Design Document

Visual/Interface Designs

Data Requirements

**Data Entities**

**Users**

|  |  |
| --- | --- |
| Users ID | INT (Parent Key) |
| Name | String/Text |
| Email | String |
| Password | String (Hashed) |
| Role | Enum (Admin, User) |
| Address | String |

**Products**

|  |  |
| --- | --- |
| Products ID | Integer (Parent Key) |
| Title | String/Text |
| Description | String |
| Price | String (Hashed) |
| Stock Quantity | Integer |
| Images | Arrays of string (URLs) |

**Orders**

|  |  |
| --- | --- |
| Orders ID | INT (Parent Key) |
| Product ID | Integer |
| Order date | Datetime |
| Order departure date | Datetime |
| Order arrival date | Datetime |
| Quantity | Integer |
| Total Price | Decimal/Float |
| Status | Enum (Pending, Shipped, Delivered, Cancelled) |

In the above tables are the various data requirements needed in certain sections of the prototype. They show what is needed for each section to perform its task well without any issues.

Algorithms Designs

Test Strategy

The test strategy I will be conducting will describe the order in which I plan to test the solution then into listing the types of testing required to be done on the solution.

The solution will be tested starting with its functionality, checking if the function of the prototype is an important aspect to consider, like checking if the prototype after being designed works (does it load up when launched). Functionality testing can also be taking into other parts of the prototype, like the log in and register page, the function giving to those pages is to create user accounts and allow them access to the website with personalised advantages at the end. If that doesn’t work, then the prototype mat as well be useless. Moving on, the prototype after creation must be able to handle a reasonable number of users that use it at simultaneously without resulting in any issues. The numbers can increase gradually over time, but as a starter, the prototype should be able to carry at least 500 users simultaneously. Another form that should be a concern is how secure the prototype will be, we must be able to protect users’ data and information and keep it that way to maintain the trust of future users. Following up with testing how compatible with prototype is with multiple devices. This may be difficult as we have only set thoughts of the prototype being used on devices like monitors and laptops, making it work on smaller devices has not yet been done or strongly considered. After that, we can as well check previous codes while building the prototype to ensure old mistakes won’t affect future results. Also checking how fast and stable the prototype works is of necessity. Sometimes slow response from the prototype may be due to slow internet, but other times the prototype is not just working properly, that is why it is advised that the speed and stability of the prototype is good and does not take more than 3 seconds to load.

**Types of testing needed for the prototype**

Functional Testing: Functional testing is defined as a type of testing that verifies that each function of the software application works in conformance with the requirement and specification. This testing is not concerned with the source code of the application. Each functionality of the software application is tested by providing appropriate test input, expecting the output, and comparing the actual output with the expected output.

Unit Testing: Unit testing is the process of testing the smallest parts of your code, like individual functions or methods, to make sure they work correctly.

Non-Functional Testing: Non-functional Testing is a type of software testing that is performed to verify the non-functional requirements of the application. It verifies whether the behaviour of the system is as per the requirement or not. It tests all the aspects that are not tested in functional testing.

Integration Testing: Integration testing is a software testing technique that focuses on verifying the interactions and data exchange between different components or modules of a software application. The goal of integration testing is to identify any problems or bugs that arise when different components are combined and interact with each other.

Loading Testing: This is a type of performance testing that determines the performance of a system, software product, or software application under real-life-based load conditions.

Usability Testing: Usability Testing is a type of testing, that is done from an end user’s perspective to determine if the system is easily usable.

Security Testing: Security testing is a process intended to detect flaws in the security mechanisms of an information system and as such help enable it to protect data and maintain functionality as intended.

Performance Testing: Performance Testing is the process of analysing the quality and capability of a product. It is a testing method to determine the system’s performance in terms of speed, reliability, and stability under varying workloads.

Compatibility Testing: Compatibility testing is software testing that comes under the non-functional testing category, and it is performed on an application to check its compatibility (running capability) on different platforms/environments.

Acceptance Testing: To find the defects missed during the functional testing phase.

UI Testing: UI testing is the process of validating an application’s visual and interactive elements to ensure they function correctly and provide a smooth user experience.

Regression Testing: Regression testing is performed when changes are made to the existing functionality of the software or if there is a bug fix in the software.

System Testing: System testing is a type of software testing that evaluates the overall functionality and performance of a complete and fully integrated software solution. It tests if the system meets the specified requirements and if it is suitable for delivery to the end-users. This type of testing is performed after the integration testing and before the acceptance testing.