|  |  |  |
| --- | --- | --- |
| 5COSC021W Coursework 2 - INDIVIDUAL template 2024\_25  * Use this template to structure the individual part of coursework2. Ensure that the correct information is in each white box. The advice for each box is basic guidance to help you focus your answer. * YOU MUST USE THIS TEMPLATE FOR THE INDIVIDUAL PART OF COURSEWORK 2.  The current size of the boxes is not indicating how much you should write; change their size as you need.When you save the file, put your name and registration number in the file name, eg ‘5COSC003W\_cwk2\_Individual\_Kelly\_Garret\_12345678.doc’.Sections in the template that don’t have any text will receive no marks. The code files are used to ensure that what is written in the template is supported with what was implemented. However, code files only will not receive any marks and will not be used as submission of part of the template. Similarly, templates submission without the submission of code files will receive no marks.  * In order for the tutors to be able to assess your work you must: * Submit a zipped project folder of the **COMPLETE** working project (i.e. the parts of each group member incorporated in one program, not just your part). If you have not been able to incorporate your part with that of the group, then submit only your part – it should be able to run though by itself. The folder should include all the necessary files (including databases) to run as a project on Django and SQLite.   -          Make sure that the submitted project will run using the software provided by the University. Contact your tutor if you have any problems with this.  -          Make sure that the project folder should contain all files necessary to run the program e.g. databases etc.  -          Make sure that file I/O code does not use absolute file paths.  -          Make sure that the submission contains all usernames and passwords necessary to test the program.   * Include a link to a video describing/running the complete application – each team can produce one video, but all submissions must include the link that video. If your work is not integrated with that of the group, you can upload your own video of your work. | | |
| **Surname** | Mehmet | |
| **Forename** | Gilgil | |
| **Registration No:** | W2046446 | |
| **By submitting this coursework you agree to the following:** | | |
| I confirm that I understand what plagiarism is and have read and understood the section on Assessment Offences in the Essential Information for Students. The work that I have submitted is entirely my own. Any work from other authors is duly referenced and acknowledged. | | I confirm |
| **List here the team name and the other members of your group** | 5SE02\_03\_H  Adnan Choudhury  Thanaphol Promsilp  Aleeza Muneer  Muhammad Hashmi | |

|  |
| --- |
| Code – Code functionality – individual element as agreed and integration to application - maintainability applied (35 marks) |
| **Guidance:**   * Include here the **names of the files** that make up the individual element of your application. Where necessary you may have to include files that were co-authored with other members of your team (eg for the database etc) * Explain what each file does and the functionality of the code. Detail whether all the code in the file is your own product or a product with other team members. Include the names of the other authors where applicable. Authorship information should also be commented at the beginning of each of your files. * Comment on the maintainability of your code. Reflect on your coding standards and conventions used, discuss code reusability and use of comments. All text in the template must be specific to your application and supported with examples from your implementation. * Explain how your code was integrated with the code of other members of the group so that to produce one application * Marking of this section will also include assessment on the submitted files, the overall functionality of your work, as well as defence of your work during the demonstration in order to ensure that what is written in this template is supported by the implementation that was submitted. * Use more than one page! |
| **Information on how to run project:**  **SuperUser(admin page) log in credentials:**  **Username: admin**  **Password: hello123**  To run project, enter in the terminal in vscode:  cd djangodb (if you are not already in the correct directory)  python manage.py runserver  Example login to use for employee summary (As vote wasn’t implemented correctly, we had to use dummy data to view summary):  Engineer:  **All passwords are Group123** **for the following users:**  Engineer John Doe: Employee ID – 123  Team Leader Adnan Choudhury: Employee ID – 12345  Department Leader James Doe: Employee ID – 1234  Senior Manager One Zero: Employee ID - 101  **File names:**  **HTML:**  home.html, mainpage.html, password\_reset\_done.html, password\_reset\_form.html, password\_reset\_sent.html, password\_reset.html, signin.html and signup.html.  **CSS:**  signAreaStyle.css and bootstrap.min.css (premade bootstrap file for CSS).  **JS:**  bootstrap.bundle.js (I did not use this file but was instructed to include it when downloading bootstrap in the head tag of my html.)  **Django:**  Files I created myself: forms.py, urls.py (in app)  Files provided by Django that I worked on: settings.py, urls.py (in the project folder not app folder), apps.py, models.py and views.py.  **SQLite database:**  Designed/ created in models.py by me, I entered instances for Card, Department, Team etc.  In shared files, such as views.py my teammates have their own code. I will only discuss code that I have solo authored.  **Explaining what each file does:**  **home.html:**  Landing page of the site. Has a brief about section for the site and two links. One of them send the user to a log in page and the other sends the user to a sign-up page.  **signin.html:**  From the landing page as well as the sign-up page and reset password page you can enter the sign in page. It asks for two details the users employee ID and their password. You can then submit the form which will be authenticated (will explain further when explaining Django code). If the details are correct, it will take you to the main page, if the details are incorrect, it will display a message telling the user.  **signup.html:**  From the landing page as well as the login page you can enter the sign-up page. This page allows you to submit a form to create an account with all the relevant details such as employee ID to the employee role. It has verification and displays relevant error messages where needed. It is done by extending built in Django packages and classes and customising them to fit this project (will explain further when explaining Django code).  **password\_reset\_done.html, password\_reset\_form.html, password\_reset\_sent.html and password\_reset.html:**  These 4 pages uses Djangos built in password reset. The password\_reset.html asks the user to input an email. If the email is valid and exists in the database, an email will be sent from the specified email account to the entered email account with instructions on how to set the password and the user will also be redirected to password\_reset\_sent.html page letting them know the email was sent. The email will contain a link to the password\_reset\_form.html page where you can reset your password. Once you have successfully reset your password, it will redirect you to the password\_reset\_done.html page where it lets the user know the password has been reset and the user can click the link below to go to the sign in page.  **mainpage.html:**  After signing in with the correct credentials, the user is sent to the main page. This page contains a navigation bar that has links to the other areas of the site such as summary, health check or profile pages. It also has a logout button which allows the user to sign out. The main content of the page is just explaining what the links in the navigation bar do, as well as a brief description of how the health check works. This page can only be accessed if the user is logged in with a valid login. Finally, the navigation using bootstrap can resize and change with the change in view port showing responsive design.  **signAreaStyle.css:**  Has custom CSS for the sign in page that I couldn’t implement with bootstrap. It ensures that the user a clean UI experience when the viewport shrinks, it moves the content around to ensure it all fits on screen in a clean way.  **settings.py:**  settings.py comes with a lot of code prebaked into it. I added a variable to indicate to Django where to find my static files so that I can use images and CSS for my html. In the INSTALLED\_APPS variable I indicated my app name. In TEMPLATES variable I indicated to Django where to look for my html files. Finally, to send the email for the reset password, I indicated the email provider I will use, whether I will use TLS or SSL and a Gmail account to send the email from.  **urls.py:**  I indicated to Django to send all root URL traffic to my apps urls.py file so that I can display my pages.  **apps.py:**  Indicated the name of my app I created in Django.  **urls.py:**  The urls.py file maps various URL patterns to their corresponding views. In this file, I defined the routes for user authentication (sign in, sign out, and sign up) and other important pages in the application. Each route is linked to a specific view function, which handles the logic for rendering the associated HTML template. The name attribute of each path allows me to refer to these URLs in templates or code, making it easier to manage and update URLs. Additionally, I utilized Django’s built-in authentication views for password reset functionality, ensuring that users can request password resets and complete the process securely.  **views.py:**  The views.py file contains functions that handle the logic for rendering the different pages in the application. In this file, I defined views for handling user authentication (login, signup, and logout) and for rendering several key pages (home, main page, health check, summary, and user profile). For login, I authenticate the user using their credentials and either log them in or show an error message. The signup view processes the custom SignUpForm to create new users, ensuring that the input is valid before saving. The logout view logs the user out and displays a success message. To secure access to the main, health check, summary, and user profile pages, I used the @login\_required decorator to ensure that only authenticated users can access them. Messages are used throughout to provide feedback to the user during different actions, such as successful logins, logouts, and form submissions.  **models.py:**  The models.py file defines several key models that represent different entities within the system. The Department model represents a department within the organization, with attributes for the department ID, name, and location. The Team model is linked to the Department model and represents different teams within each department. The UserProfile model extends the default User model to associate each user with a specific department, team, and role, providing a way to manage user-related data and permissions. The Session model stores information about work or meeting sessions attended by users, with a unique session ID and a timestamp. The Card model represents an entity that users interact with, possibly for voting or feedback purposes, with a name, description, and optional negative description. Finally, the Vote model tracks user votes associated with cards during sessions, storing information about the vote state, trend, comments, and links to the respective Card and Session. These models create the foundation for the application's data structure and relationships between different entities.  **admin.py:**  The admin.py file registers models such as Department, Team, UserProfile, Session, Card, and Vote to be managed through the Django admin panel. This allows administrators to easily create, read, update, and delete instances of these models directly from the admin interface.  **forms.py:**  The forms.py file defines the SignUpForm, which extends Django's built-in UserCreationForm to handle user registration with custom fields. In this form, I’ve added fields for email, first name, last name, and role. The role is presented as a dropdown menu with predefined choices. Custom CSS classes and placeholders are applied to the form fields for styling. The Meta class specifies that the form is based on the User model and includes fields like username, password, and role. Additionally, the form's \_\_init\_\_ method customizes the appearance of each field, such as adding helpful hints and setting placeholders. The save method is overridden to create a corresponding UserProfile instance, linking the role selected by the user with the created User.  **Maintainability, Coding standards and comments:**  I used meaningful and consistent names for variables, methods, and classes. For instance, in the views.py file, I named functions like signin, signup, and signout to clearly reflect their purpose (user authentication).  Proper indentation was followed to ensure that the code is clean and easily readable. This helps prevent errors and makes the code more understandable. The SignUpForm class is a reusable component for handling user registration.  By inheriting from Django’s UserCreationForm, I am leveraging existing functionality while extending it to meet the specific needs of my application (e.g., adding the role field). This reduces redundancy and makes the form customizable in the future without having to rewrite the code.  I’ve added meaningful comments throughout the code to make it easier for others (or my future self) to understand the purpose of each part of the code. For example, in views.py, I added comments to explain what each function does, such as “#Handle user login for the sign in view” and “#Handle user signup for the signup view”. This will help my teammates and teachers quickly understand the purpose of each view.  By following these practices, my code is structured in a way that makes it easier to maintain and extend. If a change needs to be made in the user registration process, for example, it can be done centrally in the SignUpForm class. Likewise, if a new feature related to user profiles is required, the UserProfile model can be extended without impacting other parts of the application.  **Group code integration:**  To integrate our code into a single, functional application, we used GitHub as our main collaboration platform. This allowed each team member to work on different parts of the project while keeping everything organized in one shared repository. Once we completed our individual tasks we combined our work.  Clear and consistent communication was key throughout the project. We held regular meetings to discuss progress, identify any overlaps or potential conflicts, and make sure everyone was aligned with the overall structure of the application. We also used messages and comments in GitHub and our group chat to quickly resolve smaller issues.  Our work from coursework 1 (CW1) played a big role in guiding our development. Since we had already laid out the basic structure and some of the core logic in CW1, we were able to build on that foundation. This made it easier to divide the work logically and ensure that all our contributions would fit together smoothly.  **Tutorial use:**  Alongside my own coding skills and the things, I learned during this module, I used 3 tutorials as guidelines for my project and customised them to fit the needs and scope of the project. They are linked below:  <https://www.youtube.com/watch?v=EqjRhO5CK6A>  <https://www.youtube.com/watch?v=CTrVDi3tt8o>  <https://www.youtube.com/watch?v=sFPcd6myZrY> |
| Your code – Code Quality – version control (5 marks) |
| **Guidance:**   * Comment on your version control and how you ensured that your work was compatible with that of your group members. Support your text with examples from your version control history. * Marking of this section will also include assessment on the submitted files, the overall functionality of your work, as well as defence of your work during the demonstration. |
