

CP10 - Designing a web app for facilitating the submission of low risk ethics applications

Client name: CHAI Lab

Group name: COMP3888_T17_07_Group1

Tutor name: Regina Wang

Group members details:

| | |
|------------------------|------------------------------|
| Ben Raj | (sraj2279, 480564951) |
| Alex Del Favero | (adel8312, 480423887) |
| Zhixin Gao | (zgao9758, 480519320) |
| Shuai Sun | (ssun5590, 480025728) |
| Chengwei Peng | (cpen2847, 480188609) |

School of Computer Science UG Capstone Projects

SOFT3888/COMP388/COMP3988/COMP5615/INFO360

Table of Contents

| | |
|--|-----------|
| Executive summary | 4 |
| Introduction | 5 |
| 2 System Specifications and Design | 7 |
| 2.1 Overview | 7 |
| 2.1.1 User | 7 |
| 2.1.2 Administrator | 11 |
| 2.1.3 Ethics Committee | 12 |
| 2.1.4 Full list of user stories | 12 |
| 2.2 Details: | 16 |
| 2.2.1 Discarded User Stories | 16 |
| 3 Quality of Work | 25 |
| 3.1 Test Plan | 25 |
| 3.2 Test class | 27 |
| 3.3 Test objects | 27 |
| 3.3.1 Context | 27 |
| 3.3.2 Model | 27 |
| 3.3.3 Database | 27 |
| 3.3.4 Request / Response | 28 |
| 3.3.5 Template | 28 |
| 3.3.6 Styling | 28 |
| 3.3.7 Behaviour | 28 |
| 3.4 Automatic Testing | 28 |
| 3.5 Testing Design | 29 |
| 3.5.1 Equivalence partitioning and boundary analysis | 29 |
| 3.6 Test result | 31 |
| 3.7 Test coverage | 31 |
| 3.8 Acceptance Testing | 33 |
| 3.9 Test Summary | 41 |
| 3.10 Tools used to build the system. | 41 |
| 3.10.1 Database tools: | 42 |
| 3.10.2 Source code management: | 42 |
| 3.11 Information search/research and discipline knowledge use and application: | 43 |
| 3.11.1 INFO1110 | 44 |
| 3.11.2 ISYS2120 | 44 |
| 3.11.3 SOFT2412 | 44 |
| 3.11.4 INFO2222 | 44 |

| | |
|---|-----------|
| 3.11.5 Django website | 44 |
| 3.11.6 W3SCHOOLS, Youtube | 44 |
| 4 Quality of Group Processes | 45 |
| 4.1 Tooling for development, management and allocation of tasks | 45 |
| 4.2 Evidence of collaboration and teamwork | 45 |
| 4.3 Allocation of group roles, potential risks, constraints | 47 |
| 4.4 Use of bitbucket and slack and other tools | 47 |
| 4.5 Work with clients | 50 |
| 4.6 Critique | 52 |
| Appendix | 54 |
| A1 | 54 |
| A3 | 54 |
| A4 | 54 |
| References | 55 |

Executive summary

Our group has taken on the challenge of designing a web application to facilitate the submission of low risk ethics applications for CHAI Lab. Our goal is to shorten the amount of time it takes to receive feedback on a submitted application through implementing user and reviewer friendly features. This report details our web application's system specifications and design, how our group maintained quality in our code and how our group worked together to achieve our goal.

We consulted with our client to design 30 user stories, which captured the features to implement into our project. The user stories described a secure web application that allowed researchers to create any number of low risk ethics applications. The application process filters out high risk ethics applications before the user answers the questionnaire. The questionnaire is designed to reduce reviewer response time, by allowing users to select pre-approved template answers and indicating so. The application also allows for users to generate participant information statements and consent forms to distribute to study participants. Once the user has completed their application, they can submit it to the ethics committee for approval through the web app.

The website was developed using HTML, CSS and JavaScript for the front-end and the python web-framework Django as the back-end. Django uses the MVT (Model, View, Template) design pattern. It consists of three main components: the model, view and the template. The model is a data access layer which handles the data in the database. The template is a presentation layer which handles user interaction. The view renders the template and handles the exchange of data between the user and the model.

According to the user stories, we use different types of testing such as unit tests and acceptance testing, also testing techniques to help us design test cases. Before we submit our new work, we make sure our work will pass the previous test cases and the designed test cases. After that, we can get the test result, which reflects the quality of our new tasks.

Our group used Django, Python, as back-end, and HTML, CSS, Javascript, and JQuery as front-end to build the website. We used SQLite as a database at first and transferred to PostgreSQL. We used the knowledge of some previous unit of classes and learned more deeply from the online resources. To manage the source code, we used bitbucket.

Teams used bitbucket, Jira Issue, zoom, and slack in the development of the project. Bitbucket is used by the team to update the source code together, the wiki page of bitbucket would also be used weekly to contain the client meeting, group meeting, in-tutorial meeting, weekly plan, risk management, and tracker logbook. Zoom is a

meeting tool for the team to use. Slack is a chat tool for the team to discuss the project. Team members made a great collaboration with team work. The team uses slack, zoom and email to communicate with the client to discuss and collect the feedback of the project. Team members abide by the Extreme Programming roles to do what they need to do and what they are required to do.

Introduction

The University of Sydney's current ethics approval platform, does not suit the needs of the modern researcher. There are many issues that impact the usability of the platform. The platform caters for the review of low and high risk ethics. It also has a number of usability issues such as spelling and grammar mistakes and visual bugs. Though the main issue with the current platform is the low rate of approval for ethics submissions. It can often take up to 6 months for an application to be approved. This leads to large amounts of research time wasted per year and on occasion, can discourage researchers from pursuing certain areas of study.

The purpose for designing and building a web application to handle the submission of low risk ethics applications is to increase the usability of the site, speed of application completion and ultimately, speed of response from the ethics committee. It is our vision that after improving the quality of life and response time for ethics approval, more students and researchers will be inspired to make further investigations into our world, improving society through the completion of more research projects.

After completion, we will have achieved a high performant, secure, and fully featured web application to handle the creation and submission and approval of low risk ethics applications. Our project is made from many achievements, each can be described as a feature of the web app. The web app allows users to create an application from scratch, and presents them with all necessary questions to determine whether an ethics application should be approved or denied. Our project also comes with an inbuilt filter, to redirect users who indicate they're application is of high risk, to USYD's existing ethics approval platform. We also have provided features to customise forms that are required to provide to participants in research studies, such as Participant Consent Forms and Participant Information Statements. The feature that will make the most impact on the speed of completion and speed of review is our template-answer functionality. This allows the user to answer a short-answer question, using an answer from previously approved applications. These answers are clearly marked, so a reviewer can approve this question and move on without wasting time. Finally, our application is designed to be modular, such that questions can be added and removed from the database through a simple admin interface to adapt to the changing requirements of USYD's ethics committee.

The key stakeholders in our application are parties from the University of Sydney. The Computer-Human Adapted Interaction Lab (CHAI) directed the project and described their requirements. CHAI Lab will be a key stakeholder as they will also be a primary user, since the majority of research projects they undertake are of low risk. All researchers at the USYD who intend to submit their low-risk projects for ethics

approval, as well as the USYD ethics committee who review these applications are stakeholders, as they stand to benefit from the improvements that our project will make to the system. Finally, if our project is of sufficient quality to be useful to CHAI Lab, an independent team of software developers will review and adapt our program to be suitable for deployment to the University of Sydney servers. For this reason, they are stakeholders in the design of our project. Direct stakeholder interaction is likely restricted to the submission and review of ethics applications between researcher and ethics committee.

The resources used can be separated into parts: front end, back end, architecture and testing. We used HTML to build the structure of the web pages, then styled with CSS and added javascript sparingly, for intractability. Our backend was based on python, with a django framework layered on top. Django interacted with its in-built sqlite database to store our data. We took an image of our web app using Docker and uploaded it to an AWS EC2 instance, which runs our application persistently. We used the selenium package for python to perform virtual user tests.

Multiple risks have been identified that impacted the project during development. These risks include changes in project scope, end user data security and end user resistance to change. All of these present levels of risk for our group to manage.

2 System Specifications and Design

2.1 Overview

2.1.1 User

Upon opening the web app, a standard user will be greeted with a well styled login page where they will enter their username and password or register a new account ([Login/Register user stories](#)).

Sign In

Username

Password

invalid username/password

☐ Remember me

Log in

[Forgot password?](#)

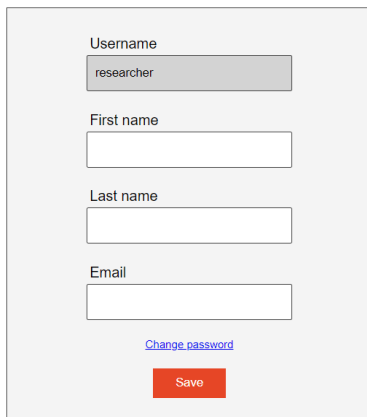
[Sign up](#)

A user without an account created will not be able to access any further material without signing in (CT0G-171). This addresses the data security portion of the client rationale and will be accepted based on whether a user can successfully log in with an account they have created and that their access is successfully restricted before logging in. After logging in, the user is redirected to their manage-application page (CT0G-193), where they can create a new application (CT0G-48), update existing applications (CT0G-53) or delete an incomplete application (CT0G-50). Further, applications can be sorted by their displayed attributes, such as: start date, last edited date, percent completion date, application ID, department of study and CI (CT0G-172). The management of applications is essential to the clients rationale of each user possibly handling multiple applications through this web app.

| <div style="display: flex; justify-content: space-between; align-items: center;"> <div> <div> THE UNIVERSITY OF SYDNEY </div> </div> <div> Home Profile Logout </div> </div> | | | | | | |
|--|-----------------|----------------|---|-------------|--------------|-----------|
| Manage Applications | | | | | | |
| <input type="text" value="Search.."/> | | | <div style="background-color: #f44336; color: white; padding: 2px 5px; text-align: center;">Start new application</div> | | | |
| Id | Project Title ▲ | Date Created ▲ | Last Modified ▲ | Status | Supervisor ▲ | Operation |
| 2 | | 29 10 2021 | Nov. 7, 2021 | APPROVED | | ⋮ |
| 3 | | 07 11 2021 | Nov. 7, 2021 | APPROVED | | ⋮ |
| 5 | Demo 2 | 07 11 2021 | Nov. 7, 2021 | DISAPPROVED | Alex | ⋮ |
| 7 | Demo 4 | 07 11 2021 | Nov. 7, 2021 | APPROVED | Jim | ⋮ |
| 8 | Demo 5 | 07 11 2021 | Nov. 7, 2021 | DISAPPROVED | Julie | ⋮ |
| 9 | | 09 11 2021 | Nov. 9, 2021 | IN PROGRESS | | ⋮ |

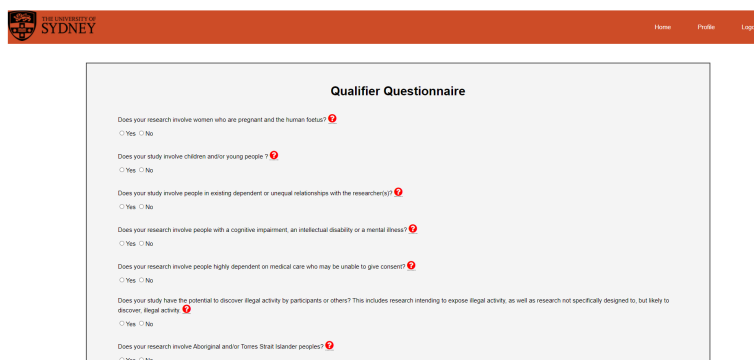
Users are able to edit their username and email from the edit profile screen (CT0G-47).

Personal Details



A screenshot of a web form titled "Personal Details". The form is contained within a light gray box. It has four input fields: "Username" (pre-filled with "researcher"), "First name", "Last name", and "Email". Below the "Email" field is a blue link that says "Change password". At the bottom of the form is a red button labeled "Save".

Clicking the start application button from the manage-applications screen will take the user through a qualifier questionnaire process, where they will be informed of the definition of “low-risk” ethics, and advised to navigate to the existing ethics approval platform if they know their application isn’t low risk. Clicking “next” will redirect the user to the qualifier questionnaire, where they will answer a series of questions to determine whether their application is “low-risk” ([CT0G-220](#)).



A screenshot of a web page titled "Qualifier Questionnaire". The page has a red header bar with the University of Sydney logo on the left and "Home Profile Logout" on the right. The questionnaire consists of six questions, each with "Yes" and "No" radio button options. The questions are: 1. "Does your research involve women who are pregnant and the human fetus?" 2. "Does your study involve children and/or young people?" 3. "Does your study involve people in existing dependent or unequal relationships with the researcher?" 4. "Does your research involve people with a cognitive impairment, an intellectual disability or a mental illness?" 5. "Does your research involve people highly dependent on medical care who may be unable to give consent?" 6. "Does your study have the potential to discover illegal activity by participants or others? This includes research intending to expose illegal activity, as well as research not specifically designed to, but likely to discover, illegal activity." 7. "Does your research involve Aboriginal and/or Torres Strait Islander peoples?"

This ultimately makes sure no user will be submitting a high-risk application, following the client’s rationale of designing a platform that facilitates low-risk ethics.

