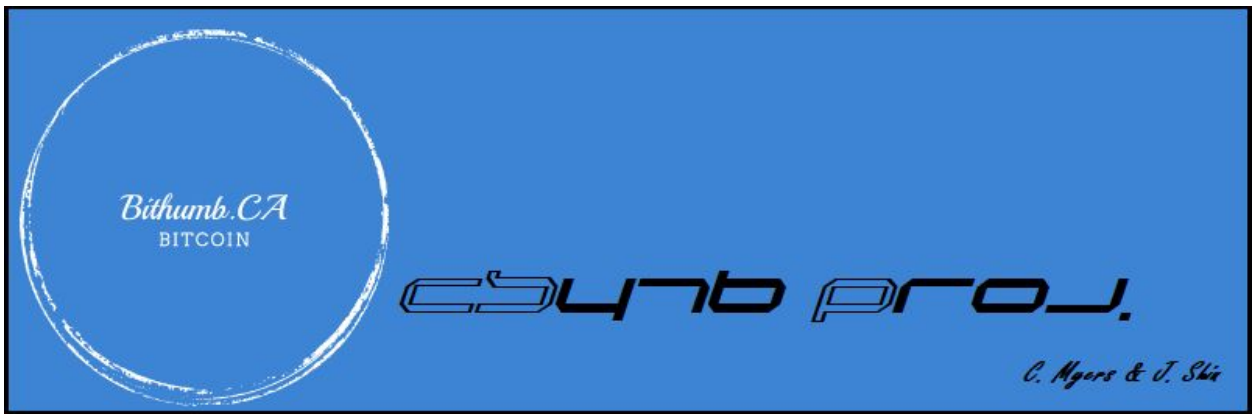


CS476: Group Project

Carter Myers **200 361 566**

Junho Shin **200 256 632**

<Table of Contents>

1. Problem Definition

Cryptocurrency is a digital asset that is starting to gain popularity in the public. However, there are many that do not understand the complexities of these non-traditional currencies. Concepts such as decentralization, generation, and -- the most appealing -- investing can be confusing and too technical for most lay people to understand. To show how bitcoin and other cryptocurrency investing actually works (on a rudimentary level), we are proposing a simple game as an introduction to cryptocurrencies.

To establish the virtual environment, we are going to make two parts in our project: the “mining” game and simple investment procedure. There are various way to obtain cryptocurrency, one way that many people are interested in is mining. Therefore, we are going to simplify the mining process as an arcade game, where users “mine” to earn in-game currency by playing. In cryptocurrency mining, as time goes, the difficulty escalates, as we will attempt to mimic with the arcade game difficulty. After earning in-game currency, users are able to purchase in-game collectibles, such as trophies. Additionally, users can “invest” in some cryptocurrency with a relative exchange rate. This way, users can simulate investing in cryptocurrency -- while earning (or losing) in-game currency -- with no real-life risk.

2. Economic Feasibility Study

Bitcoin has been of popular interest lately in the news. Its volatile, yet greatly rising price has caused excitement within the investing community. Before we attempt to build the proposed game system (that emulates a bitcoin mining and investing), let's investigate any similar services that are available. A quick search shows that there are multiple different games associated with bitcoin, mostly simple gambling sites that allow users to bet using bitcoins. This includes traditional casino games[1] as well as newly created games[2]; however, these services already require users to have an account with bitcoin(s). They offer no real educational purpose, and really only serve as gambling websites.

On the other hand, there exist some systems that have no ties to "real world" bitcoin, and they purely serve as games (some with an educational twist). For example, Bitcoin Millionaire[3] is a quiz-type game with dozens of questions to test a user's knowledge on bitcoin and cryptocurrencies in general. However, after these questions have been exhausted, the game is over (as there is nothing to earn), which makes this game small-scale with a short lifespan. On the other hand, the similarly named Bitcoin Billionaire[4] is a mobile game that simply simulates investing with bitcoin, but -- other than the name -- the game takes no information from real bitcoin. This is in contrast of another major type of bitcoin games, those which base in-game aspects on real-life bitcoin.

Of course, the aforementioned gambling sites fall into this category, but there are games that use concepts of cryptocurrency in creative ways, not just as a currency. Spells of Genesis[5] has based its in-game currency (cards) on blockchain technology,

---

[1] faucetgame.com
[2] bitkong.com
[3] webapp.bitcoinmillionaire-app.com
[4] noodlecake.com/games/bitcoin-billionaire
[5] spellsofgenesis.com

which mean these cards can be traded outside of the game altogether, and players truly own their cards. The overall content of the game, however, has no direct connection to bitcoin, and users don't need any knowledge of cryptocurrency to play. Users also don't need to know anything about bitcoin when playing Imperatum[6], but the game does feature a mode based on real time bitcoin prices. Basically, the game bases its difficulty on the current price of bitcoin and changes accordingly. This seems like an innovative, unique design for a game, but it still does not offer any education about the cryptocurrency.

While all of these games offer value in their own right, we wish to develop a system that has a combination of all of these components. We wish to develop

a) a simulation of mining/investing of bitcoin that is

b) based on real time data of bitcoin while

c) does not require users to pay with real bitcoin and

d) aims to educate users through this process.

While none of these details are novel in themselves, we hope that a mix of all of them will provide a unique system. We also want our users to feel a sense of accomplishment or direction towards a goal (unlike some previous systems), so, in addition, we will have an in-game currency for users that can be exchanged for small in-game changes (such as a different background). This in-game currency will also serve as the score that users judge their level of play on.

In summary, while no particular aspect of our proposed system is original, we hope that the unique combination of features will appeal to players, both those that are unfamiliar with cryptocurrency (in an effort to educate them) and those that are well-versed in the current technological craze (in an effort to entertain them).

[6] news.bitcoin.com/role-playing-game-imperatum-adds-dynamic-bitcoin-mode-difficulty

3. Software Requirement

a. Functional Requirement

i. Guest User

1. Create account
2. View Scoreboard
3. View Forum
4. View Item Shop
5. View About Us
6. View Main page with opening video

ii. Registered User

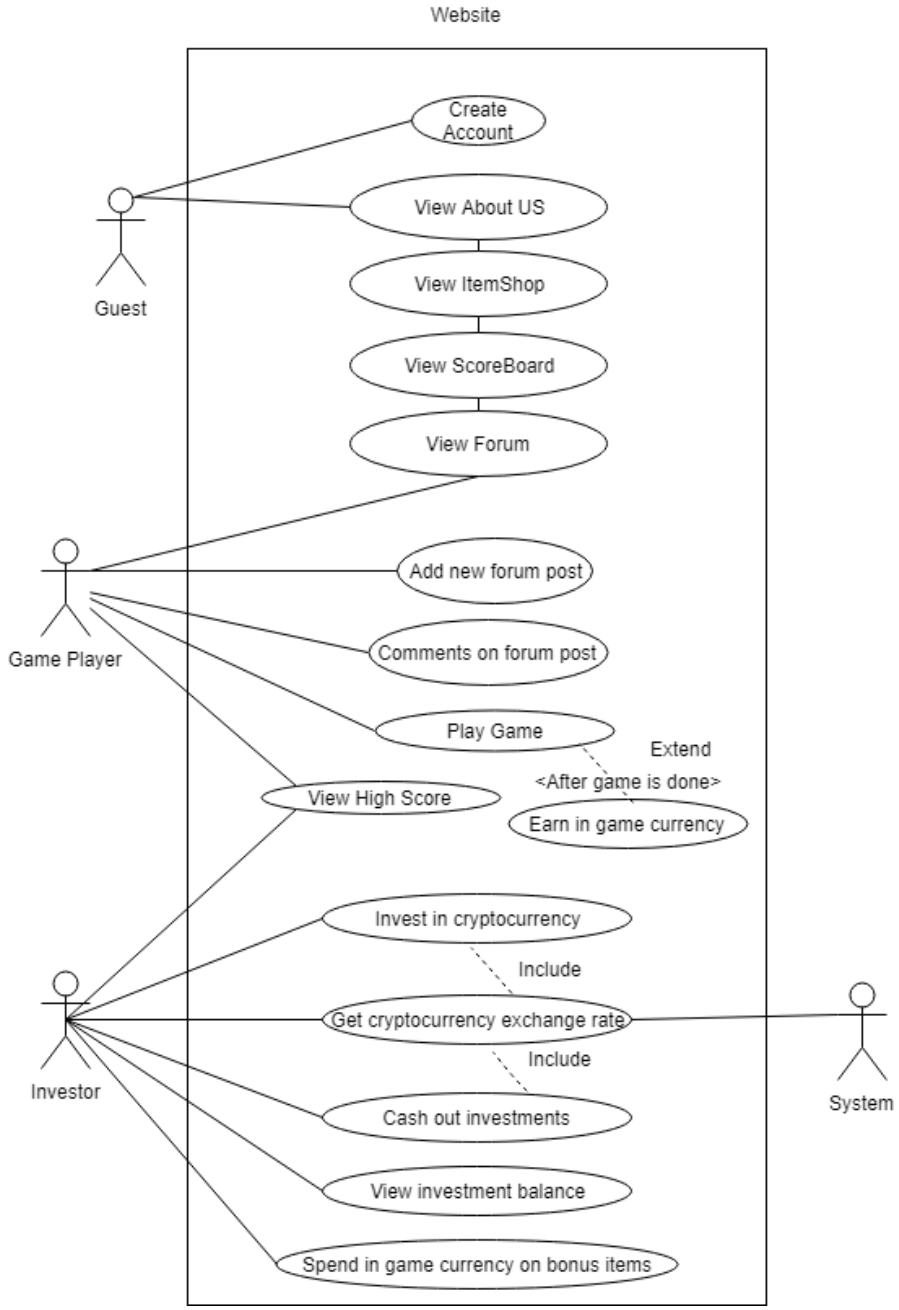
Game Player role

1. Log in / out
2. Play game on main page
3. Earn in-game currency through playing game
4. View high scores via scoreboard and profile page
5. Detail of user account that provided by profile page
6. Forum function use
    a. Add new thread
    b. Managing own threads / comments
    c. Reply to own / others thread
    d. Game becomes more difficult as lay goes on

Investor role

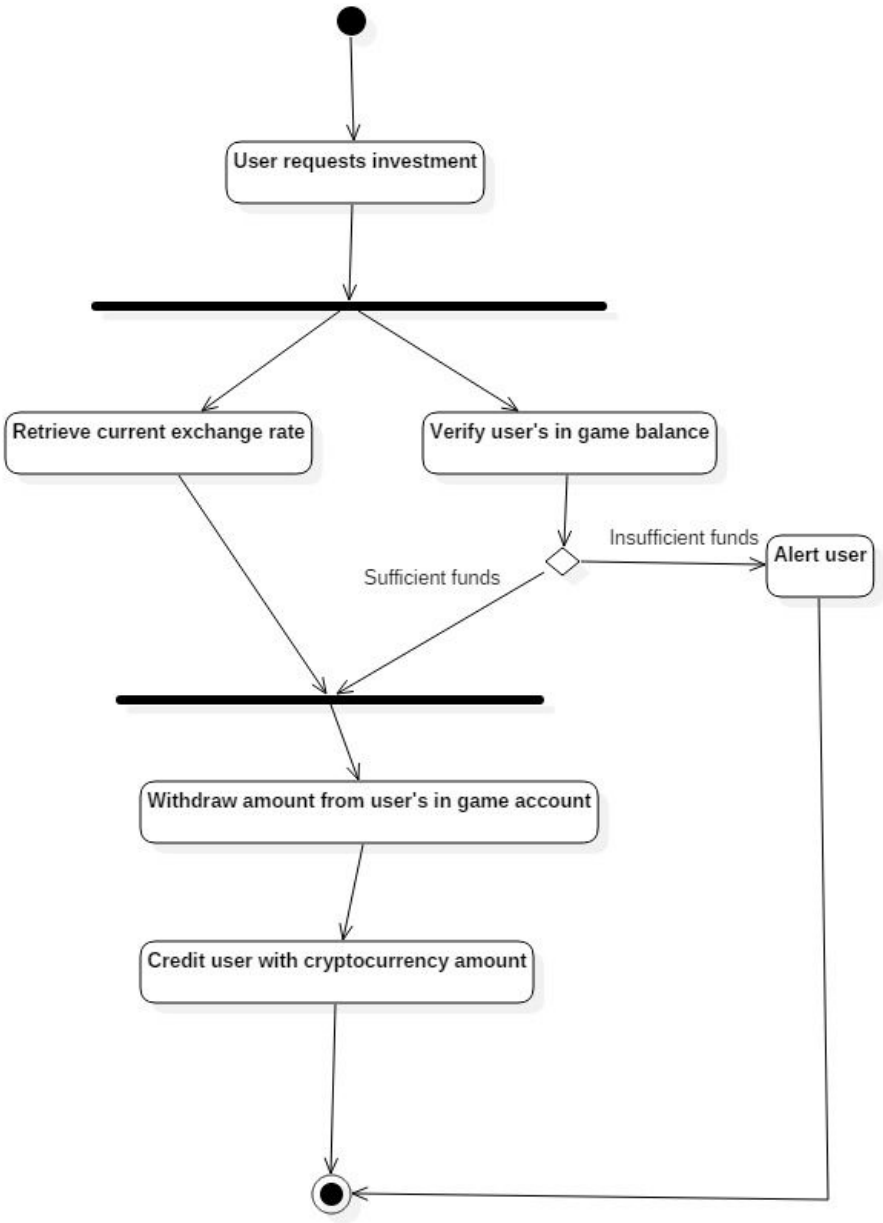
1. “Invest” in cryptocurrency with current exchange rate
2. View current balances for in-game currency and investments
3. Cash-out investments for in-game currency
4. View the cryptocurrency rate change graph
5. Buy items at Item Shop
6. “Invest” in cryptocurrency (at current exchange rate)

b. Use Case Diagram



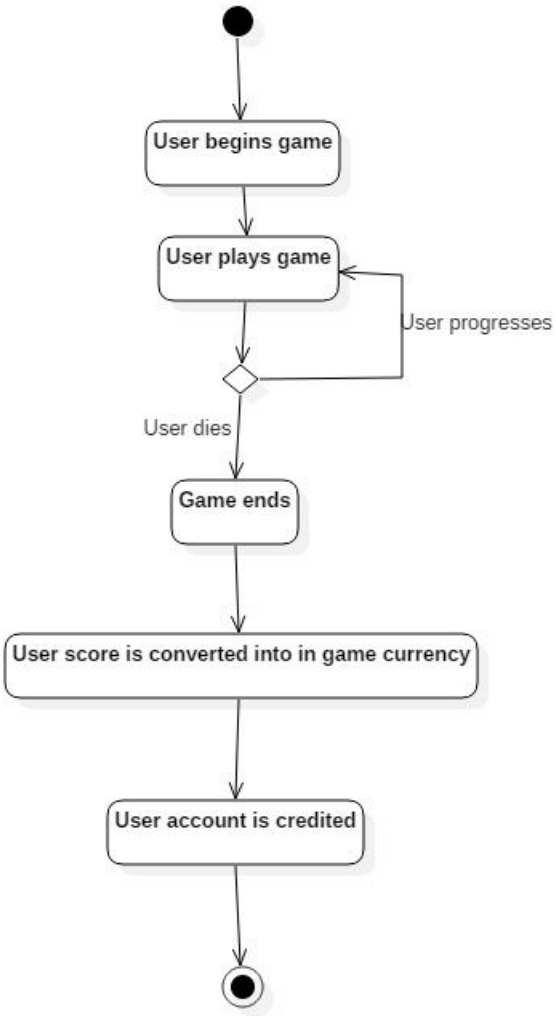
(Fig. uc1. General Use Case Diagram)

There are three users in general case which are the system, end-user, and the guest. End-user can be classified into two parts, game player and investor. End- user activity can be specified as above diagram (Fig. uc1)



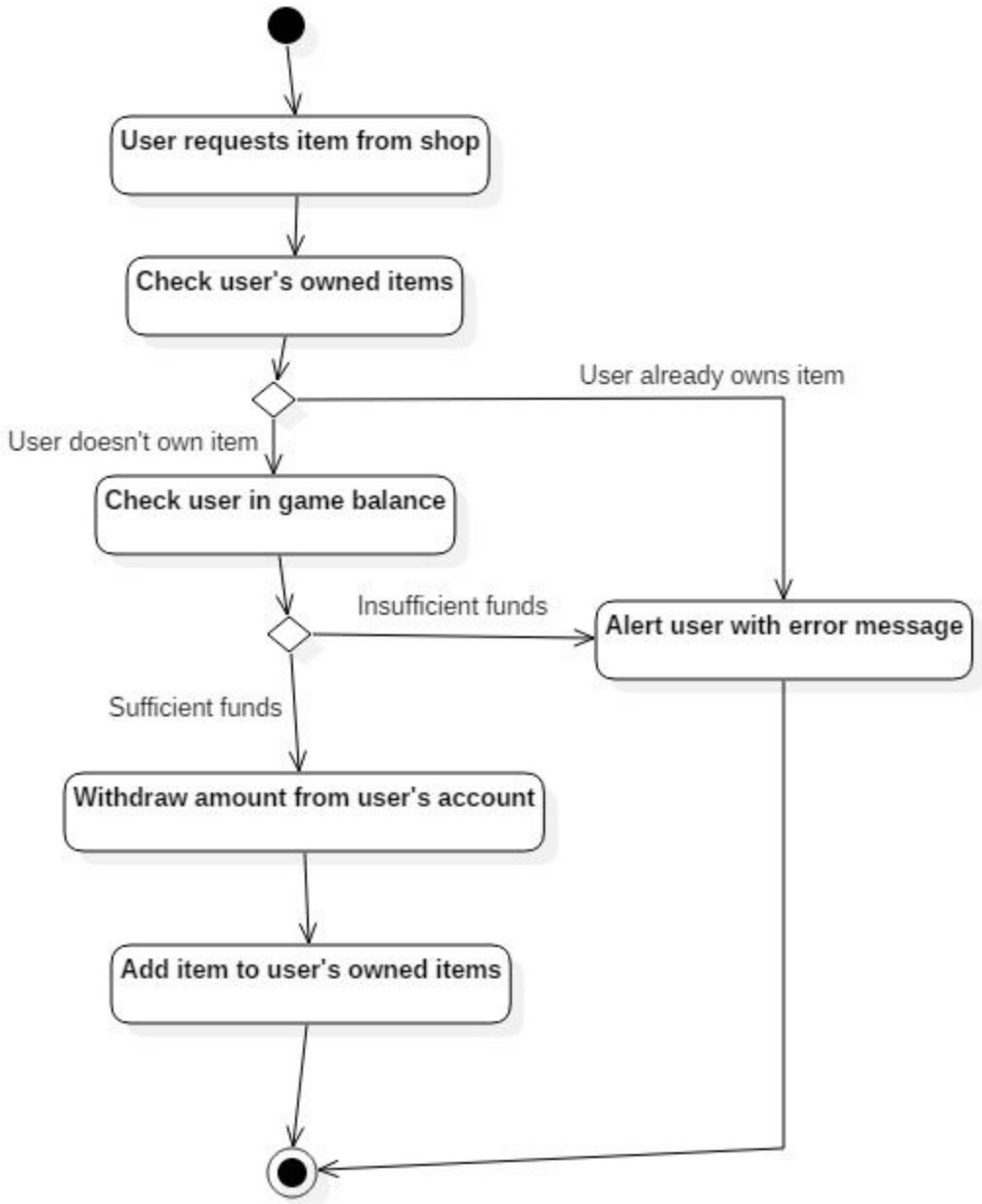
(Fig. UC2. Use Case 1)

Investors (as described in Fig. uc1) are able to cash out the cryptocurrency into in-game USD currency depending on real time rate (Bitcoin vs. USD). End-user will not be able to cash it out into USD currency if the coin user has is not enough. Alert pop-up window will notice end-user.



