BIRDIE

Crypto-Currency Trading Application

Robert Chung; Naveen Janarthanan; Quan Nguyen; Rafael Valdez

Executive Summary

Hoth is a group of software engineers at Seattle Pacific University (SPU). The Hoth team is comprised of four members: Robert Chung (lead), Rafael Valdez (lead), Quan Nguyen (librarian), and Naveen Janarthanan (recorder). We are planning on developing a web application, named Birdie (subject to change) that will help cryptocurrency – specifically alt-coin – traders with managing their portfolios. Our initial clients are two admins of MiniWhales, Kevin Dimaano and Kris Estilloso.

Birdie will help traders by incorporating automation and will serve as a central location for logging/tracking. Basically, making trading much easier, by allowing the application to make trades for you, based on what you tell the application to do. The trader's information will be automatically synced with major cryptocurrency exchanges and allow the user for more flexibility and control in trading.

The following sections will dive deeper into details of our proposed system.

Team Information

Team Name: Hoth

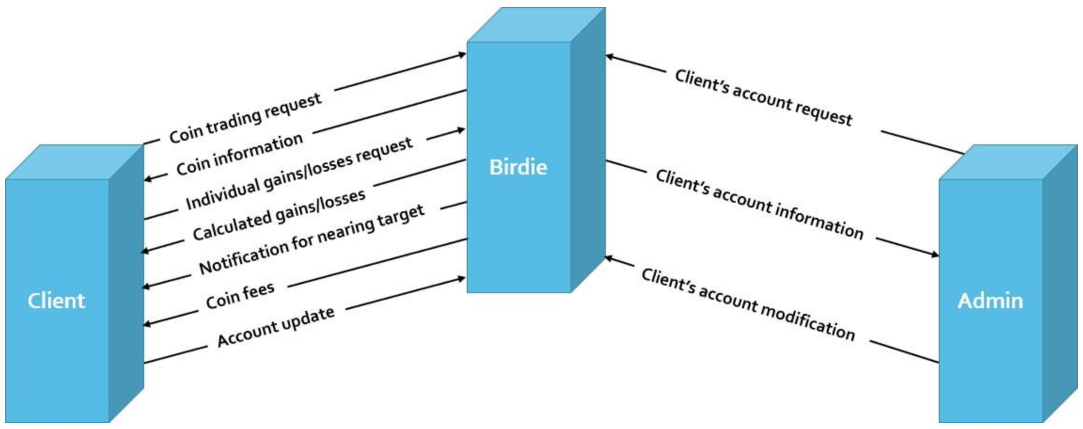
Team Members:

Name	Email	Cell Phone
Robert Chung	Chungr1@spu.edu	206-743-2198
Naveen Janarthanan	njanarthanan@spu.edu	206-422-5043
Quan Nguyen	nkmquan@spu.edu	425-974-9078
Rafael Valdez	Valdezr1@spu.edu	425-321-8618

Contents

1. Project Description

1.1 System Description and Rationale



Context Diagram of Birdie

The vision of the Birdie project is to create a financial system for cryptocurrency management. It is a web application that allows users to trade their cryptocurrencies more easily. Another major features include automatic buy/sell based on user's inputs, earnings/losses calculation, and trading activity log.

Birdie is designed for the next generation of new and experienced traders. Its intuitive interface, accurate response, and highly-secured database help users to trade effectively and efficiently. Traders can set up their buying/selling targets. And Birdie will do all of the buy/sell commands when all conditions are met. Traders will have more time to plan their strategies instead of paying attention to the screens waiting for the right moments to make the buy/sell decisions.

1.2 System Scope

To improve both efficiency of trading, tracking, and user interaction with crypto-currency, Hoth envisions a web application that can be created authenticating users to allow individual tracking, trading, and limitations of gains and losses, in certain crypto-currency markets.

This web application will be available to anyone with access to the internet and individuals will be able to purchase, trade, and track their coins. Users will be allotted to connect to their Binance accounts to attain their information and allow the web application to track the currencies they are involved in or are watching. Moreover, the web application will allow the users to quickly view their losses and gains over a day, week, month, and year. This application must be mobile responsive, allowing users to access the application via mobile device and tablet.

The system is will potentially be extended to be available as a mobile application if time allows, but this functionality is not a guarantee. The application will also attempt to implement real-time currency tracking; however, this functionality is not a guarantee either. Furthermore, by the first iteration, we won't be implementing flash-dump protection. Test mode will also be looked at in later prototypes.

1.3 System and Development Constraints

System and Development Constraint	Description
Binance Crashing	We are using Binance's API, so Binance crashing would cause our application not to work.
Real money at risk	We are dealing with actual money so there needs to be a high level of security.
Four people working on something that could be potentially big	Our team is small, and this project may require more people.
No budget, as it is a school project	We have no funding for the project, so it may be hard to do all we want to do.
Security Concerns	Since we are storing sensitive data, our security has a potential to be compromised.
Internet Connection	Our application will require internet connection to create the API calls.

Project and Team Proposal

PART 1 – TEAM

Team Name

Hoth ____

Team Members

(Initial) Leader:

Robert Chung, Rafael Valdez

(Initial) Recorder:

Naveen Janarthanan

(Initial) Librarian:

Quan Nguyen

Members:

Robert Chung, Naveen Janarthanan, Quan Nguyen, Rafael Valdez

PART 2 – PROJECT

Working Name of System

Birdie

Project Sponsor

- Name: Kevin Dimaano
- Email: rkdimaano@hotmail.com
- Organization / Department: MiniWhales
- Phone: 425-501-9065

Project Scope

We plan on building a web-based API that allows users to trade and track crypto-currencies conveniently. This system will interact with a given exchange such as Binance, Bittrex, or GDAX to:

- Tracks all of the buy and sell activities of users.
- Track user's positions in the market (mostly alt-coins)
- Records values of the currencies overtime.

- Automatically makes buy/sell decisions based on users' inputs.
- Possible machine learning integration.

Why you have chosen this project

Cryptocurrencies are interesting and will most likely become a part of the future. This Is an opportunity not only to learn more about cryptocurrencies, but also to provide tools that will help traders (real-world application). In addition, developing this system gives us opportunities to learn new technologies/tools which are potentially useful for our careers in future.

Team Preparation – a statement of relevant previous coursework or practical experience in your chosen development environment.

Courses taken: Data Structure I, Data Structure II, Application Programming, Netcentric Computing, Database Management, Algorithms, previous internships, hackathons.

We are strongly committed to the project. In addition, each team member is familiar with certain aspects of developing building the system such as SDLC, programming languages, environments, and tools which we plan on using.

SYSTEM REQUEST – (Birdie) (01/12/2018)

Project Sponsor

- Name: Kevin Dimaano and Kris Estilloso
- Email: rkdimaano@hotmail.com
- Organization / Department: MiniWhales
- Phone: 425-501-9065

Opportunity Statement

Cryptocurrencies are becoming a major tech-boom and an economic phenomenon. Although cryptocurrencies have existed since 2009, it has recently gained a lot of attention and have created an economy and market of hundreds of billions of dollars. As of now, there are limited tools available for easy trading and tracking. Therefore, there is a great opportunity to enter into this market space at this time.

Proposed Product

Background and Context:

We are proposing a system for MiniWhales that will be a web application that integrates with cryptocurrency services and allow users to track their earnings, losses, and better control their cryptocurrency.

Initial Vision and Scope:

- ➢ Graph and/or chart of earnings and losses over periods of time
- ➢ Real-time tracking of coins
- ➢ Integration with multiple cryptocurrency services/exchanges
- ➢ Ability to trade coins through our application
- ➢ Ability to incorporate automatic stop-loss and earning target buys/sells

Stakeholders:

- ➢ Traders (both at MiniWhales and outside)
- ➢ Exchanges
- ➢ Dr. Arias (wants a working application by end of Spring Quarter 2018)

Expected Benefits:

- ➢ Allowing users to gain a greater control of their cryptocurrency
- ➢ Allowing users to trade coins more easily
- ➢ Allowing users to calculate trends in earnings and losses
- ➢ Further exposure to cryptocurrencies

<u>Other Issues or Constraints to be considered (what else you would like us to know):</u>

- ➢ API calls require reliance on cryptocurrency services (i.e. Bittrex and Binance)
- ➢ User authentication requires security and encryption
- ➢ User support and reliability of buy/sell orders
- ➢ SECURITY

2. System Services

As a part of the system services, the functional requirements are described, linked to the use case numbers, and given a set of cases that serve as initial acceptance tests. This test-driven approach will help mitigate any risk that may be overlooked during the implementation phase. Moreover, a use-case diagram is provided to allow a general overview of each use case and which stakeholders are involved. Per use-case, an in-depth description accompanies each use case allowing for a clearer overview of the visual provided.

2.1 Function Requirements

Stop loss and/or limit (by percentage and by value):	Each user will be able to set a limit and capacity on their losses and gains, respectfully, ensuring a trade when the value of a coin has reached the boundaries set. Several target values can be set to sell a certain amount when the target is passed.
Use Cases:	0, 1
Initial Acceptance Tests:	• Users are able to set both upper boundary and lower boundary of limit independently per coin. • Set stop loss and limit through percent • Set stop loss and limit as value • Attempt to have an equal minimum/maximum to current value of coin • Sell a negative amount of coins or percent of their coins • Attempt to set multiple targets per stop loss (100 targets per coin) • Attempt to sell more than 100% of coins • Attempt to set percentage of targets to less than $0.00.

Calculate Profits and Loss	Users will be able to calculate their profits and losses in a variable time interval (based on user's choice). Calculations will be based off of the starting value of their coins in certain time frame.
Use-Cases	0, 2, 4
Initial Acceptance Tests	• Attempt to calculate profits and loss in different time intervals (minutes, hours, day, etc.) • Attempt to set a minimum time interval before coins were purchased • Attempt to set time interval to a future date • Attempt to input a later time as the minimum as opposed to the maximum • Attempt to input a prior time as the maximum as opposed to the minimum • Attempt to input invalid format

Notifications for nearing target	User can set up the price target for his coin at which he wants to sell it. When the current price is close to the target, Birdie will notify the user by sending text messages to his phones.
Use-Case(s)	0, 3, 1
Initial Acceptance Tests21	• Add multiple coins to the watch-list • Allow notifications when phones are off/not available. • Only one notification per *x* time (as to prevent mass notifications from jiggles in market) • Notification preferences all work (email/phone/etc)

User registration	User is required to create an account to track his/her trading information.

Use-Case(s)	**0**
Initial Acceptance Tests	• Successfully create an account • Attempt to create with same username/email • Set up 2FA • Set up exchange specific authentication/links • User verification (such as photo ID) • Referrals (?) • Attempt to create an account with important information missing. • Password requirements (1 uppercase, letters and numbers for example) • User agreement when they sign up for our services • User forgets password • User attempting to delete account • Users attempt to modify their account

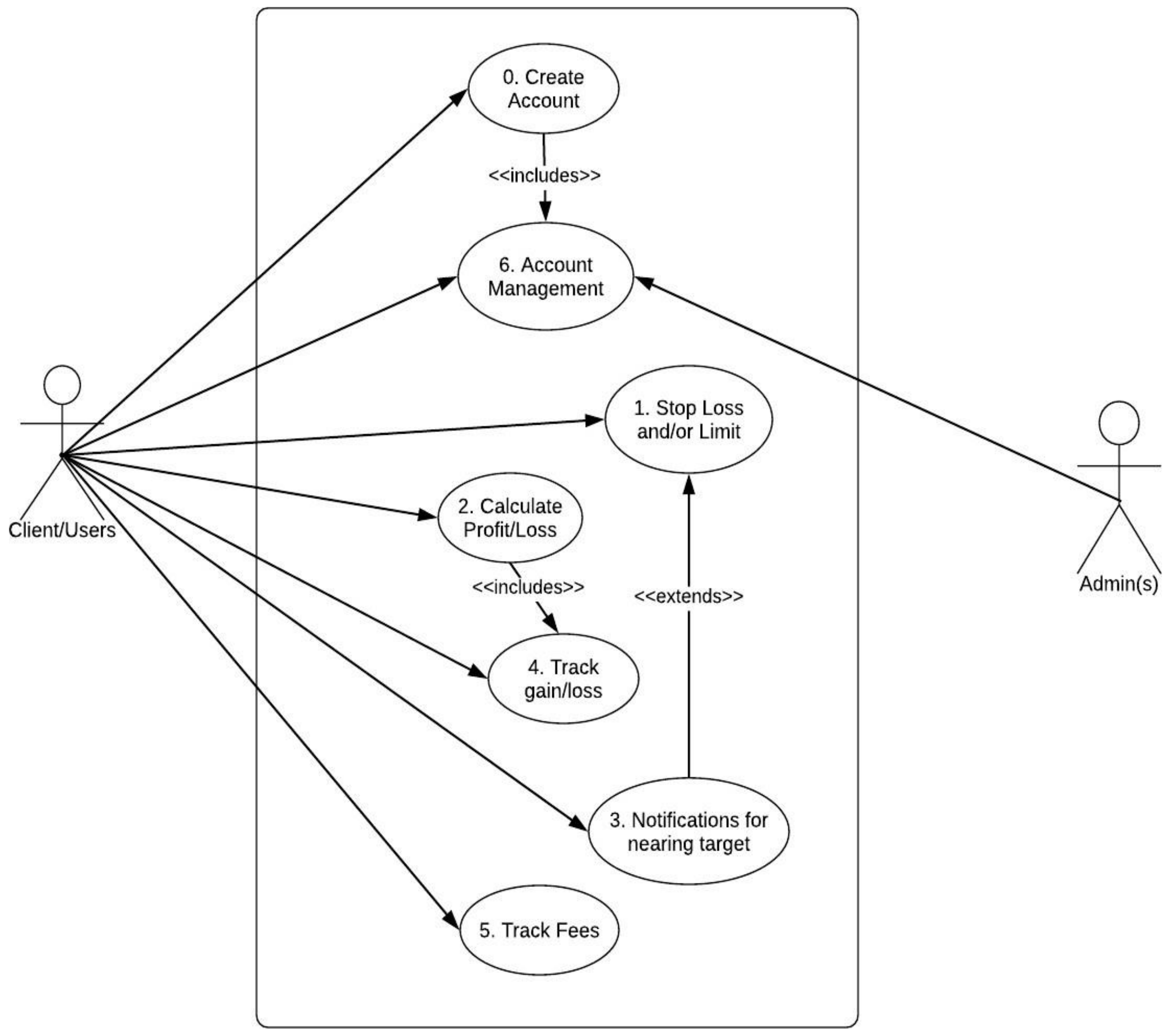
Account Management	The users and the admins will be able to edit, delete, and generally manage accounts.
Use Cases:	0,6
Initial Acceptance Tests:	• Admins attempt to delete an account • Admins attempt to modify an account • Users will attempt to delete their own account. • Users will attempt to delete another user's account • Admins will attempt to delete their own account • Users will attempt to modify their account • Admins will attempt to modify their account • Admins will attempt to modify a user's account • Users will attempt to modify another user's account.

