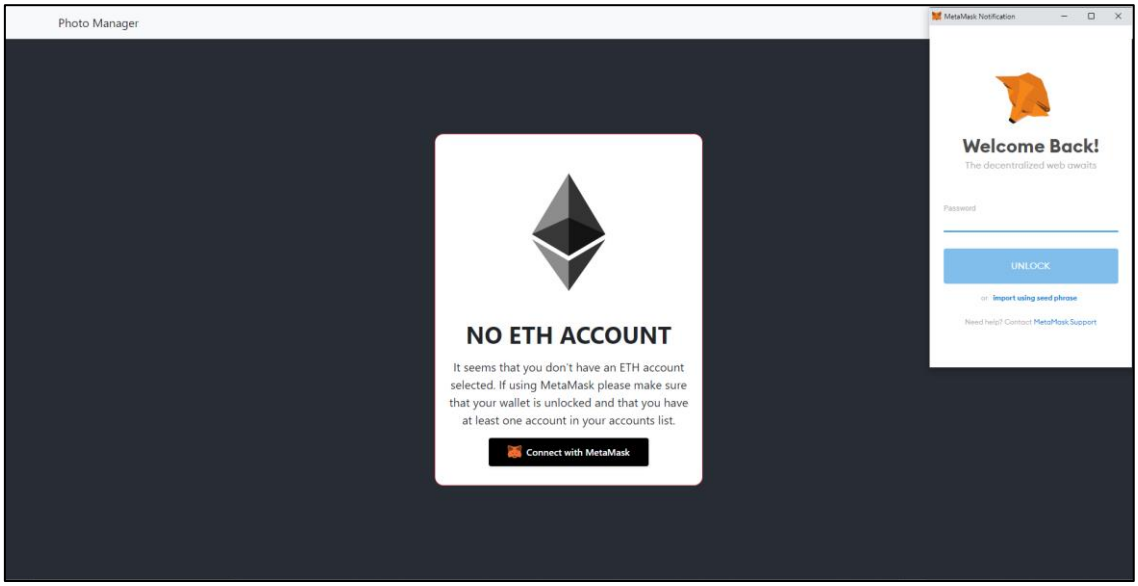
5. PENGUJIAN SISTEM

Pada bab ini akan membahas pengujian sistem berdasarkan desain sistem yang telah dibahas pada bab sebelumnya. Pengujian sistem dilakukan dengan mengukur ether yang dibutuhkan pada *gas price* yang telah disesuaikan dan melakukan perbandingan antara implementasi *Ethereum Claims Registry* orisinal dibandingkan dengan kombinasi keccak256 pada sistem ini.

5.1. Pengujian Web Application

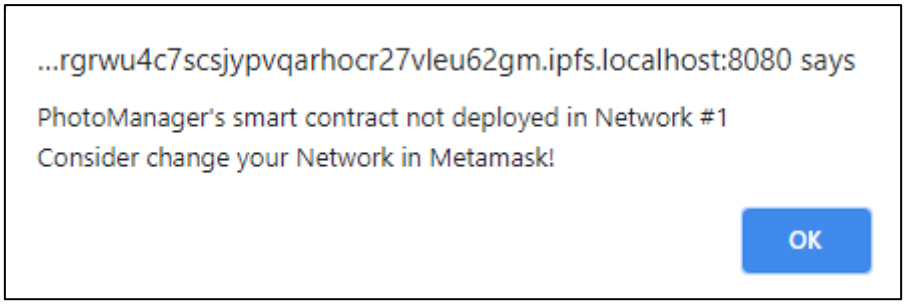
5.1.1. Pengujian Halaman Home

Saat user pertama kali mengakses halaman *website*, maka akan muncul halaman autentikasi dimana *user* harus melakukan *login* pada *wallet metamask* untuk menghubungkan *website* ke jaringan *ethereum*. Jika *user* tidak memiliki *metamask* atau tidak melakukan *login*, maka *website* tidak akan *redirect* ke halaman *home* dan user tidak bisa menggunakan keseluruhan fitur *website*.



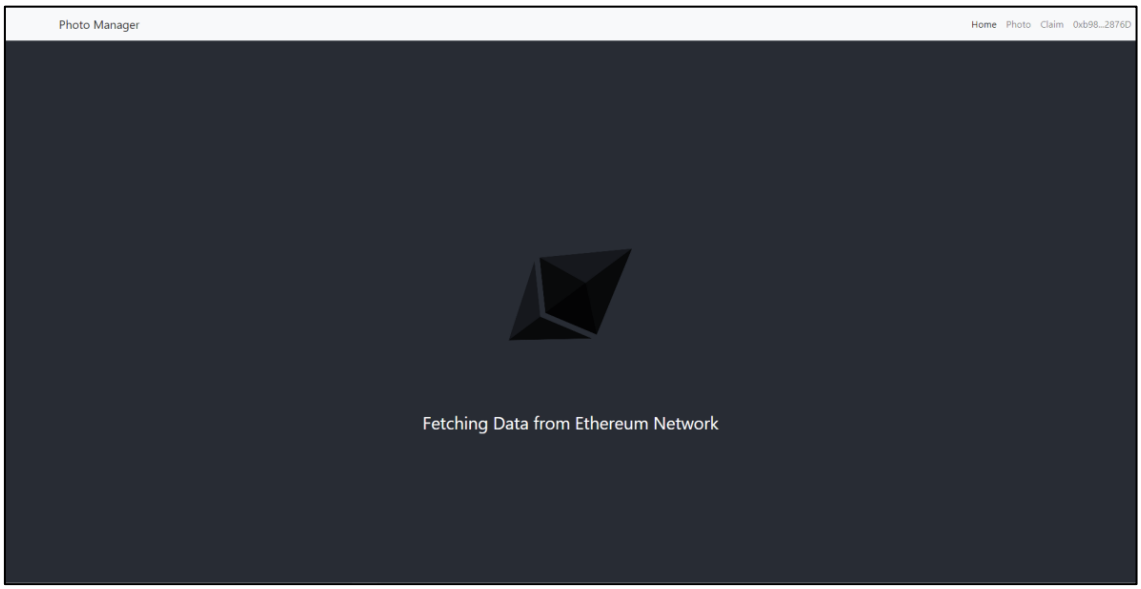
Gambar 5.1 Pengujian *autentikasi*

Saat user sudah melakukan *login* ke dalam *wallet metamask*, sistem akan melakukan koneksi terhadap *smart contract* yang berada di *network* yang dipilih oleh *user*. Apabila *smart contract* tidak ditemukan pada *network* yang dipilih maka akan muncul peringatan seperti yang ditampilkan pada gambar 5.2. Solusinya *user* dapat mengganti *network* sesuai dengan *network* lokasi *smart contract* di-*deploy*, lalu melakukan *refresh* pada halaman *website*.



