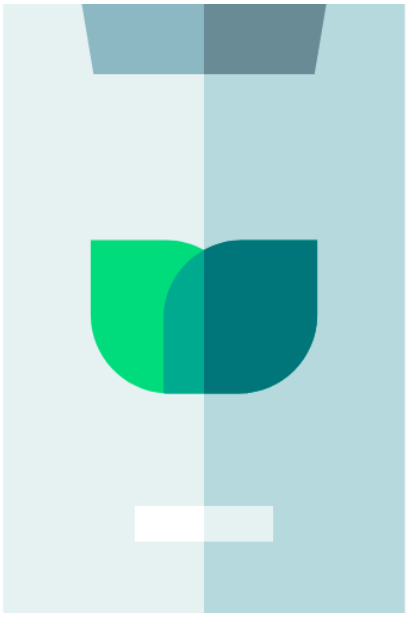
Complete Documentation of Audition Application



- Requirements
- Design
- Development
- Test
- Deployment

Requirement Analysis

An application to serve REST API calls to be dockerized and deployed on cloud.
Requirement 1: Users

1. An Onboarding Admin User
2. Regular User who can register via admin user action

Requirement 2: Database

1. MongoDB for Application
2. Collection for Users and Password
3. Collection for Comments

Requirement 3: Content

1. User Name as unique string in User Collection
2. Password stored as hash in User Collection
3. Comments as string values to be stored in Comment Collection
4. Comment ID as unique string to be stored in Comment Collection
5. User Name as string to refer to User Collection

Requirement 4 : Action

1. Submit Comment differentiated by Comment ID by User Name
2. Delete Comment differentiated by Comment ID by User Name
3. View all comments from Database
4. Check Comment for Palindrome feature differentiated by Comment ID by User Name

Requirement 5: UI

1. UI for Login
2. UI for Register
3. UI for Dashboard
4. UI table on Dashboard to view comments and initiate delete and check palindrome actions
5. UI text area to submit new comments

Requirement 6: Containerization

1. Docker file for creating Docker Image
2. Docker Image for the application
3. Uploaded Docker image on Docker Hub
4. Docker Compose file for Execution

