Test Document

for

The Dorm Room Dealer

Version <1.1>

Prepared by Group # 15 Group Name: Tech Jedis

Akhil Sagwal	200078	asagwal20@iitk.ac.in
Akshat Gupta	200085	akshatg20@iitk.ac.in
Devesh Shukla	200322	devesh20@iitk.ac.in
Milan Anand Raj	200584	manandraj20@iitk.ac.in
Sarthak Motwani	200887	sarthakm20@iitk.ac.in
Rahul Rustagi	200756	rrustagi20@iitk.ac.in
Akshat	200080	akshat20@iitk.ac.in
Vaibhav Chirania	211134	vaibhavc21@iitk.ac.in

Course: CS253A

Mentor TA: *Sumit*

Date: 02/04/2023

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
1.1	Akhil Sagwal Akshat Gupta Sarthak Motwani Devesh Shukla Milan Anand Raj Akshat Rahul Rustagi Vaibhav Chirania	This is the first version of the test document, containing details of unit tests, integration tests and system tests.	02/04/2023

1 Introduction

Mention the test strategy
(Manual testing? Test automation? …)
We did both automated and manual testing for our software. We tried to use automated testing as much as we could, but for some sets of APIs, the required user interaction was quite complex to automate, and hence, we resorted to manual testing for such sets of APIs.

When was the testing conducted?
(in parallel with the implementation? After the implementation was completed? …)
The testing of the application was done both during the implementation phase as well as after completing the implementation part. During the implementation, we mostly tested by hand (by using print statements) while writing each function independently and when putting them together. Because of this approach, we successfully found and removed many bugs in the code even before the implementation part was fully done.

After the implementation, we started automating our tests using the Python standard library module ***unittest***, which Django's unit tests use. During this phase, we collaboratively wrote unit tests for the various APIs and suggested the required changes to the developers of those APIs.

Who were the testers?
(the developers? Independent testers? …)
For this project, our team members were both developers and testers of the application. We tested APIs that we developed as well as those that were implemented by other developers in the team.

What coverage criteria were used?
The testing covered the code exhaustively. All functions were tested with sufficient numbers and a variety of test cases. Also, all possible conditions inside each branching statement were tested using test cases that were set up correctly.

Have you used any tool for testing?
We have used the Python standard library module ***unittest***, used by Django’s unit tests. This module defines tests using a class-based approach. It was used for automated testing of various functions we had implemented for our application.

2 Unit Testing

(Note: All the test functions corresponding to each of the following tests are present in the tests_views.py file in the Tests directory in the project.)

1. User Login: Successful Login of a User

Unit Details: Unit for a User Login - [login ()]
Manual Testing: Yes
Automated Testing: Yes
Test Owner: *Sarthak Motwani (200887)*
Test Date: *[03/25/2023]*
Test Class: LoginViewTestCase
Test Function: test_login_view_success()
Test Results: A user, whose login credentials exist in the database can successfully log in and is redirected to the 'homepage'.
Structural Coverage: The branch in the login view function, which is executed when a user is successfully authenticated, is covered by this test.

2. User Login: Unsuccessful Login of a User

Unit Details: Unit for a User Login - [login ()]
Manual Testing: Yes
Automated Testing: Yes
Test Owner: *Sarthak Motwani (200887)*
Test Date: *[03/25/2023]*
Test Class: LoginViewTestCase
Test Function: test_login_view_failure()
Test Results: A user, whose login credentials do not exist in the database is not able to log in and the system prompts a message saying 'Invalid Username/Password'.
Structural Coverage: The branch in the login view function, which is executed when a user is not authenticated, is covered by this test.

3. New User Register: Successful Registration of a User

Unit Details: Unit for a new User Register - [register()]
Manual Testing: Yes
Automated Testing: Yes
Test Owner: *Sarthak Motwani (200887)*
Test Date: *[03/25/2023]*
Test Class: RegisterViewTestCase
Test Function: test_register_view_success()

Test Results: A new user, who registers with a new username, a new email and enters the same passwords (two password fields are there when a user registers and both these passwords should match for successful registration) is able to successfully register. His details are successfully added to the database, and he is redirected to the login page.
Structural Coverage: The branch in the register view function, which is executed when a user has entered a new username, a new email and both passwords as the same, is covered by this test case.

