Project 1 Report

Ece Nur ŞEN - 150150104

Joshgun Rzabayli - 150160901

Main Objective

Main objective for this project is to create a user interface where a user can connect to the bitcoin network, listen to the blocks in the network, check the validity of the blocks, create a transaction and a wallet where the user can store the owned bitcoins.

Duration

Duration for this project is in total four weeks. The deadline is 15th of November, 2020.

Planning

Main objectives:

1. Communication with the bitcoin network
2. Creating keys
3. Storing the user account in database
4. Tracking and displaying the blocks
5. Checking the validity of blocks
6. Making transaction

Running a Node on Bitcoin Network

Blockchain consists of nodes communicating with one and other in peer-to-peer network structure. In order to be a part of the blockchain; to make transactions, display recent blocks or create keys, one must need to run a node on Bitcoin [1]. To obtain a bitcoin node, we have to run a node. We could have run a node by downloading from bitcoincore.org but instead we used Docker [2]. As a team, we are using two different operating systems: Windows and MacOS. Since we are using different operating systems hence two different environments, it was highly possible to get incompatible environment errors while developing our project together. In order not to face these kinds of problems, we used Docker . Instead of running our bitcoin nodes on our local computers, we run our bitcoin nodes on docker containers. Through docker-compose, we created an environment that we both can use without getting any incompatible environment errors.

Only difference between running a node through downloading bitcoincore and through docker and docker-compose is, docker enables us to create exactly the same environment to run a node on both of our computers.

Communication with Bitcoin Network (Testnet)

In order to communicate with our nodes, we used Bitcoin Core RPC [3]. RPC enables us to directly communicate with our node through HTTP Post requests. After directly communicating with our nodes, the node publishes our requests to network. Then, we get the answer accordingly through our node. One can either use RPC to communicate with nodes through terminal commands or HTTP requests. In order to accomplish our objectives, we used several RPC functions. For every objective, I will state the RPC functions that we used in the "Web Application Flow" part of the report.

Bitcoin has three different networks: regtest, testnet and mainnet. Firstly, we started testing our RPC requests in regtest. Regtest is a great beginner network for people like us that started to learn bitcoin recently. In regtest, there is only one node and it is the node that you run.

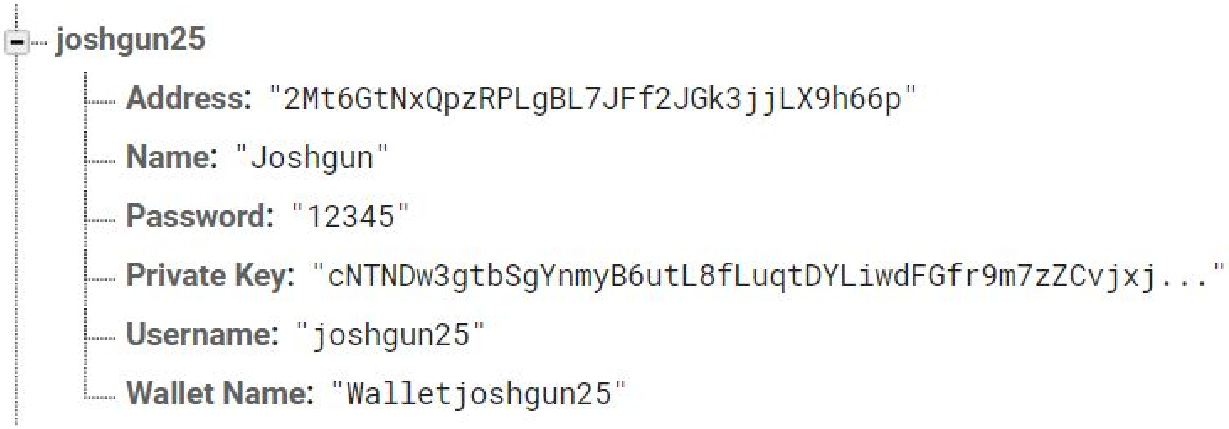
After regtest, we moved on to testnet. We used testnet3. Regtest was great but there was only one node in the network, there were no blocks to listen from others and we didn't have any coins. In testnet, we created several addresses to create transactions between one and other. We used coinfaucet to gain testcoins, since we don't know how to mine. We will return our testcoin, when we complete our project, in order for people to use it. In testnet, we tested all of our objectives and we completed all of them.