| For this project, I used Git for version control to manage my code and track changes throughout the development process. I primarily worked individually, but I still made sure to commit regularly and use clear, descriptive commit messages so that it would be easy to understand what each update involved. This helped me stay organized and made it easier to track my progress over time.    Above is an example of one of the commits I made, where I wrote a clear summary of what I changed, where I made the change, and why I made the change. For example, in this case, I added comments to the areas I worked on and removed old, unnecessary JavaScript files that were no longer needed after moving form validation to Django on the backend. Writing detailed commit messages like this meant that if any problems came up later, I could easily look through the history to figure out when and why something was added or changed, without having to dig through the code manually.  Here you can see a snapshot of part of my commit history. Each commit represents a specific, logical change to the project, such as "Changed signup form," "Created and implemented reset password," and "Home page changes, created new pages." I made sure to break my work into smaller tasks and commit regularly, rather than saving everything for one large update. This approach made it easier to pinpoint when a specific feature was added or when a bug might have been introduced.  Even though I primarily worked alone on this project, I treated it like a group project by committing frequently and ensuring that each commit was meaningful and easy to understand. This way, if group members had been involved, they would have been able to pull the latest changes without confusion and without causing merge conflicts. I also ensured that my code was always in a stable state before pushing any changes, which would have made team integration smoother.  Overall, using version control not only made the development process more organized and manageable but also gave me a safety net if something went wrong. It allowed me to roll back to previous versions easily, experiment without risking the stability of the main project, and maintain a clean and functional final submission. It also helped me build good habits for working in larger teams in the future, where proper version control practices are even more important. |

|  |
| --- |
| Your code – Code Quality – output of test plans (10 marks) |
| **Guidance:**   * List your **test plan and the output** for your individual part **and** that of the overall application once you integrated with the elements of your group members. * Marking of this section will also include assessment on the submitted files, the overall functionality of your work, as well as deference of your work during the demonstration. |
| Sign UpTest Plan  |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Use Case Test ID** | **Use Case Test Name** | **User Requirement** | **Precondition** | **Postcondition** | | UC01 | Sign up | User should be able to self-register for the health check | User heads to the sign-up page | New account is created and user is sent to the login page |  Positive Case  |  |  | | --- | --- | | **User Case Test ID** UC01  Positive Test Case | | | Action | Inputs | Expected Output | Actual Output | Test result | Comments | | Enter valid signup details and submit | Employee ID: w2046446  First name: Mehmet  Last name: Gilgil  Email: [gilgilmehmet05@gmail.com](mailto:gilgilmehmet05@gmail.com)  Password: Groupwork123  Confirm: Groupwork123  Role: Engineer  Sign up button press | User is redirected to the login page; message appears letting the user know the account was created. User details appear in database. | User is redirected to the login page; message appears letting the user know the account was created. User details appear in database. | PASS | User is created, stores all the user details. User profile is created to store the user role. | | Click on login link at the bottom of the form | Click on login link at the bottom of the form | Redirects user to the login page | Redirects user to the login page | PASS | The user is sent to the login page. |  Negative Case  |  |  | | --- | --- | | **User Case Test ID** UC01  Negative Test Case | | | Action | Inputs | Expected Output | Actual Output | Test result | Comments | | Sign up with missing fields | Employee ID:  First name: Mehmet  Last name: Gilgil  Email: [gilgilmehmet05@gmail.com](mailto:gilgilmehmet05@gmail.com)  Password: Groupwork123  Confirm: Groupwork123  Role: Engineer  Sign up button press | Indicates to user which area of the form is missing and doesn’t create a user/ doesn’t allow form submission. | Indicates to user which area of the form is missing and doesn’t create a user/ doesn’t allow form submission | PASS | It shows the first missing input field. For example, if both first name and last name is empty it will first ask the user to fill in name and then ask the user to fill in last name. | | Trying to sign up with an invalid email. | Employee ID: w2046446  First name: Mehmet  Last name: Gilgil  Email: badmail  Password: Groupwork123  Confirm: Groupwork123  Role: Engineer  Sign up button press | Display a message letting the user know there was a problem when submitting the form. Doesn’t create a user/ doesn’t allow form submission. | Display a message letting the user know there was a problem when submitting the form. Shows