Track Gain/Loss	The trader will be able to look at their history and see total gains or losses over a

	period of time and (as of now) given exchange.
Use Cases:	0,2,4
Initial Acceptance Tests:	• Check profit/loss for 1 day, 7 day, 1 month, 1 year, and total • Check profit/loss given time frame (say 1-12-18 to 3-15-18) • Check profit/loss given an exchange • Attempt to check invalid time range • User is not logged in but attempts to check profit/loss • Give gain/loss via percentage and/or value • Check gain/losses on specific coins • Attempt to check a coin that doesn't exist • Attempt to check a coin that one has no history in

Track Fees	Traders can look at all accumulated fees throughout their trades. Each coin may have their own respective fees, so traders can look at the implications of each coin.
Use Cases:	0,5
Initial Acceptance Tests:	• Check current fees for each coin • Notify user if fees change • Calculate the amount loss in fees • Check what fees were for each transaction • Attempt to check a coin that doesn't exist

2.2 Use-Case Diagram



2.3 Use-Case Description

Use-Case: Create Account	**ID**: 0
Actors: Client/Users	**Primary Actor**: Client/Users

Stakeholders and interest:
Client/Users: Need to be able to create an account to store data, and access the service

Brief description:
Users of the application will be able to create an account for the application

Relationships:

Association: Client/User
Include: Account Management
Extend: --
Generalization: --

Trigger: User clicks create account

Type (circle one): [✓ **External**] Temporal

Basic Flow of Events:

1. User clicks "Create an Account" button in the application interface
2. User creates password
3. User enter personal information, such as, name, address, email and password
4. User enters payment method
5. User enter cryptocurrencies they want to enter
6. The account is created
7. User info is saved

Subflows:

NA

Extensions / Alternate Flows:

S2:
If password fields do not match, ask user to enter password again
If credit information is incorrect ask user to enter information again

Use-Case: Stop Loss and/or Limit	**ID**: 1
Actors: Client/Users	**Primary Actor**: Client/Users
Stakeholders and interest: Client/Users: Need to be able set a limit on their losses and gains	
Brief description: Gives the user the ability to set a limit on the fluctuation of the value of their coin, so the trade is ensured when the value of the coin has reached the set boundaries	
Relationships: **Association**: Client/User **Include**: -- **Extend**: 3. Notification for nearing target **Generalization**: --	
Trigger: User clicks create Stop Loss and/or Limit button **Type** (circle one): [✓ **External**] Temporal	
Basic Flow of Events: 1. User clicks "Create Stop Loss" button 2. User sets a Stop Loss value 3. Stop Loss is displayed on the graph OR 1. User clicks "Create Limit" button 2. User sets a Limit value 3. Limit is displayed on the graph	
Subflows: NA	
Extensions / Alternate Flows: S-3: If value nears Stop Loss or the Limit the user is notified	

Use-Case: Calculate Profit/Loss	**ID**: 2
Actors: Client/Users	**Primary Actor**: Client/Users
Stakeholders and interest: Client/Users: Need to be able to know how much they made or lost	
Brief description: Users of the application will be able to calculate how they gained of lost in a time interval, which is set by the user	
Relationships: **Association**: Client/Users **Include**: 4. Track gain/loss **Extend**: -- **Generalization**: --	
Trigger: User clicks Calculate Profit/Loss button **Type** (circle one): [✓ **External**] Temporal	
Basic Flow of Events: 1. User clicks calculate profits/losses 2. User enters the time interval in which they wish to see their profits/loss 3. The profits/loss are displayed	
Subflows: NA	
Extensions / Alternate Flows: S2: If the user enters an invalid time period, the user is asked to enter a valid time period	

Use-Case: Notifications for nearing target	**ID**: 3
Actors: Client/Users	**Primary Actor**: Client/Users
Stakeholders and interest: Client/Users: Will be interested in getting notifications when the value nears their set targets	
Brief description: A notification is sent out to the user when the price target of when the user wants to sell their coin nears. Notification is sent out via text message	
Relationships: **Association**: Client/User **Include**: -- **Extend**: Stop/Loss Limit **Generalization**: --	
Trigger: User sets notifications **Type** (circle one): [✓ **External**] Temporal	
Basic Flow of Events: 1. The user clicks set up notifications 2. The user enters the price target 3. When the price target nears the user is sent a text	
Subflows: S1: 1. The user chooses how they want to be sent the notification, email/phone 2. User chooses what kind of notification they want to be sent	
Extensions / Alternate Flows: S3: User is sent a notification when they reach the Stop/Loss limit	

Use-Case: Track gain/losses	**ID**: 4
Actors: Client/Users	**Primary Actor**: Client/Users
Stakeholders and interest: Client/Users: want to be able to track their gains and losses	
Brief description: Users are able to track their gains and losses over a period of time.	
Relationships: **Association**: Client/User **Include**: -- **Extend**: -- **Generalization**: --	
Trigger: User clicks button **Type** (circle one): [✓ External] Temporal	
Basic Flow of Events: 1. User clicks Track Gains/Losses 2. User chooses a time period 3. User chooses an exchange 4. The gains/losses are displayed	
Subflows: NA	
Extensions / Alternate Flows: NA	

Use-Case: Track Fees	**ID**: 5
Actors: Client/Users	**Primary Actor**: Client/Users
Stakeholders and interest: Client/Users: They would like to know their accumulated fees from their trades	
Brief description: Each coin may have it's own trading fees, so users can look at the accumulated fees of each coin or they can look at the total fees	
Relationships: **Association**: Client/Users **Include**: -- **Extend**: -- **Generalization**: --	
Trigger: User clicks Track fees button **Type** (circle one): [✓ External] Temporal	
Basic Flow of Events: 1. User clicks Track Fees 2. Fees are displayed	
Subflows: S1: User chooses between total fees or individual coin fees Based on user choice, one of these is displayed	
Extensions / Alternate Flows: NA	

Use-Case: Account Mangement	**ID**: 6
Actors: Client/Users Admin	**Primary Actor**: Admin
Stakeholders and interest: Clients/Users and Admin: Will be able to edit accounts	
Brief description: Users and Admins will be able to edit, delete and manage accounts	
Relationships: **Association**: Client/User, Admin **Include**: -- **Extend**: -- **Generalization**: --	
Trigger: **Type** (circle one): [✓ **External**] Temporal	
Basic Flow of Events: 1. User/Admin click “Account Management” 2. Various options are displayed 3. User/Admin is asked if they are sure they want to make these changes 4. Changes are saved	
Subflows: S2: If admin clicks account management, different and more options will come up vs when a user click account management. Options could include: delete, change personal info, change payment method, etc.	
Extensions / Alternate Flows: NA	

3. Initial Data Requirements

This section includes a UML Class Model and detailed description for each class. Section 4.2 provides attributes and planned operations for each class. A UML diagram shows the design of the system. Nonfunctional requirements are constraints on how the system will be operated. We will be going through various nonfunctional requirements, which are operational requirements, performance requirements, security requirements, cultural and political requirements and process requirements.

3.1 Operational Requirements

The software to begin, will work on a web browser, with later releases having the potential of working on Android and iOS.

3.2 Performance Requirements

The response time, for each time a button is clicked should be almost immediate, at most a delay of 1 or 2 seconds being acceptable.

3.3 Security Requirements

General users have access to all the processes they need to invest in crypto currencies and to manage their accounts. They will also be able to delete their account if they wish. Administrators will have to verify the user, before they are able to join.

3.4 Cultural and Political Requirements

We will have to verify that people making accounts are real, so we will need to check personal information.

3.5 Project Constraints

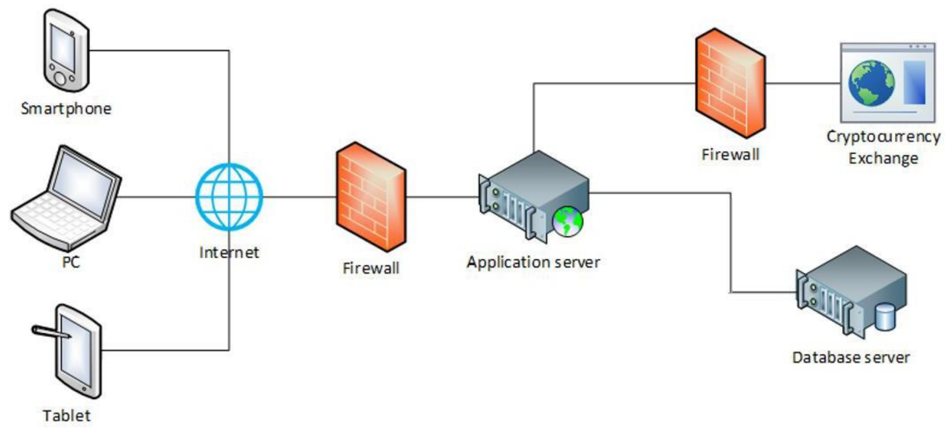
We don’t have funding and as of now this is a school project so we may not have all the resources that we need.

4.Development and Operating Environments

This section describes the overall system architecture including requirements for hardware, software, security and environment on which the server and client of Birdie can operate. One can also find the development environment and required tools which are used to build Birdie.

4.1 System Architecture

The system is three-tiered client-server architecture. Most of the program logic and data processing are handled by the application server. Birdie will use Application Program Interface (API) to interact and share data with a given cryptocurrency exchange. A database is responsible for the data access logic and data storage. All user's account information and password will be securely stored there. An internet browser on user computer will make requests and perform presentation of pages on the application server.



4.2 Hardware Environment

Birdie requires an application server to host application and programming logics. A database will be used to stored user's account information and password. The database will run on or a separate hardware from the application server. Any device with a web browser and internet connection can be used to access Birdie.

4.3 Software Environment

Birdie is designed run on all personal computers (PC). The latest version of a web browser (Chrome, Firefox, Edge or Safari) and internet connection are required. The system will require two software interfaces: user's device OS and the operating system running on the application servers.

4.4 System Security

The user's account and password will be stored on a database which will be secured by a cryptographic hash function (probably SHA-256). Only the system administrators can modify the database. The database will be backed up daily.