After the user passes the qualifier questionnaire and clicks “next” on the success page, the cover sheet for a new application will appear for the user to fill in details about their application ([CT0G-173](#)).

Application Form Cover Sheet

Application Form
 Complete the details on this tab, the questions on the questionnaire tab and upload study documents (e.g. Participant information statements, consent forms, questionnaires and interview questions) in the document tab. Once complete press the submit button.
 PLEASE NOTE THAT STUDENTS CANNOT BE THE CHIEF INVESTIGATOR.
 PLEASE DO NOT ADD THE CHIEF INVESTIGATION TO THE INTERNAL INVESTIGATOR LIST.

Summary
 Provide a lay summary of the project.

Protocol Title
 Protocol Title:

Chief Investigator Details
 First Named Chief Investigator Name:
 First Named Chief Investigator ID:
 School/Centre:
 Role:
 chief investigator

After entering their details into the coversheet, the user can click “save” and click the “ethics questionnaire” tab in the navigation bar, to be redirected to the questionnaire screen. Here, the user will see a section header displayed with information about the current section as well as multiple choice and short answer questions underneath. The user can find tips and guide icons next to questions, assisting the user in answering (CT0G-61). This addresses the client's rationale of speeding up the ethics submission process by assisting users if they get stuck filling out the questionnaire.

Section A

Section A is designed to distinguish between staff and student projects. In addition, this Section also seeks to identify projects that have been approved by other ethics committees.

Is this project a University of Sydney student project ONLY ?
☐ Yes
☐ No

Indicate whether this project has been or will be submitted to any other ethics committees
☐ Yes
☐ No

Section B

Section B is designed to determine whether your study falls within the National Statements definition of low or negligible risk. Throughout this section, you may be asked specific additional questions where you indicate that your study involves particular participant and/or project types. Please note that the option Possible Recruitment with reference to specific participant populations indicates that these people MAY be recruited into your study, but are not the specific population of interest. If this population is the focus of your study, you should select Yes. Please answer the following questions

Does your research involve women who are pregnant and the human foetus?
☐ Yes
☒ No

Does your study involve children and/or young people ?
☐ Yes
☒ No

Does your study involve people in existing dependent or unequal relationships with the researcher(s)?
☐ Yes
☒ No

Questions in Section B are taken straight from the qualifier questionnaire and are prefilled to avoid redundancy. Underneath short answer questions, a user will find a template selection option (CT0G-163). This allows the user to select a pre-approved answer for the question. If a pre-approved answer is selected, it will be marked in grey and will show in a separate answer box in red(CT0G-164).

Describe how you will identify and select potential participants for recruitment into the study. You should include information about how you will obtain contact details for potential participants. [?](#)

Guide: click the below button and choose the answer type, then click template answers in the textbox, it will automatically copy to the first textbox

Template Answers

Think Aloud

Crowd Sourced

Lab Study


This study relates to the evaluation of software to give students feedback on their progress, so the potential participants will include students and members of the teaching team of courses who understand student feedback systems that are currently in place or the courses whose feedback is being provided.

This study relates to the evaluation of software to give students feedback on their progress, so the potential participants will include students and members of the teaching team of courses who understand student feedback systems that are currently in place or the courses whose feedback is being provided.

Guide: Enter you own answers in the below textbox

This feature is implemented as a part of our aim, to reduce the turnaround time for applications approved. If an approver can see that an answer is provided from a pre-approved application, marking that answer will be far quicker. Questions will be shown to the user based on their answers, for example the user will be asked if participants in the study will be reimbursed. If the user answers yes, follow up questions about how they will be reimbursed will show. If the answer is no, follow up questions will not be shown (CTOG-75). This feature will address the aim of speeding up the application process for the user. The features will be deemed successful if the average speed of completion of the application is lower than the current ethics application process.

At any point throughout the application, the user can choose to select the forms tabs in the top right hand corner of the navigation bar. They are then presented with a



THE UNIVERSITY OF SYDNEY

[Home](#)
[Profile](#)
[Logout](#)

[Cover Sheet](#)
[Ethics Questionnaire](#)
[Participant Information Statement](#)
[Participant Consent Form](#)

Create Participant Information Statement

Please change and adapt the questions in the following template to suit your project. Make sure to provide answers to those questions underneath.

PIS is:

Styles

Format

B

I

U

S

Link

Image

Table

Code

Source

Name:

Faculty:

Address:

Contact Details:

Project Name:

What is the study about?

The following are important for the validity of the study:

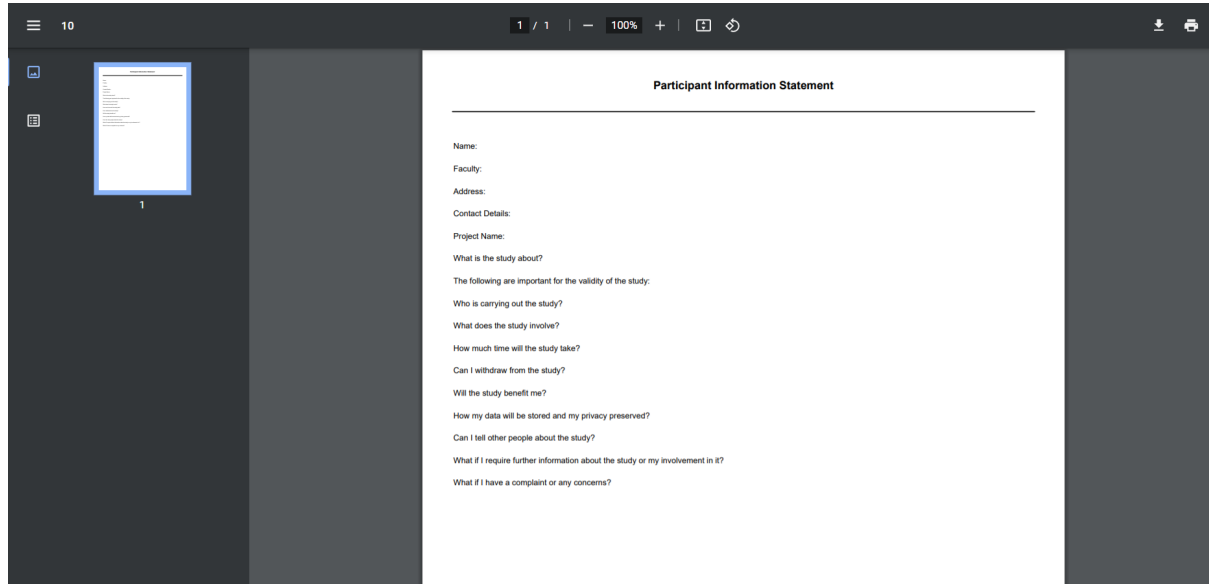
Who is carrying out the study?

Submit

template of a form from a previously approved application. They can modify this text

within a richtext box, customising the font and the format.

After the form is customised and edited, the user can save, and navigate to the view option in the form drop down, and view the form as a PDF. The form can be downloaded and distributed to participants from here.



The screenshot displays a web application interface for creating and viewing forms. On the left, a sidebar contains a document icon and a thumbnail of the current form, labeled '1'. The main content area shows a 'Participant Information Statement' form. The form includes the following fields and questions:

- Name:
- Faculty:
- Address:
- Contact Details:
- Project Name:
- What is the study about?
- The following are important for the validity of the study:
- Who is carrying out the study?
- What does the study involve?
- How much time will the study take?
- Can I withdraw from the study?
- Will the study benefit me?
- How my data will be stored and my privacy preserved?
- Can I tell other people about the study?
- What if I require further information about the study or my involvement in it?
- What if I have a complaint or any concerns?

The top of the interface features a navigation bar with a hamburger menu, a page number '10', and a toolbar with icons for zooming (1 / 1, 100%) and printing.

This process can be complete for the Participant Information Statement ([CT0G-391](#) | [CT0G-392](#)) and Participant Consent Form ([CT0G-390](#) | [CT0G-393](#)).

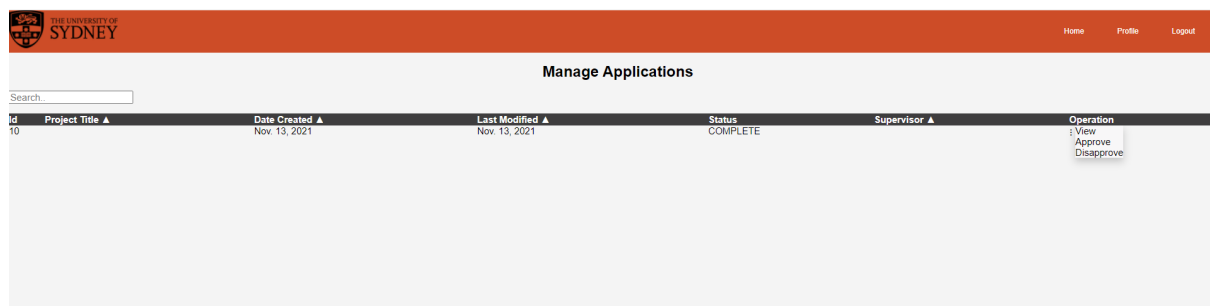
2.1.2 Administrator

An administrator has a completely different interface within the web app. The administrator has access to all elements of the database and can modify any object. This includes: questions, answers, users, groups and applications ([CT0G-135](#)). The Administrator user account is designed to adapt the web application to the changing requirements of the ethics committee, as the committee may add, change or remove questions that are featured in the questionnaire. The administrator user has no access to the regular user interface.



2.1.3 Ethics Committee

The Ethics Committee user is another type, separate to admins and regular users. The Ethics committee is only able to see submitted applications from regular users on their home-page. They have the ability to view and approve or decline applications (CT0G-167). They can also edit their profile and sort applications.



2.1.4 Full list of user stories

| ID | Description | Acceptance Criteria |
|----|--|---|
| 1 | As a user I want to be able to create an account on the web app, so that my data can be stored and seperated from other accounts | Interface for account creation is interactable If all fields are filled as per specification, account details are saved to the database Account will not be Created unless all forms filled in Account will not be Created unless password is according to specification |
| 2 | As a user, I want to edit my profile after I create my account, in case any information has changed so that my information stays up to date. | Interface for profile editing is interactable "Profile" button on home screen leads to edit profile screen On clicking "save" details are saved to database |

| | | |
|----|---|--|
| | | Username cannot be edited |
| 3 | As a user I want to be able to log in to my account so that I can manage my existing applications and start new ones | If username and password match a pair stored in the database, redirect to home page If username and password don't match, error message is thrown All available fields are interactable |
| 4 | As a user, I want to be able to start an ethics application from the home screen | "Start new application" button on home page redirects to the welcome page |
| 5 | As a user, I want to be able to see the questions that are applicable to my ethics application, so that I don't waste any time answering unnecessary questions | Section A Q1, Section D Q4, Section E Q1 and Section E Q9 all show additional questions if answered "yes" These additional questions do not show if the previous questions are unanswered or answered "no" |
| 6 | As a user, I want to be able to delete an ethics application if my research project changes scope, so that I don't waste any space on my home screen. | After clicking "withdraw" from the operations column in the home screen, application is removed from the database and cannot be seen from the home screen |
| 7 | As a user, I want to be able to start an application and continue progress at a different date and have my application's state saved, so that I don't have to finish an application in a single sitting | Clicking "save" button at the bottom of the application page, saves all data on the application to the database Application can be edited through the "edit" button on the home screen Application can only be edited if it is in the "in progress" status |
| 8 | As a user, I want to download my ethics application in the form of a PDF, so that I can keep a record of my application on my local computer | Clicking "view" button from the operations column in the home screen, redirects to a PDF version of the application PDF includes all answers entered, as well as all questions Clicking the download icon from the PDF view page downloads the PDF |
| 9 | As a user, I would like to see tips and guides while filling out the application to assist completion speed. | Hovering over orange question mark icons next to questions on the applications, presents tips to assist answers for some questions |
| 10 | As an admin, i want to be able to see a UI that that shows me all the questions in the ethics applications so that i can edit, remove and add questions. | Admin account can add/remove/edit questions without compromising the structure of the ethics application Admin has access to this interface through the admin login |
| 11 | As an ethics committee member, I want to be able to see a user interface that shows all the | Users who are members of the ethics committee group can see all applications that are "Complete" in their home page |

| | | |
|----|--|---|
| | application forms, so that I can approve or disapprove submissions. | Ethics committee members can view applications, meaning all answers that application author has filled in, show in the view Ethics committee can approve an application, removing it from their homepage. Application status changes to "approved" on application author's home page Ethics committee can deny an application, removing it from their homepage. Application status changes to "deny" on application author's home page |
| 12 | As a user, I want to be able to edit my cover sheet if my application is not submitted so that I can update my application details if anything changes. | User is able to edit their coversheet in the same interface that they edit their ethics application. Edits save to the database and can be viewed even after re-navigation to the page. Can only edit coversheet if application is in "in progress" status |
| 13 | As a user, I want to be informed of whether my application is of high risk or not, before beginning my application so that I don't waste time completing an application that is not facilitated by this web app | User is directed to qualifier questionnaire after clicking "start new application" on home screen, and selecting "next" on welcome screen Qualifier questionnaire contains questions from Section B which discriminate against high risk ethics. If User selects the correct answers (No to all answers), they are redirect to success screen and can continue with the application If User selects a single false answer, user is redirect to failure screen. |
| 14 | As a user, I want to fill out necessary information into a coversheet page, before beginning my application should be able to fill out the relevant information in the coversheet before beginning the application. | Accepted if there are fields for this information, and once "save" is clicked, it is stored in the database. And viewed/edited at a later date. |
| 15 | As a user, I want to be able to view the application start date, last edited date, percent completion date, application ID, department of study and CI from the home screen, so that I can discriminate against my existing applications | Fields are displayed on the home page and accurately represent the information entered for the coversheet, from date created/modified and status. Project title, last modified, status and supervisor change according to updates made to them |
| 16 | As a user, I want to use the website even if I am colourblind. | Able to discriminate against all text/clickable objects with colour blind filter on computer. |
| 17 | As a user, I want questions that have been answered via a pre-approved template to be indicated accordingly so that the reviewer knows what work is mine, | Ethics committee can distinguish between which part of the answer is pre-templated and which part is the user's own words |