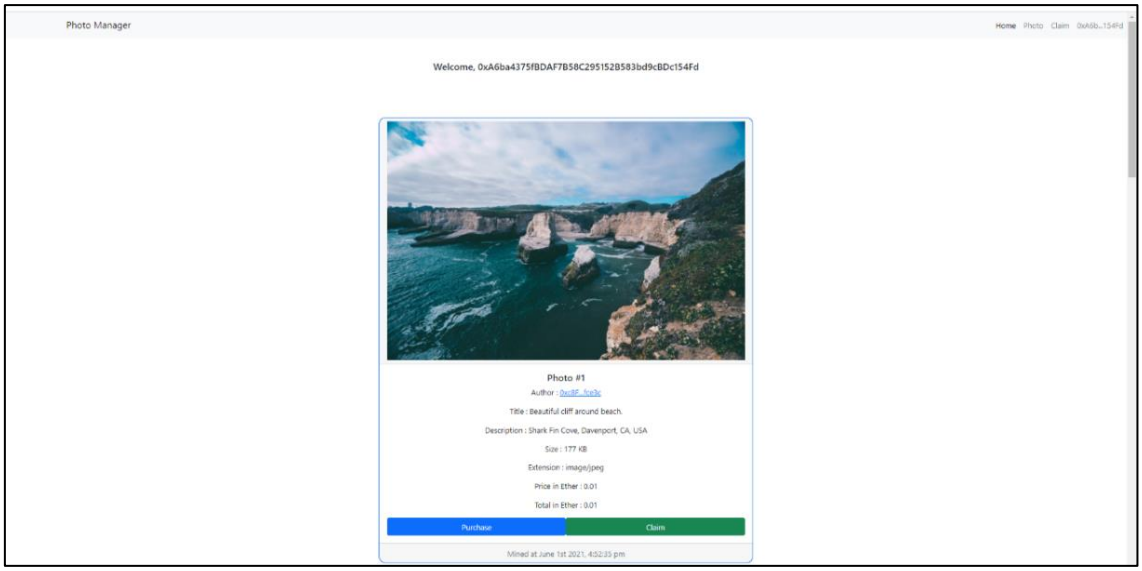
Gambar 5.2 *Smart contract tidak ditemukan*

Apabila *login* sudah benar dan *network* yang dipilih sudah sesuai dengan lokasi *smart contract* di-deploy, maka *website* akan melakukan *redirect* ke halaman *loading* untuk melakukan *fetch data* pada jaringan *ethereum* seperti Gambar 5.3.



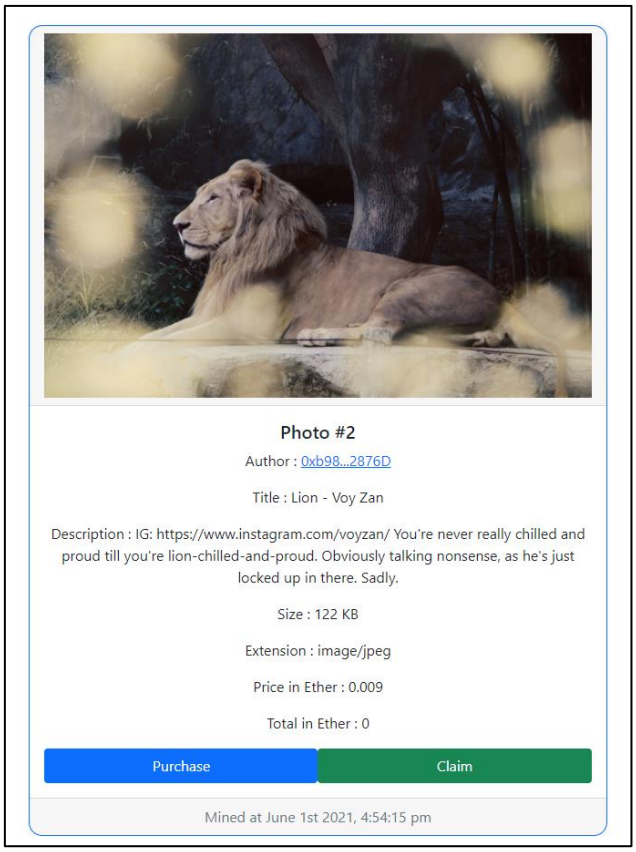
Gambar 5.3 Pengujian *loading*

Setelah *website* berhasil melakukan *fetch data* dari *blockchain network*, maka *website* akan redirect ke halaman *home* yang merupakan halaman utama dalam sistem konten fotografi. Halaman ini berisikan semua *list* konten fotografi yang terdaftar pada jaringan *ethereum*, seperti yang ditampilkan pada Gambar 5.4.

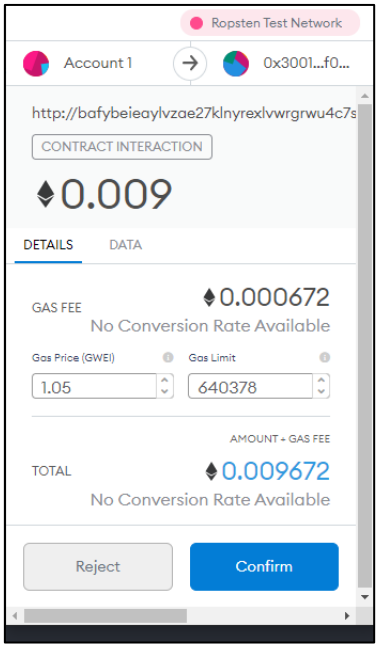


Gambar 5.4 Pengujian *fetch* konten fotografi

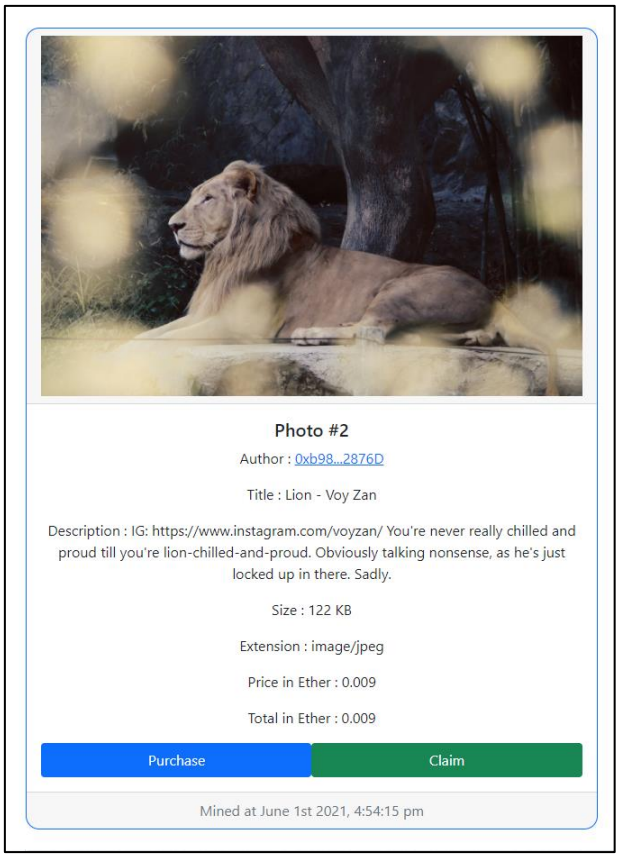
Pada bagian *list* foto, *user* bisa melakukan *make claim* pada sebuah konten fotografi dengan melakukan klik pada tombol *make claim* seperti pada Gambar 5.5. Setelah itu *user* harus melakukan konfirmasi transaksi pada *metamask* agar melanjutkan transaksi ke jaringan *ethereum* seperti pada Gambar 5.6. Jika *user* sudah pernah melakukan *make claim* pada sebuah konten fotografi, maka *user* dimungkinkan untuk melakukan *claim* untuk mendapatkan *hash* foto dengan melakukan klik pada tombol *get claim* seperti pada Gambar 5.7. Lalu user akan di *redirect* ke halaman *loading* dan akan mendapatkan notifikasi berupa *hash* foto yang di-*claim* seperti pada Gambar 5.8.



