Real Estate Portfolio Optimization
Final Project Report

Blake Roberts	Project Lead / Backend
Colton Goode	Meeting Scribe / Backend
Kevin Johnson	Quality Control / 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

Table of Contents **1**
- Figures 3
- Tables 3
- Definitions 3

Executive Summary **4**

1.0 Requirements Specification **5**
- 1.1 Functional Requirements 5
- 1.4 Non-functional Requirements 5

2.0 System Design & Development **6**
- 2.1 Design Plan 6
- 2.2 Design Objectives 6
  - 2.2.1 System Constraints 6
  - 2.2.2 Design Tradeoffs 6
- 2.3 Architectural Diagram 7
    - Figure 1: System Block Diagram 7
  - 2.3.1 Software Block Diagram 8
    - Figure 2: Application Block Diagram Utilizing Dash Framework 8
- 2.4 Data Structures 9
  - 2.4.1 Persistent Data Structures 9
  - 2.4.2 Power BI Data Structures 10
    - First Report (Current vs. Optimal vs. Market Portfolio) 10
    - Second Report (Efficiency Frontier Graph) 10

3.0 Implementation **11**
- 3.1 Implementation Diagram 11
- 3.2 Software Used 12
- 3.3 Rationale for Software Choices 12
- 3.4 Standards and Best Practices 12

4.0 Testing, Validation, & Evaluation **13**
- 4.1 Test Plan 13
- 4.2 Unit Testing 13
- 4.3 User-Level Testing 13
- 4.4 Evaluation 13

5.0 Project & Risk Management **14**

5.1 Task Decomposition 14
5.1.1 Roles & Responsibilities 14
5.2 Project Schedule 15
5.2.1 Proposed 15
Figure 3: Proposed Project Timeline 15
5.2.2 Actual 16
Figure 4: Actual Project Timeline 16
5.3 Risks & Mitigation 17
5.3.1 Potential 17
5.3.2 Actual 17
5.3.3 Mitigation 17
5.4 Lessons Learned 17
6.0 Conclusions 18
6.1 Closing Remarks 18
6.2 Future Work 18

Figures

Figure 1: System Block Diagram
Figure 2 : Block Diagram Utilizing Dash Framework
Figure 3 : Proposed Project Timeline
Figure 4 : Actual Project Timeline

Tables

Table 1 : Persistent Data Structures
Table 2 : Power BI Data Structures

Definitions

API	Application Program Interface
CSV	Comma-Separated Values
ID	Identification
MPT	Modern Portfolio Theory
MVP	Minimal Viable Product
NCREIF	National Council of Real Estate Investment Fiduciaries
PBI	Power BI
SQL	Structured Query Language
UI	User Interface
UX	User Experience

Executive Summary

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.

The application is needed to analyze market data and user constraints to advise portfolio managers in purchase and sale decisions. The application demonstrates principles of Modern Portfolio Theory (MPT), namely, the Markowitz portfolio optimization model. The application supports many modes of portfolio analysis by handling a multitude of constraints over factors such as: market, property type, and time period. Microsoft Power BI is used to visualize and break-down the optimization results.

1.0 Requirements Specification

1.1 Functional Requirements

1. **Read current portfolio holdings.** The user will be able to upload portfolio holdings from CSV files. This would require standardizing the input format. Ideal: Read returns and covariance from a database.
2. **Read expected returns and covariance matrix for all assets.** The user will be able to upload returns and covariance matrix from CSV files. This would require standardizing the input format. Ideal: Read returns and covariance from a database.
3. **Update expected returns.** The user should have the ability to update the expected returns for specific asset groups by location, property type, or both.
4. **Define optimization constraints.** The user can add or edit constraints prior to optimization. See section on Optimization Constraints for desired functionality.
5. **Generate optimal portfolios.** Based on the user's defined constraints, the application will launch a backend process to determine the optimal portfolio holdings.
6. **Generate and visualize efficient frontiers.** The application will generate an efficient frontier from a range of risk & return values. It will show individual markets, property types, and the current portfolio compared to the frontier.
7. **Visualize current holdings.** The application will show current portfolio holdings by geography, property, etc. Show top N holdings. Summarize expected returns and risk.
8. **Visualize optimal portfolio results.** The application will show optimal portfolio holdings by geography, property, etc. It must show top N holdings and summarize expected returns and risk.
9. **Visualize differences between optimal and current holdings.** The application will summarize differences between the optimal and current portfolios and display in maps, charts, plots, etc.
10. **Recommend actions based on current holdings.** Recommendations for buy and sell decisions given the current portfolio holdings will be shown. The application will attempt to justify decisions based on increasing expected returns, reducing risk, or both.
11. **Share results.** Results will be exported to a report or share via email.