| | | |
|----|---|---|
| | and what is templated | |
| 18 | As a user, I want to see previously approved answer's to that question and be able to enter them into my answer field, so that it is easier for me to complete my application | Sample answers are available for 80% of short answer questions (where available) Sample answer is Input into the sample answer text box on click. |
| 19 | As a user, I want to be able to access the web app from a chrome web browser. | Web app is able to be accessed from a chromium browser (The Universtiy of Sydney's recommended browser), with no http errors. |
| 20 | As a user, I want my data to be stored securely so that I don't have to worry about entering confidential information in my application | Users data cannot be accessed by other accounts Unauthenticated users cannot access data from other users |
| 21 | As a user, I want to create a participant information statement based on a provided template and style the form through a rich-text field | Template text appears in rich text field on creation of a new form. Once edited and saved to the database by pressing "save", form holds its state and can be viewed/edited later. Create participant information statement button redirects user to Create participant information form page |
| 22 | As a user, I want to create a participant consent form based on a provided template and style the form through a rich-text field | Template text appears in rich text field on creation of a new form. Once edited and saved to the database by pressing "save", form holds its state and can be viewed/edited later. Create participant consent form button redirects user to Create participant information form page |
| 23 | As a user, I want to view my PIS form as a PDF and download to my computer so that I can distribute it to my participants | User can view the same form that they save Form can be downloaded by clicking the download icon, it without data compromise. View participant information statement button redirects user to Create participant information form page |
| 24 | As a user, I want to view my PCF form as a PDF and download to my computer so that I can distribute it to my participants | User can view the same form that they save Form can be downloaded by clicking the download icon, it without data compromise. View participant consent form button redirects user to Create participant information form page |
| 25 | As a user, I want to search for an application in my home screen so I can quickly search for an application I want to view/edit | User is able to search for any of the fields attached to the application using search bar on home screen |
| 26 | As a user, I want to change my password with email assistance if I forget my password, so that I am not | User can successfully following the password reset directions and changes their password via the email reset link |

| | | |
|----|--|--|
| | locked out permanently. | Password reset is saved in the database |
| 27 | As a user, I want to sort my applications in the order of status, date created, last modified, title, supervisor | User can sort in applications order of Title User can sort in applications order of Date created User can sort in applications order of Last Modified User can sort in applications order of Supervisor |

2.2 Details:

2.2.1 Discarded User Stories

| User Story | Description | Justification | Issue Link |
|-----------------|--|--|---|
| Autosave | During the ethics questionnaire answers are autosaved | Too complex to implement due to the use of a python web-framework. | https://shyam-raj.atlassian.net/browse/CT0G-158 |
| Version control | If an application is denied. The application can be edited and resubmitted as another version. | Not enough time to complete. | https://shyam-raj.atlassian.net/browse/CT0G-162 |
| Browser timeout | During a questionnaire the browser will close after a certain period of inactivity. | Not enough time to complete. | https://shyam-raj.atlassian.net/browse/CT0G-168 |

2.2.2 Key Changes

Show/hide icon

On the login and register pages, the client requested that a user should be able to toggle their password between dots and plaintext. Our team handles this by adding an eye icon that when pressed, activates a piece of javascript that toggles the password.

Sign In

Username

Password

Hello

☐ Remember me

Log in

[Forgot password?](#)

[Sign up](#)

Generation of PCF and PIS forms

The client originally requested that the PIS and PCF forms should be automatically generated, after the completion of the ethics questionnaire, by substituting answers from the ethics questionnaire into the forms; however, during implementation we found that it did not make sense to implement it in this way as the perspective in which the answers were written differed from the way answers should be answered in the PCF and PIS forms. So, instead we created a feature in which the user could create their own PCF and PIS forms by utilizing a rich text box as shown below.

Create Participant Consent Form

Please change and adapt the questions in the following template to suit your project.

PCF rt:

Styles | Format | **B** | *I* | U | ~~S~~ | | | | | | | | | |

Name:

Faculty:

Address:

Contact Details:

Project Name:

In giving my consent I acknowledge that:

1. The procedures required for the project and the time involved have been explained to me, and any questions I have about the project have been answered to my satisfaction.
2. I have read the Participant Information Statement and have been given the opportunity to discuss the information and my involvement

Submit

Sort function

On the “Manage Applications” page a list of applications started by the user is displayed. The client requested that these applications should be able to be

THE UNIVERSITY OF SYDNEY

[Home](#)
[Profile](#)
[Logout](#)

Manage Applications

Start new application

| Id | Project Title | Date Created | Last Modified | Status | Supervisor | Operation |
|----|---------------|--------------|---------------|-------------|------------|-----------|
| 2 | | 29 10 2021 | Nov. 7, 2021 | APPROVED | | |
| 3 | | 07 11 2021 | Nov. 7, 2021 | APPROVED | | |
| 5 | Demo 2 | 07 11 2021 | Nov. 7, 2021 | DISAPPROVED | Alex | |
| 7 | Demo 4 | 07 11 2021 | Nov. 7, 2021 | APPROVED | Jim | |
| 8 | Demo 5 | 07 11 2021 | Nov. 7, 2021 | DISAPPROVED | Julie | |
| 9 | | 09 11 2021 | Nov. 9, 2021 | IN PROGRESS | | |

sorted by each column. This was implemented by a javascript algorithm that can sort data alphabetically and numerically.

2.2.3 Above and Beyond

One feature that the client did not ask for but was implemented was the forgot password feature. Our team felt that it was necessary to include this feature for the website's security. It works by asking for the user's email and sending a unique reset password link which allows the user to change their password. This adds additional security when an attacker that has access to a user's account; they will not be able to change the password unless they have access to the user's email account as well.

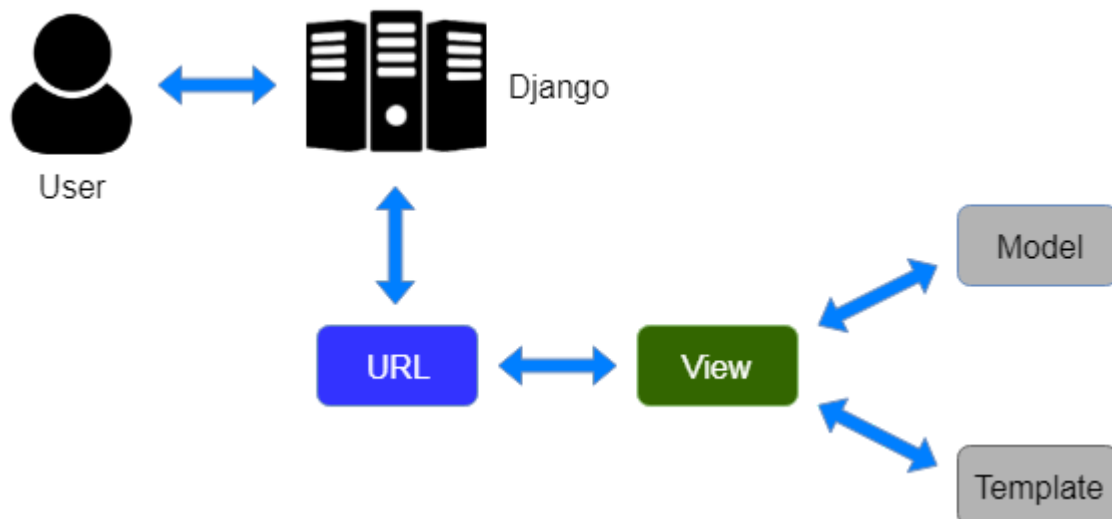
Another feature we felt was necessary to implement was the three-dot drop-down menu shown in the image above under the “Operation” column. Before this, there was a list of operations such as “edit”, “view”, “download” etc. This made the row overly cluttered as there was not enough space for all of the text. Now all of those operations are under a drop-down menu making it more readable for the user. This is shown in the image below.



2.3 System Structure Overview

The website was developed using HTML, CSS and JavaScript for the front-end and the python web-framework Django as the back-end. Django uses the MVT (Model, View, Template) design pattern. It consists of three main components: the model, view and the template.

The model is a data access layer which handles the data in the database. The template is a presentation layer which handles user interaction. The view renders the template and handles the exchange of data between the user and the model. The following graph represents the MVT control flow.



To start a Django project we use the “django-admin startproject [name of project]” command. This command auto-generates a bunch of folders and files that are

needed to establish a Django project. Here is an image of the directory structure if the project's name was "mysite".

```
mysite/  
  manage.py  
  mysite/  
    __init__.py  
    settings.py  
    urls.py  
    asgi.py  
    wsgi.py
```

The outer mysite root directory acts as the container for the project. The manage.py file is a command-line utility that lets you interact with the project in several ways such as running the website using the "python3 manage.py runserver" command. The inner mysite directory represents the python package for the project. __init__.py is just an empty file that notifies python that this directory should be considered a python package. The settings.py file handles the configuration of the python project such as defining a connection to a database. urls.py declares all of the urls that can be reached when the website is running. asgi.py and wsgi.py are not relevant for our project.

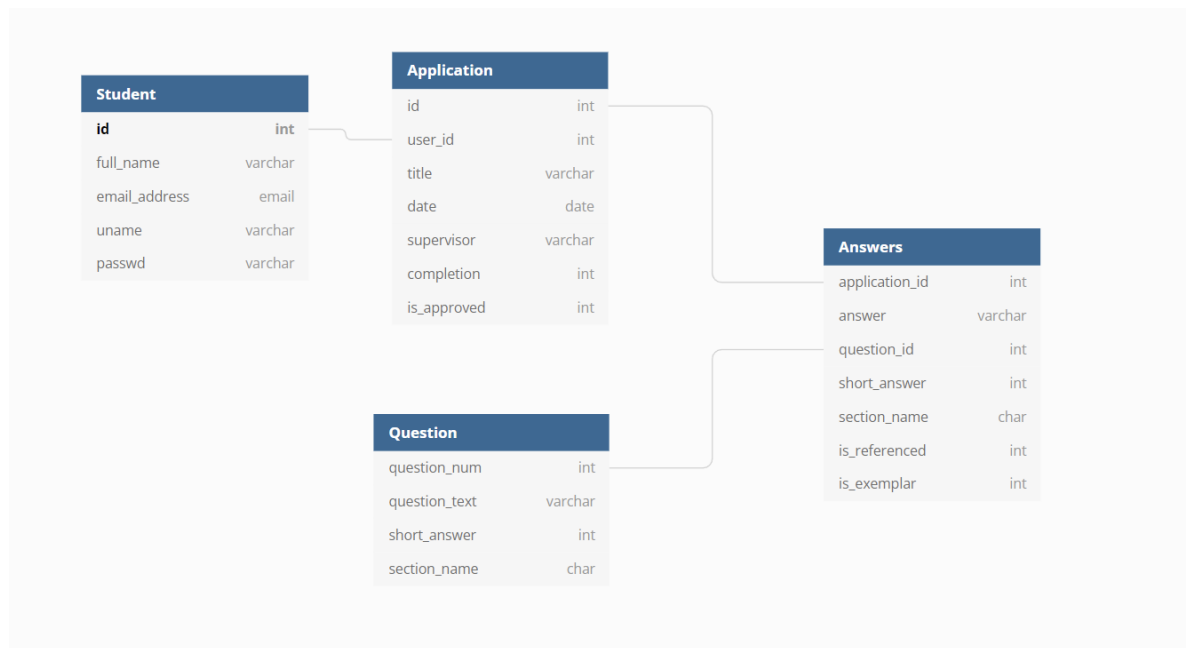
After creating the project we used the "python3 manage.py startapp [app name]" command to create an app. An app is a web application that does something - e.g. the login and register system. For each component of the website we created a new app. An app consists of a python package that follows a certain convention. Django automatically generates the basic directory structure of an application. Here is an image of the directory structure if the app was called "polls".

```
polls/
  __init__.py
  admin.py
  apps.py
  migrations/
    __init__.py
  models.py
  tests.py
  views.py
```

The models.py file houses all the models which are shown as classes. A class represents a table in the database. Each class can have multiple fields, each of which represents a column in the database. Here is an example of a model in our project.

```
class Question(models.Model):
    question_num = models.IntegerField(
        primary_key=True, unique=True, null=False)
    text = models.TextField(null=False)
    is_short_answer = models.IntegerField(null=False)
    section_name = models.CharField(max_length=1, null=False)
    tips = models.TextField(max_length=300, null=True, default='')
    is_qualifier_question = models.IntegerField(null=False)
```

Before coding all of the models, a database schema was used. Here is a screenshot of that.