the problem on the signup form above the input field. Doesn’t create a user/ doesn’t allow form submission. | PASS | Message is a popup at the top of the screen. The user is told where the/what the error is above the relevant input field. | | Trying to signup with password that doesn’t fit in specified criteria | Employee ID:  First name: Mehmet  Last name: Gilgil  Email: [gilgilmehmet05@gmail.com](mailto:gilgilmehmet05@gmail.com)  Password: weak  Confirm: weak  Role: Engineer  Sign up button press | Display a message letting the user know there was a problem when submitting the form. Doesn’t create a user/ doesn’t allow form submission. | Display a message letting the user know there was a problem when submitting the form. Shows the problem on the signup form above the input field. Doesn’t create a user/ doesn’t allow form submission. | PASS | Message is a popup at the top of the screen. The user is told where the/what the error is above the relevant input field. The password input fields are reset. | | Trying to signup with password that doesn’t match | Employee ID:  First name: Mehmet  Last name: Gilgil  Email: [gilgilmehmet05@gmail.com](mailto:gilgilmehmet05@gmail.com)  Password: Groupwork123  Confirm: Groupwork123567  Role: Engineer  Sign up button press | Display a message letting the user know there was a problem when submitting the form. Doesn’t create a user/ doesn’t allow form submission. | Display a message letting the user know there was a problem when submitting the form. Shows the problem on the signup form above the input field. Doesn’t create a user/ doesn’t allow form submission. | PASS | Message is a popup at the top of the screen. The user is told where the/what the error is above the relevant input field. The password input fields are reset. |  Sign InTest Plan  |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Use Case Test ID** | **Use Case Test Name** | **User Requirement** | **Precondition** | **Postcondition** | | UC02 | Sign in | User should be able login to the account they made for the health check. | User heads to the login page and already has a registered account. | If the user has entered valid details, take them to the main page of the site. |  Positive Case  |  |  | | --- | --- | | **User Case Test ID** UC02  Positive Test Case | | | Action | Inputs | Expected Output | Actual Output | Test result | Comments | | Enter valid sign in detail and submit | Employee ID: w2046446  Password: Groupwork123 | Sends user to main page | Sends user to main page | PASS | The form is submitted, program checks for details in the database. If they exist, the user is logged in and taken to the main page. | | Click on the sign-up link | Click on sign up link at the bottom of the form | Sends user to sign up page | Sends user to sign up page | PASS | The user is sent to the sign-up page. | | Click on the forgot password link | Click on forgot password link at the bottom of the form | Sends user to forgot password page | Sends user to forgot password page | PASS | The user is sent to the forgot password page. |  Negative Case  |  |  | | --- | --- | | **User Case Test ID** UC02  Negative Test Case | | | Action | Inputs | Expected Output | Actual Output | Test result | Comments | | User attempts to sign in with incorrect or non-existent details | Employee ID: 1111111  Password: incorrectpassword | Do not allow the user to login, keep them on the sign in page. | Do not allow the user to login, keep them on the sign in page. A message pops up telling the user they have entered incorrect details and to try again. | PASS | The form is submitted, program checks for details in the database. If they do not exist, the user is returned to the sign in page with an error message at the top. | | User attempts to sign in with incomplete details | Employee ID: w2046446  Password: | Do not allow the user to login, keep them on the sign in page. | Do not allow the user to login, keep them on the sign in page. A message pops up telling the user they have entered incorrect details and to try again. | PASS | The form is submitted, program checks for details in the database. As there cannot be a user without a password or Employee ID, the user is returned to the sign in page with an error message at the top. |  Forgot PasswordTest Plan  |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Use Case Test ID** | **Use Case Test Name** | **User Requirement** | **Precondition** | **Postcondition** | | UC03 | Forgot password | User should be able to reset their password if they have forgotten. | User heads to the forgot password page and has an account registered with a valid email. | User password is updated they can now log in with their new credentials. |  Positive Case  |  |  | | --- | --- | | **User Case Test ID** UC03  Positive Test Case | | | Action | Inputs | Expected Output | Actual Output | Test result | Comments | | User enters email to reset password | Email: [gilgilmehmet05@gmail.com](mailto:gilgilmehmet05@gmail.com) | Tells user email has been sent, and an email is sent to the users specified email. | Tells user email has been sent, and an email is sent to the users specified email. | PASS | Functionality is working as expected. User receives email promptly, no issues encountered. | | Click on login link at the bottom of the form | Click on login link at the bottom of the form | Redirects user to the login page | Redirects user to the login page | PASS | The user is sent to