



ALUMNI DOCKER API TUTORIAL

Kashish Lathidadia

Alumni Docker API

- This tutorial is a step-by-step guide to help you install and run the Alumni Docker API.
- **What is Alumni Docker API**
- It is a REST API application built using Docker, Node.js and MongoDB
- The application runs on <http://localhost:8080>
- The application lets you create new users, login, access files on a private AWS S3, etc.
- You can interact with the application HTTP requests to various end-points e.g.
 - /api/test/all – Public Content
 - /api/auth/signup – Register new user
 - /api/auth/signin – Login user (Responds with JWT token)
 - /api/test/user (+ JWT token) – User Content page
 - /api/user/files (+ JWT token) – List all files from AWS s3 server

0. -Prerequisites

- This tutorial assumes you know how to run commands in a local terminal.
- You have Docker installed on your machine. (If not, please see instructions below).
- The example commands in this tutorial are run on a Bash terminal shell. You'll need to modify the commands according to your own terminal (CMD, Windows PowerShell, etc.)
- **Docker**
- What is Docker?
 - Docker is an open source containerization platform. It enables developers to package applications into containers—standardized executable components combining application source code with the operating system (OS) libraries and dependencies required to run that code in any environment.
- How to install Docker?
 - Please follow instructions on <https://docs.docker.com/get-docker/>
 - Once you've installed Docker, type the following (without the quotes " ") in your local terminal to make sure everything is ok
 - "docker -v"
 - You should see an output like
 - Docker version 20.10.....

• MobaXterm Personal Edition v21.0 •
(X server, SSH client and network tools)

- › Your computer drives are accessible through the `/drives` path
- › Your DISPLAY is set to `192.168.2.70:0.0`
- › When using `SSH`, your remote DISPLAY is automatically forwarded
- › Each command status is specified by a special symbol (✓ or ✗)

• Important:

This is MobaXterm Personal Edition. The Professional edition allows you to customize MobaXterm for your company: you can add your own logo, your parameters, your welcome message and generate either an MSI installation package or a portable executable. We can also modify MobaXterm or develop the plugins you need.
For more information: <https://mobaxterm.mobatek.net/download.html>

06/04/2022 22:56.10 /home/mobaxterm > docker -v
Docker version 20.10.12, build e91ed57

06/04/2022 22:56.17 /home/mobaxterm > █

1. –Navigate to Docker Image

- Download the 'AlumniDocker' image file from the assignment submission drop-box.
- Using the command below, Navigate to the folder where you saved the image file (I saved mine inside D:\ drive AlumniDockerAPI folder")
 - " cd 'D:\AlumniDockerAPI' "
 - You can use the "ls" command to list the contents of the folder to ensure you can see the file.

```
• MobaXterm Personal Edition v21.0 •
(X server, SSH client and network tools)

> Your computer drives are accessible through the /drives path
> Your DISPLAY is set to 192.168.2.70:0.0
> When using SSH, your remote DISPLAY is automatically forwarded
> Each command status is specified by a special symbol (✓ or ✗)

• Important:
This is MobaXterm Personal Edition. The Professional edition
allows you to customize MobaXterm for your company: you can add
your own logo, your parameters, your welcome message and generate
either an MSI installation package or a portable executable.
We can also modify MobaXterm or develop the plugins you need.
For more information: https://mobaxterm.mobatek.net/download.html

06/04/2022 22:56:10 /home/mobaxterm docker -v
Docker version 20.10.12, build e91ed57

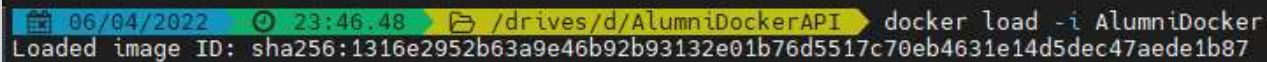
06/04/2022 22:56:17 /home/mobaxterm cd "D:\AlumniDockerAPI"

06/04/2022 23:37:20 /drives/d/AlumniDockerAPI ls
AlumniDocker

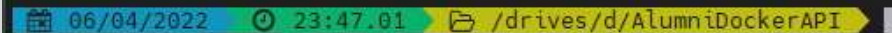
06/04/2022 23:41:10 /drives/d/AlumniDockerAPI
```

