**DOS-Project -Part 1-Bazar**

|  |  |
| --- | --- |
| **Student id** | **Name** |
| 12027669 | Mahmoud maher arafat |
| 12028561 | Ahmad Ashayer |

**Introduction**

The Bazar platform aims to provide a smooth and efficient shopping experience for book lovers, as we offer only four titles,. These books cover topics ranging from mastering distributed operating systems to meeting the challenges of college life, and each offers valuable insights and guidance.

**Technologies Used**

* Web Framework :node js , Npm express
* Data Storage:SQLite databases
* virtual systems:docker
* vscode
* postman

**Program Design**

The program is designed on a microservices approach, and contains two microservices (Catalog, Request) , each of which contains properties, and there is a front-end file, the purpose of which is to filter the user's request and choose the appropriate microservice. The last part is Docker, and Docker in general is a software platform that allows you to quickly build, test, and deploy applications. Docker packages programs into standardized units called containers that contain everything the program needs to run including libraries, system tools, code, and runtime.

Catalog Server: To manage the book catalog, including inventory levels, book prices, and topics, and display them to the user. The data display is divided into two parts: Search and Info.

The order server: undertakes the purchasing process, verifies the availability of the quantity allocated for a specific book from the stock, and seeks help from the catalog server. When the order server seeks help from the catalog server, the book’s inventory is reduced before orders are completed.

**User Operations**

**Catalog**

* get("/search/..."): Users can search for books by subject. If you want, you can do not put anything and it will return to you all the books available in the store. The request is processed by the front end, which queries the catalog server.
* get("/catalog/info/id"): Users request details about a book using its item number, and the catalog server returns the relevant information.

**order**

* get("/purchase/id"): The front end sends purchase orders to the order server, which then verifies the availability of the requested item from stock using the catalog server before proceeding, and the catalog server is required at this point to reduce the quantity. Available in stock.

**Front server**

By managing book information retrieval, catalog searching, and book purchase processing, it acts as an intermediary between the client and back-end services, directing them to the relevant back-end services and providing responses in return. The most important part of error handling is returning an "error status" for any exceptions encountered. This configuration is an example of a microservices architecture, which makes scalability and modular development easier.

**REST**

Each exposes specific REST endpoints to connect to, and commits to HTTP methods (GET to retrieve, POST to create, PUT to replaces ,DELETE to deletes the specified resource). This facilitates a decoupled architecture where services interact through well-defined APIs.

**DOCKER**

The docker file has the responsibility for create three containers for each service which are: Order, Front, and Catalog, the docker file after run it create Linux image (Ubuntu) and three images and also download the packages needed for run this project, and it’s done by only two commands which are:-

**Docker-compose build**:-which is for creating the containers and required packages.

**Docker-compose up**:-which is for run all the containers.