the login page. | | Email has link to reset password page | Link to reset password page in email | Email contains a valid link that directs the user to the password reset page. | Email contains a valid link that successfully redirects to the password reset page. | PASS | Link is functional and redirects correctly; no errors found. Email formatting is clear. | | Tells user to reset password | Password: Coursework123  Confirm: Coursework123 | User is redirected to a page that lets them know their password was successfully reset, with a link to the login page | User is redirected to a page that confirms password reset success, with a link to the login page. | PASS | Password reset completed without errors, detail are updated in the database and user can now use new details. Confirmation page and login link displayed correctly. |  Negative Case  |  |  | | --- | --- | | **User Case Test ID** UC03  Negative Test Case | | | Action | Inputs | Expected Output | Actual Output | Test result | Comments | | User enters invalid or unregistered email to reset password | Email: invalidemail@example.com | System should inform the user that the email is not registered and not send any email. | System says the email has been sent even though the email is not registered. | FAIL | System should not send reset emails to unknown addresses. This could confuse users and create security issues. | | User enters invalid email format | Email: user@com | System should block the user and ask for a valid email address. | System correctly blocks the invalid email and asks for a proper one. | PASS | Email format validation works well, no issues here. | | User leaves email field empty and submits | Email: (empty) | System should prevent submission and tell the user the field is required. | System shows a message asking the user to enter an email. | PASS | Good handling of empty fields, the user is clearly guided to fill it in. | | User submits mismatched passwords on reset page | Password: Coursework123 Confirm: Coursework321 | System should catch the mismatch and not allow password reset. | System correctly detects the mismatch and stops the process. | PASS | Password confirmation works as expected, stops errors before they happen. | | User submits weak password (e.g., "12345") | Password: 12345 Confirm: 12345 | System should reject the password and ask for a stronger one. | System rejects the weak password and prompts for a better one. | PASS | Password strength rules are working well and protecting user accounts. |  Main PageTest Plan  |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Use Case Test ID** | **Use Case Test Name** | **User Requirement** | **Precondition** | **Postcondition** | | UC04 | Main Page | After logging on user is taken to main page where there is a brief description of the page as well as links to other pages | User enters valid login details and logs into the system | User is now on the main page and can access other areas of the programme. |  Positive Case  |  |  | | --- | --- | | **User Case Test ID** UC04  Positive Test Case | | | Action | Inputs | Expected Output | Actual Output | Test result | Comments | | Log out of the account and return to sign in page | Press on the logout button | Returns user to the sign in page | Returns user to the sign in page | PASS | Gives the user a message letting them know they successfully signed out of their account, and they are returned to the sign in page. | | Click on a page in the navigation bar | Press on Health Check | Takes user to the page where the health check will be | Takes user to the page where the health check will be | PASS | Test was done before group work was combined. | | Click on a page in the navigation bar | Press on Summary | Takes user to the page where the Summary page will be | Takes user to the page where the Summary page will be | PASS | Test was done before group work was combined. | | Click on a page in the navigation bar | Press on User Profile | Takes user to the page where the User Profile page will be | Takes user to the page where the User Profile page will be | PASS | Test was done before group work was combined. |  Negative Case  |  |  | | --- | --- | | **User Case Test ID** UC04  Negative Test Case | | | Action | Inputs | Expected Output | Actual Output | Test result | Comments | | Attempts access main page without signing in | Enters this URL: http://127.0.0.1:8000/mainpage/ | Does not allow user to access the main page | Does not allow user to access the main page | PASS | Default Django error screen is used. |  Voting PageTest Plan  |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Use Case Test ID** | **Use Case Test Name** | **User Requirement** | **Precondition** | **Postcondition** | | UC05 | Voting page | User can cast a vote. | User is logged in and heads to voting page | The user makes a vote |  Positive Case  |  |  | | --- | --- | | **User Case Test ID** UC05  Positive Test Case | | | Action | Inputs | Expected Output | Actual Output | Test result | Comments | | User submits vote | Selects team and state to vote | User is taken to screen letting them know their vote is submitted | User is taken to screen letting them know their vote is submitted | PASS | User vote submits but is not connected to database |  Negative Case  |  |  | | --- | --- | | **User Case Test ID** UC05  Negative Test Case | | | Action | Inputs | Expected Output | Actual Output | Test result | Comments | | User submits