Finally, our plan was to move on to mainnet, where we can really be a part of the bitcoin network. Sadly, we couldn't manage to do that. Learning the bitcoin network and getting familiar with the network took so much of our times. After we completed all of our objectives on testnet, since the whole mainnet is 350GB, we couldn't have necessary time to download and test our project on mainnet.

Database

We used Google Firebase RealTime Database [4] to store the users' account information in the project. The Firebase Database is a good option to collaborate across diverse devices easily and it integrates with Firebase Authentication to provide strong user-based security. Moreover, this database is cloud based and no need for a server.

We made a table called "Users" in the database to keep the necessary data of every user such as name, wallet name, address, private key and etc. The "pyrebase" library[5] of Python is used for database operations.



```
joshgun25
    Address: "2Mt6GtNxQpzRPLgBL7JFf2JGk3jjLX9h66p"
    Name: "Joshgun"
    Password: "12345"
    Private Key: "cNTNDw3gtbSgYnmyB6utL8fLuqtDYLiwdFGfr9m7zZCvjxj..."
    Username: "joshgun25"
    Wallet Name: "Walletjoshgun25"
```

Figure 1: Example data from Database

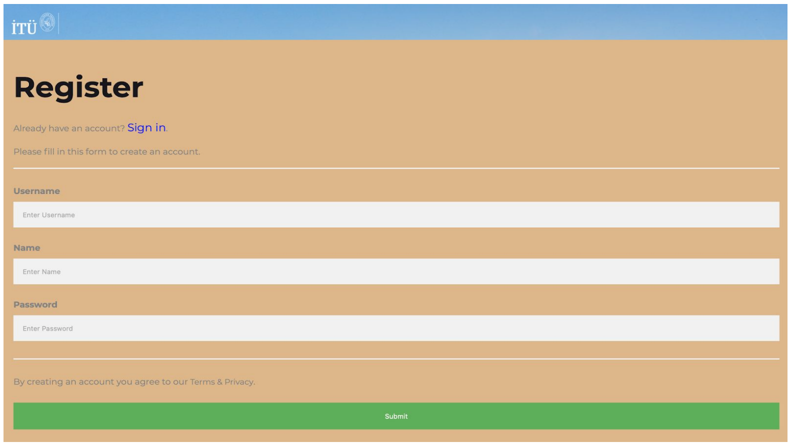
Graphical User Interface

The Python Flask [6] web framework is used for the GUI development. HTML and CSS are used for the interface of the web app. The whole project is made in a Python Flask environment which is easy to operate and Python has plenty of various functions.
The Bitcoin functions were written in separate Python files and integrated into Web application.

Web Application Flow

Registration Process

Firstly the user has to register to create a bitcoin wallet. When the user fill the register submit the register form; “createwallet”, “getnewaddress”, ”dumpprivkey” RPC methods are used in order to accomplish registration. After all the necessary information is created, the user data is stored in the Database. The registration page is shown in Figure 2.



Register

Already have an account? Sign in.

Please fill in this form to create an account.

By creating an account you agree to our Terms & Privacy.