2. –Load the Image into Docker

- Run the following command to load the image into docker
 - “docker load -i ./AlumniDocker”



```
06/04/2022 23:46.48 /drives/d/AlumniDockerAPI docker load -i AlumniDocker
Loaded image ID: sha256:1316e2952b63a9e46b92b93132e01b76d5517c70eb4631e14d5dec47aede1b87
```



```
06/04/2022 23:47.01 /drives/d/AlumniDockerAPI
```

3. –The docker run command

- The *docker run command* first creates a writeable container layer over the specified image, and then starts it using the specified command.
- “`docker run -p 49160:8080 -d <your username>/`”
- Running your image with `-d` runs the container in detached mode, leaving the container running in the background. The `-p` flag redirects a public port to a private port inside the container.
- Once the container is running, you can use the “`docker ps`” command to get a list of running containers.
- From the list, get the container ID and use it to get the logs “`docker logs <container id>`”

Unit Test Plan and Result

Project name:	Alumni API		
Module (e.g., class) name:	Exit Email, Sign-up, Sign-in, Get File		
Designed by:	Kashish Lathidadia	Design date:	April 08, 2022
Executed by:	Marlon Robancho	Execution date:	April 10, 2022

Test case #	Title	Test data (and steps)	Expected Result	Actual Result	Success (Y/N)
1	Create Exit Email	Please see postman screenshots below	Exit email created successfully	Exit email created successfully	Y
2	List all exit emails	Please see postman screenshots below	Returns all exit emails	All exit emails have been returned	Y
3	Signup	Please see postman screenshots below	Email successfully signed up	Email successfully signed up	Y
4	Signup – already registered	Please see postman screenshots below	Returns message, Failed! Username is already in use!	Returned message, Failed! Username is already in use!	Y
5	Sign-in	Please see postman screenshots below	Signed in successfully	Signed in successfully	Y
6	Sign-in - invalid password	Please see postman screenshots below	Returns message, Invalid Password!	Returned message, Invalid Password!	Y
7	Get File List	Please see postman screenshots below	Returns all file list for the specific user	All file list for the specific user has been returned	Y
8	Get File	Please see postman screenshots below	Returns specific file for the specific user	Specific file for the specific user has been returned	Y

Create Exit Email

+

☰

...

Alumni API

> auth

▼ user

GET Get File

GET Test all

GET Test user

GET Test moderator

GET Test admin

GET Get File List

GET Get File

▼ admin

GET List all exit emails

POST Create Exit Email

GET Update Exit Email

Alumni API / admin / Create Exit Email

POST localhost:8080/api/admin/exitEmail/create

Params Authorization Headers (9) Body ● Pre-request Script Tests ● Settings

1 pm.test("Body matches string", function () {

2 pm.expect(pm.response.text()).to.include("mrobanch04X@gmail.com");

3 });

Body Cookies Headers (10) Test Results (1/1) Status: 200 OK

Pretty Raw Preview Visualize JSON

1 {

2 "id": "62538c9ef883c6341c282cd5",

3 "email": "mrobanch04X@gmail.com",

4 "vv": 0

5 }

+

☰

...

Alumni API

> auth

▼ user

GET Get File

GET Test all

GET Test user

GET Test moderator

GET Test admin

GET Get File List

GET Get File

▼ admin

GET List all exit emails

POST Create Exit Email

GET Update Exit Email

Alumni API / admin / Create Exit Email

POST localhost:8080/api/admin/exitEmail/create

Params Authorization Headers (9) Body ● Pre-request Script Tests ● Settings

1 pm.test("Body matches string", function () {

2 pm.expect(pm.response.text()).to.include("mrobanch04X@gmail.com");

3 });

Body Cookies Headers (10) Test Results (1/1) Status: 200 OK Time: 1

All Passed Skipped Failed

PASS Body matches string

List all exit emails

+

☰

...

Alumni API

>

auth

>

user

GET Get File

GET Test all

GET Test user

GET Test moderator

GET Test admin

GET Get File List

GET Get File

admin

GET List all exit emails

POST Create Exit Email

GET Update Exit Email

Alumni API / admin / List all exit emails

GET

localhost:8080/api/admin/exitEmail/findAll

Params

Authorization

Headers (7)

Body

Pre-request Script

Tests

Settings

Headers < 6 hidden

	KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/>	x-access-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjYyNT...	

