5COSC021W Coursework 2 - INDIVIDUAL template 2023_24

- 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	Samuel
Forename	Cucicea
Registration No:	W1873364

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	Ahsan Ziad Velislav Penev Pavel Strelcovs Mehedi Ahsan

1. Code – Code functionality – individual element and integration to application (20 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.
- 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 our project, we divided the work into the following apps:

- **Authentication – Responsible for the authentication part of the project.**
- **Bookings – Responsible for the bookings page journey.**
- **InventorySystem (core app) – The main app.**
- **Products – Responsible for handling the products.**
-

My primary contributions to the project were focused on developing critical functionalities within the Authentication and Booking segments. I was responsible for implementing both front-end and back-end components, but mostly back-end ensuring seamless integration and functionality.

Authentication

Contributors: Backend mainly lead by me, but all members of the team done something at some point in each file, and front end same, everyone changed a color or done some positioning at some point.

views.py:

- Implemented user login and session management functions to handle user authentication processes securely; also created business logic(in views.py) for user sign-up such that needs to be approved by admin and until then gets rendered the approval page.
- Wrote the entire authentication logic, integrating Django's authentication system to security measures such as password validation that it cannot be as the name of the user or length of the password which guides user to more secure.

templates/authentication/user_login.html:

- Developed the HTML structure for the user login page, providing a user-friendly interface for authentication.

static/authentication/css/user_login.css:

- Styled the login page to align with the application's overall design aesthetic, enhancing the user interface with custom CSS.

models.py:

- Description: Defines the user model extended from Django's default User model, adding additional fields such as phone number and address.
- **Functionality**: Includes classes like CustomUser, which inherits from AbstractUser, providing the flexibility to add more fields to the user model as required.

Personal Contribution: Designed the extended user model to include additional fields and implemented methods to handle user-specific queries.

- I**ntegration**: Interacts with views.py for fetching and storing user data during registration and login processes, such that user need to be approved by admin.
- Team Contribution: All team members contributed to code refactoring after deciding to deviate from the initial database design.

Admin.py:

- In admin, I created a class called CustomUserAdmin which registers CustomUser. I was responsible for integrating it with the model, ensuring the admin has necessary columns for managing the users, such as date joined, name, and email. Also added a method for hashing user passwords to enhance security and privacy. Lastly, I created actions for the admin which call functions such as approve user, revoke user, or delete user.

-

Bookings

- **Contributors:** All team members were involved in the bookings app, especially in the backend. However, the main developers for the model, admin, and views to handle the logic were me and Velislav, with others focusing more on the front end.

  files

  templates/booking/main.html and **static/booking/css/main.css:** Team members styled the main booking page from the initial rough page, with backend done, allowing users to view and manage bookings effectively. I contributed personally to the layout so that the CSS file was shareable across multiple pages for code reusability.

  views.py: Implemented backend functionalities for booking history management and processing booking requests, involving complex database interactions to handle user inputs, validate dates, and retrieve booking history from the booking model.

  - **Personal Contribution:** Developed robust backend logic to support booking operations, including data validation, primarily in the model and admin.
  - **models.py:** Modified the initial booking model created initially by us to include more attributes which later in the views were used to associate a booking better to a product respectively to user.
  -
  - **Admin.py :** includes methods such us , approve booking ,return_item.short_

reject booking, send_alert ,return item ,mark_as_overdue remove_as_overdue ,remove_returned,mark _returned. The above actions have the functions specific, also in the view is integrated such that when admin marks as returned it adds the dates in the admin Django administrations column.

Products

- **Contributors:** All team members were involved in the products app however the main contributor were Velislav and Me on the blackened all ensured that followed the design from the model in the database however we had to deviate a bit from the design as we came to conclusion that in design was not fitting complelty our needs and the business logic.

Templates

- **templates/products/products_page.html:** Provides the HTML structure for displaying products on the website, also extends the base html file.

