global
Making everyone's day brighter

T&D ECP Hackathon Task Handbook

Marking Criteria and Submissions

Points will only be awarded if a helper agrees that a submission meets the acceptance criteria specified for the task.

If you need any help with any of the submissions, either ask someone in your team or one of the helpers.

Points Available

Backend and frontend – 58 points including tasks 1-14
Product – 60 points including tasks 1-11
Data – 50 points including tasks 1-4
Demo – 5 points
Bonus tasks and extra points

Backend and Frontend

Whenever carrying out frontend/backend tasks, make sure to create a new branch off the main branch.

Name the new branch like so:

- "task-<task_number>-submission"
- e.g.: "task-1-submission"

When you are ready to submit your solution, push your branch to the repo and create a pull request.

Notify a SME/helper of your submission and they will then be able to review your PR and award points. Once the PR has been reviewed and points given, the branch can be merged to the main branch.

Product

When carrying out product tasks, provide any documents as evidence in the repo under a separate folder called "Product".

One of the helpers/mentors will then be able to review your submission and award the relevant points. Inside the folder, create a sub-folder with the task number as the name. As above, points will only be awarded if submission criteria are met.

Data

When working on the data tasks, please create a new table in snowflake and perform any alterations on that and name it as your team's name.

Remember points are awarded for the queries so you should keep them in the console/save as a text file. Points are going to be awarded for the dashboard/visualization, but it does not need to be uploaded on the server.

Frontend and Backend Tasks

Task 1: Add your team's name – Frontend (1 point)

Description:
The first task is to add your team's name into the following files:

- ***"frontend/public/index.html"***
- ***"frontend/src/components/app/Header.tsx"***

Acceptance Criteria:

- Your team's name is visible in the browser tab label.
- Your team's name is visible on the shop website.

Task 2: Change the shop header colour – Frontend (1 point)

Description:
This shop is feeling a bit blue. Change the colour of the shop header to any other colour than blue.

File Locations:

- ***"frontend/src/style/Header.scss"***

Acceptance Criteria:

- The colour of the shop header is a different colour to the dark blue option.
- It can be a different shade of blue.

Task 3: Add to basket – Backend (2 points)

Description:
Products can be displayed on the page. However, clicking ***Add to basket*** does nothing. In ***BasketService.java*** fix the ***addToBasket()*** method. This can be done by implementing the ***add()*** method in ***BaksetRepository.java***. You will need to re-run the backend to see any backend changes on the frontend.

File Locations:

- ***"backend/src/main/java/org/global/ecp/hackathon/app/basket/BasketService.java"***
- ***"backend/src/main/java/org/global/ecp/hackathon/app/basket/BasketRepository.java"***

Acceptance Criteria:

- The ***addToBasket()*** method is fixed.
- Products can be added to the basket.

Task 4: View products in the basket – Frontend (2 points)

Description:

A user should be able to add products to the basket. However, once they have added them and clicked on the shopping cart icon to see what's in their basket, nothing appears! Looks like the problem is in the ***CheckoutPage.tsx*** component.

File Locations:

- ***"frontend/src/components/pages/CheckoutPage.tsx"***

Acceptance Criteria:

- When a user clicks on the shopping cart icon any products that have been added to the basket are shown in the checkout page.

Task 5: Remove from basket – Backend (2 points)

Description

Users should be able to add products to the basket and view them in the checkout page. However, users should also be able to remove products from the basket when clicking the ***Remove from basket*** button. In ***BasketService.java*** fix the ***removeFromBasket()*** method.

Acceptance Criteria:

- The ***removeFromBasket()*** method is fixed.
- Products can be removed from the basket.

Task 6: Show total cost of products – Frontend (3 points)

Description:

User can add and remove products from the basket. Users can also view products in the basket on the checkout page. This task is to calculate and display how much the products in the basket are going to cost.

Fix the ***getTotalCostOfProducts()*** method in the ***getTotalCostOfProducts.ts*** file to return the total cost of all the products in the basket.

File Locations:

- ***"frontend/src/functions/getTotalCostOfProducts.ts"***

Acceptance Criteria:

- The ***getTotalCostOfProducts.ts*** method is fixed.
- When checking out, the total cost of products in the basket is displayed.

Task 7: Calculate shipping – Frontend (3 points)

Description:

User should be able to see the total cost of the products in the basket. However, when clicking checkout and viewing the checkout summary, there is no shipping cost information. Add the shipping cost to the checkout summary.

