Design

Write 4 log function for Ex2, Ex3, Ex4 and Ex5. Name them as

1. bkc\_logging\_data (use data field to store loggings)

2. bkc\_logging\_to (use to/recipient field to store loggings)

3. bkc\_logging\_batch\_data (use data field with batching)

4. bkc\_logging\_batch\_data\_hash (use data field with batching and hash)

The function 1 can also be reused by function 3 and function 4 because they all use data field to store loggings.

In function 3 and function 4, some global variables is created to store the current status of the program.

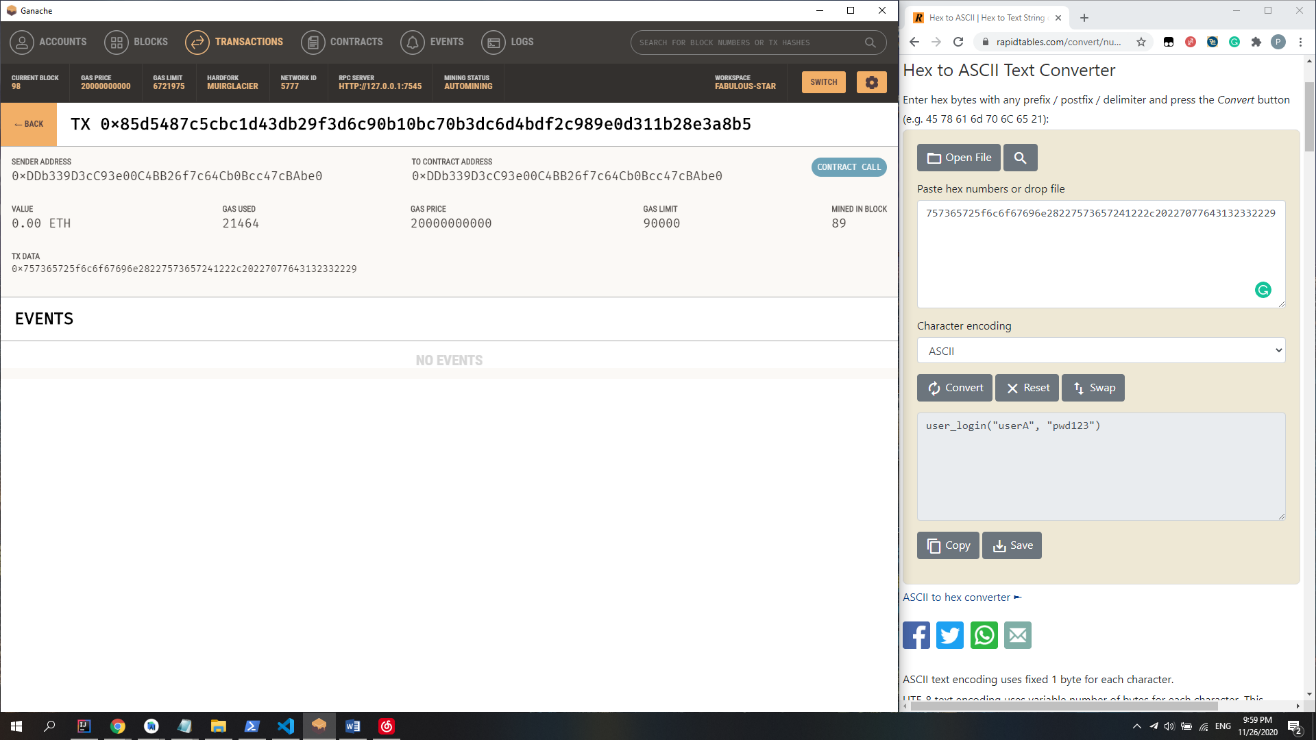
In function 2, String.prototype.padEnd is used for padding.