vote without selecting a team | No team  Submit | Ask user to select a team | Ask user to select a team | PASS | User can’t submit form without filling it in. |  Summary PageTest Plan  |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Use Case Test ID** | **Use Case Test Name** | **User Requirement** | **Precondition** | **Postcondition** | | UC06 | Summary | User can view summary. | User is logged in and heads to summary page | The user is able to see relevant summary |  Positive Case  |  |  | | --- | --- | | **User Case Test ID** UC06  Positive Test Case | | | Action | Inputs | Expected Output | Actual Output | Test result | Comments | | User clicks on summary page | Engineer, Team Leader, Department Leader or senior manager clicks on summary page | User can see relevant summaries | User can see relevant summaries | PASS | Relevant summaries are displayed. | | User clicks on summary page | Engineer, Team Leader, clicks on summary page | User can see previous votes | User can see previous votes | PASS | Previous votes and comments are displayed. |  Negative Case  |  |  | | --- | --- | | **User Case Test ID** UC06  Negative Test Case | | | Action | Inputs | Expected Output | Actual Output | Test result | Comments | | User with no department or team clicks on summary page | User clicks on summary page | Tells user there’s no summary. | Tells user there’s no summary. | PASS | The summary areas and previous votes are all empty |  User Profile PageTest Plan  |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Use Case Test ID** | **Use Case Test Name** | **User Requirement** | **Precondition** | **Postcondition** | | UC07 | User Profile | User can change their relevant details. | User is logged in and heads to Profile page. | The user has changed their details and redirected to correct page. |  Positive Case  |  |  | | --- | --- | | **User Case Test ID** UC07  Positive Test Case | | | Action | Inputs | Expected Output | Actual Output | Test result | Comments | | Enter valid name or email to update name or email | First name: Hello  Email: hello@gmail.com | Updates name and email | Updates name and email, lets the user know details have been changed. | PASS | User details was updated in the database. | | Enter valid password | Password: Group123  Password: Group123 | Updates password. | Updates password, lets the user know the password has been changed, user is redirected to sign in page. | PASS | User details was updated in the database. |  Negative Case  |  |  | | --- | --- | | **User Case Test ID** UC07  Negative Test Case | | | Action | Inputs | Expected Output | Actual Output | Test result | Comments | | Entering password that doesn’t match | Password: Group123  Confirm: Project123 | Give error message and prevent form submission. | Give error message and prevent form submission. | PASS | Gives reason, in this case telling the user the passwords have to match. | | Entering invalid password | Password: hello123 | Give error message and prevent form submission. | Give error message and prevent form submission. | PASS | Gives reason, in this case password was too simple. | |

|  |
| --- |
| Professional conduct – Communication (10 marks)Seeking and using feedback |
| **Guidance:** List here, in a table, constructive feedback you have received from your team-mates, industry mentors and your tutor on your work and how you used it.  Feedback: **Be descriptive and specific** – but do not use quotations  Date Received: when was the feedback given  Source: Who gave you the feedback  How you used it: How did it inform your actions/changes |
| |  |  |  |  | | --- | --- | --- | --- | | Feedback | Date received | Given By | How you used it | | Recommended to build a separate landing page | 19/04 | Hasan | Built a separate landing page so that users can navigate easily between login and signup page. Made the user experience cleaner. | | Recommended to build a reset password page | 23/04 | Hasan | Built a page where user can use their existing account details to reset their password. | | Recommended to put meaningful error messages. | 20/04 | Thanaphol | Whenever users attempt to submit an invalid form, an error message will pop up telling them it wasn’t able to be submitted. | | Provide a log out message and a signup message to let the user know. | 23/04 | Thanphol | When the user signs up or signs out, message pops up letting them know. This makes the user experience more friendly. | | Recommended the use of breakpoint to ensure the signup pages still look clean on a smaller viewport | 23/04 | Aleeza | Included a break point that restructures the content to ensure application is user friendly even on smaller viewports. | | Recommended redirects to pages after form submission | 20/04 | Aleeza | Recommended the user should be redirected to the sign in page after creating their login details. Makes the application more user friendly. | | Instructed me to separate the attributes of Card | 19/04 | Adnan | Each card had 2 descriptions separating them would make implementation far easier for voting page. | | Told me to use contrast checkers when deciding on colours for main | 22/04 | Adnan | Ensured accessibility for visually impaired user who may struggle with small fonts or contrast. | |

|  |
| --- |
| Giving constructive feedback |