Gambar 5.5 Pengujian *make claim*



Gambar 5.6 Konfirmasi transaksi *make claim*



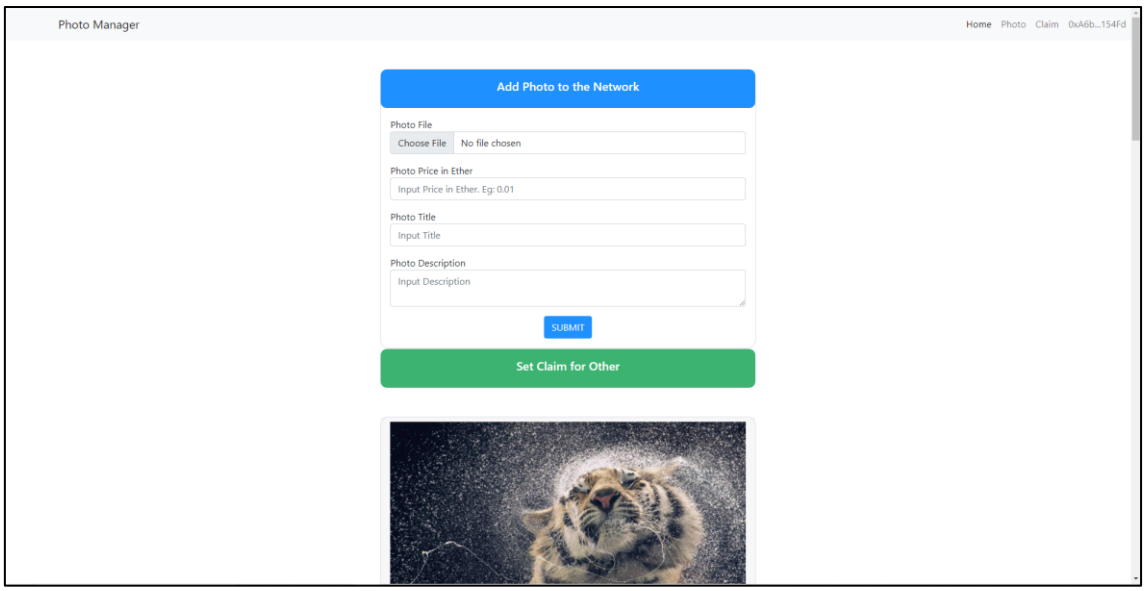
Gambar 5.7 Foto setelah *make claim*

Gambar 5.8 *Get claim* pada Foto

5.1.2. Pengujian Halaman Photo

Pada halaman ini, *user* dapat melakukan pengelolaan konten fotografi diantaranya fitur *get all photo*, *add photo*, *edit photo state*, dan *set claim*. Saat pertama kali dibuka, *website* akan melakukan *fetch* data ke jaringan *ethereum* lalu ditampilkan seperti Gambar 5.9. Fitur *add photo* memungkinkan user untuk menambahkan konten fotografi ke jaringan *IPFS* dan sekaligus menambahkan informasi konten fotografi ke dalam jaringan *ethereum* seperti *author*, hash IPFS,

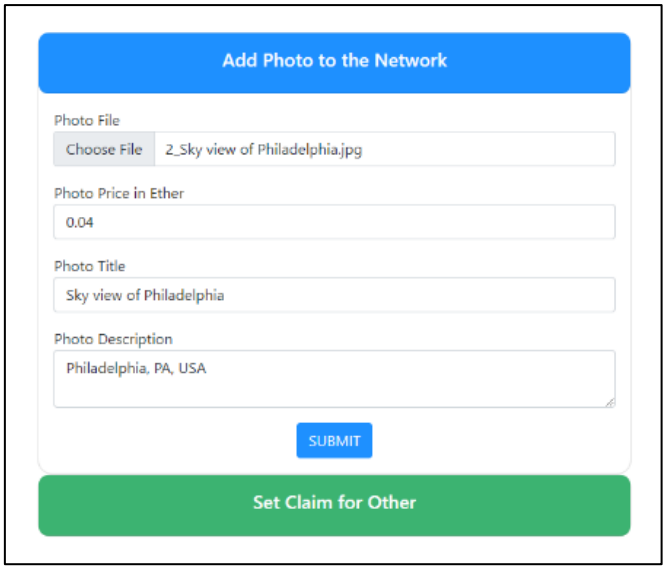
title, *description*, dan *price in ether*. Fitur *edit photo state* memungkinkan *user* untuk mengubah *state* pada foto yang dimiliki, *state* pada foto berpengaruh dalam munculnya konten foto pada halaman *home*. Ketika konten foto dalam keadaan *active* maka foto akan muncul pada halaman *home*, jika dalam keadaan *inactive* maka foto tidak akan muncul pada halaman *home* dan *user* tidak dapat melakukan *make claim* terhadap foto yang *inactive*.



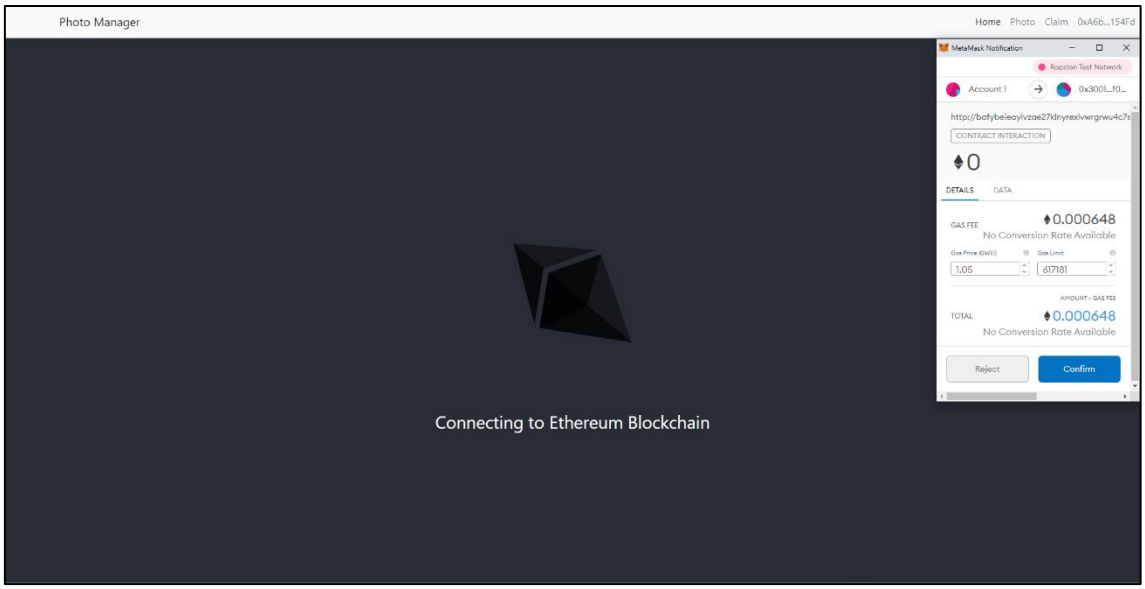
Gambar 5.9 Pengujian halaman *photo*

Pada bagian *add photo*, *user* harus mengisi semua *input* yang ada pada form dan *user* harus melakukan *upload* foto yang diinginkan lalu melakukan klik pada tombol *submit* seperti pada Gambar 5.10. Setelah itu *website* akan *redirect* ke halaman *loading* dan meminta user melakukan konfirmasi transaksi pada *MetaMask* agar melanjutkan transaksi ke jaringan *ethereum* seperti pada Gambar 5.11. Setelah transaksi sudah divalidasi oleh *miner*, maka foto