Shipping is calculated in the following way:

- For every £1, shipping costs 10p, so if something costs £10, the shipping will be £1.
- Then if the total cost of the items is over or equal to £50, then the shipping is free and costs £0.

Users would also like to see the total cost of the order, which is the sum of the total cost of products in the basket and the shipping costs.

File Locations:

- ***"frontend/src/components/checkout/CheckoutSummary.tsx"***

Acceptance Criteria:

When clicking checkout and viewing the checkout summary, the following is displayed:

- The total shipping cost of the products in the basket
- The total cost of shipping + the cost of the products in the basket

Task 8: Implement checkout method – Backend (2 points)

Description:

Users can now see their checkout summary and how much everything is going to cost. However, when clicking the ***Complete Checkout*** button, users navigate back to the products page, but the products are still in the basket. Implement the ***checkout()*** method in the ***BasketService.java*** class and clear the basket of products.

Acceptance Criteria:

- The ***checkout()*** method in ***BasketService.java*** is implemented.
- When clicking the ***Complete Checkout*** button, the user navigates back to the products page and the basket is cleared.

Task 9: Track order history – Backend (6 points)

Description:

This task is to track order history. Every time a user checks out using the ***Complete Checkout*** button on the checkout summary, the following methods are called:

- The ***generateNewOrder()*** method, which calls the backend and sends an order request, which contains all the products that were in the basket on checkout and the total cost of the products and shipping.
- The ***checkout()*** method in the frontend application, which will call the backend, and use the ***checkout()*** method in ***BasketService.java.***

This task is to implement the ***createOrder()*** method in the ***OrderService.java*** file.

To do this take the following steps:

1. Generate a random order ID using the ***UUID.randomUUID()*** method.
2. Generate a ***LocalDateTime*** stamp using the ***now()*** method so that we can keep track of when the order was made.
3. Create an order using the ***Order.java*** object.
4. Return null in the ***createOrder()*** method if the basket is empty.
5. Store the order object in the ***OrderRepository***.
   - You will need to add the map storage yourself.
   - The orders should be stored in a map of UUID's as the keys and the order objects as the values.
   - Look at the ***ProductRepository.java*** file for guidance on how to create the storage and implement methods needed to add orders to the map.
   - Ignore the ***populateProductsMap()*** method in the ***ProductRespository.java*** file.

Acceptance Criteria:

- When clicking the ***Complete Checkout*** button an order request is sent the ***createOrder()*** controller method, which will create and store an order.
- Using the ***@Slf4j*** annotation, a log output of the generated order should also appear in the backend terminal.
- Also, orders should not be made if the basket is empty.

Task 10: View orders history – Frontend (6 points)

Description:

Order history can be tracked. Let's add the ability to view order history in the frontend. There is already a blank orders page.

You will need to do the following:

- Implement ***fetchOrders*** method in ***fetchOrders.ts*** to make an API call to get all our orders from the backend.
- There is also an API URL constant in ***"frontend/src/api/apiConstants.ts".***
- Then map the list of orders into an ***Orders.tsx*** component that can be displayed on the website.
- An example of how to do this can be seen with the ***Products.tsx*** component.
- Make sure to uncomment the ***getAll()*** method in ***OrderRepository.java*** from task 9 before attempting this task.
- There is already an ***Order.tsx*** component set up which can be used with the ***Orders.tsx*** component.
- You will also need to complete the ***OrderBody.tsx*** component. Follow the ***ProductBody.tsx*** as an example.

Acceptance Criteria:

- Once the basket is checked out and an order is created, that order should be displayed on the orders page.

Task 11: Email notification – Backend (5 points)

Description:

Users can now buy products and see any orders made. This task is to send an email notification to the company email server anytime an order is made. The email service can be implemented to do this. The email should contain your team's name and an order summary.

Use Spring's ***SimpleMailMessage*** object to send the email.

- The message from should be set like so:
  - ***" no-reply@<team_name>com".***
- The message should be to: ***"task@submission.com"***
- The subject should contain the order ID.
- The message body should look like this example:

```
Date of order: 2023-05-29
Time of order: 18:41:10.754185
Ordered Products:
Product 1
This is a description for product 1
£1.0
Quantity x 2
Product 2
This is a description for product 2
£2.0
Quantity x 2
Total cost of order: £0.0
```

You can test this by using a fake SMTP server. You can run this server using the following steps:

In the terminal run:

```
docker pull gessnerfl/fake-smtp-server

cd <your/path/to/e-commerce-repo>/e-commerce-app/smtp-server

docker compose up
```

- Then navigate to http://localhost:8080/ to see the smtp server user interface.
- You should see your emails appear there.