4. New User Register: Unsuccessful Registration of a User

Unit Details: Unit for a new User Register - [register()]
Manual Testing: Yes
Automated Testing: Yes
Test Owner: *Sarthak Motwani (200887)*
Test Date: *[03/26/2023]*
Test Class: RegisterViewTestCase
Test Function: test_register_view_password_mismatch()
Test Results: A new user, who attempts to register with a new username, a new email but enters different passwords is not able to register. The system prompts a message, “Password does not match”.
Structural Coverage: The branch in the register view function, which is executed when a user has input both passwords differently, is covered by this test case.

5. New User Register: Unsuccessful Registration of a User

Unit Details: Unit for a new User Register - [register()]
Manual Testing: Yes
Automated Testing: Yes
Test Owner: *Sarthak Motwani (200887)*
Test Date: *[03/26/2023]*
Test Class: RegisterViewTestCase
Test Function: test_register_view_username_exists()
Test Results: A new user, who attempts to register with a username that already exists in the database is not able to register. The system prompts a message indicating the entered username has already been taken.
Structural Coverage: The branch in the register view function, which is executed when a user has entered an existing username, is covered by this test case.

6. New User Register: Unsuccessful Registration of a User

Unit Details: Unit for a new User Register - [register()]
Manual Testing: Yes

Automated Testing: Yes
Test Owner: *Sarthak Motwani (200887)*
Test Date: *[03/26/2023]*
Test Class: RegisterViewTestCase
Test Function: test_register_view_email_exists()
Test Results: A new user, who attempts to register with an email that already exists in the database is not able to register. The system prompts a message indicating that the entered email already exists in the database.
Structural Coverage: The branch in the register view function, which is executed when a user has entered an existing email, is covered by this test case.

7. User Logout: Successful Logout

Unit Details: Logout of a User- [logout()]
Manual Testing: Yes
Automated Testing: Yes
Test Owner: *Sarthak Motwani (200887)*
Test Date: *[03/26/2023]*
Test Class: LogoutViewTestCase
Test Function: test_logout()
Test Results: A user, who was already logged in into his user account, is successfully able to logout upon clicking the logout button and is redirected to the login page.
Structural Coverage: Entire logout view function is executed by this test case.

8. Adding a New Item: Item added Successfully

Unit Details: Unit for Adding a new item- [additem()]
Manual Testing: Yes
Automated Testing: Yes
Test Owner: *Milan Anand Raj (200584)*
Test Date: *[03/25/2023]*
Test Class: AddItemViewTestCase
Test Function: test_additem_valid()
Test Results: A logged in user, who enters all the product details in valid formats, including the start date of his product before the end date, while adding an item, is able to successfully add his item in the database. *Note that the system does not allow the user to enter any detail in any invalid format before submitting the form (for example, the system will not allow the user to enter a string of characters in place of Base Price and so on).*
Structural Coverage: The branch in the additem view function, which is executed when a user has entered all the product details in valid formats and start date <= end date, is covered by this test case.

9. Adding a New Item: Adding Item Unsuccessful

Unit Details: Unit for Adding a new item- [additem()]
Manual Testing: Yes
Automated Testing: Yes
Test Owner: *Milan Anand Raj (200584)*
Test Date: *[03/25/2023]*
Test Class: AddItemViewTestCase
Test Function: test_additem_invalid()
Test Results: A logged in user, who enters all the product details in valid formats, but the start date of his product is after the end date, while adding an item, is not able to add his item in the database. The system prompts a message indicating that the start and end dates are invalid.
Structural Coverage: The branch in the additem view function, which is executed when the start date >= end date is covered by this test case.

10. Display of Items in Homepage: Search By a Particular Category

Unit Details: Unit for Home Page- [home()]
Manual Testing: Yes
Automated Testing: Yes
Test Owner: *Milan Anand Raj (200584)*
Test Date: *[03/25/2023]*
Test Class: HomeViewTestCase
Test Function: test_home_view_with_valid_category()
Test Results: A logged-in user who searches the live products on the homepage with a particular valid category can successfully view the products belonging to that category and no product of any other category.
Structural Coverage: The branches in the home view function, which are executed when the user posts a particular category while searching by category, are covered by this test case.

11. Display of Items on the Homepage

Unit Details: Unit for Home Page- [home()]
Manual Testing: Yes
Automated Testing: Yes
Test Owner: *Milan Anand Raj (200584)*
Test Date: *[03/26/2023]*

Test Class: HomeViewTestCase
Test Function: test_home_view_with_all_categories()
Test Results: A logged-in user, who doesn't search a particular category or searches all categories of live products on the homepage can successfully view all the live products.
Structural Coverage: The branches in the home view function, which are executed when the user posts "All categories" while searching by category or doesn't post a specific category, are covered by this test case.

