Anazom Release Summary (6 pages MAX)

Team members

Name and Student id	GitHub id	The number of story points (or ideal hours for tasks) that a member was an **author**.
Paul Scarmardo	PaulScarmardo	7
Yuexin Xu	yx155	13
Ayden Kilpatrick	Ayden-Kilpatrick	4
Jessica Oramous	jessoramous	2

Each group member is responsible for counting their own story points. It is the group leader's duty and responsibility to make sure they are accurate. Please keep in mind that we will check your GitHub stats (go to: "Graphs" on your GitHub project page, for example). Note, if your email and GitHub id are not linked properly you will not be counted properly.

You will lose 1 mark if links below are not clickable.

Project summary (max one paragraph)

Elevator pitch description at a high-level

Our project is an e-commerce site named Anazom. The site allows different types of users (buyers, sellers, admins) to login and perform their desired actions before logging out. Buyers can search and purchase products from sellers, and admins can oversee the actions of these buyers and sellers. The site also includes an option for all types of users to view past transactions and allows buyers to return and receive refund for products that have been purchased previously. The site is implemented using Python and HTML, with the framework being Django.

Velocity and a list of user stories (for **example**) and non-story tasks for each iteration

(make sure the iteration is a clickable link to the milestone/sprint on GitHub)

Total: 13 stories, 26 points over 13 weeks

Iteration 1 (9 stories, 16 points)

US #1: Approve/Block Seller's Products [1 points] [Status: Pushed]
US #2: Search for Different Users [2 points] [Status: Pushed]
US #3: Add item to Cart [1 points] [Status: Pushed]
US #4: Search for an Item [1 points] [Status: Pushed]
US #5: Adding Item to Site [3 points] [Status: Pushed]
US #6: Changing Item on Site [2 points] [Status: Pushed]
US #7: Accept Payment [1 points] [Status: Pushed]
US #8: Login/Logout [3 points] [Status: Pushed]
US #9: Return Product [2 points] [Status: Pushed]
SRS Documentation: [Status: Done]
Set up GitHub: [Status: Done]

Update README: [Status: Done]
Meeting Documentation: [Status: Done]
Plan for next sprint: [Status: Done]
Have meaningful commit messages: [Status: Done]
Commits referenced with issue: [Status: Done]
Pull Requests: [Status: Done]
Equal Participation: [Status: Done]

Iteration 2 (4 stories, 10 points)
US #10: Checkout [3 points] [Status: Pushed]
US #11: View Shopping Cart [2 points] [Status: Pushed]
US #12: Remove Item from Cart [2 points] [Status: Pushed]
US #13: View Order History [3 points] [Status: Pushed]
Use Case Diagrams: [Status: Done]
UML Class Diagrams: [Status: Done]
UML Sequence Diagrams: [Status: Done]
UML State and Activity Diagrams: [Status: Done]
Architectural and Component Design Diagrams: [Status: Done]
Task Breakdowns: [Status: Done]
Workload Balance: [Status: Done]
Plan for Next Release: [Status: Done]
Frequent Meetings: [Status: Done]

Iteration 3 (0 stories, 0 points)
Implement 3 of 4 features: [Status: Done]
Software Test Plan document: [Status: Done]
Continuous Integration: [Status: Done]
Acceptance Testing: [Status: Done]

