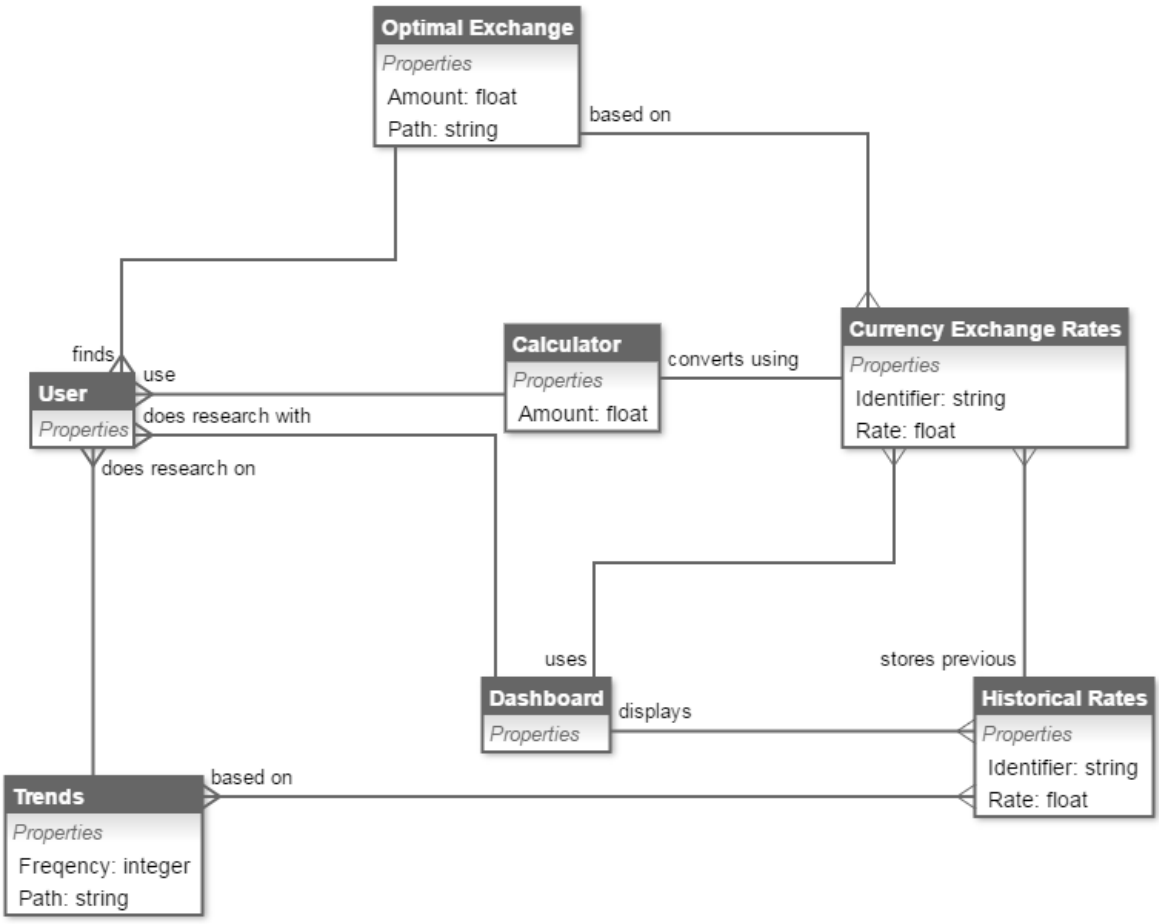
CurrencyHFT

Cha Ching

PROJECT ELABORATION

Jack Cusick
James Nakashian
Patrick Engelsman
Matthew Cordone
Ayushi Mishra

Domain Model:



The domain model depicts the conceptual class objects of the website. Each box in the domain model represents a conceptual class object, with details of the properties of its attributes encapsulated in the Properties section. Objects have relationships with other objects, which are represented as connecting lines with words describing the relationship between boxes. Connecting lines ending in a fork represent an action involving multiple objects, whereas an non-forked line represents an action involving one object.

User Stories:

User stories capture the different ways that users may interact with the website. Notes were taken on the team's discussion of each user story, and requirements for testing the implementation of the user story were generated.