(Fig. UC3. Use Case 2)

This is an end-user as game player. They can start the game to mine cryptocurrency, and scores will be stored as game currency(cryptocurrency) after game ends.



(Fig. UC4. Use Case 3)

End-user purchases items that can be used in game. A checking function will decide how to display item list (grey out if item is owned by user already). Alert window will be popped up if the user proceed to buy greyed-out item. Then, credit balance will be checked. If the user does not have enough amount

then it also goes to alert window. Otherwise, the item user selects will be stored into user's item inventory.

c. Software Qualities

i. Correctness

- By focusing on testing procedures with sufficient time will reduce any correctness errors. Finding all the possible cases to be tested for higher correctness. Further test procedure details will be on the last chapter.

- We aim to reuse current libraries, frameworks, and other code as much as possible to ensure our system is correct. Since many operations (such as user log in operations, which are handled by our framework) are not new to this system, we can find well-known and already well-tested solutions.

ii. Robustness

- The system allows users to carry out normal actions and if something goes wrong then the user is notified with an informed error.

- Our web application will provide the users limited options to avoid robustness problems. We will do this by only showing operations which the user has valid credentials to complete (e.g. if they are logged in).

iii. Security

- All standard security procedures (such as password hashing, preventing cross-site scripting) should be taken into account in this aspect. However, these basic security features would not be enough today.

- Therefore we decided to run our application on the commercial web server that has practical security features already built-in. Due to our size of web application, the monthly pay for the server was very reasonable ($12/m). It also gave us huge time saving for

our project schedule.

iv. **Performance (Time Efficiency)**

- Every interaction with the user (i.e. every page that is loaded) should only take a maximum of 2 - 3 seconds to respond.

- Time efficiency will be tested on the last chapter of this report, further way of improvement will discussed.

v. **User Friendliness / Usability**

- The website will have a UX design which will be easy to use, and the paddling and the margins will be according to the World Wide Web Consortium web standards. User instructions will also be available on the site to guide users through its use.

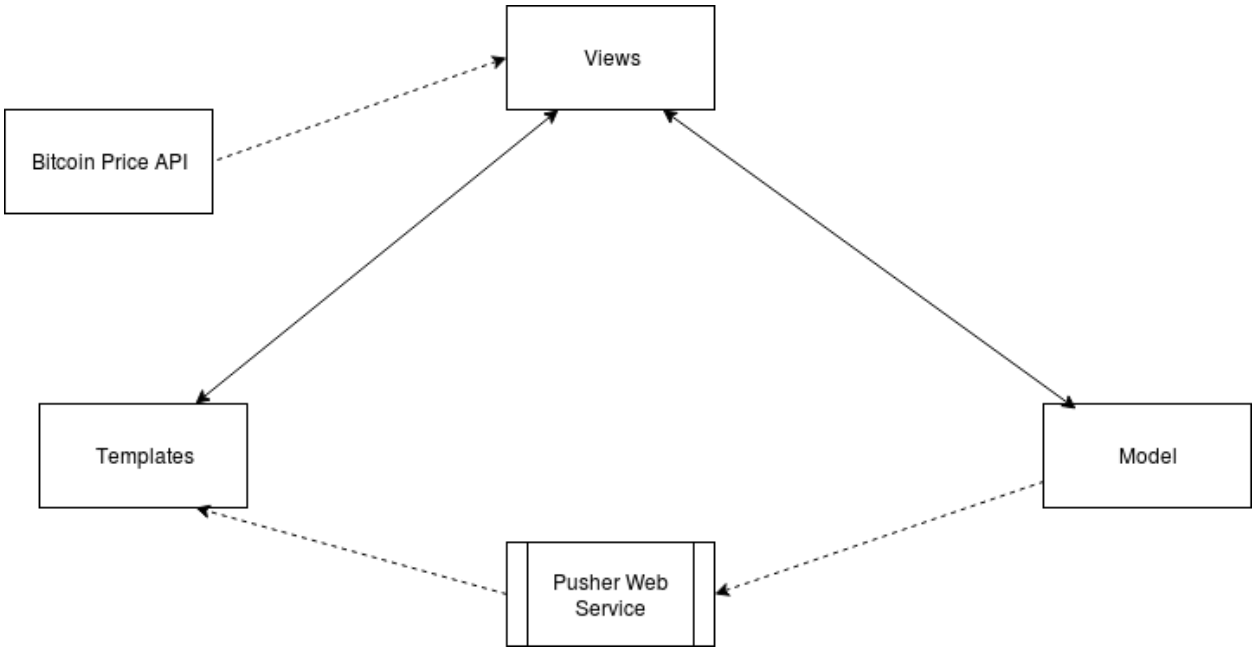
- We decided to use Bootstrap framework to build our front end side. The most of users are familiar with this design that bootstrap creates.

- From the above robustness quality, minimizing the user input also provides user friendliness. Maximizing the number of inputs for users creates a confusing and complex interface, and does not allow for users to easily communicate with.

4. Design Specification

a. Logical Software Architecture

The logical software architecture we chose for this project is a modification of the widespread Model-view-controller (MVC) architecture, called Model-template-view (MTV). The true differences between the two architectures are very slim, where the only real difference is that “views” in MVC are referred to as “templates” in MTV and the “controller” in MVC is referred to as a “view” in MTV[7]. Contrary to the MVC style of thinking, this view does not deal with the actual presentation of the data. Instead, the view is more similar to the business logic by querying the database, preparing information to be displayed in a specific template, and also saving information from users to models in the database.



Also featured in this high-level diagram of our software architecture are the extra web services and APIs that our system uses to fetch data and communicate between components. More specific information on these services

[7] djangobook.com/model-view-controller-design-pattern

is outlined in parts 5 and 6.

Our main motivation for choosing this architecture is, of course, that our back-end framework Django is designed with this architecture in mind, and it is virtually impossible to operate with Django in another architecture due to its structure. While the creators of Django say that different components within the framework itself can be considered to fit into the MVC model, some sources claim that the actual framework itself serves as the controller[8], as it handles each incoming request and selects the proper view (or template) to be displayed.This is done by parsing the request's URL and then calling a view function defined by the user. In addition, the framework abstracts any database accesses, similar to a controller. This complex behaviour shows that a view in MTV does not act as a holistic controller (as it takes help from the framework itself), but, for our purposes, we will consider any Django views to be analogous to the controller. In essence, MTV can be regarded as the same as the MVC architecture; however, to comply with the naming conventions of our framework (Django), it is clearer to address each component in the MTV architecture.

As MTV is so similar to MVC, it has much of the same advantages. This includes the ability to develop separate components simultaneously, reusability (because each component can be separated and reused), high cohesion (as each component/model can be designed for specific, self-contained operations), and low coupling.

8 https://doc.lagout.org/programmation/Django/Django%20Design%20Patterns%20and%20Best%20Practices%20%5BRavindran%202015-03-26%5D.pdf

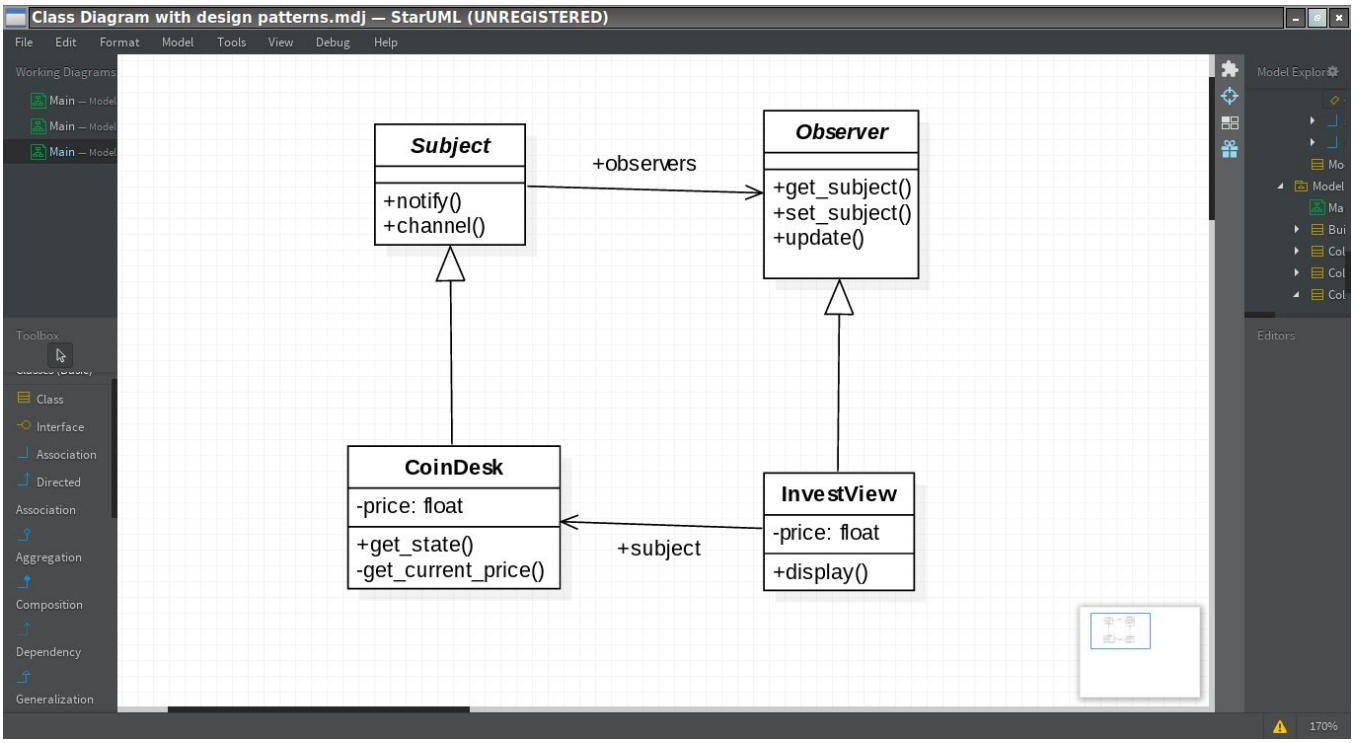
b. Design Pattern Notes:

<Observer Pattern>

The first pattern we chose for our project was the Observer Pattern. One reason being that it works well with the MVC architecture (which is closely related to our MTV architecture), but the main reason is to allow necessary, constant updates for our users' templates. In our specific example, where we simulate investing in Bitcoin with our in-game currency, it is important that users see the current exchange rate to make an informed decision, and this value changes very often due to Bitcoin's volatility. For this reason, we have a single subject continuously polling the price (using bitcoin_price_api) and updating observers whenever the price changes.

Our exact implementation involves a messaging service called Pusher that allows events on our web server (such as a change in state of the subject) to push data to clients. This happens by using a distinct name over a channel (similar to a web socket), but this is incompatible with the traditional Observer method, since each observer must register itself with the subject during runtime. In our case, the observer has no access to the subject during runtime, but it does know the subject's name. So each subject broadcasts any changes it encounters on a channel using its own name, and any observers register themselves to that channel to receive the pushed data. While each subject is unaware of all of its observers specifically, it can still see if there are any observers waiting for updates, and updates are only pushed when necessary. While slightly different than the original pattern, this is still very effective.

Here is our class diagram to implement the observer pattern:



And here is the pseudocode for each class. It should be noted that, in this specific case, once the observer receives the update in price, it simply displays it without any decision logic.

```
observer/models.py

class Subject:

    def channel():

        return self.name


    #this is the notify function with a push method

    def notify():

        #connect to messaging service

        channel = channel()
```

```
        pusher = pusher.connect(channel)
        #only notify if there are observers on the channel
        if not pusher.occupied(channel):
            return
        pusher.send_message(channel, self.get_state())
```

---

observer/models.py

```
class Observer:
    def get_subject(self):
        return self._subject

    def set_subject(self, subject):
        self._subject = subject

    def update(subject):
        channel = subject.channel()
        pusher.connect(channel)
        pusher.listen(channel)
```

---

bitcoin_price_api/exchanges/coindesk.py

```
class CoinDesk inherits Subject:
    def get_state():
        #query the api and return the price
        return self.get_current_price()
```

```
    #this is the set_state function
    Def get_current_price():
        #the api is queried here to get the current price
```

```
invest/views.py
Class InvestView inherits Observer:
    #constructor that sets the subject
    def __init__(subject):
        self.set_subject(CoinDesk)

    def display():
        #this webpage contains the logic for registering and
        display (see below)
        return 'templates/invest/invest.html'
```

To be completely clear, here is the basic algorithm for updating prices.

Subject polling and notifications:

```
bitcoin_price_api/update_price.py
while True:
    new_state = subject.get_state()
    if new_state != old_state:
        old_state = new_state
        subject.notify()
```

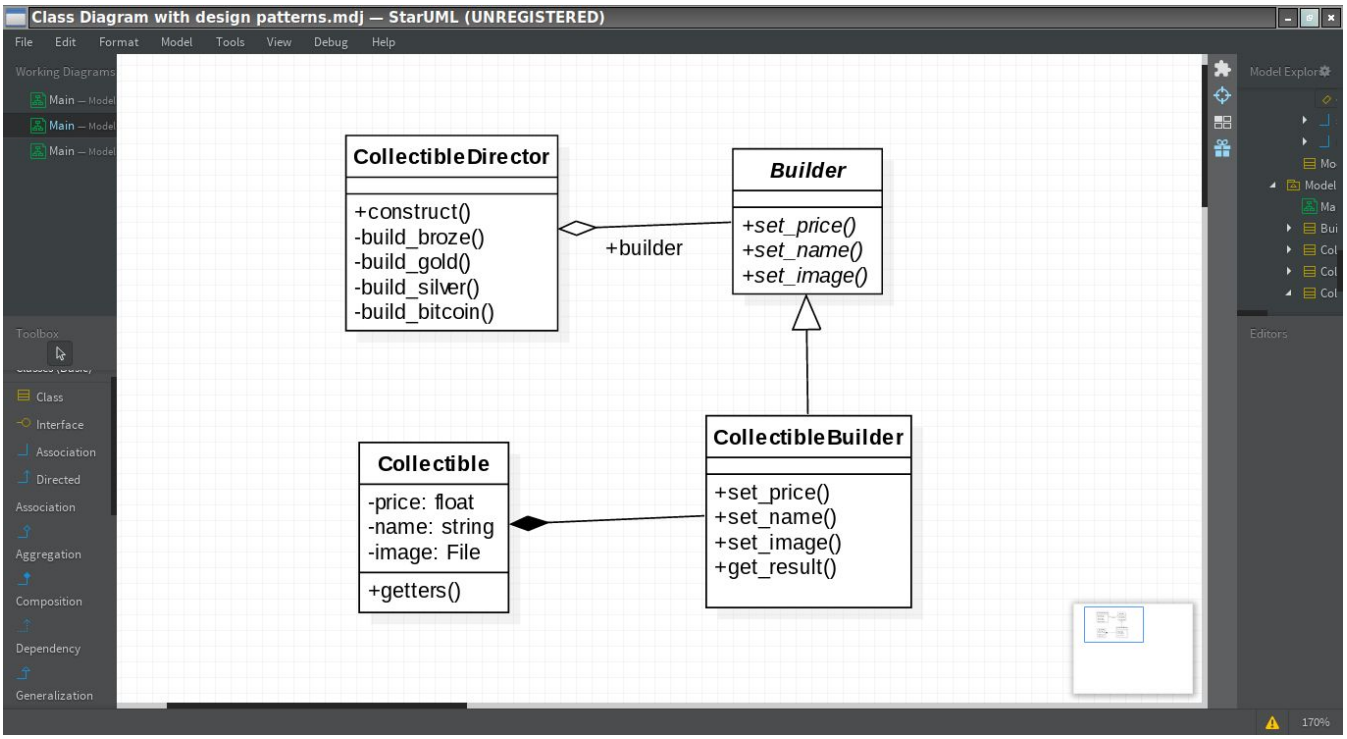
Observer registration and display logic:

```
templates/invest/invest.html

Invest.update(subject)     #register to specific channel

if event:

      #update price on page
```

<Builder>

The next design pattern we chose was the builder pattern. While relatively simple, the builder pattern can be very useful in abstracting the creation of complex objects. Since we use a specialized object of Collectibles in our project (which contains a fixed number at any given time), we've decided to create these objects using the builder pattern, which not only allows us to easily construct, modify, and update these objects, but we can easily add or remove objects with -- most importantly -- different representations. Currently, all of the collectibles have the same structure, but it is possible that we may want to add a "limited-time collectible" or a collectible that is collected using different methods (e.g. by obtaining a highscore). Overall, this builder design pattern offers a very loosely coupled approach to constructing our collectibles.

