



Banking Application Capstone Presentation

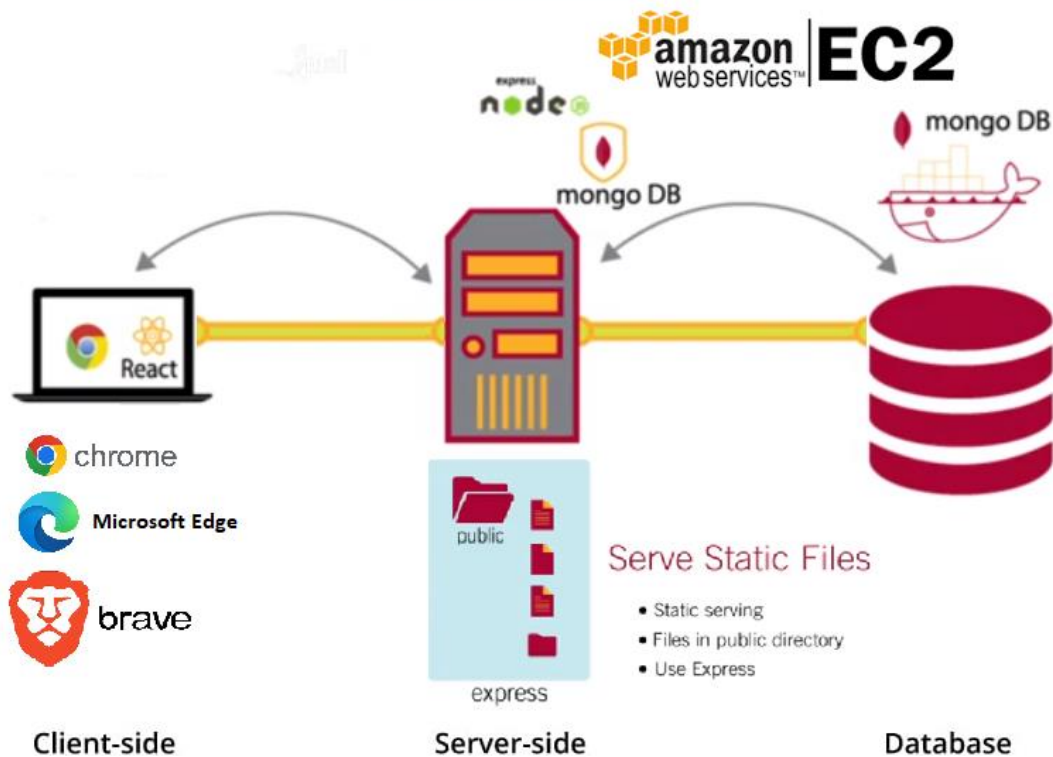
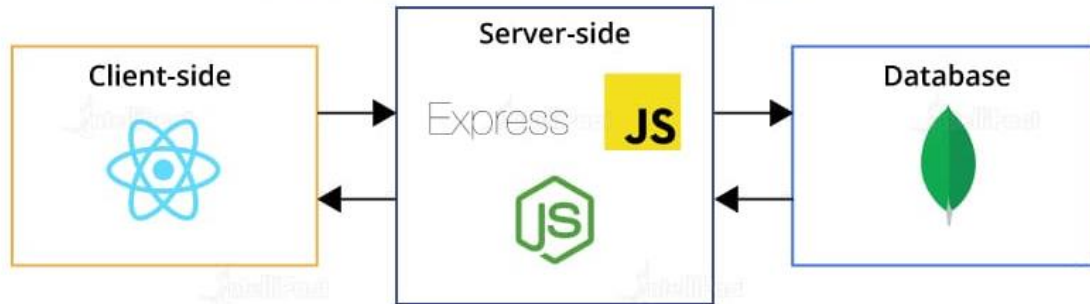
Omar Cerda Villalobos



