Final Report for MathApp 2023 Spring

1. Summary

Based on the client requirements and the legacy, our client seeks an online bookstore to sell his mathematical textbooks to users. The bookstore website includes a sign up and login pages for users, the general functions of purchasing with Paypal and credit card. And the website should provide a page for users to read their purchased book content. The client also wanted to edit the purchase code and email subscription, view all transactions and users. But when we first checked the legacy, it had a lot of unimplemented features, and the backend server service couldn't even work.

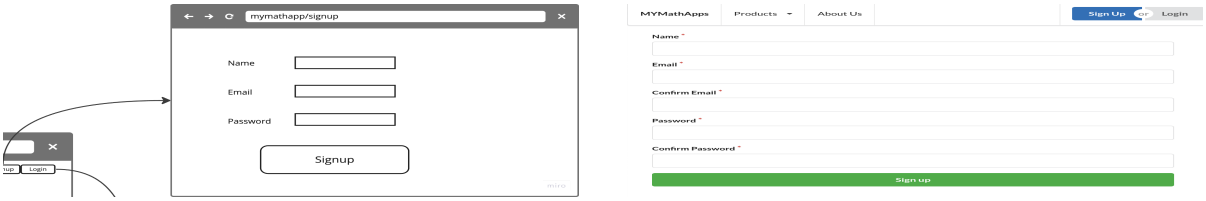
Thus, to rebuild the whole project, we only kept the basic framework of the legacy which is React js and Nest js. We created several new web pages based on the React legacy project, and rebuilt a totally new backend Nest js project with a new MySQL database. The functions implemented include login and sign up with email verification, purchase with Paypal or credit card, read the textbook content, admin page to manage customer information, transaction, purchase code, email subscription ets. We could confirm that the website is now operational after overall and systematic testing.

2. User Stories

Feature1: Sign up the website:（Point : 1）

User Story	**Implementation Status**
As a customer, I want to sign up the website So that I can create my own account using email I want to use my own account to access this website and maintain my information.	All completed

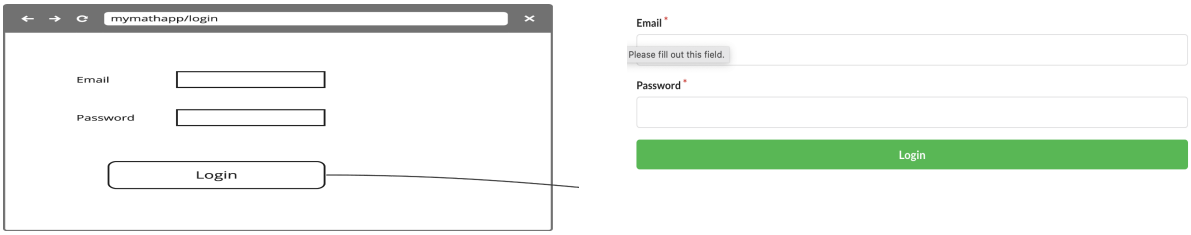
Lofi → Website



Feature2: Login in the website :（Point : 1）

User Story	Implementation Status
As a user, I want to be able to login to the website So that I can access my personal account information and perform various actions. I expect to see a login page with fields for me to enter my email address and password.	All completed

Lofi → Website



Feature3: Activate Account :（Point : 2）

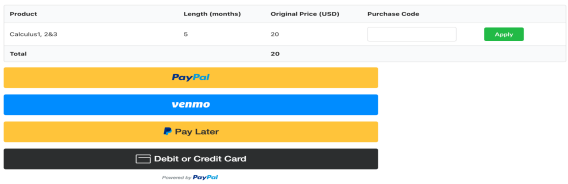
User Story	Implementation Status
As a user signing up for the website,I want my email to be verified as valid to get confirmation. So that I can ensure that I am receiving important notifications and updates from the website.	All completed

I expect to see a unique verification link that I can click to verify my email address after signup.	

Feature4: **General Purchase with Paypal or credit card :**（Point : 3）

User Story	**Implementation Status**
As a future customer, I want to be able to purchase item So that I can choose a plan for certain months and make payment for it with Paypal or credit card I want to purchase the access to the textbooks for a period of time with Paypal or credit card	All completed