1.4 Non-functional Requirements

1. The system will use only open source libraries and frameworks.
2. The system's reliance on specific types of infrastructure must be minimal.
3. Optimization takes no longer than 5 seconds to calculate.
4. A team of portfolio managers or analysts can use the application at one time.

2.0 System Design & Development

2.1 Design Plan

The project requires an application consisting of a user interface allowing the analysis and construction of models based on real estate market data. Reasonable proposed designs and approaches would be software solutions. As per the requirements of the project, both a native application and a web application are feasible.

A final approach would be to develop a Python Flask server integrated with the Python Dash framework by Plotly. The resultant web application would be hosted on Principal servers. This application would be reachable by any computer within Principal's network via an internet browser. This design would only require knowledge in Python which makes it an ideal candidate for this project.

2.2 Design Objectives

The application must suit the unique needs of assisting a real estate portfolio manager in decisions over the purchase or sale of real estate investments.

2.2.1 System Constraints

Because the application will be maintained by Principal's Data Science team, the preferred language to use is Python. Furthermore, a web app is desirable, but JavaScript and HTML is to be avoided. Dash by Plotly is thus a key framework to satisfy these constraints.

2.2.2 Design Tradeoffs

Python and Dash were used in development. This was not ideal for developing a full fledge web application. Dash posed many constraints on UI/UX which determined the viability of a feature where with JavaScript the feature would have been fully feasible.

