**CS673 Software Engineering**

**Team Five - NoteAnt**

**Software Test Document**

| Team Member | Role(s) | Signature | Date |
| --- | --- | --- | --- |
| Frankie | Requirement leader | *Guancheng Huang* | 9 Dec |
| Chris | Design and Implementation leader | Wenhao Tian (Chris) | 9 Dec |
| Nicholas | Team Leader | *Nicholas Narmada* | 9 Dec |
| Wayne | Configuration leader | *Yichen Li* | 9 Dec |
| Siyuan Wan | Security Leader | *Siyuan Wan* | 9 Dec |
| Yibo(Wilbur) | QA Leader | *Yibo Wang* | 9 Dec |
|  |  |  |  |
|  |  |  |  |

**Revision history**

| **Version** | **Author** | **Date** | **Change** |
| --- | --- | --- | --- |
| **Iter3** | **Siyuan Wan** | **9 Dec** | **First** |
|  |  |  |  |

[Testing Summary](#_sm5odwyvuk3j)

[Manuel Tests Reports](#_pqso2mbjyzx4)

[Automated Testing Reports](#_mtfbusfb0eq3)

[Testing Metrics](#_rijyjeu2ojqa)

[References](#_15tmymhipvdv)

[Glossary](#_8n34lvocupub)

# Testing Summary

# **In our project, we separate it to frontend test and backend test.**

**For frontend test**:

# Wayne is responsible for login page testing

# Siyuan is responsible for homepage testing

# Yibo is responsible for document page testing.

# **We use Cypress end to end method for frontend testing.**

# For LoginPage, Wayne tested whether can be logged in successfully.

# For HomePage, we set 6 methods. For create button, whether can be created successfully by click it or press ”enter”. For delete button, when click it should be delete doc successfully. For show summary button, we should see details successfully. For double click document card, we should make sure it can be navigated to doc page successfully. For the sign out button, when user want to switch account, sign out and login again.

# For DocPage, we set 4 methods. We want to edit content in text editor. For “save” button, we make sure that Color change from blue(Save) to green(Saved)and back to blue(Save) after 5 second. For generate button, when click it, we should make sure tips can be generated successfully. For back button, click it and can be back to homepage.

# **For backend test:**

**We use Mocha for backend unit testing**

# For loginPage, Frankie tested login method with mocked google user information, the following points were checked:

* 1. If user is new, create User in DB then return an access token
  2. If user is not new, return an access token
  3. If google token is invalid, return error

1. For middleware, Chris tested it with mock access token
   1. If token is valid, decode token and store User in req.user, pass to next()
   2. If token is invalid, return 401
   3. If token is missing, return 403
2. For User page, Chris tested it with mock user\_id, doc\_id
   1. getUser()
      1. Successfully return User and user’s docs
      2. Return error if unsuccessful
   2. createDoc()
      1. Successfully create new Doc, then return new Doc id
      2. Return error if unsuccessful
   3. deleteDoc()
      1. Successfully delete doc from mongoDB
      2. Return error if unsuccessful
3. For Document Page, Nicholas tested it with mock Doc and User
   1. getDoc()
      1. Successfully extract Doc with given doc\_id and user\_id
      2. Return error if not found
   2. updateContent()
      1. Successfully found and update Doc with given doc\_id and user\_id
      2. Return error if not found
   3. updateSummary()
      1. Successfully found doc with given doc\_id and user\_id
      2. Successfully request and receive response from OpenAI
      3. Successfully update the Doc summary
      4. Return error if not found

