Test Document

for

Health++

Version 1.1

Prepared by

Group: 5 **Group Name:** StarBugs

Name	Roll no.	E-Mail
Anuj	210166	anuj21@iitk.ac.in
Pulak Gautam	210791	pulakg21@iitk.ac.in
Dev Gupta	210318	devgupta21@iitk.ac.in
Abhimanyu Sethia	190023	sethia@iitk.ac.in
Deepanshu	210310	deepanshu21@iitk.ac.in
A. Atulya Sundaram	210001	aatulya21@iitk.ac.in
Kushagra Bhatnagar	210550	kushagrab21@iitk.ac.in
Pinkesh Mahawar	210719	pinkeshm21@iitk.ac.in
Pradeep Chahal	210733	pradeepc21@iitk.ac.in
Vaishnavi	211140	vaishnavi21@iitk.ac.in

Course: CS253

Mentor TA: *Anuj Shrivastava (anujsh@cse.iitk.ac.in)*

Date: 2 April 2023

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
1	All the team members	Added the tests done by different members and the results.	02/04/23

1 Introduction

Because of the complexity of the project, it was not feasible to implement thorough automated testing and thus most of the testing is done manually, by developers (team members). Although we had used Django *tests.py* which provides an inbuilt Django testing framework to verify system integration like creating and handling http requests during the implementation phase itself to test and debug problems with integration.

Our project testing was conducted in mainly two phases –
During the implementation: We focussed on the correctness of our code and ensure that the implemented features are working properly, also the frontend was thoroughly tested to be behaving correctly on different screen sizes and that the options/features are easy to find on the website. Here the testing was done by the member developing the feature itself.
After the implementation: We focussed on ensuring the implemented features do not interfere with each other. Also, we validated our project satisfies the end-user requirements correctly. Further, features like security, confidentiality which could not be tested during the development were tested in this phase.

Most of the testing is done by our team (developers), we had to go through the code multiple times and verify our understanding of the code through manually testing as the actual implementation of some of the in-built Django features were not known to us. We also reviewed each other's code to minimize unhandled errors.
We also had the overall system testing done by some of our friends to get review about the system's performance and ease of use.

We used multiple coverage criteria during our software testing –
Code coverage: Ensured that all the lines of code in the website's implementation have been executed during testing. This was done to eliminate any dead code/bug that was unintentionally added during implementation.
User interface coverage: Ensured that all the elements in the user interface are working perfectly. It includes testing the layout, design, and responsiveness of the website on different devices and maintaining cross-browser compatibility.
Path coverage: Ensured that all possible paths through a program's source code are executed during testing. Manually tested for the possible combinations of decision points to identify potential errors or defects in the code.

2 Unit Testing

1. Ambulance booking page –

Our testing involved verifying whether the page accurately captures all inputs and whether it displays appropriate error messages for missing user inputs.

Unit details: bookambulance/base.html

Test owner: Kushagra Bhatnagar

Test Date: 25/03/2023

Test Results: (PASSED) The page is rendering properly and taking inputs. Form Validation using Javascript is working properly.

Structural Coverage: Path Coverage has been tested by testing over all different types of erroneous inputs.

2. Blood Donation Request page –

Tested that the frontend form captures the required details for blood donation request and that the requests from database are shown appropriately on the page.

Unit details: blood_donate/templates/home.html

Test owner: Anuj

Test Date: 25/03/2023

Test Results: (PASSED) The page is rendering properly and taking inputs. Form Validation using Javascript is working properly, all the data sent from backend is shown appropriately in the open requests table.

Structural Coverage: Tested for code coverage to find unintentional code.

3. Ambulance booking page –

Our testing involved verifying whether the page accurately captures all inputs and whether it displays appropriate error messages for missing user inputs.

Unit details: bookambulance/base.html

Test owner: Kushagra Bhatnagar

Test Date: 25/03/2023

Test Results: (PASSED) The page is rendering properly and taking inputs. Form Validation using Javascript is working properly.

Structural Coverage: Path Coverage has been tested by testing over all different types of erroneous inputs.

4. Appointment booking page –

Testing whether the appointment form after submitting creates a appointment in Doctor’s list, and patient’s list, and also set the doctor as “busy” during the time-slot inputted.

Unit details: appointment_form.html

Test owner: Anuj

Test Date: 25/03/2023

Test Results: (PASSED) The page is rendering properly and taking inputs.

Form Validation using JavaScript is working properly and all the data is passed by backend in appointment list appropriately.