Figure 2: Registration Page

Login Process

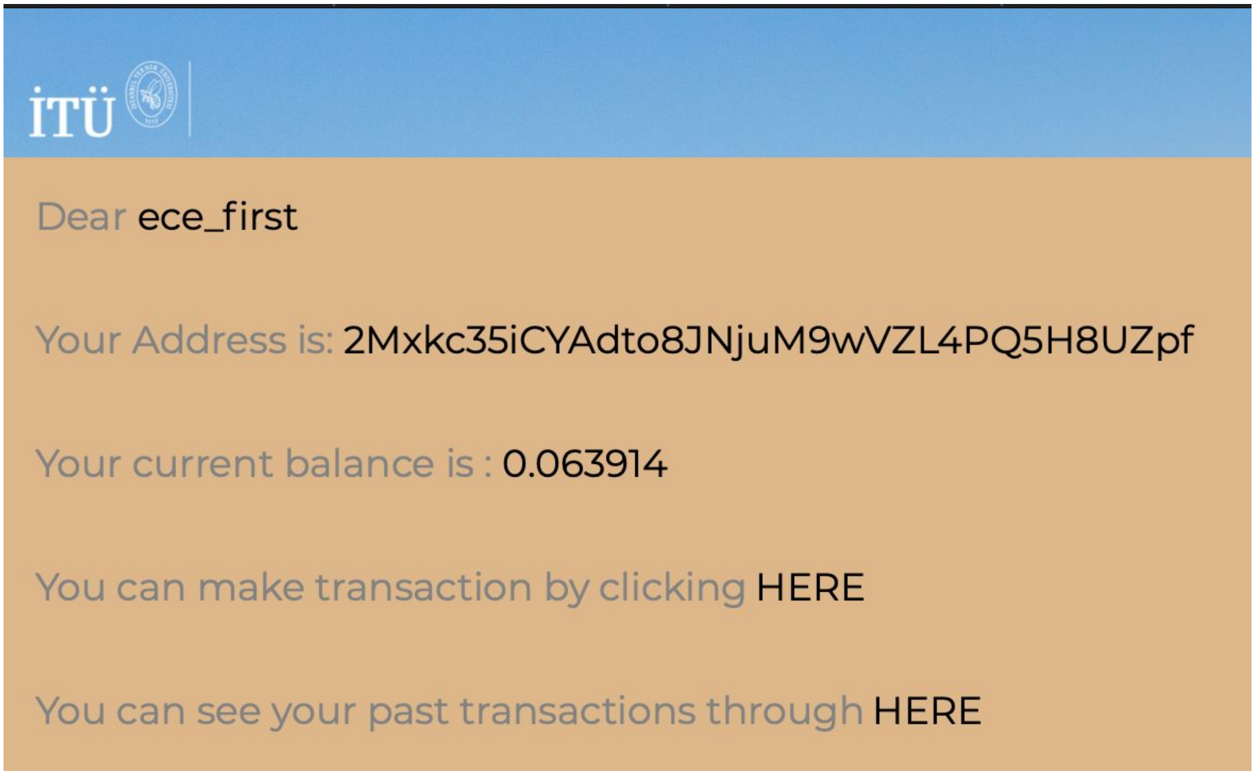
The user enters his/her username and password to login the Wallet App. The user's credentials are checked from the database and returns the result message in the backend. If the user exist, his/her bitcoin wallet is loaded by the "Load Wallet" RPC method.

Logout Process

If the user clicks the "Logout" button, all user information will removed from the session and the "Unload Wallet" RPC method will be called in the backend.

Wallet Information

Wallet information of the logged in user is displayed in the Wallet Information page as shown in Figure 3. The balance data of the user is obtained by "getbalance" RPC method.



Dear ece_first

Your Address is: 2Mxkc35iCYAdto8JNjuM9wVZL4PQ5H8UZpf

Your current balance is : 0.063914

You can make transaction by clicking HERE

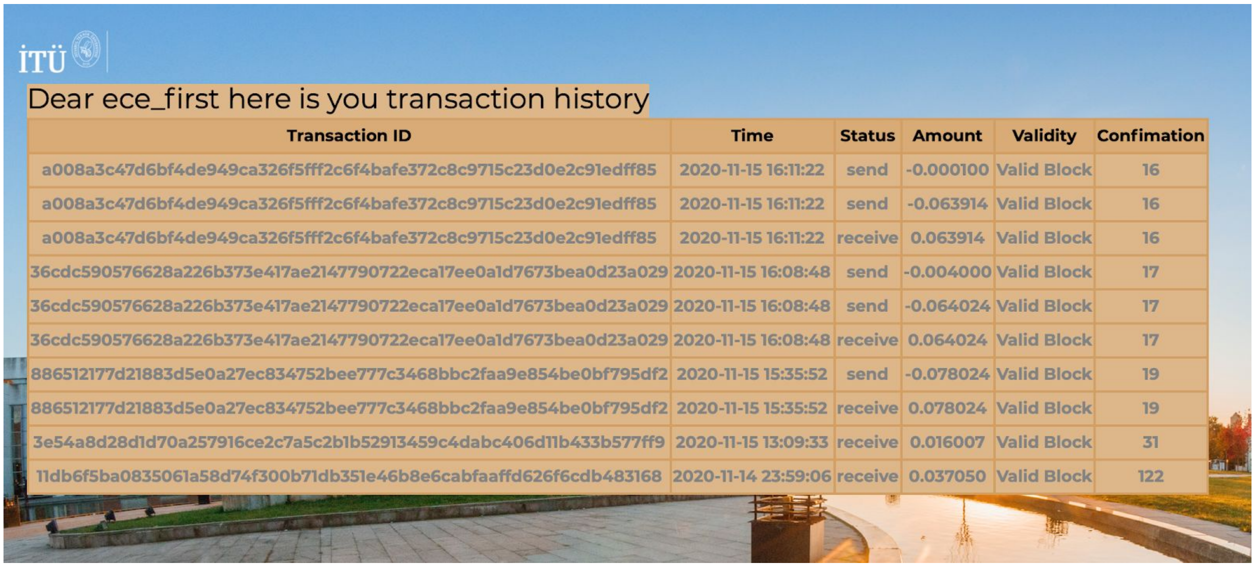
You can see your past transactions through HERE