Requirement 7: Deployment

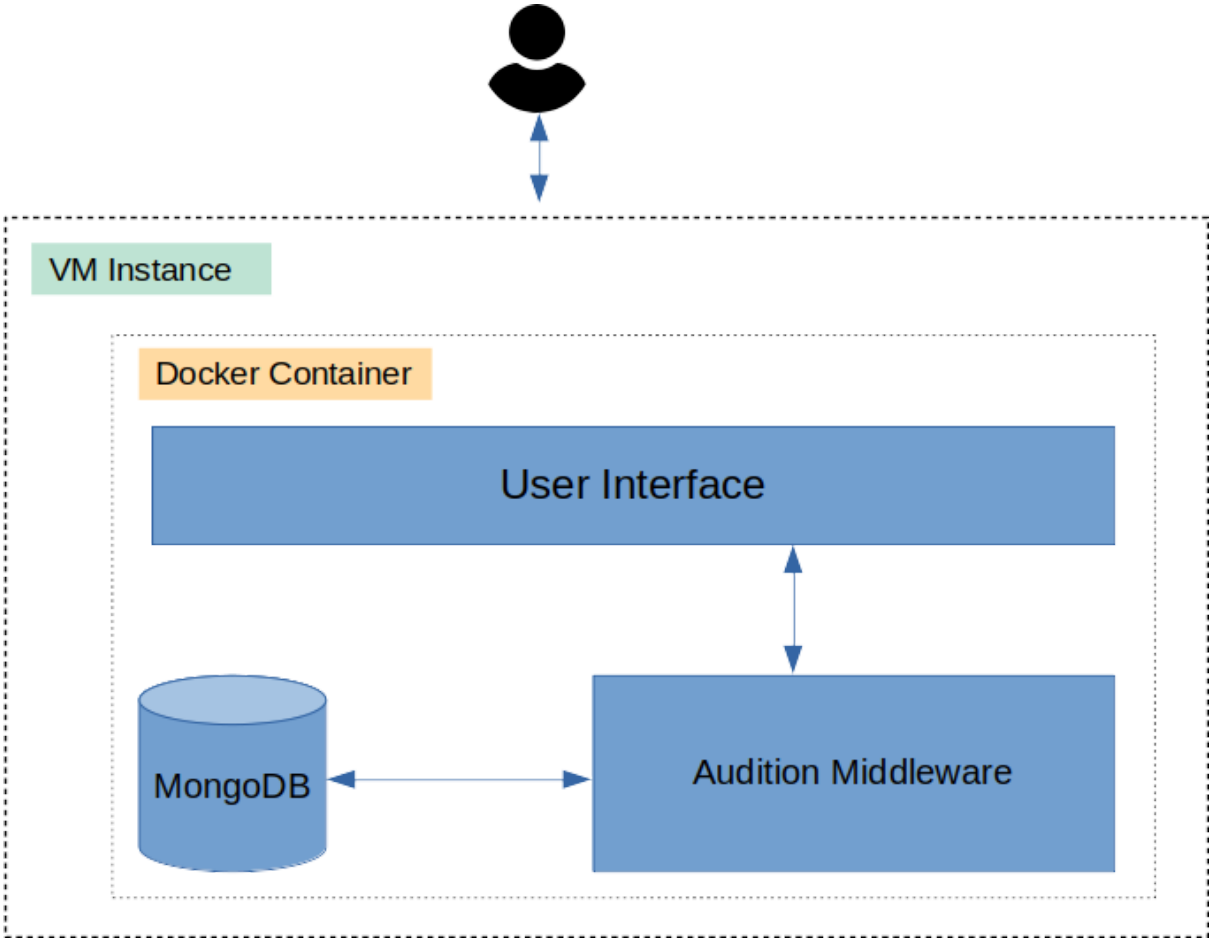
1. VM creation
2. Firewall Configuration
3. Volume Configuration
4. Cloud Native Deployment of Application

Requirement 8 : Monitoring

1. Build capability to (Monitoring/Traceability/metrics)

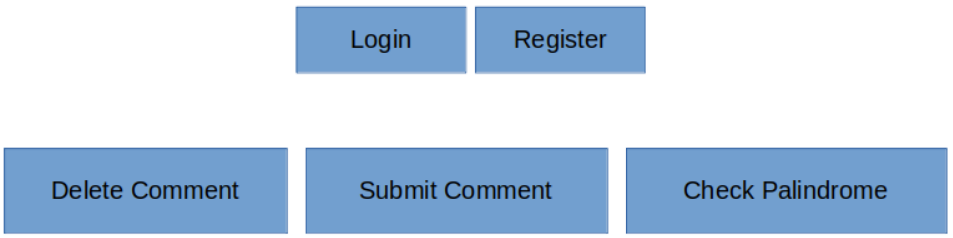
Designing the Product Architecture

A simple design of the product architecture.



Technology Used :