# Manual Testing Report

* **For frontend test**
* Test case 1, Google Login
* New or old: New
* Test items: Use accessToken and can be logged in to homepage
* Test priority: high
* Dependencies (to other test case/requirement if any): Important for all follow steps
* Preconditions: None
* input data: Google email account
* Test steps: Click it by google account
* Postconditions: None
* Expected output: Logged in successfully
* Actual output: Logged in successfully
* Pass or Fail: Pass
* Bug id/link: (this should link to your github issue id)
* Additional notes: None
* Test case 2, create new doc
* New or old: New
* Test items: Click Crete button and we can create a new document
* Test priority High
* Dependencies (to other test case/requirement if any): Based on log in
* Preconditions: Need login first
* input data: title
* Test steps: click “create” button -> input title -> click “create new” or press “enter” -> to doc page
* Postconditions: None
* Expected output: new doc card showed on dashboard and navigated to doc page
* Actual output: new doc card showed on dashboard and navigated to doc page
* Pass or Fail: Pass
* Bug id/link: (this should link to your github issue id)
* Additional notes:None
* Test case 3, delete doc
* New or old: New
* Test items: Click delete button and delete doc card
* Test priority: Medium
* Dependencies (to other test case/requirement if any): Should create doc first
* Preconditions: login and create doc
* input data: none
* Test steps: Click “delete” button -> this card will be disappeared
* Postconditions: none
* Expected output: Card is be deleted
* Actual output: Card id be deleted
* Pass or Fail: Pass
* Bug id/link: (this should link to your github issue id)
* Additional notes:none
* Test case 4, show summary of doc
* New or old: New
* Test items: Click “show summary button” and see summary in modal
* Test priority: Low
* Dependencies (to other test case/requirement if any): create doc
* Preconditions: create doc first
* input data: none
* Test steps: Click “show summary” button -> see summary
* Postconditions:none
* Expected output: see
* Actual output: see
* Pass or Fail: Pass
* Bug id/link: (this should link to your github issue id)
* Additional notes:none
* Test case 5, double click card
* New or old: New
* Test items: Double click card and navigated to doc page
* Test priority: low
* Dependencies (to other test case/requirement if any): create doc
* Preconditions: create doc
* input data: none
* Test steps: double click -> to doc page
* Postconditions:none
* Expected output: success to doc page
* Actual output: success to doc page
* Pass or Fail: Pass
* Bug id/link: (this should link to your github issue id)
* Additional notes:none
* Test case 6, sign out function
* New or old: New
* Test items: click “sign out” to login page
* Test priority: Low
* Dependencies (to other test case/requirement if any): login
* Preconditions: Login
* input data: None
* Test steps: click ”sign out” -> to login page
* Postconditions:none
* Expected output:back successfully
* Actual output: back successfully
* Pass or Fail: pass
* Bug id/link: (this should link to your github issue id)
* Additional notes:none
* Test case 7, text editor
* New or old:New
* Test items: We can type content in text editor area
* Test priority: High
* Dependencies : Create new doc and nav to doc page
* Preconditions: Create new doc and nav to doc page
* input data: every thing you want
* Test steps: type everything you want
* Postconditions: none
* Expected output: type successfully
* Actual output: type successfully
* Pass or Fail: pass
* Bug id/link: none
* Additional notes:none
* Test case 8, save content
* New or old: New
* Test items: Color change from blue(Save) to green(Saved)and back to blue(Save) after 5 second
* Test priority: high
* Dependencies: create new doc
* Preconditions: create new doc
* input data: none
* Test steps: type something -> click save -> saved
* Postconditions: none
* Expected output: none
* Actual output:none
* Pass or Fail: pass
* Bug id/link: (this should link to your github issue id)none
* Additional notes:none
* Test case 9, generate tips(GPT)
* New or old: New
* Test items: After type something and click generate button, you can see the answer from GPT.
* Test priority: high
* Dependencies: create \ type
* Preconditions: create \ type
* input data: type something
* Test steps:type something -> click generate -> see answer
* Postconditions:none
* Expected output: see answer from GPT
* Actual output:see answer from GPT
* Pass or Fail: pass
* Bug id/link: (this should link to your github issue id)none
* Additional notes:none
* Test case 10, back button to home page
* New or old:New
* Test items: click it and can be back to homepage.
* Test priority: Low
* Dependencies: nav to doc first
* Preconditions: nav to doc first
* input data:none
* Test steps: click “back” -> to homepage
* Postconditions: none
* Expected output: back to homepage successfully
* Actual output: back to homepage successfully
* Pass or Fail: pass
* Bug id/link: none
* Additional notes:none