Issues

1. Not familiar with some JavaScript lib functions and NodeJS functions, like the hash function crypto.

Screenshots

Task 2



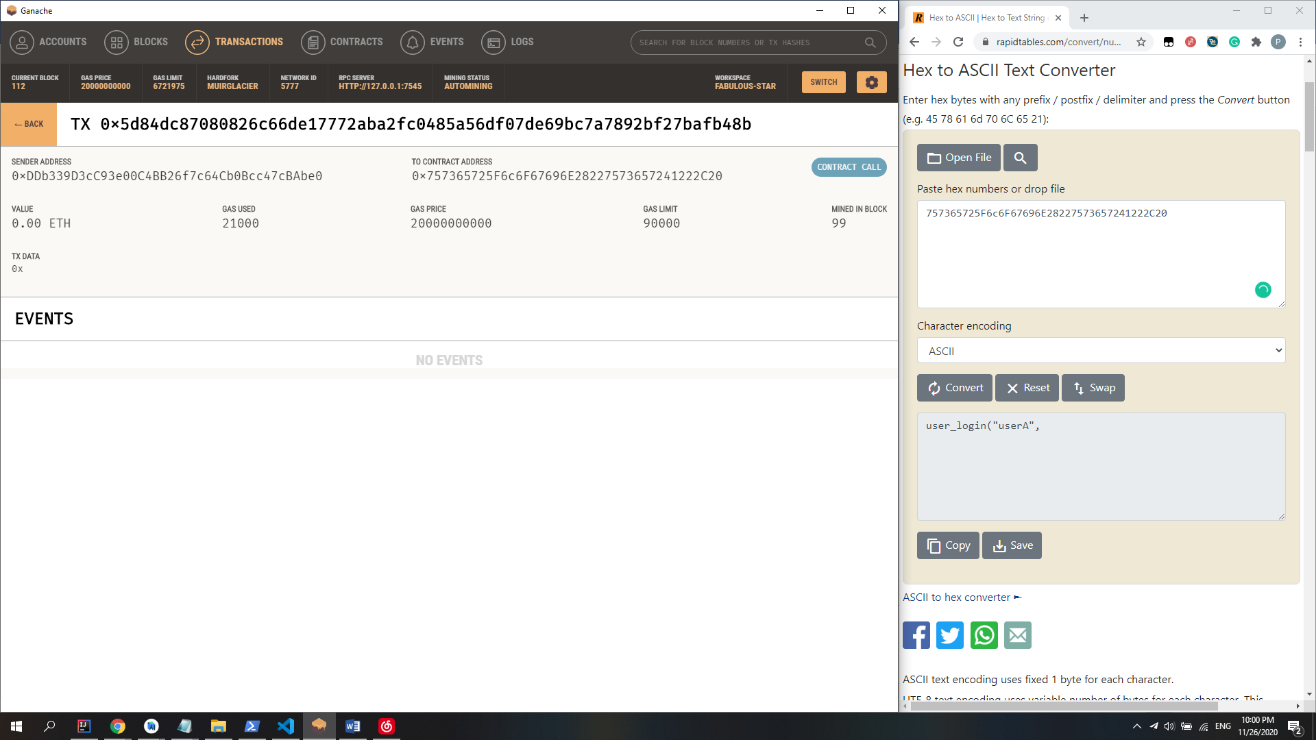
Task 2 – Store in data field

Transaction Fee

About 21500 gas per Tx \* 10 transactions

Because the data length of each Tx is not fixed, the fee of each Tx is also not fixed.