Static

- **static/products/css/products_page.css:** is the specific css for the products page.

models.py:

- **Description**: Establishes the data models for products, including fields for name, price, stock quantity, and description.

admin.py:

- Functionality: Customizes the Django admin for product management, enabling administrators to easily add, edit, and delete product entries, also includes fiels such as status, quantity, date added for a comprehensive reporting.
- **Personal Contribution:** Configured the admin interface to include more columns in the admi.
- **Integration**: Interacts with the admin and views.py to provide a reliable backend for product operations. Such that when a product is out of stock then is not available for booking and is not added in the bookings page(cart)

views.py:

- **Integrations:** Works in conjustion bookings page and bookings_history page where all the bussines logic is ensured. The following functions were adopted to meet the functionalities in the front end in the forms: - create_booking, choose_dates, remove_from_booking_cart ,choose_dates , confirm_booking, bookings_page(Basically renders the bookings page).
-

Over all as a Team Leader I had to help my team with backend more ensuring that we follow the design and we don't miss the required functionalities.

2. Your code – Code Quality – Maintainability (10 marks)

Guidance:

- 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.
- 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.

The maintainability of the code was a central focus throughout the development of our application. After initial challenges, including multiple team members working independently which led to redundant code, we established a set of coding standards and project structures that significantly improve our code's maintainability.

We organized our Django application into distinct apps for each major component:

- Authentication, Bookings, InventorySystem, and Products. This separation adheres to the Single Responsibility Principle, ensuring that each app has a clear focus and reduces complexity.
- Within each app, we structured the directories consistently:
  - **static/authentication/css** for CSS files to style the authentication-related pages.
  - **static/authentication/images** for all visual assets used within the authentication app.
  - **templates/authentication** for all HTML templates, ensuring a clear pathway from views to their respective templates.

Each function and class has comments where necessary to explain the purpose and logic.

- **Modular CSS:** CSS files are organized to maximize reusability. Common styles are defined in **base.css**, for example in for sidebar and footer. Also, authentication pages share same CSS files.
- The bookings history and bookings page share same CSS, and some input validation from the view.
- In **views.py** within the Authentication app, the login and registration logic is encapsulated in functions that are used across different user interaction scenarios, such as logging in and signing up.

3. Your code – Code Quality – version control (10 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.

We used GitHub for version control and implemented the following strategy:

- **Repository Setup**: We created a repository named "Inventory System," which included a ReadMe file in the main branch. We utilized feature-based branching, where each feature was developed in a separate branch to ensure isolated testing before integration.
- **Code Reviews and Merging**: After completing work on a feature, we created a pull request. One of us would review the code and approve the merge request. This process helped ensure that only quality code was merged into the main branch. In cases of errors or accidental changes, we reverted to the latest working commit, allowing us to seamlessly continue development.
- **Commit Frequency:** During meetings, we agreed to make frequent commits. This ensured that every team member could continue working from the most recent and stable feature updates or after bug fixes.
- **Merge Conflict Resolution**: We resolved merge conflicts collaboratively, ensuring that each team member agreed with the changes and understood the updated codebase. Our approach minimized merge conflicts and promoted a thorough understanding of the codebase among all team members.

This strategy helped us maintain a high-quality, stable codebase and ensured that our work was compatible with the efforts of other team members.

Example commit message:

```
-User will not be able to make a booking if is not in stock
- messaging added such that user is informed if successful or not
- other things look for yourself
- more validations for dates choosing in views.py
- typo fixed, booking to booking in url.py
```

4. 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.

Open WebApp	www.schoolinventorymanagement.com	School inventory management landing page	School inventory management landing/welcome page	yes	yes	
Student enter correct Username/password & press log in	username : student password : Student1!	Login successful	Login successful redirects to products	yes	yes	
Staff enter correct Username/password & press log in	username : staff password : Staff1!	Login successful	Login successful redirects to products	yes	yes	
Forgot password is used by user to change password	User enters new password	Password updated	When pressed nothing happens	no	no	Not Implemented
Admin enter correct Username/password & press log in	username : aaa@gmail.com password : aaa	Login successful redirects to admin panel	Login successful redirects to admin panel	yes	yes	
Login	Test Case ID			Login-B		
Login - Test Case	Test Priority			High		
N/A	Post-Requisite			N/A		
Action	Inputs	Expected Output	Actual Output	Test Browser	Test Result	Test Comments
Open WebApp	www.schoolinventorymanagement.com	School inventory management landing page	School inventory management landing/welcome page	yes	yes	
Student enter incorrect Username/password & press log in	username : false password : false	Username or password incorrect, try again or Sign Up	login Unsuccessfull	yes	yes	
Staff enter incorrect Username/password & press log in	username : false password : false	Username or password incorrect, try again or Sign Up	login Unsuccessfull	yes	yes	
Admin enter incorrect Username/password & press log in	username : false password : false	Username or password incorrect, try again	login Unsuccessfull	yes	yes	