Body

Cookies

Headers (10)

Test Results (1/1)

⊕

Status: 200 OK

Pretty

Raw

Preview

Visualize

JSON

⌵

⌵

```
1  [
2    {
3      "_id": "625381c09ff83d49a4f2a497",
4      "email": "root@abc.com",
5      "_v": 0
6    },
7  ]
```

+

☰

...

Alumni API

>

auth

>

user

GET Get File

GET Test all

GET Test user

GET Test moderator

GET Test admin

GET Get File List

GET Get File

admin

GET List all exit emails

POST Create Exit Email

GET Update Exit Email

Alumni API / admin / List all exit emails

GET

localhost:8080/api/admin/exitEmail/findAll

Params

Authorization

Headers (7)

Body

Pre-request Script

Tests

Settings

```
1  pm.test("Body matches string", function () {
2    pm.expect(pm.response.text()).to.include("mrobanchoiX@gmail.com");
3  });
```

Body

Cookies

Headers (10)

Test Results (1/1)

⊕

Status: 200 OK

All

Passed

Skipped

Failed

PASS

Body matches string

Signup

Alumni API / auth / Signup

POST

localhost:8080/api/auth/signup

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "username": "user_miobancho3X",
3   "email": "miobancho3X@gmail.com",
4   "password": "123456"
5 }
```

Body Cookies Headers (10) Test Results (1/1) Status: 200 OK

Pretty Raw Preview Visualize JSON

```
1 {
2   "message": "User was registered successfully!"
3 }
```

Alumni API / auth / Signup

POST

localhost:8080/api/auth/signup

Params Authorization Headers (8) Body Pre-request Script Tests Settings

```
1 pm.test("Body matches string", function () {
2   pm.expect(pm.response.text()).to.include("User was registered successfully!");
3 });
```

Body Cookies Headers (10) Test Results (1/1) Status: 200 OK

All Passed Skipped Failed

PASS

Body matches string

Signup – already registered

Alumni API / auth / Signup - already registered

POST localhost:8080/api/auth/signup

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "username": "user_mrobanchoiX",
3   "email": "mrobanchoiX@gmail.com",
4   "password": "123456"
5 }
```

Body Cookies Headers (10) Test Results (1/1) Status: 400 Bad Request

Pretty Raw Preview Visualize JSON

```
1 {
2   "message": "Failed! Username is already in use!"
3 }
```

Alumni API / auth / Signup - already registered

POST localhost:8080/api/auth/signup

Params Authorization Headers (8) Body Pre-request Script Tests Settings

```
1 pm.test("Body matches string", function () {
2   pm.expect(pm.response.text()).to.include("Failed! Username is already in use!");
3 });
```

Body Cookies Headers (10) Test Results (1/1) Status: 400 Bad Request

All Passed Skipped Failed

PASS Body matches string

Sign-in

[illegible]

Alumni API / auth / Signin

POST

localhost:8080/api/auth/signin

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Settings

```

1 pm.test("Body matches string", function () {
2   pm.expect(pm.response.text()).to.include("roles");
3 });

```

Body

Cookies

Headers (10)

Test Results (1/1)

Status: 200 OK

All

Passed

Skipped

Failed

PASS

Body matches string

Sign-in - invalid password

This screenshot shows a Postman interface for a REST client. The left sidebar displays a collection named 'Alumni API' with a folder 'auth' containing several endpoints. The 'POST Signin - invalid password' endpoint is selected. The main panel shows the request configuration: a POST request to 'localhost:8080/api/auth/signin' with a JSON body containing 'username': 'user_mrobancholx' and 'password': '789@'. The 'Body' tab is active, showing the raw JSON. The status bar at the bottom indicates a '401 Unauthorized' response.