2.3 Architectural Diagram

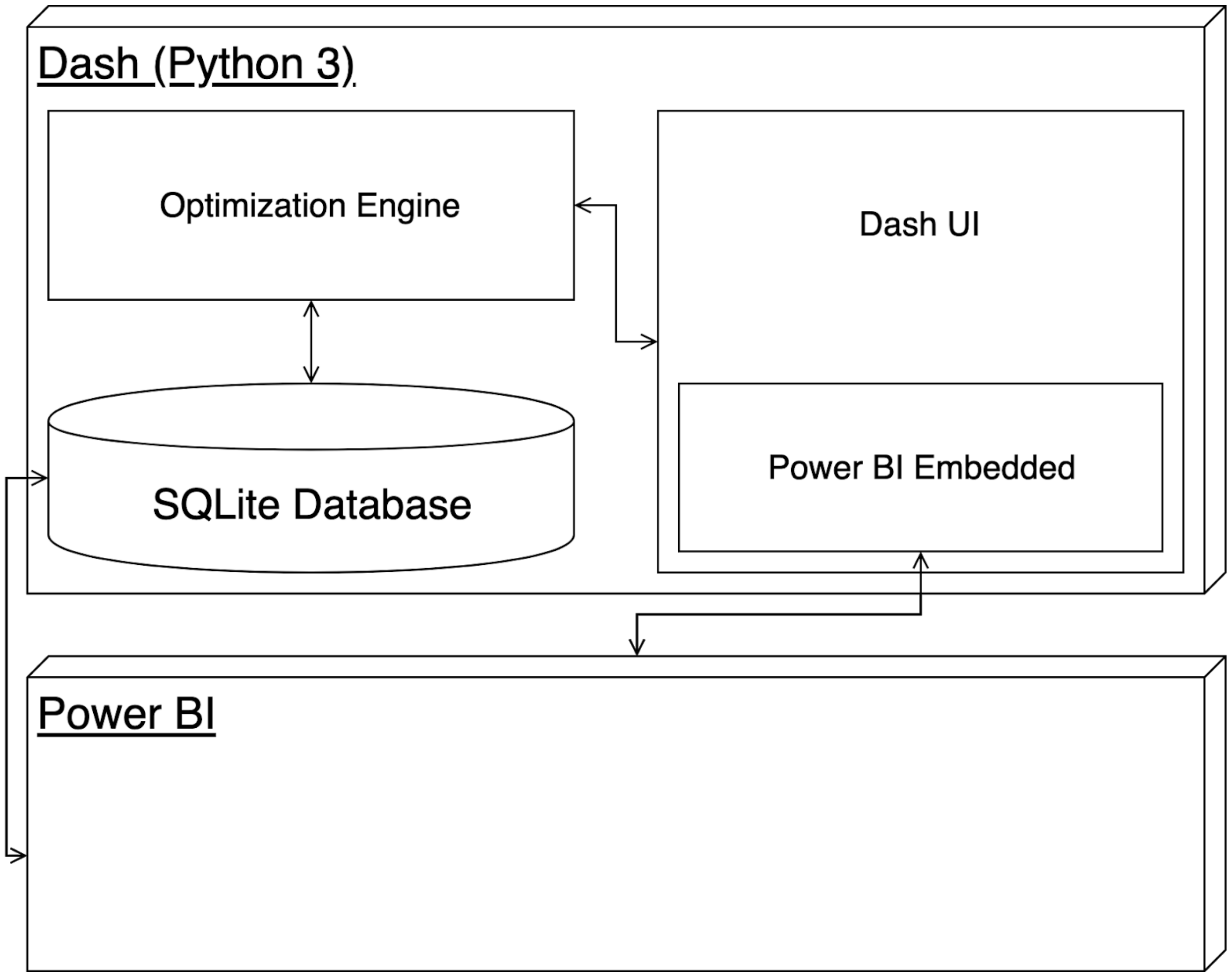


Figure 1: System Block Diagram

Our application is built entirely with Python with an embedded Power BI visuals.

2.3.1 Software Block Diagram

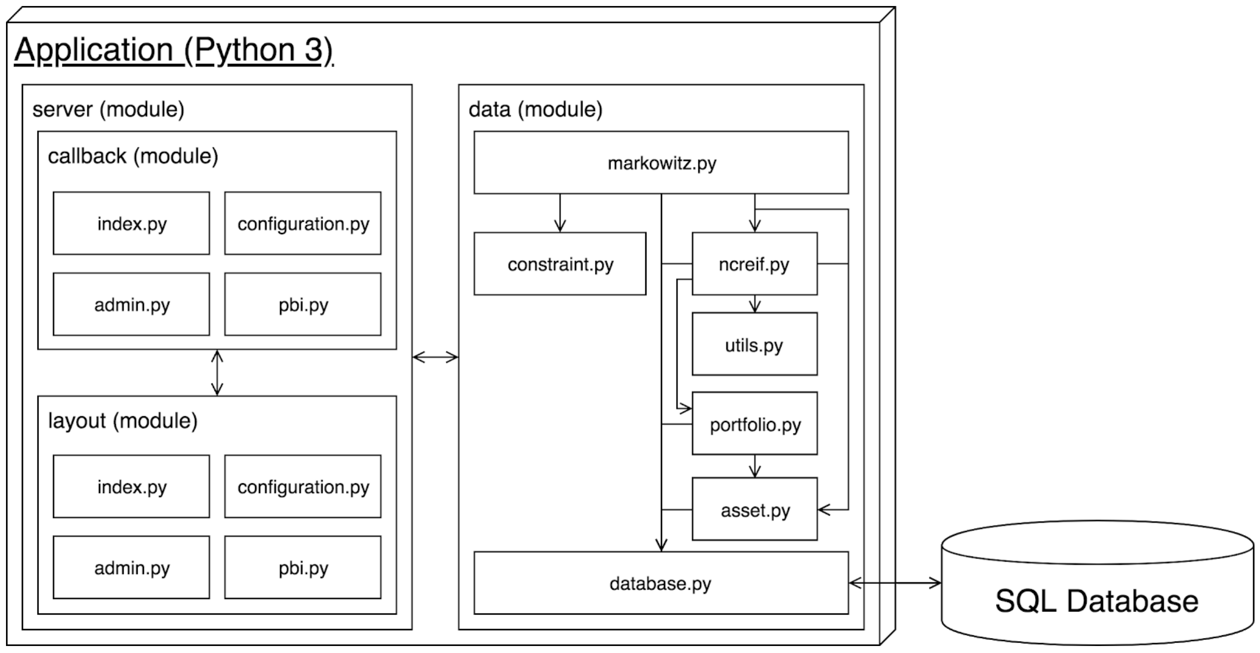


Figure 2: Application Block Diagram Utilizing Dash Framework

This shows the breakdown of our Python application and the communication between components.