ber	Action	Inputs	Expected Output	Actual Output	Test Browser	Test Result	Test Comments
1	Open WebApp	www.schoolinventorymanagement.com	School inventory management landing page	School inventory management landing/welcome page	yes	yes	
2	Select SignUp and fill in personal details	Name - TestName Surname - TestSurname Phone Number - 0123456789 Email – test@test.com Password – Test@123	User account is created, and the account needs approval.	User account is created, and the account needs approval.	yes	yes	
3	Select Contact Us Button for additional help when signing up	Clicking on "Contact us" button	Contact us page opens	Contact us page opens	yes	yes	

enario ID	SignUp	Test Case ID			SignUp-D		
se Description	SignUp - negative Test Case	Test Priority			High		
uisite	N/A	Post-Requisite			N/A		
ecution Steps :							
ber	Action	Inputs	Expected Output	Actual Output	Test Browser	Test Result	Test Comments
1	Open WebApp	www.schoolinventorymanagement.com	School inventory management landing page	School inventory management landing/welcome page	yes	yes	
2	Select SignUp and fill in personal details (invalid email)	Name - TestName Surname - TestSurname Phone Number - 0123456789 Email – test!test.com Password – Test@123	User receives an error message indicating invalid email format.	User receives an error message indicating invalid email format.	yes	yes	
3	Select SignUp and fill in personal details (invalid number)	Name - TestName Surname - TestSurname Phone Number - 012345678 Email – test@test.com Password – Test@124	User receives an error message indicating invalid phone number format. Must be 10 numbers	User receives an error message indicating invalid phone number format. Must be at most 15 numbers	yes	yes	
4	Select SignUp and fill in personal details (invalid password)	Name - TestName Surname - TestSurname Phone Number - 0123456789 Email – test!test.com Password – Test	User receives an error message indicating password is not secure. Must include numbers, capital letters, and a special character	User receives an error message indicating password is not secure. Too common	yes	yes	

Test Case Description	Admin Positive Test Case	Test Priority			High		
Pre-Requisite	N/A	Post-Requisite			N/A		
Test Execution Steps :							
S.Number	Action	Inputs	Expected Output	Actual Output	Test Browser	Test Result	Test Comments
1	Admin Login	Username - Admin1 Password - Admin1!	Access granted	Access granted	yes	yes	
2	Add equipment	Add new equipment	Equipment shows in the list to other users	Equipment shows in the list to other users	yes	yes	
3	Approve sign up	User sign up request approved	User account is activated and premissions are given according to staff/student policy	User account is activated and premissions are given according to staff/student policy	yes	yes	
4	Generate reports	Admin writes and records a report, generates an inventory status report, equipment usage report and warranty report	Reports must be generated and shown on the system	n/a			
5	Approve equipment booking	Admin approves user equipment	Equipment is given to user and shown on record	Equipment is given to user	yes	yes	
6	Remove equipment	Admin removes existing equipment	Equipment removed from the list	Equipment removed from the list	yes	yes	
7	Update Existing product details	Admin navigates to product management and changes product count to 20.	Equipment to show item count of 20.	Equipment to show item count of 20.	yes	yes	
8	Update user account details	Admin navigates to user account details and inputs an existing user ID, then changes account type to Admin	Given user account is updated into an admin role and is given administrative level access	Given user account is updated into an admin role and is given administrative level access	yes	yes	
9	Remove user account	Admin navigates to user management and selects remove user account. Inputs an existing user ID and presses button to remove account.	Given user account is deletd from the system.	Given user account is deletd from the system.	yes	yes	