Figure 3: Wallet Information Page

User Transactions

After login, the user can view his/her transaction history. In order to obtain transaction history, we used, "listtransactions" RPC method. Every transaction has validity status. If the block where the transaction is in, has 6 or more confirmation that block is considered valid because it is highly likely to end up in the consensus block.

In transaction history, users can also see their newly created transactions and check their validity. The transaction history page is shown in Figure 4.



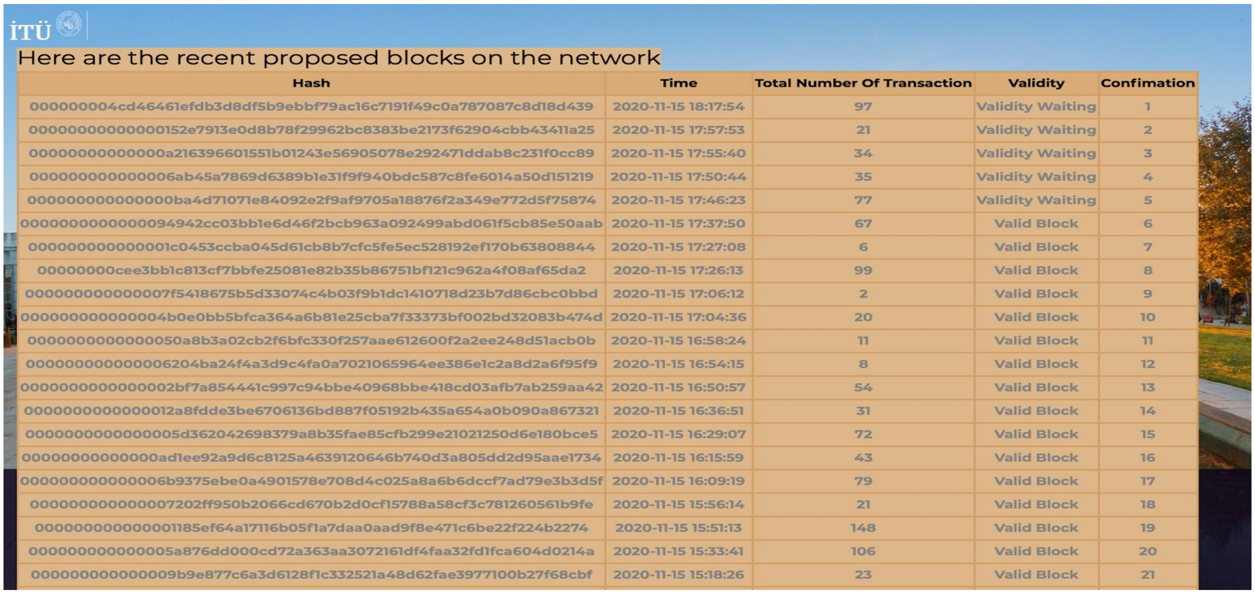
Dear ece_first here is you transaction history

