AppCharge BE Home Challenge

- **Please ask me any questions you have - shai@appcharge.com**
- **Business Background:**
  - Appcharge is providing mobile gaming studios with a white-label web store solution to sell their virtual game goods outside of the game directly to their players with reduced fees.
- **Purpose**
  - This challenge examines relevant basic skills for a successful BE developer in Appcharge.
- **Mandatory recruitments**
  - The code works well without any bugs / other crashes
  - Well structure logic and clean code
  - Good explanations about how and why you chose to implement it as you did
  - The code is written in Nest js TS
  - You can choose whatever external database/key-value store you want, but it should support high-scale workloads
  - The service should run as a docker image
  - The system (Service and database/key-value store) should run with docker-compose
- **Less important**
  - Low-level code efficiency
  - Complicated design patterns - keep things simple
- **The needed implementation - create a REST Nest.js web server that exposes the following APIs:**
  - **Login**
    - Request
      - playerId
      - Password
    - Logic
      - Validate that this is a valid player user and generate a unique session id
    - Response
      - A login session id
  - **Offer sets CRUD endpoint with this schema example**

```javascript
{
  "gameId": "33",
  "avlabilty": 15,
  "offerSetName": "bundle medium",
  "offerSetId": "63e8ac11df807b5c13656c69",
  "sku": "222",
```

```
  "priceInCents": "3900",
  "currency": "usd",
  "products": [
    {
      "amount": 500,
      "sku": "002",
      "name": "dice medium"
    },
    {
      "amount": 1500,
      "sku": "005",
      "name": "coins medium"
    }
  ],
}
```

        - **Order**
            - Request
                - ➢ Credit card details
                - ➢ OfferSetId
                - ➢ Session id
            - Logic
                - ➢ Validate that the user loged in
                - ➢ Validate basic credit card details
                - ➢ Validate OfferSet availability and reduce it
                - ➢ Record the order in a DB table
            - Response
                - ➢ Encrypted new order id
                    - Use aes-256-cbc encryption algorithm
- **How to submit**
    - The system should be ready to lunch with the user and offers set data injected into the chosen DB in the setup
    - Upload the code to a Github repo and share it with me - user name - **shaiboujuappcharge**
    - Add a README file that:
        - Explains the system and application design
        - Some explanation about the internal code design
        - Details instructions on how to run the system - use docker-compose
            - ➢ I expect that it would run with a single command