Test Scenario ID	ViewEquipment	Test Case ID			Admin-J		
Test Case Description	Admin Negative Test Case	Test Priority			High		
Pre-Requisite	N/A	Post-Requisite			N/A		
Test Execution Steps :							
S.Number	Action	Inputs	Expected Output	Actual Output	Test Browser	Test Result	Test Comments
1	Admin Login (wrong details)	Invalid details	Error message, no access	Error message, no access	yes	yes	
2	Add equipment(missing type)	Equipment details are not correctly filled in	Error message, letting admin know whats missing	n/a	no	no	
3	Approve sign up(approve nonexistent sign up)	Admin attempts to approve sign up that does not exist	Sign up must not show up in the list	Sign up must not show up in the list	yes	yes	
4	Approve equipment booking (staff equipment given to student)	Admin approves staff equipment for a student	Error message, not allowing admin to grant access	n/a	no	no	
5	Remove equipment(equipment that does not exist)	Admin attempts to remove nonexisting	Equipment not in the list	Equipment not in the list	yes	yes	
6	Remove user account (invalid ID)	Admin inputs an invalid ID of a nonexistent account	Error message, letting admin know account with such Id does not exist	n/a	no	no	

.Number	Action	Inputs	Expected Output	Actual Output	Test Browser	Test Result	Test Comments
1	Interface interaction	User interacts with the interface	User interaction brings them to chosen location	User interaction brings them to chosen location	yes	yes	
2	Password covered	User enters password	********	********	yes	yes	
3	Missing information is shown	User enters missing information into sign up	Highlighted red whats missing	Highlighted red whats missing	yes	yes	
4	Out of code equipment is highlighted	Locate out of code equipment	Highlighted red	Highlighted red	yes	yes	
5	Date close to return deadline highlighted	User list of equipment booked, locate close to end date	Highlighted orange	n/a	no	no	
6	Enlarge text	Press button to enlarge text	Text enlarged	n/a	no	no	
7	Make text smaller	Press button to make text smaller	Text size reduced	n/a	no	no	
8	Equipment Limit bookings	Student attempt to book more than 3 equipment	Error message shown, limit reached	no	no	no	
9	Login button	User enters correct login and clicks on login button	login successful and brings user to main product list page	login successful and brings user to main product list page	yes	yes	

Test Scenario ID	ViewEquipment	Test Case ID			ViewEquipment-E		
Test Case Description	ViewEquipment List Positive Test Case	Test Priority			High		
Pre-Requisite	N/A	Post-Requisite			N/A		
Test Execution Steps :							
S.Number	Action	Inputs	Expected Output	Actual Output	Test Browser	Test Result	Test Comments
1	Log in, navigate to equipment list	Open Product page and navigate through equipment list.	Product page opens and list of equipment is displayed	Product page opens and list of equipment is displayed	yes	yes	
2	Log in, search for equipment in the list	User enters Laptop into search option	All laptops products are displayed	All laptops products are displayed	yes	yes	
3	Log in, filter equipment by various criteria	User filters by availability	All available equipment is shown	All available equipment is shown	yes	yes	
4	Log in, sort equipment by predefined fields	User selects Onsite in sorting field	Only onsite products are displayed	Only onsite products are displayed	yes	yes	
5	Open product details	User selects a product and clicks on it to open a page.	Product details page successfully opens	Product details page successfully opens	yes	yes	

Test Scenario ID	ViewEquipment	Test Case ID			ViewEquipment-F		
Test Case Description	ViewEquipment List Negativbe Test Case	Test Priority			High		
Pre-Requisite	N/A	Post-Requisite			N/A		
Test Execution Steps :							
S.Number	Action	Inputs	Expected Output	Actual Output	Test Browser	Test Result	Test Comments
1	Log in, search equipment that isnt on the list	Navigate to equipment list and enter dog	Nothing is displayed	Nothing is displayed	yes	yes	
2	Equipment declined to student	Student selects equipment that is for staff only	Item is not selected	n/a	no	no	Not Implemented
3	Onsite product is booked for more than a day	User tries booking an onsite product from 10/03 to 14/03	Booking unsuccessful message shows that product is oniste and can only be booked for a day and used onsite.	n/a	no	no	Not Implemented
S.Number	Action	Inputs	Expected Output	Actual Output	Test Browser	Test Result	Test Comments
1	Book Equipment	User books available equipment, Selecting a laptop for 1 month	Booking is processed and status of approval is visible with a description of the booking showing the product and date of return	User can choose product to add to cart and choose the date of return when booking. Booking is then processed and a message poppes up notifying the user of status of approval.	Yes	Yes	
2	Get Overdue Alerts	Equipment return date has gone past the return date	Alert should be sent to the user	Alerts are sent in a red font inside bookings history	Yes	Yes	
3	Booking History Tracking	User visits previous bookings	All past bookings are shown	All past bookings are shown	Yes	Yes	
4	Booking a previous booking	User books a previous booking from history bookings page	Booking is processed successfully without more inputs from user	User can book previous booking, needs to add a new return date	Yes	Yes	
5	Cancelling an existing booking	User opens bookings page and navigates to cancel booking. Inputs booking ID and selects cancel booking button	Booking is cancelled successfully with a confirmation page showing	Only admin can cancel/delete booking	no	no	Part was not implemented

