# FLOWER VALLEY HOSTED ON IBM CLOUD

**Project Report Submitted to National Skill Training Institute for Women, Trivandrum in the Partial Fulfillment and Requirements for IBM Advanced Diploma(Vocational) in IT, Networking and Cloud**

**By**

**Alfiya Fazil**

**ADIT/TVM/19/001**

**ADIT (2019-2021)**

**Under the supervision of**

**Mr. Poovaragavan Velumani (Master Trainer, Edunet Foundation)**

**GOVERNMENT OF INDIA**

**MINISTRY OF SKILL DEVELOPMENT & ENTREPRENEURSHIP**

**DIRECTORATE GENERAL OF TRAINING**

**July 2021**

# CONTENTS

**ABSTRACT**

**1. INTRODUCTION**

1.1. OBJECTIVE

1.2. PROJECT DESCRIPTION

1.3. SCOPE OF WORK

**2. SYSTEM ANALYSIS**

2.1. PROPOSED SYSTEM

**3. SOFTWARE DEVELOPMENT ENVIRONMENT**

**4. SYSTEM DESIGN**

4.1. DATA MODEL DIAGRAM

4.2. USE CASE DIAGRAM

4.3. FLOW CHART

**5. SYSTEM REQUIREMENTS**

5.1. SOFTWARE SPECIFICATION

5.2. HARDWARE SPECIFICATION

**6. APPENDICES**

6.1. SOURCE CODE

6.2. SCREENSHOTS

**7. CONCLUSION**

**8. REFERENCE**

## ABSTRACT

The project titled ‘Flower Valley hosted on IBM Cloud’ is a simple Node.js Web Application which has been hosted on IBM Cloud Foundry. ’Flower Valley’, which is a photography collection of flowers, uses MongoDB Atlas as the data store, Express as the routing system, Express-session to track the user's session, and Mongoose to connect with MongoDB.

# 1. INTRODUCTION

## 1.1 OBJECTIVE

The objective of this project is to develop a web application which can be accessed from any device at any time. The web application serves as a floral photographic collection, with only approved users having access.

## 1.2 PROJECT DESCRIPTION

The purpose of this project is to develop an easily accessible online flower photography collection. This application is simply accessible to users that have access to the internet. In order to view the flower gallery, users must first register and then log in.

The application runs on IBM Cloud Foundry's Node.js runtime and stores user credentials in MongoDB Atlas, the global cloud database service for modern applications. The front-end design of the application is dependent on HTML, CSS, JavaScript, Bootstrap and jQuery. Users are identified with their Email ID, and authentication is accomplished when the user provides a password that matches with their Email ID. This way, only authorized users can view the photographs published on the website.

## 1.3 SCOPE OF WORK

With every new smartphone featuring a better camera than the last, it’s never been easier or, quite frankly, more fun to take photos. But what do you do with your pictures after you’ve taken them? How do you share them with friends and family? Do you only want to send your pictures to specific people, or do you want to upload them for the entire internet to see? ‘Flower Valley’, being an online personal floral photography collection answers these questions. Users who wish to view the photo collection can do so by creating an account at ‘Flower Valley' and viewing the photographs there.

By enabling continuous delivery via IBM Cloud, ‘Flower Valley' can be expanded to include more photography collections in the future, rather than being limited to just floral photographs. It can be turned into a portfolio that displays the users' many talents, such as cooking, painting, drawing, fashion, and so on. Moreover, web hosting on the IBM Cloud network enables faster speed and performance, easier load balancing between multiple server environments, and safety from server hardware issues.

# 2. SYSTEM ANALYSIS

## 2.1 PROPOSED SYSTEM

The system proposed here is an online floral picture collection that is easily accessible to users who have access to the internet. Users must first register and then log in to view the flower gallery. The application uses IBM Cloud Foundry's Node.js runtime and MongoDB Atlas, a worldwide cloud database solution for modern applications, to store user credentials. ’Flower Valley’, uses MongoDB Atlas as the data store, Express as the routing system, Express-session to track the user's session, and Mongoose to connect with MongoDB. HTML, CSS, JavaScript, Bootstrap, and jQuery are all used in the application's front-end design.