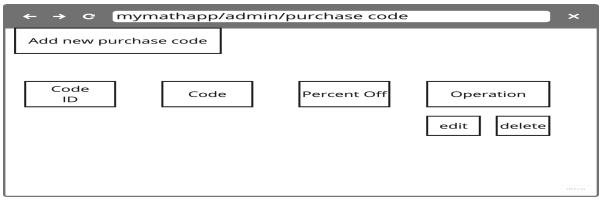
Lofi → Website

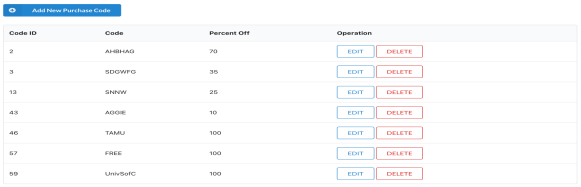


Feature 5: **Add purchase code:**（Point : 2）

User Story	**Implementation Status**
As the seller, I want to have a page to add, edit and delete purchase code. So that I can provide discount code to specific customer with name and percent off number I want to offer some specific customers a discount when they pay for the items.	All completed

Lofi → Website





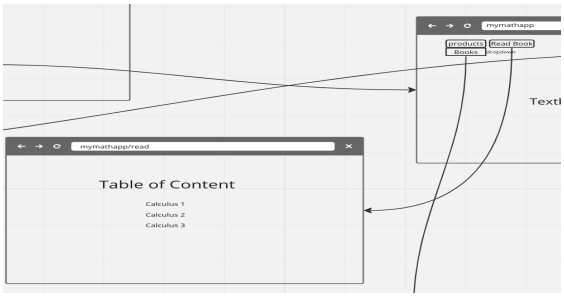
Feature 6: **Use verification email:**（Point : 2）

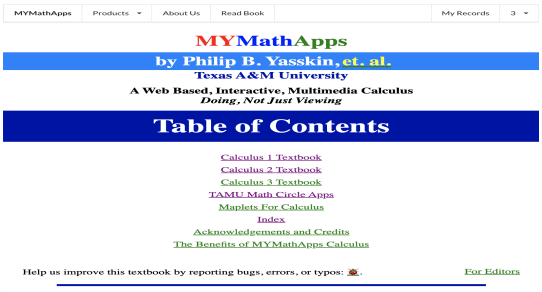
User Story	**Implementation Status**
As a user from an authorized organization who has already purchased the book for its member. So that I can gain access to the textbooks without making payment. I want to use my verification email to bypass the payment and read the textbook directly.	All completed

Feature 7: **Access purchased textbook:**（Point : 3）

User Story	**Implementation Status**
As a current user, I want to be able to access textbooks. So that I can read the textbook on this website that I purchased. I want to be recognized as a user who has purchased the textbooks and be able to read them before expiration.	All completed

Lofi → Website

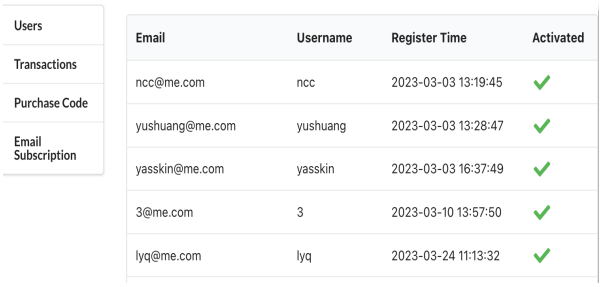


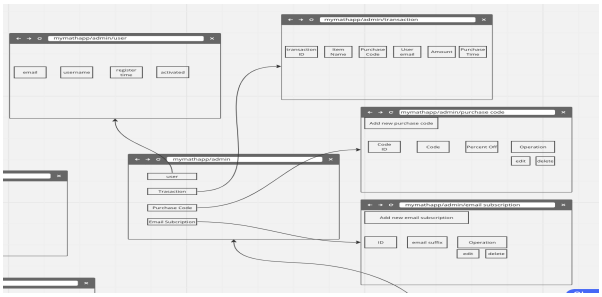


Feature8: **Admin Page to manage customer information:**（Point : 2）

User Story	Implementation Status
As the seller, I want a separate admin page. So that I can manage the users' information including accounts, transactions and record to the textbooks. I want to be able to view all the users that have purchased the book and the transaction history.	All completed

Lofi → Website



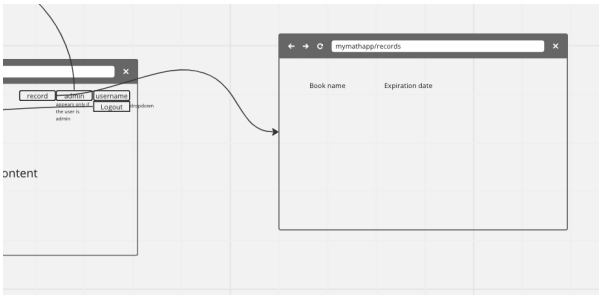


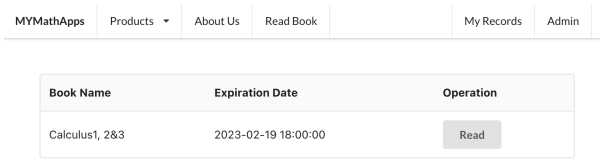
Feature9: **Customer Record Page**（Point : 1）