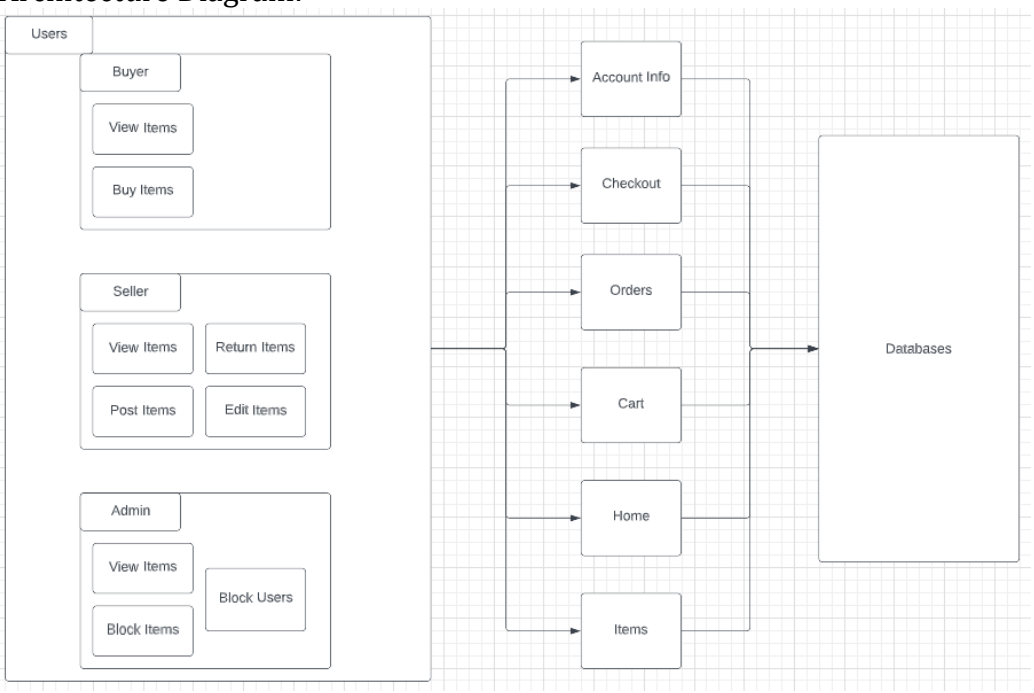
Iteration 4, Release (0 stories, 0 points)
Implement last feature: [Status: Done]
Project Name Release document: [Status: Done]
Continuous Integration: [Status: Done]

Overall Arch and Design

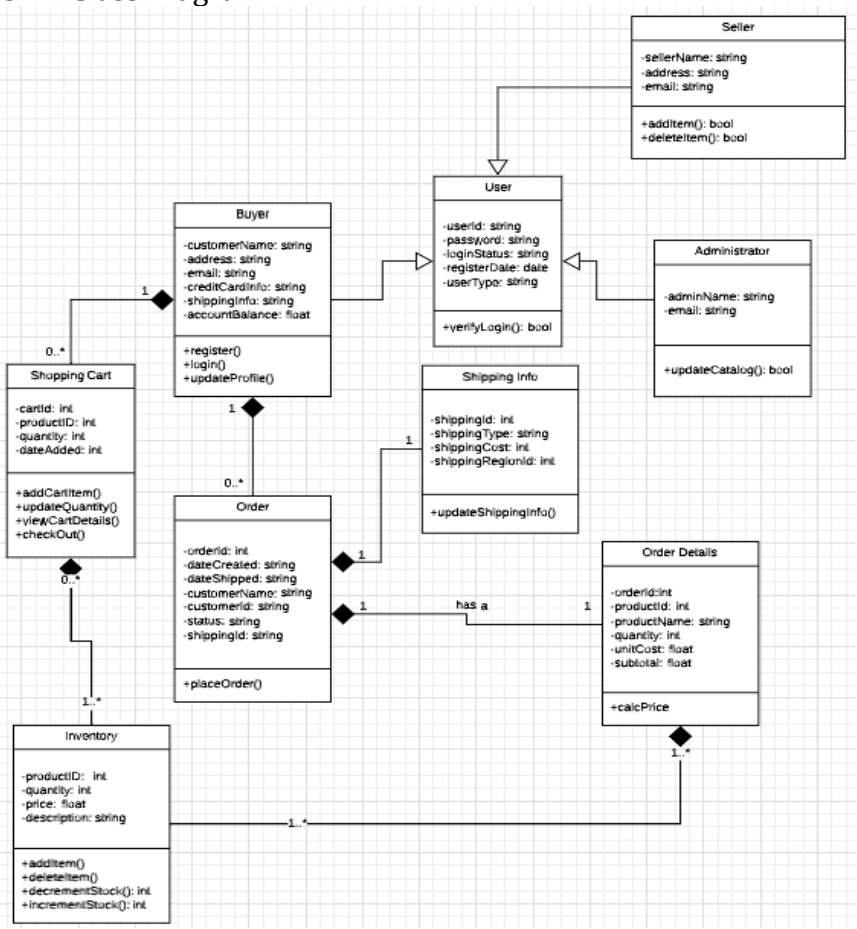
Show us the overall architecture (block diagram) in your system with an architecture diagram.
Show the UML class diagram for your system. If you have multiple packages, show the diagram for at least one package that has more than 10 classes. You can also include these diagrams in your stories on GitHub (by providing URLs).