To create these model tables in the database we use the command “python3 manage.py migrate”. If we had made changes to the model we run “python3 manage.py makemigrations” first. These changes are documented in the migrations folder.

To access all of the tables from the admin panel, they have to be registered in the admin.py file. Here is a screenshot of that.

```

from django.contrib import admin
from accounts.models import Application, Question, Answers, CoverSheetAnswers, CoverSheetQuestion
from django.contrib.auth.models import User, Group

admin.site.register(Application)
admin.site.register(Question)
admin.site.register(Answers)
admin.site.register(CoverSheetQuestion)
admin.site.register(CoverSheetAnswers)

```

To input large amounts of data into the database we used python scripts. Here is an example of a python script used to input all of the questions in the ethics questionnaire into the database.

```

from accounts.models import Question, CoverSheetQuestion

Question.objects.bulk_create([
    Question(question_num=101, text = 'Is this project a University of Sydney student project ONLY ?',
    is_short_answer=0, section_name='A', tips = 'Tips: i.e. ethics application restricted to the activities of the
    student research project', is_qualifier_question = 0),
    Question(question_num=102, text = 'Select appropriate student classification:', is_short_answer=1, section_name=
    tips = 'Tips: ', is_qualifier_question = 0),
    Question(question_num=103, text = 'Indicate whether this project has been or will be submitted to any other etl
    committees', is_short_answer=0, section_name='A', tips = 'Tips: ', is_qualifier_question = 0),
    Question(question_num=201, text = 'Does your research involve women who are pregnant and the human fetus?',
    is_short_answer=0, section_name='B', tips = 'Tips: ', is_qualifier_question = 1),
    Question(question_num=202, text = 'Does your study involve children and/or young people ?', is_short_answer=0,
    section_name='B', tips = 'Tips: i.e. younger than 18 years', is_qualifier_question = 1),
    Question(question_num=203, text = 'Does your study involve people in existing dependent or unequal relationship
    with the researcher(s)?', is_short_answer=0, section_name='B', tips = 'Tips: ', is_qualifier_question = 1),
    Question(question_num=204, text = 'Does your research involve people with a cognitive impairment, an intellectu
    disability or a mental illness?', is_short_answer=0, section_name='B', tips = 'Tips: ', is_qualifier_question =
    Question(question_num=205, text = 'Does your research involve people highly dependent on medical care who may b
    unable to give consent?', is_short_answer=0, section_name='B', tips = 'Tips: ', is_qualifier_question = 1),
    Question(question_num=206, text = 'Does your study have the potential to discover illegal activity by particip
    or others? This includes research intending to expose illegal activity, as well as research not specifically
    designed to, but likely to discover, illegal activity.', is_short_answer=0, section_name='B', tips = 'Tips: ',
    is_qualifier_question = 1),
])

```

Some apps contain a forms.py file. This file is for creating unique forms or modifying inbuilt forms. Forms are used to take input from the user and to perform some kind of operation in the database. Here is an example of a form used for registering a user.

```

class CreateUserForm(forms.UserCreationForm):
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.fields['first_name'].required = True
        self.fields['last_name'].required = True
        self.fields['email'].required = True

    class Meta:
        model = User
        fields = ['first_name', 'last_name', 'username',
                  'email', 'password1', 'password2']

```

The views.py handles web requests and returns a web response. Every function in the views.py file encapsulates a block of logic which handles input/output from the database and renders a template. Each function is mapped to a url in the urls.py file. As an example we will use the editUserPage function shown below.

```

@login_required(login_url=loginPage)
@allowed_users(allowed_roles=['researcher', 'staff'])
def editUserPage(request):
    form = UpdateUserForm(instance=request.user)
    user = request.user
    if request.method == "POST":
        form = UpdateUserForm(request.POST, instance=request.user)
        if form.is_valid():
            form.save()
            return redirect('editprofile')

    context = {'form': form, 'user': user}
    return render(request, 'edit_profile.html', context)

```

This function is mapped to a url and activates when that url is called. The url in urls.py is shown below.

```

path('edit_profile/', views.editUserPage, name='editprofile'),

```

The editUserPage function then renders the template “edit_profile” as shown in the image. All templates used in an app are stored in a templates folder. If the template uses a stylesheet or javascript those files are stored in a static folder. To use the files in the static folder the html file needs to use the { % load static % } function. An example is shown below.

```

{% load static %}
<link rel="stylesheet" type="text/css" href="{% static 'login/style.css' %}">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.4/css/all.min.css">
<script type="text/javascript" src="{% static 'login/show-hide.js' %}"></script>

```

Finally, the tests.py file is used for performing unit tests for the application. Here is an example of a unit test for the accounts app.


```

class AccountsTest(TestCase):

    @classmethod
    def setUpTestData(cls):

        researcher = Group(name = "researcher")
        staff = Group(name = "staff")
        staff.save()
        researcher.save()
        cls.user = User.objects.create_user(username = 'john', password='johnpassword')
        cls.admin_staff = User.objects.create_user(username='admin', password='adminpassword')
        cls.admin_staff.groups.add(staff)
        cls.user.groups.add(researcher)
        cls.factory = RequestFactory()

```

3 Quality of Work

3.1 Test Plan

From the criteria, we have seen that there are different kinds of testing techniques that can be used in different requirements. These techniques can be divided into functional testing and non-functional testing.

For the functional testing, we have implemented unit tests, integration testing, regression testing, system testing and user acceptance testing. For the non-functional testing, we have implemented the usability testing.

First, we implemented the unit tests. It is important for test coverage in our project. Also, there are some functions that are used to update the database and redirect pages. We did not test the functions of converting Pdf, since it comes from external libraries. The details of unit testing are in the appendix. The unit tests are done by the django testing framework, which inherits from python unittest package. The testing framework has some additional assertions, such as checking small amounts of the html and the validation of forms. This can be used in checking some test objects like models or templates.

```

class AccountsTest(TestCase):

    @classmethod
    def setUpTestData(cls):

        researcher = Group(name = "researcher")
        staff = Group(name = "staff")
        staff.save()
        researcher.save()
        cls.user = User.objects.create_user(username = 'john', password='johnpassword')
        cls.admin_staff = User.objects.create_user(username='admin', password='adminpassword')
        cls.admin_staff.groups.add(staff)
        cls.user.groups.add(researcher)
        cls.factory = RequestFactory()

    def testNoAllInput(self):

        #No email

        data = {'first_name': 'Test', 'last_name': 'Test', 'username': "Test", 'password1':

        form = CreateUserForm(data)

        self.assertFalse(form.is_valid(), "The test did not pass")

```

Then, for some sequential procedures, we used integration testing to help us combine different test cases. We used equivalence partitioning and boundary analysis to help us design test cases. For example, the checking of text fields and redirection can be combined together when all the inputs are correct. This enables us to reduce the amount of test cases.

Then we will automatically run the previous test cases before we submit our new tasks. The regression testing makes sure that our new contribution will not break the other parts. The old test cases can be changed if required. Also, since we use CI/CD for running automatic testing, which will run test cases from different applications, the system testing was implemented in this way.

Then, Acceptance testing is our final functional testing. It checks the criteria we set before to make sure that the web application works as the client desires.

For the non-functional testing, we try to implement the usability testing by giving project prototypes to students who are working with the ethic applications. In addition, without the users, we set up a bunch of virtual user testing by using the selenium package for each application in the project. Also, we set up some tasks for testers to run, and these tasks come from the criteria. It is essential to test whether our website is easy to use and users can work on their tasks without any difficulty.

We have not implemented the API testing, UI testing, stress testing and performance testing. We have not much experience with UI design and testing. For the API testing, Since we use the django framework, which database and application are separated. In addition, since our website has not many users and not a big scale of it, we did not find a way to check stress and performance.

Also, we have not implemented much security testing since the django framework provides developers with most of the protection functions. For example, we can install many middlewares from the django framework, such as csrf and clickjacking middlewares, which will apply built-in functions to prevent csrf attack and hijacking attack.

```
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
```

Also, changing the setting in “setting.py”, such as changing DEBUG to False, can hide the error messages on the webpage when it fails to load. In addition, since the django framework provides API for the database access, which it uses model and fields to represent entities and attributes, it can prevent css attack.

3.2 Test class

Since our project has different applications, each application is associated with several user stories. Therefore, for each application, there is a test suite, which is written in the “tests.py” file. There are 10 applications, which are “accounts”, “approvelist”, “coversheet”, “managelist”, “qualifier”, “questionnaire”, and “viewforms”.

Each application includes several user stories. For each requirement, we have tested several objects which are listed below.

3.3 Test objects

The test objects are elements that have been tested by using different testing techniques, such as unit tests, integration testing, regression testing, usability testing and acceptance testing. In a test suite of an application, Several test objects have been tested to achieve acceptance criteria.

3.3.1 Context

Context is a dictionary with data from the database. It can be passed to the templates, which are used to insert it to HTML pages. The data could be model instances after retrieving from the database and filtering. Therefore, what we have tested is checking whether filtering by some functions is correct and the templates can get all instances they need. Context is easily shown on the webpage, we can check it by acceptance testing.

3.3.2 Model

A model contains the essential fields and behaviours of the data we store. Generally, each model maps to a single database table. In other words, it converts the classes and text fields to entities and attributes in the database. What we have tested is checking whether all the constraints of fields in models work successfully. These models have been tested by unit tests.

3.3.3 Database

Database is updated when users make post requests or get requests. What we have tested is checking whether the update operation is correct. It involves different apps in our project and different unit functions. Therefore, we have tested our database with the request method and response

objects. We combine assertions of request methods and the database together, which is incremental integration testing.

3.3.4 Request / Response

The main request methods in our project are post and get. These methods involve webpage redirection, data update etc. Therefore, what we have tested is checking the status code and url after redirection, also the instances in the database. We also use incremental integration testing for this module.

3.3.5 Template

The template is used for showing the text information which is retrieved from the database by several functions and it combines these data with static text, styling and also some dynamic behaviours. Context, which is passed to the template, is a dictionary with data from the database. All templates use jinja2 to combine the context with html, which is hard to test by using unit testing. We have not tested the tags and filters of the jinja2 language since it is a part of the django framework. Therefore, what we have tested is checking the contents of the HTML page and also the context of a template.

3.3.6 Styling

The styling of our web application consists of the layout of block elements, the text elements and the colour combination, which are tested by usability testing and acceptance testing. Since our web application provides users with several different forms and a questionnaire, it is important to design a clear layout of a large amount of text information. Also, good colour combinations can enhance readability. This enables users to read the forms and questionnaire and fill in their information better in the web application. Over the last few weeks clients gave some feedback to us about the styling based on her experiences. The project also will be given to two testers who are Phd students and need to use the web application later. The feedback from them can also become acceptance criteria.

3.3.7 Behaviour

The behaviour of web pages in our projects means the update of contents in the web pages. For example, when people search an application, the webpage will update and it returns the result. Most of the behaviours are written by javascript, since they do not need to interact with the database. We have not tested them by using unit tests. However, these behaviours can be tested by usability testing and acceptance testing.

3.4 Automatic Testing

Django provides us with a testing framework, which inherits from the python unittest. It can run all the tests in the apps of our project. In addition, we use CI/CD on bitbucket to help us run the test automatically. When developers submit their work, the pipeline will run the tests and check whether the work passes all tests. Therefore, it is a kind of regression testing since the new code not only needs to be tested with new tests but also needs to be tested with other tests. There is an example of the process.

| | | | | | |
|------|----|---|--------------|------------|--------|
| #421 | ZG | Add extra field to view for normal and staff view zhixin gao 49602a1 answers | ✓ Successful | 7 days ago | 52 sec |
| #420 | ZG | Add extra field to view for normal and staff view zhixin gao 6d1ac3a answers | ✓ Successful | 8 days ago | 22 sec |
| #419 | ZG | Add extra field to view for normal and staff view zhixin gao 453d5b6 answers | ✓ Successful | 8 days ago | 23 sec |
| #418 | ZG | Add extra field to view for normal and staff view zhixin gao bfaae8e answers | ❌ Failed | 8 days ago | 27 sec |
| #417 | ZG | Add extra field to view for normal and staff view zhixin gao d95289d answers | ✓ Successful | 8 days ago | 22 sec |
| #416 | ZG | Merged in admintests (pull request #101) added admin virtual ... zhixin gao bfcda06 master | ⏸ Paused | 8 days ago | 0 sec |
| #415 | ZG | Merged master into admintests zhixin gao 3c6681e admintests | ⏸ Paused | 8 days ago | 0 sec |

3.5 Testing Design

We use different testing techniques to help us design the test cases before we start our new user stories, such as partitioning, boundary analysis etc. Also, we apply this technique in different testing, such as unit tests, integration testing etc. The table of equivalence partitioning is shown below.