12. Display of Items on the Homepage

Unit Details: Unit for Home Page- [home()]
Manual Testing: Yes
Automated Testing: Yes
Test Owner: *Milan Anand Raj (200584)*
Test Date: *[03/26/2023]*
Test Class: HomeViewTestCase
Test Function: test_home_view_with_future_items()
Test Results: A logged-in user is able to see all the future items on the home page.
Structural Coverage: The part of the home view function responsible for displaying the future items is covered by this test case.

13. Calling of productStatus() and sendMail() functions in Home Page

Unit Details: Unit for Home Page- [home()]
Manual Testing: Yes
Automated Testing: Yes
Test Owner: *Akhil Sagwal (200078)*
Test Date: *[03/28/2023]*
Test Class: HomeViewTestCase
Test Function: test_home_view_calls_product_status_and_send_mail()
Test Results: The home view function was successfully able to call the productStatus() view function and the sendMail() view function().
Structural Coverage: The part of code in the home view function, which calls the productStatus() view function and the sendMail() view function, is covered by this test case.

14. Sending of Mails to Product Owner and Highest Bidder

Unit Details: Unit for Home Page- [home()]
Manual Testing: Yes
Automated Testing: Yes
Test Owner: *Sarthak Motwani (200887)*

Test Date: *[03/27/2023]*
Test Class: SendMailViewTest
Test Function: test_sends_emails()
Test Results: The home view function successfully called the sendMail function. Emails are successfully sent to the highest bidder (sharing the seller's contact details) and to the product's owner (sharing the highest bidder's contact details). Emails are successfully sent only for those items whose end dates have passed and their win emails have not been sent yet.
Structural Coverage: The branch of the sendMail function, which is executed when the highest bidder of the item exists, is covered by this test case. The sendMail() function does nothing if no highest bidder exists for an item.

15. Display of Items in Dashboard

Unit Details: Unit for Dashboard- [dashboard()]
Manual Testing: Yes
Automated Testing: Yes
Test Owner: *Akhil Sagwal (200078)*
Test Date: *[03/28/2023]*
Test Class: DashboardViewTest
Test Function: test_dashboard_view()
Test Results: A logged in user on the dashboard in his account is successfully able to see his selling history (live auction products, past auction products, and future auction products), his bidding history (live bids and past purchases) as well as his profile details.
Structural Coverage: The part of the dashboard view function displaying details of the user and of items associated with a user at appropriate sections is covered by this test case.

16. Auction stop and Sending of Mails to Product Owner and Highest Bidder through Dashboard (when an auction is closed by the seller before the end date)

Unit Details: Unit for Dashboard - [dashboard()]
Manual Testing: Yes
Automated Testing: Yes
Test Owner: *Devesh Shukla (200322)*
Test Date: *[03/28/2023]*
Test Class: SendMailViewTest
Test Function: test_dashboard_view_stops_auction_and_sends_mails()
Test Results: When a logged in user (seller) stopped the auction for a particular live item by posting the end auction request through the dashboard, the status of the item

successfully changed from ‘live’ to ‘past’. If the highest bidder of that product exists, emails are successfully sent to the highest bidder (sharing the contact details of the seller) and to the owner of the product (sharing the contact details of the highest bidder). If not, no emails are sent.

Structural Coverage: The part of the dashboard function, which is executed when the user posts an end auction request for a live item, is covered by this test case.

17. Edit User’s Profile Details

Unit Details: Unit for Edit Profile- [edit_profile()]

Manual Testing: Yes

Automated Testing: Yes

Test Owner: *Devesh Shukla (200322)*

Test Date: *[03/28/2023]*

Test Class: EditProfileTestCase

Test Function: test_edit_profile_view_GET()

Test Results: When a logged in user lands on the edit profile page, his current details are successfully retrieved.

Structural Coverage: The part of the edit_profile view function responsible for rendering the current details of the user is covered by the test case.

18. Edit User’s Profile Details

Unit Details: Unit for Edit Profile- [edit_profile()]

Manual Testing: Yes

Automated Testing: Yes

Test Owner: *Devesh Shukla (200322)*

Test Date: *[03/30/2023]*

Test Class: EditProfileTestCase

Test Function: test_edit_profile_view_POST()

Test Results: When a logged in user posts new details on the edit_profile page, the new profile details are successfully updated, and the user is redirected to the dashboard page.