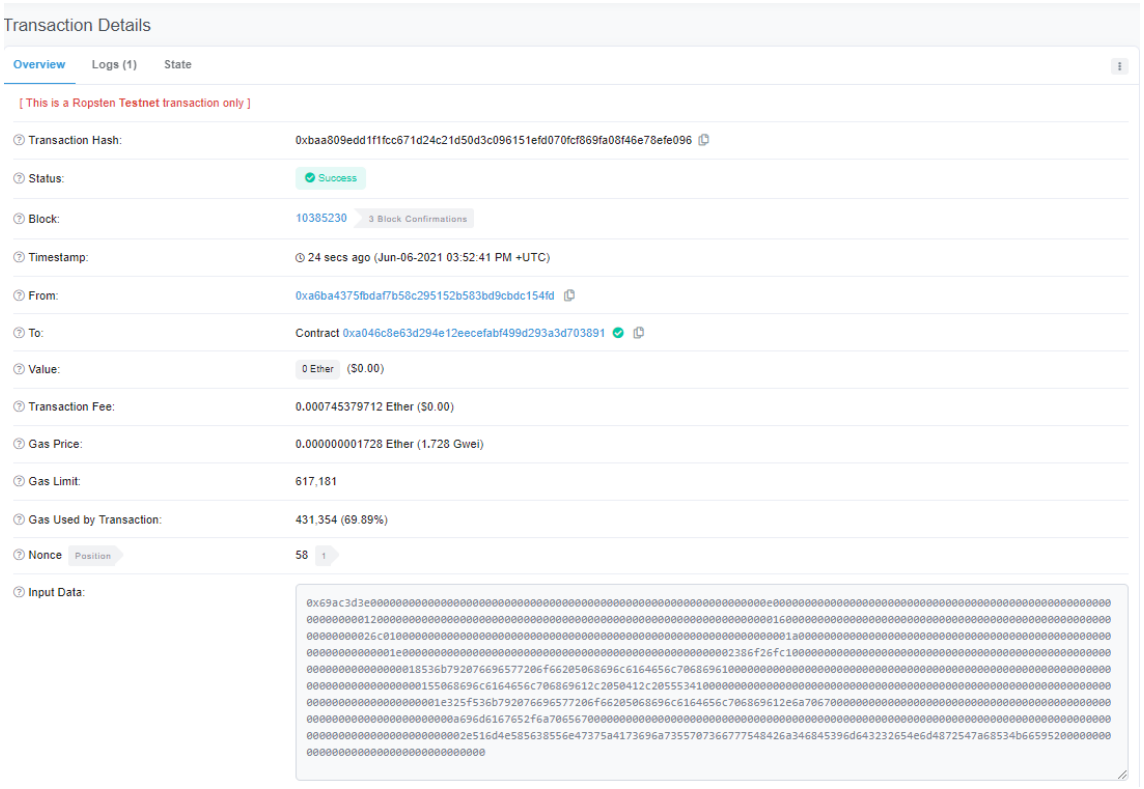
akan muncul pada halaman website dan user dapat melihat transaksi pada *MetaMask* yang bisa diredirect ke *etherscan* untuk detail transaksi seperti pada Gambar 5.12.



Gambar 5.10 Pengujian *add photo*

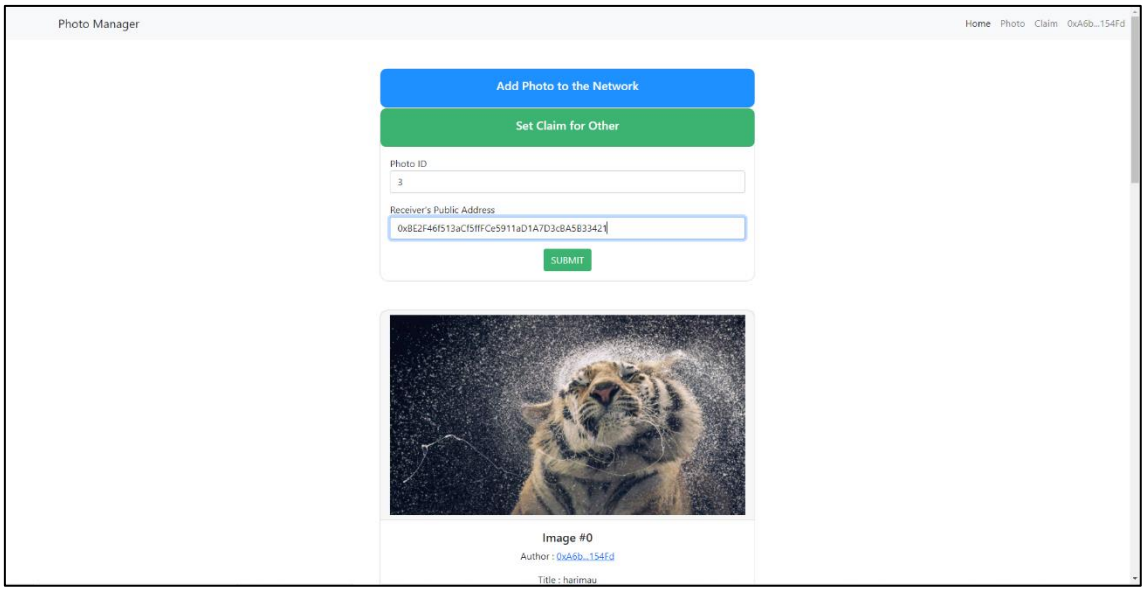


Gambar 5.11 Konfirmasi transaksi pada fitur *add photo*

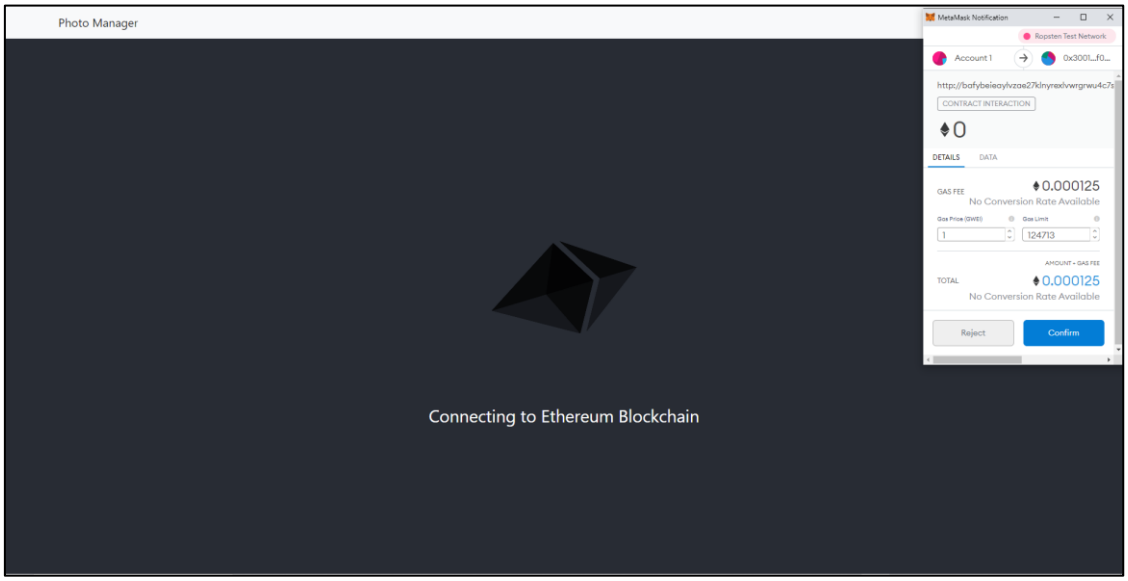


Gambar 5.12 *Detail* transaksi *add photo* pada *etherscan*

Pada bagian *set claim*, user harus mengisi semua *input* yang ada pada form berupa *photo id* dan *receiver public address* seperti pada Gambar 5.13. Setelah mengisi form, user dapat melakukan klik pada tombol *submit* untuk melakukan konfirmasi transaksi pada *MetaMask* agar melanjutkan transaksi seperti pada Gambar 5.14.