User Story	Implementation Status

As a customer, I want to be able to view my purchase history and the expiration dates of my purchases. So that I can keep track of what I have bought and when it will expire. I expect to see a "My Record" page or section on the website where I can access my purchase history and expiration dates.	All completed

Lofi → Website

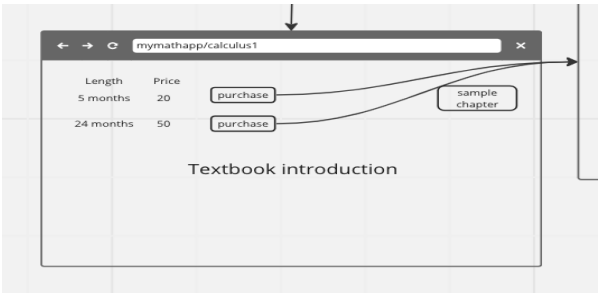


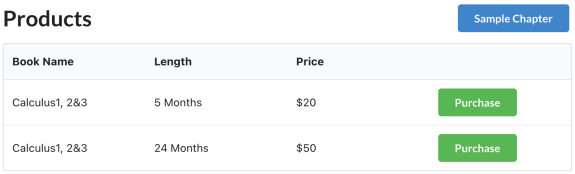


Feature10: Price Page to checkout（Point : 1）

User Story	**Implementation Status**
As a customer of the website, I want a price page. So that I can select and purchase products easily from the price/product page to the checkout page. I expect to see a list of available products with details such as the product name, description, price…	All completed

Lofi → Website

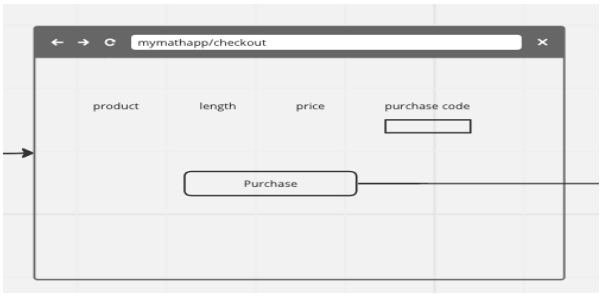


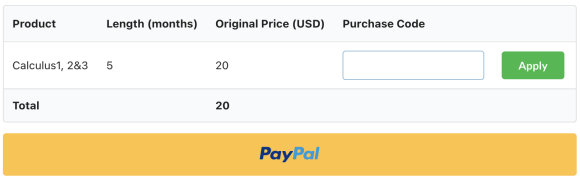


Feature11: **Apply purchase code**（Point : 2）

User Story	**Implementation Status**
As a customer of the website, I want to be able to apply a purchase code to my order. So that I can receive any applicable discounts or promotions. When I am on the checkout page, I expect to see a field labeled "Purchase Code" where I can enter the code I have received. After I enter the code, I expect to see a message confirming that the code has been applied to my order.	All completed

Lofi → Website





Feature12: API Authentication:（Point : 3）

User Story	Implementation Status
As the website maintainer, I want to add API authentication for each website So that a visitor user can not access some websites which require the user to login in. Any normal user can not access the admin pages if they are not admin. Any request directly sent to the back server API can't get a response. When users visit a website which they do not have access to, they will be denied and redirected to the main/error page. When a malicious user sends a request to server API, it will get an unauthorized error prompt.	All completed

3. Legacy Project

The legacy project has two branches, the first branch has a very long history from 2 years ago. It is a very complicated NestJS back-end server but has many dependency problems and is even unable to start.
The second branch was developed by the team last semester but is very simple. They only implement one feature - login. They don't have any other contribution for this project. Therefore in iteration 0, after analyzing the legacy code, we decided to keep the basic structure from the first branch and totally drop the second branch, but designed and implemented all the features by ourselves. Also, we modified the database from MariaDB to MySQL and also removed and combined redundant entitles based on our understanding of this project.

4. Team Role

Product owner: Shuyu Wang
Scrum master: Cheng Niu
Members: Zhiting Zhao, Shuang Yu, Yongqing Liang, Yun Du

5. Accomplished Features in Iterations

In iteration 0, we analyzed the legacy code and set up the environment.
In iteration 1, we finished signing up and login, worth 2 points.
In iteration 2, we finished accessing the purchased textbook and API authentication, worth 6 points.
In iteration 3, we finished the checkout page, adding purchase code, and admin pages, worth 5 points.
In iteration 4, we finished applying the purchase code, record pages, and paypal purchase , worth 6 points.
In iteration 5, we finished activating accounts and using verification email, worth 4 points.