Acceptance Criteria:

- We can receive email notifications anytime an order is made.
- The emails we receive should have the order summary as shown in the example.
- The subject should be the order ID.
- The to and from should be as specified in the description.

Task 12: Complete orders – Backend (5 points)

Description:

Orders history can now be tracked and viewed on the frontend. Let's add the functionality to complete an order.

The following steps can be taken:

- In the ***OrderController.java*** class, create a POST controller method called ***completeOrder()*** which will be able to change the completed status of an order to true. The POST method should have the path ***("/complete")***.
- The ***completeOrder()*** method should take in a request parameter called ***orderId*** that is a UUID.
- The method should then call a method in the order service that sets the completed status of the order that matches the given ID to true.
- This method should use the ***@Slf4j*** annotation to log out messages if the order id does not exist or when the order has been completed.
- The POST method should return a no content response entity and the ***OrderService.java*** method should be void.
- You'll also have to implement a ***getById()*** method in ***OrderRepository.java***, to get orders by ID.
- It should also be able to handle null exceptions, if the ID given doesn't not exist in the orders map.

Acceptance Criteria:

- When sending a POST request to the backend with an existing order ID, the completed status of order with that ID is set to true.
- Handle null exceptions if ID does not exist in repository.

Task 13: Complete order button – Frontend (5 points)

Description:

Let's now add the functionality to complete orders on the frontend. Add a button to the ***Order.tsx*** component labelled ***Complete Order***. This button will make a fetch request to our backend and call the ***completeOrder()*** POST method in the order controller. The button should also not appear for already completed orders.

For a similar example, look at how the product component works with the ***Add to basket*** and ***Remove from basket*** buttons. You will also have to implement a fetch request in the ***fetchOrders.ts*** file to send an API call to complete an order.

Acceptance Criteria:

- Each order component on the orders page has a button labelled ***Complete Order*** which will call the backend ***completeOrder()*** POST method and without manually refreshing the page will update the completed status of the orders on the page.
- The button will not appear for already completed orders.

Task 14: Product management page – Frontend & Backend (15 points)

Description:

Currently the products are statically stored. There are only ever 3 products. This task is to have dynamic products that can be added and removed.

Frontend:

- Create a product management page that has the option to add a product.
- This page should have a form where users can provide a product name, a description and product price.
- There should also be a button to add the product.
- Once added the user should navigate back to the products page.
- Also, each product should have a remove button that will remove that product from the inventory.
- Both functions should call the backend.
- Ensure there is CSS too!

Backend:

- The static products will need to be removed from our product repository.
- There are two controller methods already for products, ***createProduct()*** and ***deleteProductById()***.
- The ***create()*** method in the ***ProductService*** will need to be implemented.
- It should generate a unique ID that does not already exist, create a product and then store this product in the repository.
- The method should also return the product.
- Products should also not be able to have the same ID or name.

Acceptance Criteria:

- The static products are removed.
- There should be a product management page that has a form to add products.
- Once a product has been added, a POST call to the ***createProduct()*** backend controller method should be made and add the product to the storage map.
- The new product should be displayed in the frontend application without a manual refresh of the page.
- There should also be a button on each new product to remove this product from our inventory.
- Once the button is pressed, the product should be removed from the frontend view without a manual refresh and our backend map.

Task 15: Extra feature – Final main Frontend/Backend task

- Implement an extra feature. It can be anything!
- After the demos each team will vote for their favourite feature.
- Teams can't vote for themselves!
- The winning feature will earn (15 points), 2nd place will get (10 points) 3rd place will get (5 points).

Product Tasks

Task 1 – Problem Statement (1 point)

Description:
What problem do you want to solve? The problem statement is used to help to understand if the product is successful.

Examples:
- Meet demand w/o manual intervention?
- Make it more accessible for people globally or with disabilities?
- Ticketmaster: You have tickets for events, and you want to sell them.
- eBay: Make it accessible for people to buy and sell items.

Acceptance Criteria:
- A one-line summary of the problem that the product/planned improvements are trying to solve.