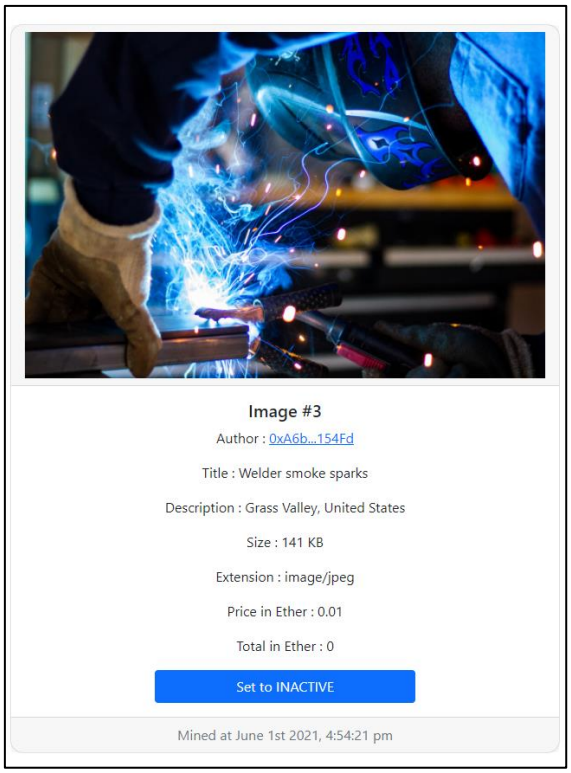


Gambar 5.13 Pengujian *set claim*

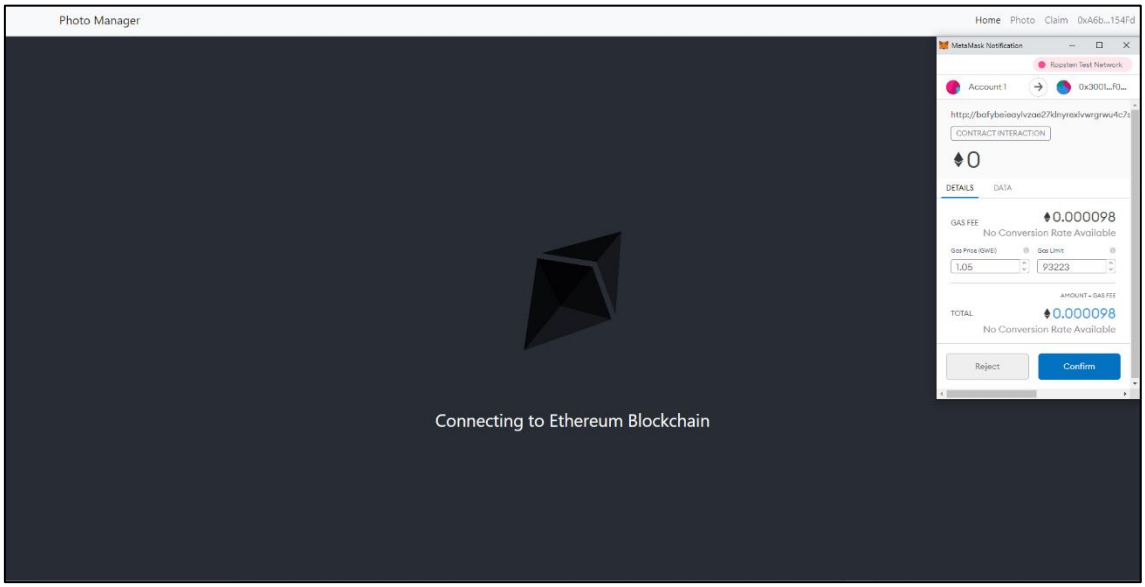


Gambar 5.14 Konfirmasi transaksi pada *set claim*

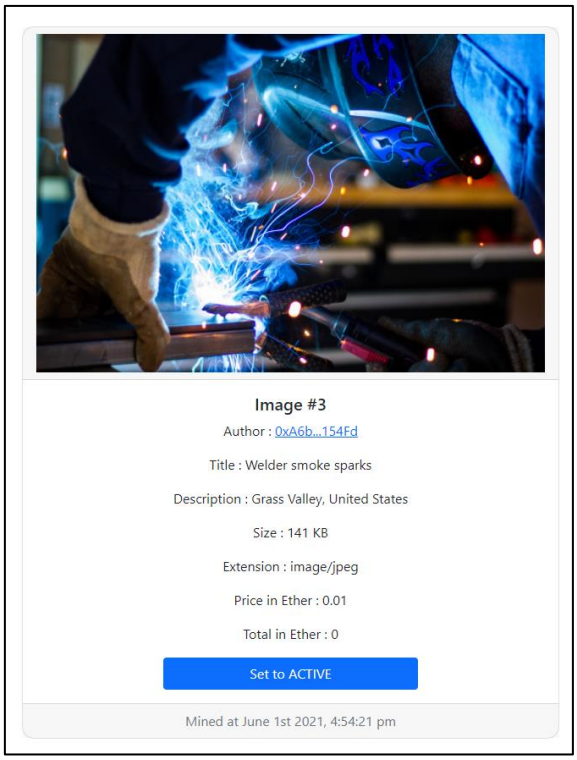
Pada bagian *list* foto, user dapat mengubah *state* dari foto dengan melakukan klik pada tombol *set to inactive* seperti pada Gambar 5.15. Lalu *user* harus melakukan konfirmasi pembayaran seperti pada Gambar 5.16, jika transaksi sudah tervalidasi maka tombol pada foto yang terubah *state*-nya akan berubah menjadi *set to active* seperti pada Gambar 5.17.



Gambar 5.15 *State* foto sebelum diubah



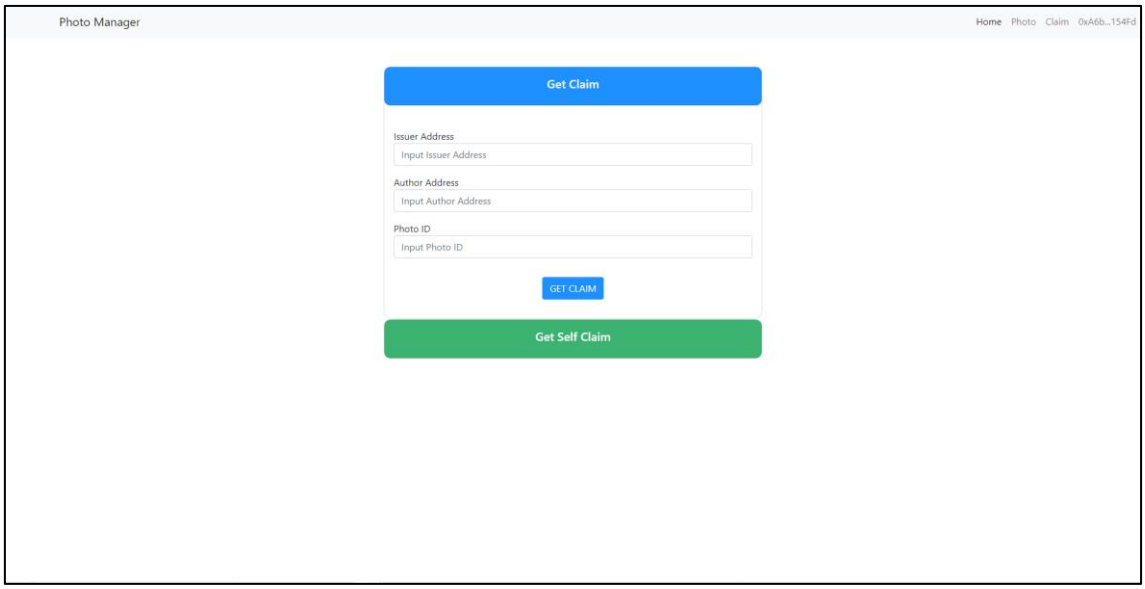
Gambar 5.16 Konfirmasi transaksi pada fitur *edit state*