6. Customer Meeting

Customer meeting date/time/place: Every Friday: 4:00pm-5:00pm; Online
During customer meetings, we showed what we completed with live demos and what we plan to do next week. The customer gives his feedback during each meeting including appreciation and satisfaction for our work, and some improving suggestions for the products.

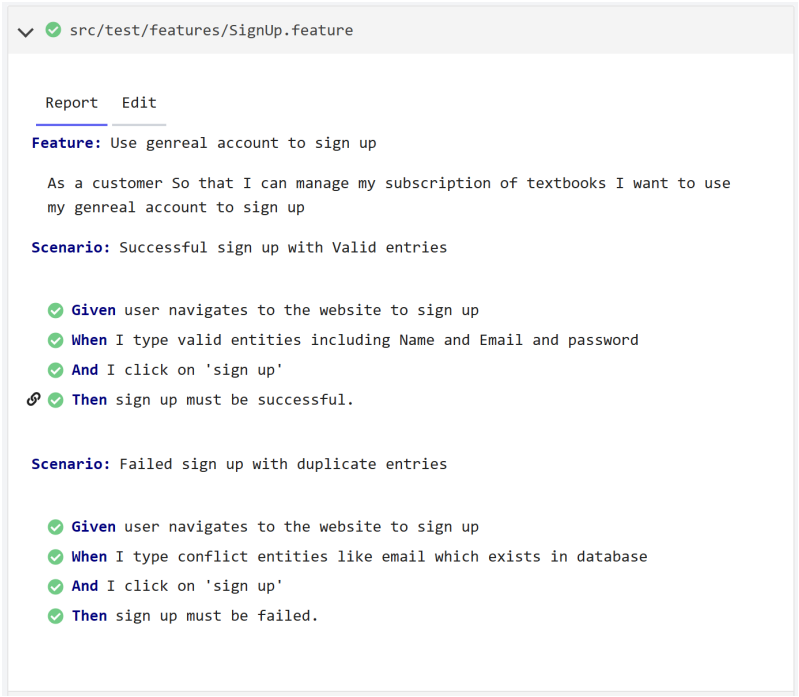
7. Test Result

a. Frontend: Use cucumber, design scenarios to simulate the operations
    i.

```
  Scenario: Successful Login with Valid entries # src/test/features/googleLogin.feature:13
 The ChromeDriver could not be found on the current PATH, trying Selenium Manager
 The ChromeDriver could not be found on the current PATH, trying Selenium Manager
    Given user navigates to the website lo login
    When I type 'yushuang@me.com' and 'yushuang' as email and password
    And I click on 'login'
    Then login must be successful.
http://localhost:3000/

2 scenarios (2 passed)
7 steps (7 passed)
```

    ii.


src/test/features/SignUp.feature
Report
Edit
Feature: Use genreal account to sign up
As a customer So that I can manage my subscription of textbooks I want to use my genreal account to sign up
Scenario: Successful sign up with Valid entries
Given user navigates to the website to sign up
When I type valid entities including Name and Email and password
And I click on 'sign up'
Then sign up must be successful.
Scenario: Failed sign up with duplicate entries
Given user navigates to the website to sign up
When I type conflict entities like email which exists in database
And I click on 'sign up'
Then sign up must be failed.

iii.

src/test/features/Login.feature

Report Edit

Feature: Use genreal account to login

As a customer So that I can manage my subscription of textbooks I want to use my genreal account to login

Scenario: Successful Login with Valid entries

Given user navigates to the website to login
Given user navigates to the website to login
When I type 'yushuang@me.com' and 'yushuang' as email and password
And I click on 'login'
Then login must be successful.

Scenario: Unsuccessful Login with Invalid Password

Given user navigates to the website to login
When I type 'yushuang@me.com' and 'yushuang123' as email and password
And I click on 'login'
Then unsuccessful login due to an invalid password.

Scenario: Unsuccessful Login with Invalid Email

Given user navigates to the website to login
When I type 'ddddddddd@me.com' and 'yushuang' as email and password
And I click on 'login'
Then unsuccessful login due to an invalid email.

iv.

src/test/features/accessAdminPage.feature

Report Edit

Feature: Access Admin Page

As the seller So that I can access the user transaction purchase code in my admin page I want to manage the customer info

Scenario: Failed accessing admin page as general user

Given user is in home page
When not login as admin
And go to the admin url
Then the user will be redirect to the home page.

v.

src/test/features/records.feature

Report Edit

Feature: Access Records Page

As the user So that I can check my records and expiration date in record page I want to manage the customer info

Scenario: Failed accessing record page as not login

- **Given** user is in main page
- **When** not login
- **And** go to the record url
- **Then** the user will be redirect to main page.