Transaction ID	Time	Status	Amount	Validity	Confimation
a008a3c47d6bf4de949ca326f5fff2c6f4bafe372c8c9715c23d0e2c91edff85	2020-11-15 16:11:22	send	-0.000100	Valid Block	16
a008a3c47d6bf4de949ca326f5fff2c6f4bafe372c8c9715c23d0e2c91edff85	2020-11-15 16:11:22	send	-0.063914	Valid Block	16
a008a3c47d6bf4de949ca326f5fff2c6f4bafe372c8c9715c23d0e2c91edff85	2020-11-15 16:11:22	receive	0.063914	Valid Block	16
36cdc590576628a226b373e417ae2147790722eca17ee0a1d7673bea0d23a029	2020-11-15 16:08:48	send	-0.004000	Valid Block	17
36cdc590576628a226b373e417ae2147790722eca17ee0a1d7673bea0d23a029	2020-11-15 16:08:48	send	-0.064024	Valid Block	17
36cdc590576628a226b373e417ae2147790722eca17ee0a1d7673bea0d23a029	2020-11-15 16:08:48	receive	0.064024	Valid Block	17
886512177d21883d5e0a27ec834752bee777c3468bbc2faa9e854be0bf795df2	2020-11-15 15:35:52	send	-0.078024	Valid Block	19
886512177d21883d5e0a27ec834752bee777c3468bbc2faa9e854be0bf795df2	2020-11-15 15:35:52	receive	0.078024	Valid Block	19
3e54a8d28d1d70a257916ce2c7a5c2b1b52913459c4dabc406d11b433b577ff9	2020-11-15 13:09:33	receive	0.016007	Valid Block	31
11db6f5ba0835061a58d74f300b71db351e46b8e6cabfaaffd626f6cdb483168	2020-11-14 23:59:06	receive	0.037050	Valid Block	122

Figure 4: User Transaction History Page

Recent Proposed Blocks

After the Login, the user will be available to see the recent proposed block from the network in a table form as given in Figure 5.



Here are the recent proposed blocks on the network

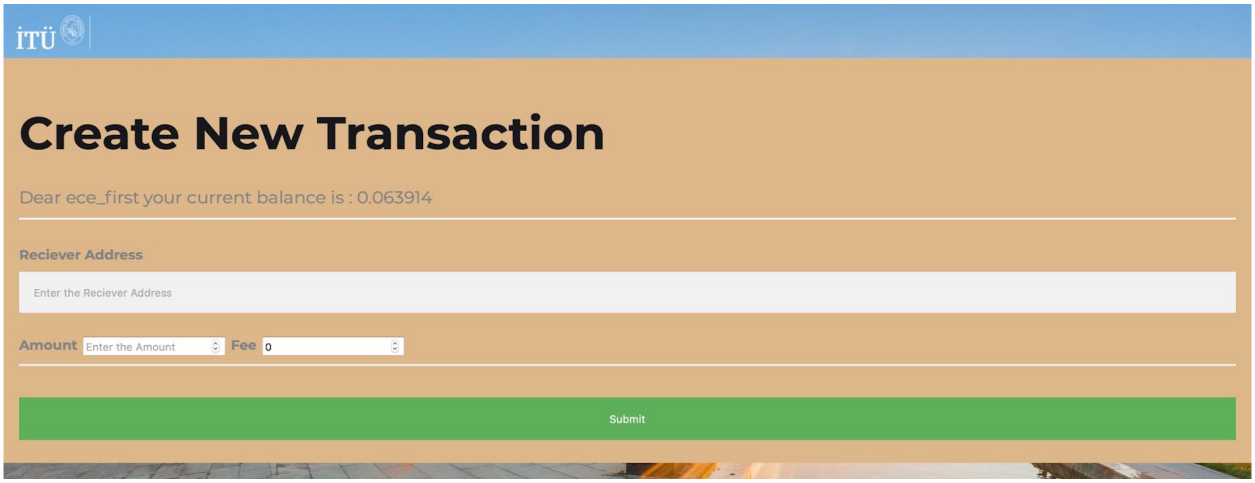
Hash	Time	Total Number Of Transaction	Validity	Confimation
000000004cd46461efdb3d8df5b9ebbf79ac16c7191f49c0a787087c8d18d439	2020-11-15 18:17:54	97	Validity Waiting	1
0000000000000152e7913e0d8b78f29962bc8383be2173f62904cbb43411a25	2020-11-15 17:57:53	21	Validity Waiting	2
00000000000000a216396601551b01243e56905078e292471ddab8c231f0cc89	2020-11-15 17:55:40	34	Validity Waiting	3
000000000000006ab45a7869d6389b1e31f9f940bdc587c8fe6014a50d151219	2020-11-15 17:50:44	35	Validity Waiting	4
00000000000000ba4d71071e84092e2f9af9705a18876f2a349e772d5f75874	2020-11-15 17:46:23	77	Validity Waiting	5
0000000000000094942cc03bb1e6d46f2bcb963a092499abd061f5cb85e50aab	2020-11-15 17:37:50	67	Valid Block	6
000000000000001c0453ccba045d61cb8b7cfc5fe5ec528192ef170b63808844	2020-11-15 17:27:08	6	Valid Block	7
00000000cee3bb1c813cf7bbfe25081e82b35b86751bf121c962a4f08af65da2	2020-11-15 17:26:13	99	Valid Block	8
000000000000007f5418675b5d33074c4b03f9b1dc1410718d23b7d86cbc0bbd	2020-11-15 17:06:12	2	Valid Block	9
000000000000004b0e0bb5bfca364a6b81e25cba7f33373bf002bd32083b474d	2020-11-15 17:04:36	20	Valid Block	10
0000000000000050a8b3a02cb2f6bfc330f257aae612600f2a2ee248d51acb0b	2020-11-15 16:58:24	11	Valid Block	11
000000000000006204ba24f4a3d9c4fa0a7021065964ee386e1c2a8d2a6f95f9	2020-11-15 16:54:15	8	Valid Block	12
000000000000002bf7a854441c997c94bbe40968bbe418cd03afb7ab259aa42	2020-11-15 16:50:57	54	Valid Block	13
0000000000000012a8fdde3be6706136bd887f05192b435a654a0b090a867321	2020-11-15 16:36:51	31	Valid Block	14
000000000000005d362042698379a8b35fae85cfb299e21021250d6e180bce5	2020-11-15 16:29:07	72	Valid Block	15
00000000000000ad1ee92a9d6c8125a4639120646b740d3a805dd2d95aae1734	2020-11-15 16:15:59	43	Valid Block	16
000000000000006b9375ebe0a4901578e708d4c025a8a6b6dccf7ad79e3b3d5f	2020-11-15 16:09:19	79	Valid Block	17
000000000000007202ff950b2066cd670b2d0cf15788a58cf3c781260561b9fe	2020-11-15 15:56:14	21	Valid Block	18
00000000000001185ef64a17116b05f1a7daa0aad9f8e471c6be22f224b2274	2020-11-15 15:51:13	148	Valid Block	19
000000000000005a876dd000cd72a363aa3072161df4faa32fd1fca604d0214a	2020-11-15 15:33:41	106	Valid Block	20
000000000000009b9e877c6a3d6128f1c332521a48d62fae3977100b27f68cbf	2020-11-15 15:18:26	23	Valid Block	21