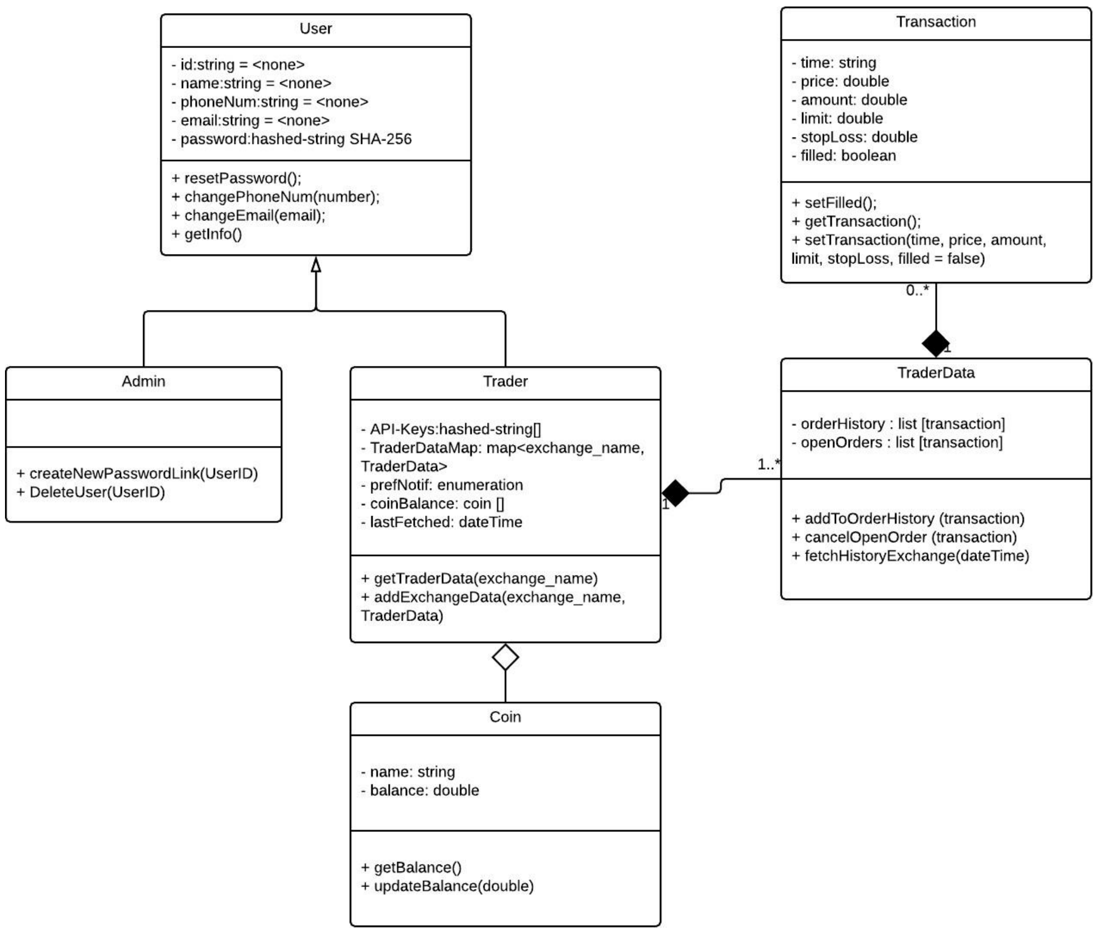
The servers will have the standard firewall included with the operating systems. There will be recovery operations and weekly routine backup for the servers. Incoming data to the servers will be checked for injection and illegitimate data. These firewalls and backups will be provided by AWS or AZURE.

4.5 Development Environment

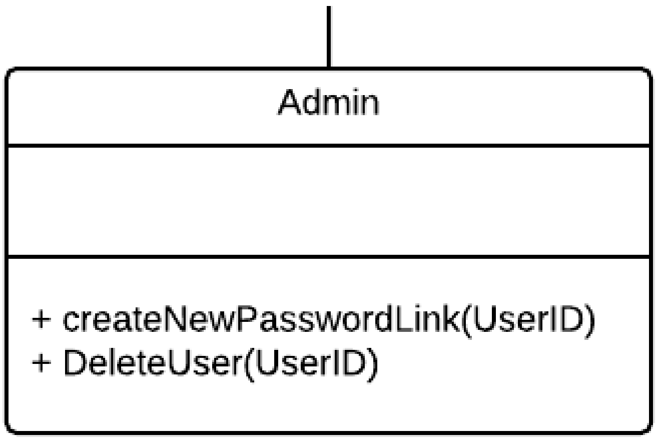
Tool	Use	Reason for being chosen
SPU server/GitHub/Heroku	Front-end design, Domain name	Domain name availability, reliability and scalability.
AWS/SPU Server/AZURE	Back-end server	Reliability, scalability, maintenance.
IntelliJ, WebStorm	Client-Side Scripting/coding, Server-Side Scripting/coding	Easy to code and debug, many useful built-in tools.
LucidChart	UML, Use-case diagrams	Free, easy to use.
Visio 2013	System Architecture	Free, easy to use.
Balsamiq	GUI design	Free, easy to use.

5. Data Design

5.1 Class Diagram

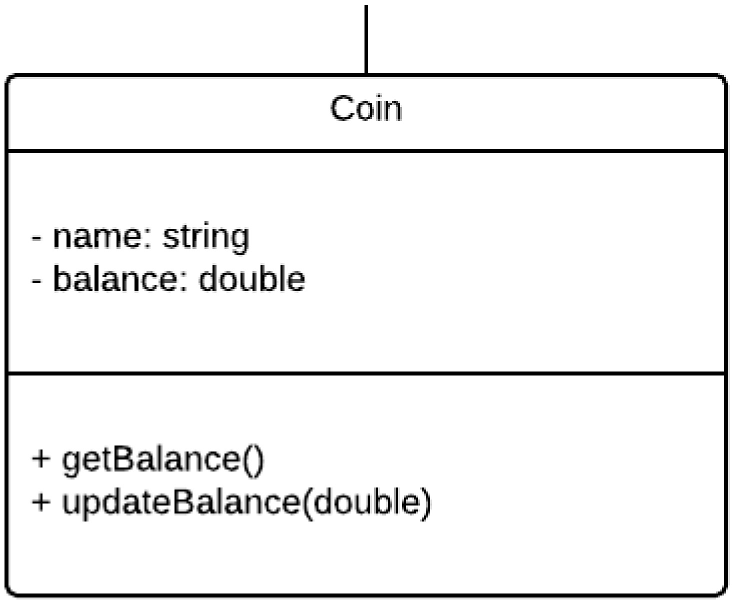

User
- id:string = <none>
- name:string = <none>
- phoneNum:string = <none>
- email:string = <none>
- password:hashed-string SHA-256
+ resetPassword();
+ changePhoneNum(number);
+ changeEmail(email);
+ getInfo()
Transaction
- time: string
- price: double
- amount: double
- limit: double
- stopLoss: double
- filled: boolean
+ setFilled();
+ getTransaction();
+ setTransaction(time, price, amount, limit, stopLoss, filled = false)
0..*
1
Admin
+ createNewPasswordLink(UserID)
+ DeleteUser(UserID)
Trader
- API-Keys:hashed-string[]
- TraderDataMap: map<exchange_name, TraderData>
- prefNotif: enumeration
- coinBalance: coin []
- lastFetched: dateTime
+ getTraderData(exchange_name)
+ addExchangeData(exchange_name, TraderData)
TraderData
- orderHistory : list [transaction]
- openOrders : list [transaction]
+ addToOrderHistory (transaction)
+ cancelOpenOrder (transaction)
+ fetchHistoryExchange(dateTime)
1..*
1
Coin
- name: string
- balance: double
+ getBalance()
+ updateBalance(double)

5.2 Data Dictionary (Metadata)



The admin class is an inherited class from the generic User class. The admin's sole responsibility is to maintain accounts.

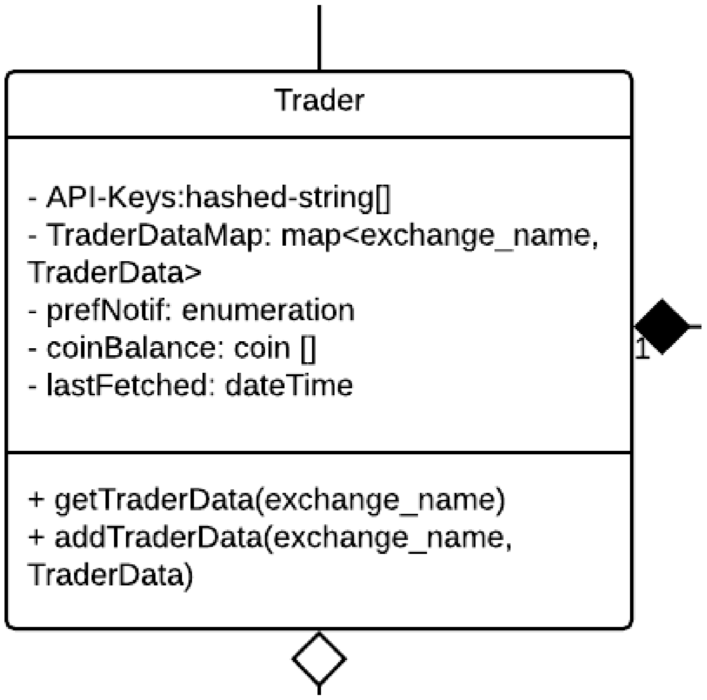
Operation	Return Type	Visibility	Scope	Parameters	Description	isQuery
createNewPasswordLink	**void**	**Public**		**UserID: int**	**Creates a reset password link for a given user**	**No**
DeleteUser	**void**	**Private**		**UserID: int**	**Deletes the account of the user given by UserID**	**No**



Coin is a class that holds the name of the coin and the balance associated with that coin for a specific user.

Attribute	Type	Visibility	Multiplicity	Initial Value
Name	String	Private	1	<None>
Balance	Double	Private	1	<None>

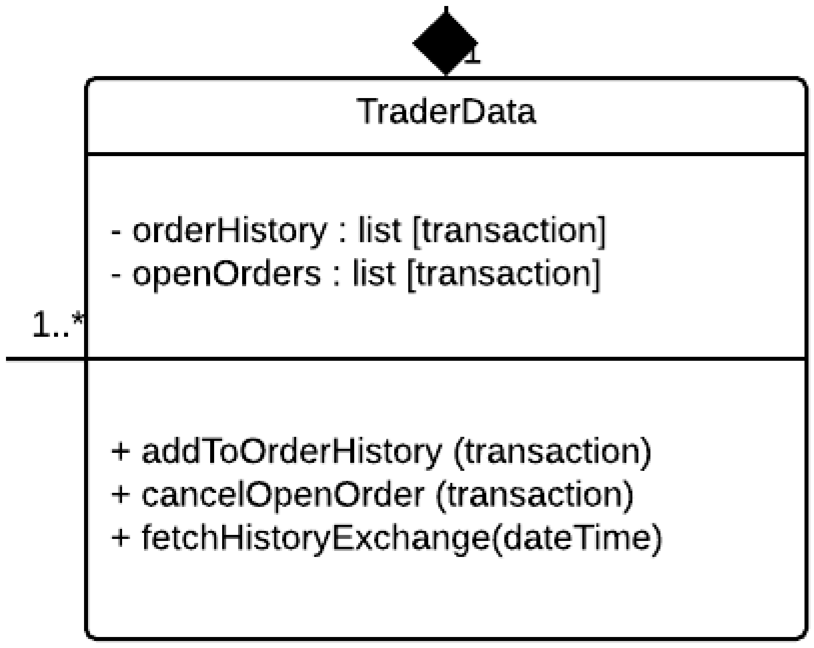
Operation	Return Type	Visibility	Scope	Parameters	Description	isQuery
getBalance	Double	Public		<None>	Returns the balance of the coin	Yes
updateBalance	Void	Public		Balance: double	Sets the balance of the coin to the given parameter	No


Trader
- API-Keys:hashed-string[]
- TraderDataMap: map<exchange_name, TraderData>
- prefNotif: enumeration
- coinBalance: coin []
- lastFetched: dateTime
+ getTraderData(exchange_name)
+ addTraderData(exchange_name, TraderData)
1

Trader is a class that has information and methods involving a trader. Has his/her API keys, exchange/trade information, and notification settings. This class also holds all of the trader's coin balances. The lastFetched dateTime field is for the server to know how much data to fetch from binance.

Attribute	Type	Visibility	Multiplicity	Initial Value
API-Keys	Hashed-string[]	Private	1	<None>
TraderDataMap	Map<string, TraderData>	Private	1	<None>
PrefNotif	Enumeration	Private	1	<None>
coinBalance	Coin []	Private	1	<None>
lastFetched	dateTime	Private	1	<None>

Operation	Return Type	Visibility	Scope	Parameters	Description	isQuery
getTraderData	Trader Data	Public		exchange_name : string	Returns the trader data for the given trader and for the given exchange	Yes
addExchangeData	Void	Public		exchange_name : string TraderData : TraderData	Adds a new traderData to an associated exchange	No

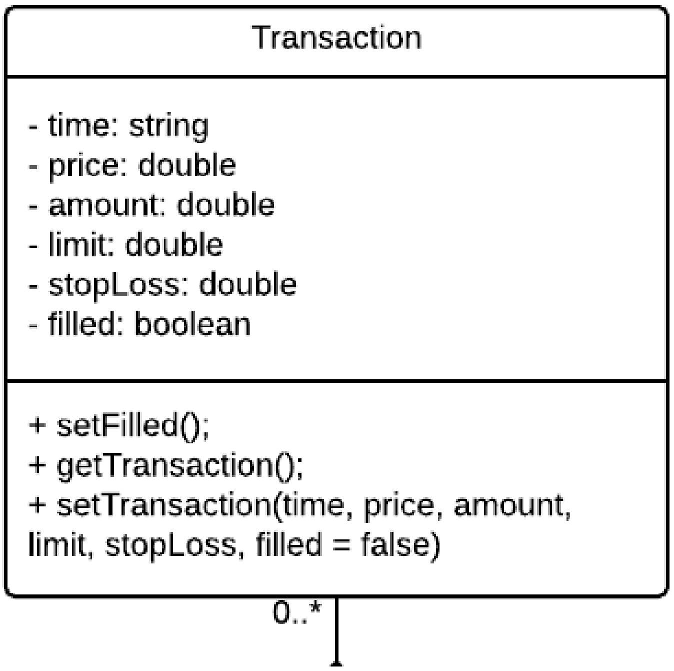


1
TraderData
- orderHistory : list [transaction]
- openOrders : list [transaction]
1..*
+ addToOrderHistory (transaction)
+ cancelOpenOrder (transaction)
+ fetchHistoryExchange(dateTime)

TraderData has the history and current open orders for a given trader. TraderData is an aggregate of transactions.