#	Story	Notes	Testing
US1	User converts some amount of a starting currency to a max amount of a desired currency.	Useful for travelling to other countries. Task allotted 20 story points (40 hours)	Test algorithm correctness by running known parameters and verify results. Test a range of values. Test a range of banks and currencies. Test invalid inputs (0 value, negative value, non-numeric input).
US2	User finds a minimum amount of a starting currency required to exchange for some amount of a desired currency	Useful for buying things from foreign countries. Essentially the same as US1, but performing conversion in reverse. Task allotted 2 story points (4 hours)	Test algorithm by specifying amount of target currency and verify the results. Test various different currencies and banks. Test invalid inputs (0 value, negative value, non-numeric input).
US3	User finds best exchange path without specifying an amount	If the user doesn't enter an amount of either currency (US1/US2), the optimal exchange path should still be found. Task allotted ½ story points (1 hour)	Test algorithm with no amount specification in both starting and ending currency, verify results. Test various different currencies and banks.
US4	User views trends on profitable exchange paths and other historical data	If we provide trends based on frequency, how do we define what "most frequent" is? Need to store trends in DB. Task allotted 13 story points (26 hours)	Test updates/additions to database. Test trend finding and verify results. Test graphic display of trends and historical data. Test different starting and ending dates for trends.
US5	User excludes a currency/bank from the search.	Considered excluding based on location, but decided it was too difficult to implement. Increases feasibility of user actually following exchange path,	We will implement the main path finding algorithm (the main algorithm testing will have already been done). Test the algorithm with no exclusions, then test again using the same inputs with different

		since they may limitations on what banks/currencies are accessible to them. Task was given 2 story points (4 hours) for completion.	exclusions and verify results. Test with various different exclusions. Test excluding all the options at once.
US6	User limits the maximum number of exchanges that can be used to find the most efficient path.	This would be useful for a situation where there would be fees on the transactions, or user just doesn't want to make a lot of transactions. Task was given 1 story point (2 hours) for completion.	Test finding the optimal path using the main algorithm with no limit, then test again with limiting the maximum number of nodes in the path to one less than the previously found most efficient path, and make sure it is still returning a successful efficient path. Test high number of allowed transactions. Test low number of allowed transactions. Test invalid inputs (negative or 0, non-number).
US7	User is traveling and finds the best way of exchanging currency within a specific bank.	The implementation would include the ability to select only a certain bank. Task was given 1 story point (2 hours) for completion.	Test the main algorithm using the exclusion limiting the use to only the exchange rates from one bank and ensure it returns an efficient exchange path. Test different banks selected.
US8	User calculates a direct exchange between a starting and ending currency.	Changed from feature on Arbitrage page to independent Calculator tool. Calculator present on Navigation bar, accessible from any part of the site. Task was given 5 story points (10 hours) for completion.	Test correctness of calculation by entering known parameters and verifying results. Test front-end display of calculator tool. Test with various banks and test with no specified bank. Test with no amount (use exchange rate on a single unit of starting currency as default)
US9	User researches the forex market	Forex dashboard, with graphs on exchange rates over time and tickers (like stock market info). User can select which currency exchange to examine. Task was given 13 story points (26 hours) for completion.	Test accuracy of graphs of several different currency exchange rates. Test selecting different tickers for view. Test front-end display of graphs, tickers.

Supplemental Specifications:

The development team generated a list of requirements of the project using the FURPS model of Functionality, Usability, Reliability, Performance, and Supportability. Each requirement was analyzed from the point of view of the users and the development team. For each requirement, appropriate tests were determined, and the requirement was ranked by priority as “must”, “should”, or “could”.

Functionality Requirements			
#	Requirement	Characteristics	Priority
F1	Users are able to find optimal paths between two currencies with no starting or ending amount specified	User can find out what the best path between any two currencies is even if they do not specify a value to convert. This feature will be tested by developers by using the feature with currencies where the optimal path is known to the tester and verifying the result	must
F2	Users are able to specify an amount of the starting currency to convert and find how much of the ending currency is produced	User can find out how much of the ending currency they would obtain using the optimal path when inputting an amount of the starting currency. This feature will be tested by developers using currencies where the optimal path and in turn the resulting currency amount is known to the tester, and verifying the result	must
F3	Users are able to specify an amount of the ending currency needed and find how much of the starting currency is required	User can find out how much of the starting currency they would need using the optimal path to obtain a certain ending currency value. This feature will be tested by developers using currencies where the optimal path and in turn the required currency amount is known to the tester, and verifying the result	must
F4	Users are able to exclude currencies and/or banks from the search	User can avoid using certain currencies or banks in their exchange if they choose and still find the optimal path within those parameters. This feature will be tested by developers using a two	should