Here is our specific class diagram for the builder pattern.



Here is the pseudocode for each class:

```
builder/models.py

class Builder:

      #these are all purely abstract methods

    def set_price(self, price):

        pass


    def set_name(self, name):

        pass


    def set_image(self, src):
```

```
        pass
```

---

invest/models.py

```
class CollectibleBuilder inherits Builder:

     #constructor

     def __init__(self):

         self.collectible = Collectible()


     def set_price(self, price):

          self.collectible.price = price


     def set_name(self, name):

          self.collectible.name = name


     def set_image(self, src):

          self.collectible.image = src


     def get_result(self):

          return self.collectible
```

---

invest/construct_collectibles.py

```
class CollectibleDirector:

     #these are all static methods

     def construct():
```

```
    CollectibleDirector.build_bronze()
    CollectibleDirector.build_silver()
    CollectibleDirector.build_gold()
    CollectibleDirector.build_bitcoin()


def build_bronze():
    builder = CollectibleBuilder()
    builder.set_name("Bronze trophy")
    builder.set_price(100)
    builder.set_image('collectibles/bronze.png')


def build_silver():
    builder = CollectibleBuilder()
    builder.set_name("Silver trophy")
    builder.set_price(1000)
    builder.set_image('collectibles/silver.png')

def build_gold():
    builder = CollectibleBuilder()
    builder.set_name("Gold trophy")
    builder.set_price(10000)
    builder.set_image('collectibles/gold.png')
```

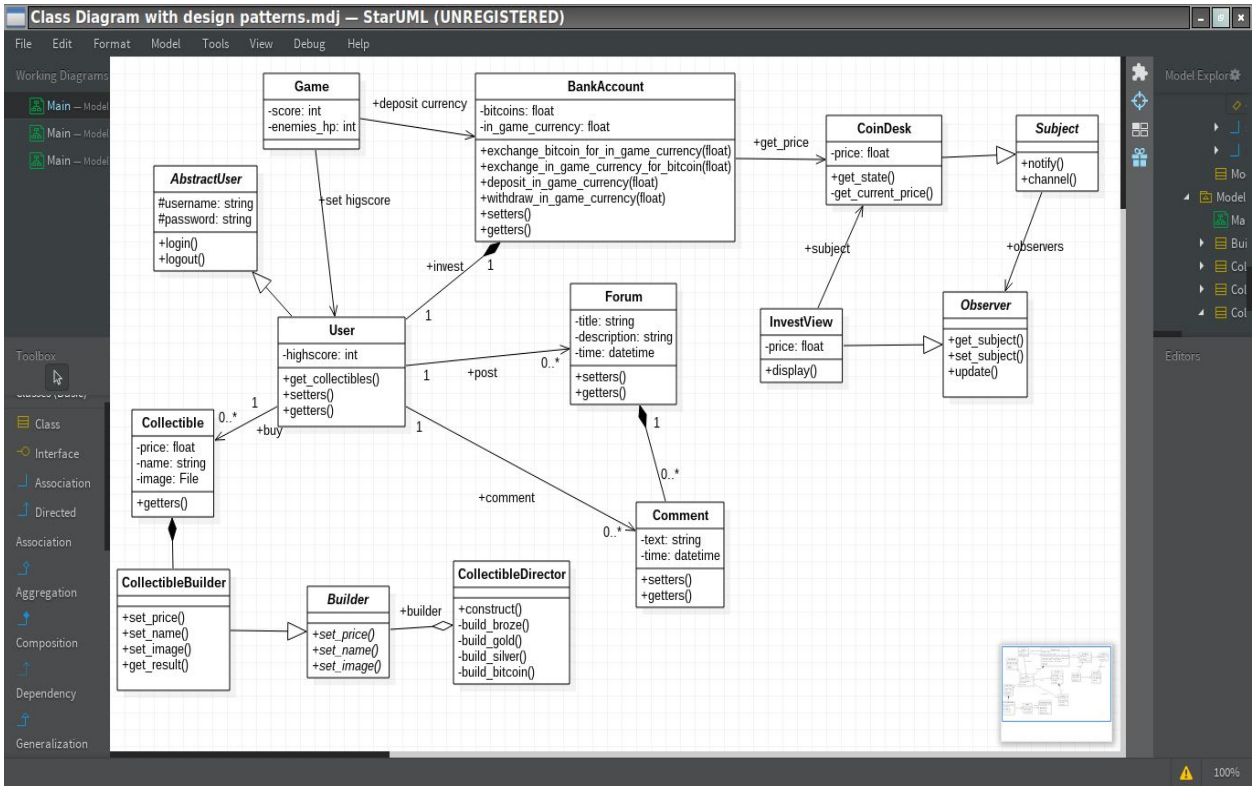
```
    def build_bitcoin():
        builder = CollectibleBuilder()
        builder.set_name("Bitcoin trophy")
        builder.set_price(100000)
        builder.set_image('collectibles/bitcoin.png')
```

And here is the main algorithm used to construct all of the collectibles objects.

```
invest/construct_collectibles.py
#only create if there are no collectibles (because we are creating
    all at once here)
if len(Collectible.objects.all()) > 0:
    print "Collectibles not empty. Exiting script."
else:
    #and construct them all:
    CollectibleDirector.construct()
```

c. Class Diagram

On the following page is our complete class diagram. It should be noted that, for sake of simplicity, setters and getters for classes are abbreviated (unless the explicit methods are needed for a design pattern). Also for the sake of simplicity, even though some methods inherit from the built-in Model class of our framework (to facilitate database operations), this class is omitted from the diagram, as it does not contribute in any other significant design.



d. UML tools

For our UML tool, we chose starUML[9] to model our design for this project. Here are a list of pros and cons of using this program.

Pros:

- Intuitive GUI - it's very simple to select and position specific graphical objects in this program
- Workstation Oriented - We can work offline/standalone without network on a system.
- Widely used - Any problems are easy to search online, and a lot of experienced users provide solutions. Furthermore, there are even some simple tutorials freely available.
- Open Source - this program is free to use, especially helpful for students.

Cons:

- Registration - Often, the program will ask to register to purchase the product, which can become annoying when working.
- Rigid formats - When trying to edit attributes or class names, certain characters are not permitted within the field, making it hard to accurately name some components.

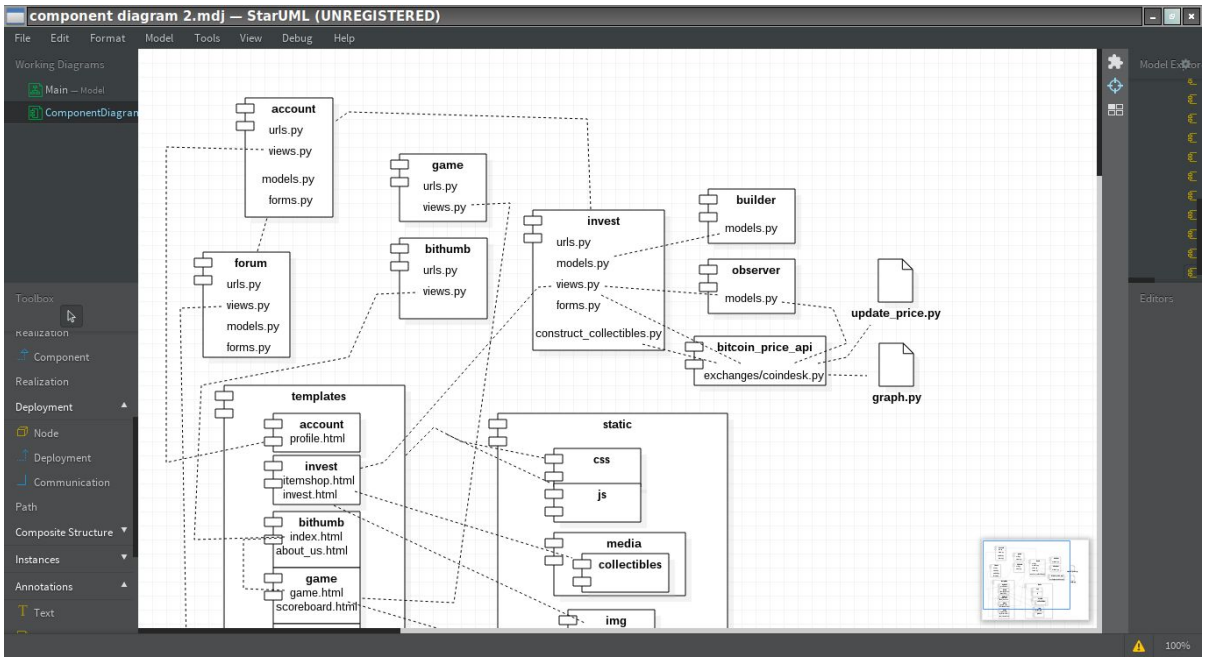
Overall, we found StarUML a helpful tool, and we would reuse this in future projects.

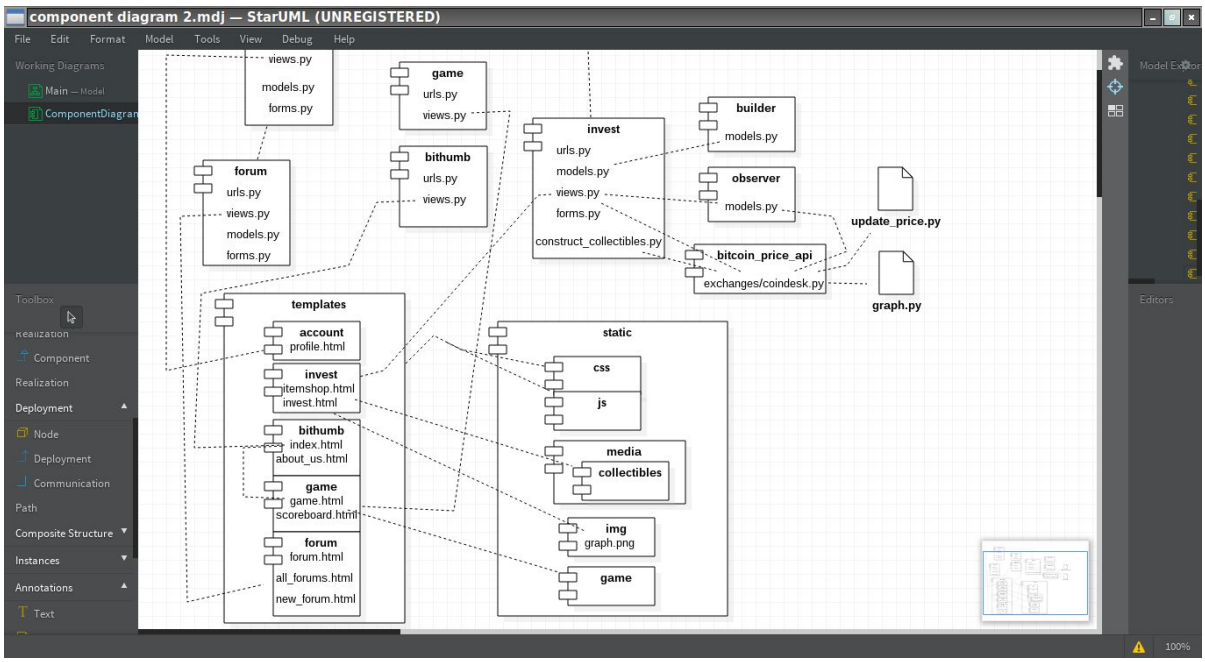
[9] http://staruml.io/

5. Program

a. Component Diagram

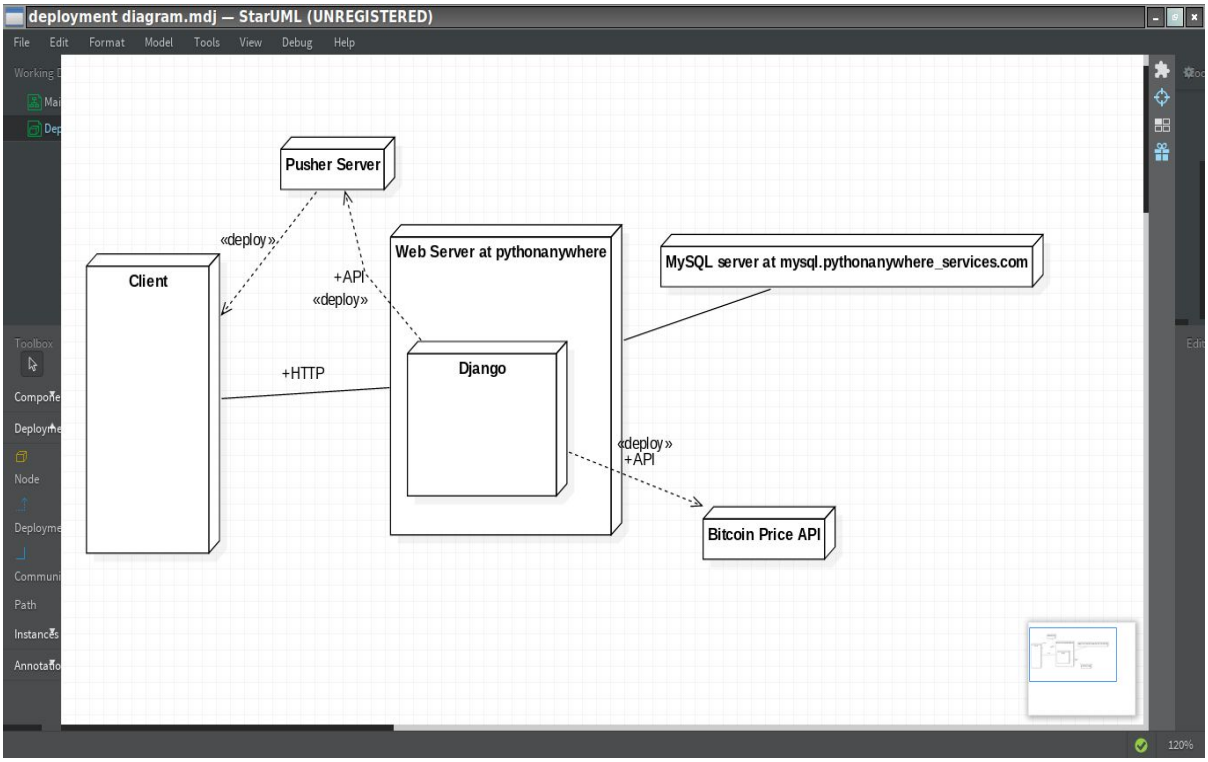
Our component diagram is shown below with all relevant connections. It should be noted that any file within the same component obviously can connect to all other files within the same component.





b. Deployment Diagram

Here is our deployment diagram.