Part 1: Front-End Architecture, Authentication, And App Diagram

Application Overview Diagram

The 3-tier Architecture of MERN Stack

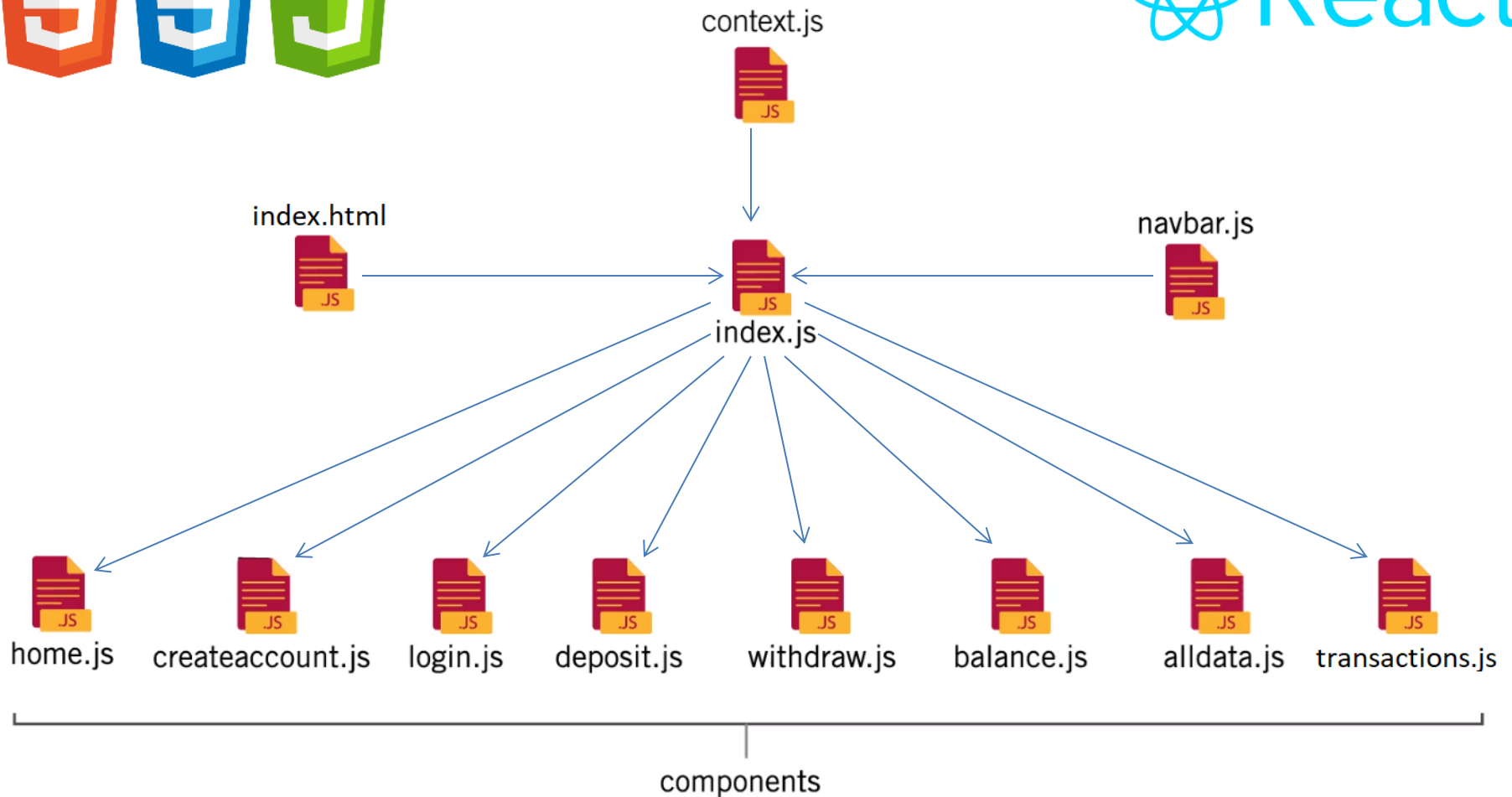


```
MITXPRO_MERN
├── Capstone Option 1_starter
│   ├── build \ public
│   │   ├── alldata.js
│   │   ├── balance.js
│   │   ├── bank_logo.jpg
│   │   ├── context.js
│   │   ├── createaccount.js
│   │   ├── deposit.js
│   │   ├── home.js
│   │   ├── index.html
│   │   ├── index.js
│   │   ├── login.js
│   │   ├── navbar.js
│   │   ├── styles.css
│   │   ├── transactions.js
│   │   ├── withdraw.js
│   │   └── node_modules
│   └── public
│       ├── dal.js
│       ├── index.js
│       ├── mongo_test.js
│       ├── package-lock.json
│       ├── package.json
│       └── us-east-2.pem
```