Alumni API / auth / **Signin - invalid password**

POST localhost:8080/api/auth/signin

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "username": "user_mrobancholx",
3   "password": "789@"
4 }
```

Body Cookies Headers (10) Test Results (1/1) Status: 401 Unauthorized

Pretty Raw Preview Visualize JSON

```
1 {
2   "accessToken": null,
3   "message": "Invalid Password!"
4 }
```

This screenshot shows the same Postman interface as the first, but with the 'Tests' tab selected. A test script is written to verify that the response body contains the message 'Invalid Password!'. The test results show a 'PASS' status, indicating the test was successful. The status bar still shows '401 Unauthorized'.

Alumni API / auth / **Signin - invalid password**

POST localhost:8080/api/auth/signin

Params Authorization Headers (9) Body Pre-request Script **Tests** Settings

```
1 pm.test("Body matches string", function () {
2   pm.expect(pm.response.text()).to.include("Invalid Password!");
3 });
```

Body Cookies Headers (10) Test Results (1/1) Status: 401 Unauthorized 1

All Passed Skipped Failed

PASS Body matches string

Get File List

+

☰

...

Alumni API

auth

POST Signin

POST Signin - invalid password

POST Signup

POST Signup - already registered

user

GET Get File

GET Test all

GET Test user

GET Test moderator

GET Test admin

GET Get File List

GET Get File

admin

GET List all exit emails

POST Create Exit Email

GET Update Exit Email

Alumni API / user / **Get File List**

GET localhost:8080/api/user/files

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Headers 6 hidden

	KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/>	x-access-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjYyNT...	
	Key	Value	Description

Body Cookies Headers (10) Test Results (1/1) Status: 200 OK

Pretty Raw Preview Visualize JSON

```
1  [
2    {
3      "Key": "root@abc.com/",
4      "LastModified": "2022-04-10T02:05:18.000Z",
5      "ETag": "\"d41d8cd98f00b204e9800998ecf8427e\"",
6      "ChecksumAlgorithm": [],
7      "Size": 0,
8      "StorageClass": "STANDARD"
9    },
10   ]
```

+

☰

...

Alumni API

auth

POST Signin

POST Signin - invalid password

POST Signup

POST Signup - already registered

user

GET Get File

GET Test all

GET Test user

GET Test moderator

GET Test admin

GET Get File List

GET Get File

admin

GET List all exit emails

POST Create Exit Email

GET Update Exit Email

Alumni API / user / **Get File List**

GET localhost:8080/api/user/files

Params Authorization Headers (7) Body Pre-request Script Tests Settings

1 pm.test("Body matches string", function () {
2 pm.expect(pm.response.text()).to.include("root@abc.com/");
3 });

Body Cookies Headers (10) Test Results (1/1) Status: 200 OK

All Passed Skipped Failed

PASS Body matches string

Get File

Alumni API

auth

POST Signin

POST Signin - invalid password

POST Signup

POST Signup - already registered

user

GET Get File

GET Test all

GET Test user

GET Test moderator

GET Test admin

GET Get File List

GET Get File

admin

GET List all exit emails

POST Create Exit Email

GET Update Exit Email

Alumni API / user / Get File

GETlocalhost:8080/api/user/file?key=root@abc.com/magic-bg.png

ParamsAuthorizationHeaders (7)BodyPre-request ScriptTestsSettings

Headers6 hidden

KEY	VALUE	DESCR
<input checked="" type="checkbox"/> x-access-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjYyNT...	
Key	Value	Descri

BodyCookiesHeaders (10)Test Results (1/1)Status: 200 OK

PrettyRawPreviewVisualizeJSON