The client node contains the presentation layer and runs on mobile and PC on Firefox, Chrome, and IE/Edge browsers. The Web server deploys the business logic and is run with the service pythonanywhere. We were unable to locate the exact location of this server, but it is somewhere in the U. S. Finally, using the same service as our database provider, a separate server hosts our datastore using MySQL. There are also third party servers (Pusher Server and the Bitcoin Price API)

that our system uses, as described in part 6.

c. List of classes

Directory	Class Name	File
Account	SignupForm(forms.ModelForm)	account/forms.py
	LoginForm(forms.Form)	
	User(AbstractUser)	account/models.py
bitcoin_price_api	CoinDesk(Subject)	bitcoin_price_api/coindesk.py
	BitcoinPriceApiConfig(AppConfig)	bitcoin_price_api/apps.py
builder	BuilderConfig(AppConfig)	builder/apps.py
forum	ForumForm(forms.ModelForm)	forum/forms.py
	CommentForm(forms.ModelForm)	
	Forum(models.Model)	forum/models.py
	Comments(models.Model)	
invest	CollectibleConstructor	invest/construct_collectibles.py
	BitcoinToInGameCurrencyForm(forms.Form)	invest/forms.py
	InGameCurrencyToBitcoinForm(forms.Form)	
	BankAccount(models.Model)	invest/models.py
	Collectible(models.Model)	
	CollectibleBuilder(Builder)	
	BankAccountToCollectible(models.Model)	
	Invest(Observer)	invest/views.py
observer	ObserverConfig(AppConfig)	observer/apps.py
	Observer(View)	observer/models.py
	Subject(object)	

d. Tables of system Data

Here are screenshots of all of our database tables. To include all relevant columns, some queries are limited.

```
mysql> select substring(password, 1, 20), username, bank_account_id, highscore from account_user;
+----------------------------+-----------+----------------+-----------+
| substring(password, 1, 20) | username  | bank_account_id | highscore |
+----------------------------+-----------+----------------+-----------+
| pbkdf2_sha256$36000$       | carter    |              1 |        56 |
| pbkdf2_sha256$36000$       | new       |              2 |         0 |
| pbkdf2_sha256$36000$       | asdfasd   |              3 |         0 |
| pbkdf2_sha256$36000$       | shin202j  |              4 |       150 |
| pbkdf2_sha256$36000$       | highscore |              5 |        18 |
+----------------------------+-----------+----------------+-----------+
5 rows in set (0.00 sec)

mysql> select * from invest_bankaccount;
+----+-----------+-----------------+
| id | bitcoins  | in_game_currency |
+----+-----------+-----------------+
|  1 | 0.01628214 |            1.00 |
|  2 | 0.00000000 |            0.00 |
|  3 | 0.00000000 |            0.00 |
|  4 | 0.02996449 |            0.00 |
|  5 | 0.00208337 |            0.00 |
+----+-----------+-----------------+
5 rows in set (0.00 sec)
```

```
mysql> Select * from invest_collectible
    -> ;
+----+-----------+----------------+---------------------------+
| id | price     | name           | image                     |
+----+-----------+----------------+---------------------------+
|  5 |    100.00 | Bronze trophy  | collectibles/bronze.png   |
|  6 |   1000.00 | Silver trophy  | collectibles/silver.png   |
|  7 |  10000.00 | Gold trophy    | collectibles/gold.png     |
|  8 | 100000.00 | Bitcoin trophy | collectibles/bitcoin.png  |
+----+-----------+----------------+---------------------------+
4 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM invest_bankaccounttocollectible;
+----+------------+----------------+
| id | account_id | collectible_id |
+----+------------+----------------+
|  1 |          4 |              5 |
|  2 |          5 |              6 |
+----+------------+----------------+
2 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM forum_comments LIMIT 5;
+----+--------------------------+----------------------------+---------+---------+
| id | text                     | time                       | post_id | user_id |
+----+--------------------------+----------------------------+---------+---------+
|  2 | asdf                     | 2018-03-20 16:22:10.380959 |       2 |       1 |
|  3 | ok                       | 2018-03-21 23:09:33.726240 |       1 |       1 |
|  4 | ㅁㄴㅇㄹ                  | 2018-03-24 00:18:53.547229 |       6 |       4 |
|  6 | FORUM TESTING, SAY CHEESE | 2018-03-24 21:29:15.387151 |       8 |       5 |
+----+--------------------------+----------------------------+---------+---------+
4 rows in set (0.00 sec)

mysql> SELECT * FROM forum_forum LIMIT 5;
+----+------------------------+-----------------+----------------------------+-----------+
| id | title                  | description     | time                       | poster_id |
+----+------------------------+-----------------+----------------------------+-----------+
|  1 | This is the first forum | or is it???     | 2018-03-20 05:07:38.892859 |         1 |
|  2 | asdflkj                | asdflkj         | 2018-03-20 16:22:01.197151 |         1 |
|  4 | Weclome to Bithumb     | 다국어 확인      | 2018-03-24 00:17:41.337619 |         4 |
|  5 | 다국어 확인             | 다국어 확인      | 2018-03-24 00:18:12.852589 |         4 |
|  6 | 多國語 確認             |                 | 2018-03-24 00:18:43.825387 |         4 |
+----+------------------------+-----------------+----------------------------+-----------+
5 rows in set (0.00 sec)
```

e. Our website is located at www.bithumb.ca.

6. Technical Documentations

a. Programming Languages

i. Javascript
   1. AJAX requests/handling
   2. Game

ii. HTML / CSS / JS
   1. Front-end presentation

iii. Python
   1. Back-end logic and events

b. Reused algorithms and programs

- Reused/modified code from bitcoin-price-api[10], most importantly, the CoinDesk class in exchanges/coindesk.py

- Reused/modified code from django-pusherable[11] and pusher-http-python[12] This includes the Pusher library and some methods in the classes InvestView and CoinDesk (see django-pusherable)

[10] https://github.com/dursk/bitcoin-price-api
[11] https://github.com/pusher/django-pusherable
[12] https://github.com/pusher/pusher-http-python

c. Software tools and environment

- Django
  - This Python framework is used for our entire backend framework.

- Easy-thumbnails
  - This Django library is used on our templates (.html pages) to easily display pictures

- Matplotlib
  - This is a Python library for data visualization, and it is used by a script on our server to generate a graph based on the history of Bitcoin exchange rate

- Pusher
  - This is a messaging service that (along with it's APIs mentioned above) allows our models to communicate asynchronously with our templates (as in our Observer pattern)

- Github
  - This is used for version control and deployment of our source code.

- Google Drive
  - Online group work
    - Scheduling for group meeting
    - Data sharing
    - Online documentation work

- Construct 3[13] (HTML5 game framework)

    - 2D game framework with intuitive GUI
    - Used 4 layers(stage, start -> game -> death -> score update)

    - Supporting Ajax to communicate with DB
        - Final scores => our DB

    - Game reused resources
        - Player picture[14] (Teemo)
        - Enemy picture[15] (Nasus)
        - Smiling Teemo[16] (the last layer of the game)
        - Rest resources are supported by Construct 3 (Paid)

- Pingdom[17] (web performance tool)
    - Testing each routine (function) of the web application
    - Loading time with specific details (diagnosis)

- IoTcube[18] (TLS vulnerability testing tool)
    - Transport Layer Security test
    - Examine for both physical and logical network architecture.
    - Supporting mitigation and technical documents for developers

- Biteable[19] (video maker for opening)
    - Online video maker and streaming service

---

[13] https://www.construct.net/kr
[14] http://deck-heroes.wikia.com/wiki/File:Super_Teemo.png
[15] http://leagueoflegends.wikia.com/wiki/File:Nasus_Render.png
[16] https://play.google.com/store/apps/details?id=com.companyname.chickenegg
[17] https://tools.pingdom.com
[18] https://iotcube.korea.ac.kr/#testSeg
[19] https://biteable.com/

    - Full paid version will offer non-ad environment
    - This is the solution for our time efficiency problem of web application

- Bootstrap[20] (HTML/CSS/JS framework)
    - The world biggest HTML/CSS/JS library
    - Every web pages contain bootstrap library calls
    - To design our pages layout

- Fiverr[21] (Logo Creator)
    - Main page logo
    - In-game logo

[20] https://getbootstrap.com/
[21] https://www.fiverr.com

7. Acceptance Testing

a. Functional Test

i. Dynamic main page and game score updates

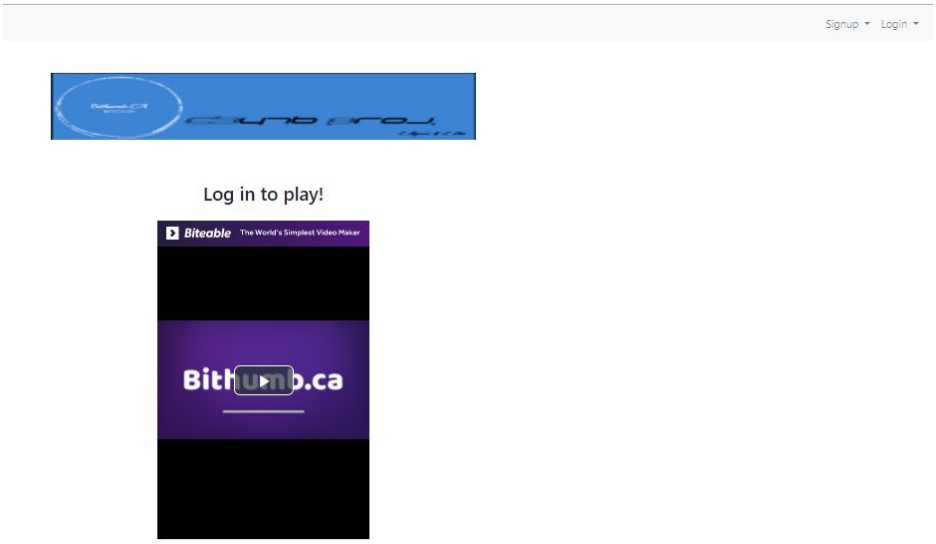
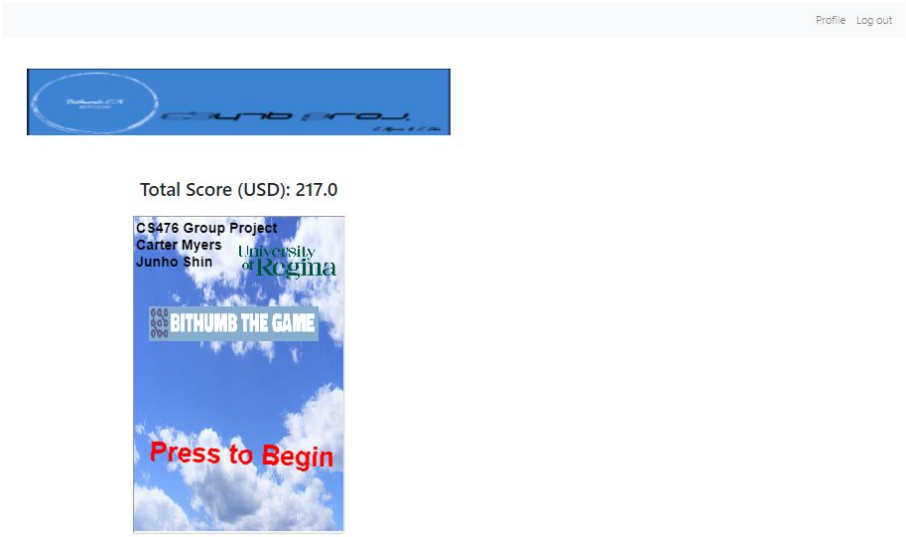
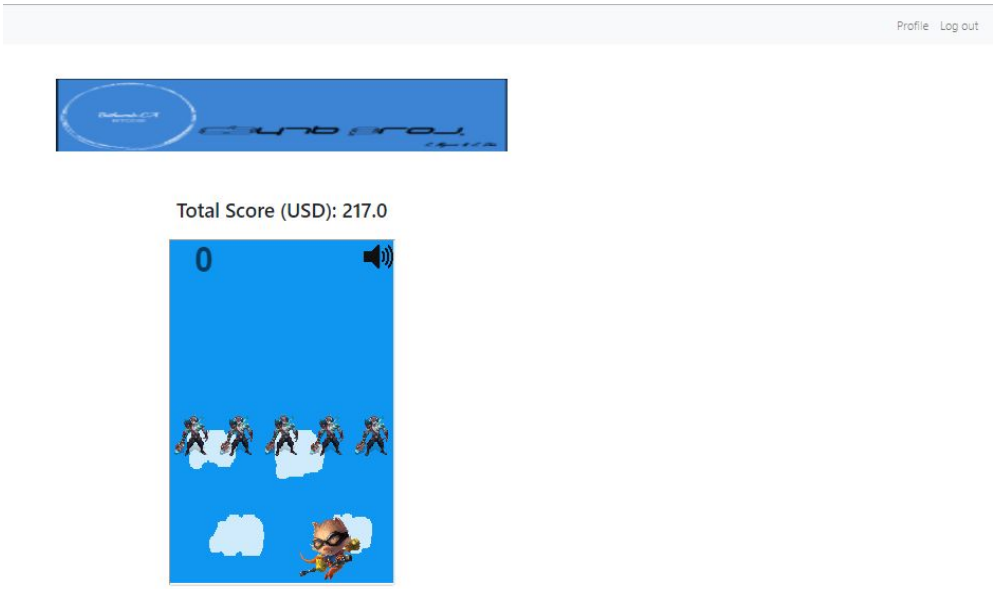


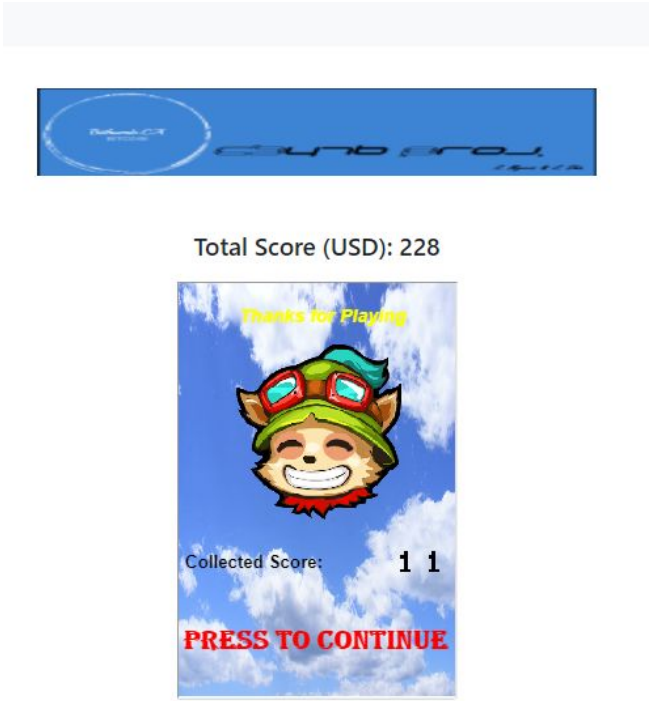
Figure of main page as in user of guest status.



After logging in as a registered user, the game frame will be shown, and users can begin playing.

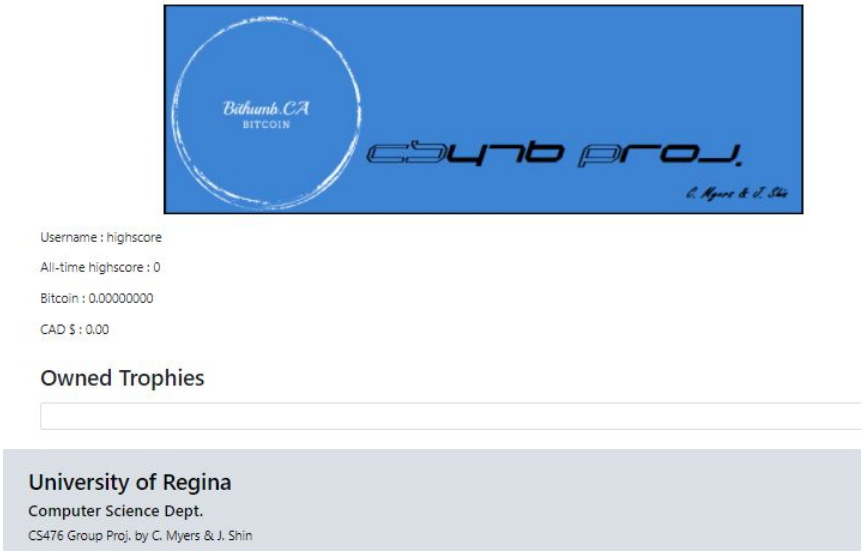


Game playing ...

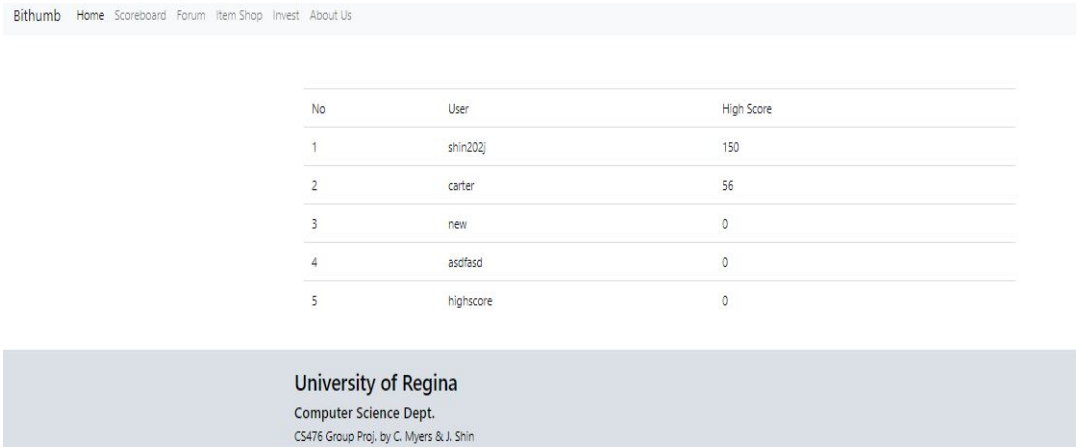


Final score will be updated onto total score of the current user.
It will be added cumulatively.

ii. **High score tracking**



Newly made user's profile



As an account is created, it will update the scoreboard right away. Now it's showing "highscore" user has 0 as highest game score in he/she has ever played

Total Score (USD): 18

After playing first game, user earned 18 scores from the game and it's added onto "Total Score"

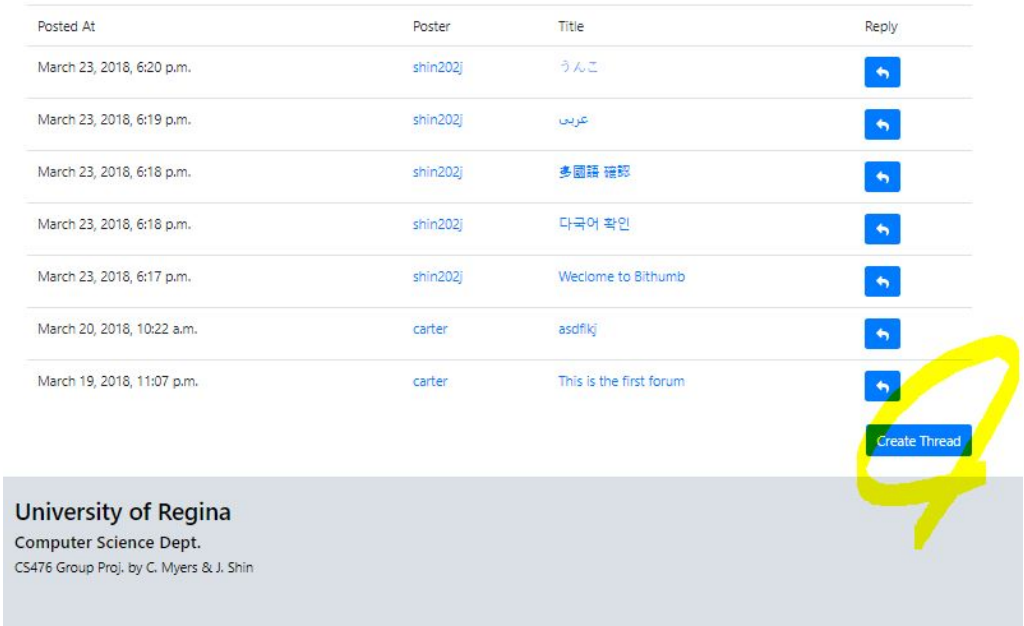
op Invest About Us

No	User	High Score
1	shin202j	150
2	carter	56
3	highscore	18
4	new	0
5	asdfasd	0

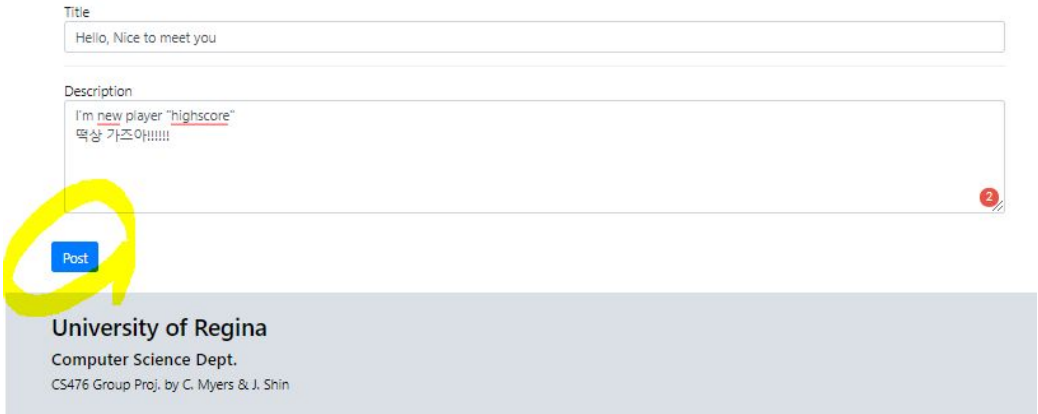
University of Regina
Computer Science Dept.
CS476 Group Proj. by C. Myers & J. Shin

Next, the user's highest score record has been updated and it is applied in the above list. Score Board list is re-sorted as well.