Structural Coverage: The part of the edit_profile view function responsible for handling a post request of the new details of the user’s profile is covered by this test case.

19. Details of a Particular Live Item

Unit Details: Unit for View and Bid Item - [biditem()]

Manual Testing: Yes
Automated Testing: Yes
Test Owner: *Devesh Shukla (200322)*
Test Date: *[03/30/2023]*
Test Class: BidItemTestcase
Test Function: test_biditem_view_live_item()
Test Results: When a logged in user clicks on "view and bid" on a particular live item, he is successfully redirected to the webpage displaying the details of that particular item.
Structural Coverage: The entire biditem view function was covered in this test case.

20. Bidding a Particular Live Item: Successful Bid

Unit Details: Unit for Registering a bid for an item- [SuccessfullBid()]
Manual Testing: Yes
Automated Testing: Yes
Test Owner: *Akshat (200080)*
Test Date: *[03/27/2023]*
Test Class: SuccessfulBidTestcase
Test Function: test_successfull_bid()
Test Results: When a logged in user enters a bid (for an item whose owner is not the user himself) greater than the current price of the item, the bid is successfully registered with the updated current price and the updated highest bidder of the item. *Note that the system does not allow the user to enter a bid smaller than the item's current price*. Email is successfully sent to the item's owner indicating that the product is bidded by a certain amount.
Structural Coverage: The branch of the SuccessfullBid view function responsible for registering a valid bid, updating item details accordingly, and sending emails is covered by this test case.

21. Bidding a Particular Live Item: Unsuccessful Bid

Unit Details: Unit for Registering a bid for an item- [SuccessfullBid()]
Manual Testing: Yes
Automated Testing: Yes
Test Owner: *Akshat (200080)*
Test Date: *[03/27/2023]*
Test Class: SuccessfulBidTestcase
Test Function: test_bid_on_own_item()

Test Results: When a logged in user enters a bid for an item whose owner is the user himself, the bid is not registered and the system prompts a message indicating that the user cannot bid his own item.
Structural Coverage: The branch of the SuccessfullBid view function responsible for handling the case when a user bids his own item is covered by this test case.

22. Updating of Product Status

Unit Details: Unit for Update of the status of an Item- [ProductStatus()]
Manual Testing: Yes
Automated Testing: Yes
Test Owner: *Rahul Rustagi (200756)*
Test Date: *[03/27/2023]*
Test Class: SuccessfulBidTestcase
Test Function: test_product_status_live()
Test Results: A live item, whose sold status was initially “unsold” is updated to “Bidded”, when a bidder bids for the product.
Structural Coverage: The branch of the ProductStatus code responsible for updating the sold status of the item when it is live, and there exists a highest bidder for that item is covered by this test case.

23. Updating of Product Status

Unit Details: Unit for Update of the status of an Item- [ProductStatus()]
Manual Testing: Yes
Automated Testing: Yes
Test Owner: *Rahul Rustagi (200756)*
Test Date: *[03/27/2023]*
Test Class: SuccessfulBidTestcase
Test Function: test_product_status_past()
Test Results: A past item, whose sold status was initially “Bidded" is updated to “Sold”, when there exists a highest bidder for that item.
Structural Coverage: The branch of the ProductStatus code responsible for updating the sold status of the item when it is past, and there exists the highest bidder for that item is covered by this test case.

24. Updating of Product Status

Unit Details: Unit for Update of the status of an Item- [ProductStatus()]
Manual Testing: Yes
Automated Testing: Yes
Test Owner: *Rahul Rustagi (200756)*

Test Date: *[03/27/2023]*
Test Class: SuccessfulBidTestcase
Test Function: test_product_status_future()
Test Results: A future item's sold status is successfully assigned to "Yet to be Auctioned".
Structural Coverage: The branch of the ProductStatus code responsible for updating the sold status of the future item is covered by this test case.

25. Updating of Product Status

Unit Details: Unit for Update of the status of an Item- [ProductStatus()]
Manual Testing: Yes
Automated Testing: Yes
Test Owner: *Rahul Rustagi (200756)*
Test Date: *[03/27/2023]*
Test Class: SuccessfulBidTestcase
Test Function: test_product_status_none()
Test Results: A live or a past item's sold status, for which there exists no highest bidder, is successfully assigned to "Unsold".
Structural Coverage: The branch of the ProductStatus code responsible for updating the sold status of the item when it is live or past, and there exists no highest bidder for that item is covered by this test case.