Task 3



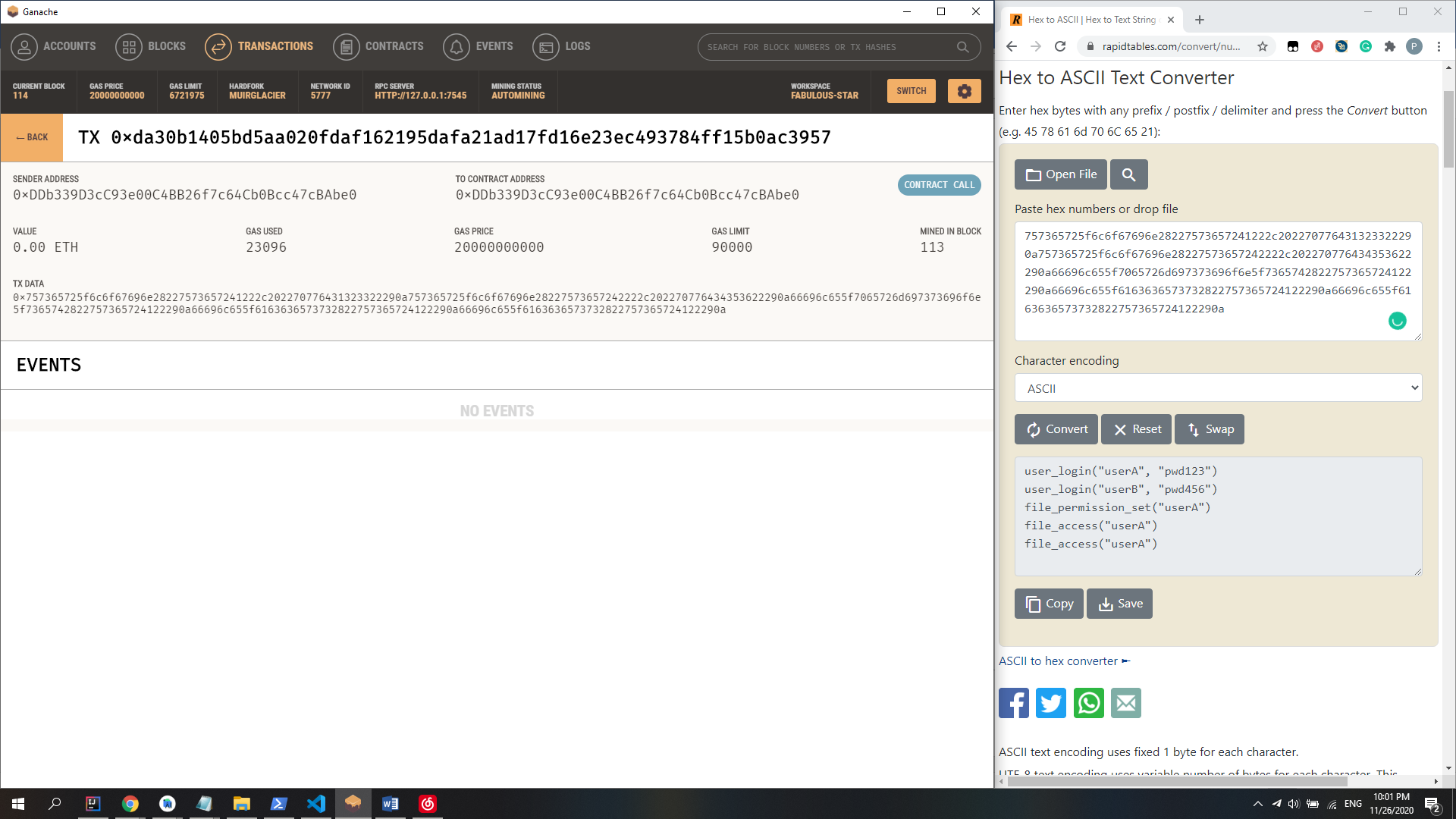
Task 3 – Store in to/recipient field

Transaction Fee

About 21000 gas per Tx \* About 15 transactions

The Tx fee is fixed. However, because we can only put 20 bytes data in the to/recipient field. There will be more than 10 transactions even if we only call the function 10 times, which will lead to a much higher total fee.