2.4 Data Structures

2.4.1 Persistent Data Structures

The following tables are consumed by the Markowitz optimization algorithm.

NCREIF (table)	Asset (table)	Portfolio (table)
Market Property Type YYQ Total Return Appreciation Return Income Return End Market Value	ID Portfolio ID Market Property Type Region Expected Total Return Current Weight (Holdings)	ID Name
Optimization (table)	Optimization Run (table)	Constraint (table)
ID Portfolio ID Risk Min Risk Max Risk Step	ID Optimization ID Max Risk Actual Risk Expected Total Return	ID Optimization ID Type Key Min Max
Optimization Weight (table)		
ID Run ID Asset ID Optimal Weight (Holding)		

2.4.2 Power BI Data Structures

The following data structures (tables) will be consumed by Power BI to visualize its graphics. The first row denotes the table headers. The second row denotes an example data value.

First Report (Current vs. Optimal vs. Market Portfolio)

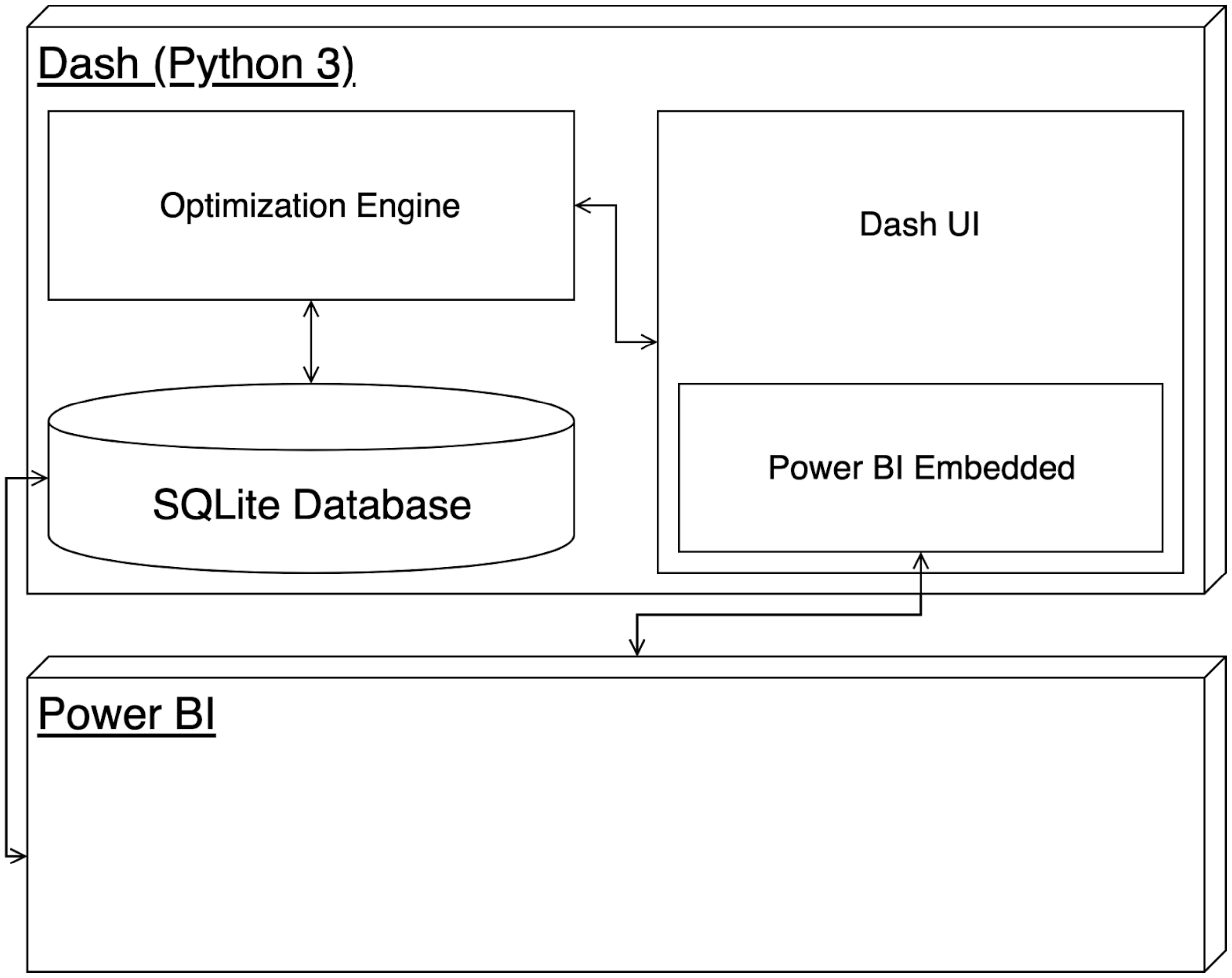
Market	**Property Type**	**Region**	**Weight**	**Market Value**	**Expected Return**	**Holding Type**
“New York”	“R”	E	0.05	50000000 or as ‘%’?	0.03	“NCREIF” or “current” or “optimal”