Front-End Architecture



Front-end Architecture



Authentication

login.js

```
function Login(){
  const [show, setShow] = React.useState(true);
  const [status, setStatus] = React.useState('');

  return (
    <Card
      bgcolor="secondary"
      header="Login"
      status={status}
      body={show ?
        <LoginForm setShow={setShow} setStatus={setStatus}/> :
        <LoginMsg setShow={setShow} setStatus={setStatus}/>
      />
  )
}
```

```
function handle(){
  console.log(email,password);
  if (!validate(email, 'email', props.setStatus)) return;
  if (!validate(password, 'password', props.setStatus)) return;

  fetch('/account/login/${email}/${password}')
    .then(response => response.text())
    .then(text => {
      try {
        const data = JSON.parse(text);
        props.setStatus('');
        props.setShow(false);
        console.log('JSON:', data);
      } catch(err) {
        props.setStatus(text)
        console.log('err:', text);
      }
    });
}
```

```
function LoginMsg(props){
  return(<>
    <h5>Success</h5>
    <button type="submit"
      className="btn btn-light"
      onClick={() => props.setShow(true)}>
      Authenticate again
    </button>
  </>);
}
```



index.js

```
// login user
app.get('/account/login/:email/:password', function (req, res) {

  dal.find(req.params.email).
    then((user) => {

      // if user exists, check password
      if(user.length > 0){
        if (user[0].password === req.params.password){
          res.send(user[0]);
        }
        else{
          res.send('Login failed: Wrong password!');
        }
      }
      else{
        res.send('Login failed: User not found!');
      }
    });
});
```



dal.js

```
// find user account
function find(email){
  return new Promise((resolve, reject) => {
    const customers = db
      .collection('users')
      .find({email: email})
      .toArray(function(err, docs) {
        err ? reject(err) : resolve(docs);
      });
  });
}
```





xPRO



Part 2: Database And API

Database

The image displays a MongoDB Compass interface and a terminal window. The top left shows the MongoDB Compass window for 'localhost:27017/myproject'. It lists two collections: 'transactions' (25 documents, 20.48 KB storage) and 'users' (19 documents, 20.48 KB storage). A blue arrow points from the 'transactions' collection to the right-hand MongoDB Compass window, which shows the 'myproject.transactions' collection with two documents. Another blue arrow points from the 'users' collection to the bottom-right MongoDB Compass window, which shows the 'myproject.users' collection with two documents. The bottom left shows a terminal window with the following commands and output:

```
> .MONGOSH
ec2-user@ip-172-31-30-127:~
[ec2-user@ip-172-31-30-127 ~]$ ls
build dal.js index.js mongo_test.js node_modules package.json package-lock.json
[ec2-user@ip-172-31-30-127 ~]$ mongo
MongoDB shell version v5.0.19
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("695fc852-8797-4687-a9aa-050a6c3a6bce") }
MongoDB server version: 5.0.19

Warning: the "mongo" shell has been superseded by "mongosh",
which delivers improved usability and compatibility. The "mongo" shell has been deprecated and will be removed in
an upcoming release.
For installation instructions, see
https://docs.mongodb.com/mongodb-shell/install/

-----
The server generated these startup warnings when booting:
  2023-07-28T02:21:30.013+00:00: Access control is not enabled for the database. Read and write access to data
and configuration is unrestricted
-----

> show dbs
admin      0.000GB
config     0.000GB
local      0.000GB
myproject  0.000GB

> use myproject
switched to db myproject

> show collections
customers
transactions
users

> db.users.find()
{ "_id" : ObjectId("64c33bf34f26093522da698a"), "name" : "Mike", "email" : "mike@mit.edu", "password" : "12345678", "balance" : 1500 }
{ "_id" : ObjectId("64c33c554f26093522da698c"), "name" : "tom", "email" : "tom@mit.edu", "password" : "12345678", "balance" : 5100 }
{ "_id" : ObjectId("64c33c764f26093522da698d"), "name" : "Ana", "email" : "ana@mit.edu", "password" : "12345678", "balance" : 1500 }
```