# 3. SOFTWARE DEVELOPMENT ENVIRONMENT

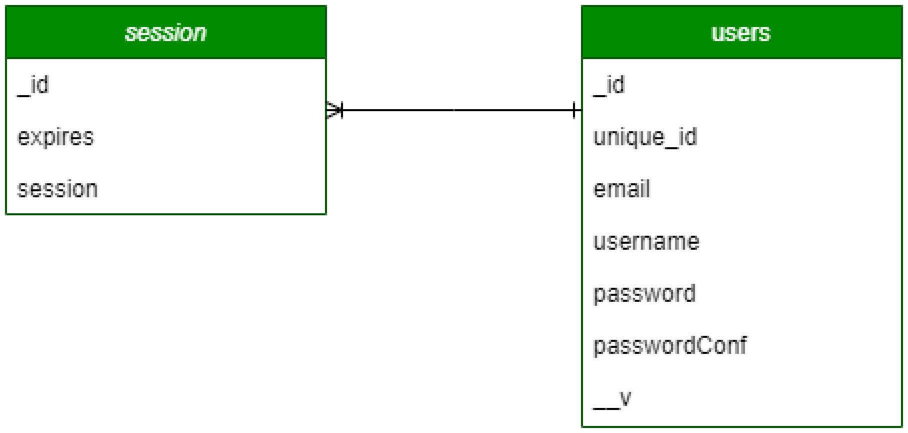
‘Flower Valley’ is a simple online floral photography that runs on IBM Cloud Foundry’s Node.js runtime and stores user credentials in MongoDB Atlas, the global cloud database service for modern applications. Since Node.js is a lightweight, scalable and open-source language platform, it improves the efficiency of the development process by bridging the gap between frontend and backend applications. The Express framework, on the other hand, is built on top of the node.js framework and offers a robust set of features for building single-page, multi-page, and hybrid web applications. The goal of using MongoDB Atlas as the database service is to create a data store that is high-performing, highly available, and scales automatically.

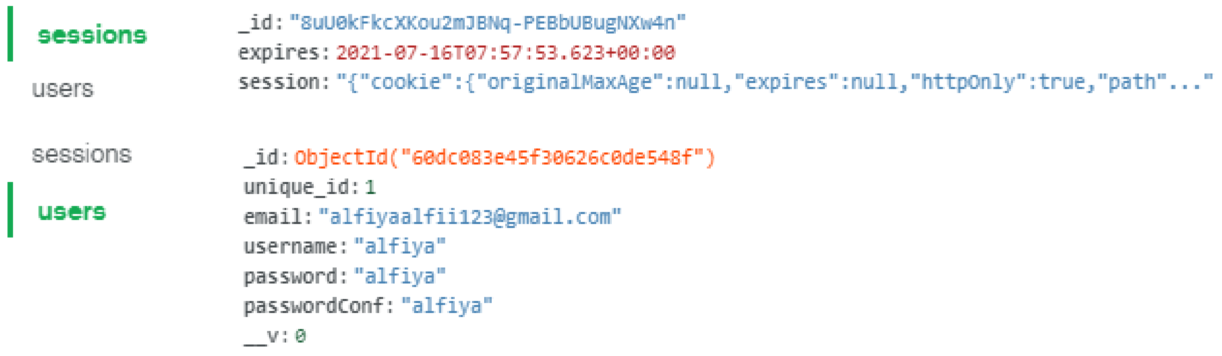
The common node packages used in the application includes Express, Express-session, Mongoose and Body-parser. Express is utilized as the routing system, Express-session as user session tracker, Mongoose as a connector to MongoDB Atlas and Body-parser as a parser. While hosted, the application utilizes IBM Cloud Foundry’s *sdk-for-nodejs* which is the default buildpack for Node.js applications in IBM Cloud.

# 4. SYSTEM DESIGN

## 4.1 DATA MODEL DIAGRAM

NoSQL, unlike SQL which has ER and class diagrams, has neither names nor constraints for data modeling diagram(s). The obvious reason is NoSQL's lack of hard and fast relationship rules, which aims to get a developer started with minimum requirements. The diagram below represents the collections and documents used in the web application, where *session* stores the user's session details and *users* stores the user credentials and the example that follows are documents. .