Resources:
- You can find an example problem statement and vision statement here: https://globalradio.atlassian.net/wiki/spaces/GPOM/overview?homepageId=2198471754

Task 2 – Vision Statement (2 points)

Description:
Having a vision statement helps to prioritize decisions. This task asks you what is your vision statement for the product at the end of the development process?

Examples:
- DAX outdoor want to become the leading DOOH (Digital Out of Home) in the world.
- Simplifying the checkout process and making the best possible user experience.

Acceptance Criteria:
- 8 – 100 words on your vision for the product at the end of the development process.

Resources:
- You can find an example problem statement and vision statement here: https://globalradio.atlassian.net/wiki/spaces/GPOM/overview?homepageId=2198471754

Task 3 – User Journey Map (3 points)

Description:
Having a map of your user journey helps you to understand what you are trying to achieve and how to achieve it. This task asks you to create a user journey map for how you expect your final product (not necessarily what will be dev done by the end of the Hackathon) to look in terms of an end-to-end journey for the user.

Example:
- Ticketmaster: landing page, seat plan and checkout with address.

Acceptance criteria:
- Have a diagram drawn in a specialist software such as Miro, Figma, Draw.io illustrating your end-to-end user journey.

Resources:
- https://dribbble.com/resources/user-journey-maps-guide?utm_campaign=2022-11-02&utm_medium=email&utm_source=insider-20221102
- https://www.interaction-design.org/literature/article/10-free-to-use-wireframing-tools

Task 4 – Strategy (OKRs) (3 points)

Description:
Your strategy is your vision broken into measurable indicators. The Objectives and Key Results (OKRs) represent what you want to achieve and/or expect as an outcome of your development efforts. They are mostly department based, depending on what you are building, but will be key for your companywide objectives and values.

Examples:
- If wanting to simplify an end-to-end process is the vision - then reducing the check-out process from 11 clicks to 5 is the OKR.
- Or if wanting to have the most engaging social media site is the vision, then the OKR is to increase the engagement rate (as measured by no of likes & comments on a post divided by the number of views or impressions it receives) *100 from X% to Y%.

Acceptance criteria:
- Produce minimum 3 hypothetical OKRs for your project.

Task 5 – Roadmap (5 points)

Description:
Create a Roadmap for 6 months in the future. Think about discovery time, refinement, dependencies and what the MVP Minimal Viable Product will be. Chronologically think about what is best to do. Also include some sort of justification for your decisions.

Acceptance Criteria:
- You will have created a roadmap that illustrates 6 months in the future for your store.
- You will also be able to justify your roadmap decisions.

Task 6 – Competitor Analysis (3 points)

Description:
Competitor Analysis is an essential part of finding out how to position your product against others in the market, it will help you determine your unique selling point (USP). Find similar websites, minimum of 5, what are they doing well and what are they doing badly? Why have you chosen them? Do you think you can do any better? What will make your website stand out from theirs?

Example:
- If your website is for a major bank, your competitors would be Barclays, Lloyds, HSBC and Nat West's websites.
- You might want to consider other, newer tech like Monzo, Revolut or Klarna.

Acceptance Criteria:
- A paragraph (max 500 words) on who are your key competitors and how what you are offering differs from theirs.

Resources:
- https://miro.com/blog/porters-five-forces/
- https://getlucidity.com/strategy-resources/introduction-to-the-six-forces-model/

Task 7 – Market Research (3 points)

Description:
Market research is critical in understanding who your future user/customer is. Carry out market research on who is most likely to buy your products (inventory wise) and use the website.

Example:
- If you are starting a new bank, focussing on providing, easy, no hassle BNPL (buy now pay later) loans.
- You might want to search around for who are the principal users of those types of loans, their age group, professional class, gender, profession, etc.

Acceptance Criteria:
- A paragraph (max 500 words) on who your target market segment is and the evidence you have that they have a demand for your product.

Resources:
- https://global.com/advertise/insight/case-studies/
- https://global.com/advertise/insight/case-studies/ebay-tiktok/

Task 8 – User Personas (3 points each)

Description:
Who are we designing for? User personas elaborate the answer to this question. Develop maximum 5 user personas for the different users you want to attract to the site. Not only demographics but personality types: i.e., indecisive, time-constrained, rich, just wants to get straight to the point. What data do we have about them?