Task 5



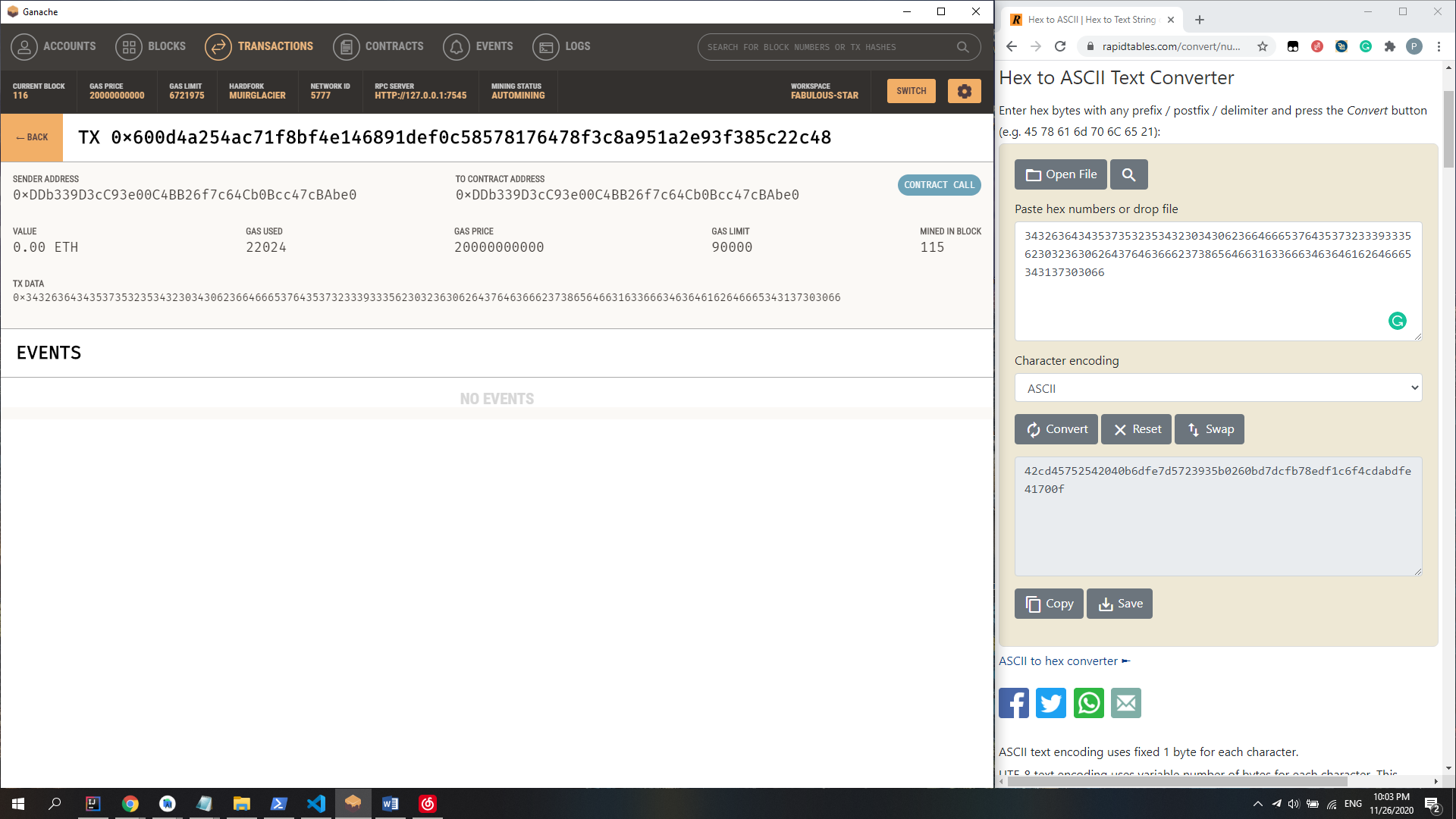
Task 5 – Store in data field and batched

Transaction Fee

About 23000 gas per Tx \* 2 transactions

Because the data length of each Tx is not fixed, the fee of each Tx is also not fixed. Compared to Task 2 the transaction number is reduced to 2. The increased fee due to the longer data field is much less than the transaction fee of a transaction with an empty data field, so we finally get a much less total transaction fee.

Task 6



Task 6 – Store in data field, batched and hashed

Transaction Fee

About 22000 gas per Tx \* 2 transactions

By hashing the payload, we will get a length-fixed and much shorter string to send. This way is the cheapest.