The bottom right shows the MongoDB Compass window for 'localhost:27017/myproject.transactions'. It displays two documents in the 'transactions' collection:

```
{ "_id": { "$oid": "64c034ac863f2911b0857292" }, "date": { "$date": "2023-07-25T20:46:36.977Z" }, "email": "alex@gmail.com", "type": "Deposit", "amount": 1250 }
{ "_id": { "$oid": "64c034f2863f2911b0857293" }, "date": { "$date": "2023-07-25T20:47:46.468Z" }, "email": "frank@mit.edu", "type": "Deposit", "amount": 350 }
```

The bottom right also shows the MongoDB Compass window for 'localhost:27017/myproject.users'. It displays two documents in the 'users' collection:

```
{ "_id": { "$oid": "64bbf710eda93a27d0172e2d" }, "name": "peter", "email": "peter@mit.edu", "password": "secret01", "balance": 9159 }
{ "_id": { "$oid": "64bbf742eda93a27d0172e2e" }, "name": "thomas", "email": "thomas@mit.edu", "password": "secret02", "balance": 9439 }
```

API

index.js

```
var express = require('express');
var app = express();
var cors = require('cors');
var dal = require('./dal.js');

// used to serve static files from public directory
app.use(express.static('build/public'));
// app.use(express.static('public'));

app.use(cors());

// create user account
app.get('/account/create/:name/:email/:password', function (req, res) {

  // check if account exists
  dal.find(req.params.email).
    then((users) => {

      // if user exists, return error message
      if(users.length > 0){
        console.log('Error: User already exists');
        res.send({'Error': 1});
      }
      else{
        // else create user
        dal.create(req.params.name, req.params.email
          then((user) => {
            console.log(user);
            res.send(user);
          });
      }
    });

  // login user
  app.get('/account/login/:email/:password', function (req, res) {
```

dal.js

```
const MongoClient = require('mongodb').MongoClient;
const url = 'mongodb://localhost:27017';
//const url = 'mongodb://doadmin:H0p4h9L3T2mo7Z56@myproject-109eca8f.mongo.ondigitalocean.com:27017';
let db = null;

// connect to mongo
MongoClient.connect(url, {useUnifiedTopology: true}, function(err, client) {
  console.log("Connected successfully to db server");

  // connect to myproject database
  db = client.db('myproject');
});

// create user account
function create(name, email, password){
  return new Promise((resolve, reject) => {
    const collection = db.collection('users');
    const doc = {name, email, password, balance: 0};
    collection.insertOne(doc, {w:1}, function(err, result) {
      err ? reject(err) : resolve(doc);
    });
  });
}

// create transaction
function transaction(email, typeTrans, amount){
  return new Promise((resolve, reject) => {
    const collection = db.collection('transactions');
    const dateTime = new Date();
    const doc = {dateTime, email, typeTrans, amount};
    collection.insertOne(doc, {w:1}, function(err, result) {
      err ? reject(err) : resolve(doc);
    });
  });
}

// all users
function all(){
  return new Promise((resolve, reject) => {
    const customers = db
      .collection('users')
      .find({})
      .toArray(function(err, docs) {
        err ? reject(err) : resolve(docs);
      });
  });
}

// transactions
function transactions(){
  return new Promise((resolve, reject) => {
    const customers = db
      .collection('transactions')
      .find({})
      .toArray(function(err, docs) {
        err ? reject(err) : resolve(docs);
      });
  });
}
```

dal.js

```
// find user account
function find(email){
  return new Promise((resolve, reject) => {
    const customers = db
      .collection('users')
      .find({email: email})
      .toArray(function(err, docs) {
        err ? reject(err) : resolve(docs);
      });
  });
}

// find user account
function findOne(email){
  return new Promise((resolve, reject) => {
    const customers = db
      .collection('users')
      .findOne({email: email})
      .then((doc) => resolve(doc))
      .catch((err) => reject(err));
  });
}
```