```
1 {
2   "AcceptRanges": "bytes",
3   "LastModified": "2022-04-10T02:06:56.000Z",
4   "ContentLength": 292832,
5   "ETag": "\"7ce885d798a87ca207e569d97adb420\"",
6   "ContentType": "image/png",
7   "Metadata": {},
8   "Body": {
9     "type": "Buffer",
10    "data": [
```

Alumni API

auth

POST Signin

POST Signin - invalid password

POST Signup

POST Signup - already registered

user

GET Get File

GET Test all

GET Test user

GET Test moderator

GET Test admin

GET Get File List

GET Get File

admin

GET List all exit emails

POST Create Exit Email

GET Update Exit Email

Alumni API / user / Get File

GETlocalhost:8080/api/user/file?key=root@abc.com/magic-bg.png

ParamsAuthorizationHeaders (7)BodyPre-request ScriptTestsSettings

1 pm.test("Body matches string", function () {
2 pm.expect(pm.response.text()).to.include("Body");
3 });

BodyCookiesHeaders (10)Test Results (1/1)Status: 200 OK

AllPassedSkippedFailed

PASSBody matches string

Project Id Sheet with Deployment Guide

Project Name: “Alumni API”

Team name and logo:

Name: Samurai

Logo:



Team Members:

Name	Team contact information (e-mail)
Gundu Rakesh Kumar	rgundu0928@conestogac.on.ca
Stromeliuk Kyrlo	kstromeliuk0872@conestogac.on.ca
Lathidadia Kashish	klathidadia1887@conestogac.on.ca
Robancho Marlon	mrobancho3958@conestogac.on.ca

About Alumni API :

The application is for Ex-Employees of a company to get details and documents post their exit. This application helps companies for not giving full access to ex-employees except specified requirements.

Main target:

Decrease HR managers time spent finding, scanning and sending sensitive documents of the former employees because these employees will have access to their profiles and documents even after they leave the company. This will reduce repetitive work of HR managers, they can spend on other issues.

Quick Start

To start the application there are a few simple steps:

1. HR manager collects emails from the future users and sends them to the administrator so they could add to the database these emails as verified;
2. With the link to emails in database HR managers upload files, related to this profile;
3. To get access, users input their emails, given in step 1, new login, password and confirm registration via email account. Input their login and password and start using Alumni API;
4. To test, open in your browser using the address: <http://localhost:8080/> .

Deployment Instructions:

1. Install MongoDB, Docker and Visual Studio Code applications on your computer;
2. Install NODE using “**npm** install” command, and start with “**npm** start” command;

How to Create the Database:

As soon as MongoDB is installed on your computer the application will connect and run automatically.

How to Install and Configure the Web Site

The crucial phase is installing Docker on the server. The Dockerfile manages the environment.

Waiver

The undersigned permit Conestoga College Information Technology faculty to use the finished application and other deliverables for classroom demonstrations and the marketing of Conestoga College Information Technology programs. Conestoga College Information Technology faculty agree to use the source code only for marking purposes, and not to disclose or distribute the source code to any third party.

Full name:

Signature:

Date:

Gundu Rakesh Kumar



15 March 2022

Stromeliuk Kyrylo



15 March 2022

Lathidadia Kashish



15 March 2022

Robancho Marlon



15 March 2022

Team Status Report

TEAM NUMBER: four TEAM NAME: SAMURAI.

DATE: 2022 04 15
YYYY MM DD

TEAM MEMBER NAME:

Name	Team contact information (e-mail)
Gundu Rakesh Kumar	rgundu0928@conestogac.on.ca
Stromeliuk Kyrylo	kstromeliuk0872@conestogac.on.ca
Lathidadia Kashish	klathidadia1887@conestogac.on.ca
Robancho Marlon	mrobancho3958@conestogac.on.ca

Location	Team Meeting / Team, Activity	Present (list of initials)	Date YYYY.MM.DD	Duration (nearest .25 hr)