3.5.1 Equivalence partitioning and boundary analysis

We tried to combine similar output to one test case. For example, when we have to register the Accounts, we should test the input for the form. Here is the list for equivalence partitioning.

| Fields | Positive | Negative |
|----------|--|---|
| email | <ul style="list-style-type: none"> Contains @ and Contains .com at the end | <ul style="list-style-type: none"> No contains @ No .com at the end |
| password | <ul style="list-style-type: none"> Mix of integer and letter and More than 8 digits Equal to 8 digit and mix of integer and letter | <ul style="list-style-type: none"> Entire integer and less than 8 digits |
| Username | <ul style="list-style-type: none"> Only allow digit, letter, +/=/ | <ul style="list-style-type: none"> Have one of the special character |

From the list we can see that we need to meet all positive cases to create a successful account. Therefore, we can design a test case which combines all positive cases together. However, for the password field, we cannot test “equal to 8 digit” and “more than 8 digit” together, Therefore, we should make 2 test cases for positive fields. It can be calculated as the $1 \times 2 \times 1 = 2$ cases. Therefore, the test will fail if one negative situation (such as no .com) happens. For the negative cases, we can make 2 test cases for verification. Here are examples of the cases

Negative cases

```

def test_registePage_wrongpost1(self):

    data = {'first_name': 'Test', 'last_name': 'Test', 'username': "Test", 'email': 'hello@gmail', 'password1': '123'

    request = self.factory.post('/register/', data, follow=True)

    request.user = AnonymousUser()

    response = registerPage(request)

    self.assertEqual(response.status_code, 200)

    self.assertContains(response=response, text='<title>Register</title>', html=True)

def test_registePage_wrongpost2(self):

    data = {'first_name': 'Test', 'last_name': 'Test', 'username': "Test", 'email': 'hellogmail.com', 'password1': '123'

    request = self.factory.post('/register/', data, follow=True)

    request.user = AnonymousUser()

    response = registerPage(request)

    self.assertEqual(response.status_code, 200)

    self.assertContains(response=response, text='<title>Register</title>', html=True)

```

Positive cases

```

def test_registerPage_post1(self):

    data = {'first_name': 'Test', 'last_name': 'Test', 'username': "T1st+-", 'email': 'hello@gmail.com', 'password1': '123'

    request = self.factory.post('/register/', data, follow=True)

    request.user = AnonymousUser()

    response = registerPage(request)

    self.assertEqual(response.headers['Location'], '/login/')
    self.assertEqual(response.status_code, 302)

    testers = User.objects.filter(username="T1st+-")

    self.assertEqual(len(testers), 1)
    self.assertEqual(testers[0].first_name, "Test")
    self.assertEqual(testers[0].last_name, "Test")
    self.assertEqual(testers[0].email, 'hello@gmail.com')
    self.assertEqual([testers[0].groups.all()[0].name, 'researcher'])

def test_registerPage_post2(self):

    data = {'first_name': 'Test', 'last_name': 'Test', 'username': "T1st+-", 'email': 'hello@gmail.com', 'password1': '123'

    request = self.factory.post('/register/', data, follow=True)

    request.user = AnonymousUser()

```

However, most of our functions receive requests as parameters, which are not necessary to test. Also, most of our forms have no much constraints on the input. Therefore, we will apply these techniques for personal information settings. The functions of it involve register, editing profile.

3.6 Test result

The test cases run together and the test result is listed below. The test cases involves the test objects such as forms, models, request methods etc. Some test objects are combined in one test case.

```
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.....
-----
Ran 21 tests in 2.116s

OK
Destroying test database for alias 'default'...
```

3.7 Test coverage

The test coverage covers branches in 10 applications. It achieves over 80 % , which meets our requirements. Since some of our user stories are about viewing forms and downloading pdf, which is not necessary to be tested by unit tests. These can be done by using acceptance testing or usability testing. For each application, the test suites have tested most of the functions in “view.py” and also checked the url redirection.

| Name | Stmts | Miss | Cover |
|-------------------------------------|-------|------|-------|
| PISform/__init__.py | 0 | 0 | 100% |
| PISform/apps.py | 4 | 0 | 100% |
| PISform/tests.py | 1 | 0 | 100% |
| PISform/urls.py | 4 | 0 | 100% |
| PISform/utils.py | 12 | 7 | 42% |
| PISform/views.py | 18 | 7 | 61% |
| Project/__init__.py | 0 | 0 | 100% |
| Project/admin.py | 3 | 0 | 100% |
| Project/apps.py | 3 | 0 | 100% |
| Project/asgi.py | 4 | 4 | 0% |
| Project/settings.py | 26 | 0 | 100% |
| Project/urls.py | 3 | 0 | 100% |
| Project/wsgi.py | 4 | 4 | 0% |
| accounts/__init__.py | 0 | 0 | 100% |
| accounts/admin.py | 8 | 0 | 100% |
| accounts/apps.py | 4 | 0 | 100% |
| accounts/decorators.py | 21 | 5 | 76% |
| accounts/forms.py | 22 | 0 | 100% |
| accounts/migrations/0001_initial.py | 8 | 0 | 100% |
| accounts/migrations/__init__.py | 0 | 0 | 100% |
| accounts/models.py | 52 | 1 | 98% |
| accounts/tests.py | 106 | 0 | 100% |
| accounts/urls.py | 4 | 0 | 100% |
| accounts/views.py | 53 | 1 | 98% |
| add_questions.py | 3 | 3 | 0% |
| add_templates.py | 6 | 6 | 0% |
| approvelist/__init__.py | 0 | 0 | 100% |
| approvelist/apps.py | 4 | 0 | 100% |
| approvelist/tests.py | 45 | 0 | 100% |
| approvelist/urls.py | 3 | 0 | 100% |
| approvelist/views.py | 33 | 7 | 79% |
| consentform/__init__.py | 0 | 0 | 100% |
| consentform/apps.py | 4 | 0 | 100% |
| consentform/tests.py | 24 | 16 | 33% |
| consentform/urls.py | 4 | 0 | 100% |
| consentform/utils.py | 12 | 7 | 42% |
| consentform/views.py | 17 | 6 | 65% |
| coversheet/__init__.py | 0 | 0 | 100% |
| coversheet/apps.py | 4 | 0 | 100% |
| coversheet/forms.py | 21 | 0 | 100% |
| coversheet/tests.py | 64 | 0 | 100% |
| coversheet/urls.py | 6 | 0 | 100% |
| coversheet/views.py | 95 | 5 | 95% |
| manage.py | 12 | 2 | 83% |
| managelist/__init__.py | 0 | 0 | 100% |
| managelist/apps.py | 4 | 0 | 100% |

| | | | |
|----------------|------|-----|------|
| approve/ | 45 | 0 | 100% |
| approve/ | 3 | 0 | 100% |
| approve/ | 33 | 7 | 79% |
| consentform/ | 0 | 0 | 100% |
| consentform/ | 4 | 0 | 100% |
| consentform/ | 24 | 16 | 33% |
| consentform/ | 4 | 0 | 100% |
| consentform/ | 12 | 7 | 42% |
| consentform/ | 17 | 6 | 65% |
| coversheet/ | 0 | 0 | 100% |
| coversheet/ | 4 | 0 | 100% |
| coversheet/ | 21 | 0 | 100% |
| coversheet/ | 64 | 0 | 100% |
| coversheet/ | 6 | 0 | 100% |
| coversheet/ | 95 | 5 | 95% |
| manage/ | 12 | 2 | 83% |
| managelist/ | 0 | 0 | 100% |
| managelist/ | 4 | 0 | 100% |
| managelist/ | 51 | 0 | 100% |
| managelist/ | 4 | 0 | 100% |
| managelist/ | 12 | 7 | 42% |
| managelist/ | 31 | 8 | 74% |
| qualifier/ | 0 | 0 | 100% |
| qualifier/ | 4 | 0 | 100% |
| qualifier/ | 21 | 0 | 100% |
| qualifier/ | 1 | 0 | 100% |
| qualifier/ | 3 | 0 | 100% |
| qualifier/ | 77 | 18 | 77% |
| questionnaire/ | 0 | 0 | 100% |
| questionnaire/ | 4 | 0 | 100% |
| questionnaire/ | 1 | 0 | 100% |
| questionnaire/ | 3 | 0 | 100% |
| questionnaire/ | 54 | 44 | 19% |
| upload/ | 0 | 0 | 100% |
| upload/ | 4 | 0 | 100% |
| upload/ | 0 | 0 | 100% |
| upload/ | 1 | 0 | 100% |
| upload/ | 3 | 0 | 100% |
| upload/ | 3 | 1 | 67% |
| viewforms/ | 0 | 0 | 100% |
| viewforms/ | 4 | 0 | 100% |
| viewforms/ | 10 | 0 | 100% |
| viewforms/ | 0 | 0 | 100% |
| viewforms/ | 1 | 0 | 100% |
| viewforms/ | 4 | 0 | 100% |
| viewforms/ | 12 | 7 | 42% |
| viewforms/ | 31 | 21 | 32% |
| TOTAL | 1060 | 187 | 82% |

3.8 Acceptance Testing

| No. | Requirements\Acceptance Testing | 1 | 2 | 3 | 4. | 5 |
|-----|--|------------------------------------|--|---|--|---|
| 1.1 | Interface for account creation is interactable | Users can fill in the form | When users click the button, if all the fields are filled in correctly, then the page will redirect to "login" | When users click the button, if one of the fields is wrong in format, then the page will give wrong message | The webpage can show the password when you click the "eye icon", then when you click again, the password will hide | When users click "Go back", it will redirection to the page |
| 1.2 | If all fields are filled as per specification, | One user instance has been created | The user instance is in a group which | | | |

| | | | | | | |
|-----|---|---|---|-----------------|--|--|
| | account details are saved to the database | in database, and it has the info of username, password, email, firstname, and lastname | is call "researcher" | | | |
| 1.3 | Account will not be Created unless all forms filled in | When users click the button, if one of the fields is wrong in format, then the page will give wrong message | No new user instance is in the database. | | | |
| 1.4 | Account will not be Created unless password is according to specification | When users click the button, if password field is wrong in format, then the page will give wrong message | | | | |
| 2.1 | Interface for profile editing is interactable | Users can fill in the form | When users click the "save", it will save information they fill in on the webpage | | | |
| 2.2 | "Profile" button on home screen leads to edit profile screen | "Profile" button lead to the page | "Profile" button are in every page, except for login, register page | | | |
| 2.3 | On clicking "save" details are saved to database | The database update the details for every fields | | | | |
| 3.1 | If username and password match a pair stored in the database, redirect to home page | Authentication works | It will redirect to home page if it is successful | | | |
| 3.2 | If username and password don't match, error message is thrown | Authentication fails | It will give error message if it fails | | | |
| 3.3 | All available fields are interactable | The form can be filled in | | | | |
| 4.1 | "Start new application" button | "Start new application" | When users click button, it | Only researcher | | |

| | | | | | | |
|-----|---|--|--|--|--|--|
| | on home page redirects to the welcome page | button is clickable | will redirect to the "welcome" page | can see the "start new application" button | | |
| 5.1 | Section A Q1, Section D Q4, Section E Q1 and Section E Q9 all show additional questions if answered "yes" | When users answer "yes", these questions the page will show additional questions | | | | |
| 5.2 | These additional questions do not show if the previous questions are unanswered or answered "no" | When users answer "no" on one of the questions, the additional questions of that one will not be shown | | | | |
| 6.1 | After clicking "withdraw" from the operations column in the home screen, application is removed from the database and cannot be seen from the home screen | Withdraw button should be in 3 dot drop down menu on managelist page | Each application should have withdraw functions | When users click withdraw button, this application will be removed from database | Once the application is removed, it will not be shown on the list page | |
| 7.2 | Application can be edited through the "edit" button on the home screen | "Edit" button should be in 3 dot drop down menu on managelist page | Each application should have "edit" function | When you click "edit", it will redirect it to the "questionnaire" page | | |
| 7.3 | Application can only be edited if it is in the "in progress" status | For each application, when the status is "IN PROGRESS", the menu will have "edit" operation | For each application, if the status is not "IN PROGRESS", there is no "edit" operation in 3 dot drop down menu | | | |
| 8.1 | Clicking "view" button from the operations column in the home screen, redirects to a PDF version of the application | For each application, if the status is not "IN PROGRESS", the menu will have "view" operation | When you click "view" operation, the home page will redirect to a PDF version of the application | The PDF version can be shown | | |
| 8.2 | PDF includes all answers entered, as well as all questions | All questions are included | For each questions there is an answer | The answer can be none if the field hasn't been filled in | | |
| 8.3 | Clicking the download icon from | Can download pdf on view | | | | |

| | | | | | | |
|------|--|--|---|--|--|--|
| | the PDF view page downloads the PDF | page | | | | |
| 9.1 | Hovering over orange question mark icons next to questions on the applications, presents tips to assist answers for some questions | There are mark icons for some questions | When the mouse hovers over the icon, it will show tips | Tips can be inserted/deleted by admin | | |
| 10.1 | Admin account can add/remove/edit questions without compromising the structure of the ethics application | Admin accounts can add/edit remove questions on admin page | The database will update once they save the change | The pages of questionnaire will update the questions | | |
| 10.2 | Admin has access to this interface through the admin login | User can go to admin page by url "localhost/login" | Admin page will check the username and password | | | |
| 11.1 | Users who are members of the ethics committee group can see all applications that are "Complete" in their home page | For each application, the id, title, supervisor, date, status and operation menu will be shown | All applications which status is "Complete" will be shown on "approvelist" page | | | |
| 11.2 | Ethics committee members can view applications, meaning all answers that application author has filled in, show in the view | View operation is in 3 dot drop down menu for all applications in the list | When users click "view", it will redirect to view page | In view page, all answers are shown | | |
| 11.3 | Ethics committee can approve an application, removing it from their homepage. Application status changes to "approved" on application author's home page | Approve operation is in menu | When you click "approve", the page will not show this application | The status of this application will change to "approved" in the database | | |
| 12.1 | Users are able to edit their coversheet in the same interface that they edit their ethics application. | The questionnaire can redirect to cover sheet page without quitting | The cover sheet page can redirect to questionnaire without quitting | | | |
| 12.2 | Edits save to the database and can be viewed even after re-navigation to the page. | | | | | |