Structural Coverage: Path Coverage has been tested by testing over all different types of erroneous inputs.

5. Health records–

This test is to check whether when doctor gives Prescription to a patient, it gets added into the medical history of patient or not and whether it displays appropriate error messages for missing user inputs.

Unit details: view_health_records.html

Test owner: Kushagra Bhatnagar

Test Date: 25/03/2023

Test Results: (PASSED) The page is rendering properly and taking inputs.

Form Validation using JavaScript is working properly and all the data is passed by backend in appointment list appropriately.

Structural Coverage: Path Coverage has been tested by testing over all different types of erroneous inputs.

6. Get Appointments –

This test is to check whether, the table lists the relevant appointments correctly, i.e. if the user is a Patient, it shows the list of appointments with different doctors, and if the user is the doctor.it shows their appointments with different patients.

Unit details: views.getappointments()

Test owner: Dev Gupta

Test Date: 25/03/2023

Test Results: (PASSED)The list shows the relevant appointments accordingly

Structural Coverage: Conditional coverage has been tested by testing overall test cases.

7. Register User–

This test is to check whether after the registration process is completed, a new user is created in the database or not.

Unit details: views.registerUser()

Test owner: Dev Gupta

Test Date: 25/03/2023

Test Results: (PASSED)After Registration, a new user is added into the database, with the correct attributes as given by the user.

Structural Coverage: Test Coverage includes data validation testing and data integration testing.

8. Create Health Record–

This test is to check whether after the doctor gives their prescription to the patient whether it gets added to their medical history or not and with the correct details.

Unit details: views.create_health_record()

Test owner: Dev Gupta

Test Date: 25/03/2023

Test Results: (PASSED) After submitting, the medical record gets correctly added to medical history of the patient with the correct details.

Structural Coverage: The Function was tested using many different users and with many different inputs.

9. Leave application response–

This test is to check whether the function is working for accept or rejection to a leave application or not.

Unit details: views.Application_response()

Test owner: Pinkesh Mahawar

Test Date: 25/03/2023

Test Results: (PASSED) after acceptance or rejection, this information is transferred to the students.

Structural Coverage: Conditional Coverage has been tested for overall test cases.

10. Login User–

This test is to check whether relevant user data for given inputs is exists or not.

Unit details: views.loginUser()

Test owner: Pinkesh Mahawar

Test Date: 25/03/2023

Test Results: (PASSED) After submitting, the user goes to homepage to corresponding role.

Structural Coverage: The Function was tested using many different username and passwords.

3 Integration Testing

1. User Authentication

Tested that the login module has integrated properly with other features, like appointment booking and leave application. Ensured that only the users who are entitled to see a particular page can view it.

Module Details: Login functionality and role-based access to different data and features
Test Owner: Anuj
Test Date: 03/29/2023 - 03/30/2023
Test Results: (PASSED) Only logged in users can use the specific features, also person with student role can not access the features of DUGC or doctor and so on.
Additional Comments: Found a glitch that allowed students access to view applications page that is meant for DUGC admin, later resolved.

2. Health records

Health Booklet should be visible to the patient, doctor should be able to add prescription to the patient's health booklet and the changes should reflect to the patient.

Module Details: health booklet and add health record module
Test Owner: A Atulya Sundaram
Test Date: 03/31/2023 - 03/31/2023
Test Results: (PASSED) Prescriptions added by the doctor on a patient's account are visible to that patient and doctor can see previous records that has been added on the patient's account.
Additional Comments: No issues found

3. Blood Donation

The Blood donation page shows each user the requests that the user can accept based on his own blood group from the user profile.

Module Details: User class and blood_donate function
Test Owner: Deepanshu
Test Date: 03/30/2023 - 03/30/2023
Test Results: (PASSED) The open blood request page is filtered on the basis of user's blood group from user profile.
Additional Comments: No issues found

4 System Testing

1. **Requirement:** The site should launch properly, the images and graphics are rendered appropriately, and the interface is responsive.

Test Owner: Pradeep Chahal

Test Date: 03/31/2023 - 03/31/2023

Test Results: (PASSED) the homepage renders as expected with appropriate graphical interface and responsiveness.

Additional Comments: The landing homepage is rendered properly, and all the graphics and images were rendered accordingly with appropriate response and reactiveness.

2. **Requirement:** User Sign In / Sign Up.

