# Written task

You have inherited a legacy codebase of an e-commerce application written using outdated frameworks. The code is difficult to maintain and has many performance issues. How would you approach refactoring this codebase to improve its maintainability and performance?

Please outline your strategy and steps in a writing. (We want to understand your thinking)

Answer:

Ought to commence with:

* What features this legacy codebase supports.
  + Categorize these features based on its usage of users/clients along with the severe performance issues.
  + Based on this categorization, we can decide on which features to be taken up first.
  + Can start work on these features, keeping the old legacy codebase still deployed, however asking the users to gradually move to new site with initial set of features, to maintain their trust and gradually moving towards building a new project with same features.
  + This will ensure the old deployed site is still in place, with the new site available, and the users can use both with no disruption in the existing features.
  + To support this, any login pertaining data needs to be migrated first for easy transition of the users.
  + For appropriate planning, the agile framework will come in handy, for incremental feature delivery. This framework will allow
    - Understanding the existing features/requirement,
    - Planning development work,
    - Testing,
    - Deployments,
    - Reviews
    - Feedback
    - that will forms a sprint
* Need to segregate the UI and API such that any technology can be used for both, avoiding any direct dependency on each other.
  + The API url should be configured as plug and play in the UI.
  + This will ensure independent upgradation to each, to avoid again falling into the trap of legacy codebase.
  + API can use technologies like .Net that by itself facilitates easy upgradation
  + For the UI, any technologies from simple HTML/CSS to Vuejs/Nuxt/CSS, can be used.
* This entire process will not be an overnight event, however with an incremental approach neither the developers/QAs nor the users will be overwhelmed working or validating the features.