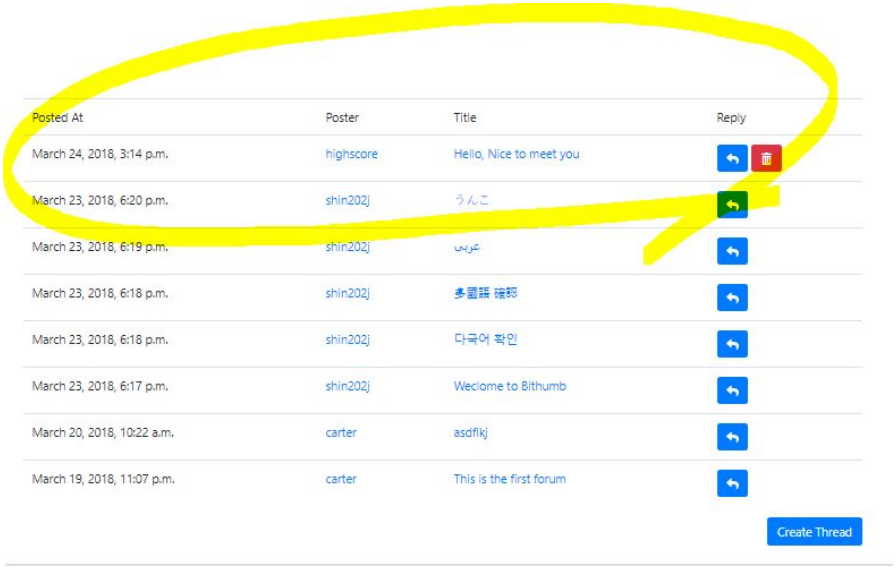
iii. **Forum functional test**



As the same user, he/she can visit forum section and “create thread” button will be shown as above (only for registered users).



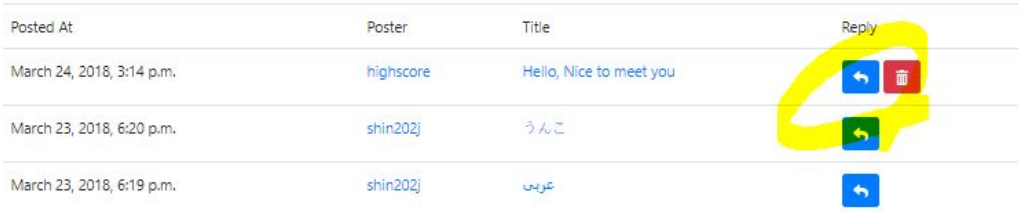
If user clicks create thread button, page for forum creation will be show as above. User can press post button after filling the title and description text box.



Posted At	Poster	Title	Reply
March 24, 2018, 3:14 p.m.	highscore	Hello, Nice to meet you	
March 23, 2018, 6:20 p.m.	shin202j	うんこ	
March 23, 2018, 6:19 p.m.	shin202j	عربى	
March 23, 2018, 6:18 p.m.	shin202j	多國語 確認	
March 23, 2018, 6:18 p.m.	shin202j	다국어 확인	
March 23, 2018, 6:17 p.m.	shin202j	Weclome to Bithumb	
March 20, 2018, 10:22 a.m.	carter	asdfikj	
March 19, 2018, 11:07 p.m.	carter	This is the first forum	

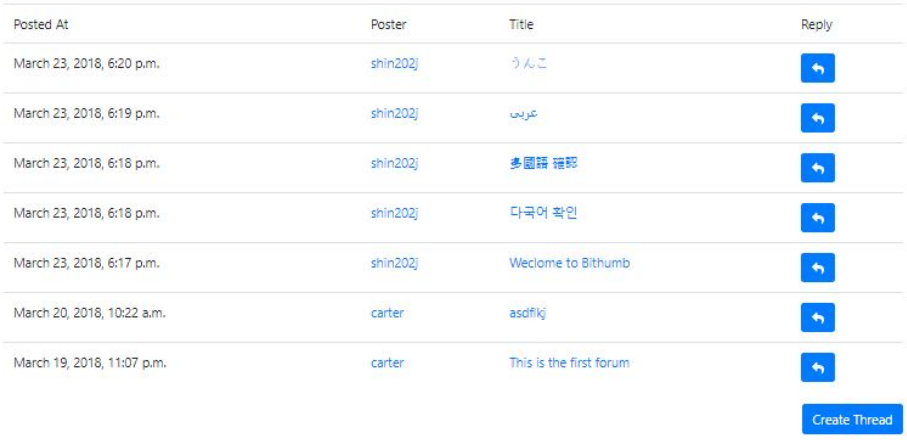
Create Thread

New thread is on thread list now.



Posted At	Poster	Title	Reply
March 24, 2018, 3:14 p.m.	highscore	Hello, Nice to meet you	
March 23, 2018, 6:20 p.m.	shin202j	うんこ	
March 23, 2018, 6:19 p.m.	shin202j	عربى	

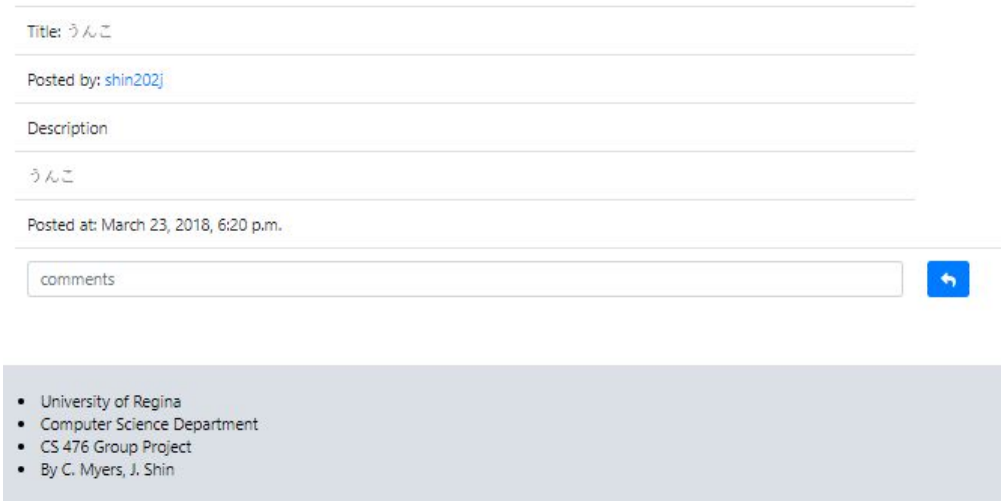
User can also reply to his/her own thread and will also see the “delete” button next to the thread that the user owns.



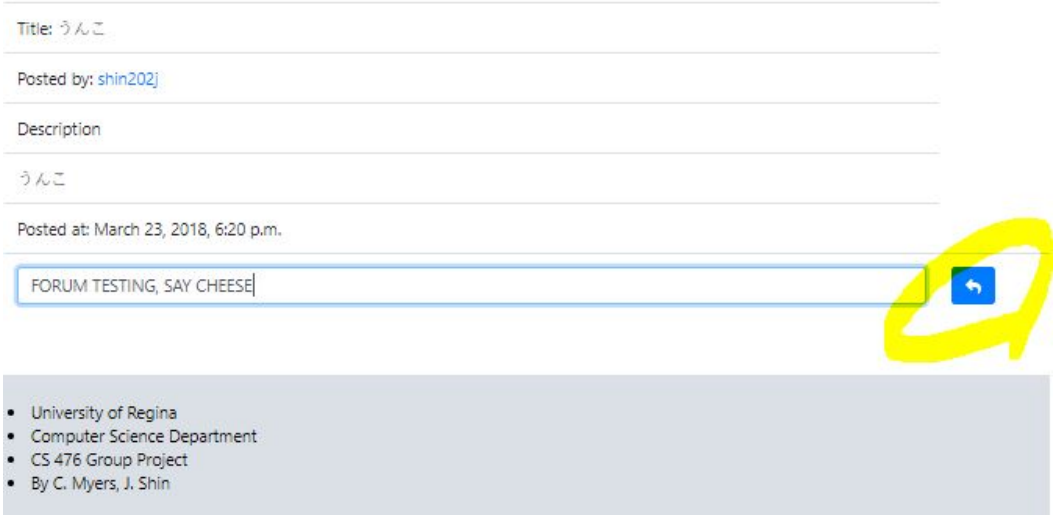
Posted At	Poster	Title	Reply
March 23, 2018, 6:20 p.m.	shin202j	うんこ	
March 23, 2018, 6:19 p.m.	shin202j	عربى	
March 23, 2018, 6:18 p.m.	shin202j	多國語 確認	
March 23, 2018, 6:18 p.m.	shin202j	다국어 확인	
March 23, 2018, 6:17 p.m.	shin202j	Weclome to Bithumb	
March 20, 2018, 10:22 a.m.	carter	asdfikj	
March 19, 2018, 11:07 p.m.	carter	This is the first forum	

Create Thread

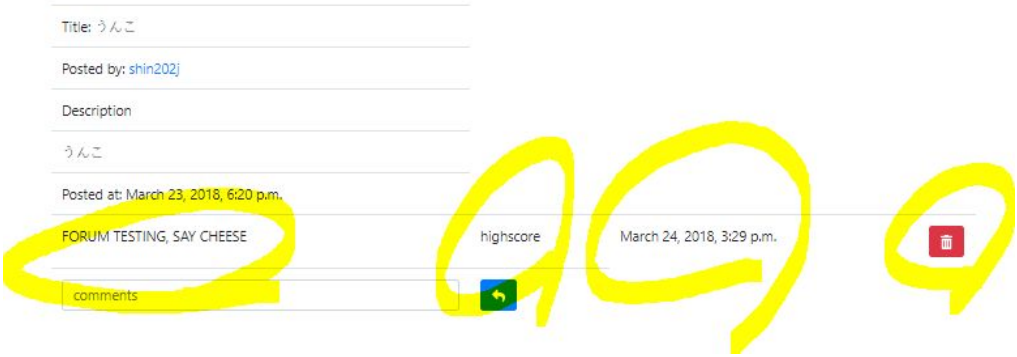
Thread has been deleted as above -- screenshot after clicking delete button.



After clicking reply button, user will see above “comment” page so the user can reply on the thread.

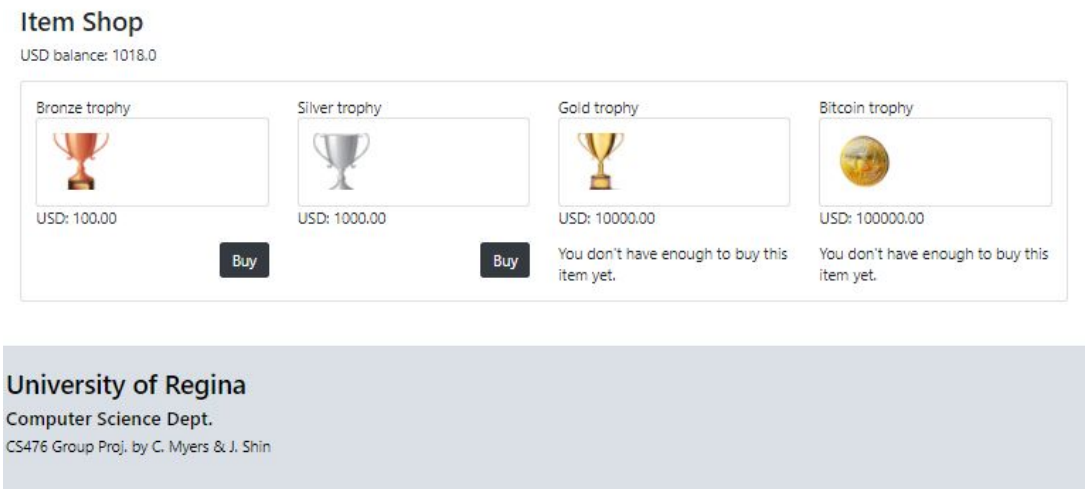


Fill the comment box then clicking replay button.

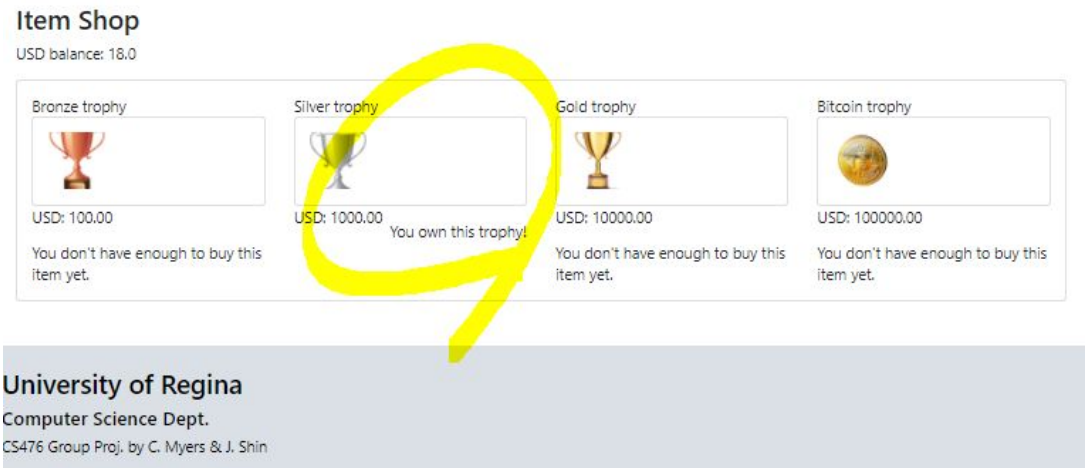


Then, comment will be up below the original thread, and also user name who wrote the comment, timestamp, and “delete” button. This delete button is also shown to only the user it belongs to.

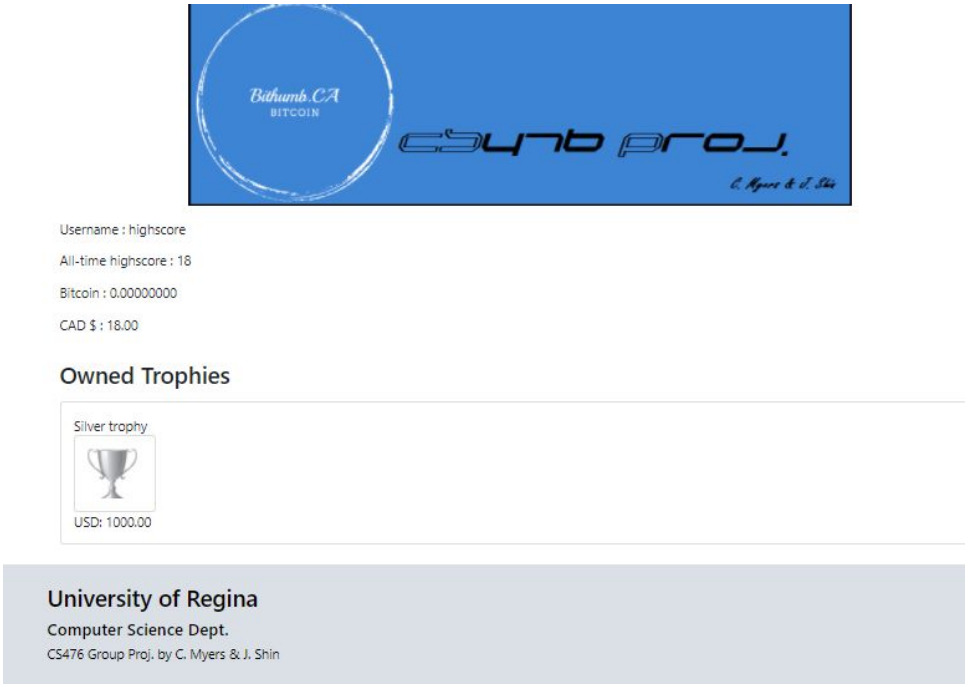
iv. **Item Shop**



User's total score (in USD) will allow the user to buy a items at item shop. As above screenshot, user owns 1018.0 USD, and this makes the bronze and silver trophies available to purchase.