(SEE NEXT PAGE)

Architecture Diagram:



UML Class Diagram:



Link to architecture diagram:
https://github.com/PaulScarmardo/Group9/wiki/Architecture-and-Components-Diagram
Link to UML class diagram:
https://github.com/PaulScarmardo/Group9/wiki/UML-Class-Diagram

Infrastructure

For each library, framework, database, tool, etc

Name and link
Max 1 paragraph description of why you are using this framework.
Max 1 paragraph description of other alternatives and why you didn't choose them.

Django: https://www.djangoproject.com/
We are using the Django framework for our project because we chose to code our back end with Python. Our group members were all familiar with Python and Django just happens to be a popular framework that supports building Python based websites. The team leader of our group also has experience with using Django on his previous web application projects and was able to help us start with this project's coding process.

There are other alternatives for Python based frameworks such as Flask and Web2py, both of which are used for web development. We did not choose to use them for this project because none of our group members had any previous experience with using these frameworks. These frameworks are also less well-known compared to Django and therefore have less resources available.

SQLite3: https://sqlite.org/index.html
We are using SQLite3 as our database for the website because it is the default database the Django uses. We did not have to manually set up a database because within the framework that we used, there was a file that links to the database, and we were able to modify and extract information from that file directly. Being able to do so makes the coding process much simpler, hence why we decided to go with the default option.

Other alternatives for databases would include Oracle and IBM, both of which are widely used for web development. However, as mentioned, these databases are not part of the default databases included in the Django framework and therefore would require manual implementation. We chose to avoid unnecessary implementations and instead focus more on creating a functioning website.

CircleCI: https://circleci.com/
We are using CircleCI as our integration tool for automated unit testing because it is free and simple to set up. Automated testing tools are very helpful when it comes to ensuring the proper functionality of the website at all times. Instead of having to run the test cases manually after making a change to the code, CircleCI does it for us.

There are other automated testing tools such as the one available on GitHub. However, we did not choose to use that one because CircleCI was the first tool that we were introduced to. After reading through a few guides on how to setup CircleCI, we were able to link it to our repository without any problems.

Name Conventions

List your naming conventions or just provide a link to the standard ones used online.

For example: Java naming conventions

Standard Python naming conventions:
https://visualgit.readthedocs.io/en/latest/pages/naming_convention.html

Code

Key files: top **5** most important files (full path). We will also be randomly checking the code quality of files. Please let us know if there are parts of the system that are stubs or are a prototype so we grade these accordingly.

File path with a clickable GitHub link	Purpose (1 line description)
https://github.com/PaulScarmardo/Group9/blob/main/store/views/login.py	Processes user login
https://github.com/PaulScarmardo/Group9/blob/main/store/views/home.py	Loads store page
https://github.com/PaulScarmardo/Group9/blob/main/store/views/signup.py	Creates a new user account
https://github.com/PaulScarmardo/Group9/blob/main/store/views/addProduct.py	Adds a new product to site
https://github.com/PaulScarmardo/Group9/blob/main/store/views/checkout.py	Checks out buyer's shopping cart

Testing and Continuous Integration

Each story needs a test before it is complete. If some class/methods are missing unit tests, please describe why and how you are checking their quality. Please describe any unusual/unique aspects of your testing approach.

List the **5** most important unit test with links below.