| | | | | | | |
|------|--|---|--|---|--|--|
| 12.3 | Can only edit cover sheet if application is in "in progress" status | The edit function will only show in the menu when the status is "IN PROGRESS" | | | | |
| 13.1 | User is directed to qualifier questionnaire after clicking "start new application" on home screen, and selecting "next" on welcome screen | Redirect from "homepage" to "qualifier" | Redirect from "qualifier" to "welcome" | The redirection should happen when users click the button | | |
| 13.3 | If User selects the correct answers (No to all answers), they are redirect to success screen and can continue with the application | Redirection from qualifier questionnaire to "success" page | Pre-condition: answers should be correct | | | |
| 13.4 | If User selects a single false answer, the user is redirected to the failure screen. | Redirection from qualifier questionnaire to "fail" page | Pre-condition: answers are not correct | | | |
| 14.1 | Accepted if there are fields for this information, and once "save" is clicked, it is stored in the database. And viewed/edited at a later date. | When you edit and then save, the data will change | | | | |
| 15.1 | Fields are displayed on the home page and accurately represent the information entered for the cover sheet, from date created/modified and status. | Date and status are shown on the "managelist" page when an application is created (similar to 11.1) | | | | |
| 15.2 | Project title, last modified, status and supervisor change according to updates made to them | These fields can be updated in coversheet page | | | | |
| 16.1 | Able to discriminate against all text/clickable objects with a colour blind filter on the computer. | | | | | |

| | | | | | | |
|------|---|--|--|--|--|--|
| 17.1 | Ethics committee can distinguish between which part of the answer is pre-templated and which part is the user's own words | Users know where the pre-templated answer | Users know where is the answer they have written | | | |
| 18.1 | Sample answers are available for 80% of short answer questions (where available) | Sample answers are available | | | | |
| 18.2 | Sample answer is Input into the sample answer text box on click. | Sample answers can be typed | | | | |
| 21.1 | Template text appears in rich text field on creation of a new form. | Template text in the database | Template text can be edited | | | |
| 21.2 | Once edited and saved to the database by pressing "save", form holds its state and can be viewed/edited later. | "Save" button is clickable | | | | |
| 21.3 | Create participant information statement button redirects user to Create participant information form page | Check the redirection | | | | |
| 22.1 | Template text appears in rich text field on creation of a new form. | Users can edit rich text | | | | |
| 22.2 | Once edited and saved to the database by pressing "save", form holds its state and can be viewed/edited later. | "Save" button is clickable | | | | |
| 22.3 | Create participant consent form button redirects user to Create participant information form page | Users can create consent form by clicking the button | | | | |
| 24.2 | Form can be downloaded by | Users can | | | | |

| | | | | | | |
|------|---|--|--|--|--|--|
| | clicking the download icon, without data compromise. | download the forms | | | | |
| 24.3 | View participant consent form button redirects user to Create participant information form page | User can view the forms | | | | |
| 25.1 | User is able to search for any of the fields attached to the application using search bar on home screen | User can search the application | | | | |
| 26.1 | User can successfully following the password reset directions and changes their password via the email reset link | User can change their password by following the instructions | | | | |
| 26.2 | Password reset is saved in the database | Database has been changed | | | | |
| 27.1 | User can sort in applications order of Title | User click the sort arrow of title | | | | |
| 27.2 | User can sort in applications order of Date created | User click the sort arrow of Date created | | | | |
| 27.3 | User can sort in applications order of Last Modified | User click the sort arrow of Last Modified | | | | |
| 27.4 | User can sort in applications order of Supervisor | User click the sort arrow of supervisor | | | | |

Details:

| | |
|-----|---|
| 1.1 | <ul style="list-style-type: none"> 1, 2, 5 - RegisterTests folder in VirtualUser 3, 4 - usability testing with the client |
| 1.2 | <ul style="list-style-type: none"> 1, 2 - test_regiesterPage_post in the ./Project/accounts/tests.py |
| 1.3 | <ul style="list-style-type: none"> 1 - RegisterTests folder in VirtualUser 2 - test_registePage_wrongpost1,2 in the ./Project/accounts/tests.py |

| | |
|------|---|
| 1.4 | <ul style="list-style-type: none"> • 1 - RegisterTests folder in VirtualUser |
| 2.1 | <ul style="list-style-type: none"> • 1,2 - ProfileTests folder in VirtualUser |
| 2.2 | <ul style="list-style-type: none"> • 1,2 - ProfileTests folder in VirtualUser • 3 - usability Tests |
| 2.3 | <ul style="list-style-type: none"> • 1,2 - ProfileTests folder in VirtualUser, testEditUserPage in the ./Project/accounts/tests.py |
| 3.1 | <ul style="list-style-type: none"> • 1,2 - login functions in the ./Project/accounts/tests.py |
| 3.2 | <ul style="list-style-type: none"> • 1 - login functions in the ./Project/accounts/tests.py • 2 - login test folder in VirtualUser |
| 3.3 | <ul style="list-style-type: none"> • Usability Testing |
| 4.1 | <ul style="list-style-type: none"> • 1, 2, 3 - ManagelistTest folder in VirtualUser |
| 5.1 | <ul style="list-style-type: none"> • Usability testing |
| 5.2 | <ul style="list-style-type: none"> • Usability testing |
| 6.1 | <ul style="list-style-type: none"> • Usability testing, and also test_delete in the ./Project/managelist/tests.py |
| 7.2 | <ul style="list-style-type: none"> • Usability testing |
| 7.3 | <ul style="list-style-type: none"> • Usability testing |
| 8.1 | <ul style="list-style-type: none"> • Usability testing |
| 8.2 | <ul style="list-style-type: none"> • Usability testing |
| 8.3 | <ul style="list-style-type: none"> • Usability testing |
| 9.1 | <ul style="list-style-type: none"> • Usability testing |
| 10.1 | <ul style="list-style-type: none"> • adminTest folder in the virtualuser |
| 10.2 | <ul style="list-style-type: none"> • adminTest folder in the virtualuser |
| 11.1 | <ul style="list-style-type: none"> • mangementTest folder in the virtualuser • Usability testing |
| 11.2 | <ul style="list-style-type: none"> • ApproveTest folder in the virtualuser |
| 11.3 | <ul style="list-style-type: none"> • ApproveTest folder in the virtualuser • testApprove/testDisapprove in the ./Project/approvelist/tests.py |
| 12.1 | <ul style="list-style-type: none"> • Usability testing |
| 13.1 | <ul style="list-style-type: none"> • Usability testing |
| 13.3 | <ul style="list-style-type: none"> • QualifiersTest folder in the virtualuser |

| | |
|--------------|--|
| 13.4 | <ul style="list-style-type: none"> • QualifiersTest folder in the virtualuser |
| 15.1 15.2 | <ul style="list-style-type: none"> • Usability testing |
| 17.1 | <ul style="list-style-type: none"> • Usability testing |
| 18.1 18.2 | <ul style="list-style-type: none"> • Usability testing |
| 19.1 | <ul style="list-style-type: none"> • Usability testing |
| 26 | <ul style="list-style-type: none"> • Usability testing |
| 27 | <ul style="list-style-type: none"> • Usability testing |

3.9 Test Summary

We have made unit tests and acceptance tests. The limitation will be that we have no much experience on testing the front-end side and dynamic animation of web pages. What we can do is to use virtual user testing and usability testing to make sure every template is correct. Also, due to the time limit, we can not do further testing on the performance, stress and security.

3.10 Tools used to build the system.

This may include:

What tools, and why used, decisions, rationale

Languages, frameworks used for each part:

Team decided to use Django as the framework of the project, Python as back-end and HTML, CSS, Javascript as front-end. Team chose Django because there are some advantages using Django rather than other popular frameworks. Firstly and the most important is that Django uses the Python language, which is suitable for all the team members because all team members prefer using Python over other programming languages, and one of our team members is familiar with Django. That would break the constraint of the different languages. Secondly, Django aims to follow Python's batteries included philosophy. Django can provide functionalities including Free API; URL routing, default admin section, HTTP libraries, etc. Thirdly, Django has the Model-View-Template (MVT) design pattern. Model-View-Template determines the structure and framework of the website using Django in managing the data into a database which means the model is a database table. Since the main function of our web app is to create an easier questionnaire for researchers to complete, we need to build a convenient interaction between the database and the template data. The view in the Django can receive the HTTP requests and send the

HTTP responses. There is an interaction between a model and a template which could complete a response. All the templates are using the front-end layer and the dynamic component of the HTML from a Django application. Model manages the data through a database just like a database table. View would receive the requests from HTTP and send HTTP responses. The template is the front-end layer basically with the interaction with dynamic HTML components.

3.10.1 Database tools :

We firstly use SQLite as the database. Because it is lightweight and fully ACID-compatible, allowing safe access from multiple processes or threads. Whatsmore, SQLite is written in ANSI-C, providing a simple API, and it runs in UNIX (Linux, Mac OS-X, Android, IOS) and Windows, which prevents the platform inconsistency. Since we do not need the server at first, SQLite supports serverless (Django, 2021). SQLite is included in Python and the built-in database of Django, which we can process our work without configuring in our system. We acknowledge that we need to change the database table frequently through the project, it is important that our database is easy to update or change the database table content. SQLite can be connected statically or dynamically as required by the model in the Django framework. Also, to save the users' datas and reflect the users' questionnaires' answers on the front-end, SQLite can access its datas directly from view in Django. However, at week 12, the client is required to upload it to AWS server. Because SQLite is not suitable for the server, we transferred from SQLite to PostgreSQL. Since we have finalized the database table on SQLite, this eliminates the limitation of PostgreSQL. And compared to SQLite, PostgreSQL has more functionality and is more secure.

3.10.2 Source code management:

To manage the source code, we use bitbucket. Because all team members are working on the same codebase, to prevent the conflict. Each teammate has their own branch. Before we start on coding, we always pull from our master branch to update our own branch codes and we program on our own branch. After completing the work, we will always create the pull request to let other group members review and refactor the code before we merge codes to our master branch. This can prevent our master code from bugs and conflicts and allow us to work concurrently.

Pull requests

[Create pull request](#)

| Summary | Activity | Reviewers | Builds |
|---|---|---|---|
|  complete function button <code>answers</code> → <code>master</code> zhixin gao - #108, created 3 days ago, updated 3 days ago | |  |  |
|  update and fix bugs of navbar <code>admin_test</code> → <code>master</code> Alvin Peng - #109, created 3 days ago, updated 3 days ago | |  |  |
|  Answers <code>answers</code> → <code>master</code> zhixin gao - #105, created 4 days ago, updated 3 days ago | |   |  |
|  admin <code>admin_test</code> → <code>master</code> Alvin Peng - #106, created 4 days ago, updated 3 days ago |  1 |  |  |
|  Forms styling <code>forms-styling</code> → <code>master</code> Shyam Raj - #107, created 3 days ago, updated 3 days ago | |  |  |
|  update template guide <code>answers</code> → <code>master</code> zhixin gao - #104, created 5 days ago, updated 5 days ago | |  |  |
|  Answers <code>answers</code> → <code>master</code> zhixin gao - #103, created 7 days ago, updated 6 days ago | |  |  |
|  added admin virtual user tests <code>admintests</code> → <code>master</code> Shyam Raj - #101, created 2021-11-03, updated 2021-11-03 | |  |  |
|  virtual test for PIS/ PCF <code>navbarcoversheet</code> → <code>master</code> Shuai Sun - #102, created 2021-11-03, updated 2021-11-03 | |  |  |
|  Navbarcoversheet <code>navbarcoversheet</code> → <code>master</code> Shuai Sun - #100, created 2021-11-03, updated 2021-11-03 |  2 |  |  |
|  added styling to new_PIS and new_PCF files <code>forms-styling</code> → <code>master</code> Shyam Raj - #99, created 2021-11-01, updated 2021-11-02 | |  |  |
|  virtual test part 2 <code>navbarcoversheet</code> → <code>master</code> Shuai Sun - #98, created 2021-11-01, updated 2021-11-01 | |  |  |
|  Changing to function views <code>changing_to_function_views</code> → <code>master</code> Alex Del Favero - #97, created 2021-10-31, updated 2021-10-31 | | No reviewers |  |
|  Justify the second navigation bar and side navigation bar <code>navbarcoversheet</code> → <code>master</code> Shuai Sun - #96, created 2021-10-29, updated 2021-10-29 | |  |  |
|  Answers <code>answers</code> → <code>master</code> zhixin gao - #95, created 2021-10-28, updated 2021-10-28 | |  |  |
|  finished template answers script <code>templatequs</code> → <code>master</code> Shyam Raj - #93, created 2021-10-28, updated 2021-10-28 | |  |  |
|  implement the side bar <code>navbarcoversheet</code> → <code>master</code> Shuai Sun - #94, created 2021-10-28, updated 2021-10-28 | |  |  |
|  virtualtest <code>navbarcoversheet</code> → <code>master</code> Shuai Sun - #92, created 2021-10-28, updated 2021-10-28 | |  |  |
|  Templatequs <code>templatequs</code> → <code>master</code> Shyam Raj - #91, created 2021-10-26, updated 2021-10-26 | |  |  |
|  Answers <code>answers</code> → <code>master</code> zhixin gao - #90, created 2021-10-26, updated 2021-10-26 | |  |  |
|  Rich text implement <code>rich_text_implement</code> → <code>master</code> Alex Del Favero - #89, created 2021-10-25, updated 2021-10-25 | | No reviewers |  |
|  add the content of the tips to all the questions of qualifier quiz and ethics questionnaire <code>navbarcoversheet</code> → <code>master</code> Shuai Sun - #88, created 2021-10-22, updated 2021-10-22 | |  |  |
|  styling the tips of qualifier quiz page and ethics questionnaire page <code>navbarcoversheet</code> → <code>master</code> Shuai Sun - #87, created 2021-10-21, updated 2021-10-21 | |  |  |
|  modify the qualifier quiz questions <code>navbarcoversheet</code> → <code>master</code> Shuai Sun - #86, created 2021-10-21, updated 2021-10-21 | |  |  |
|  Testing <code>testing</code> → <code>master</code> Alvin Peng - #83, created 2021-10-14, updated 2021-10-21 | |   |  |

To protect our source code, Django can hide the website's source code. The framework can prevent the website from XSS and CSRF attacks, SQL injections, etc. It can notify of a large amount of security mistakes.