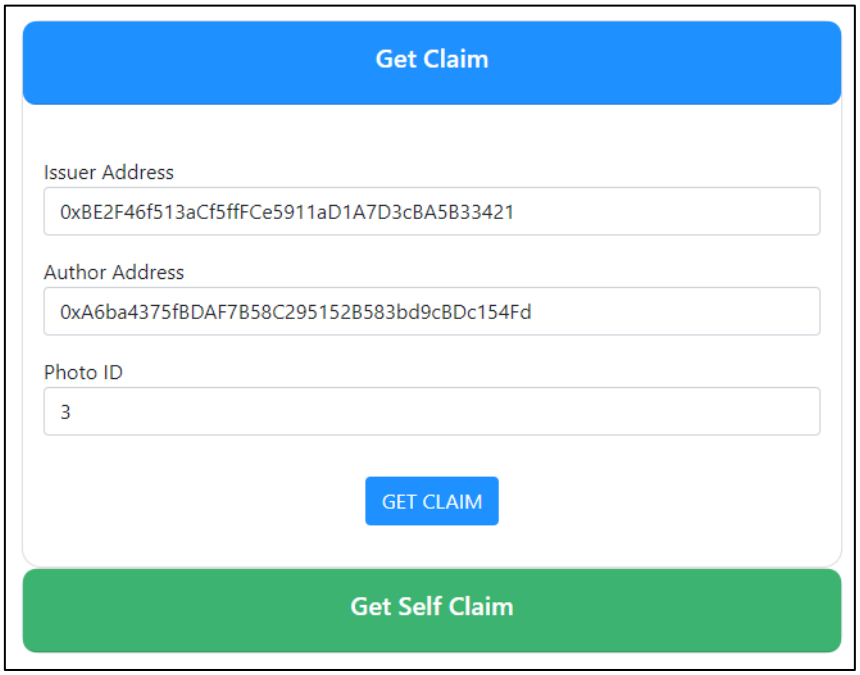
Gambar 5.17 *State* foto setelah diubah

5.1.3. Pengujian Halaman Claim

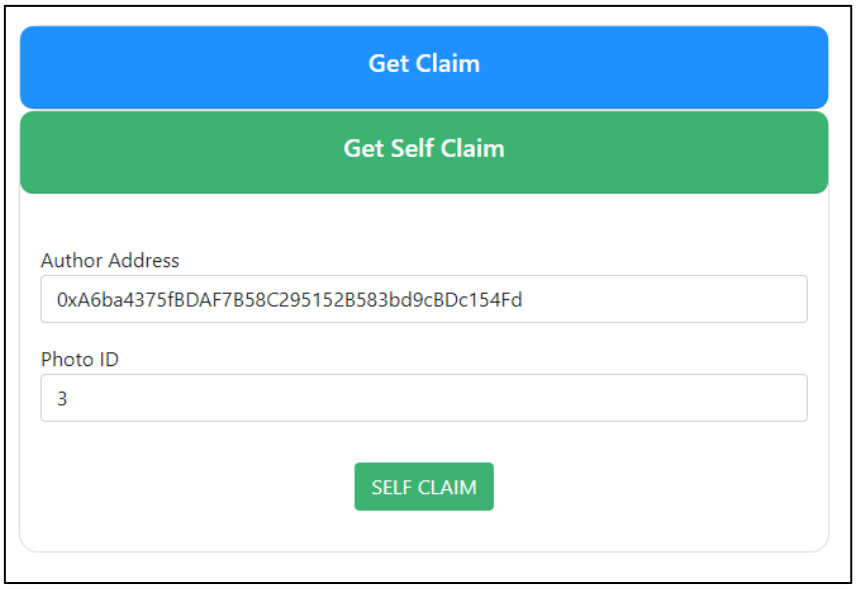
Pada Halaman ini, *user* dimungkinkan untuk mendapatkan *claim* diantaranya fitur *get self claim* dan *get claim* seperti pada Gambar 5.18. Untuk menggunakan fitur *claim*, *user* harus mengisi *form* yang sesuai dengan fitur yang dibutuhkan seperti pada Gambar 5.19 dan Gambar 5.20. Fitur *claim* dapat dilakukan pada konten fotografi yang memiliki *state inactive* dan fitur *claim* bebas biaya transaksi karena hanya menggunakan *call method* pada jaringan *ethereum*. Setelah melakukan klik pada tombol *claim* maka *website* akan *redirect* ke halaman *loading* dan *user* akan mendapatkan notifikasi berupa *hash* dari foto seperti pada Gambar 5.21.



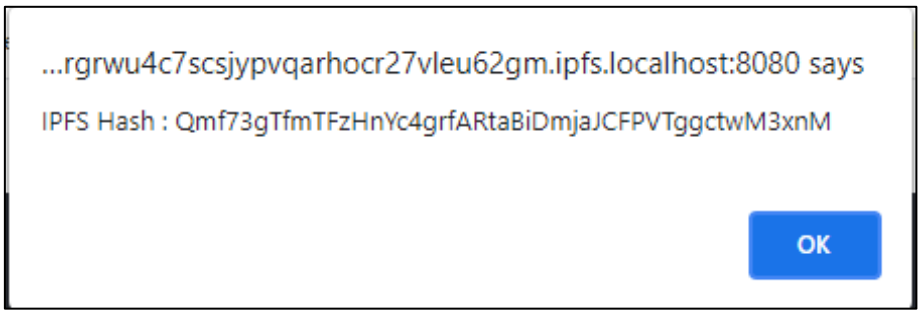
Gambar 5.18 Pengujian halaman *claim*



Gambar 5.19 Form get *claim*



Gambar 5.20 Form *get self claim*



Gambar 5.21 Hash dari fitur *claim*

5.2. *Cost* pada *Ethereum*

Pengujian *cost* pada *ethereum* dilakukan untuk mendapatkan jumlah *cost* yang dibutuhkan oleh sistem konten fotografi. Parameter yang dibutuhkan dalam penghitungan adalah gas yang dibutuhkan, harga gas, harga *ether* terhadap rupiah. Data *gas price* yang ditampilkan pada tabel 5.1 didapatkan dari website etherscan yang diakses pada tanggal 6 Juni 2021. *Gas price* dibagi menjadi 3 jenis yaitu *high*, *average*, dan *low*. Masing-masing jenis memiliki *range* diantaranya *max*, *average*, dan *min*. Smart contract pada pengujian di-deploy pada *test network*, sehingga dibutuhkan penyesuaian dengan parameter *gas price* untuk mendapatkan jumlah biaya yang mendekati *main network*. Penyesuaian *gas price* dilakukan dengan menambahkan jenis *gas price* dengan *range* yang sama kemudian dibagi 3 seperti pada

tabel 5.1. Data harga *ether* terhadap rupiah didapatkan dari *website coinmarket* yang diakses pada tanggal 6 Juni 2021, dengan harga Rp 37.511.701 per ether seperti yang ditampilkan pada tabel 5.2.

Tabel 5.1

Gas Price pada *Etherscan*

Historical Gas Oracle Prices	**Gas Price in Gwei**		
	High	**Average**	**Low**
Max	100	36	16
Average	15	12	10
Min	11	10	5
Penyesuaian	42	19	10

Sumber: https://etherscan.io/gastracker

Tabel 5.2

Ether Price terhadap R*upiah*

Ether (ETH)	**Rupiah (IDR)**
1	Rp37.511.701

Sumber : https://coinmarketcap.com

Tabel 5.3

List Foto untuk Pengujian

No	**Photo File (JPG)**	**Description**	**IPFS Hash**	**Price in Eth**	**Size**	**Dimension in Pixels**
#1	Beautiful cliff around beach	Shark Fin Cove, Davenport, CA, USA	QmR95tG8zcHzKJrRvNjdUR16KtB3KnCeddZA6AtVSF7nZh	0.01	176 KB	1050x700
#2	Sky view of Philadelphia	Philadelphia, PA, USA	QmNXV8UnG7ZAsijsUpsfwuHBj4hE9md22eNmHrTzhSKfYR	0.01	154 KB	1050x700

#3	Lion	You're never really chilled and proud till you're lion-chilled-and-proud. Obviously talking nonsense, as he's just locked up in there. Sadly.	QmXn8r25e9Z7neo4aRTwBvU2QDzBHsyB1XDzphnGww9TYD	0.01	121 KB	1050x700
#4	Lion Cub	Big yawn from a small big cat in Sabi Sands reserve which is part of the Kruger National Park in South Africa	QmVn99fHEzCcyCkW2APeGvVQmyJni4ERvf1DJCBvt4PeWZ	0.01	137 KB	1050x700
#5	Welder smoke sparks	Grass Valley, United States	Qmf73gTfmTFzHnYc4grfARtaBiDmjaJCFPVTggctwM3xnM	0.01	141 KB	1050x700

Gas yang diperlukan pada tiap foto berbeda karena data yang disimpan ke storage variable pada ethereum blockchain berbeda, gas yang diperlukan akan semakin meningkat jika data yang disimpan semakin banyak. Ukuran foto tidak secara signifikan mempengaruhi harga foto, karena yang disimpan pada Ethereum blockchain adalah hash yang didapatkan dari IPFS dengan ukuran 48 string. Data yang dikirim ke dalam ethereum berupa string dan int256 kemudian diformat menjadi struct dalam sebuah mapping.