		currencies where the optimal path is different when certain banks and currencies are excluded and verifying that the banks/currencies are excluded	
F5	Users are able to set maximum number of transactions in path	User can decide the number of exchanges they are willing to make and the optimal path will reflect their choice. This feature will be tested by developers using two currencies that have different optimal paths when the max length is adjusted and verifying the results	should
F6	Users are able to use Currency Calculator to find direct exchange rates between two currencies in a certain bank	User can compute a direct currency conversion at a specific bank using the Currency Calculator. This will be tested by performing conversions with the Currency Calculator and verifying the results.	must
F7	Users are able to use Currency Calculator to find direct exchange rates between two currencies with no bank selected	User can compute a direct currency conversion based on the average exchange rates of all applicable banks using the Currency Calculator. The average rate is used if no bank is specified. This will be tested by performing a conversion without specifying a bank and verifying the results.	must
F8	Users are able to find historically profitable exchange paths within a selected timespan	Users can submit starting date and ending date and view graphical representations of the most efficient and most profitable exchange paths in that timespan. The team will test this by submitting various dates and verifying that the information returned is accurate.	should
F9	Users are able to find the current most profitable or most efficient exchange paths	User can input a starting currency and/or ending currency. User will be given the most efficient exchange rate paths. The team will test this by trying different starting and ending currencies and verifying the results.	must
F10	Users are able to view	User can navigate to a Forex	must

	graphs of exchange rates on the Forex Dashboard	Dashboard page and immediately view an interactive graph displaying the most common currency exchange's historical data (EUR/USD). User can select a different currency pair, which changes the graph to display the inputted currency exchanges historical data. This feature will be tested by selecting different currency pairs and verifying the result.	
F11	Users are able to view specific tickers on the Forex Dashboard	User can navigate to a Forex dashboard page and immediately view ticker data for the 18 most popular (liquid) currency pairs. These pairs were selected by Forex market makers due to their popularity in the market. The ticker displays information on the current exchange rate. A more detailed view of any currency pair can be explored by selecting an exchange, which is linked to the interactive graph. User can select a particular bank to update the tickers to reflect that bank's data. The feature will be tested by verifying that the displayed ticker info is accurate updates to selected banks.	should
F12	Users are able to add multiple charts to the Forex Dashboard	User can click a button to add a new chart on the Forex Dashboard so they can view multiple charts at once, up to a limit. User can select what exchange to view on each chart. This will be tested by adding multiple charts and verifying each chart displays correctly.	could

Usability Requirements			
#	Requirement	Characteristics	Priority
U1	Navbar stays at the top of every page	Navbar always floats at the top of the page, no matter what page the user is on. The team will test this by going to each page and scrolling, verifying that the navbar remains at the top of the	must

		page and is functional.	
U2	Navbar provides links to other pages and access to Currency Calculator	The Navbar provides quick links to pages and features, and provides access to the Currency Calculator. The team will test this by clicking each link on the toolbar to verify that the site goes to the correct page and/or opens the Currency Calculator.	must
U3	Users are able to intuitively navigate and use all features of the website. Interface elements should be easy to understand and easy to learn.	Users should be able to navigate the website and reach desired pages and features without guidance from the development team. Users should be able to understand and use interface element to achieve desired goals (within the functionality of the site). Performing desired tasks on the site should be intuitive. This will be tested by conducting a usability tests. Independent users will be asked to navigate and interact with the site. Users will be observed by the team and will also return feedback in the form of a survey.	should
U4	Interface action and elements are consistent	User flow and site display should be consistent when performing similar tasks. (i.e. calculations using various constraints should follow consistent prompts to user and generate consistent graphic representations) GUI elements should be consistent throughout the site. The team will test this by verifying that all elements are visually consistent. User flow consistency will be tested with usability tests described above in U3. Users will provide feedback regarding consistency in work flow and also visual consistency.	must
U4	Screen layout, color, and brand are aesthetically pleasing	Users should be attracted to the site's appearance. The appearance should appeal to the target audience. The look and feel of the site should represent the brand and culture of Currency HFT. We	should

		will test this with usability tests described above in U3. Users will provide feedback regarding appearance in the form of a survey.	
U5	Website works on common browsers	The site is fully functional on popular modern browsers, including (but not limited to) Firefox and Google Chrome. The team will test this by loading the site on clean installation of the latest releases of the supported browsers and verifying that all features are functional and display correctly.	must
U6	Website works on IOS and Android smartphones	A mobile version of the site is fully functional on IOS and Android smartphones. The team will test this by loading the site on their IOS and Android smartphones and verifying that all features are functional and display correctly.	must

