**uDemo**

**Requirements**

uDemo is a simple users management system that manages users information. It provides the ability to register, administer, maintain, generate reports of these users. As part of on boarding and maintenance, users account status can be in any of (Registered, Verified and Deactivated).

Users have roles that gives them the authority to perform assigned functions. They are two types of roles namely; USER and ADMIN.

**Analysis:**

* Applicants register on uDemo.
* Udemo sends application receipt email to user.
* Udemo verifies application details and sends confirmation email to applicant.
* Applicant confirms email.
* Udemo updates applicant to user.
* Confirmed user is on-boarded on uDemo to become active and access on the system
* User off-boarding maintains account status from active to inactive

**Design:**

There are different design consideration depending of the portion of the overall system architecture. Here, we have a layered approach that structures the system into five layers namely; Domain, Service, Messaging, Data Access Object (DAO) and Web API designs.

The system will be event driven and asynchronous, loosely coupled and component based. This decisions are well justified in that it allows design and development to start small but with the foundation to grow with ease and flexibility.

Incorporating layering, events and asynchronous capabilities, the design will be modularly built in a monolith style.

**Domain**: In uDemo there are the following entities; User, Role and Message

User: A person that makes use of the functionalities of uDemo. Requires a role to have access on the system. The structure of a User is described by an ID, first name, last name, email.

Role: Is a security entity used to assign authority a User can perform on the system. The structure of a Role is described by ID, roleName etc

Message: Represent a pattern of information exchange. Different types of information is communicated between users and this is represented as messages. It structures is described as follows: ID, From, To, Subject and Text.

**DAO/Repository:** Data Access Object is a design pattern for communication between an application and a database system. The DAO implementation in uDemo is the Repository. Repository is a further abstraction of the DAO pattern that enhanced the benefits of plain DAO.

Accordingly, the following entities are mapped to their repository object to enable their data access:

User/UserRepository

|  |  |  |
| --- | --- | --- |
| s/n | Enity | Repository |
| 1 | User | UserRepository |
| 2 | Role | RoleRepository |

**Service Layer:** Service or business logic layer is an abstraction between the Web and DAO layer as a separation of concern. The service layer encapsulates complex business logic, process and rules as may be present in the operations of a problem domain.

The service layer abstracts business operations into service interfaces. These service interfaces defines the contracts on how the business operations can be used by clients. The contract spell out the methods signatures and their parameters or the Application Programming Interface (API).

The following are the business operations present in uDemo;

1. Application registration
2. Applicant verification
3. User on-boarding
4. User off-boarding

The above business operations will be implemented in the below service interfaces:

|  |  |  |
| --- | --- | --- |
| s/n | Service | Service Implementation |
| 1 | UDemoService | UDemoServiceImpl |
| 2 | EmailService | EmailServiceImpl |

**Web API:** By design is RESTful, meaning it follows the best practice approach of resource identification, resource representation, endpoint identification and action identification. The web layer is the transport and data exchange layer between the system and its users.

The web layer is responsible for incoming and outgoing requests and the implementation is handled in an MVC pattern.

Resource Identification: From our analysis, we have users resource.

|  |  |  |
| --- | --- | --- |
| HTTP Method | URI | Controller Handler |
| GET | /api/users/{Id} | getUser() |
| GET | /api/users | getAllUsers() |
| POST | /api/users | createUser() |
| PUT | /api/users/{Id} | updateUser() |
| GET | /api/users/email/{email} | verifyUser() |
| DELETE | /api/users/{Id} | deleteUser() |

Resource Representation:

User:

{

"id": 10,

"title": "Phd",

"email": "ken@mailinator.com",

"password": "ordepjkddwuenvhfjj",

"enabled": false,

"firstName": "Nkoko",

"lastName": "Kenneth",

"mobilePhone": "54558836789",

"dateRegistered": "22/03/2021",

"dateVerified": null,

"dateDeactivated": null,

"verified": false,

"status": "REGISTERED",

"roles": []

}

Endpoint Identification:

To expose uDemo endpoint, base URI will be mapped to default network domain (localhost) exposed on port 8085.

Network host: http://localhost:8085/api/users

Action Identification:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| HTTP Method | Resource Endpoint | Input | Success Response | Error Response | Description |
| GET | /users | Body:Empty | Status:200 | Status:500 | Retrieve all users |
| POST | /users | Body: New user | Status:201 | Status:500 | Creates a user |
| PUT | /users | N/A | N/A | Status:400 | Forbidden action |
| DELETE | /users | N/A | N/A | Status:400 | Forbidden action |
| GET | */users/*{userId} | Body:empty | Status:200 | Status: 400 or 500 | Retrieve an existing user |