3.11 Information search/research and discipline knowledge use and application:

3.11.1 INFO1110

Knowledge of python basics was obtained from this unit. These basics include variables and expressions, flow control, loops and lists, functions, modules, methods and classes. This knowledge was then applied when building the back-end of the website.

3.11.2 ISYS2120

Knowledge of SQL and databases was obtained from this unit and applied in the project. The team applied this knowledge to build a database schema for the website. This schema was then transformed into django models and input into the database in the form of users, ethics applications, cover sheets, participant information sheets and participant consent forms.

3.11.3 SOFT2412

Knowledge of agile processes was obtained in this unit. This knowledge includes the use of: git, continuous integration pipelines, unit testing, usability testing and user stories.

3.11.4 INFO2222

Knowledge of web development was obtained in this unit. Use of front-end tools such as HTML and CSS were used to build the web pages. Knowledge of the python flask web-framework allowed for a seamless transition into learning the python django web-framework which we used to build the back-end of the website.

3.11.5 Django website

Knowledge of developing the Django framework was obtained from the official Django website. All team members learned through the tutorial supported by the official Django website. Knowledge of setting up the environment and building the SQLite database are also obtained from the official Django website. We also learned to maintain the back-end from the website.

3.11.6 W3SCHOOLS, Youtube

Knowledge of the front-end which are HTML, CSS, Javascript and JQuery were obtained from the W3 school website and youtube online tutorial video. We built all the webpage frames through HTML and styling with CSS, linking to the templates on the Django framework to build the connection with the back-end. We learned more deeply about HTML and CSS to fulfill the requirements of the project through these two sources. Since our webpage should be dynamic, we learned Javascript and JQuery to program the behavior of web pages from the two sources. For example, the show and hide functionality on questionnaire, the three dot drop down menu functionality, search functionality, sort functionality and so on.

4 Quality of Group Processes

4.1 Tooling for development, management and allocation of tasks

In the developing period, the team uses bitbucket with Jira Software to record all the tasks, progress, process, and result.

Team clone the code from the remote repository to their own local end. Team members use 'git pull origin master' before they modify the code every time to collect the newest code done by other team members and then they keep doing their own code modifying. Team members would create a new branch to modify the code which means they can not modify the code in the master branch directly, that is used for maintaining the correctness of the remote repository.

After they achieve their tasks in their own branch named 'ownbranch' for example, they would 'git add .'; 'git commit -m new_content'; 'git push origin 'ownbranch'.

After those operations, the modified code has already been push to the remote branch also called 'ownbranch'.

After this, team members would go to the bitbucket to create a new pull request which contains all the changes made by this team member.

Other team members would have a look and check the new modified code. They can approve and merge the new modified code to the origin master repository.

As for the management and allocation of tasks, the team used Jira Issues to allocate the tasks and track the progress of the tasks by labeling the tasks to be "to do", "in progress" and "Done".

Manager allocated the tasks and assigned the team members to do the tasks in weekly meetings. All the team members would follow the Jira issue assignee to finish the tasks allocated by manager one by one. And all the team members agree with the allocation of tasks from the manager. If there are any questions, team members would ask and discuss using slack. As for the allocated programming tasks amount, that would be 1-3 tasks for each team member with fair allocation.

Here is the link of Jira:

[Jira](#)

Here is the link of bitbucket:

[Bitbucket](#)

Here is the link of slack:


[Slack](#)

Documentation of the project link for the Project:


[Documentation](#)

4.2 Evidence of collaboration and teamwork


Team members using slack to communicate with each others about the project progress:


Ben Raj 12:48 PM


so for template ans. One text box?


Shuai Sun 12:49 PM


it needs an instruction for users to know how to use that template ans
as for one text box or not, not sure


Alex Del Favero 12:50 PM


One template box
Just make the text colored that comes from the template answer


Zinc Gao 12:54 PM

And when the user edit the colored template answer, it will still be colored?


Alex Del Favero 12:55 PM

Only the template text be computed


Ben Raj 12:55 PM

also if we have to reference the application id does that mean we have to add all approved application answers to it?

This table is the XP roles of the team from Week 6 to Week 12:
The XP roles from Week 2 to Week 5 is in appendix A1

| | Alex | Ben | Alvin | Shuai | Zinc |
|---------|----------|---------|--------|------------|---------|
| Week 6 | Customer | Manager | Tester | Programmer | Tracker |
| Week 7 | Customer | Manager | Tester | Programmer | Tracker |
| Mid-Sem | Customer | Manager | Tester | Programmer | Tracker |
| Week 8 | Customer | Manager | Tester | Programmer | Tracker |
| Week 9 | Customer | Manager | Tester | Doomsayer | Tracker |
| Week 10 | Customer | Manager | Tester | Doomsayer | Tracker |
| Week 11 | Customer | Manager | Tester | Doomsayer | Tracker |
| Week 12 | Customer | Manager | Tester | Doomsayer | Tracker |

These are links of the Evidence of collaboration and teamwork:

[Week 6 Tutorial Meeting Minutes](#)
[Week 6 Group Meeting Minutes](#)
[Week 6 Client Meeting Minutes](#)
[Week 6 Weekly Plan](#)
[Week 6 Status Report](#)
[Week 7 Tutorial Meeting Minutes](#)
[Week 7 Weekly Plan](#)
[Week 7 Status Report](#)
[Mid-sem Group Meeting Minutes](#)
[Mid-sem Weekly Plan](#)
[Mid-sem Risk Management](#)
[Week 8 Tutorial Meeting Minutes](#)
[Week 8 Group Meeting Minutes](#)
[Week 8 Weekly Plan](#)
[Week 8 Status Report](#)
[Week 8 Tracker Logbook](#)

[Week 9 Tutorial Meeting Minutes](#)
[Week 9 Group Meeting Minutes](#)
[Week 9 Client Meeting Minutes](#)
[Week 9 Weekly Plan](#)
[Week 9 Status Report](#)
[Week 9 Risk Management](#)
[Week 9 Tracker Logbook](#)
[Week 10 Tutorial Meeting Minutes](#)
[Week 10 Group Meeting Minutes](#)
[Week 10 Client Meeting Minutes](#)
[Week 10 Weekly Plan](#)
[Week 10 Status Report](#)
[Week 10 Risk Management](#)
[Week 10 Tracker Logbook](#)
[Week 11 Tutorial Meeting Minutes](#)
[Week 11 Group Meeting Minutes](#)
[Week 11 Weekly Plan](#)
[Week 11 Status Report](#)
[Week 11 Risk Management](#)
[Week 11 Tracker Logbook](#)
[Week 12 Tutorial Meeting Minutes](#)
[Week 12 Group Meeting Minutes](#)
[Week 12 Weekly Plan](#)
[Week 12 Status Report](#)
[Week 12 Risk Management](#)
[Week 12 Tracker Logbook](#)
[Week 13 Tutorial Meeting Minutes](#)
[Week 13 Client Meeting Minutes](#)
[Week 13 Tracker Logbook](#)
[Week 13 Status Report](#)

4.3 Allocation of group roles, potential risks, constraints

All the group roles are team members chosen by themselves, and they would always be in the same roles in these weeks. that would increase the potential risk because the team members would forget what other roles should do, they may forget the responsibility of different roles and other team members cannot track timely.

Constraints would appear if the team always did not change the roles.

What's more, Gary was our team member before week 8, but he dropped this course. Therefore, our team members' numbers changed from 6 to 5 now. The potential risks would be for per team member, they need to do more than planned before which belong to Gary Tasks. Team members need to spend more time on the extra tasks.

4.4 Use of bitbucket and slack and other tools

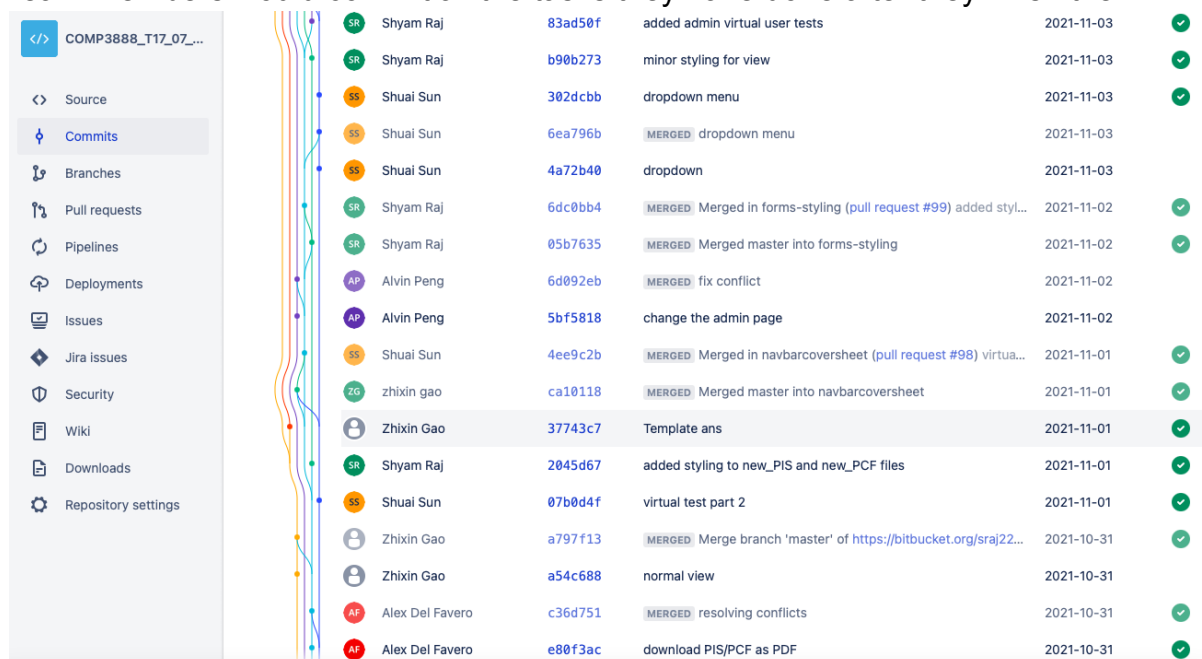
Team members used Zoom to have the meeting to discuss or allocate the tasks from the client requirements.

Team used Bitbucket to unify the developing code. Team members clone the code from the remote repository from the master, and they used 'git pull origin master' to collect the newest code from the remote repository. They would create a branch to modify their new code called 'ownbranch' for example.

After they finish modifying the new code, they need to 'git add .'; 'git commit -m newcontent'; 'git push origin ownbranch'. The code in the remote repository would collect these new branches.

After finishing these operations, team members would create a new pull request using bitbucket. Other team members would approve and merge that pull request and the new code would be uploaded to that remote repository.

Team members would commit all the tasks they have done after they finish them.