Reliability Requirements			
#	Requirement	Characteristics	Priority
R1	The website loads when user goes to its URL	Users are brought to the homepage of the site when they enter it's URL while on RPI intranet. The team will test this by entering the URL on a variety of devices connected to RPI's LAN and various Wifi networks.	must
R2	Database is maintained, updated daily, and website should reflect updates	The database of currency exchange information supporting the site is kept in good condition, backed up in multiple locations, able to be rolled back to previous states, updated daily, and consistently be reflected by the site. The team will test this by ensuring that the database is backed up and has rollback features, verifying that the update systems work, and testing connection to the database from the site. The team will also test to ensure	must

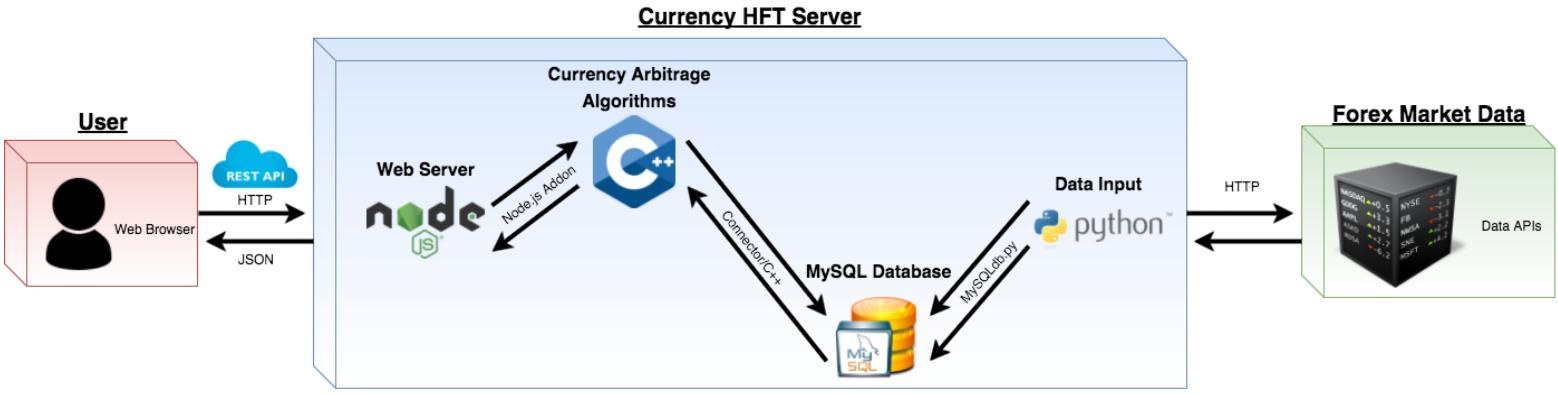
		the site reflects changes to the database.	
R3	Server is fully functional and consistently supports website	The server that hosts the site meets all the requirements for running the frontend and backend of the application, and does not experience egregious downtime. The team will test this by testing each feature of the site and examining performance metrics of the server.	must

Performance Requirements			
#	Requirement	Characteristics	Priority
P1	The website loads quickly	Each page of the website fully loads quickly, within 1s at most. The team will test this by loading the site on different devices, browsers, and Internet connections and measuring the load time.	should
P2	The website can handle multiple simultaneous requests	The site is able to serve multiple users making different or the same requests at a time. The team will test this by manually having multiple users use the site at once, or by creating tests to automate this behavior, and measuring performance metrics.	should
P3	Features return results quickly	All features return their results in a reasonable amount of time. This may be dependent on the particular feature. The team will test this by measuring the response times of submitting requests to each feature.	should