Backend test

* Test case 11, login
* New or old: New
* Test items: backend login method
* Test priority: high
* Dependencies: none
* Preconditions: none
* input data: Google JWT token
* Test steps: using postman posting a token
* Postconditions: none
* Expected output: Access Token
* Actual output: Access Token
* Pass or Fail: pass
* Bug id/link: none
* Additional notes:none
* Test case 12, middleware
* New or old: New
* Test items: VerifyToken method
* Test priority: high
* Dependencies: none
* Preconditions: none
* input data: {req.headers.Authorization: token}
* Test steps:
* Postconditions: none
* Expected output: correct verified message and decoded object
* Actual output: correct verified message and decoded object
* Pass or Fail: pass
* Bug id/link: none
* Additional notes:none
* Test case 13, getUser()
* New or old: New
* Test items: getUser() method
* Test priority: high
* Dependencies: none
* Preconditions: none
* input data: {user\_id: user1}
* Test steps:
* Postconditions: none
* Expected output:
* Actual output: user and docs
* Pass or Fail: pass
* Bug id/link: none
* Additional notes:none
* Test case 14, createDoc()
* New or old: New
* Test items: createDoc() method
* Test priority: high
* Dependencies:
* Preconditions: logined
* input data: title
* Test steps:
* Postconditions: none
* Expected output: {newDoc\_id}
* Actual output: {newDoc\_id}
* Pass or Fail: pass
* Bug id/link: none
* Additional notes:none
* Test case 15, deleteDoc()
* New or old: New
* Test items: deleteDoc() method
* Test priority: high
* Dependencies:
* Preconditions:
* input data: doc\_id
* Test steps:
* Postconditions:
* Expected output: {res return message “delete success”}
* Actual output: {res return message “delete success”}
* Pass or Fail: pass
* Bug id/link: none
* Additional notes:none
* Test case 16, getDoc()
* New or old: New
* Test items: getDoc()
* Test priority: high
* Dependencies:
* Preconditions:
* input data: doc\_id
* Test steps:
* Postconditions:
* Expected output: {Document}
* Actual output:{Document}
* Pass or Fail: pass
* Bug id/link: none
* Additional notes:none
* Test case 17, updateContent()
* New or old: New
* Test items: updateContent()
* Test priority: high
* Dependencies:
* Preconditions:
* input data: Content, docId
* Test steps:
* Postconditions:
* Expected output: {updatedDoc}
* Actual output: {updatedDoc}
* Pass or Fail: pass
* Bug id/link: none
* Additional notes:none
* Test case 18, updateSummary()
* New or old: New
* Test items: updateSummary()
* Test priority: high
* Dependencies:
* Preconditions: has user and userDoc
* input data:
* Test steps:
* Postconditions:
* Expected output: {updatedSummary}
* Actual output: {updateSummary}
* Pass or Fail: pass
* Bug id/link: none
* Additional notes:none

# Automated Testing Report

* **For frontend test**

All of our test code are under ./code/client/Cypress/e2e. We use Cypress.

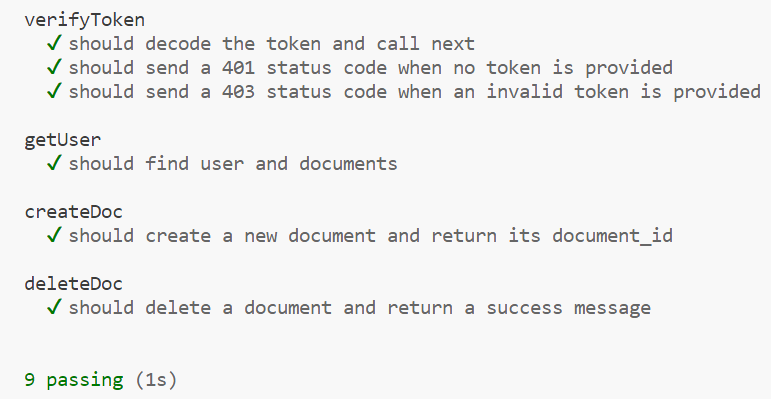
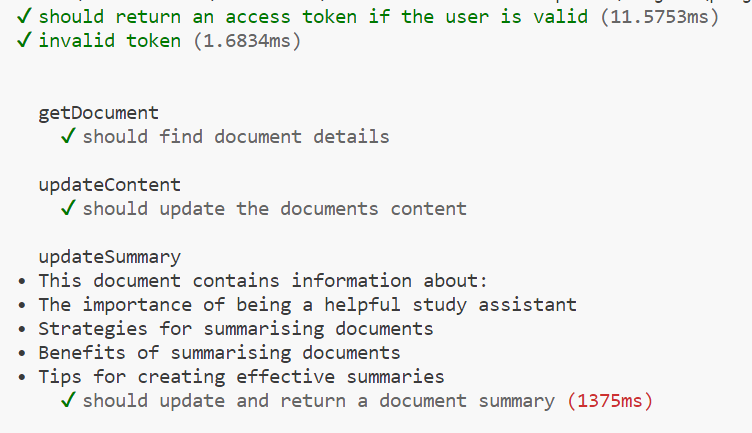
We do end to end testing for all frontend functions. We wrote test step by step in Cypress and run it.

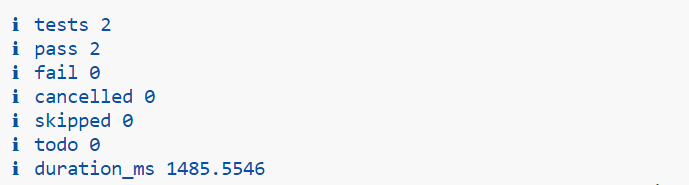
**For backend test**

All of our test code are under ./code/server/test. We use Mocha, Sinon, and Chai.

We do unit testing for every controller methods. We test with mock input

and stub MongoDB functions, req and res methods, other methods used in the controller methods. Also, we set the expected response behavior.





# Testing Metrics

# Pass rate = Number of tests passed / Number of tests

We aimed to achieve a 100% pass rate for all our functionalities and methods are essential to the proper functioning of the application.

# References /

**None**

# Glossary

**None**