Attribute	Type	Visibility	Multiplicity	Initial Value
orderHistory	list[transaction]	Private	1	<None>
openOrders	list[transaction]	Private	1	<None>

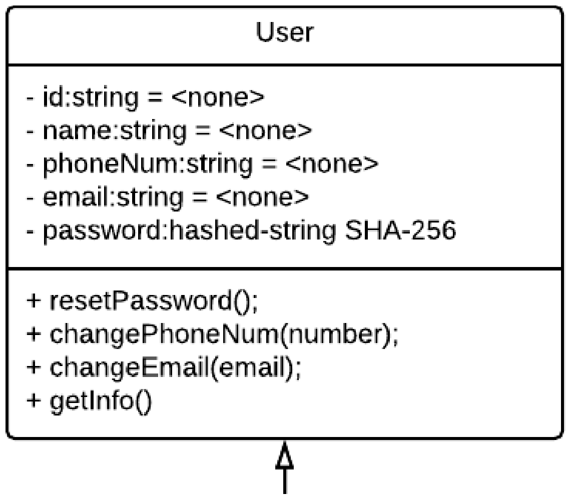
Operation	Return Type	Visibility	Scope	Parameters	Description	isQuery
addToOrderHistory	void	Public		transaction: transaction	adds an open order to the trader's history	No
cancelOpenOrder	void	Public		transaction: transaction	cancels an open order and therefore returns funds back to user	No
fetchHistoryExchange(dateTime)	void	Public		lastFetched: dateTime	Gets the order history of a user from lastFetched until current	Yes



Transaction tracks a certain transaction that a trader might have made; similarly, can be viewed as an order.

Attribute	Type	Visibility	Multiplicity	Initial Value
time	string	Private	1	<None>
price	double	Private	1	<None>
amount	double	Private	1	<None>
limit	double	Private	1	<None>
stopLoss	double	Private	1	<None>
filled	boolean	Private	1	false

Operation	Return Type	Visibility	Scope	Parameters	Description	isQuery
setFilled	void	Public		<none>	Sets the filled flag to be true, closing the transaction	No
getTransaction	transaction	Public		<none>	returns the transaction data	Yes
setTransaction	void	Public		(all attributes)	sets the transaction with the newly given values.	No



User has all the basic information, getters/setters for a typical user of the system. Extends to trader and admin classes.

Attribute	Type	Visibility	Multiplicity	Initial Value
id	string	Private	1	<None>
name	string	Private	1	<None>
phoneNum	string	Private	1	<None>
email	string	Private	1	<None>
password	hashed-string SHA-256	Private	1	<None>

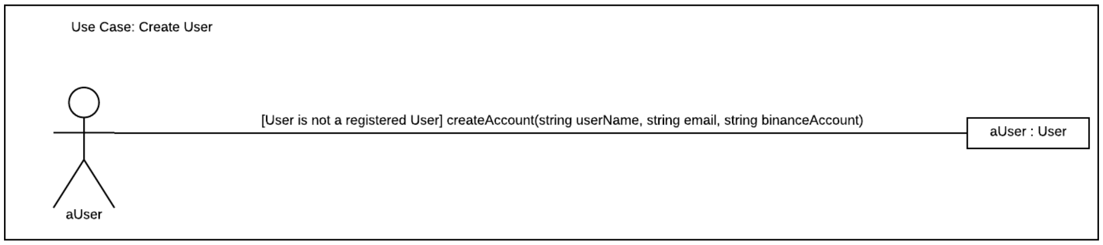
Operation	Return Type	Visibility	Scope	Parameters	Description	isQuery
resetPassword	void	Public		<none>	starts the process of resetting the password for a given user	No
changePhoneNum	void	Public		number: string	changes the number of the user with the newly given number	No
changeEmail	void	Public		email: string	changes the email of the user with the newly given email	No
getInfo	user	Public		<none>	Returns the aggregate data of the user	Yes

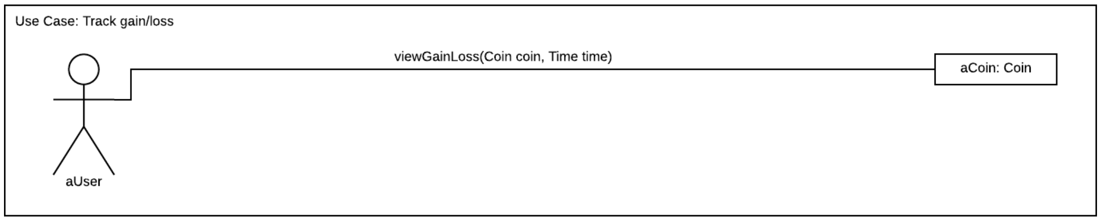
6. Behavioral Model

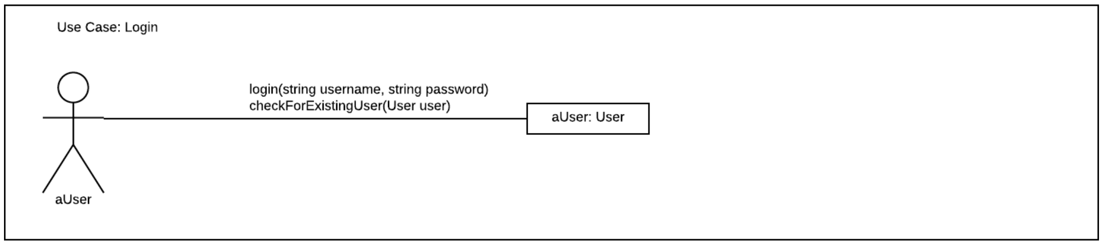
The diagrams in this section outline the use cases and their interactions with the user. Connected to an object, we can call the suggested methods to be able to invoke the functionality designed per use case. In addition to the collaboration diagrams, below is a state diagram where we can visualize the flow of the application's user interface and the applications responses based on the user-invoked events. These diagrams are designed to help developers visualize the vision of this application.

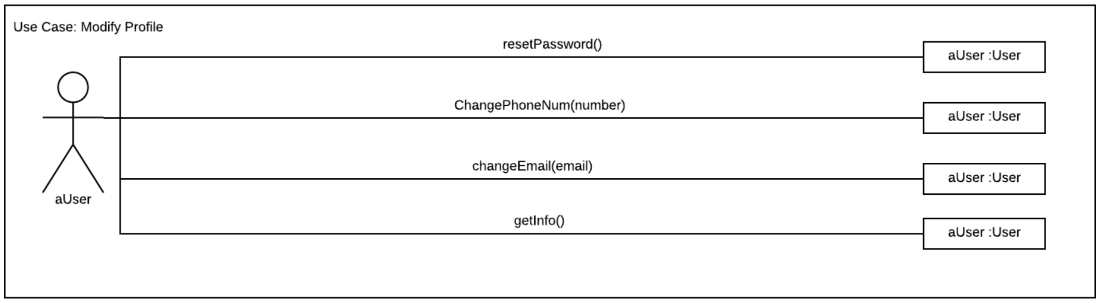
6.1 Collaboration Diagram

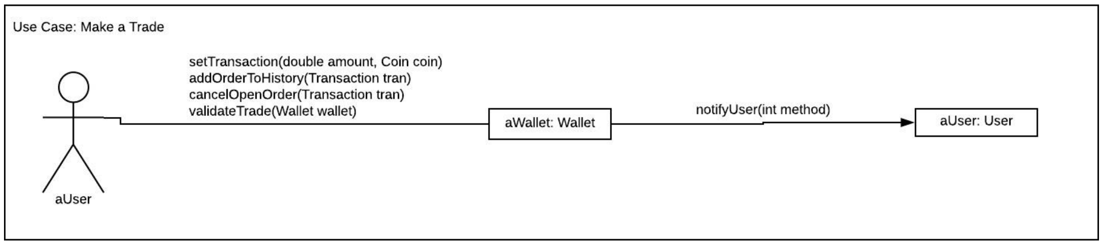
Below is the collaboration diagram.

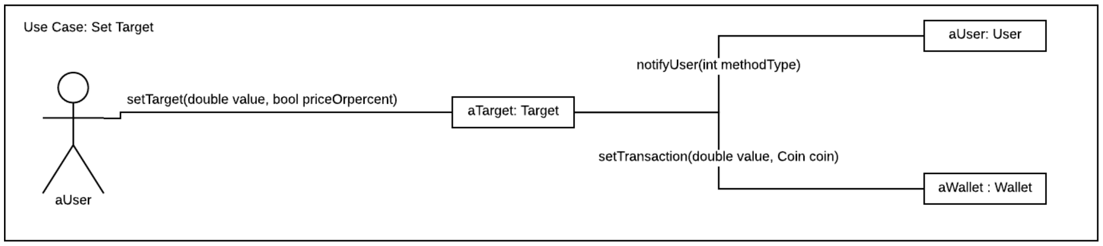

Use Case: Create User
[User is not a registered User] createAccount(string userName, string email, string binanceAccount)
aUser : User
aUser

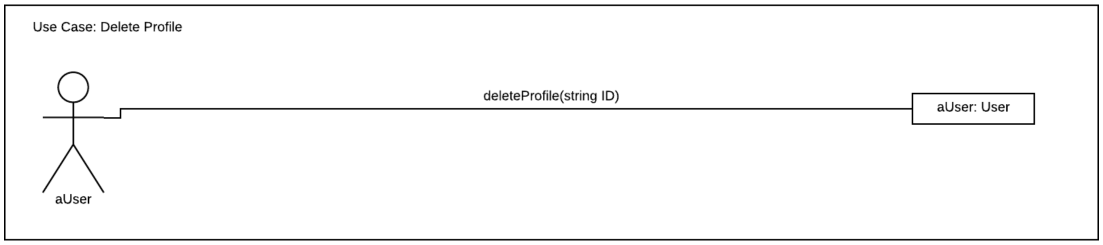


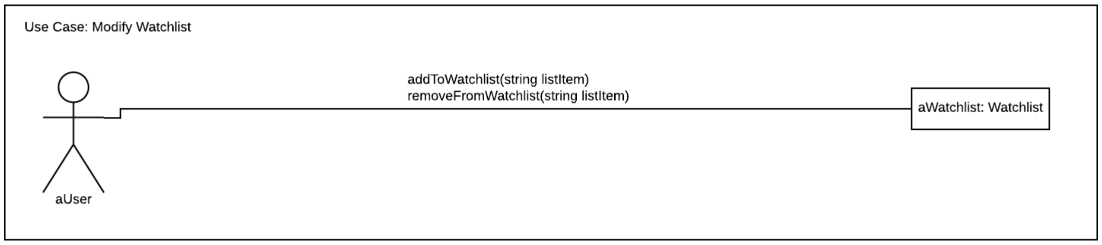

Use Case: Login
login(string username, string password)
checkForExistingUser(User user)
aUser: User
aUser


Use Case: Modify Profile
resetPassword()
aUser :User
ChangePhoneNum(number)
aUser :User
changeEmail(email)
aUser :User
getInfo()
aUser :User
aUser


Use Case: Make a Trade
setTransaction(double amount, Coin coin)
addOrderToHistory(Transaction tran)
cancelOpenOrder(Transaction tran)
validateTrade(Wallet wallet)
aWallet: Wallet
notifyUser(int method)
aUser: User
aUser

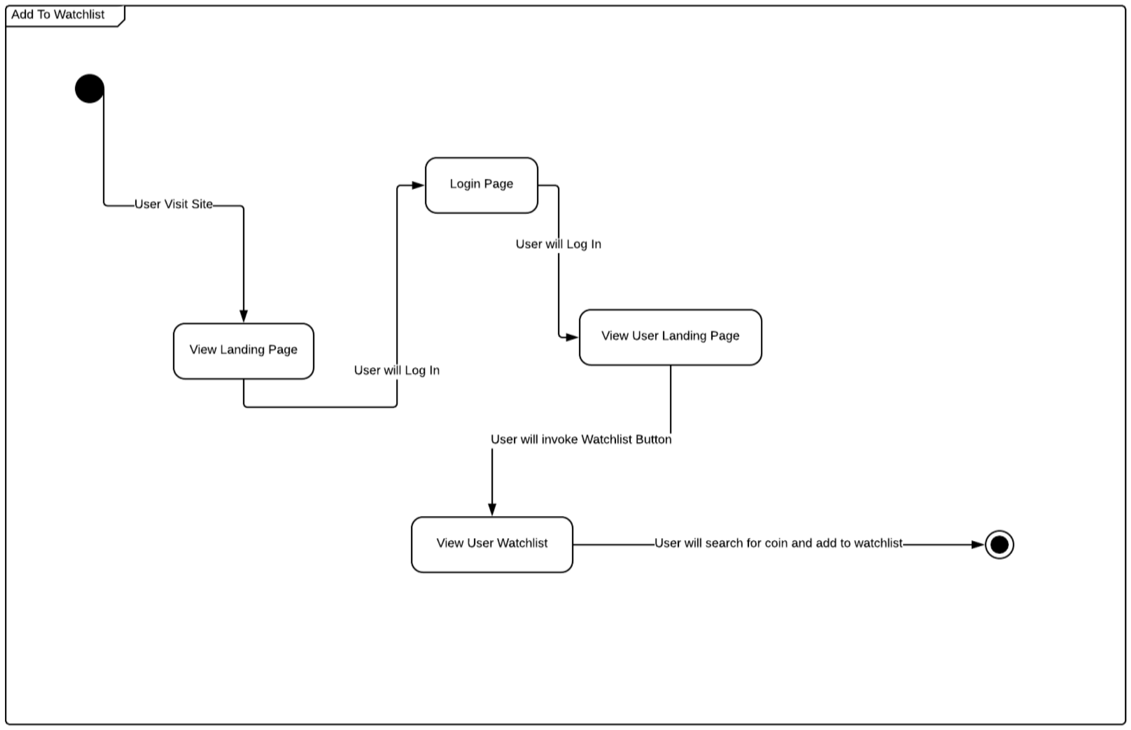

Use Case: Set Target
aUser: User
notifyUser(int methodType)
setTarget(double value, bool priceOrpercent)
aTarget: Target
setTransaction(double value, Coin coin)
aWallet : Wallet
aUser