Scenario: Successfully accessing record page as a user

- **Given** user is in main page
- **When** login as user
- **And** go to the record url
- **Then** the user will see the record table.

vi.

src\test\features\TransactionPage.feature

Report Edit

Feature: Access Admin Page

As the seller So that I can access the user transaction purchase code in my admin page I want to manage the customer info

Scenario: Successful accessing transaction page as admin user

- **Given** user is at home page
- **When** logged in as admin
- **And** go to the transaction url
- **Then** the user should be at transaction page and see info.

Scenario: Failed accessing transaction page as general user

- **Given** user is at home page
- **When** not logged in as admin
- **And** go to the transaction url
- **Then** the user will not see any data.

vii.

Scenario: Unsuccessful Login with unconfirmed email

- **Given** user sign up and navigates to the website to login without confirming
- **When** I type in email and password
- **And** I click on 'login'
- **Then** unsuccessful login due to an email address unconfirmed

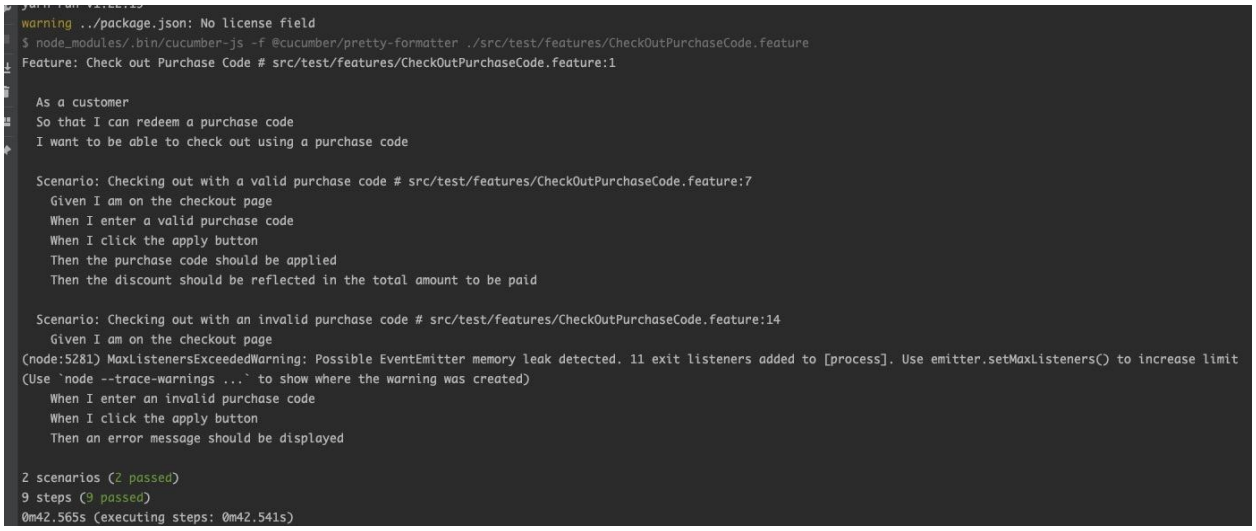
Scenario: Successful activation

- **Given** user sign up
- **When** user go to the activate url
- **Then** account is activated in activate link

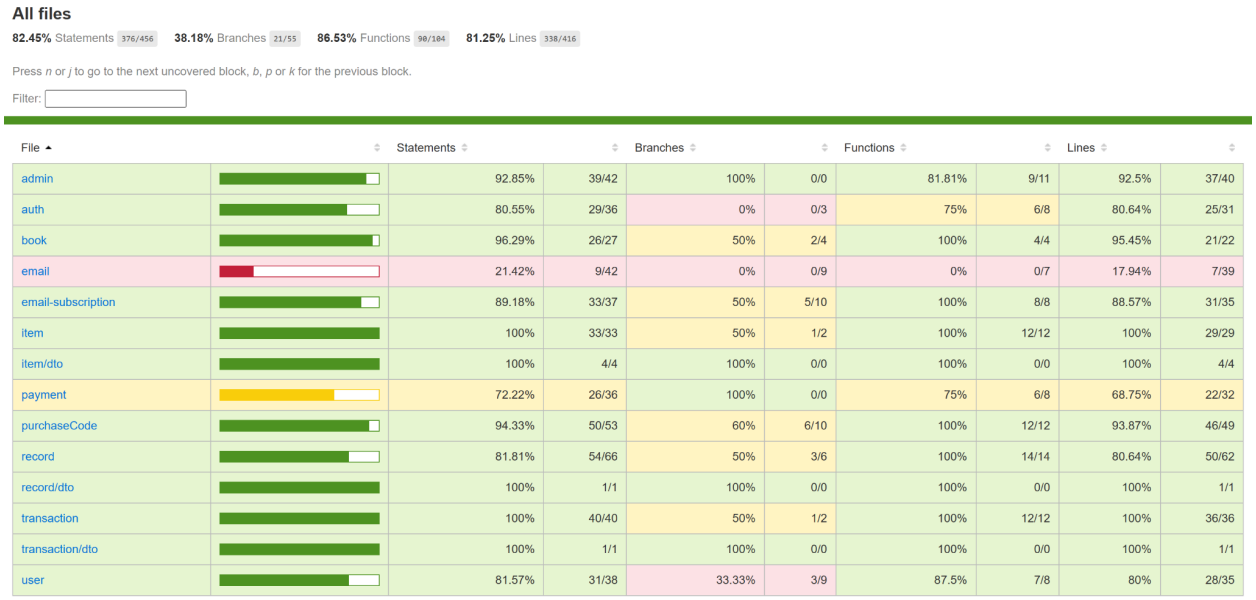
Scenario: Unsuccessful activation due to wrong url