Second Report (Efficiency Frontier Graph)

Name	Type	Expected Return	Risk
The name of this data point with respect to its type. For a market type data point, the name may be “New York” or “Boston”.	“current_portfolio”, “optimal_portfolio”, “ncreif_portfolio”, “market”, “property_type”, “region”, “custom”	Percentage as a decimal (i.e. 0.03)	Percentage as a decimal (i.e. 0.03)

3.0 Implementation

3.1 Implementation Diagram



Our implementation closely follows our planned architecture.

3.2 Software Used

The application is written entirely in Python to handle backend components as well as the user interface. To handle user requests and routing in the application, Flask, which is a micro web framework, was used for the backend. We made use of Dash by Plotly to generate interactive interfaces for the user.

On top of the Python-based tools, we also utilized Microsoft's Power BI Embedded services hosted on the Azure cloud platform. Power BI is a business analytics solution that lets us visualize our data into dashboards and embed those dashboards in our web application. We used an open source library, adal, to handle authentication with Azure Active Directory. Through the Power BI REST API we are able to build dynamic dashboards and reports that are customizable by the user.

3.3 Rationale for Software Choices

We decided to use Python for our base language as Python offers a diverse toolset for data manipulation and has been used by most of the team before. Also, Principal has more experience with Python. Our UI is built using Dash by Plotly on top of a Flask server, both of which are Python frameworks. Our original idea was to use Plotly for the data visualizations, and thus keep all of our processing in Python. However, it was later decided that we would use Microsoft's Power BI for data visualization, as Principal has more familiarity with Power BI.

3.4 Standards and Best Practices

We used Git for version control. Git allowed us to use Merge Requests for code reviews to keep our source code clean, readable, and understandable. We followed Python standards and guidelines for our code syntax.

4.0 Testing, Validation, & Evaluation

4.1 Test Plan

There were two main facets of testing to be done in this project. The first was unit testing. Because out application used Python, the module unittest was used. The second facet of testing was user-level testing. This required us to give test application users (our client Principal) a Google form to fill out while they used our application in their day-to-day. Both of these testing plans are further detailed below.

4.2 Unit Testing

For unit testing we used unittest, a Python module. Because of the many feature requests that continued to stream consistently throughout the semester, our code coverage for unit tests was quite low.

4.3 User-Level Testing

We coordinated user testing sessions with Principal. Our app has execution instructions that, either, the tester or proctor can follow. The Google Form link is sent with a version number, so the results can be used effectively. Also, the link is active; the form can be updated on-the-fly and show changes immediately. There are sections for qualitative and quantitative responses. There are various aspects that are ranked 1-5 for each section of our app and text boxes for longer responses or suggestions. Google Form summarizes all responses by averaging any quantitative response and listing all qualitative together. Individual responses are automatically saved in a Google Sheet (or downloaded as a csv). The Frontend team has access to the Sheet, so they can immediately see each response and reference when iterating over the design. After finishing a round of testing, all responses are downloaded and saved for record keeping.

4.4 Evaluation

The evaluation of the service came from the survey that we got portfolio managers to respond to our survey (results). We also had weekly meetings with our Principal team that aided us to understand the needs of our clients.

5.0 Project & Risk Management

5.1 Task Decomposition

Task assignment was flexible throughout the project. We split our group into a Frontend and Backend team. Blake and Cole worked primarily on the Python portfolio optimizer. Lee worked primarily on the UI Dash application.Kevin worked primarily on the Power BI data visualizations. Nick worked primarily on User Testing and helped the Backend integrate PBI.