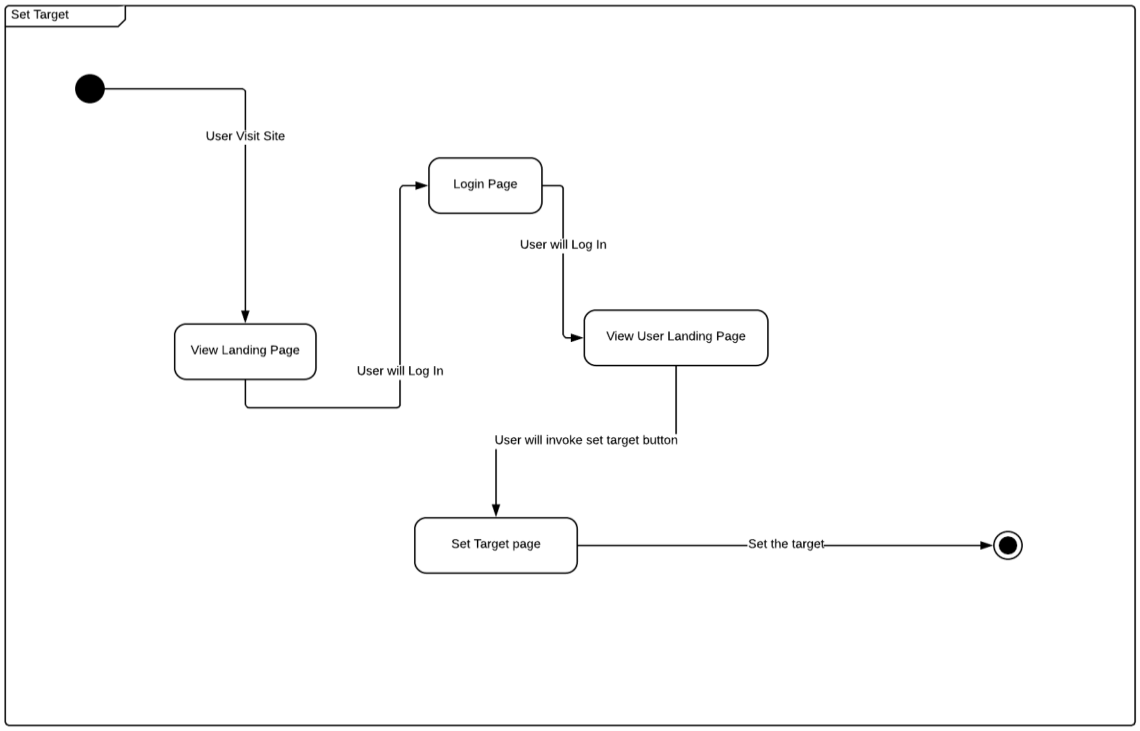

Use Case: Delete Profile
deleteProfile(string ID)
aUser: User
aUser

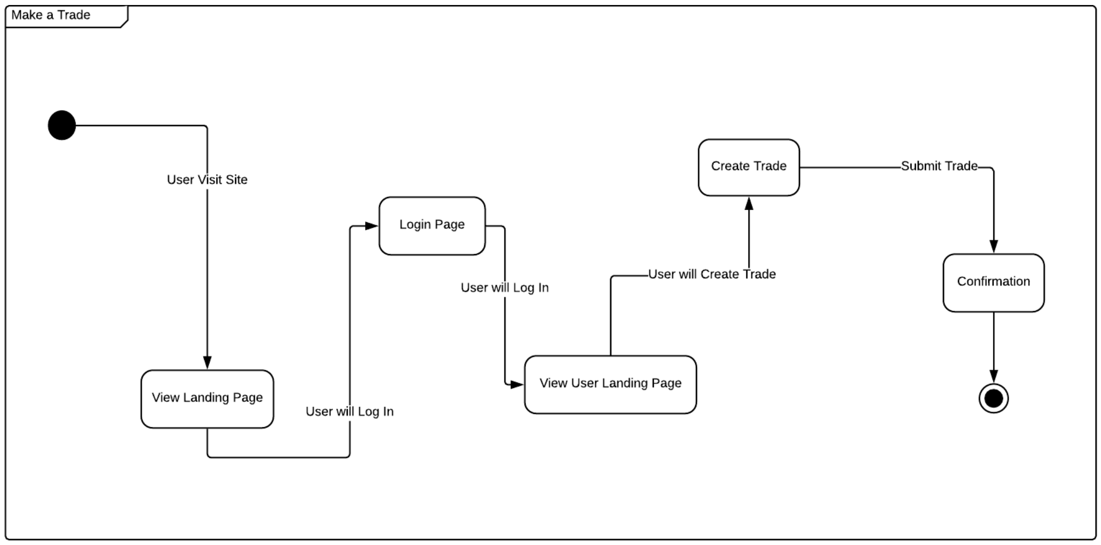

Use Case: Modify Watchlist
addToWatchlist(string listItem)
removeFromWatchlist(string listItem)
aWatchlist: Watchlist
aUser

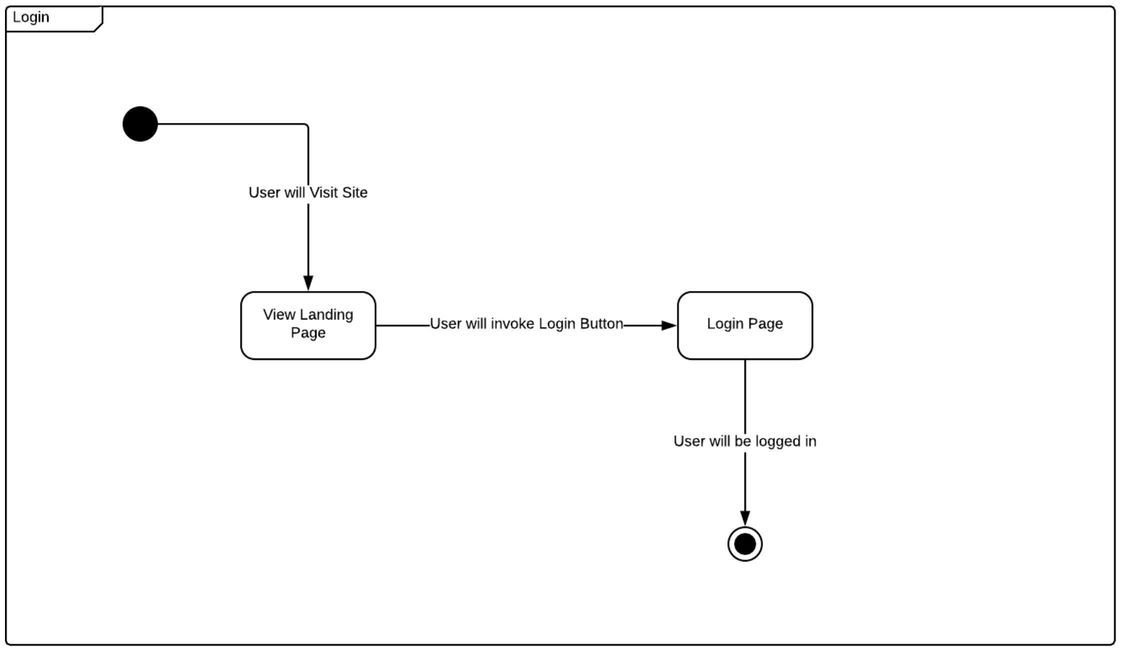
6.2 State Diagram

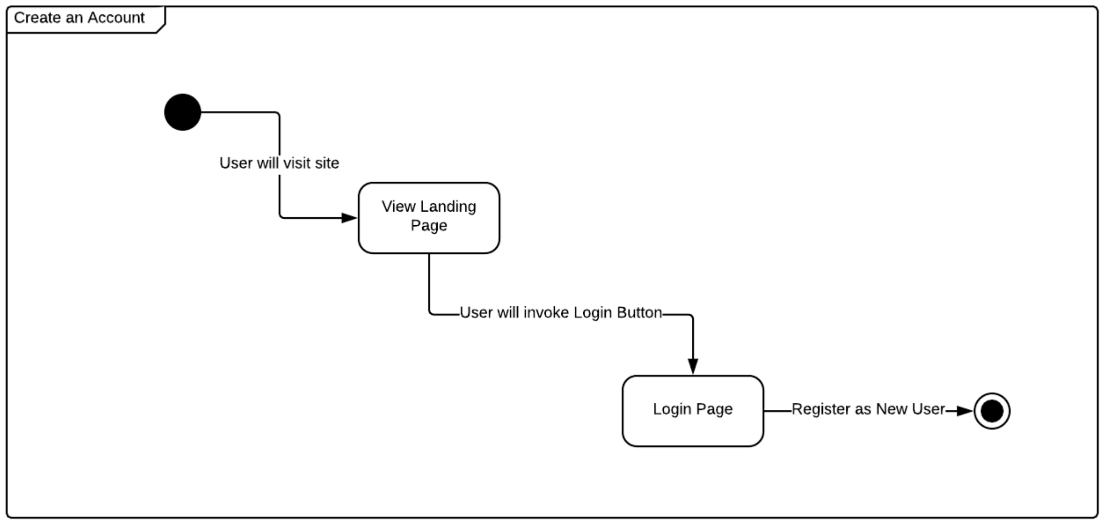
Below is the State Diagrams:

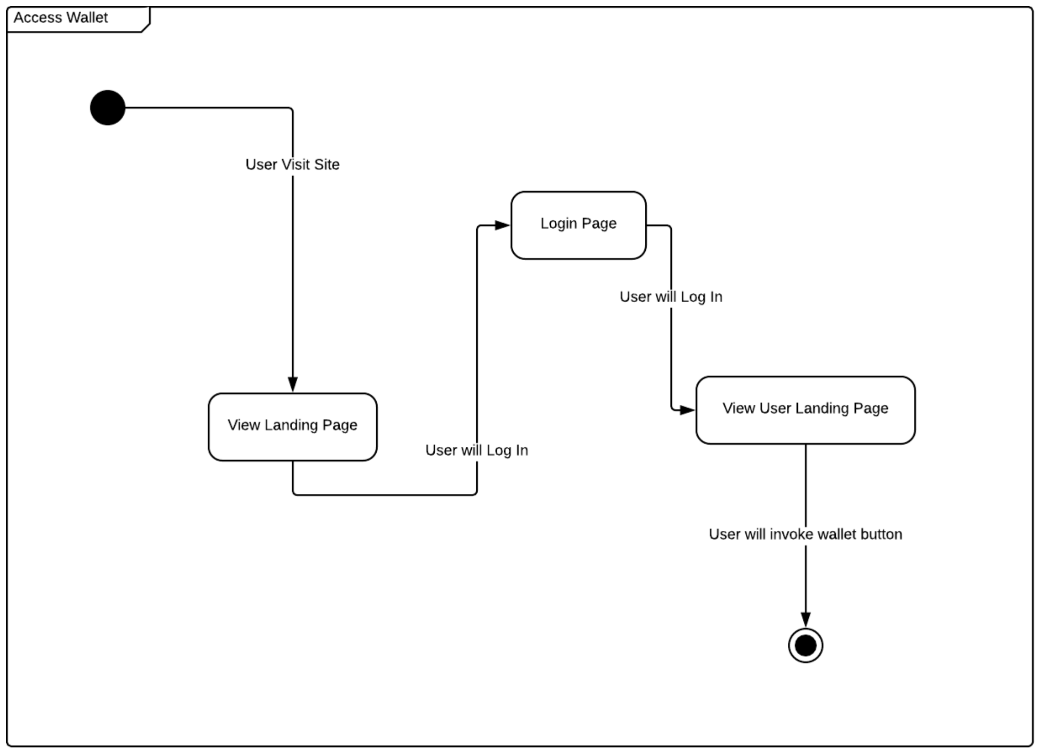


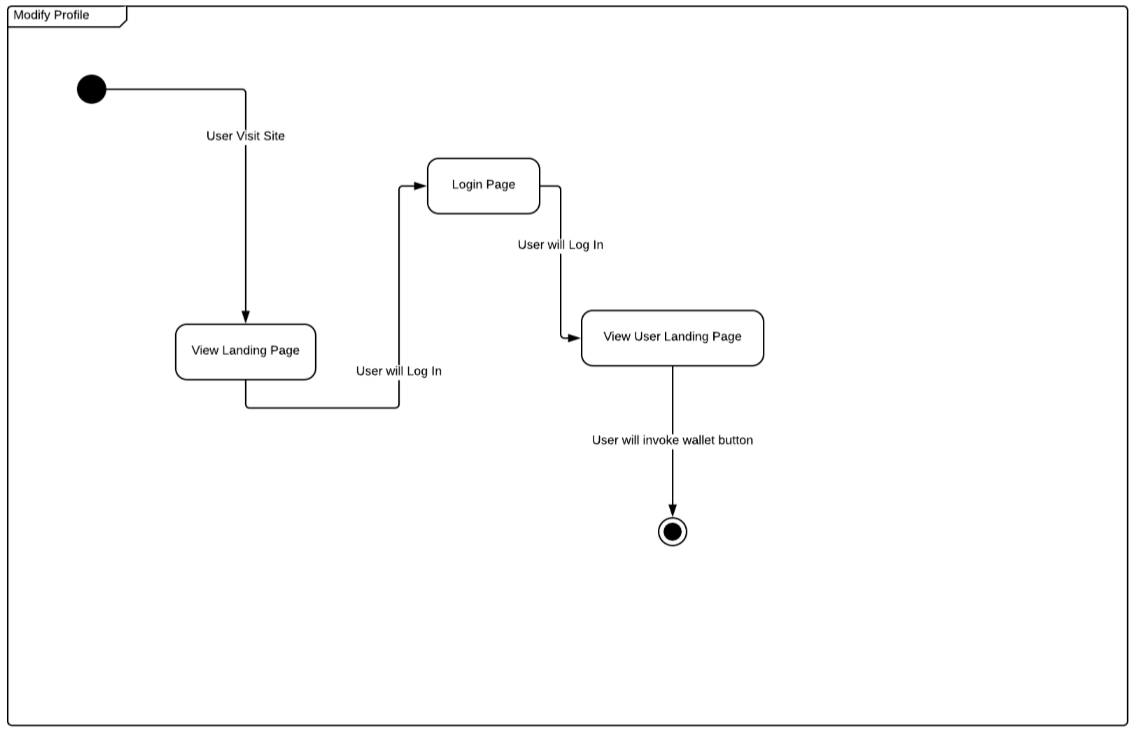

Set Target
User Visit Site
Login Page
User will Log In
View Landing Page
User will Log In
View User Landing Page
User will invoke set target button
Set Target page
Set the target

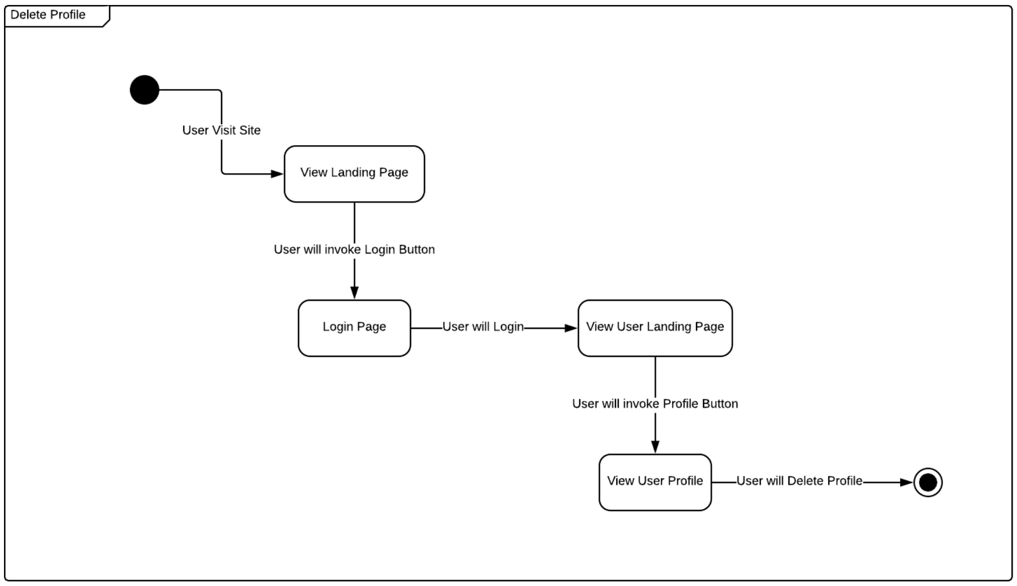

Make a Trade
User Visit Site
Login Page
User will Log In
Create Trade
Submit Trade
User will Create Trade
Confirmation
View Landing Page
User will Log In
View User Landing Page

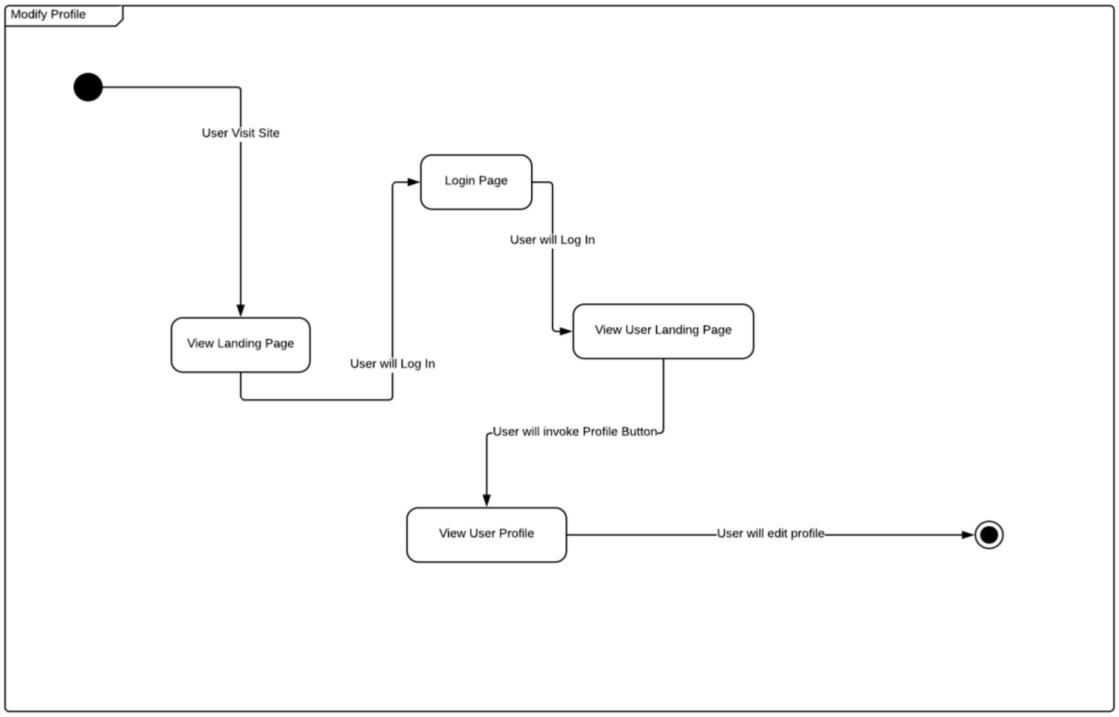

Login
User will Visit Site
View Landing Page
User will invoke Login Button
Login Page
User will be logged in

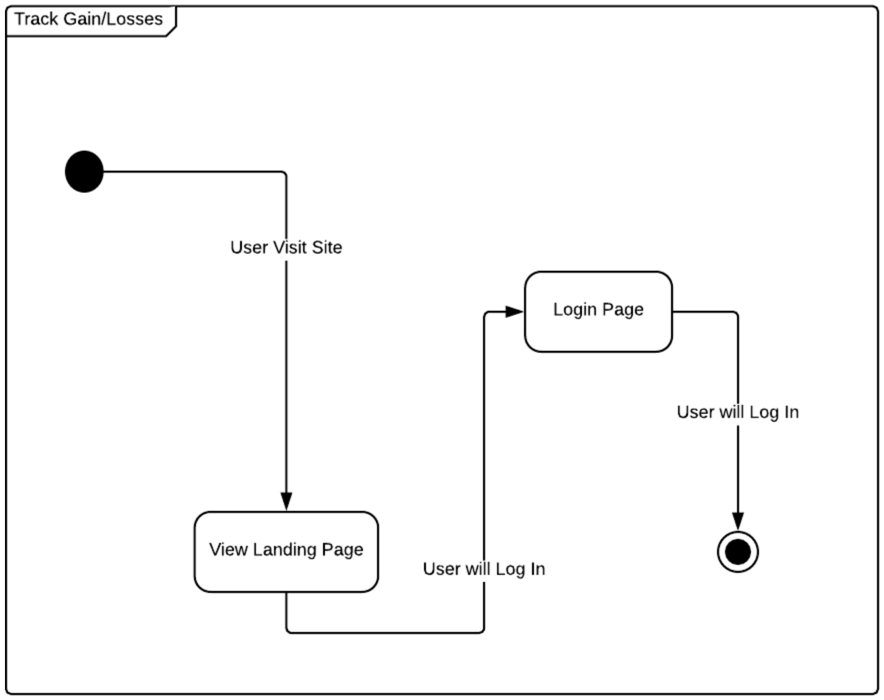

Create an Account
User will visit site
View Landing Page
User will invoke Login Button
Login Page
Register as New User


Access Wallet
User Visit Site
Login Page
User will Log In
View User Landing Page
View Landing Page
User will Log In
User will invoke wallet button


Modify Profile
User Visit Site
Login Page
User will Log In
View User Landing Page
View Landing Page
User will Log In
User will invoke wallet button


Delete Profile
User Visit Site
View Landing Page
User will invoke Login Button
Login Page
User will Login
View User Landing Page
User will invoke Profile Button
View User Profile
User will Delete Profile


Modify Profile
User Visit Site
Login Page
User will Log In
View User Landing Page
View Landing Page
User will Log In
User will invoke Profile Button
View User Profile
User will edit profile


Track Gain/Losses
User Visit Site
Login Page
User will Log In
View Landing Page
User will Log In

7. User Interface Design Plan

Below are general mock-ups to show the prospected design of the application. These mock-ups allow developers to spend less time in design implementations, while more time with the logic of the application. In addition, it helps by grabbing feedback to any potential clients to allow for a greater feedback on the design and flow of the application.

7.1 User Interface Requirements and Constraints

The UI will be coded using JS libraries. The landing page will have options, such as logging in, create trade, set target, account information etc. The Landing page will also display a list of cryptocurrencies as well as how much they've gained or lost in a chosen time period. We recommend using chrome as the browser. As far as constraints go, formatting from desktop to mobile may not be as clean as we like it to be.

7.2 Window Navigation Diagram

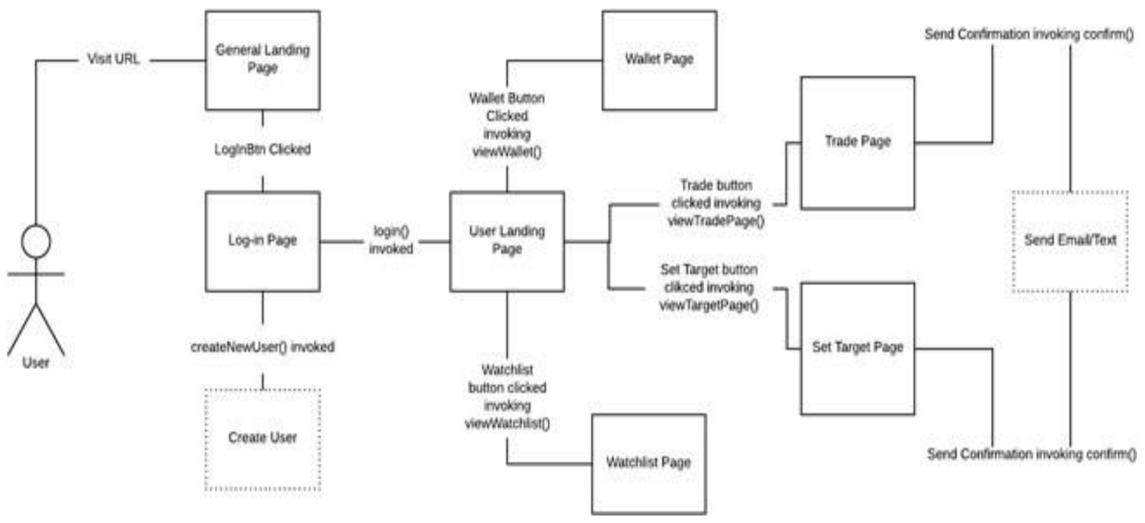

Visit URL
General Landing Page
LoginBtn Clicked
Log-in Page
login() invoked
createNewUser() invoked
Create User
User
Wallet Page
Wallet Button Clicked invoking viewWallet()
User Landing Page
Watchlist button clicked invoking viewWatchlist()
Watchlist Page
Trade button clicked invoking viewTradePage()
Set Target button clikced invoking viewTargetPage()
Trade Page
Set Target Page
Send Confirmation invoking confirm()
Send Email/Text
Send Confirmation invoking confirm()

Figure labels based on Mockups below:

Figure 1.: Represents the landing page in which a general user will be greeted on. About Birdie will simply state our services and features. General unlogged users can go to the login page to create a new account. General users will be allowed to have real-time viewing of coin's current prices for a selected amount of time.

Figure 2.: Represents the login page in which the user will have an option to login or create an account if they have not done so. Things to include down the road could be a terms and conditions checkbox and some type of Binance account verification via a modal.

Figure 3.: Represents the landing page of a user that has an account and is logged in will be greeted on. Coin Table will be clickable and show the values and targets of the selected coin to the left.

Figure 4.: Chart from the wallet where it can show OVERALL gain/loss since a certain time-period for all coins they are participating in. Features to potentially add is a "Total Value of coins participating in" row.