Tabel 5.4

Pengajuan *Cost* pada Fitur *Add* dengan *High Gas Price*

Photo	**Gas Used**	**Gwei**	**ETH**	**IDR**
Photo #1	515045	42	0.021632	Rp843,119.29
Photo #2	431354	42	0.018117	Rp706,118.64
Photo #3	543734	42	0.022837	Rp890,082.66
Photo #4	521130	42	0.021887	Rp853,080.32
Photo #5	431306	42	0.018115	Rp706,040.07
Total	2442569		0.102588	Rp3,998,440.98
Rata-rata	488514	42	0.020518	Rp799,688.20

Tabel 5.5

Pengujian *Cost* pada Fitur *Add* dengan *Average Gas Price*

Photo	**Gas Used**	**Gwei**	**ETH**	**IDR**
Photo #1	515045	19	0.011331	Rp441,633.91
Photo #2	431354	19	0.00949	Rp369,871.67
Photo #3	543734	19	0.011962	Rp466,233.77
Photo #4	521130	19	0.011465	Rp446,851.60
Photo #5	431306	19	0.009489	Rp369,830.51
Total	2442569		0.053737	Rp2,094,421.47
Rata-rata	488513.8	19	0.010747	Rp418,884.29

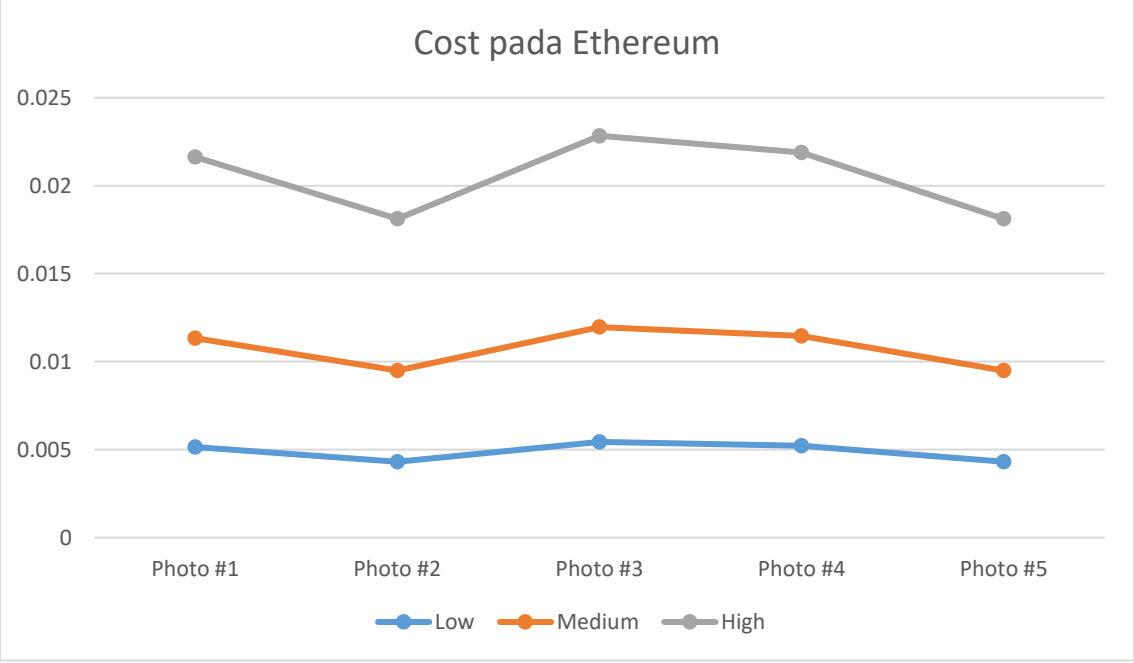
Tabel 5.6

Pengujian *Cost* pada Fitur *Add* dengan *Low Gas Price*

Photo	**Gas Used**	**Gwei**	**ETH**	**IDR**
Photo #1	515045	10	0.00515	Rp200,742.69
Photo #2	431354	10	0.004314	Rp168,123.49
Photo #3	543734	10	0.005437	Rp211,924.44

Photo #4	521130	10	0.005211	Rp203,114.36
Photo #5	431306	10	0.004313	Rp168,104.78
Total	2442569		0.024426	Rp952,009.76
Rata-rata	488513.8	10	0.004885	Rp190,401.95

Pada hasil pengujian Implementasi *Ethereum Claims Registry* dan *InterPlanetary File System* pada *high gas price*, mendapatkan total sebanyak 0.102588 *ETH* atau Rp3,998,440.98 dan rata-rata sebanyak 0.020518 *ETH* atau Rp799,688.20. Pada *average gas price* mendapatkan total sebanyak 0.053737 *ETH* atau Rp2,094,421.47 dan rata-rata sebanyak 0.010747 *ETH* atau Rp418,884.29. Pada *low gas price* mendapatkan total sebanyak 0.024426 ETH atau Rp952,009.76 dan rata-rata sebanyak 0.004885 ETH atau Rp190,401.95. Pada Gambar 5.22 menampilkan perbandingan *cost ether* yang dibutuhkan dengan *gas price* yang telah disesuaikan.



Gambar 5.22 Grafik *cost* pada *ethereum* berdasarkan *gas price*

5.3. Perbandingan Ether dan Stuktur pada Implementasi Ethereum Claims Registry dengan Keccak256 dibandingkan orisinal

Perbandingan dilakukan melalui *Remix – Ethereum IDE* menggunakan *injected web3 environment* lalu *deploy smart contract* ke *ropsten test network*. *Ether* yang digunakan untuk

deploy smart contract merupakan *test ether* yang dapat di-request pada *rospten ethereum faucet*. Transaksi dilakukan satu demi satu lalu lalu dicatat seperti yang ditampilkan pada Tabel 5.7, Tabel 5.8, dan Tabel 5.9.

Tabel 5.7

Perbandingan *Ether* pada *Deploy Smart Contract*

Deploy Smart Contract	**Gas**	**GWEI**	**ETH**	**IDR**
Ethereum Claims Registry dan Keccak256	2805011	10	0.0280501	Rp1,093,274.27
Ethereum Claims Registry Orisinal	2833026	10	0.0283303	Rp1,104,193.33
Selisih (Keccak256 – Orisinal)	-24234		-0.0002802	-Rp10,946.06
Efisiensi % (Keccak256)	1%			

Tabel 5.8

Perbandingan *Ether* pada Transaksi *Make Claim*

Transaksi Make Claim (asumsi harga foto 0 eth)	**Gas**	**GWEI**	**Ether**	**IDR**
Ethereum Claims Registry dan Keccak256	105844	10	0.0010584	Rp41,253.50
Ethereum Claims Registry Orisinal	105466	10	0.0010547	Rp41,106.17
Selisih (Keccak256 – Orisinal)	378		0.0000037	Rp147.33
Efisiensi (Keccak256)	-0.4%			

Tabel 5.9

Perbandingan *Ether* pada Transaksi *Set Claim*

Transaksi Set Claim	**Gas**	**GWEI**	**Ether**	**IDR**
Ethereum Claims Registry dan Keccak256	91696	10	0.000917	Rp35,739.21
Ethereum Claims Registry Orisinal	91306	10	0.000913	Rp35,587.20
Selisih (Keccak256 – Orisinal)	390		0.000004	Rp152.01
Efisiensi (Keccak256)	-0.4%			

Dari hasil pengujian diatas, maka *Ethereum Claims Registry* dan *Keccak256* dapat menghemat jumlah *ether* pada *deploy smart contract* dibandingkan *Ethereum Claims Registry*

orisinal, dengan selisih 24.234 gas dan 0.00028 ETH atau Rp10.919,06 seperti yang ditampilkan pada Tabel 5.7. Pada fitur *make claim Ethereum Claims Registry* dan *Keccak256* mempunyai beban *ether* lebih banyak, sebesar 378 gas dan 0.0000003 ETH atau Rp147,33 seperti yang ditampilkan pada Tabel 5.8. Pada fitur *set claim Ethereum Claims Registry* dan *keccak256* mempunyai beban *ether* lebih banyak, sebesar 390 gas dan 0.000004 ETH atau Rp152,01 yang ditampilkan pada Tabel 5.9.