3 Integration Testing

1. Checking the interface between the signup and the login module

Module Details: register and login modules tested

Test Owner: Vaibhav Chirania (211134)

Test Date: *28/03/2023*

Test Results: *On successfully signing up for the application using the valid credentials, the user gets redirected to the login page.*

2. Checking the interface between the login module and the home module

Module Details: login and home modules tested

Test Owner: Vaibhav Chirania (211134)

Test Date: *28/03/2023*

Test Results: *On entering the correct credentials on the login page, the user is redirected to the homepage, showing the available products for bidding.*

3. User authentication fails on entering incorrect credentials

Module Details: login and UI framework tested

Test Owner: Vaibhav Chirania (211134)

Test Date: *28/03/2023*

Test Results: *On entering the wrong credentials in the login form, the user gets an alert message of incorrect credentials and thus cannot access the authorization-protected endpoints.*

***4.* Checking the effect of logout on the application**

Module Details: logout module tested

Test Owner: Vaibhav Chirania (211134)

Test Date: *28/03/2023*

Test Results: *On logging out from the application, the user gets redirected to the login page and the associated cookies get cleared so as not to authorize the user anymore.*

5. **Already existing users cannot register i.e., already existing username cannot be used while signing up**

Module Details: register module and authorization middleware tested
Test Owner: Vaibhav Chirania (211134)
Test Date: *29/03/2023*
Test Results: *On choosing a username that already exists in the database, the user gets an alert message showing that a user with the same username already exists.*

6. **All endpoints except the home, signup and login page are accessible only when a user is authorized**

Module Details: authorization middleware along with all other modules tested
Test Owner: Akshat Gupta (200085)
Test Date: *29/03/2023*
Test Results: *To access the GET and POST endpoints except for the one for register and login, can only be accessed when a user is logged into the system, i.e., have an active session.*

7. **User can view details of the item and also bid for it by selecting the "View and Bid" button**

Module Details: home and bidItem module tested
Test Owner: Akshat Gupta (200085)
Test Date: *29/03/2023*
Test Results: *By selecting the "View and Bid" button, the user can see in detail any of the live products listed on the home page. By pressing the button, the user is redirected to the '/bidItem' page, where the product details are listed. The user can bid for the product, provided he/she enters a valid bid (bid must be greater than the current price of the product)*

8. **User can filter products through Search bar**

Module Details: home module and UI framework tested
Test Owner: Akshat Gupta (200085)
Test Date: *29/03/2023*
Test Results: *By using the drop-down menu on the top of the homepage and pressing the "Search" button, the user will only see the list of products that match the 'tag' selected by the user. This includes both live auctions and future auctions.*

9. User can move through products listed on multiple pages

Module Details: home module and UI framework tested
Test Owner: Akshat Gupta (200085)
Test Date: *29/03/2023*
Test Results: *The home page has been paginated to contain not more than six live products on the first page. The user was successfully able to see other listed products by selecting the subsequent pages.*

10. User can add a product for auction and is redirected to home page

Module Details: home module and addItem module tested
Test Owner: Akshat Gupta (200085)
Test Date: *29/03/2023*
Test Results: *The 'addItem' module allows the user to fill in the details of their product and list it for auction. Upon a successful listing, the user is redirected to the home page, where they can see their product. If the user enters any invalid details (for e.g., illegal start or end dates), then the user encounters an error message and stays on the addItem module only.*

11. User can manually end an auction prematurely

Module Details: home, dashboard, and sendMail modules tested
Test Owner: Akshat Gupta (200085)
Test Date: *30/03/2023*
Test Results: *The user can end an auction for their product prematurely by pressing the "End Auction" button present below their live listing on the dashboard page. After pressing the button, the user stays on the same page, but the product is correctly removed from the home page. Also, after ending the auction, the seller and highest bidder at that time both receive the mail correctly.*

12. User can edit their profile information

Module Details: edit_profile module and dashboard module tested
Test Owner: Akshat Gupta (200085)
Test Date: *30/03/2023*
Test Results: *The user can change their personal information, including name, email, contact no, profile image and hall, by pressing the "Edit Details" button present on the dashboard. Upon pressing the button, the user is redirected to '/edit_profile', where he/she can change their information.*