Supportability Requirements			
#	Requirement	Characteristics	Priority
S1	Code is understandable by	All members of the development team	should

	all members of development team	are able to understand code written by each other member. The team will test this by having each member review code written by other members and rating their understanding of the code.	
S2	Code follows a documented coding standard	All code follows a standardized and well documented coding standard. The team will verify this by reviewing all code for violations of the coding standard.	must
S3	Code follows object-oriented design principles	All code (where applicable) follows object-oriented design principles, as taught by the course and other courses at RPI. The team will verify this by reviewing all code and ensuring that object-oriented design principles are followed.	must
S4	Code is testable	All code is written in a manner that allows for testing and verification of results. The team will ensure that this is the case by writing unit tests for the code and examining code coverage of the tests.	must

Deployment Diagram:



The above deployment diagram depicts the technologies and interconnections utilized on the Currency HFT server.

Node.js is an open-source JavaScript runtime environment for developing server tools. On the Currency HFT server, it is used as the primary web server. Node.js was chosen over other web server technologies, like Apache and Django, due to it's ease of implementation and inherent RESTful API. This allows the front-end and back-end to easily pass data wrapped in JSON objects.

While Node.js is great for RESTful APIs, it is not optimal to use for computationally intensive algorithms. For this, the team uses C++. C++ was chosen over other languages primarily for it's speed. Forex market data is plentiful and graph computation is expensive. C++ is also commonly known by the team and is ideal for object oriented programming.

To connect the C++ algorithms with the Node.js web server, a Node.js addon is used. This involves using a Node.js addon build tool called node-gyp. Node-gyp compiles C++ into binaries that can function as libraries for the Node.js server. Objects are passed between Node.js and C++ through the V8 JavaScript Engine.

The back-end C++ algorithms retrieve market data from a MySQL database, hosted on the server. To connect to the database, an Oracle provided library called Connector/C++ is used. MySQL was chosen over other databases because of its popularity and extensive documentation. A relational database is used for increased speed and because stock market data is well formated.

Data is fed into the MySQL database through python scripts. Various python scripts query Forex market data APIs and populate the MySQL tables with historical and real-time data. MySQLdb.py is a python library for connecting to SQL databases. Python was chosen for it's ease of implementation and extensive libraries for API requests and web scraping (if needed).

Use Cases:

The development team generated use cases based on the goals of actors and the corresponding system responses out the website.

Use Case: FindOptimalPath	
Identifier	UC1
Description	The FindOptimalPath use case models a User searching for the optimal sequence of exchanges between two currencies
Actors	User
Preconditions	User is on the “Arbitrage” page
Flow of Events	1. The use case starts when User scrolls down to the “Optimal Exchange” section 2. The site displays a list of options for “Starting Currency” and “Ending Currency” with default values selected and input fields for values, and a “More Specifications” expandable menu 3. User selects a “Starting Currency” and “Ending Currency” 4. If User inputs a starting value: 4.1. The ending value is made unselectable Else if User inputs an ending value: 4.1. The starting value is made unselectable 5. If User opens the “More Specifications” menu: 5.1. The site displays options for “Banks”, “Currencies”, and an input field for “Max Number of Exchanges” 5.2. User selects which banks to use in exchange 5.3. User selects which currencies to use in the exchange 5.4. User inputs max number of exchanges 6. User submits search query to the site 7. If the inputted values are valid: 7.1. Site displays the results Else: 7.1 Site displays an error message describing the invalid input 7.2 The use case returns to step 2
Postconditions	The site displays the optimal path given the inputted specifications in graphical form, along with the value received at the end of exchange or value to start with if the corresponding starting or ending values were specified.

Use Case: UseCurrencyCalculator	
Identifier	UC2
Description	The UseCurrencyCalculator use case models a User converting starting currency to a desired currency using current exchange rate
Actors	User
Preconditions	User is on any page
Flow of Events	1. The use case starts when User selects the “Calculator” in the Navigation bar 2. The site displays the “Calculator” on the screen, with option lists for “Starting Currency”, “Ending Currency”, and “Bank” with default values selected, and an input field for “Amount” 3. User selects desired “Starting Currency” 4. User selects desired “Ending Currency” 5. User selects desired “Bank” 6. User inputs desired “Amount” 7. User submits calculation request to the site 8. If an invalid input is detected: 8.1. Site displays an error message describing the invalid input 8.2. The use case returns to step 2 Else: 8.1. Site displays converted currency amount
Postconditions	The site displays the calculated end currency using current exchange rate between the selected start and end currencies at a selected bank. If bank is not specified, an average exchange rate is used.

