user schema

user_id- mongo
name
age ?
email
virtuals (aggregation of cash and credit)

? account schema

account_id - mongo
owner_id - (find from user schema)
cash (default 0) ?
credit (default 0) ?

DB Model

CRUD

create user
update credit (min 0)
update cash (deposit, withdraw)
update credit (transfer)
read a user (user info)
read all users
filter users by cash, by name, by credit
delete user

VALIDATIONS

- can't add existing user
- deal with filter results
- error handling

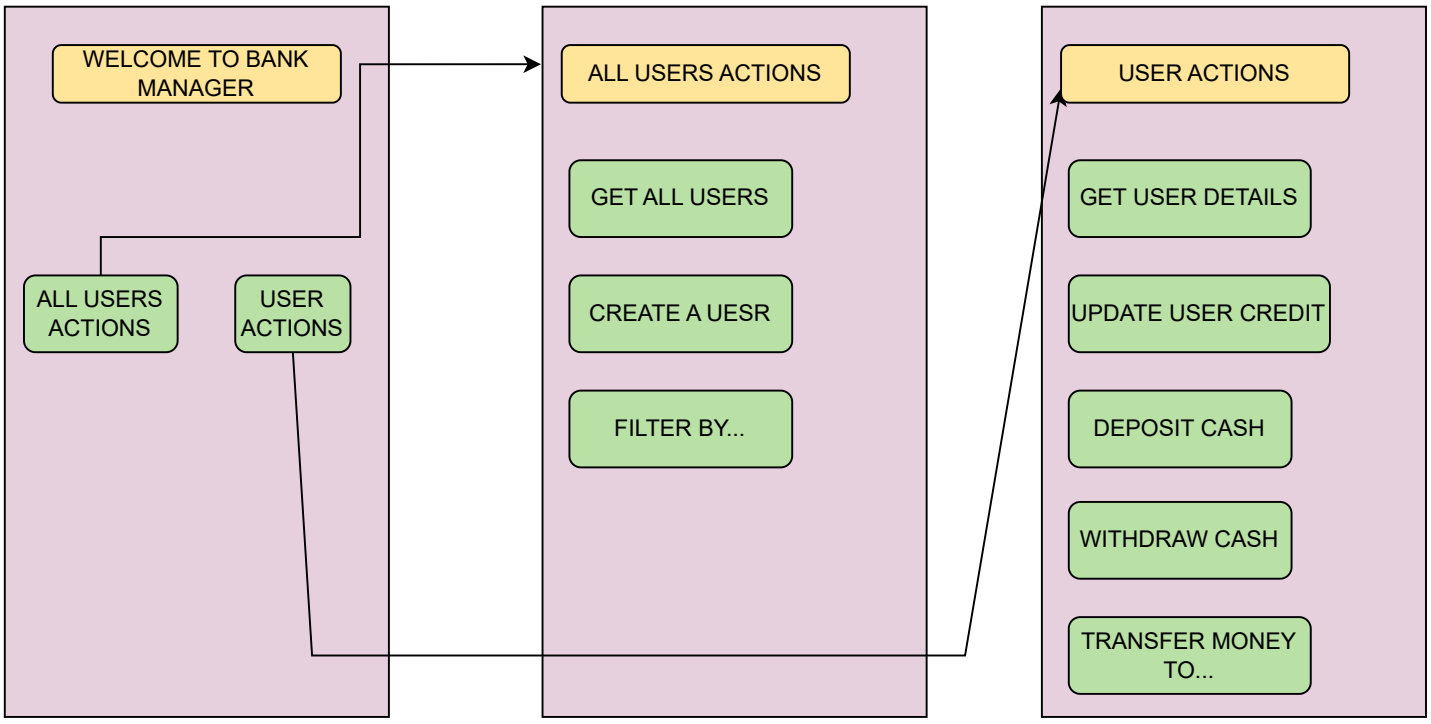
controllers

middleware -
advanced results

API requests

GET ("/" users)
GET ("/:id" user)
POST ("/users", create user)
PUT ("/users/credit/:id")
PUT("users/transfer/:id"), body: user, amount
PUT("users/updateCash/:id" , deposit/withdraw cash)
GET ("/users/findUsersByCash" , query)
GET ("/users/findUsersByName" , query)
GET ("/users/findUsersByCredit" , query)
DELETE

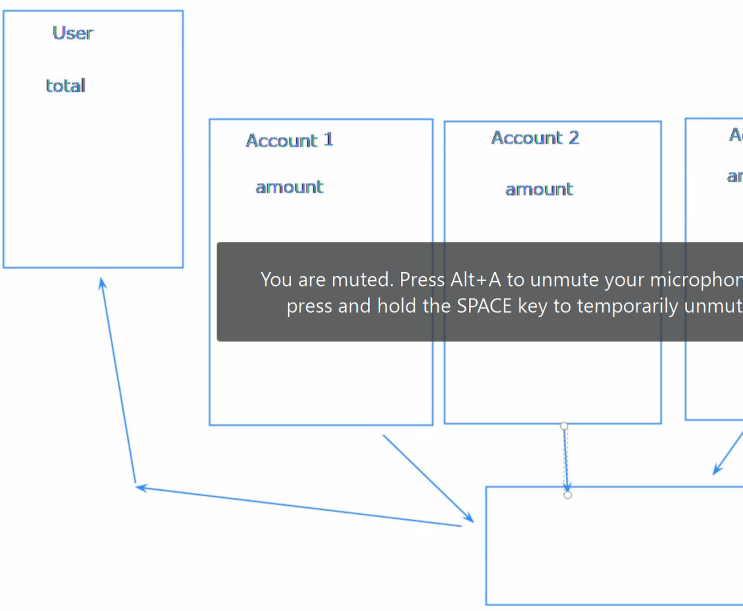
routes



1 day for BACKEND + MONGODB + POSTMAN

technologies:
express, mongoose, nodemon, dotenv, slugify, express-async-handler
cors , concurrently?, vite