- **Given** user sign up
- **When** user go to wrong activate url
- **Then** account is not activated in activate link

viii.



b. Backend: use jest to test and give out the coverage



All files

82.45% Statements 376/456 **38.18%** Branches 21/55 **86.53%** Functions 90/104 **81.25%** Lines 338/416

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

Filter:

File		Statements		Branches		Functions		Lines	
admin		92.85%	39/42	100%	0/0	81.81%	9/11	92.5%	37/40
auth		80.55%	29/36	0%	0/3	75%	6/8	80.64%	25/31
book		96.29%	26/27	50%	2/4	100%	4/4	95.45%	21/22
email		21.42%	9/42	0%	0/9	0%	0/7	17.94%	7/39
email-subscription		89.18%	33/37	50%	5/10	100%	8/8	88.57%	31/35
item		100%	33/33	50%	1/2	100%	12/12	100%	29/29
item/dto		100%	4/4	100%	0/0	100%	0/0	100%	4/4
payment		72.22%	26/36	100%	0/0	75%	6/8	68.75%	22/32
purchaseCode		94.33%	50/53	60%	6/10	100%	12/12	93.87%	46/49
record		81.81%	54/66	50%	3/6	100%	14/14	80.64%	50/62
record/dto		100%	1/1	100%	0/0	100%	0/0	100%	1/1
transaction		100%	40/40	50%	1/2	100%	12/12	100%	36/36
transaction/dto		100%	1/1	100%	0/0	100%	0/0	100%	1/1
user		81.57%	31/38	33.33%	3/9	87.5%	7/8	80%	28/35

8. TDD Process

For every iteration, we first decide on which feature we are going to tackle, and arrange them by priority. The team then analyzed and decided on how each feature will be implemented. Afterwards, the team generates the basic unit tests for each feature, and the rest of the team develops the features based on the tests. A benefit of the process is that it's rather easy to control the code to its basic necessities, hence reducing the redundancy.

However, there are some problems, for example, the front end cucumber test can be really dependent on the detailed implementation, hence requiring some back and forth change. Also, there often will be some unforeseen detail changes in actual implementation of the feature, which then requires changes in the testing process, and could lag the progress if serious.

9. Configuration Management Approach

a. Using git for version control

   We use branches and tags to manage different versions. We maintain a main branch for iteration release, and two branches: deploy-heroku, deploy-dev3, for different kinds of deployment. Also we checkout a new branch when our team members start to develop a new feature. After they finish, a new test feature will be created to run the corresponding feature, and all branches will be merged into the main feature after the test passes.

We also use different tags to mark some important version for each iteration and deploy.

b. Managing environment variables
   The different configuration files can be used to configure the variables required by the application in different environments, such as database connection strings, keys, etc. We have different environments for the application: development, test, deploy to Heroku, deploy to Client's server. In NestJS, we use the config module to manage environment variables, while in React, we use the dotenv module.
c. Managing configuration files
   We only add the .env.example configuration files to the public Git repository, the other sensitive files were transported to other team members after encrypt using RSA public keys.

10.Issues with Heroku

The problem for deploying to Heroku is that our project is not like other teams. Most of their backend servers were written by Ruby on Rail, which are responsible for both logic control and the website rendering of the user interface.
However, our project is a front-end and back-end separate project, which means we must find a way to combine them and deploy them into one Heroku project. Also, our project includes many static html website files as the math textbook, we must include them in the Heroku project too.
Through the tough exploration for a long time, we eventually understand that we need to write a script to separately build the front-end and backend projects, and move all front-end production html files into backend's public folder as static page files. And redefine the route proxy for the backend and the server API address for the frontend project. Also we need to maintain another .env.heroku file to keep the essential password for mail service and database. But this .env.heroku must not be stored in other branches and uploaded to other git repositories excluding the heroku repository.
We also replaced the Yarn package manager to NPM package since the Yarn needs to use some yarn cache which rises some compatible, and write a

11. Issues with GitHub

GitHub is our main tool during the process. Generally it is very useful in managing the project, monitoring if any problem occurs, and controlling the risk. However, it does take some time to get used to. The merging before iterations can be a headache, since different developed parts might be dependent on each other. Sometimes wrong merging or rebase can accidentally occur, which requires revisit and separation of the merged files.