After the user clicks buy button (in this case, on silver trophy). Now user owns Silver trophy and others are not available to buy because of the lack of total score the user has.

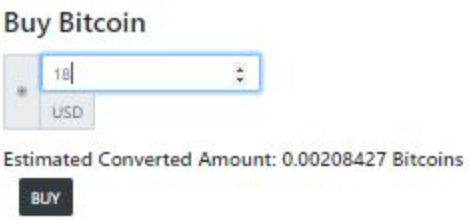


User can check items status at user profile page

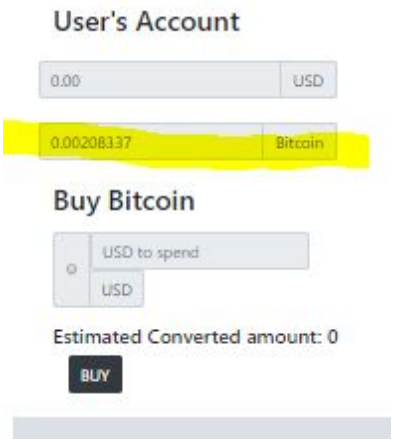
v. **Investment**

Logged in user will be available to see above investment page. It shows recent real-world bitcoin value changing “graph” and “Bitcoin Exchange rate” which are provided by server. The server collects this data with *bitcoin_price_api*, and then creates the graph using the Python library matplotlib. Update frequency details are at Ch.4 b

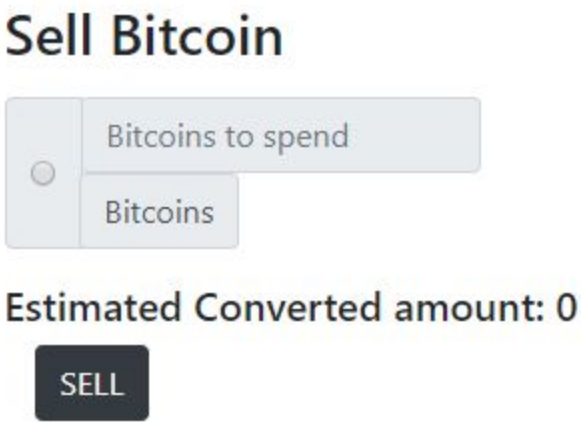
User’s Account section displays user’s current amount of scores(USD). And also displays Bitcoin amount user owns.



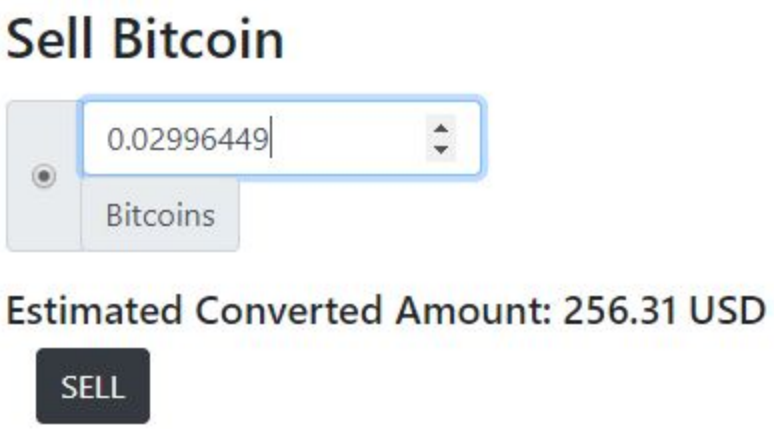
User need to check the radio box to enable the text box. Above case is buying bitcoin, below message informs converted amount dynamically as user types in.



Once user hits the buy button, bitcoin amount that user just bought goes into user's bitcoin account.



To sell Bitcoins user has, user can enable the radio box at Sell Bitcoin section.



User can check the current amount of bitcoins and it will be same procedure as buying bitcoin.

b. Robustness Test

i. Unicode Testing on forum

Title

일본의 어용우익잡지의 클래스

Description

13. 김정은의 비명이 들린다.

참고로 이 잡지의 아마존 평점은 별4개 이상(5개 만점)
http://piks.nl/dBK

Post

University of Regina
Computer Science Dept.
CS476 Group Proj. by C. Myers & J. Shin

Charset: UTF-8

Title: 일본의 어용우익잡지의 클래스

Posted by: shin202j

Description

월간 Hanada 2018년 5월호 - 840엔 1. 재무성의 공문서 위조와 아사히의 덫 :아사히 신문이 아베 총리가 지인에게 땅을 헐값으로 매각한 재무성의 문서에서 위조가 드러났다고 아사히가 보도함. 2. "공소장"을 바꾼 아사히신문 3. 아사히 신문은 무조건 폐간해야 한다 :글쓴이인 "하쿠타 나오키(百田尚樹)"는 일본의 대표적 우익작가. 4. 재무성 문서의 바른 이해법 5. 가케학원의 수의학부는 일본 제일이다! :가케 학원은 아베의 친구가 운영하는 교육재단으로, 불법 토지인허가, 학교 인허가, 토지 매입 등의 의혹을 받고 있음. 6. 아베 아키에 부인 비판은 현대의 마녀사냥 7. 위안부 허위보도에 대한 반성 없음 :아사히는 위안부문제를 최초로 보도한 신문임. 취재원의 취재에 문제가 있었다고 사과는 했으나 위안부를 '허위'라고 한 적은 없음. 8. 가고이케 부부를 이용하는 반아베 세력 ;가고이케는 아베 총리 부인이 뒤를 봐줘서 '아베 신조 기념 초등학교'를 지으려다 사기죄로 구치소 수감중인 인물. 9. 아베가 그만두면 누가 그 뒤를 이을 수 있나 10. 국가의 위기란 새벽이다 11. 독재자 시진핑의 미소에 속지 마라 12. 북한 위기를 돌파할 수 있는 것은 아베 총리뿐이다 13. 김정은의 비명이 들린다. 참고로 이 잡지의 아마존 평점은 별4개 이상(5개 만점) http://piks.nl/dBK

Posted at: March 24, 2018, 8:46 p.m.

comments

Successfully posted

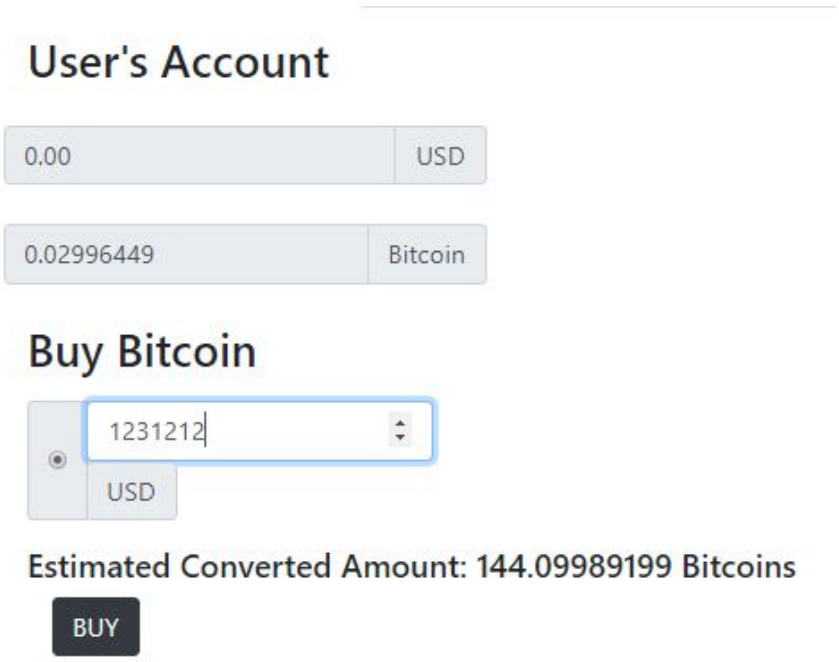
March 24, 2018, 8:46 p.m.	shin202j	일본의 어용우익잡지의 클래스
March 24, 2018, 3:22 p.m.	carter	next page
March 24, 2018, 3:22 p.m.	carter	hello
March 24, 2018, 3:21 p.m.	carter	does pagination work here
March 24, 2018, 3:20 p.m.	carter	Here is a report
March 23, 2018, 6:20 p.m.	shin202j	うんこ
March 23, 2018, 6:19 p.m.	shin202j	عربى
March 23, 2018, 6:18 p.m.	shin202j	多國語 確認
March 23, 2018, 6:18 p.m.	shin202j	다국어 확인
March 23, 2018, 6:17 p.m.	shin202j	Weclome to Bithumb

Create Thread

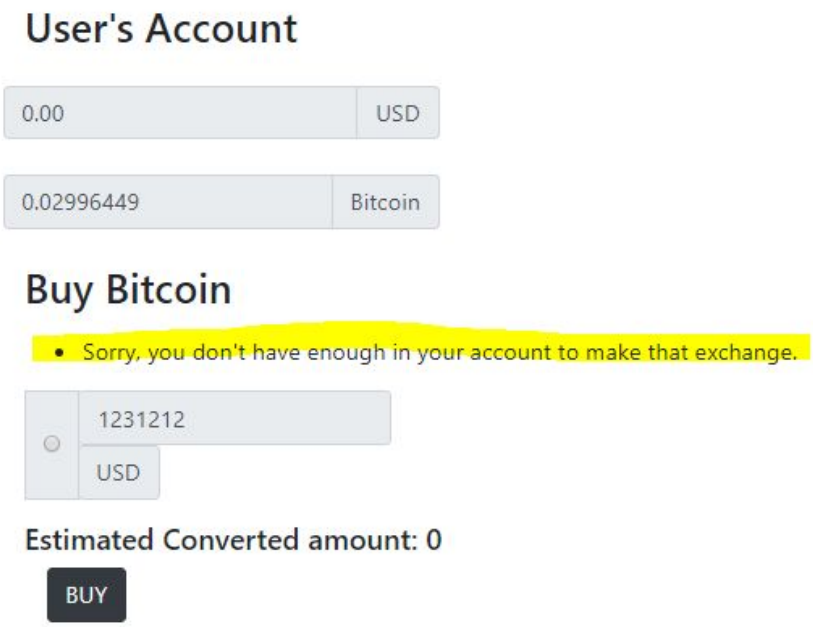
1 2 Next

Thread list with multi-language inputs

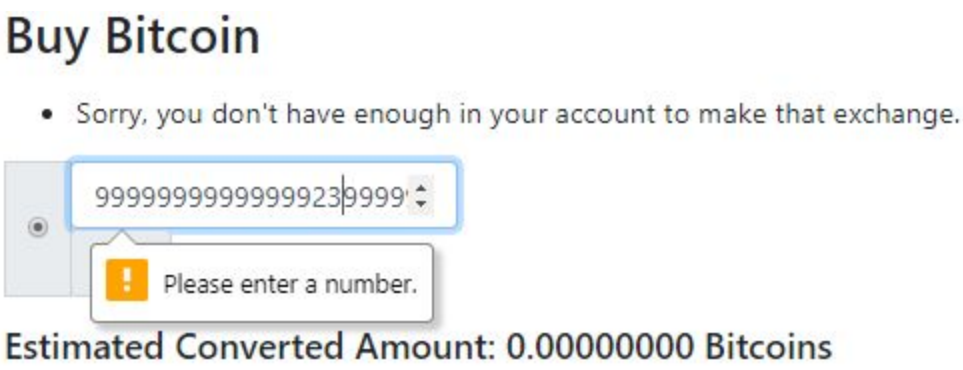
ii. **Investment page vulnerability test**



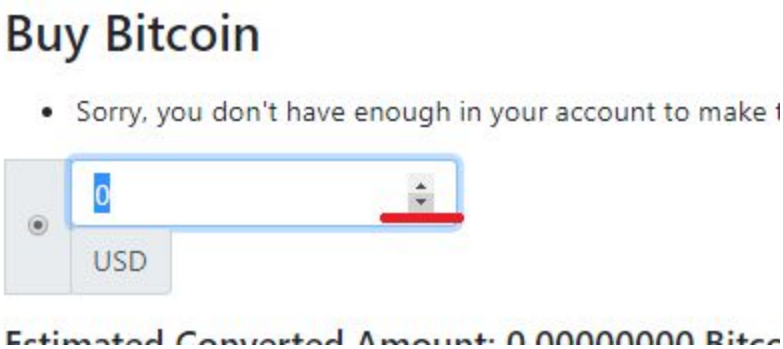
User tries to buy bitcoin with higher USD than user has



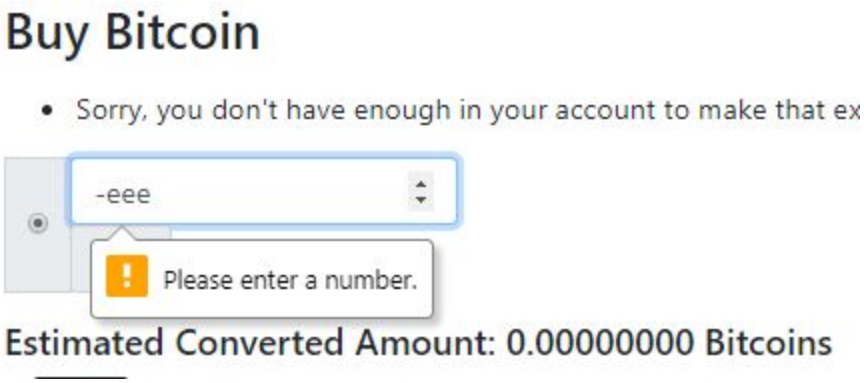
An error message informs the user.



Inform with error message when user input is too high



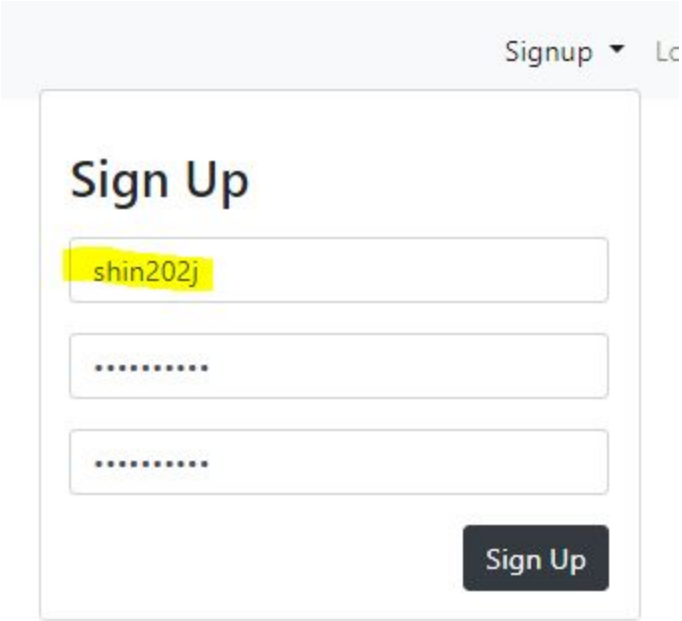
User also can input the amount with small scroll button in text box.
This also indicate 0 is the minimum value use can input.



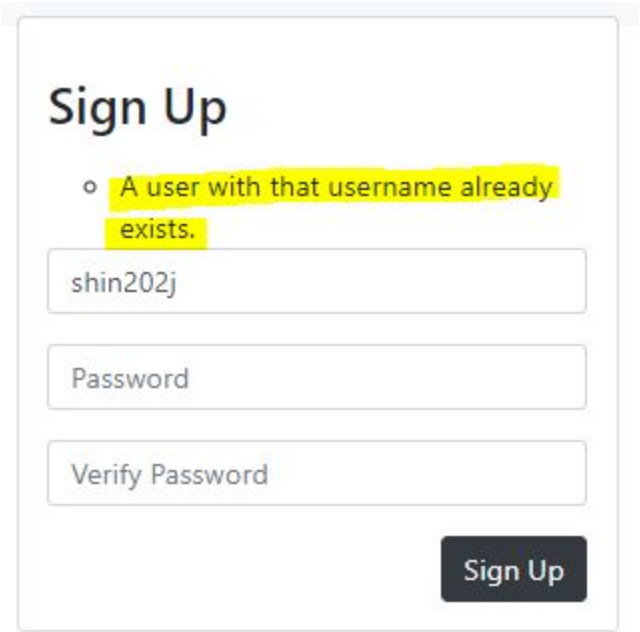
Only numbers can go through

Also, this robustness features are applied on “sell bitcoin” section.

iii. **User Account Vulnerability Test**

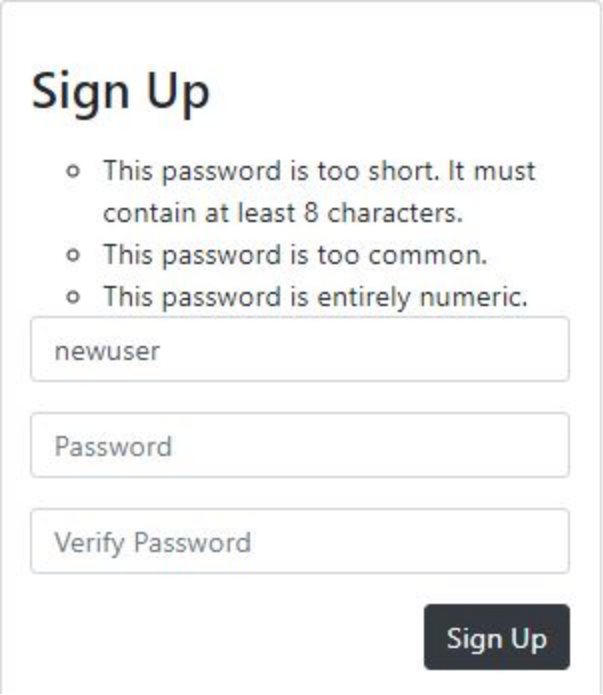


Creating account with username that already exists.

