. These data inputs include imported files, database queries, and expected return variations before the portfolio is optimized.
UI elements will be tested for correctness, intuitiveness, and reliability. These UI elements include graphs displaying portfolio data, tables containing portfolio data, labels, and buttons.

Estimated Resources and Project Timeline

Estimated Resources

Personnel Effort Requirements

1. 3 team meetings per week (with or without the client)
    a. Weekly project update meeting with client
    b. Weekly meeting with adviser
    c. Weekly technical meeting to discuss application (With or without client)
2. At minimum of 6 hours per person per week dedicated to working on the project outside of the 3 proposed meetings
3. Ability to travel to client office in downtown Des Moines twice a semester for in person presentations

Financial Requirements

The project will require no financial requirements. Technology used is constrained to be open source and therefore free to use. Additionally, the application will be hosted by Principal and will not use a cloud provider.

Project Timeline

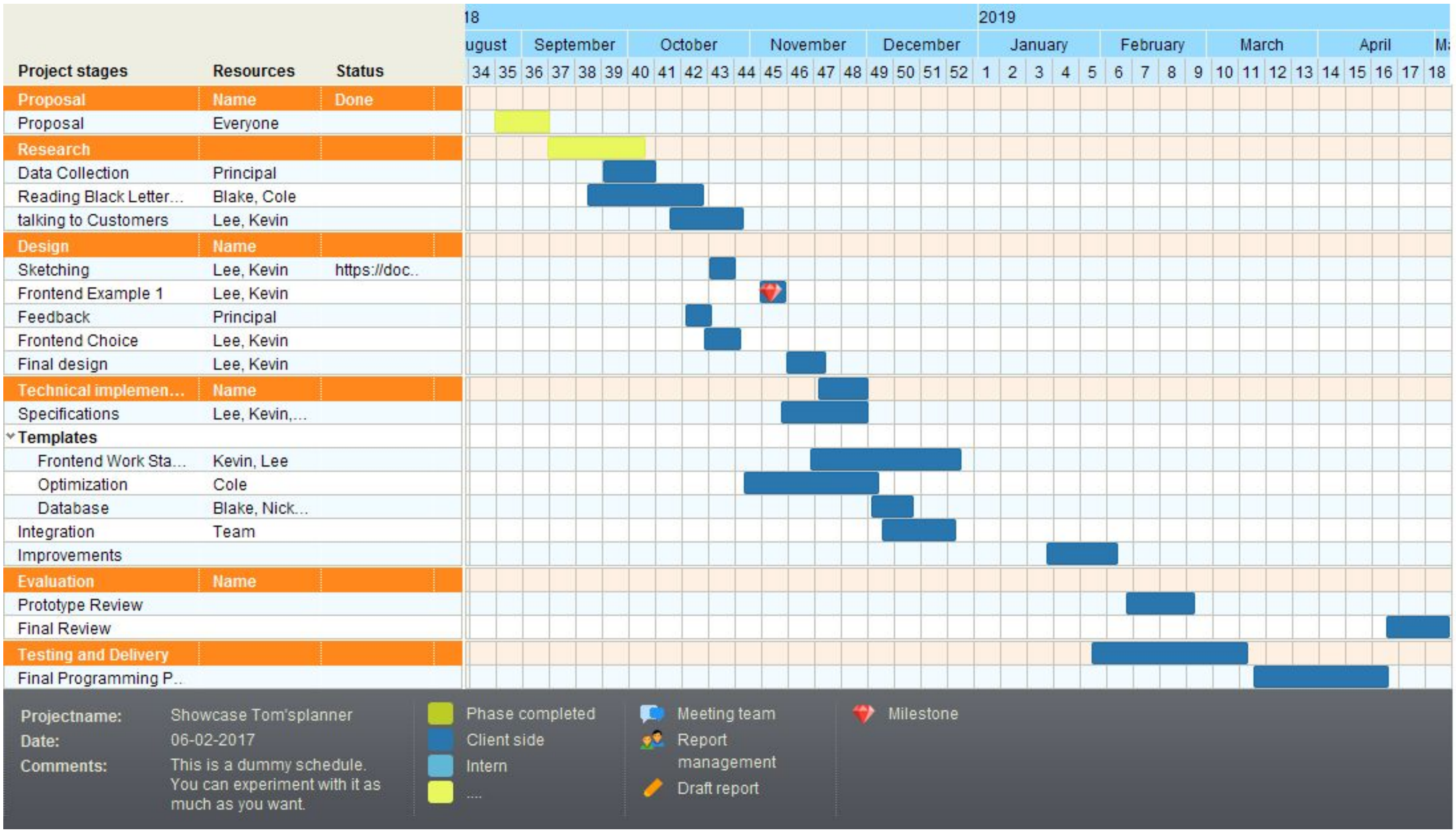

Figure 3: Project Timeline

Closure Materials

Closing Summary

So far, our team has managed to complete a couple of crucial steps. We have a working Markowitz algorithm in place. We have started to make mockup designs for our frontend.

Our goal is to create an app that will allow clients of Principal to use data science to optimize portfolios while also allowing for the specification of particular interest areas for investment.

The best plan of action for us is to create a native application that has a frontend and backend component. The backend be will written in a Python framework. That is the best option for allowing portfolio optimization while still incorporating the ability to communicate with the frontend framework. The Principal team is most familiar with it so they would be able to continue developing on it in the future. We will be using CoffeeScript for our frontend. This choice is more ambiguous and just has to deal with our personal preferences.

References

[1] T. Idzorek, A STEP-BY-STEP GUIDE TO THE BLACK-LITTERMAN MODEL. Ibbotson Associates , 2018.