Use Case: FindProfitableTrends	
Identifier	UC3
Description	The FindProfitableTrends use case models a User viewing historical trends of most efficient or most profitable currency exchange paths.
Actors	User
Preconditions	User is on the “Arbitrage” page.
Flow of Events	1. The use case starts when User scrolls to the “Historical Trends” section 2. The site displays option lists for “Starting Date” and “Ending Date” with default values selected

	3. If User changes “Starting Date”: 3.1. “Ending Dates” before the selected “Starting Date” are made unselectable 3.2. If selected “Ending Date” is made unselectable, the default value is restored 4. If User changes “Ending Date”: 4.1. “Starting Dates” after the selected “Ending Date” are made unselectable 4.2. If selected “Starting Date” is made unselectable, the default value is restored 5. User submits query to the site
Postconditions	The site displays the top efficient or profitable currency exchange paths that appeared during the selected time range

Use Case: FindProfitablePaths	
Identifier	UC4
Description	The FindProfitablePaths use case models a User searching for the current most efficient currency exchange paths
Actors	User
Preconditions	User is on the “Arbitrage” page
Flow of Events	1. The use case starts when User scrolls down to the “Profitable Exchanges” section 2. Site displays options lists for “Starting Currency” and “Ending Currency” with default values selected 3. User selects a “Starting Currency” 4. User selects an “Ending Currency” 5. User submits query to the site
Postconditions	The site displays graphical representations of most efficient paths using current exchange rates using the selected parameters.

Use Case: OpenForexDashboard	
Identifier	UC5
Description	The OpenForexDashboard use case models a User doing research on specific tickers in the forex market

Actors	User
Preconditions	User is on the “Dashboard”
Flow of Events	1. The site displays ticker data for the most liquid currency pairs as outlined by the forex market 2. The site displays an interactive chart showing the historical exchange rate of the most popular currency pair 3. The site displays a “Bank” option list with a default value selected and option lists for “Starting Currency” and “Ending Currency” with default values selected 4. If User selects a different “Bank”: 4.1. The site displays the selected bank’s ticker data 4.2. The site updates the current exchange chart to use the selected bank’s data (but the exchange remains the same) 5. If User selects different starting or ending currencies: 5.1. The site updates the current exchange chart to display the new exchange’s data
Postconditions	The site displays chart of selected currency exchange’s data and selected bank’s ticker data.

Use Case: NavigateWithNavbar	
Identifier	UC6
Description	The NavigateWithNavbar use case models a User traversing the site using the Navigation bar
Preconditions	User is on any page of the site
Flow of Events	1. The use case starts when User selects the Navigation bar 2. If User selects a link on the Navigation bar: 2.1. The site loads the linked page 3. If the User selects “Calculator” 3.1. The site displays the “Currency Calculator”
Postconditions	The site loads the desired page, or displays the “Currency Calculator” while remaining on current page.

Use Case: UpdateData	
Identifier	UC7

Description	The UpdateData use case models the time of day reaching the point when currency exchange data is to be updated
Actors	Day End
Preconditions	N/A
Flow of Events	1. The use case starts when Day End reaches 12:00AM EST 2. Day End sends a request to the system to update the currency exchange data 3. If the system successfully updates the data: 3.1. The system returns a “success” signal to Day End Else: 3.1. The system returns a “failure” signal to Day End
Postconditions	Day End receives a signal detailing if data update was successful or not

Work Breakdown Structure:

The development team generated a list of individual tasks that must be completed to fulfill every use case, as well as other project deliverables.

Use Case or Deliverable	Individual Tasks
UC1	1. Create algorithm to find optimal path between two currencies given certain constraints 2. Create front-end user interface for “Optimal Exchange” section 3. Write Javascript to generate dropdown lists 4. Create graphical view of optimal path 5. Write logic to verify inputs are valid
UC2	1. Create front-end user interface for Calculator component on Navigation bar that can be accessed at any time (using Javascript) 2. Write logic to compute direct exchange between two currencies 3. Write logic to find average exchange rate considering all banks that can facilitate that exchange
UC3	1. Create table schema to store historical trends 2. Create front-end user interface for “Historical Trends” section 3. Implement logic to retrieve trends based on dates selected by user 4. Create graphical views of trends for display
UC4	1. Write code to reuse algorithm to find top efficient paths 2. Create front-end interface for displaying “Profitable Paths”
UC5	1. Create database table housing currency pairs historical data 2. Create back-end Node.js view to return specific currency exchange data (both ticker and historical) 3. Create front-end Forex webpage, which has currency pair textbox and bank drop-down 4. Create front-end AJAX calls to get data based on textbox and drop-down. Also intial AJAX calls for 18 most popular currency pairs and default EUR/USD historical data. 5. Integrate javascript charting library for interactive graph. 6. Use return from AJAX calls to populate graph and ticker table.
UC6	1. Create wireframes and mockup 2. Create fully functioning navigation and site routing 3. Create front-end Navigation bar that always stays on screen at fixed location
UC7	1. Create trigger to update data daily 2. Design webscrapers or utilize API calls to retrieve updated data