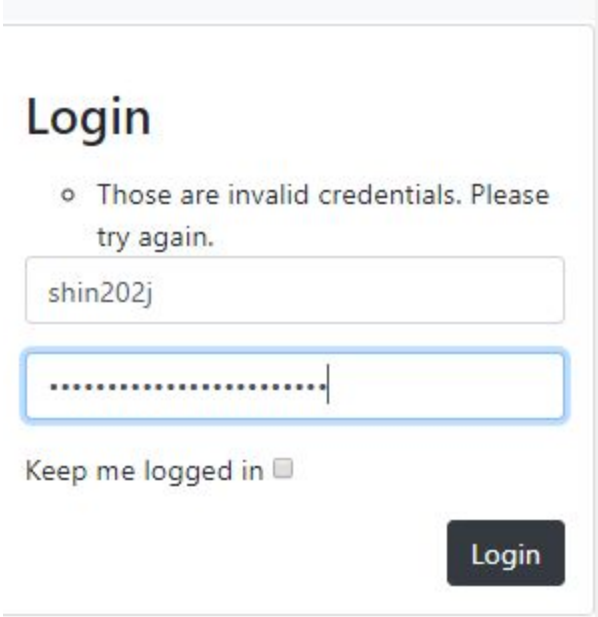


Inform the user with error message.

Creating user account with vulnerable password

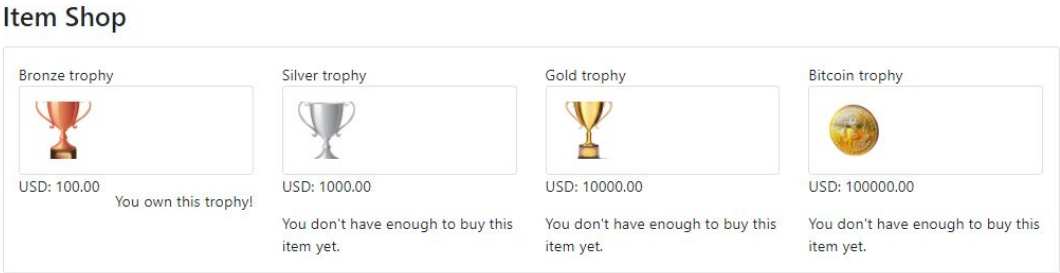


Inform the user with error message as well.



Wrong password input

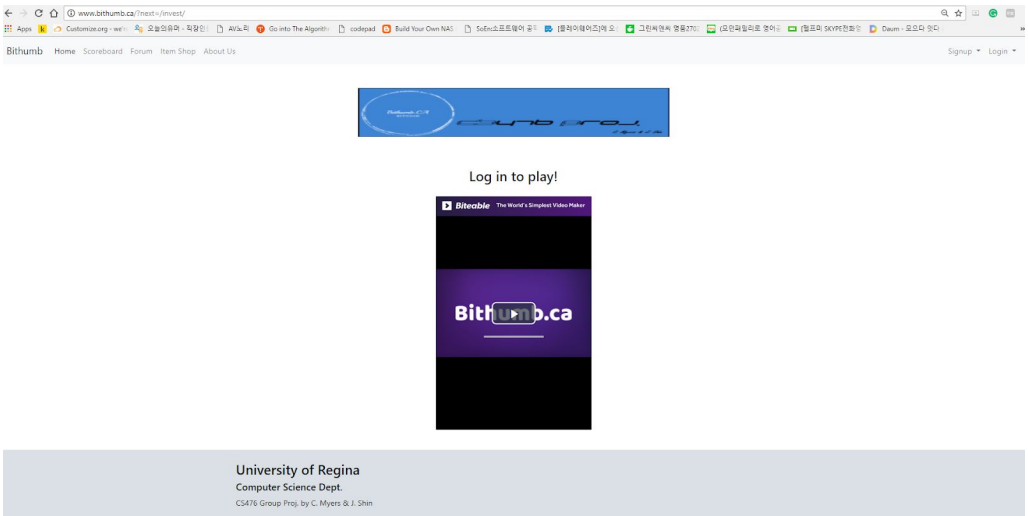
iv. **Item Shop Robustness**



Users are not allowed to have input features unless there is something that satisfies the condition. We decided not to give users a degree of freedom in this item shop area so we can prevent the any robustness problem occur.

v. Restrictions for Guest Users

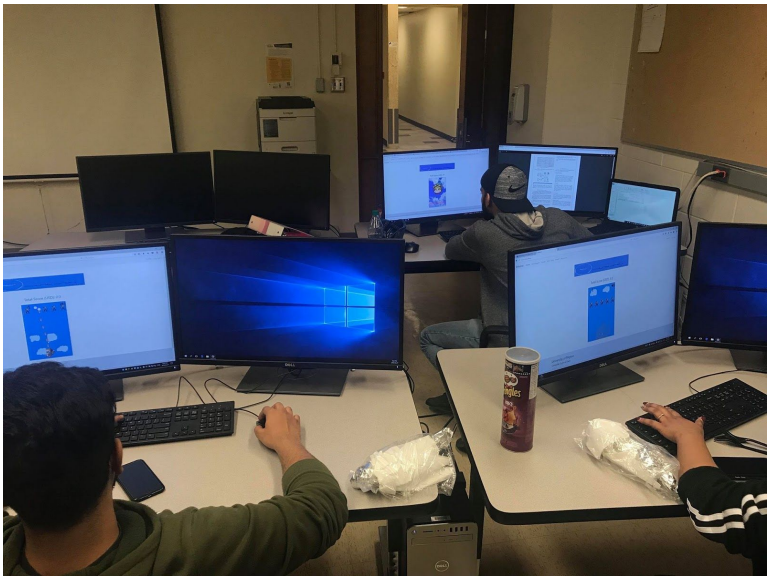
Guest user has strong restrictions. Guest users view will be controlled by server. In above picture.
guest user cannot see the investment page.



Guest user tried URL injection
www.bithumb.ca/invest
And it automatically redirects to main page.

Another URL injection that looking for account page.
Since Django provides pages dynamically, there is no chance that guest user access to profile pages that doesn't exist technically.

vi. Multi Users testing



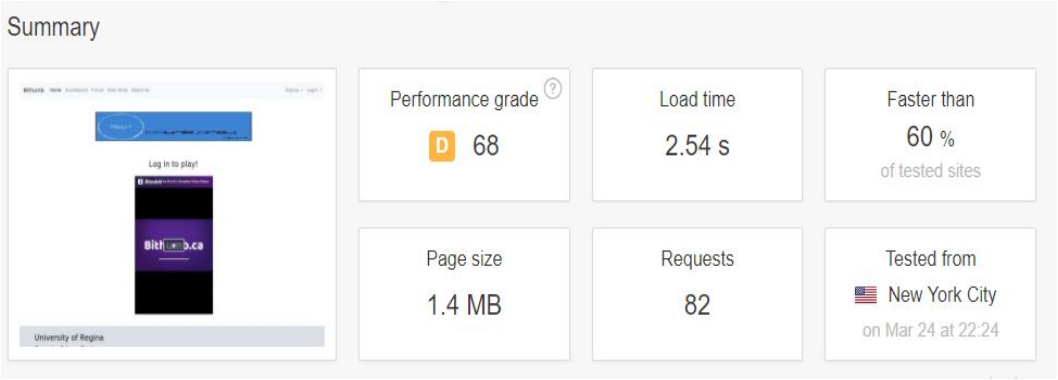
Testing ID 1 : hamza
Testing ID 2 : snehilmak
Testing ID 3 : subahturna

Testing dynamic allocated sessions for each users at same time.

c. Time Efficiency Test

Testing Tools	**Pingdom** (tools.pingdom.com)
	IoTcube (iotcube.korea.ac.kr)

i. **Main page with opening video clip**



Main page achieved a grade of D(68) and there are reasons as below.

Performance insights

GRADE		SUGGESTION
F	0	Minimize redirects
F	30	Leverage browser caching
D	69	Specify a Vary: Accept-Encoding header
C	76	Minimize request size
C	76	Specify a cache validator
A	92	Remove query strings from static resources
A	98	Serve static content from a cookieless domain
A	100	Avoid bad requests

Response codes

RESPONSE CODE	
200	OK
204	No Content
206	Partial Content
302	Moved Temporarily
307	Temporary Redirect

Content size by content type

CONTENT TYPE	PERCENT	SIZE
Script	52.3 %	763.33 KB
Other	31.6 %	462.03 KB
Image	8.7 %	126.87 KB
CSS	4.5 %	65.65 KB
HTML	2.9 %	41.64 KB
Plain text	0.0 %	679 bytes
Total	100.00 %	1.43 MB

Requests by content type

CONTENT TYPE	PERCENT	REQUESTS
Script	40.4 %	23
Image	35.1 %	20
HTML	7.0 %	4
Other	7.0 %	4
Plain text	7.0 %	4
CSS	3.5 %	2
Total	100.00 %	57

Content size by domain

DOMAIN	PERCENT	SIZE
cdn.biteable.com	45.1 %	658.01 KB
js.intercomcdn.com	29.1 %	424.48 KB
cdn.segment.com	4.7 %	68.56 KB
www.bithumb.ca	3.9 %	57.00 KB
connect.facebook.net	3.2 %	46.06 KB
other	14.1 %	206.07 KB
Total	100.00 %	1.43 MB

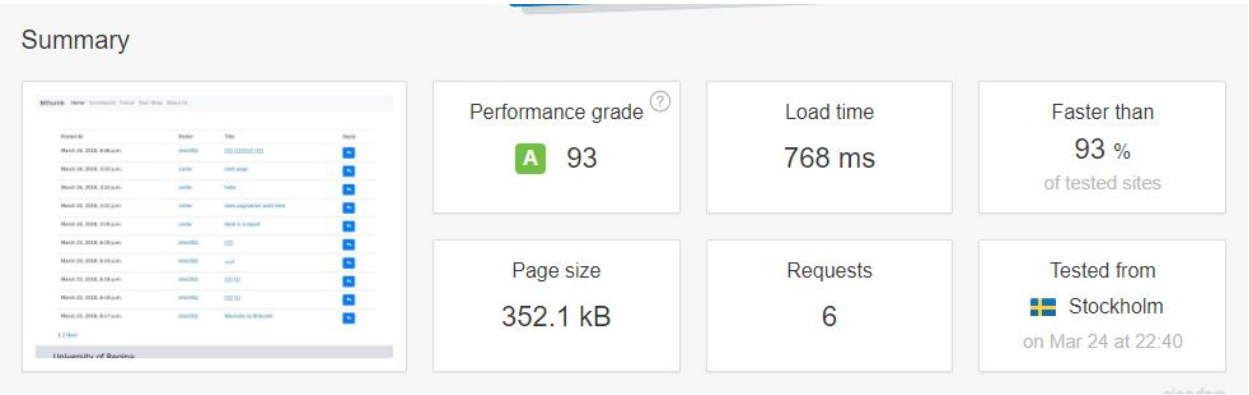
Requests by domain

DOMAIN	PERCENT	REQUESTS
cdn.biteable.com	8.8 %	5
www.bithumb.ca	7.0 %	4
www.facebook.com	7.0 %	4
s.adroll.com	5.3 %	3
connect.facebook.net	5.3 %	3
other	66.7 %	38
Total	100.00 %	57

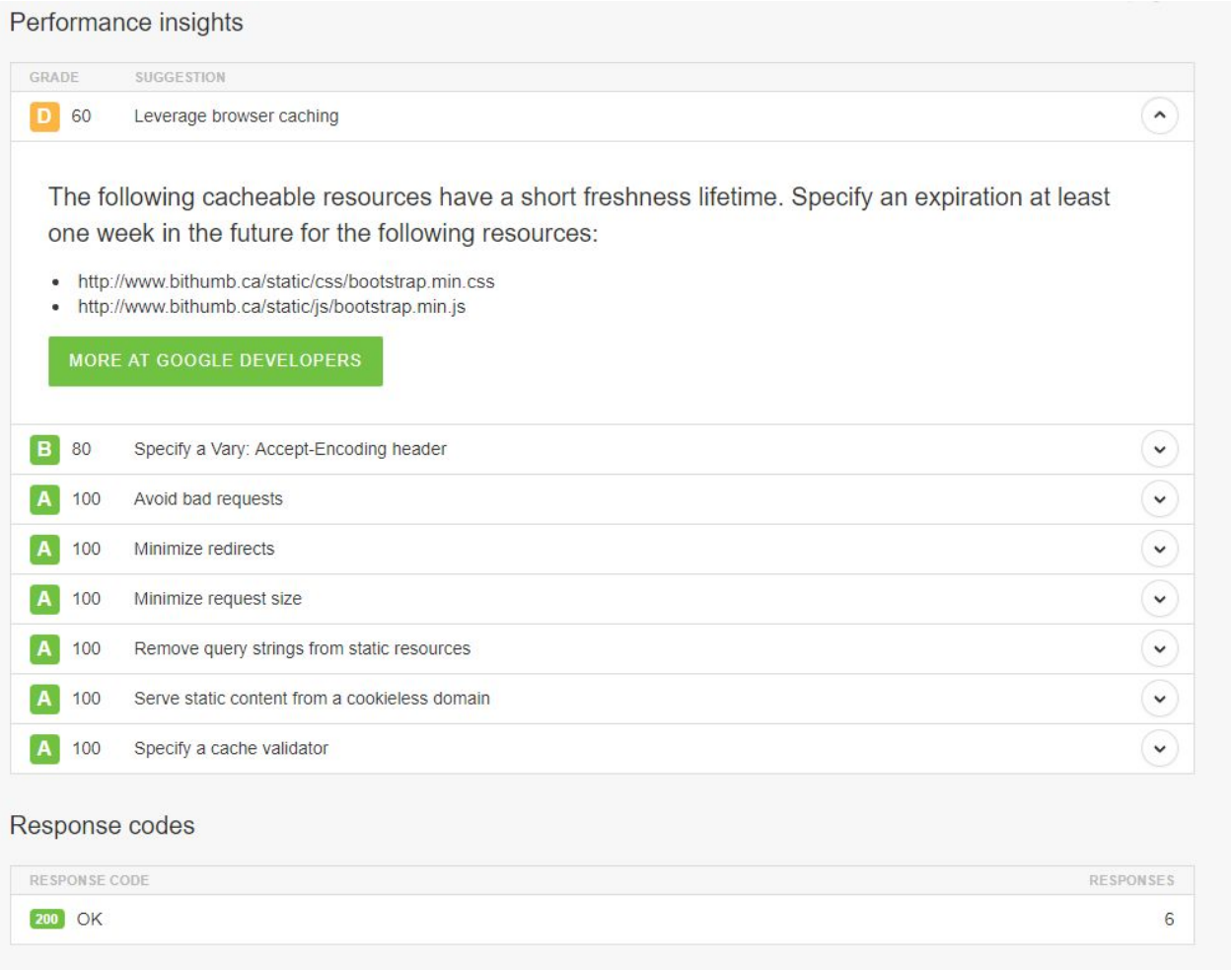
Cdn.biteable.com is a movie making application that we used for opening video. Loading the video requires the most resources. Moreover, there is a disadvantage of using free trial edition which causing advertisement redirections and it connects to FaceBook and Google.

Therefore, it is reasonable amount of loading time and others are frameworks. To resolve this mediocre mark, we should either use our own video stream or delete the opening video altogether.