Example:
- In the banking example, if you realize that a key demographic group are women ages 17-32 who are single and have incomes of < £50k.
- Then start to flesh out an instance of that market segment into an imaginary customer who fits those demographics.
- Empathise with what their concerns and pain points might be and write them down.

Acceptance Criteria:
- A short paragraph for each user persona (max 300 words each).

Resources:
- You can find some examples of user personas here: https://globalradio.atlassian.net/wiki/spaces/PD/pages/3459514401
- https://miro.com/miroverse/persona-map/
- https://miro.com/templates/personas/

Task 9 – User Stories (1 point each)

Description:
User stories divide the development tasks into manageable units of business value. They are intricately connected with the user personas developed above. Write a Maximum of 10 user stories that explain a scenario that your development work will address. Structure them into larger epics.

Example:

- With the banking example, you might have a User story that goes:
    - "As a time-pressed, financially squeezed, young woman, I want to be able to context click on a loan option when browsing my favourite online shopping sites. So that I can carry out my impulse purchases and access credit easily and efficiently with little time to think of how I'm going to pay it back."

Acceptance criteria:

- Develop user stories and use cases for the product. (As a..., I want..., So that...)

Resources:

- https://medium.com/@nic/writing-user-stories-with-gherkin-dda63461b1d2
- You can find a user story and acceptance criteria template here: https://globalradio.atlassian.net/browse/OM-1418

Task 10 – Acceptance Criteria (1 point each)

Description:
Using gherkin language (given, when then...), write acceptance criteria for each of your user stories from task 9 that is suitable for testers.

Example:

- *GIVEN* the user is on the checkout page of a partner website.
- *WHEN* they see the option button to buy now pay later with "My Bank".
- *THEN* they can click on it and get led to a sign in/up page where they can input their mobile number and in one click access the required credit.

Acceptance criteria:

- Each criterion applies to the user story and uses the Gherkin format as explained above.

Resources:

- https://mvwi.co/posts/gherkin-cucumber
- You can find a user story and acceptance criteria template here: https://globalradio.atlassian.net/browse/OM-1418

Task 11 – OKR metrics (5 bonus points)

Description:
Explain how you might measure metrics on the site that would help to evaluate performance against your OKRs.

Example:
- You might install Grafana on your server to monitor unique visitors to the site.
- You might sign up for Google Analytics to gather more information about who and how the site is being used.

Acceptance Criteria:
- A paragraph (max 500 words) on your plan for gathering metrics and how they would help evaluate performance against your OKRs.

Data Tasks

Task 1 – Copy Snowflake table (5 points)

Description:
Create a copy of the table from snowflake with your team's name. The file is downloaded as a .csv file, it has 10 columns and has been loaded to snowflake with the table name:
"INTEGRATION"."PROCESSED"."ONLINE_RETAIL_HACKATHON"

The headers are: ***InvoiceNo***, ***StockCode***, ***Description***, ***Quantity***, ***InvoiceDate***, ***UnitPrice***, ***CustomerID***, ***Country***, ***ItemTotal***, ***UnitCost***.

However, if you wish to use another database, the data can be found in the GitHub repo and read into snowflake/any database of your choice.

Example:
- ***INTEGRATION.PROCESSED.V_online_retail_team10***

Acceptance Criteria:
- A table with all columns from the original table.

Resources:
- Snowflake, SQL

NB: Please do not delete the original file or replace it as all teams will use it!

Task 2 – Business KPIs (1 point each)

Description:
Identify and state KPIs for your business that you want to measure. What are your measures for success/Key Performance Indicators

Example:
- Our company made *xxx* amount above our target on this product, etc

Acceptance Criteria:
- Maximum 5 Key Performance Indicators

Resources:
- https://www.cascade.app/blog/retail-kpis

Task 3 – Implement KPI into the table (3 points per KPI)

Description:
Write a SQL query to calculate adding extra columns as KPIs to your data.

Example:
- A calculation could be (column_name*TARGET as expense)

Acceptance Criteria:
- Additional columns to represent the KPIs that are accurate and can be visualized eventually.