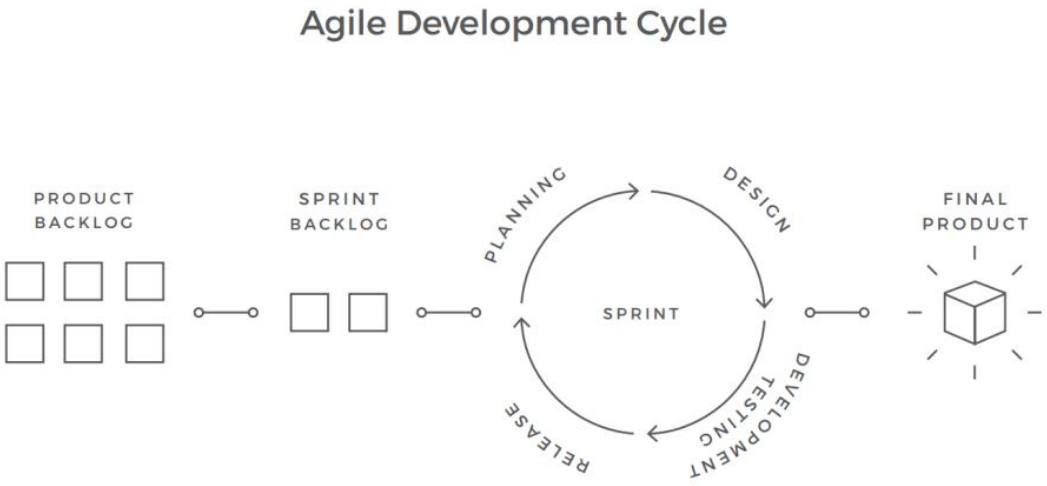
1. MongoDB
2. Golang
3. HTML CSS
4. Javascript
5. Docker
6. Docker-Compose



Functionality Map

Building or Developing the Product

Actual development process follows agile methodology for quick completion and observance.The golang/html code is generated as per design during this stage.



Product Backlog :

1. Create Golang Server
2. Create HTML pages for Login,Register & Dashboard
3. Creation DB and Collections
4. Creation User & Comment Structure
5. Creation DB Handling Methods
6. Creation HTML Template Parse Methods
7. Creation RestAPI Response Method
8. Creation of Handlers for 3 Pages
9. Creation of Handlers for Ajax calls Submit,Delete and Check
10. Completion of Application Storage and REST

Project Backlog can be traced here : https://github.com/users/elkarto91/projects/2

Verification, Validation & Testing the Product

Evaluate by comparison to the set of requirements.

No	Requirement	Status
1	An Onboarding Admin User	Done
2	Regular User who can register via admin user action	Done
3	MongoDB for Application	Done
4	Collection for Users and Password	Done
5	Collection for Comments	Done
6	User Name as unique string in User Collection	Done
7	Password stored as hash in User Collection	Done
8	Comments as string values to be stored in Comment Collection	Done
9	Comment ID as unique string to be stored in Comment Collection	Done
10	User Name as string to refer to User Collection	Done
11	Submit Comment differentiated by Comment ID by User Name	Done
12	Delete Comment differentiated by Comment ID by User Name	Done
13	View all comments from Database	Done
14	Check Comment for Palindrome feature differentiated by Comment ID by User Name	Done
15	UI for Login	Done
16	UI for Register	Done
17	UI for Dashboard	Done
18	UI table on Dashboard to view comments and initiate delete and check palindrome actions	Done
19	UI text area to submit new comments	Done
20	Docker file for creating Docker Image	Done
21	Docker Image for the application	Done
22	Uploaded Docker image on Docker Hub	Done
23	Docker Compose file for Execution	Done

24	VM creation	Done
25	Firewall Configuration	Done
26	Volume Configuration	Done
27	Cloud Native Deployment of Application	Done
28	Build capability to (Monitoring/Traceability/metrics)	NA

Feature Testing :

Local :

No	Feature	URL	Status
1	Register	http://localhost:8080/api/register?adminun=admin&adminpass=password&username=thirduser&userpass=password	Pass
2	Submit Comment	http://localhost:8080/api/submitComment?comment=unagi&commentid=122322&userid=newuser	Pass
3	View All Comment	http://localhost:8080/api/viewAllComment	
4	Delete Comment	http://localhost:8080/api/deleteComment?commentid=2195614	Pass
5	Check Palindrome	http://localhost:8080/api/checkComment?commentid=2195614	Pass

Cloud :

No	Feature	URL	Status
1	Register	http://18.234.129.168:8080/api/register?adminun=admin&adminpass=password&username=thirduser&userpass=password	Pass
2	Submit Comment	http://18.234.129.168:8080/api/submitComment?comment=unagi&commentid=122322&userid=newuser	Pass
3	View All Comment	http://18.234.129.168:8080/api/viewAllComment	
4	Delete Comment	http://18.234.129.168:8080/api/deleteComment?commentid=2195614	Pass
5	Check Palindrome	http://18.234.129.168:8080/api/checkComment?commentid=2195614	Pass