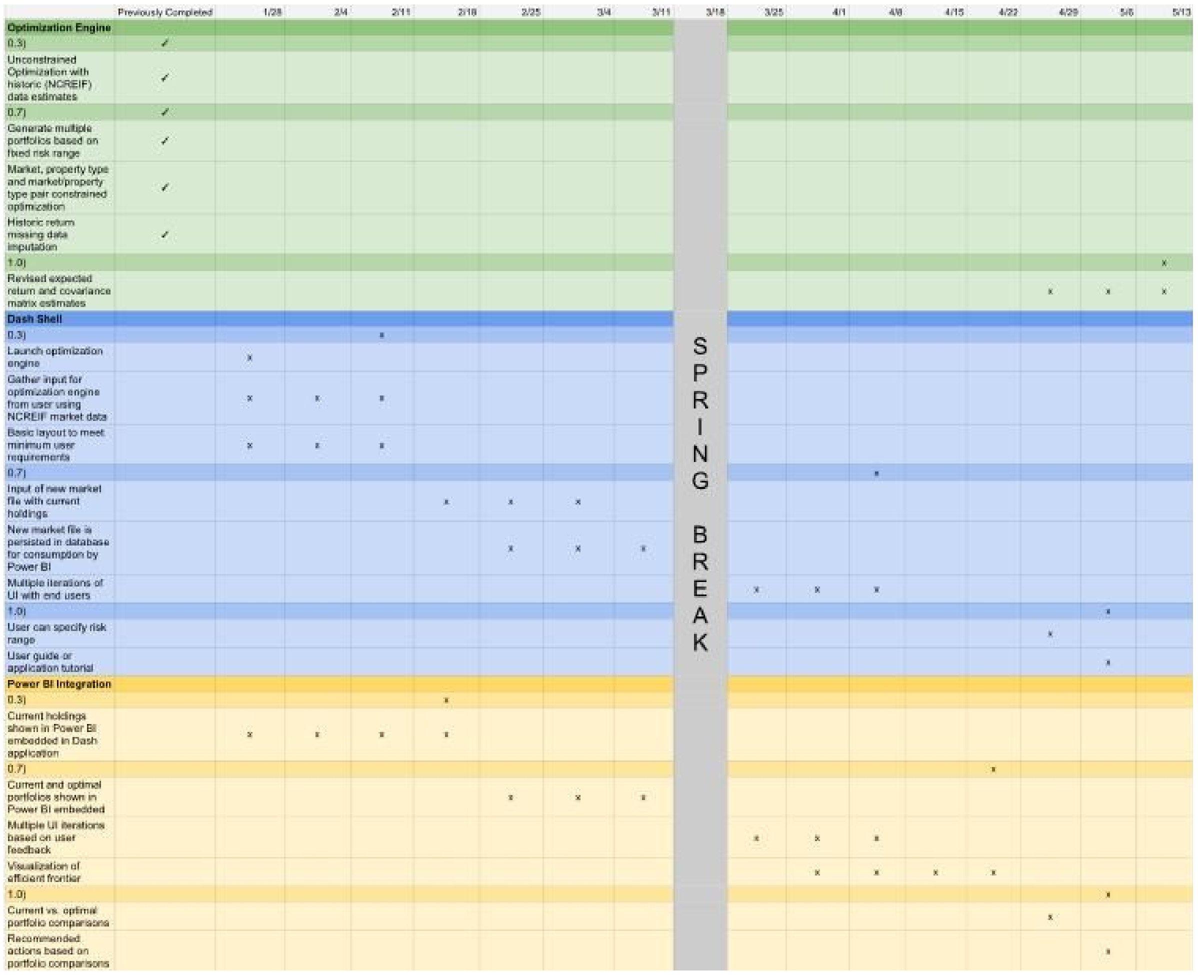
5.1.1 Roles & Responsibilities

Blake	Project Lead	Backend and Markowitz Optimization
Colton	Meeting Scribe	Backend and Communication
Kevin	Quality Control	Frontend and Power BI
Lee	Meeting Facilitator	Frontend and UI
Nick	Report Manager	Backend and User Testing

5.2 Project Schedule

5.2.1 Proposed

The following table represents our schedule for this semester. Last semester was learning about MPT and Markowitz optimization, along with planning and defining our project.