	3. Write SQL script to store historical data 4. Write SQL script to update data to current values 5. Write code for the application to pull currency exchange rates from the database
Inception	1. Write vision statement 2. Create user scenarios 3. Create Initial project schedule 4. Create Initial breakdown structure
Elaboration	1. Create Domain Model and description 2. Write User Stories, notes on each story, and testing considerations 3. Establish Supplemental Specifications (Functionality Requirements, Usability Requirements, Reliability Requirements, Performance Requirements, and Supportability Requirements) and establish priority 4. Create Use Cases and preconditions, flow of events, and post conditions 5. Create Work Breakdown Structure 6. Update project schedule made in Inception
Stakeholder Product Review #1	1. Design demo of first set of features 2. Create design Sequence diagrams 3. Create static Class diagrams 4. Create CRC cards for major classes of project 5. Create a design approach that depicts any design patterns that are going to be used
Stakeholder Product Review #2	1. Beta release of at least 80% of functionality 2. Ensure source code is well commented 3. Establish a detailed testing plan that includes quality goals and preliminary results 4. Summarize code review performed
Transition	1. Ensure final release is working and well-tested 2. Write summary of final test results 3. Create and rehearse final presentation
Testing	1. Write unit tests 2. Write integration tests 3. User acceptance tests 4. Gather user feedback
Peer Reviews	1. Everyone completes and submits their peer reviews

Updated Schedule:

The schedule from the Inception document was updated to reflect new features added to the project and current predictions of the work breakdown.

Inception

1/26 - 1/30

- *Set up means of communication*
- *Established meeting times*
- *Decided to create web app*
- *Discussed rough overview and goals of project*

1/31 - 2/2

- *Decided to use C++ for backend*
- *Created Peer Review rubric*
- *Chose Jack to be the "Costumer" of the project*
- *Set up Git milestones*

2/3 - 2/6

- *Set up a web server that uses Node.js*
- *Investigated data sources*
- *Began work on user stories*
- *Asked an industry contact for supplemental material*

2/7 - 2/13

- *Looked into use cases, discussed features*
- *Sketched out plans for front end design*
- *Discussed project timeline*
- *Bought currencyhft.com domain name*
- *Wrote server documentation*

Elaboration

2/14 - 3/6

- *Complete wire frames for application*
- *Establish specifications for the database and other backend functionality*
- *Finalize list of data sources*

Construction

***Iteration 1: Interconnectivity** (40 story points)*

2/27 - 3/6

- *Local servers setup by team members (5)*
- *Frontend can hit endpoint and display received response (10)*

- *Backend endpoints for frontend to send and receive information (15)*
- *Database API for the use by the backend to request data (10)*
- ***Demonstration:***
    - *User visits www.currencyhft.com*
    - *User clicks "See all" button*
    - *Screen echos database table from server*

Iteration 2: Basic Frontend & Data Collection (50 story points)

3/7 - 3/13

- Database manually populated with mock or historical data (5)
- Historical forex market data downloaded (5)
- Create mockups to guide front end development
- Basic UI for displaying database's exchange rates, tabular and graphical (15)
- Requests to different marketplace APIs and write web scrapers if necessary (20)
- Investigate and choose a shortest path algorithm (5)
- **Demonstration:**
    - User selects currency exchange from drop down list
    - User clicks submit
    - Screen displays exchange rate data in tabular and/or graphical view

Iteration 3: Algorithm, Database API/Collection, & More Frontend (60 story points)