Dengan implementasi *Ethereum Claims Registry* dan *Keccak256* mengubah struktur bentuk standar asal yang 3 dimensi menjadi satu dimensi, seperti yang ditampilkan pada Gambar 5.23 dan Gambar 5.24. Ethereum Claims Registry merupakan storage variable yang berbentuk mapping, memiliki key berupa bytes32 dan memiliki value berupa struct dari OwnedPhoto. Sedangkan Ethereum Claims Registry Orisinal merupakan storage variable yang berbentuk nested mapping dan value berupa struct Owned Photo, sehingga membentuk pola 3 dimensi. Ethereum Claims Registry dan Keccak256 lebih baik bila diimplementasikan pada struktur dengan dimensi lebih dari satu dengan perbedaan *struktur* hanya ada pada *storage variable* dan akan mempengaruhi cost pada smart contract. Namun pada untuk transaksi tidak ada perbedaan yang cukup signifikan seperti yang ditampilkan pada Segmen Program 5.1 dan Segmen Program 5.2. Smart contract hanya di-deploy satu kali oleh pemilik dari smart contract, Ketika smart contract sudah divalidasi oleh miner maka smart contract dapat digunakan pada system. Setelah itu user dapat melakukan operasi make claim dan set claim pada sistem yang terintegrasi dengan jaringan *ethereum blockchain*.

```
mapping(bytes32 => OwnedPhoto) public photos;

function setClaim(address _author, uint256 _id) public {
    require(lists[_id]._author == msg.sender, "You are not the author");

    PhotoList memory getPhoto = lists[_id];
    bytes32 encryptedBytes = keccak256(abi.encode(msg.sender, getPhoto._author, _id));

    claims[encryptedBytes] = OwnedClaim (msg.sender, block.timestamp, getPhoto._author, getPhoto._title, getPhoto._description, getPhoto._ipfsHash);
}
```

Gambar 5.23 Struktur pada Implementasi Ethereum Claims Registry dan Keccak256

```
mapping(address => mapping(address => mapping(uint => OwnedClaim))) private claims;

function setClaim(address _author, uint256 _id) public {
    require(lists[_id]._author == msg.sender, "You are not the author");

    PhotoList memory getPhoto = lists[_id];

    claims[msg.sender][getPhoto._author][_id] = OwnedClaim (msg.sender, block.timestamp, getPhoto._author, getPhoto._title, getPhoto._description, getPhoto._ipfsHash);
}
```

Gambar 5.24 Struktur pada Implementasi Ethereum Claims Registry Orisinal

Segmen Program 5.1 Transaksi pada Ethereum Claims Registry dan Keccak256

```
{
status : "true Transaction mined and execution succedd"
transaction hash :
"0x776b4e942d5d2bbaf93b376ad67ca8b8c3c73ca157e9d090894523e5f303e287"
from : "0xA6ba4375fBDAF7B58C295152B583bd9cBDc154Fd"
to : "PhotoManager.addPhoto(string,string,uint256,string,string,uint256)
0x52D7651926eCB55fE238349cD10F2585e72Be336"
gas : "60000 gas",
transaction cost : "492620 gas",
hash : "0x776b4e942d5d2bbaf93b376ad67ca8b8c3c73ca157e9d09089452e5f30303e287"
input : "0x69a...00000"
decoded input : { "string _title": Beautiful cliff around beach", "string _desc":
"Shark Fin Cove, Davenport, CA, USA", "string _name": "1_Beautiful Cliff Around
Beach", "uint256 _size": "100", "string _ext": "image/jpg", "string _hash_":
"QmTzaDPVDLtiKNE2ZmCACGd23PyunnwoVbCJcmGCGVWWcg", "uint256 _price": "10000000" },
logs : [{
"from": "0x52D7651926eCB55fE238349cD10F258e72Be336",
"topic": "0xd940408807ccffd8ff57cbd00e86212cacfcf6048f7b1b71d8db001b806ca3fa",
"event": "PhotoEvent",
"args": { "0": "0", "1": "Beautiful cliff around beach", "2":
"QmTzaDPVDLtiKNE2ZmCACGd23PyunnwoVbCJcmGCGVWWcg", "3": 0, "4": "1623259481",
"photoId": "0", "photoTitle": "Beautiful cliff around beach", "photoHash":
"QmTzaDPVDLtiKNE2ZmCACGd23PyunnwoVbCJcmGCGVWWcg", "_state": 0, "addedAt":
"1623259481"}
}]
value : "0 wei"
}
```

Segmen Program 5.2 Transaksi pada Ethereum Claims Registry Orisinal

```
{
status : "true Transaction mined and execution succedd"
transaction hash :
"0x62235983098b895ba791a7300aa1adb11ed4d878af1edf5a11be1e3835625590"
from : "0xA6ba4375fBDAF7B58C295152B583bd9cBDc154Fd"
to : "PhotoManager.addPhoto(string,string,uint256,string,string,uint256)
0x93801674e6649e4CAaAaf87D408A3555eC00D497"
gas : "60000 gas",
transaction cost : "492620 gas",
hash : "0x62235983098b895ba791a7300aa1adb11ed4d87af1edf5a11be1e3835625590"
input : "0x69a...00000"
decoded input : { "string _title": Beautiful cliff around beach", "string _desc": "Shark
Fin Cove, Davenport, CA, USA", "string _name": "1_Beautiful Cliff Around Beach",
"uint256 _size": "100", "string _ext": "image/jpg", "string _hash_":
"QmTzaDPVDLtiKNE2ZmCACGd23PyunnwoVbCJcmGCGVWWcg", "uint256 _price": "10000000"},
logs : [{
"from": "0x93801674e6649e4CAaAaf87D408A3555eC00D497",
```

```
"topic": "0xd940408807ccffd8ff57cbd00e86212cacfcf6048f7b1b71d8db001b806ca3fa",
"event": "PhotoEvent",
"args": { "0": "0", "1": "Beautiful cliff around beach", "2":
"QmTzaDPVDLtiKNE2ZmCACGd23PyunnwoVbCJcmGCGVWWcg", "3": 0, "4": "1623259481", "photoId":
"0", "photoTitle": "Beautiful cliff around beach", "photoHash":
"QmTzaDPVDLtiKNE2ZmCACGd23PyunnwoVbCJcmGCGVWWcg", "_state": 0, "addedAt": "1623259880"}
}]
value : "0 wei"
}
```


Dari hasil pengujian perbandingan *cost* dan *struktur*, dari tabel dan gambar diatas maka dapat disimpulkan bahwa implementasi *Ethereum Claims Registry* dan *Keccak256* lebih hemat gas saat *deploy* namun untuk transaksi *make claim* dan *set claim* mendapatkan beban gas lebih banyak daripada *Ethereum Claims Registry* Orisinal. Struktur pada *Ethereum Claims Registry* dan *Keccak256* memakan gas lebih banyak karena terdapat proses enkripsi di dalam smart contract dibandingkan *Ethereum Claims Registry* Orisinal.