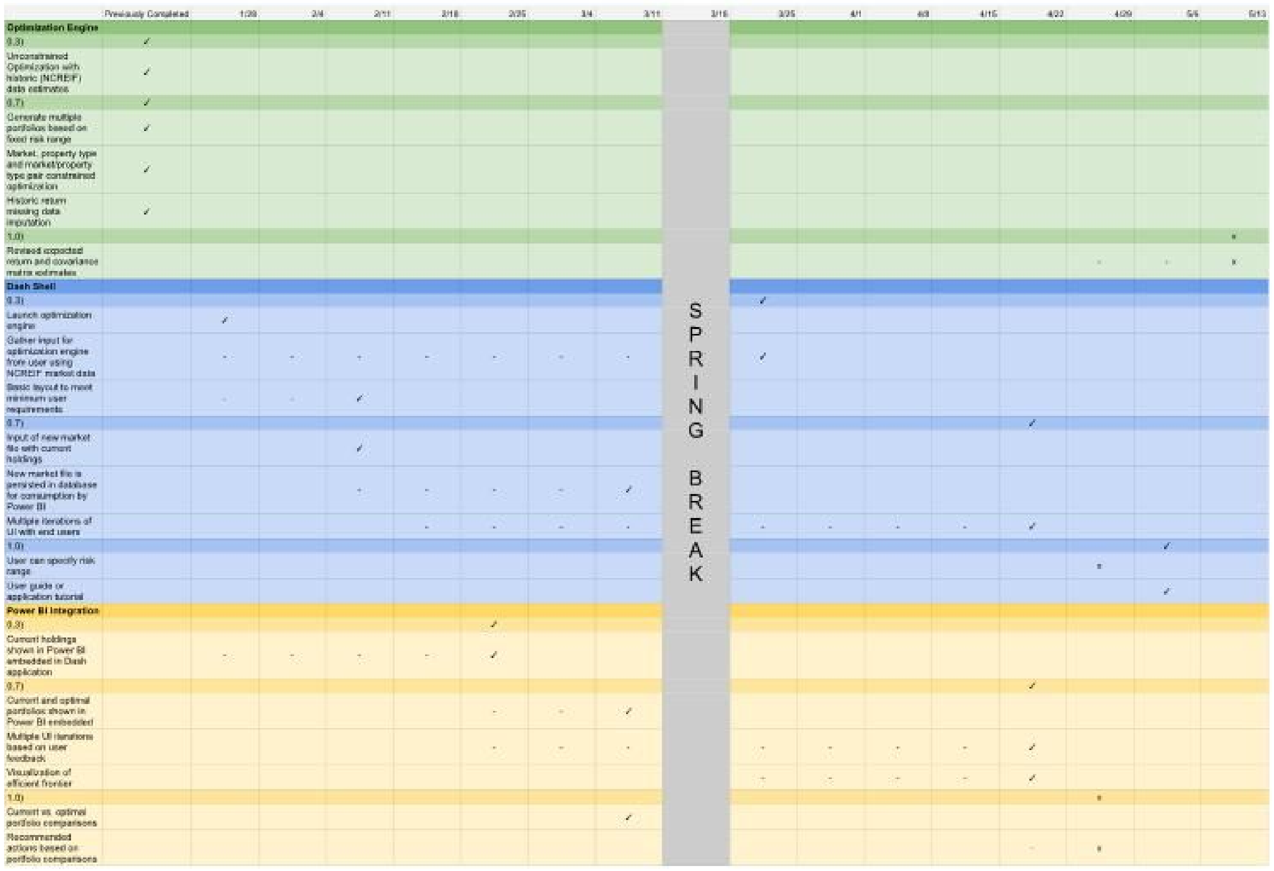


	Previously Completed	1/28	2/4	2/11	2/18	2/25	3/4	3/11	3/18	3/25	4/1	4/8	4/15	4/22	4/29	5/6	5/13
Optimization Engine																	
0.3)	✓																
Unconstrained Optimization with historic (NCREIF) data estimates	✓																
0.7)	✓																
Generate multiple portfolios based on fixed risk range	✓																
Market, property type and market/property type pair constrained optimization	✓																
Historic return missing data imputation	✓																
1.0)																	x
Revised expected return and covariance matrix estimates															x	x	x
Dash Shell																	
0.3)				x													
Launch optimization engine		x							SPRING BREAK								
Gather input for optimization engine from user using NCREIF market data		x	x	x													
Basic layout to meet minimum user requirements		x	x	x													
0.7)												x					
Input of new market file with current holdings					x	x	x										
New market file is persisted in database for consumption by Power BI						x	x	x									
Multiple iterations of UI with end users										x	x	x					
1.0)																x	
User can specify risk range															x		
User guide or application tutorial																x	
Power BI Integration																	
0.3)					x												
Current holdings shown in Power BI embedded in Dash application		x	x	x	x												
0.7)														x			
Current and optimal portfolios shown in Power BI embedded						x	x	x									
Multiple UI iterations based on user feedback										x	x	x					
Visualization of efficient frontier											x	x	x	x			
1.0)																x	
Current vs. optimal portfolio comparisons															x		
Recommended actions based on portfolio comparisons																x	

Figure 3: Proposed Project Timeline

5.2.2 Actual

Below is our actual project timeline for this semester. As shown, we completed our objectives.



	Previously Completed	1/28	2/4	2/11	2/18	2/25	3/4	3/11	3/18	3/25	4/1	4/8	4/15	4/22	4/29	5/6	5/13
Optimization Engine																	
0.3)	✓																
Unconstrained Optimization with historic (NCREIF) data estimates	✓																
0.7)	✓																
Generate multiple portfolios based on fixed risk range	✓																
Market, property type and market/property type pair constrained optimization	✓																
Historic return missing data imputation	✓																
1.0)																	x
Revised expected return and covariance matrix estimates															-	-	x
Dash Shell																	
0.3)									SPRING BREAK	✓							
Launch optimization engine		✓															
Gather input for optimization engine from user using NCREIF market data		-	-	-	-	-	-	-		✓							
Basic layout to meet minimum user requirements		-	-	✓													
0.7)														✓			
Input of new market file with current holdings				✓													
New market file is persisted in database for consumption by Power BI				-	-	-	-	✓									
Multiple iterations of UI with end users					-	-	-	-		-	-	-	-	✓			