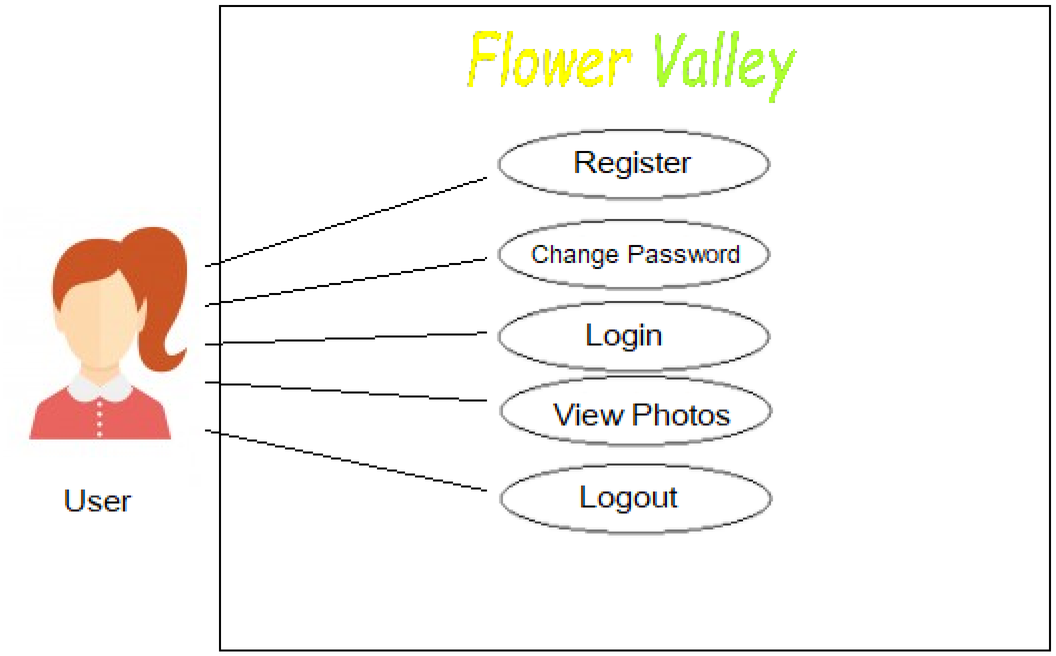




```
_id: "8uU0kFkcXKou2mJBNq-PEBbUBugNXw4n"
expires: 2021-07-16T07:57:53.623+00:00
session: "{"cookie":{"originalMaxAge":null,"expires":null,"httpOnly":true,"path"..."

_id: ObjectId("60dc083e45f30626c0de548f")
unique_id: 1
email: "alfiyaalfii123@gmail.com"
username: "alfiya"
password: "alfiya"
passwordConf: "alfiya"
__v: 0
```