Test File path with clickable GitHub link	What is it testing (1 line description)
https://github.com/PaulScarmardo/Group9/blob/main/store/tests/test_login.py#L44	Logging in with the wrong password
https://github.com/PaulScarmardo/Group9/blob/main/store/tests/test_buyer.py#L39	Adding an item to shopping cart
https://github.com/PaulScarmardo/Group9/blob/main/store/tests/test_seller.py#L33	Inputting a negative price
https://github.com/PaulScarmardo/Group9/blob/main/store/tests/test_admin.py#L39	Suspending a buyer account
https://github.com/PaulScarmardo/Group9/blob/main/store/tests/test_seller.py#L38	Inputting a negative stock quantity

List the **5** most important acceptance tests with links below.

Test File path (if you automated the test) or as comments in Github issues (if it's done manually) with clickable GitHub link	Which user story is it testing (1 line description)
https://github.com/PaulScarmardo/Group9/wiki/User-Stories#acceptance-testing-2	User Story 3 – Add item to Cart

https://github.com/PaulScarmardo/Group9/wiki/User-Stories#acceptance-testing-4	User Story 5 – Adding Item to Site
https://github.com/PaulScarmardo/Group9/wiki/User-Stories#acceptance-testing-7	User Story 8 – Login/Logout
https://github.com/PaulScarmardo/Group9/wiki/User-Stories#acceptance-testing-8	User Story 9 – Return Product
https://github.com/PaulScarmardo/Group9/wiki/User-Stories#acceptance-testing-9	User Story 10 – Checkout

Describe your continuous integration environment. Include a link to your CI.

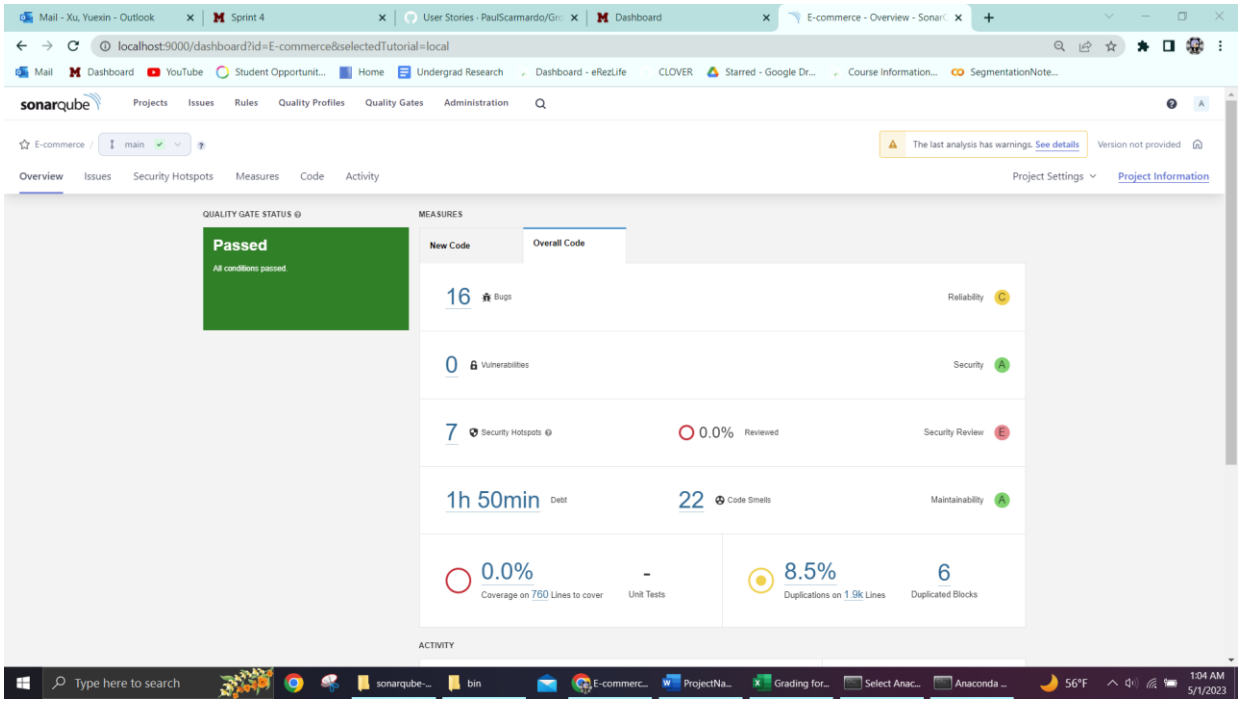
In order to create our continuous integration environment, we linked CircleCI to our project's GitHub repository. CircleCI will perform unit testing automatically whenever a change is made to the code section of our repository. After it has finished running the tests, it will create a report indicating whether the tests have passed or failed. The way CircleCI runs our tests are configured in the config.yml file, where the testing command can be found.
A link to our CircleCI can be found here:
https://app.circleci.com/pipelines/github/PaulScarmardo/Group9