Test Owner: Vaishnavi

Test Date: 03/29/2023 - 03/30/2023

Test Results: (PASSED) Existing users could login using their registered usernames and password. In case of, incorrect credentials entered, the page showed the message that error has occurred. A new user was registered using the Register option and could successfully. Login after

Additional Comments: A new user could not register using the same username as that of an existing user. It showed the error to the user that the username was already taken.

3. **Requirement:** Appointment Booking

Test Owner: Vaishnavi

Test Date: 03/31/2023 - 03/31/2023

Test Results: (PASSED) The user could book an appointment, choosing an appropriate date, which prompted the user to select the department, doctor and the available time, which booked an appointment and added to the appointment lists.

Additional Comments: A user's list of appointment were displayed in a queue in the same page as the appointment form and the receptionist could view the appointments of all the users.

We tried booking a new appointment in the time slots for a doctor and time, but it did not show the options to do so.

4. **Requirement:** Blood Donation Facilities

Test Owner: Pradeep Chahal

Test Date: 03/31/2023 - 03/31/2023

Test Results: (PASSED) The patient can put up a request for blood requirements, against his/her username and the request is visible to others.

Additional Comments: A patient can post request for blood with the prompts, blood group, blood volume or any other remarks or comments.

The volume of blood could only be in integral format.

5. **Requirement:** Ambulance Booking

Test Owner: Pradeep Chahal

Test Date: 03/31/2023 - 03/31/2023

Test Results: (PASSED) The user could book an ambulance with the required prompts for location and contact number.

And the driver was able to view the currently allotted pick-up patient with the location and contact details.

The receptionist could view the list of patients to be picked up by the ambulance.

Additional Comments: NA

6. **Requirement:** Digital Medical Records/Prescription

Test Owner: Vaishnavi

Test Date: 03/30/2023 - 04/01/2023

Test Results: (PASSED) The Doctor could create a medical record for a patient with the use of their respective usernames and could also access medical records of a patient using the same.

The patient also could view their own medical history by clicking on the ‘Medical Records’ Tab.

Additional Comments: For viewing the medical history, if the username was not valid, then it gave an error suggesting username does not exist.

7. **Requirement:** Leave application request and approval

Test Owner: Pradeep Chahal

Test Date: 03/29/2023 - 03/31/2023

Test Results: (PASSED) The patient was able to generate a leave application and could check its status.

And the DUGC could see the list of requests for approval and could approve or decline the requests.

Additional Comments: The patient was able to attach the relevant medical record entry and the prescription while requesting for a leave approval.

8. **Requirement:** Performance, Speed and Security

Test Owner: Vaishnavi

Test Date: 04/01/2023 - 04/02/2023

Test Results: (PASSED) It was able to host approximately 50 users efficiently, and all functions worked in real time.

Additional Comments: We have yet to test our software's security against non-minor threats and its functionality while hosting tens of thousands of our target users.

5 Conclusion

How Effective and exhaustive was the testing?
The following things were utilized in the testing of our web application:

1. **Django test.py**

Django test.py is a file in Django, a Python web framework, which provides a command-line interface for running tests against your Django application. The test.py file is located in the root directory of a Django project and is used to run tests for the entire project or specific Django applications within the project. The test.py file essentially provides a unittest-based test runner that can discover and run tests written using Django's built-in testing framework.

2. **Manual Testing**

To test the robustness of the web application the website was hosted on the Django development 0.0.0.0:8000 and was simultaneously tested by all the members of the team, essentially in the format of a roleplay. Each member tested as every user type i.e. “DUGC”, “Student”, “Doctor”, “Receptionist” and “Ambulance” and at the same time fixed any errors.

All the components have been tested thoroughly. As far as the team is concerned there are have been very few oversights. The only thing that has not been tested thoroughly is the security of the system in the face of attackers with malicious intent.

As such, the time constraint was a major difficult as well as the testing each and every role. We had 5 members and 6 functionalities thus resulting in 30 different possible variations in the testing, which took quite a bit of time and was very tedious. Also rapid modification of the code after the error detection was not easy.

We could have streamlined the testing process by focussing on important parts of the application first, then the rest later instead of approaching it sequentially.

Appendix A - Group Log

Date of meeting	Duration of meeting	Agenda of meeting
20th March 2023	2hr	Work Distribution for different sections of testing and user manual was done and researched about the methods and tools available for testing
26th March 2023	1hr	Reconciled the errors found during testing and some code improvements were suggested.
30th March 2023	1hr 30mins	Review of the finalized project and the documents.