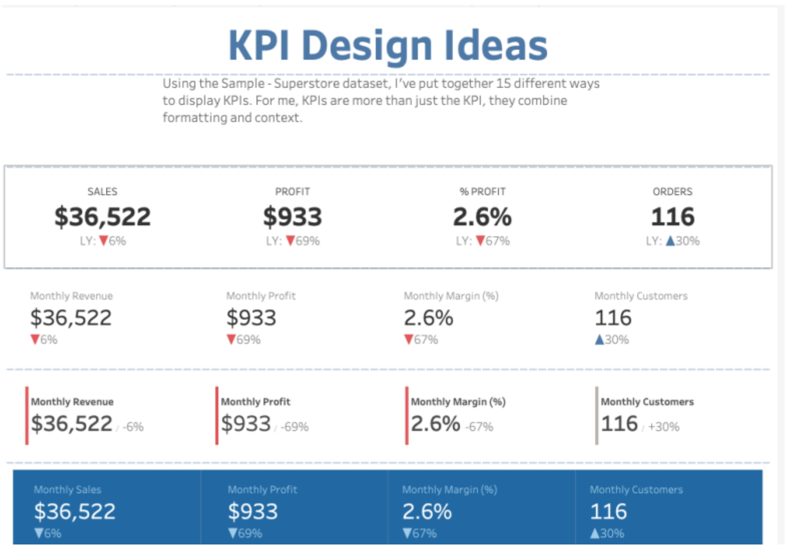
Resources:
- https://www.w3schools.com/sql/sql_count_avg_sum.asp

NB: Work and update data in your table!

Task 4 – Visualize KPIs (5 points per KPI visualisation)

Description:
Visualize the KPIs you have added into the code by connecting to tableau/ any visual tool you are familiar with. (Tableau is recommended)

Example:



KPI Design Ideas

Using the Sample - Superstore dataset, I've put together 15 different ways to display KPIs. For me, KPIs are more than just the KPI, they combine formatting and context.

SALES	PROFIT	% PROFIT	ORDERS
$36,522	$933	2.6%	116
LY: ▼6%	LY: ▼69%	LY: ▼67%	LY: ▲30%

Monthly Revenue	Monthly Profit	Monthly Margin (%)	Monthly Customers
$36,522	$933	2.6%	116
▼6%	▼69%	▼67%	▲30%

Monthly Revenue	Monthly Profit	Monthly Margin (%)	Monthly Customers
$36,522 / -6%	$933 / -69%	2.6% -67%	116 / +30%

Monthly Sales	Monthly Profit	Monthly Margin (%)	Monthly Customers
$36,522	$933	2.6%	116
▼6%	▼69%	▼67%	▲30%

Acceptance Criteria:
- A Working dashboard that visualises all the KPIs you have added in the code and is easily understood.

Resources:
- https://public.tableau.com/app/profile/adam.e.mccann/viz/KPIOptions/AllDaKPIs
- https://blog.coupler.io/tableau-kpi-dashboard/
- https://intellipaat.com/blog/tableau-kpi-dashboard/?US

Demo – (5 points)

Description

Create a demo to display everything you have achieved today! Have a look at the acceptance criteria below to make sure you include everything listed.

Acceptance Criteria

- The demo should be no longer than 5 minutes long.
- Introduce your team and team name.
- Explain to us what product you sell and what kind of market you are trying to reach.
- Present your problem and vision statement.
- Show your current application running and demonstrate all the functions from any completed tasks.
    - E.g., if you only completed up to task 9 on the frontend and backend tasks, then only show features up to task 9.
- You must also demonstrate your extra feature that you have implemented.
    - If you have not completed the extra feature, then an explanation is enough.
    - However, your extra feature will not be voted on unless it has been implemented.
- Show us what KPI's you have visualised!
- Include what's the future of your shop!

Bonus Tasks

Make me look pretty! (voted)

Add some CSS to the site, make it colourful and sleek!
Each website design will be judged at the end of the day after the demos.

- 1st – 20 points
- 2nd – 15 points
- 3rd – 10 points

Create a Logo (voted)

Create a logo for the shop. Each logo design will be judged at the end of the day after the demos.

- 1st – 10 points
- 2nd – 5 points
- 3rd – 3 points

Most creative team name and products (voted)

Think of a team/shop name and what products you will sell. These ideas will then be voted on after the demos.

- 1st – 5 points
- 2nd – 3 points
- 3rd – 1 point

Deploy to a cloud hosting platform (15 points)

- Deploy the site to a cloud hosting platform (Digital Ocean, AWS, VERCEL, GCP, Netlify are some examples).

Simple linear regression to forecast your profits (10 points)

- Using a simple linear regression algorithm, build a model that your business can use to show predict profitability.