Describe the choice of the static analysis tool and how do you run it. The static analysis tool should analyze the language that is used in the majority of your source code.

The static analysis tool that we used is SonarQube. It is a free tool that can be run locally on your machine with minimal configurations needed. It also supports a variety of languages and performs many different types of tests, such as bug detection, security checks, and code smells. The language that we used it for is Python. In order to run SonarQube, we first had to download the community edition of the .zip file. After extracting the file, we executed StartSonar.bat, and that launched a locally hosted server allows us to create a project to start the testing process. We then downloaded a scanner from SonarQube for Python. Lastly, we simply executed the scanner (sonar-scanner.bat) in the directory that contains our source code. The result of the scan is then loaded onto the locally hosted server, where we can view the specific details of the report.

Attach a report as an appendix (not counted for the 6 pages) from static analysis tools by running the static analysis tool on your source code. Randomly select 10 detected problems and discuss what you see.

(SEE NEXT PAGE)



Here are random 10 problems that SonarQube detected:

1. “Specify an exception class to catch or reraise the exception”: This problem refers to a try/except block in the customer.py file where the except portion does not have any specific exception assigned to it. It would be a better practice to have at least one specific exception other than the base case.
2. “Rename this variable; it shadows a builtin.”: This problem refers to a variable name in the cart.py file. The analysis tool suggests that instead of naming the variable “sum”, it should be named to something else because “sum” is a builtin and using it would make the code more difficult to read and maintain.
3. “Define a constant instead of duplicating this literal "user@seller.com" 3 times.”: This is a problem found in test_buyer.py. The string literal is used 3 times in 3 different test cases. A better way to code the file would be to assign this string literal to a variable at the setup method of the test class.
4. “Remove the unused local variable "response".”: This problem is found in test_buyer.py, where the variable “response” is defined and assigned but never used. It was originally intended to be used as part of the assertion statement for the test case, but the statement was changed to testing another element.
5. “Rename class "addProduct" to match the regular expression ^_?([A-Z_][a-zA-Z0-9]*|[a-z_][a-z0-9_]*)$.”: This is a problem found in addProduct.py, where the class addProduct is defined. The analysis tool suggests that a standard naming convention should be followed and the class should be renamed to AddProduct.
6. “Refactor this function to reduce its Cognitive Complexity from 18 to the 15 allowed.”: This is a problem found in checkout.py. Cognitive complexity refers to how hard the control flow of a function is to understand. The tool suggests that the if-statements in that file can be modified to reduce the cognitive complexity.
7. “Either remove or fill this block of code.”: This is a problem found in orders.py,

where there contains an elif statement that is empty (contains only “pass”). The elif statement was originally intended to be used for the admin user class, however, the admin user class is now handled separately, and the condition needs to be removed.

8. “Rename this local variable "categoryID" to match the regular expression ^[_a-z][a-z0-9_]*$.”: This is another naming convention issue, where the variable “categoryID” in home.py should be renamed to a name that is more standard. Something like “category_id” would probably work.
9. “Refactor this function to reduce its Cognitive Complexity from 27 to the 15 allowed.” This is another case of cognitive complexity found in login.py. This time, there is room for dramatic change, allowing the complexity to be reduced by 12 with some rearranging of if-else statements.
10. “Remove this commented out code.”: This is a problem found in admin.py, where there consists of a commented out line containing username, email, and password assignments. The line is actually intended to be only for information purposes, however, a better place for such a thing would be on the README file.