| **Guidance:** List here, in a table, the CONSTRUCTIVE feedback you have given to your team-mates and what was the problem you were trying to solve.  Feedback: **Be descriptive and specific** – but do not use quotations. Evidence of this should also be in the group’s Trello or other tool used by the group  Date Given: when was the feedback given  Source: To whom did you give the feedback  The problem you were trying to solve: What did you see as the issue? |
| |  |  |  |  | | --- | --- | --- | --- | | Feedback | Date given | Given to | The problem you were trying to solve | | Informed her absolute paths were not allowed and use static directory | 20/04 | Aleeza | Absolute paths were not allowed and wouldn’t function well in the coursework. | | Use proper validation so that votes with empty input fields cannot be submitted. | 20/04 | Aleeza | Users can submit without selecting a team, which would not work well. | | Recommended that a simple UI design is used | 23/04 | Hasan | Overcomplicated UI design is bad for user experience. | | Told him to test each attribute of user | 24/04 | Hasan | Testing needs to be thorough to ensure minimal bugs and smooth experience for user. Introducing a bug in the User can lead to failure in everywhere else in the project. | | Recommended layout for summary page | 22/04 | Adnan | Recommended a clean design to show each summary clearly for the user. | | Told him about there is specific requirements for each summary page | 23/04 | Adnan | Users can see summaries based on their role. It is made clear in the coursework brief. | | Reiterated to him to use a consistent design throughout the pages. | 19/04 | Thanaphol | UI was inconsistent with other pages which made a poor user experience. | | Reminded to check the website complied with accessibility guidelines. | 22/04 | Thanaphol | To ensure everything was easy to read for as many users as possible. | |
| **Professional Conduct – Reflection on the mentor and industry input - (10 marks)** |
| Guidance: Reflect here on the guidance that you have received from both the industry mentors and the Engineers of the company. Reflection on both the understanding/information received on the industry/profession and the specifically on the application implemented. Be specific, but do not use quotations or names of individuals. In particular   * What challenges related to the cwk did you discuss with the mentor, reflect on guidance provided and describe on how guidance was applied to the challenge. * Describe two other topics that were discussed with the mentor, reflect on response received, and how may have influenced cwk work or personal/professional matters related to future employment. * What is the most important guidance that you received from the visiting sky engineer related to the cwk? * How did you apply the advice you received to the cwk? * How will you apply the advice in future studies or employment? |
| Throughout this coursework, I had the chance to get advice from both an industry mentor and a visiting Sky engineer, and honestly, it made a real difference in how I approached the project and how I will approach future projects.  One of the main challenges I discussed with the mentor was how to collaborate effectively with my group. The mentor stressed the importance of having regular meetings to stay aligned and make sure everyone was on track. Based on this advice, I set up weekly meetings with my group, which really helped improve communication and made it much easier to identify and solve problems early.  Another important piece of advice was to keep in regular contact with team members outside of the meetings, just to quickly check on progress and help keep everyone motivated.  Lastly, the mentor emphasized that completing my part of the work on time was crucial because any delay on my end could slow down the whole group. This advice made me take deadlines much more seriously, and I found myself planning better to avoid last-minute rushes.  The visiting Sky engineer gave some of the most impactful guidance regarding the coursework. He helped clarify the user requirements for the project and explained how important it is to not only meet the client's stated needs but also to dig deeper and find additional needs that they might not even realize they have. This really changed the way I approached the project, making me think more critically about the features we were implementing and whether they meet what the client had in mind.  After getting that advice, I took some time to go through the coursework brief on my own to make sure I fully understood what was expected. I wanted to make sure I wasn’t just following the project loosely but hitting all the key requirements properly. While reviewing it, I realised something I had gotten wrong, at first, I had set it up so that the admin would assign user roles manually. But after looking at the brief more carefully, I noticed that users were supposed to pick their own role when they registered for the Health Check site. Spotting that meant I could make the necessary changes and helped make sure our project matched what the coursework was actually asking for.  Going forward, I will apply this advice in both future studies and employment. In future coursework, I will continue to prioritize group communication and time management to keep projects running smoothly. In a professional setting, I’ll make sure to thoroughly understand client requirements and think beyond them to create even better solutions. I believe following this guidance will help me be a stronger, more reliable developer and team member. |