3/14 - 3/20

- Implement and test shortest path algorithm (30)
- Set up continuous updating of data (5)
- Finalize backend calls to database (5)
- Add UI for the rest of frontend functionality (Calculator, Dashboard, Arbitrage) (20)
- **Demonstration:**
    - User navigates to Arbitrage page
    - User selects currency exchange from drop down list
    - User inputs value of starting currency
    - User clicks submit
    - Screen displays currency exchange path and ending value in tabular and/or graphical view

Iteration 4: Modified Versions of Algorithm & Frontend Visuals (55 story points)

3/21 - 3/27

- Modify algorithm to suit the different server requests (input ending value, exclude selected currencies/banks/locations, find profitable paths) (25)
- Add visuals to frontend for the web page functions (i.e. best exchange, profitable paths) (25)
- Database to store trends of efficient/positive paths (5)
- **Demonstration:**

    - User navigates to Arbitrage page
    - User selects currency exchange from drop down list
    - User inputs value of ending currency
    - User selects currencies/banks/locations to exclude from drop down list
    - User clicks submit
    - Screen displays currency exchange path and starting value needed in graphical view

Iteration 5: Calculator, Dashboard, & Frontend Styling (35 story points)

3/28 - 4/3

- Implement display of the Calculator functionality on backend (5)
- Implement display of the Dashboard functionality on backend (10)
- Add styling to make web page more user friendly and nice to look at (20)
- **Demonstration:**
    - Walk through new GUI of web page
    - User navigates to Calculator and performs a currency conversion
    - Screen displays converted currency
    - User navigates to Dashboard and selects an exchange rate
    - Screen displays exchange rate information graphically

Iteration 6: Finish Trends & Finish Version of Frontend (35 story points)

4/4 - 4/10

- Implement display of profitable path trends on backend (15)
- Implement display of current profitable paths on backend (10)
- Put finishing touches on initial build (10)
- **Demonstration:**
    - User navigates to Arbitrage page and selects “Trends” on profitable paths
    - Screen displays profitable path trends information graphically
    - User navigates to Arbitrage and selects current profitable paths
    - Screen displays profitable path information

Iteration 7: Documentation, Unit Testing and End to End Testing (40 story points)

4/11 - 4/17

- Finish Documentation for all parts of project (20)
- Bug fixing jam session (10)
- Complete unit and end to end testing of application (10)
- **Demonstration:**
    - Portray user scenario 1: traveler finding efficient exchange rate in one marketplace
    - Portray user scenario 2: financial professional investigating forex arbitrage opportunities
    - Utilize tools during one or both user scenarios

Transition

4/18 - 4/30

- Create and deliver final presentation
- Make changes based on feedback and testing
- Finalize and deploy application
- Finalize documentation

Contribution Summary:

All team members contributed to editing, reviewing, and approving the Elaboration deliverable for final submission. Breakdown of individual contributions is as follows:

Jack Cusick
Create and wrote Deployment Diagram, Use Case OpenForexDashboard. Contributed to Updated Schedule. Filled in F10 and F11 in the Functionality Requirements section and UC5 in the Work Breakdown Schedule.

James Nakashian
Section Descriptions, Use Case FindProfitableTrends and UpdateDate, Editing of Use Cases, URPS Supplemental Specifications, helped sketch out Domain model, helped update Schedule, User Story 9

Patrick Engelsman
Domain model, Use Case FindOptimalPath, related Functional Requirements, Work Breakdown, updated schedule

Matthew Cordone
User Stories 5 - 7; Use Case FindProfitablePaths and related Functional Requirements and Work Breakdown, Document formatting.

Ayushi Mishra
User Stories1 - 4, 8; Usability Requirements; Supportability Requirements; Use Cases UseCurrencyCalculator and NavigateWithNavbar and related Functional Requirements and Work Breakdown; Status Report; Document formatting

Status Report:

CurrencyHFT has been tracking issues, deliverables, and milestones using git. We have looked into the legality of web scraping and have decided to use web scraping for our data collection. We have compiled a preliminary list of data sources.

We have connected server-side C++ using a Node.js addon. We have started work on our database architecture and implemented a preliminary database. We have connected our C++ backend with MySQL database. We have ran some tests to confirm that objects are successfully being passed between Node.js and C++.

We have created a domain model, and determined that we will be using an adjacency matrix for efficiency of computation. We have created User Stories, and determined out Use Cases. We solidified the web page's functionality and organization. We have created wireframes for site pages.