0.5 day - refactoring + debugging

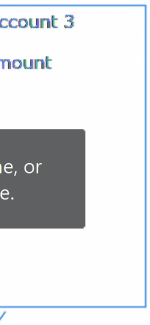


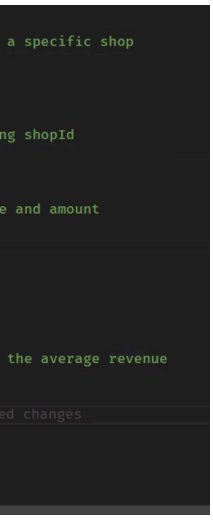

```
58
59  // Static method to get the average revenue (price * amount) of products in
60  ProductSchema.statics.getAverageRevenue = async function (userId) {
61    // Use MongoDB aggregation to calculate the average product revenue
62    const obj = await this.aggregate([
63      {
64        // $match stage filters documents to only include those with a matchi
65        $match: { user: userId }
66      },
67      {
68        // $addFields stage creates a new field 'revenue' by multiplying pric
69        $addFields: {
70          revenue: {
71            $sum: ['$cash', '$credit']
72          }
73        }
74      },
75      {
76        // $group stage groups the documents by the shop field and calculates
77        $group: {
78          _id: '$user',
79          userTotal: { $sum: '$revenue' }      You, 8 seconds ago • Uncommitt
80        }
81      }
82    ]);
83
84    // Update the shop's avgRevenue field with the calculated value
```

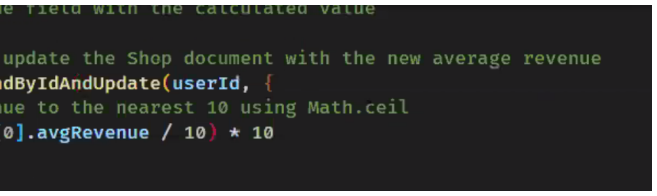
1 day for FRONTEND + cyclic

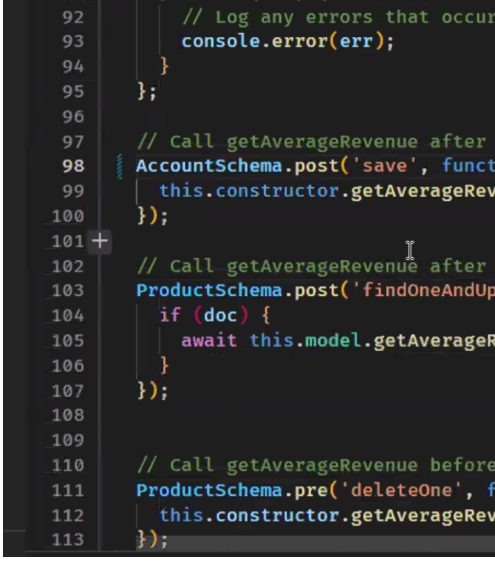
technologies:
axios, some design package

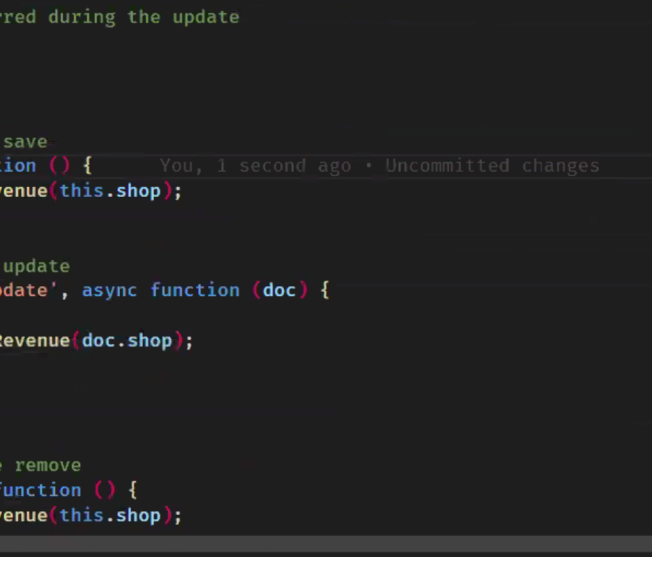
```
84      // Update the shop's avgRevenu
85      try {
86        // Use findByIdAndUpdate to
87        await this.model('Shop').fin
88          // Round the average reven
89          avgRevenue: Math.ceil(obj[
90        });
91      } catch (err) {
```


ccount 3
mount
e, or
e.


a specific shop
ng shopId
e and amount
the average revenue
ed changes


update the Shop document with the new average revenue
dByIdAndUpdate(userId, {
ue to the nearest 10 using Math.ceil
0].avgRevenue / 10) * 10


// Log any errors that occur
console.error(err);
}
};
// Call getAverageRevenue after
AccountSchema.post('save', funct
this.constructor.getAverageRev
});
// Call getAverageRevenue after
ProductSchema.post('findOneAndUp
if (doc) {
await this.model.getAverageR
}
});
// Call getAverageRevenue before
ProductSchema.pre('deleteOne', f
this.constructor.getAverageRev
});


rred during the update
save
ion () {
You, 1 second ago • Uncommitted changes
venue(this.shop);
update
date', async function (doc) {
Revenue(doc.shop);
remove
function () {
venue(this.shop);