dal.js

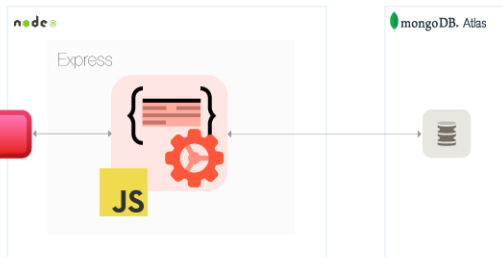
```
// update - deposit/withdraw amount
function update(email, amount){
  return new Promise((resolve, reject) => {
    const customers = db
      .collection('users')
      .updateOne({email: email},
        {balance: { $inc: amount }},
        {w: 1},
        function(err, result) {
          err ? reject(err) : resolve(result);
        });
  });
}

module.exports = {create, transaction, findOne, find, update, all, transactions};
```



{API}

POST /api/1pmom/lp
GET /api/1pmom/lp/{lp_address}
PUT /api/1pmom/lp/{lp_address}
DELETE /api/1pmom/lp/{lp_address}
GET /api/1pmom/lp/show





Part 3: Deployment, Additional Features, App Demonstration, Reflection

Deployment

Additional Features

Record each transaction
(Deposit/Withdraw) in DB.

Transactions

Select User

-- All Users --

#	Date/Time	Email	Type
1	2023-07-28T03:54:57.005Z	mike@mit.edu	Deposit
2	2023-07-28T03:56:59.818Z	tom@mit.edu	Deposit
3	2023-07-28T03:57:08.717Z	ana@mit.edu	Deposit
4	2023-07-28T03:59:01.748Z		
5	2023-07-28T03:59:26.340Z		
6	2023-07-28T03:59:38.292Z		
7	2023-07-28T04:02:30.292Z		
8	2023-07-28T04:02:44.763Z		
9	2023-07-28T04:02:55.654Z		
10	2023-07-28T04:03:23.085Z		
11	2023-07-28T04:09:01.363Z		

Transactions

Select User

john@mit.edu - John

Date/Time

1	2023-07-28T04:02:30.292Z
2	2023-07-28T04:02:55.654Z
3	2023-07-28T04:09:01.363Z

APS Bank International P.L.C. / Development by Int3ltec Group (c) 2023

Contact: int3ltec.group@gmail.com

transactions.js

```
function Transactions() {  
  const [transactionsdata, settransactionsData] = React.useState([]);  
  const [usersdata, setusersData] = React.useState([]);  
  const [selectedUser, setSelectedUser] = React.useState("");  
  
  React.useEffect(() => {  
    // fetch transactions from API  
    fetch('/account/transactions')  
      .then(response => response.json())  
      .then(transactionsdata => {  
        settransactionsData(transactionsdata);  
      });  
  }, []);  
  
  React.useEffect(() => {  
    // fetch users from API  
    fetch('/account/all')  
      .then(response => response.json())  
      .then(usersdata => {  
        setusersData(usersdata);  
      });  
  }, []);  
  
  // Filter transactions based on selected user  
  const filteredTransactions = selectedUser  
    ? transactionsdata.filter(transaction => transaction.email === selectedUser)  
    : transactionsdata;  
  
  return (  
    <div className="container">  
      <h5 className="mb-4">Transactions</h5>  
      <div className="form group">  
        <label htmlFor="userSelect">Select User</label>  
        <select  
          className="form-control"  
          id="userSelect"  
          value={selectedUser}<br/>  
          onChange={e => setSelectedUser(e.currentTarget.value)}>  
          {usersdata.map((user) => <option value={user.email}>{user.email}</option>)}  
        </select>  
      </div>  
      <table>  
        <thead>  
          <tr>  
            <th>#</th>  
            <th>Date/Time</th>  
            <th>Email</th>  
            <th>Type</th>  
          </tr>  
</thead>  
<tbody>  
          {filteredTransactions.map((transaction, index) => <tr>  
            <td>{index + 1}</td>  
            <td>{transaction.date}</td>  
            <td>{transaction.email}</td>  
            <td>{transaction.type}</td>  
          </tr>)}  
        </tbody>  
      </table>  
    </div>  
  );  
}
```