MST	Programming	Lathidadia Kashish	2022.04.15	33.00h
MST	Documents preparation	Robancho Marlon	2022.04.15	30.50h
MST	Project management	Gundu Rakesh Kumar	2022.04.15	30.00h
MST	Documents preparation	Stromeliuk Kyrylo	2022.04.15	31.00h

Team Member Status Report

TEAM NUMBER : four

TEAM NAME : SAMURAI

TEAM MEMBER NAME: Kyrylo Stromeliuk

DATE: 2022 04 15
YYYY MM DD

Category	Activity	Hours (.25 increments)
DOD Diagramming and Documenting Overall Design	Design level class diagram.	2
TDOC Technical DOCumentation	Sequence diagrams, use case diagrams.	12.5
TDOC Technical DOCumentation	WEB research and rationale	4
ENTPR ENTity class design and PRogramming	Creating statechart, entity relationship diagrams.	6
RPT Report Development	Security assessment.	6.5

Team Member Status Report

TEAM NUMBER : four

TEAM NAME : SAMURAI

TEAM MEMBER NAME: Marlon Robancho

DATE: 2022 04 15
 YYYY MM DD

Category	Activity	Hours (.25 increments)
PM Project Management	Project Plan	2
PM Project Management	Agile backlog	3.5
TDOC Technical DOCumentation	Navigation diagram	3
GUID GUI Design	Data dictionary with field level definitions	2.5
ITEST Interactive TESTing	Unit test plan and result	8
TPR Technical / lower-level PRogramming	Help systems and tutorials	7.5
UTEST Unit and integration TESTing	Unit test system plans and results	4

Team Member Status Report

TEAM NUMBER : four

TEAM NAME : SAMURAI

TEAM MEMBER NAME: Gundu Rakesh Kumar

DATE: 2022 04 15
YYYY MM DD

Category	Activity	Hours (.25 increments)
PM Project Management	Team charter	8
PM Project Management	Project charter	6.5
PM Project Management	Agile iteration plan	6
PM Project Management	Agile backlog template	4
PM Project Management	Agile release summary	2.5
RPT Report Development	Team status report	3

Team Member Status Report

TEAM NUMBER : four

TEAM NAME : SAMURAI

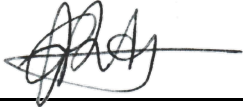
TEAM_MEMBER_NAME: Lathidadia Kashish

DATE: 2022 04 15
 YYYY MM DD

Category	Activity	Hours (.25 increments)
IWC Interacting With Clients	Client sign-off, contract	4
WPD Web Page Design	User interface prototypes	2
WEBPR WEB PRogramming	WEB development	22
DBD DataBase Design	MongoDB database	2

Client Sign-Off

I, OzzieShahmadar, acknowledge that I have received all the stated deliverables for Alumni Dashboard and that they are at an acceptable quality to me.

A handwritten signature in black ink, appearing to be 'OzzieShahmadar', written over a horizontal line.

Date: 2022-04-17

AGILE BACKLOG

Conestoga College				
Team: Kashish Lathidadia, Rakesh kumar Gundu, Marlon Robancho, Kyrro Stromeliuk				
Agile Product Feature List				
#	Description	PD est.	By	Note: PD = Person day = 4 hours; start feature # from 101 (reserve 1-100 for other requirements)
High Priority				Note: Only high priority items are prioritized, assigned to iteration, and estimated
Iteration 1				
6	Development Environment Setup		Kashish	
Iteration 2				
1	Node API - CRUD	7	Kashish	
3	React - login	6	Marlon	
4	DB connection	3	Kashish	
5	Model setup	5	Kashish	
Iteration 3				
7	React - Ability to Download documents	6	Kashish	
2	Email Verification	3	Kashish	
6	Contact HR	3	Kashish	
8	Connect Remote storage (AWS?, private sile server?)	7	Kashish	
9	API Security	2	Kashish	
Medium Priority				Note: Medium and Low are prioritized but not yet assigned to iteration, nor estimated
301	UI Beutification		Kashish	
302	Support external login (Google, FB)		Kashish	
			Kashish	
			Kashish	
			Kashish	
Low Priority				
102	Need a log for items that have been downloaded		Kashish	
601	Dockerize		Kashish	

Agile Release Summary

1	Task Name	Duration	Expected Start Date	Expected End Date	Actual End Date	Resource Name
2	Inception	20				
3	Client Letter		22/01/2022	04/02/2022	11/02/2022	
4	Team Charter (Letter to Client, IP)		22/01/2022	04/02/2022	11/02/2022	
5	Project Charter (Stakeholders Description)		22/01/2022	04/02/2022	11/02/2022	
6	IP Contract Between Client and Team Members		22/01/2022	04/02/2022	11/02/2022	