Figure 5.: The watchlist will allow users to view their watchlist and add coins to the watchlist. Adding to the watchlist could be done through the search bar as opposed to having a floating addition button.

Figure 6.: Allows users to trade coins for other currencies via market value or available selling point. Add validation checks if their wallet does not have a certain coin, not enough coins, etc. Uncertain about how to approach market value trading as opposed to selling for your own selected price. Client to verify layout trading.

Figure 7.: Allows users to set targets on coin value. These targets can automate a trade, notify the user, or both. For further additions, consider asking user what to do in the case the transaction does not happen, implement a "buy" target, where if the market gets to a certain price, buy x amount of coins, method to notify (text, email, messenger pigeon).

Figure 8.: Allows users to modify their profile information and delete their profile from the database. Additional features could include a table in which the user can view the coins they are selling in the market.

7.3 Screen Interface Design

Below are figures mapping to the window navigation diagram.

Figure 1.

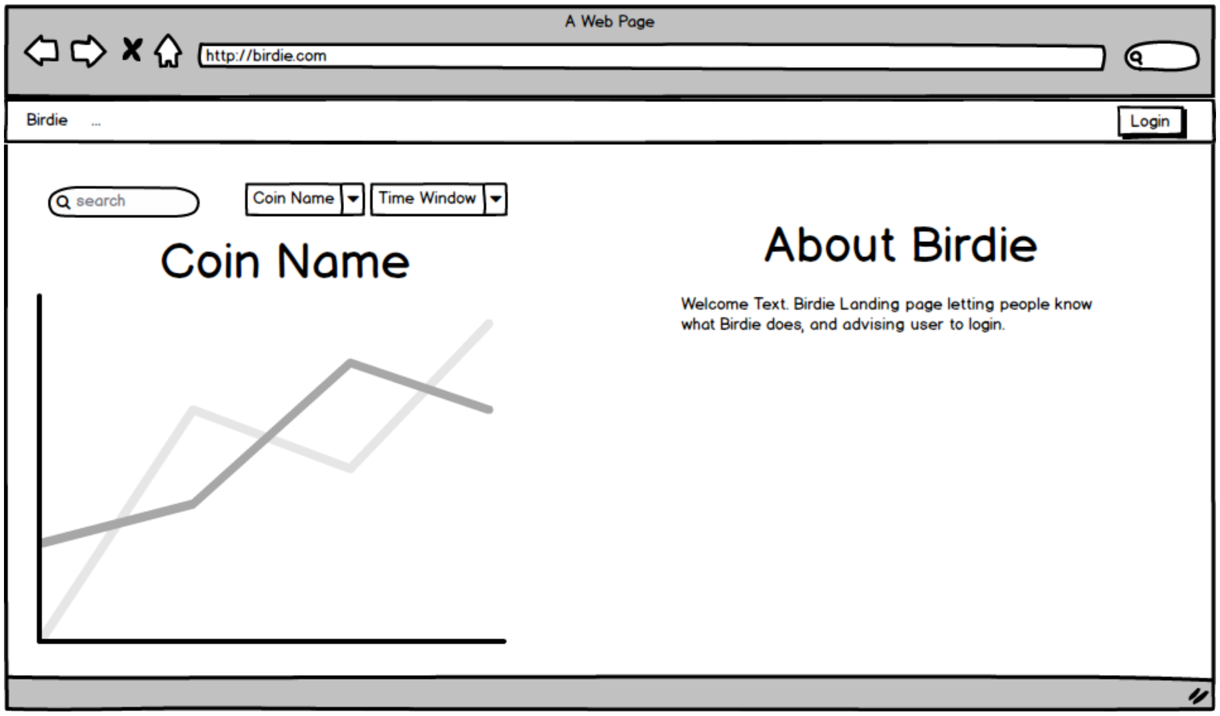

A Web Page
http://birdie.com
Birdie ...
Login
search
Coin Name
Time Window
Coin Name
About Birdie
Welcome Text. Birdie Landing page letting people know what Birdie does, and advising user to login.

Figure 2.

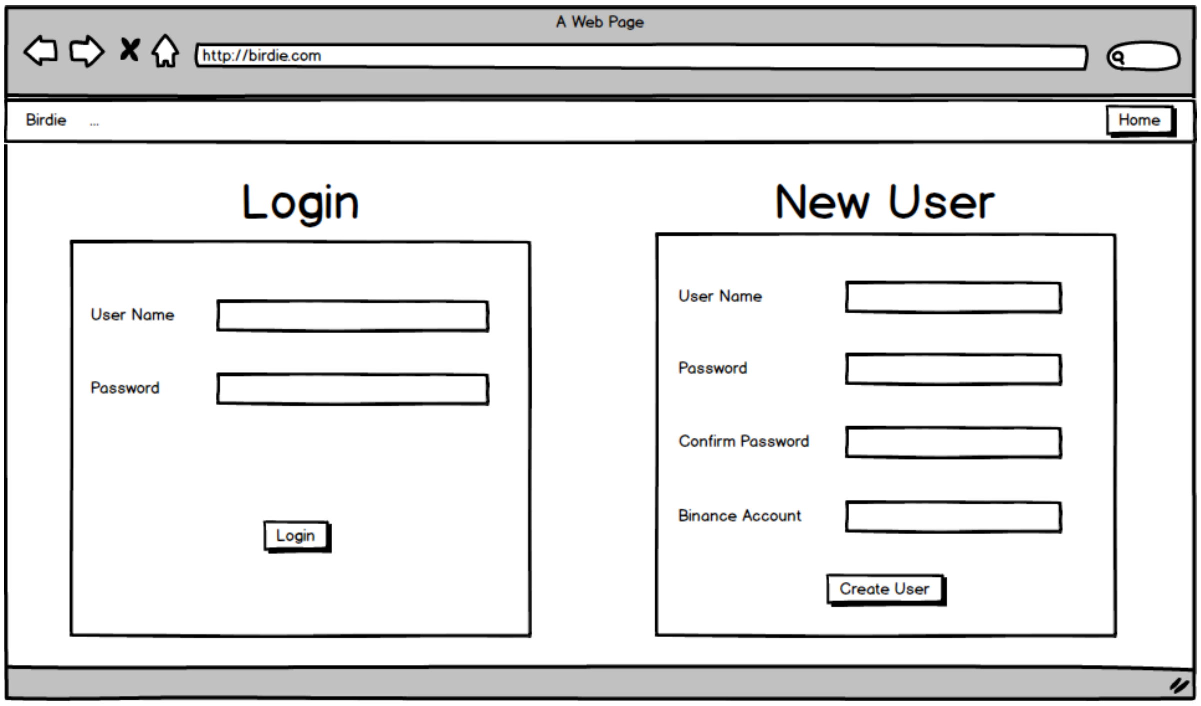
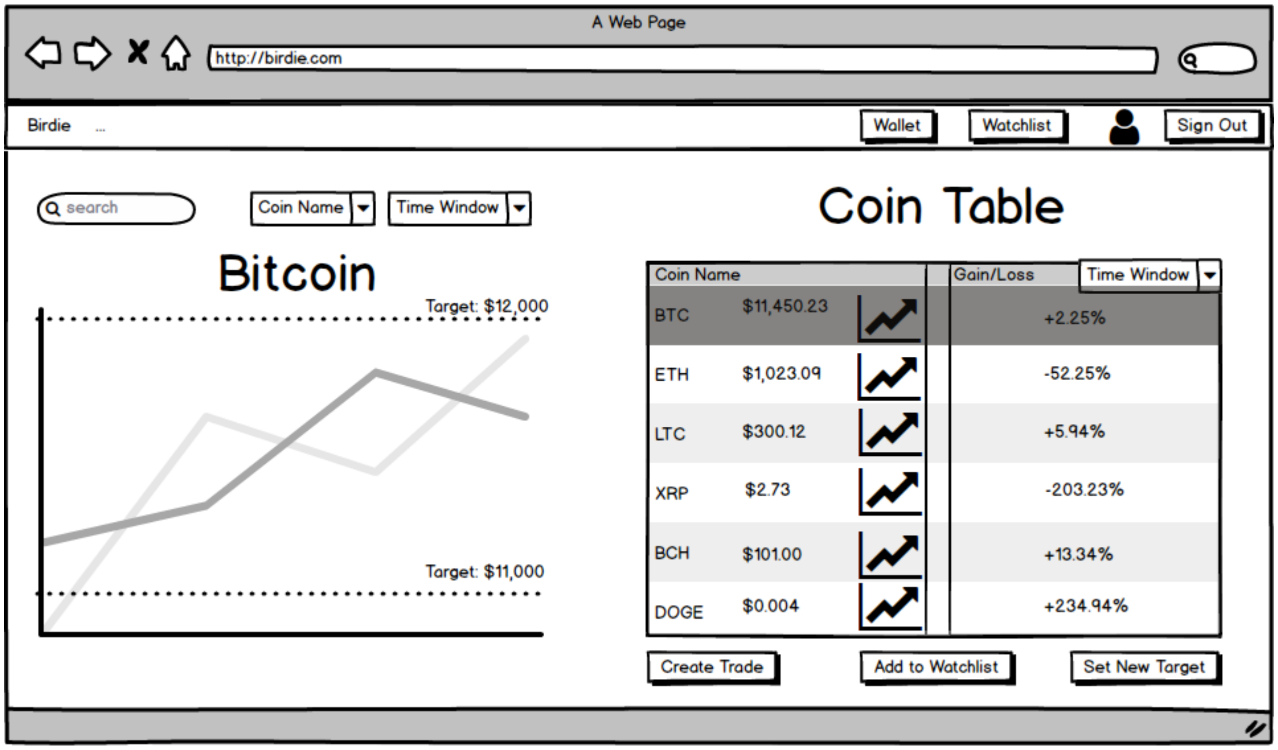
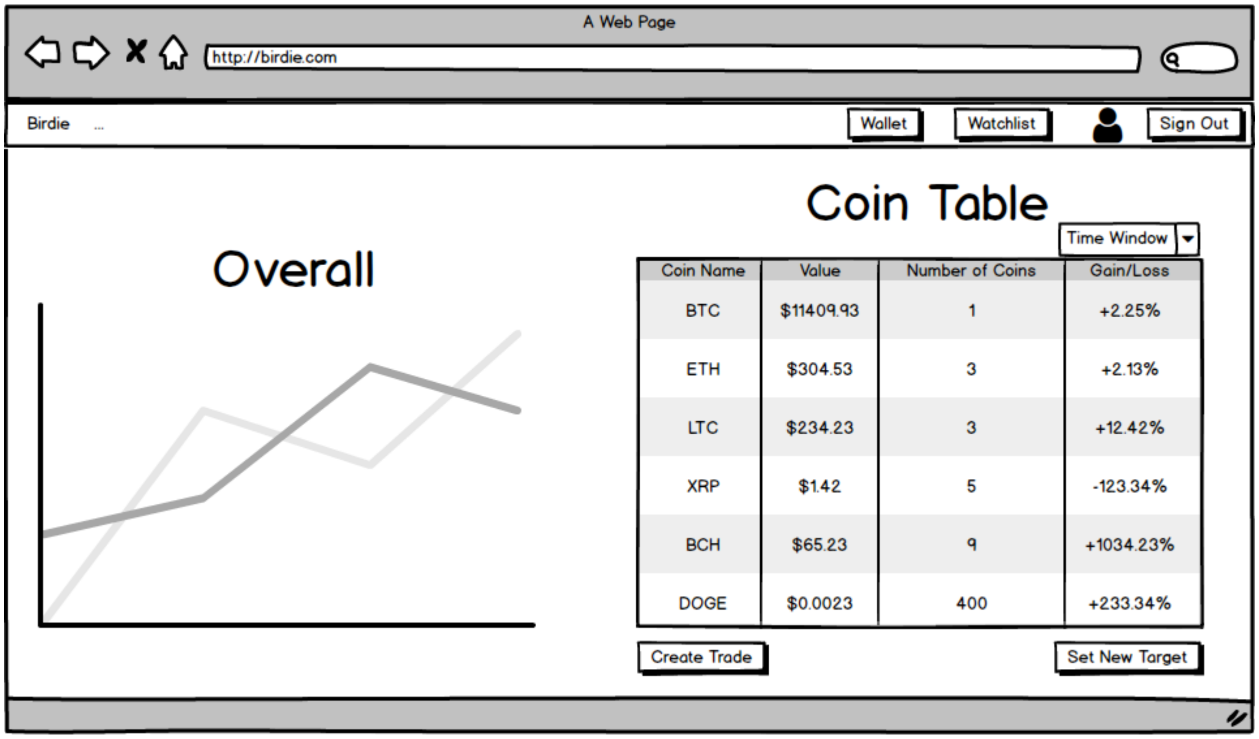


Figure 3.