Also, to remove the .env file which restore the sensitive information like password of the database, we use the useful tool called BFG Repo-Cleaner to enormously clean these sensitive files in history commit.

12. Other tools

1. Frontend - React
2. Frontend UI framework - Semantic UI React
3. Backend - Nest.js
4. Package Manager - Yarn
5. Commerce payment solution -PayPal Business
6. Test - Cucumber & Jest
7. Sensitive configuration files clean - BFG Repo-Cleaner
8. Configuration files in different environments - Dotenv & Cross-env
9. Weekly meeting - Zoom
10. Synchronize documents - Google Drive
11. Build docker image and deploy to client's server - Docker, SSH
12. RSA public key create, encryption and decryption: GnuPG

13. Deploy

1. Deploy to Heroku.
    a. Establish a package.json file into the root folder to let the Heroku recognize it is a node project.
    b. Write a postinstall script which lets the client and server use npm to install dependency and build front end files and backend js, then move the product into the server public static folder.
    c. Establish a Procfile file into the root folder and write a start script in Package.json to let Heroku start the backend server.
2. Deploy to client's server.
    a. Write a .env file in the front end's configuration file to define the backend API address.
    b. Build front end production. Upload to client's server.
    c. Build backend docker image. Upload and deploy to the docker of the client's server.