7	IP Contract Between Team Members		22/01/2022	04/02/2022	11/02/2022	
8	Project Plan		22/01/2022	04/02/2022	11/02/2022	
9	Web Search and Rationale Report		22/01/2022	04/02/2022	11/02/2022	
10	Team Status Report		22/01/2022	04/02/2022	11/02/2022	
11	Client Sign-off document		22/01/2022	04/02/2022	11/02/2022	
12	Project Summation Slides	22/01/2022	04/02/2022	11/02/2022		
14	Iteration 1 - Analysis					
15	Use Case Diagrams	14	05/02/2022	18/02/2022	18/02/2022	
16	Use Case Descriptions (Limit 5 significant)		05/02/2022	18/02/2022	18/02/2022	
17	Analysis level (Domain) Class Diagram		05/02/2022	18/02/2022	18/02/2022	
18	System Sequence Diagrams (Limit 3 significant)		05/02/2022	18/02/2022	18/02/2022	
19	Statechart Diagram(s)		05/02/2022	18/02/2022	18/02/2022	
20	Installation of the software		05/02/2022	18/02/2022	18/02/2022	
21	Home		05/02/2022	18/02/2022	18/02/2022	
22	User Login page		05/02/2022	18/02/2022	18/02/2022	
23	Client Sign-off Document		05/02/2022	18/02/2022	18/02/2022	
24	Team Status Report (Weekly)		05/02/2022	18/02/2022	18/02/2022	
25	Design level Sequence Diagrams (Limit 3 Significant)		05/02/2022	18/02/2022	18/02/2022	
26	User Interface Prototype		05/02/2022	18/02/2022	18/02/2022	
27	Client Sign-off Document		05/02/2022	18/02/2022	18/02/2022	
28	Unit Tests Plan and Results		05/02/2022	18/02/2022	18/02/2022	
29	Agile Iteration Summary		05/02/2022	18/02/2022	18/02/2022	

30						
31	Iteration 2 - Design					
32	Use Case Diagrams	26	14/02/2022	11/03/2022		
33	Use Case Descriptions (Limit 5 significant)		14/02/2022	11/03/2022		
34	Analysis level (Domain) Class Diagram		14/02/2022	11/03/2022		
35	System Sequence Diagrams (Limit 3 significant)		14/02/2022	11/03/2022		
36	Statechart Diagram(s)		14/02/2022	11/03/2022		
37	Client Sign-off Document		14/02/2022	11/03/2022		
38	Team Status Report (Weekly)		14/02/2022	11/03/2022		
39	HR Login page		14/02/2022	11/03/2022		
40	HR View Inventory page		14/02/2022	11/03/2022		
41	Data Base SetUp		14/02/2022	11/03/2022		
42	Reporting View creation for HR and user		14/02/2022	11/03/2022		
43	Design Class Diagram		14/02/2022	11/03/2022		
44	User Interface Prototype		14/02/2022	11/03/2022		
45	Navigation Diagram(s)		14/02/2022	11/03/2022		
46	Client Sign-off Document		14/02/2022	11/03/2022		
47	Agile Iteration Summary		14/02/2022	11/03/2022		
48	Working Code Demonstration		14/02/2022	11/03/2022		

49						
50	Iteration 3 - Construction and Web development					
51	Design level Sequence Diagrams (Limit 3 Significant)		21/02/2022	01/04/2022		
52	Navigation Diagram(s)		21/02/2022	01/04/2022		
53	Entity Relationship Diagram (ERD)		21/02/2022	01/04/2022		
54	Client Sign-off Document		21/02/2022	01/04/2022		
55	Data Dictionary with field level definitions or Glossary		21/02/2022	01/04/2022		
56	Working Code Demonstration		21/02/2022	01/04/2022		
57	Agile Iteration Summary		21/02/2022	01/04/2022		
58						
59	Transition		02/04/2022	07/04/2022		
60	User and System test plan(s) and results		02/04/2022	07/04/2022		
61	Help system (User doc) and (if created) tutorials		02/04/2022	07/04/2022		
62	Project ID Sheet & Deployment Guide		02/04/2022	07/04/2022		
63	Team Status Report		02/04/2022	07/04/2022		
64	Client Sign-off Document		02/04/2022	07/04/2022		
65	Source Code & Deployment to Client		02/04/2022	07/04/2022		
66	Faculty Advisor Project Submission (check the Project Submission Specification)		02/04/2022	07/04/2022		
67	Client Project Submission (check the Project Submission Specification)		02/04/2022	07/04/2022		
68	Agile Backlog		02/04/2022	07/04/2022		
69	Agile Release Summary		02/04/2022	07/04/2022		
70						
71	Demo and Judging		08/04/2022	08/04/2022		
72	Attending Demo Day		08/04/2022	08/04/2022		
73	Application Brochure		08/04/2022	08/04/2022		
74	Business Cards		08/04/2022	08/04/2022		
75	Resumes		08/04/2022	08/04/2022		
76	Showroom Slides		08/04/2022	08/04/2022		
77	Booth Display		08/04/2022	08/04/2022		