13. User receives mail regarding the sale of product upon completion of auction

Module Details: home and sendMail module tested

Test Owner: Akshat Gupta (200085)

Test Date: *30/03/2023*

Test Results: *When the user logs in to the system and visits the home page, a mail is sent to the seller and highest bidder of any of the products whose auctions might have ended post the last login.*

4 System Testing

1. **Requirement:** ***Testing the complete functionality of the item application*** *(User should be able to list a product for sale, set its details, and also be able to bid for different listed products)*
   Modules Used -
   - *addItem*
   - *bidItem*
   - *successfullBid*
   - *home*
   - *productStatus*
   - *sendMail*

Test Owner: *Akshat Gupta (200085)*
Test Date: *[28/03/2023] - [30/03/2023]*
Test Results: *All the tests related to this requirement were successfully executed.*
- *User is able to list the product on the website successfully, subject to the fact that he/she doesn't provide invalid/impossible details.*
- *In case user provides invalid details like (entering start_date before current date, entering start_date after end_date), user will be redirected to an error notification page.*
- *User is able to successfully bid for a product listed on the home page.*
- *If user places a bid lower than the current price, he/she will be shown an error.*
- *If user places a bid on their own listing, they will be shown an error.*
- *A mail is sent to the seller whenever someone bids on their product.*
- *A final mail is sent to the seller and highest bidder upon the completion of the auction.*
- *If an item is listed for a future date, then the product is shown on the "Future Auctions" section on the home page, with the appropriate details.*
- *Products listed on*

2. **Requirement:** ***Testing the maintenance of user profile through Dashboard*** *(User should be able to see and edit his/her personal information, and see their selling and buying history on the website, with the added ability to prematurely end ongoing auctions started by them)*
   Modules Used -
   - *productStatus*
   - *dashboard*
   - *sendMail*
   - *edit_profile*

- *home*

Test Owner: *Akshat Gupta (200085)*
Test Date: *[28/03/2023] - [30/03/2023]*
Test Results: *All the tests related to this requirement were successfully executed.*

- *User is able to see his/her personal information on the dashboard.*
- *User is able to see his/her selling history on the dashboard. This includes sections which show the products listed in the past, future and present.*
- *User is able to see his/her bidding history on the dashboard. This includes live bids and bids on past products.*
- *User is able to prematurely end the auction for a product that they had listed. This removes the product from the live auction on the home page.*
- *When an auction is ended prematurely, the final mail is sent to the seller and the highest bidder at that point.*
- *User is able to successfully edit their profile information and also upload a profile picture through the "Edit Details" button on the dashboard.*

….

5 Conclusion

How Effective and exhaustive was the testing?

Each component of the web-based application was tested. The functional and non-functional requirements were tested, which in turn helped us make some of the functions in views.py more precise and accurate. The ***unittest*** module in python helped in creating tests to check functions written in the views.py

Which components have not been tested adequately?

The resetting of passwords and the updating of new passwords in the database have not been adequately tested. What remains to be tested is whether the profile picture is correctly getting updated in the database and is stored as new default fields. Other than that we have written exhaustive tests for unit, integration, and system testing.
For now, we have kept aside the issues faced in operating the application on mobile. (or any other platform than PC).

What difficulties have you faced during testing?

- Because the website uses timezone-aware datetime, we were not able to supply date-time in the timezone-aware format to the test modules, and therefore, even though the tests would run correctly, it gave a warning regarding getting a date-time in naive format.

- Thinking about the exhaustive set of test cases was challenging, but writing them and debugging the output was time taking. It took a fair amount of time learning Django's utility of the ***unittest*** module and integrating it with our functions.

- Writing unit tests for functions involving the sending of emails was a challenging task.

- Determining integration stage tests was also time taking.

- System testing was to the point, as it was deciding or writing a test about the overall functioning of the system.

How could the testing process be improved?

Using automated tests for system testing and further adding comments to the functions in the application during the system implementation phase would have helped a lot in writing test functions.

Appendix A - Group Log

<*Please include here all the minutes from your group meetings, your group activities, and any other relevant information that will assist in determining the effort put forth to produce this document*>

Meeting Date	Members present	Topic Discussed
24/03/2023	All members	Discussed and Divided the tasks for implementing System, Integration and Unit testing.
31/03/2023	TA + Rest of the members present	Discussed System Testing and Integration Testing sub parts to get a better idea of how to implement them and received feedback on our already submitted version of application