index.js

```
// create transaction  
app.get('/account/transaction/:email/:amount', function (req, res) {  
  const amount = Number(req.params.amount);  
  const typeTransaction = "Deposit";  
  // check the transaction is a withdraw or deposit  
  if (amount < 0) {  
    typeTransaction = "Withdraw";  
  }  
  
  dal.transaction(req.params.email, typeTransaction, amount).  
    then((response) => {  
      console.log(response);  
      res.send(response);  
    });  
});
```

dal.js

```
// create transaction  
function transaction(email, typeTrans, amount){  
  return new Promise((resolve, reject) => {  
    const collection = db.collection('transactions');  
    const dateTime = new Date();  
    const doc = {dateTime, email, typeTrans, amount};  
    collection.insertOne(doc, {w:1}, function(err, result) {  
      err ? reject(err) : resolve(doc);  
    });  
  });  
}
```

dal.js

```
// transactions  
function transactions(){  
  return new Promise((resolve, reject) => {  
    const customers = db  
      .collection('transactions')  
      .find({})  
      .toArray(function(err, docs) {  
        err ? reject(err) : resolve(docs);  
      });  
  });  
}
```

Show the transactions of All Users or
Selected user

Application Demonstration – Create Account, Log In, Deposit, Withdraw

APS Bank International

18.117.87.233:3000/#/CreateAccount/

APS Bank

Create Account

Login

Deposit

Withdraw

Balance

All Data

Transactions

Create Account: Enter name, email and password.

Create Account

Name

Mathew

Email address

mathew@mit.edu

Password

.....

Create Account

Error: Password must be at least 8 characters long

APS Bank International P.L.C. / Development by Int3lltec Group (c) 2023

APS Bank International

18.117.87.233:3000/#/login/

APS Bank

Create Account

Login

Deposit

Withdraw

Balance

All Data

Transactions

Login: Enter email and password to login

Login

Email

tom@mit.edu

Password

.....

Login

Login failed: Wrong password!

APS Bank International P.L.C. / Development by Int3lltec Group (c) 2023

Contact: int3lltec.group@gmail.com

APS Bank International

18.117.87.233:3000/#/deposit/

APS Bank

Create Account

Login

Deposit

Withdraw

Balance

All Data

Transactions

Deposit: Select a user and enter the deposit amount

Deposit

Select User

john@mit.edu - \$11450.00

Deposit Amount

-5000

Deposit

Error: Invalid deposit amount

APS Bank International P.L.C. / Development by Int3lltec Group (c) 2023

Contact: int3lltec.group@gmail.com

APS Bank International

18.117.87.233:3000/#/withdraw/

APS Bank

Create Account

Login

Deposit

Withdraw

Balance

All Data

Transactions

Withdraw: Select a user and enter the withdraw amount

Withdraw

Select User

mike@mit.edu - \$1500.00

Withdraw Amount

1850

Withdraw

Error: Invalid withdraw amount

APS Bank International P.L.C. / Development by Int3lltec Group (c) 2023

Contact: int3lltec.group@gmail.com

Reflection

- If you started the project today, how would you structure your code differently?

If I started the project today, I would structure the code differently by following a more modular and scalable architecture. I would use design patterns such as **MVC (Model-View-Controller)** to clearly separate **business logic, views, and interactions with the database**. Additionally, I would implement the use of **controllers to handle different server routes and actions**. I would also use **TypeScript** to add static typing and improve code maintainability and robustness. Moreover, I would implement **unit and integration tests** to ensure more reliable code.

- If you started the project today, what additional features would you build?

If I started the project today, I would add features such as **two-factor authentication (2FA)** to enhance the security of user accounts. I would also implement real-time notifications so that users receive instant alerts about their transactions. **Another interesting feature would be to allow users to set savings goals and receive notifications when they are close to achieving them**. Additionally, I would consider adding support for **transfers between accounts** and the ability to link external bank accounts for comprehensive financial management.