Figure 5: Recent Proposed Blocks Page

In order to reach all recent blocks, first we obtain the most recent block's hash using the "bestblockhash" RPC method. Then by passing the hash we get block information using the "getblock" RPC method. This block information also contains the previous block's hash. After that, we repeat the same process using previous blocks' hashes until we reach our limit block number. At the current our block number is 100, hence we show the last 100 proposed blocks in the network. About the validity of the proposed blocks, we marked blocks as valid if they receive 6 or more confirmations.

Make Transaction Process

In order to create a new transaction, the user should fill the transaction form by entering Receiver Address, Transaction Amount and optionally Transaction fee. Since there is a minimum relay fee in the network, if the transaction fee is not specified, it will be counted as 0.0001 BTC which is equal to minimum relay fee. The “MakeTrans” function is called in the backend when the user clicks the submit button. To make a transaction, firstly we created a raw transaction using the “createrawtransaction” RPC method. Then we sign the raw transaction using “signrawtransactionwithwallet” RPC method. Lastly, we send the signed transaction to the network using the “sendrawtransaction” RPC method. Make Transaction Form is given in Figure 6.



Create New Transaction

Dear ece_first your current balance is : 0.063914

Reciever Address: Enter the Reciever Address

Amount: Enter the Amount Fee: 0

Figure 6: Make Transaction Page

References

[1] Bitcoin. (n.d.). Retrieved November 15, 2020, from https://bitcoincore.org/
[2] Overview of Docker Compose. (2020, November 06). Retrieved November 08, 2020, from https://docs.docker.com/compose/
[3] Bitcoin Core 0.17.0 RPC. (n.d.). Retrieved November 08, 2020, from https://bitcoin-rpc.github.io/en/doc/0.17.0/
[4] Google Firebase Realtime Database (n.d.). Retrieved November 08, 2020, from https://firebase.google.com/products/realtime-database
[5] Python Pyrebase (n.d.). Retrieved November 08, 2020, from https://pypi.org/project/Pyrebase/
[6] Python Flask (n.d.). Retrieved November 08, 2020, from https://flask.palletsprojects.com/en/1.1.x/