API :

Basic Auth for All :

No	Name	Parameter	Data Type	Test Value
1	Basic Auth Username	Username	string	admin
2	Basic Auth Password	Password	string	adminpw

Register API : http://hostname:port/api/register

No	Name	Parameter	Data Type	Test Value
1	Admin Username	adminun	string	admin
2	Admin Password	adminpass	string	password
3	User Name	username	string	myuser
4	User Password	userpass	string	mypassword

Response :


```
{
   "Status": true,
   "Msg": "success",
   "Extra": {
       "Success": true
   }
}
```

View All Comments API : http://hostname:port/api/viewAllComment

No	Name	Parameter	Data Type	Test Value
1	-	-	-	-No parameters required

Response :


```
{
   "Status": true,
   "Msg": "success",
   "Extra": [
       {
           "username": "myuser",
           "comment_id": "5677278",
           "comment": "level"
       },
       {
           "username": "myuser",
           "comment_id": "7554750",
           "comment": "racecar"
       },
       {
           "username": "myuser",
           "comment_id": "5850608",
           "comment": "LEGEND"
       },
       {
           "username": "myuser",
           "comment_id": "7625192",
           "comment": "motion"
       },
       {
           "username": "myuser",
           "comment_id": "7497121",
           "comment": "Malayalam"
       },

   ]
}
```

Submit Comment API : http://hostname:port/api/submitComment

No	Name	Parameter	Data Type	Test Value
1	Comment	comment	string	racecar
2	Comment ID	commentid	string	5677278
3	Username	userid	string	myuser

Response :


```
{
   "Status": true,
   "Msg": "success",
   "Extra": {
       "Success": true
   }
}
```


Check Palindrome API : http://hostname:port/api/checkComment

No	Name	Parameter	Data Type	Test Value
1	Comment ID	commentid	string	5677278

Response :


```
{
   "Status": true,
   "Msg": "success",
   "Extra": {
       "Success": true,
       "Msg": "Palindrome"
   }
}
```

Delete Comment API : http://hostname:port/api/deleteComment

No	Name	Parameter	Data Type	Test Value
1	Comment ID	commentid	string	5677278

Response :

```
{
   "Status": true,
   "Msg": "success",
   "Extra": true
}
```

Deployment in the Market and Maintenance

The application will be deployed as per requirements as mentioned below.

Deployment Steps :

1. Creation of Docker File
2. Creation of Image
3. Upload to Docker Hub Store
4. Creation of Docker Compose File
5. Creation of VM
6. Firewall Rule Change
7. Execution of Application

No	Docker	Commands
1	Create Docker ID	Website
2	Create New Repository	Website
3	Docker Login from CMD	docker login
4	Docker Build Image	docker build -t dockeraccountname/repo/imagetag .
5	Docker Push Image to DCR	docker push dockeraccountname/repo/imagetag

VM :

No	VM Setup	Commands
1	Update EC2 Image	sudo apt-get update
2	Install Docker in EC2	sudo apt install docker
3	Pull docker Release	sudo curl -L https://github.com/docker/compose/releases/download/1.21.0/docker-compose-`uname -s`-`uname -m` | sudo tee /usr/local/bin/docker-compose > /dev/null
4	Append Permission	sudo chmod +x /usr/local/bin/docker-compose
5	Link	sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
6	Start Service	sudo service docker start
7	Start Application	sudo docker-compose up

Next:

User Interface :

The user interface is basic and minimal , it can be made better with time.

Error Handling :

Graceful handling of errors can be written into code, for now it just throws ‘tantrums’. But logging is done pretty well, so it can be checked to trace.

Monitoring :

Will take some time, with the current workload it's quite difficult to put the hours in. Can be done in time.