The screenshot displays the Bitbucket web interface for a repository named 'COMP3888_T17_07_...'. On the left is a sidebar with navigation links: Source, Commits (selected), Branches, Pull requests, Pipelines, Deployments, Issues, Jira issues, Security, Wiki, Downloads, and Repository settings. The main area features a commit history table and a branch visualization diagram. The table lists commits by author, hash, message, date, and status. The visualization shows the branching model with lines representing branches and dots representing commits.

| Author | Commit Hash | Message | Date | Status |
|-----------------|-------------|---|------------|--------|
| Shyam Raj | 83ad50f | added admin virtual user tests | 2021-11-03 | ✓ |
| Shyam Raj | b90b273 | minor styling for view | 2021-11-03 | ✓ |
| Shuai Sun | 302dcbb | dropdown menu | 2021-11-03 | ✓ |
| Shuai Sun | 6ea796b | MERGED dropdown menu | 2021-11-03 | ✓ |
| Shuai Sun | 4a72b40 | dropdown | 2021-11-03 | ✓ |
| Shyam Raj | 6dc0bb4 | MERGED Merged in forms-styling (pull request #99) added styl... | 2021-11-02 | ✓ |
| Shyam Raj | 05b7635 | MERGED Merged master into forms-styling | 2021-11-02 | ✓ |
| Alvin Peng | 6d092eb | MERGED fix conflict | 2021-11-02 | ✓ |
| Alvin Peng | 5bf5818 | change the admin page | 2021-11-02 | ✓ |
| Shuai Sun | 4ee9c2b | MERGED Merged in navbarcoversheet (pull request #98) virtua... | 2021-11-01 | ✓ |
| zhixin gao | ca10118 | MERGED Merged master into navbarcoversheet | 2021-11-01 | ✓ |
| Zhixin Gao | 37743c7 | Template ans | 2021-11-01 | ✓ |
| Shyam Raj | 2045d67 | added styling to new_PIS and new_PCF files | 2021-11-01 | ✓ |
| Shuai Sun | 07b0d4f | virtual test part 2 | 2021-11-01 | ✓ |
| Zhixin Gao | a797f13 | MERGED Merge branch 'master' of https://bitbucket.org/sraj22... | 2021-10-31 | ✓ |
| Zhixin Gao | a54c688 | normal view | 2021-10-31 | ✓ |
| Alex Del Favero | c36d751 | MERGED resolving conflicts | 2021-10-31 | ✓ |
| Alex Del Favero | e80f3ac | download PIS/PCF as PDF | 2021-10-31 | ✓ |

Team members need to create their own branches to modify the code and not modify code from master directly.

| COMP3888_T17_07_... | | Branch | Behind | Ahead | Updated | Pull request | Builds | Actions |
|---------------------|--|--------------------------------|--------|-------|------------|--------------|--------|---------|
| | | master MAIN DEVELOPMENT | | | 4 days ago | | ✓ | ... |
| | | tick | 49 | 1 | 7 days ago | Create | ✓ | ... |
| | | newcoverpage | 321 | 1 | 2021-10-02 | Create | ✓ | ... |
| | | AnswerToDB | 352 | 2 | 2021-10-01 | Create | ✓ | ... |
| | | sort_delete | 363 | 4 | 2021-09-30 | Create | ✓ | ... |
| | | allSectiono | 440 | 2 | 2021-09-30 | Create | ✓ | ... |
| | | managelist_to_qualifier | 368 | 1 | 2021-09-29 | Create | ✓ | ... |
| | | adminTest | 468 | 1 | 2021-09-24 | Create | ✓ | ... |
| | | questionnaire | 473 | 2 | 2021-09-24 | Create | ✓ | ... |
| | | Questionnaire | 517 | 1 | 2021-09-22 | Create | ✓ | ... |
| | | newbranch | 482 | 1 | 2021-09-21 | Create | ✓ | ... |
| | | adminpage | 518 | 33 | 2021-09-16 | Create | ✗ | ... |
| | | managelist | 518 | 28 | 2021-09-16 | Create | ✗ | ... |
| | | edit_profile2 | 596 | 1 | 2021-09-11 | Create | ✓ | ... |

Bitbucket with Jira Issue is the main tool for team members to record all the tasks, progress, process, and result. That could record the process of allocation of each task, and it would be easy for team members to track which task they need to do and which task they have already finished. When the tasks are created, the Jira Backlog would show all the tasks with the label “To Do” with assignee a team member, and that team member would change label to “In Progress” when he starts doing the tasks. After he finishes the tasks, team members would change the label to “Done”. That means, for that task, it has been finished.

| | | | | | |
|---|--|-------------|----|------------|-----|
| <ul style="list-style-type: none"> Pull requests Pipelines Deployments Issues Jira Issues Security Wiki Downloads Repository settings | ✓ CT0G-374 dockerise web app image | DONE | AP | yesterday | ... |
| | ✓ CT0G-373 Create AWS account and create server | DONE | AP | yesterday | ... |
| | ✓ CT0G-387 Complete assigned sections for final group report | TO DO | SR | 3 days ago | ... |
| | ✓ CT0G-386 Complete assigned slides for final project presentation | DONE | SR | 3 days ago | ... |
| | ✓ CT0G-385 Final Report Report for Technical Documentation | IN PROGRESS | SS | 3 days ago | ... |
| | ✓ CT0G-384 Final report for Group Process Part | IN PROGRESS | SS | 3 days ago | ... |
| | ✓ CT0G-381 Complete wk12 group minutes | DONE | SR | 4 days ago | ... |
| | ✓ CT0G-365 Styling for PCF and PIS function | DONE | SR | 4 days ago | ... |
| | ✓ CT0G-362 Do Documentation (admin role) | DONE | SS | 4 days ago | ... |
| | ✓ CT0G-370 3 dot functionality for operations | DONE | AP | 2021-11-03 | ... |
| | ✓ CT0G-372 unit testing | IN PROGRESS | AP | 2021-11-03 | ... |

The Pipelines is a tool to check if the code uploaded can be built successfully.

| | | | | | |
|------|----|--|--------------|------------|--------|
| #419 | ZG | Add extra field to view for normal and staff view zhixin gao 453d5b6 answers | ✓ Successful | 8 days ago | 23 sec |
| #418 | ZG | Add extra field to view for normal and staff view zhixin gao bfaae8e answers | ❌ Failed | 8 days ago | 27 sec |
| #417 | ZG | Add extra field to view for normal and staff view zhixin gao d95289d answers | ✓ Successful | 8 days ago | 22 sec |
| #416 | ZG | Merged in admintests (pull request #101) added admin virtual ... zhixin gao bfcda06 master | ⏸ Paused | 8 days ago | 0 sec |
| #415 | ZG | Merged master into admintests zhixin gao 3c6681e admintests | ⏸ Paused | 8 days ago | 0 sec |
| #414 | ZG | Merged in navbarcoversheet (pull request #102) virtual test fo... zhixin gao b6381d4 master | ✓ Successful | 8 days ago | 24 sec |

There is also a wiki page in the bitbucket. That is a page to contain all the models such as weekly plan; group meeting minutes, client meeting minutes, in-tutorial meeting minutes, status report, tracker logbook and risk management.

Meeting minutes

- [Tutorial Meeting Minutes](#)
- [Group Meeting Minutes](#)
- [Client Meeting Minutes](#)
- [Status report](#)

Other Documentations

- [Weekly Plan](#)
- [Risk management](#)
- [Tracker logbook](#)
- [Documentation](#)

As for the tools of communicating used by team members.

Team has two slacks. The first one is used for team communication and discussing the project. The second one is used by the team to contact the client to know the requirements and the information of the project. Team members also used WeChat to discuss the tasks requirements.

[Slack for team members](#)

[Slack for team and client](#)

[Bitbucket](#)

[Jira Issue](#)

4.5 Work with clients

Team using slack to communicate with the client, the client would answer the questions using the slack.

[Slack of team with client](#)

Monday, September 6th ▾

Judy Kay 7:08 AM
@channel I see there is also email on this - but will answer it here.

There are two answers.

1. Whatever the questions are, you should build the site so that there is a set of questions and associated answers that map to an application being low risk.
2. You should be able to find the intranet information that tells researchers what the precise rules are but for your system, you should be sure it is easy to add and alter questions.

I said this yesterday, there should be a text file with a set of questions and the answer options, with a marker for the one that is needed for an application to be approved - I would like to see your spec for that.

You should use a similar approach to the other questions - though we are now getting into the implementation/

Team and client would have one meeting using Zoom every week. In the client meeting, the team would show the content and progress to the client and the client would give feedback for the team to improve.

The customer of the team would also send email to the client about the questions in the project.

Fw: Short answer questions for template answers



✉ Alexander David Del Fav...

Tuesday, 26 October 2021 at 14:15

To: ✉ Chengwei Peng; ✉ Shyam Raj; ✉ Zhixin Gao; ✉ Shuai Sun

Hi Judy,
here are the short answer questions included in the questionnaire.
Please let me know which ones you would like there to be a template for answering.

I have numbered these questions sequentially, and not by their original question numbers.

Q1. Describe how you will identify and select potential participants for recruitment into the study. You should include information about how you will obtain contact details for potential participants.

Q2. Describe how and where initial contact will be made with potential participants and how you will avoid real or perceived coercion. Copies of all relevant correspondence (e.g. email, letter of introduction, covering letter, circular/flyer etc.) need to be uploaded with your application. If you are using email addresses please outline how their use will not be in breach of privacy or spam legislation.

Q3. If a participant, or person on behalf of a participant, chooses to withdraw from the research, what specific consequences should they be made aware of, prior to giving consent? These details should be included in the Participant Information Statement.

Fw: Client meeting replacement questions



✉ Alexander David Del Fa...

Thursday, 14 October 2021 at 08:27

To: ✉ Shyam Raj; ✉ Shuai Sun; ✉ Zhixin Gao; ✉ Chengwei Peng

Hi Judy,

In place of our client meeting today, I have a few questions from the questionnaire which we would like to know if they should be included in the qualifier questionnaire.

Here are the questions:

(Already in qualifier questionnaire)

Note: are these questions high-risk? They sound low-risk.

Section B

41. Does your research involve Aboriginal and/or Torres Strait Islander peoples?

47. Does your research involve CALD (Culturally and Linguistically Diverse) people?

(Not in qualifier questionnaire)

Note: These questions sound high-risk, should they be included?

On Q 41 and 47, I think it could be low risk unless they are explicitly recruited.

Certainly the software should support that. It should also be easy to change so there should be information for the admin to indicate how they

Scope statement link:

[Scope statement](#)

4.6 Critique

Team members could communicate well. Team members could finish all the tasks that were allocated on time. For XP roles, team members could do the responsibility timely. Team now not only uses slack to communicate with the client, they also use email. That is an improvement. Team members create the branches to modify their own code and they make a clear commitment with their modification. Other team members would easily understand that. Team members could help each other and do pair programming to face the difficult goal together. That is the team. All the team members could do their responsibility of XP roles. For example, the manager of the team could set up the meetings and update the bitbucket and Jira Issue weekly. The

customer of the team could communicate with the client to collect the thinking and feedback and demo the project progress to the client. The tester of the team did lots of the testing job for the project. The tracker of the team can track the progress of the project weekly and create a logbook to show in the tutorial. The doomsayer of the team can analyse the team and the project progress to create a risk management weekly. All the team members are contributing to their project together. That is the meaning of 'team'. There are links to Risk Management weekly:

[Week 9 Risk Management](#)

[Week 10 Risk Management](#)

[Week 11 Risk Management](#)

[Week 12 Risk Management](#)

Reflections of Extreme Programming:

The customer of the team communicates with the client to know the functional requirements and the feedback of the questions asked. That XP role is the bridge between the programming team and the client. Customer helps the team to know the requirements from the client.

The manager of the team set up the group meetings and managed the bitbucket and Jira Issue. Manager creates meeting minutes and makes a weekly plan every week. Manager helps the team to operate and develop normally.

The tester of the team did a lot of the testing work for the project development. Testers help the team to check the code which all the team members made is correct or not. If the code has an error or bug, the test would tell the team member to fix that.

The tracker of the team tracks the progress of all the team members and creates a status report. Tracker helps the team to record the progress and that is good for the next weekly plan.

The doomsayer of the team analyses the risk of the team and project in the developing period weekly. Doomsayer creates risk management and that would show the problem which the team is faced with and that would facilitate the team to fix the problem and keep increasing the efficiency of developing.

Programming Practice:

Teams use bitbucket to show each commit made by different team members. The history of the commit and pull request with merge history would show in the bucket.

[Link of commit history](#)

[Link of pull request with merge history](#)

Appendix

A1

| | Alex | Ben | Alvin | Shuai | Zine | Gary |
|--------|----------|------------|------------|-----------|-----------|------------|
| Week 2 | Tracker | Manager | Customer | Tester | Doomsayer | Programmer |
| Week 3 | Manager | Doomsayer | Programmer | Customer | Tracker | Tester |
| Week 4 | Customer | Programmer | Doomsayer | Tracker | Manager | Tester |
| Week 5 | Customer | Manager | Tester | Doomsayer | Tracker | Programmer |

A3

For the unit tests, we have already discussed details in the previous section. The unit tests exist on the test.py file in each application folder. These application folders are in the Project folder. In each application folder, you can see the test.py file.

https://bitbucket.org/sraj2279/comp3888_t17_07_group1/src/master/Project/

To run the test case. Your working directory should be ./Project, which has the manage.py file. Before you run the test cases, make sure you download the django. Then you can run test cases by using the command “python manage.py test --appname” or “python manage.py test” to run all the cases. For example you can try “python manage.py test accounts” to run the test cases under accounts application.

A4

The details of acceptance tests are shown in the previous section. Most of the acceptance tests are done by virtual user testing. You can get access to them by looking at the “virtualusers” folder. To run these files, you need to run “Vuser.py” file on terminal, and “Vuser.py” is in “*Test” folder.

https://bitbucket.org/sraj2279/comp3888_t17_07_group1/src/master/virtualusers/

References

Javapoint, <https://www.javatpoint.com/django-mvt>, visited November 2021

Django website, <https://docs.djangoproject.com/en/3.2/intro/tutorial01/>, visited November 2021

Django website, <https://docs.djangoproject.com/en/3.2/intro/tutorial02/>, visited November 2021.

W3SCHOOLS Javascript tutorial, <https://www.w3schools.com/js/DEFAULT.asp> , visited November 2021.