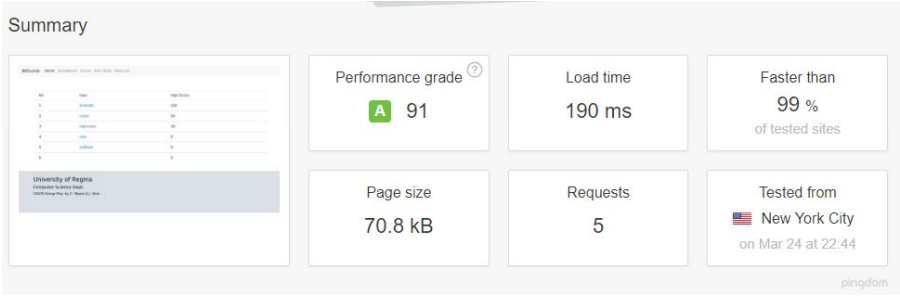
ii. Forum

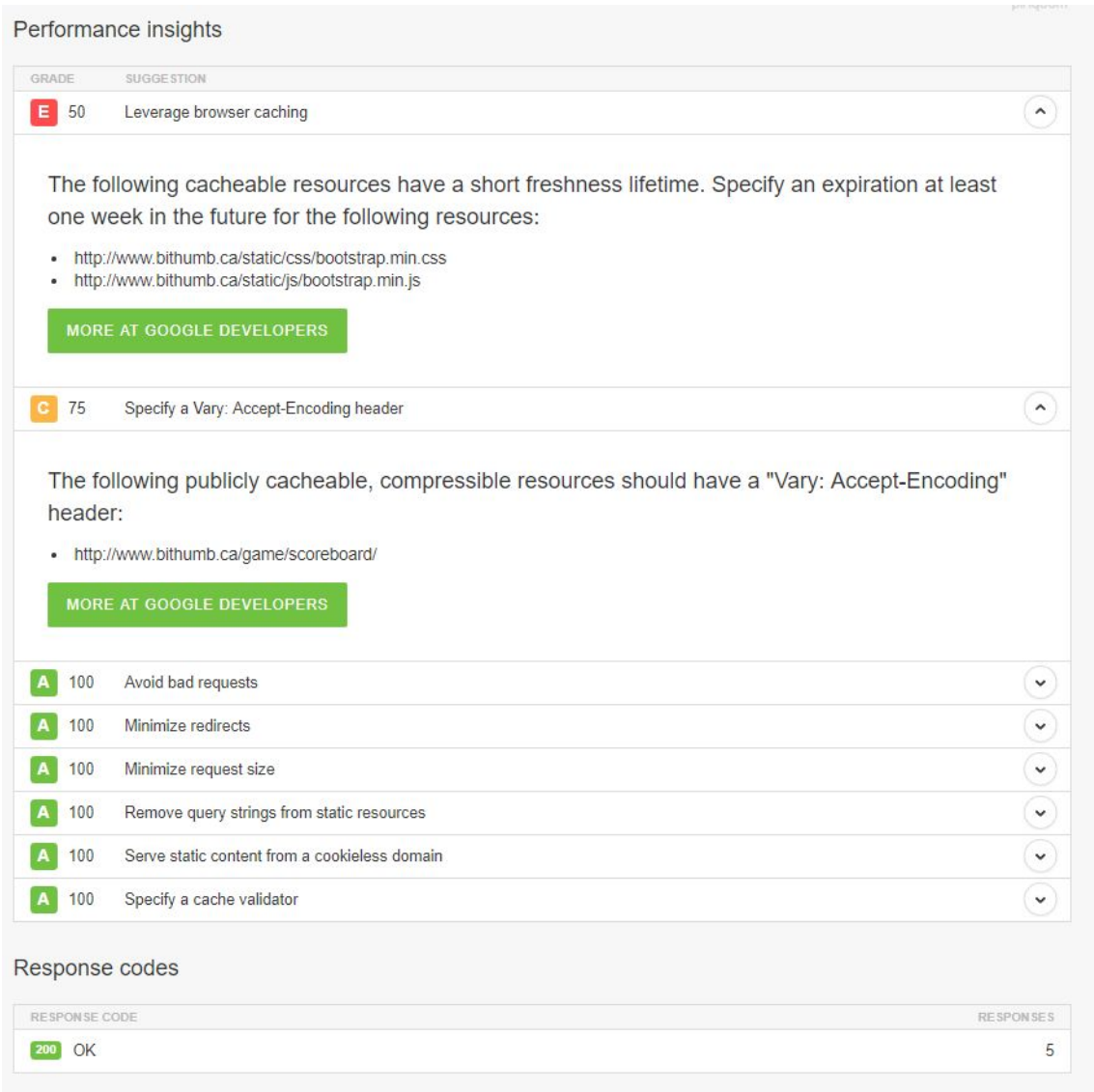


Forum's speed of service is reasonably fast enough.



iii. Score Board

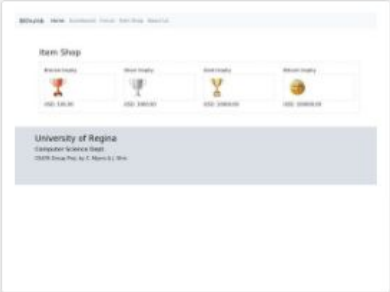




Our scoreboard is also functioning fine. Bootstrap, the framework, outsourcing would not be a matter.

iv. Item Shop

Summary



Performance grade	Load time	Faster than
B 89	200 ms	99 % of tested sites

Page size	Requests	Tested from
237.7 kB	9	New York City on Mar 24 at 22:56

Performance insights

GRADE	SUGGESTION
F 25	Leverage browser caching

The following cacheable resources have a short freshness lifetime. Specify an expiration at least one week in the future for the following resources:

- http://www.bithumb.ca/static/css/bootstrap.min.css
- http://www.bithumb.ca/static/js/bootstrap.min.js
- http://www.bithumb.ca/static/media/collectibles/bitcoin.png.200x200_q85_crop.png
- http://www.bithumb.ca/static/media/collectibles/bronze.png.200x200_q85_crop.png
- http://www.bithumb.ca/static/media/collectibles/gold.png.200x200_q85_crop.png
- http://www.bithumb.ca/static/media/collectibles/silver.png.200x200_q85_crop.png

MORE AT GOOGLE DEVELOPERS

B	87	Specify a Vary: Accept-Encoding header
A	100	Avoid bad requests
A	100	Minimize redirects
A	100	Minimize request size
A	100	Remove query strings from static resources
A	100	Serve static content from a cookieless domain
A	100	Specify a cache validator

Response codes

RESPONSE CODE	RESPONSES
200 OK	9

v. About Us

Summary

Performance grade	Load time	Faster than
A 91	141 ms	100 % of tested sites

Page size	Requests	Tested from
71.1 kB	5	New York City on Mar 24 at 23:02

pingdom

Performance insights

GRADE	SUGGESTION
E 50	Leverage browser caching

The following cacheable resources have a short freshness lifetime. Specify an expiration at least one week in the future for the following resources:

- http://www.bithumb.ca/static/css/bootstrap.min.css
- http://www.bithumb.ca/static/js/bootstrap.min.js

MORE AT GOOGLE DEVELOPERS

C 75	Specify a Vary: Accept-Encoding header

The following publicly cacheable, compressible resources should have a "Vary: Accept-Encoding" header:

- http://www.bithumb.ca/about/

MORE AT GOOGLE DEVELOPERS

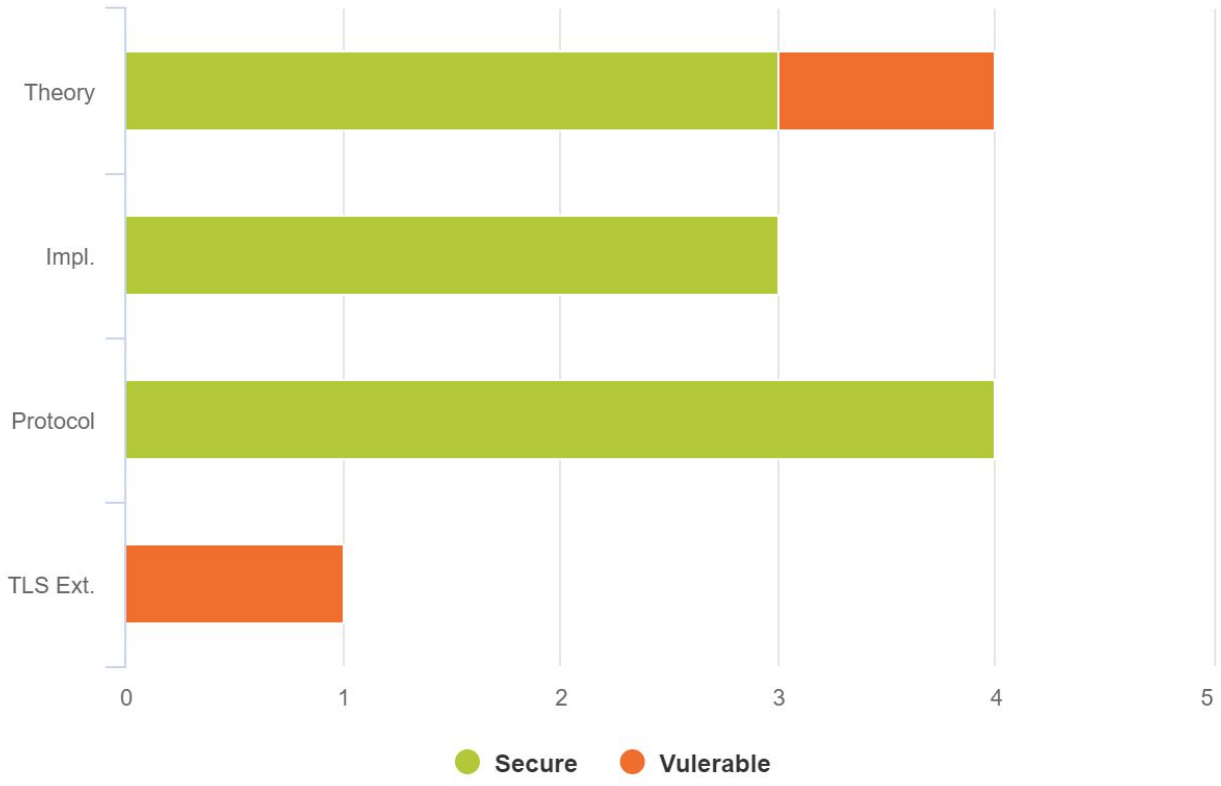
A 100	Avoid bad requests
A 100	Minimize redirects
A 100	Minimize request size
A 100	Remove query strings from static resources
A 100	Serve static content from a cookieless domain
A 100	Specify a cache validator

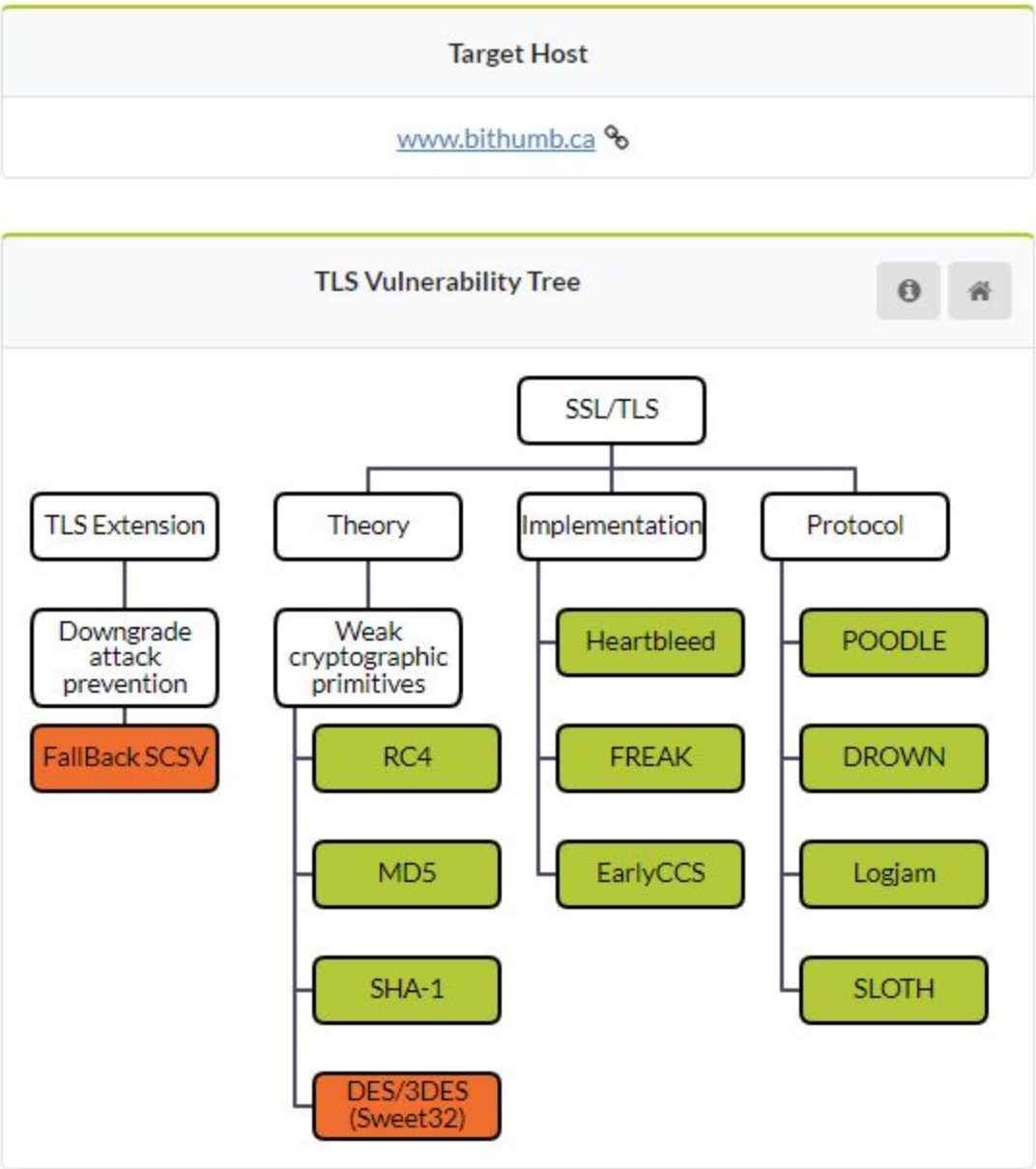
Response codes

RESPONSE CODE	RESPONSES
200 OK	5

vi. TLS vulnerability Testing (Transport Layer Security)

# Test Vulnerability	# Vulnerabilities Detected	# Passed Test
12	2	10





Certificate Information

- **Authority_key_id**
  90af6a3a945a0bd890ea125673df43b43a28dae7
- **Basic_constraints**
  -1
- **Certificate_policies**
  Null

- **Digital_signature**
  true
- **Dns_names**
  [[2, *.pythonanywhere.com], [2, pythonanywhere.com]]
- **Extened_key_usage**
  [1.3.6.1.5.5.7.3.1, 1.3.6.1.5.5.7.3.2]
- **Issuer**
  COMODO RSA Domain Validation Secure Server CA
- **Issuer_dn**
  CN=COMODO RSA Domain Validation Secure Server CA,O=COMODO CA Limited,L=Salford,ST=Greater Manchester,C=GB
- **Key_encipherment**
  true
- **Public_key_parameter**
  Sun RSA public key, 2048 bits modulus:
  290037806325292083027942767517265881974057741248657192708336862975454159238363673516281695825261603662433335751722689028953656902986317178400171015309186155103266578992917931737347182452506392695388087716418703919637947634092246235605463032815661857771752739948497422517367836425658898103656485492854421371895349920179095883022057350400258408058954925622607507378404202858150641147130215605858119788061570766509493070482695726428287269604741337615572473788441134307218885091018097537854884650255551635972703950523069742833595031000251397640973458791150048189951149999392667697470314219664680514099490796130263273991 61 public exponent: 65537
- **Publickey**
  RSA 2048 bits
- **Serial_number**
  105996461507074823452373206049615983247
- **Signature_algorithm**
  SHA256withRSA
- **Signature_algorithm_oid**
  1.2.840.113549.1.1.11
- **Subject**
  *.pythonanywhere.com
- **Subject_dn**
  null
- **Subject_key_id**
  Bb0d5558662dcfd8092c9240eaf632c3e2a5b596

- **targetServer**
  www.bithumb.ca
- **TLS support**
  true
- **Validity_end**
  Sun Sep 24 17:00:00 GMT-07:00 2017
- **Validity_start**
  Sun Dec 09 16:59:59 GMT-07:00 2018
- **Version**
  3

Fallback SCSV
Vulnerability status The TLS handshake is disconnected, The target server does not support Fallback SCSV
RFC7507
Detail Description Fallback SCSV can be employed to prevent unintended protocol downgrades between clients and servers that comply with RFC 7507 document by having the client indicate that the current connection attempt is merely a fallback and by having the server return a fatal alert if it detects an inappropriate fallback.

RFC7507[22]
Detail Description [23]

Mitigation	
IIS (Schannel)	Apache, NGINX (OpenSSL)
N/A	OpenSSL version 1.0.1j, 1.0.0o, 0.9.8zc automatically support Fallback SCSV. Please upgrade your OpenSSL version to latest version or three versions mentioned earlier.

[22] https://tools.ietf.org/html/rfc7507

[23] https://tools.ietf.org/html/rfc7507

DES/3DES(Sweet32)

Vulnerability status

The target server supports DES/3DES cipher suite

CVE-2016-2183

National Vulnerability Database (NVD)

Detail Description

The DES and Triple DES ciphers, as used in the TLS, SSH, and IPSec protocols and other protocols and products, have a birthday bound of approximately four billion blocks, which makes it easier for remote attackers to obtain cleartext data via a birthday attack against a long-duration encrypted session, as demonstrated by an HTTPS session using Triple DES in CBC mode, aka a "Sweet32" attack.

CVE-2016-2183[24]
National Vulnerability Database (NVD)[25]
Detail Description[26]

Mitigation

IIS (Schannel)	Apache, NGINX (OpenSSL)
To disable DES, 3DES on your Windows server, set the following registry key: [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Ciphers\DES 56] "Enabled"=DWORD:00000000 [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Ciphers\Triple DES 168] "Enabled"=DWORD:00000000	Remove all DES/3DES block ciphers from the ssl_ciphers list. The list of DES/3DES cipher is as follows: - EDH-RSA-DES-CBC-SHA - EDH-DSS-DES-CBC-SHA - ADH-DES-CBC-SHA - DES-CBC-SHA - EXP-EDH-RSA-DES-CBC-SHA - EXP-EDH-DSS-DES-CBC-SHA - EXP-ADH-DES-CBC-SHA - EXP-DES-CBC-SHA

Our server providing reliable hosting service.

[24] https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-2183

[25] https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-2183

[26] https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-2183

8. Contribution

a. Carter Myers

i. Back-end Implementation
    1. Construction
    2. Configuration
    3. Deployment
    4. back-end code (including all .py files)

ii. Database Implementation (MySQL)

iii. API management
    1. Bitcoin graph with matplotlib
    2. Bitcoin currency rate
    3. Game and DB interacts
    4. Pusher API

iv. Economic feasibility study

v. Activity Diagrams

vi. All design specification documents
    1. Logical software architecture
    2. Design patterns
    3. Class diagram

vii. Data table screenshots

viii. Component Diagram

ix. Revising the report

x. Preparing soft copies to submit

b. Junho Shin

i. Server(Pythonanywhere) & Domain(www.bithumb.ca) owns

ii. Bithumb the game
    1. files located under static/game

iii. HTML/CSS/JS front-end implementation

iv. Logo (main page & ingame)

v. Opening Video

vi. Software Qualities

vii. List of implemented classes/functions

viii. All acceptance testing documents
    1. Functional testing
    2. Robustness testing
    3. Time-efficiency

ix. Edit & formatting the report

x. Binding hard copies to submit

c. Shared (50/50)

i. Problem definition
ii. Functional Requirements list
iii. Use Case Diagram
iv. Deployment Diagram
v. All technical documentation
    1. List of programs
    2. List of reused algorithms
    3. List of software tools and environment

<End of the Report>
Thank you very much!
Sincerely, C. Myers & J. Shin