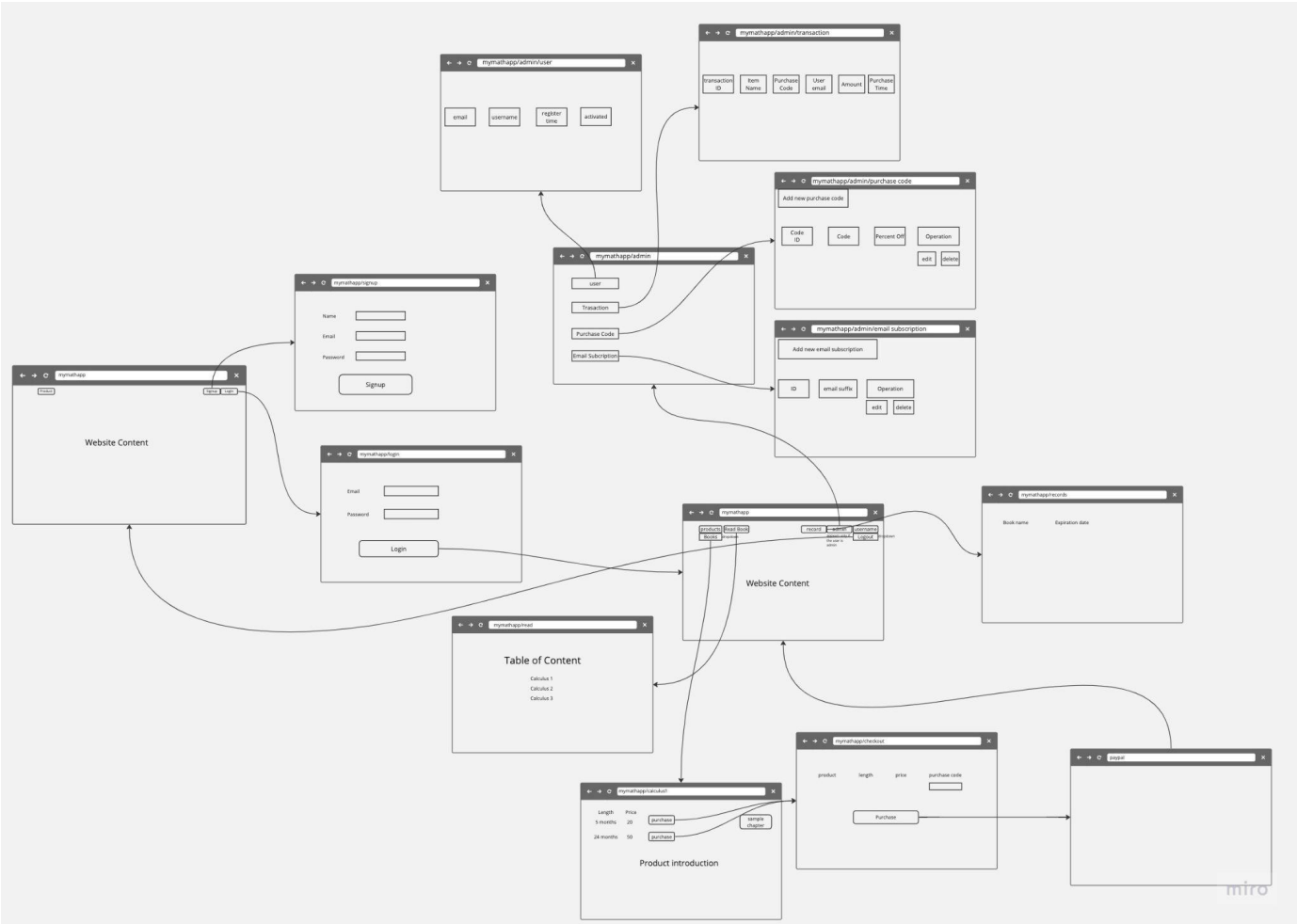
14. Links

1. Pivotal Tracker: https://www.pivotaltracker.com/n/projects/2639605
2. GitHub: https://github.com/luminous-Nc/MyMathAppSpring2023
3. Heroku deployment: https://mymathapp2023spring.herokuapp.com
4. Client Server deployment: https://dev3.mymathapps.com

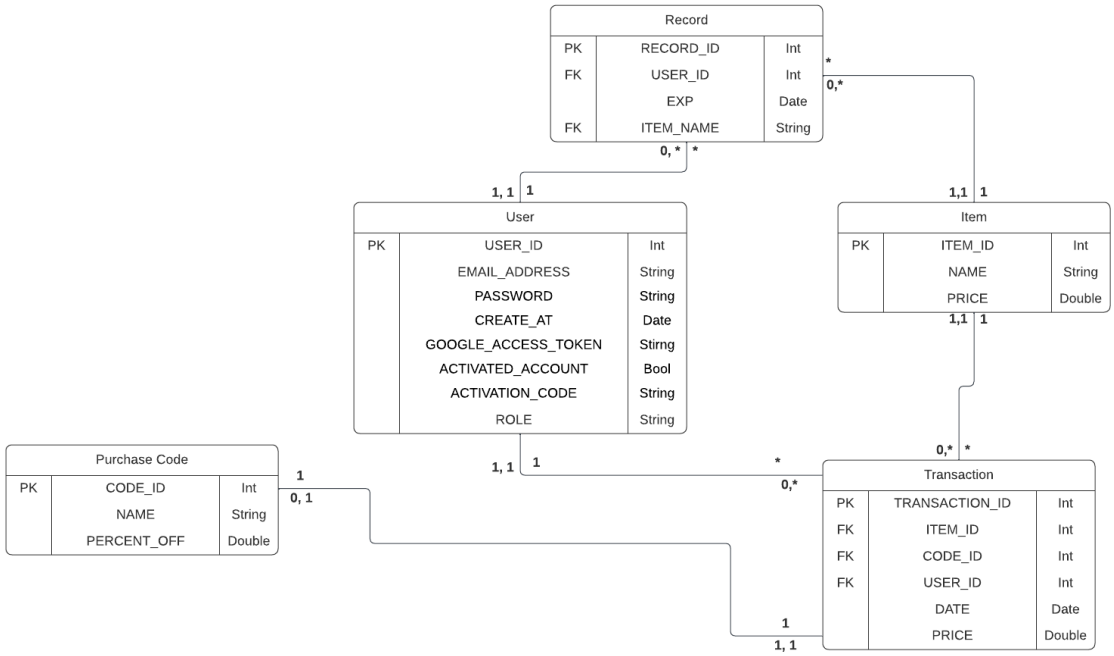
5. Presentation video : https://youtu.be/pMi2ZnE9CQ4
6. Demo video: https://youtu.be/GhCNW_nk3JQ
7. Presentation with demo: https://youtu.be/hDxpx81Zkts

15. Appendix

1. Lo-fi Design



2. Database Design


Record
PK RECORD_ID Int
FK USER_ID Int
EXP Date
FK ITEM_NAME String
User
PK USER_ID Int
EMAIL_ADDRESS String
PASSWORD String
CREATE_AT Date
GOOGLE_ACCESS_TOKEN Stirng
ACTIVATED_ACCOUNT Bool
ACTIVATION_CODE String
ROLE String
Item
PK ITEM_ID Int
NAME String
PRICE Double
Purchase Code
PK CODE_ID Int
NAME String
PERCENT_OFF Double
Transaction
PK TRANSACTION_ID Int
FK ITEM_ID Int
FK CODE_ID Int
FK USER_ID Int
DATE Date
PRICE Double
*
0,*
0, *
*
1, 1
1
1,1
1
1,1
1
0,*
*
1, 1
1
*
0,*
1
0, 1
1
1, 1