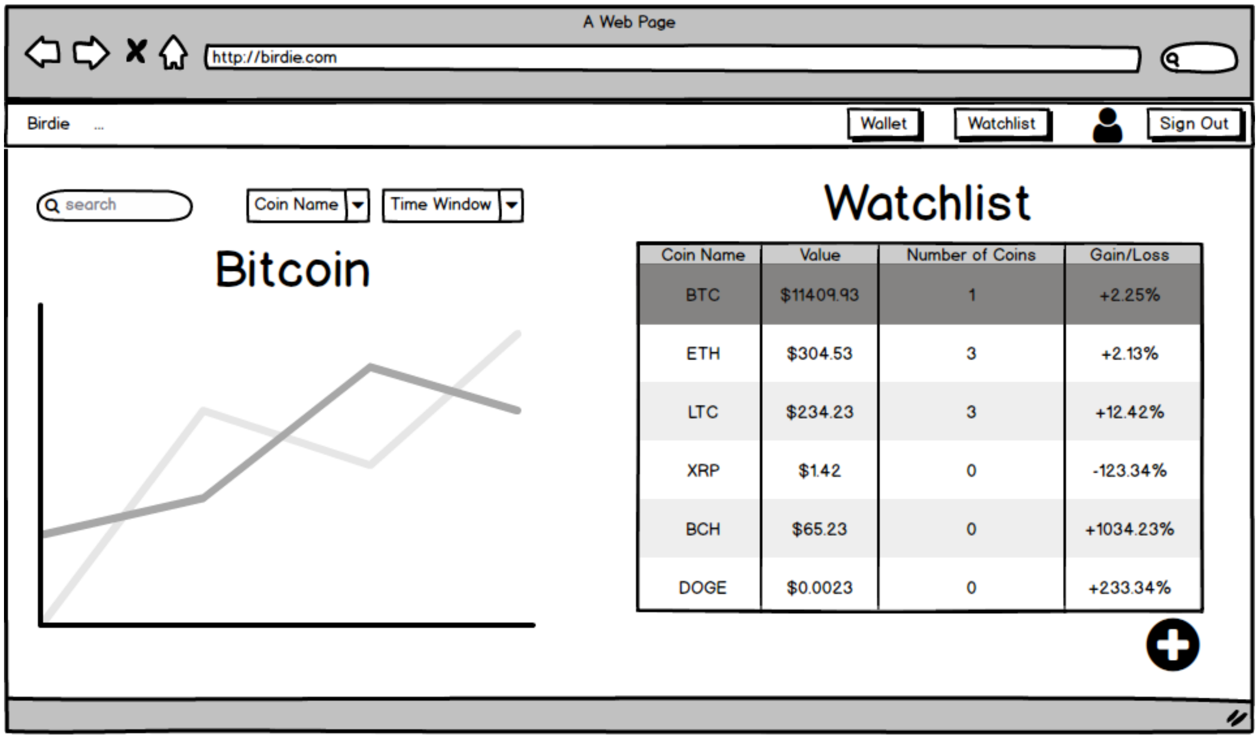
Coin Name		Gain/Loss
BTC	$11,450.23	+2.25%
ETH	$1,023.09	-52.25%
LTC	$300.12	+5.94%
XRP	$2.73	-203.23%
BCH	$101.00	+13.34%
DOGE	$0.004	+234.94%

Figure 4.



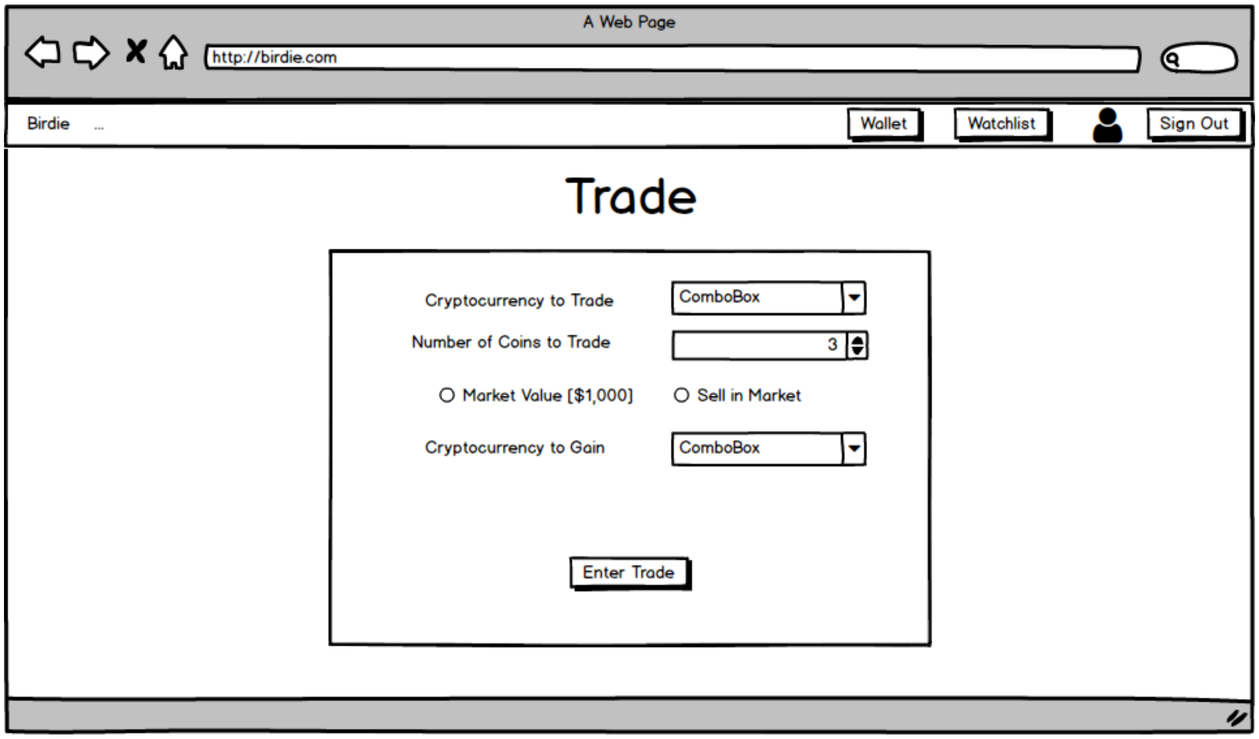
Coin Name	Value	Number of Coins	Gain/Loss
BTC	$11409.93	1	+2.25%
ETH	$304.53	3	+2.13%
LTC	$234.23	3	+12.42%
XRP	$1.42	5	-123.34%
BCH	$65.23	9	+1034.23%
DOGE	$0.0023	400	+233.34%

Figure 5.



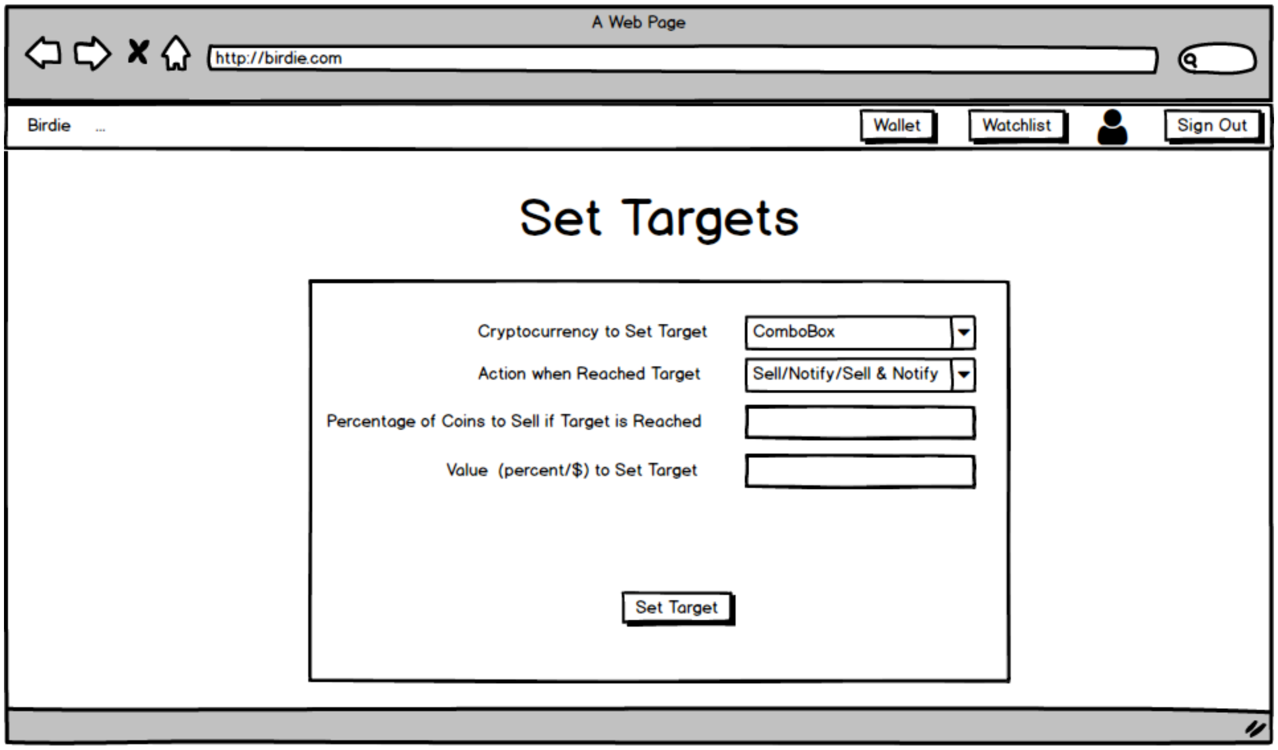
Coin Name	Value	Number of Coins	Gain/Loss
BTC	$11409.93	1	+2.25%
ETH	$304.53	3	+2.13%
LTC	$234.23	3	+12.42%
XRP	$1.42	0	-123.34%
BCH	$65.23	0	+1034.23%
DOGE	$0.0023	0	+233.34%

Figure 6.



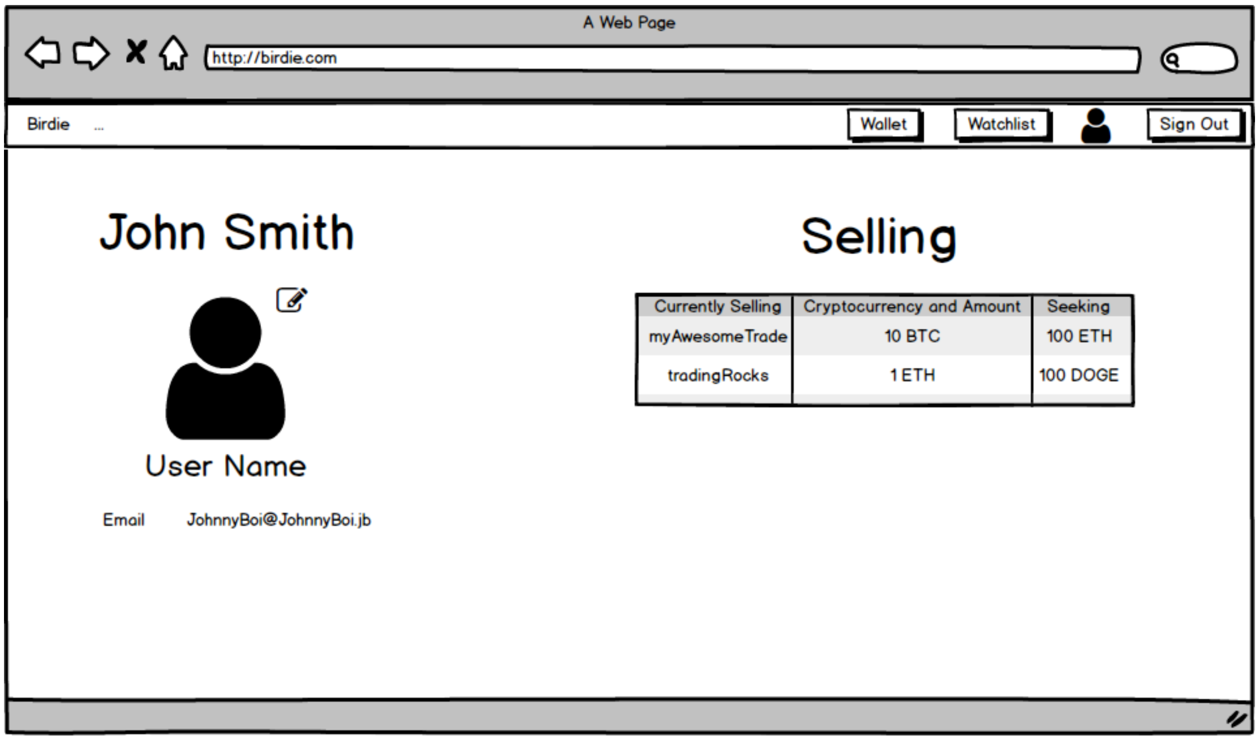
Cryptocurrency to Trade	ComboBox
Number of Coins to Trade	3
○ Market Value [$1,000]	○ Sell in Market
Cryptocurrency to Gain	ComboBox

Figure 7.



Cryptocurrency to Set Target	ComboBox
Action when Reached Target	Sell/Notify/Sell & Notify
Percentage of Coins to Sell if Target is Reached	
Value (percent/$) to Set Target	

Figure 8.



Email JohnnyBoi@JohnnyBoi.jb

Currently Selling	Cryptocurrency and Amount	Seeking
myAwesomeTrade	10 BTC	100 ETH
tradingRocks	1 ETH	100 DOGE

8. Development Plan

8.1 Development Schedule

- Development schedule will start on March 27th, with the possibility of starting on March 20th.
- Birdie will be completed on May 18th.
- After that, we will spend 2 weeks for testing the software and doing interviews/surveys with our clients, Kris and Kevin, to ensure that everything is in working order.