```
require('dotenv').config();
const express = require('express');
const bodyParser = require('body-parser');
const fs = require('node:fs');
const path = require('node:path');
const { NodeAdapterReedSolomon } =
require('@bnb-chain/reed-solomon/node.adapter');
const { Client } = require('@bnb-chain/greenfield-js-sdk');
const mimeTypes = require('mime-types');

const app = express();
process.on('unhandledRejection', (reason, promise) => {
    console.error('Unhandled Rejection at:', promise, 'reason:', reason);
    // Optionally, you can log the error details or perform other actions
here.
  });

  process.on('uncaughtException', (error) => {
    console.error('Uncaught Exception:', error);
    // Optionally, you can log the error details or perform other actions
here.
  });
const port = 4401; // Choose a port for your server

app.use(bodyParser.json());

const client =
Client.create('https://gnfd-testnet-fullnode-tendermint-us.bnbchain.org',
'5600');

const { ACCOUNT_ADDRESS, ACCOUNT_PRIVATEKEY } = process.env;

const createObject = async (req, res) => {
  try {
    const { bucketName, objectName } = req.body;

    // Get storage providers list
    const sps = await client.sp.getStorageProviders();

    // Choose the first one to be the primary SP
```

```
    const primarySP = sps[0].operatorAddress;
    console.log('primarySP', primarySP);

    setTimeout(() => {
      console.log(`Delaying for ${delayInSeconds} seconds...`);
      process.exit();
    }, 5 * 1000);

    // Get file's expectCheckSums
    const filePath = './test.txt';
    const fileBuffer = fs.readFileSync(filePath);
    const fileType = mimeTypes.lookup(path.extname(filePath));
    const rs = new NodeAdapterReedSolomon();
    const expectCheckSums = await rs.encodeInWorker(__filename,
Uint8Array.from(fileBuffer));
    fs.closeSync(fileBuffer);
    setTimeout(() => {
      console.log(`Delaying for ${delayInSeconds} seconds...`);
      process.exit();
    }, 5 * 1000);

    // Create object transaction
    const createObjectTx = await client.object.createObject(
      {
        bucketName: bucketName,
        objectName: objectName,
        creator: ACCOUNT_ADDRESS,
        visibility: 'VISIBILITY_TYPE_PRIVATE',
        fileType: fileType,
        redundancyType: 'REDUNDANCY_EC_TYPE',
        contentLength: fileBuffer.length,
        expectCheckSums,
      },
      {
        type: 'ECDSA',
        privateKey: ACCOUNT_PRIVATEKEY,
      },
    );
    setTimeout(() => {
      console.log(`Delaying for ${delayInSeconds} seconds...`);
```

```
  process.exit();
}, 5 * 1000);

const createObjectTxSimulateInfo = await createObjectTx.simulate({
  denom: 'BNB',
});

const createObjectTxRes = await createObjectTx.broadcast({
  denom: 'BNB',
  gasLimit: Number(createObjectTxSimulateInfo?.gasLimit),
  gasPrice: createObjectTxSimulateInfo?.gasPrice || '5000000000',
  payer: ACCOUNT_ADDRESS,
  granter: '',
  privateKey: ACCOUNT_PRIVATEKEY,
});

if (createObjectTxRes.code === 0) {
  console.log('create object success');
}

setTimeout(() => {
  console.log(`Delaying for ${delayInSeconds} seconds...`);
  process.exit();
}, 5 * 1000);

// Upload the object
await client.object.uploadObject(
  {
    bucketName: bucketName,
    objectName: objectName,
    body: fileBuffer,
    txnHash: createObjectTxRes.transactionHash,
  },
  {
    type: 'ECDSA',
    privateKey: ACCOUNT_PRIVATEKEY,
  },
);

console.log('Object created and uploaded successfully');
```

```
    res.status(200).json({ success: true, message: 'Object created and
uploaded successfully' });
  } catch (error) {
    console.error('Error:', error.message || error);
    res.status(500).json({ success: false, message: 'Internal server
error' });
  }
};

app.post('/api/createObject', createObject);

app.listen(port, () => {
  console.log(`Server is running on http://localhost:${port}`);
});
```