1.0)																✓	
User can specify risk range															x		
User guide or application tutorial																✓	
Power BI Integration																	
0.3)						✓											
Current holdings shown in Power BI embedded in Dash application		-	-	-	-	✓											
0.7)														✓			
Current and optimal portfolios shown in Power BI embedded						-	-	✓									
Multiple UI iterations based on user feedback						-	-	-		-	-	-	-	✓			
Visualization of efficient frontier										-	-	-	-	✓			
1.0)															x		
Current vs. optimal portfolio comparisons								✓									
Recommended actions based on portfolio comparisons														-	x		

Figure 4: Actual Project Timeline

5.3 Risks & Mitigation

5.3.1 Potential

There are two main risks that we would expect. The first is that the frontend might not be intuitive for the users to interact with at first. This can be addressed by having multiple users both within our team and at Principal use the application and provide feedback. The second main risk is 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.

5.3.2 Actual

The main risk encountered during application development was lack of ability to develop specific UI/UX features due to the rigidity of the application framework. We were limited to using Dash, which required learning and couldn't use certain functionality that JavaScript is capable of.

Another risk we experienced was with using Power BI. Also, we were required to have a PBI Pro license connected to our account; we activated a free-trial but have run out. Our advisor, Chinmay Hegde, started a trial on his account that we are able to use.

5.3.3 Mitigation

The first main risk was mitigated by regularly showing our progress to Principal, and using their feedback to improve the UI. Improving the optimizer also required feedback from Principal and many different iterations. The second main risk was mitigated by ensuring someone on our team had an active trial PBI Pro account, and that Principal has active PBI Pro licences to use when we pass on the application.

5.4 Lessons Learned

One of the biggest lessons learned were the difficulties in meeting client needs with developer skill. The biggest difficulty in this project was the use of Dash for the frontend. One big issue with Dash is the lack of ability to add functionality to dynamically added webpage elements.

6.0 Conclusions

6.1 Closing Remarks

All in all, the project was a great success. Like with any software application, there is always room for further improvement; however, promised features have been accomplished and the final product functions as intended.

All team members stood to learn a lot about software development. This was realized through many trials and tribulations. The growth of each team member as a developer is fully visible.

6.2 Future Work

The code will be handed off to Principal for further iteration. The code base has already sprung up multiple other Principal projects in related fields as a bases on how to use the Dash framework.