Test Scenario ID	ViewEquipment	Test Case ID			Reservation-H		
Test Case Description	Reservation Negative Test Case	Test Priority			High		
Pre-Requisite	N/A	Post-Requisite			N/A		
Test Execution Steps :							
S.Number	Action	Inputs	Expected Output	Actual Output	Test Browser	Test Result	Test Comments
1	Booking unavailable equipment	Student attempts to book unauthorised equipment	Student is declined automatically, message must be shown	Student is declined automatically, message must be shown	yes	yes	
2	Booking out of stock item	User books an item that is booked by other users	Booking is declined, message explaining why shows up	Booking is declined, message explaining why shows up	yes	yes	
3	Receive overdue equipment alerts	Return date hasn't been passed for item	No alert is shown	Alert shown inside bookings history	yes	yes	
4	View empty history	User with no past bookings checks history	A message must show up no bookings were recorded	A message displayed for no previous bookings made	yes	yes	
5	Cancelling a nonexistent booking	User attempts to cancel a booking with an invalid Booking ID	A message displays saying that there are no active bookings with given ID	Only admin can cancel/delete booking	No	No	Part was not implemented

5. Professional conduct – Communication (10 marks)

5.1 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	Source	How you used it
Change the colour when approved to green.	30/04/2024	Pavel	Done as requested, and it looked better.
Make the backend part for the user when choosing dates for the booking, also make them be disabled, the buttons when quantity of the product is out of stock.	01/04/2024	Pavel	We had a meeting, since was near the submission date I was not sure I will be able to implement it at first but manage to do it and it was a consistent and good feedback.
When user tries to log in in the admin panel, it should show a message in red that invalid credentials or not authorised	27/04/2024	Pavel	Asked to help me style it and added the feature. It provides guiding for student.
Improve Admin panel styling	30 april	Velislav	The panel was indeed a bit too basic, and it looked better after
Improve admin panel for user	26 april	Velislav	Due to high number of users admin should be able to filter, sort.
Booking history looks good maybe just add some spacing around start date, and end date to visualize them better	28/04/2024	Ahsan	Was a change for matters of seconds and it made easier to visualize for the user
Maybe the messages it would be better to see them in the footer of the page which would be fixed	29/04/2024	Ahsan	The change included just simple positioning and guides the user if was successful change or request or not.
If there is a way you can. Change from CustomUser in admin to Users	22/04/2024	Mehedi	Added a Metta class in the model.py in the authentication and changed it accordingly

The admin page look great .	01/04/2024	Mehedi	Imported jazmin for enhanced look in admin panel which team members liked it.

5.2 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
The positioning of the input type date should be displayed in block.	26/04/2024	Pavel	Positionings of the side buttons on the bookings page.
Create a function in the view for request booking button in the bookings page.	29/04/2024	Pavel	To create a function for the button in the form in the bookings page.
Adjust the product card class in the product page to fit better in the container div and position to right the apply filter button.	25/04/2024	Pavel	Positioning and styling in the Product Page.
Adjust in admin such that if rejects a booking, the quantity gets added back	26/04/2024	Velislav	The product should not be held after admin rejects. Should be available again if not else.
Make the filtering and sorting functional in bookings history	289/04/2024	Velislav	Filtering and Sorting should be functional in the bookings history page such that user can filter and sort by different attributes.
Add more products in the database and relevant pictures to the products.	02/04/2024	Ahsan	Few products added initially to database only for testing purposes, needed to be added the full list.
Improve styling in the sidebar, also make it start from the nav bar	28/04/2024	Ahsan	Styling after some more page links were added to the sidebar.
Contact Us page should have a side bar as well	29/04/2024	Mehedi	To follow up with mock-ups design and for user to be able to switch again to products page.
The picture in the side bar , has height too big correct it such it doesn't push the links to bottom .	30/04/2024	Mehedi	The page links were pushed down because of the picture's height.