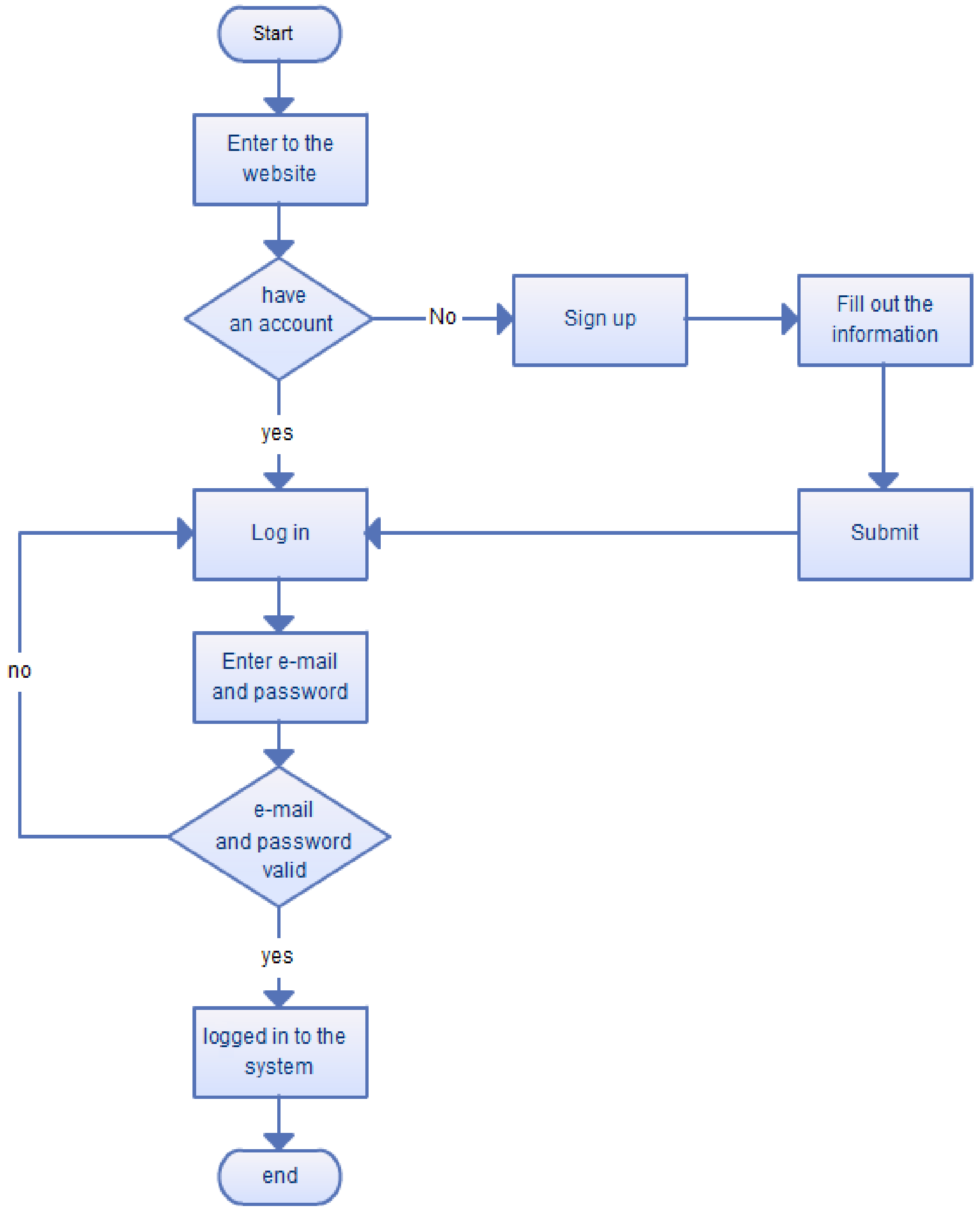
## 4.2 USE CASE DIAGRAM

A use case diagram is a visual representation of the processes that take place within the application. It summarizes the details of the application's users and their interactions with the application. The users can create an account, login, change password and view photographs displayed on the website.



## 4.3 FLOW CHART

A flowchart is a picture of the separate steps of a process in sequential order. It is a generic tool that can be adapted for a wide variety of purposes, and can be used to describe various processes, such as a manufacturing process, an administrative or service process, or a project plan.



# 5. SYSTEM REQUIREMENTS

## 5.1 SOFTWARE SPECIFICATION

| | | |
|---|---|---|
| Operating System | : | Windows 7 or above |
| Front End | : | HTML, CSS, JavaScript, jQuery, Bootstrap |
| Back End | : | Node.js, MongoDB |
| Node.js Packages | : | Express, Express-session, Mongoose |
| Code Editor | : | Visual Studio Code |
| Cloud Services | : | IBM Cloud Foundry, MongoDB Atlas |
| Browser | : | Google Chrome |

## 5.2 HARDWARE SPECIFICATION

| | | |
|---|---|---|
| RAM | : | 1 GB or above |
| Processor | : | 1 GHz or more |
| Hard Drive | : | 32 GB or above |
| Network Connectivity | : | LAN or Wi-Fi |

# 6. APPENDICES

## 6.1 SOURCE CODE

### 1. package.json

```
{
  "name": "alfiya-flowervalley-em2-project",
  "version": "1.0.0",
  "description": "Nodejs Web Application with a photography collection of
flowers",
  "main": "server.js",
  "dependencies": {
    "connect-mongo": "^3.0.0",
    "dotenv": "^8.2.0",
    "ejs": "^3.0.1",
    "express": "^4.17.1",
    "express-session": "^1.16.2",
    "mongoose": "^5.13.0"
  },
  "devDependencies": {},
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "node server.js"
  }
}
```

### 2. server.js

```
var express = require('express');
var env = require('dotenv').config()
var ejs = require('ejs');
var path = require('path');
var app = express();
var bodyParser = require('body-parser');
var mongoose = require('mongoose');
var session = require('express-session');
var MongoStore = require('connect-mongo')(session);

mongoose.connect('mongodb+srv://Alfiya:<password>@cluster0.u8wre.mongodb.n
et/FlowerValley?retryWrites=true&w=majority', {
  useNewUrlParser: true,
  useUnifiedTopology: true
}, (err) => {
  if (!err) {
    console.log('MongoDB Connection Succeeded.');
  } else {
    console.log('Error in DB connection : ' + err);
  }
});

var db = mongoose.connection;
db.on('error', console.error.bind(console, 'connection error:'));
db.once('open', function () {
});

app.use(session({
  secret: 'work hard',
  resave: true,
  saveUninitialized: false,
  store: new MongoStore({
    mongooseConnection: db
  })
}));

app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'ejs');

app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: false }));

app.use(express.static(__dirname + '/views'));

var index = require('./routes/index');
app.use('/', index);
```

```
// catch 404 and forward to error handler
app.use(function (req, res, next) {
  var err = new Error('File Not Found');
  err.status = 404;
  next(err);
});

// error handler
// define as the last app.use callback
app.use(function (err, req, res, next) {
  res.status(err.status || 500);
  res.send(err.message);
});


const PORT = process.env.PORT || 3000;
app.listen(PORT, function () {
  console.log('Server is started on http://localhost:'+PORT);
});
```

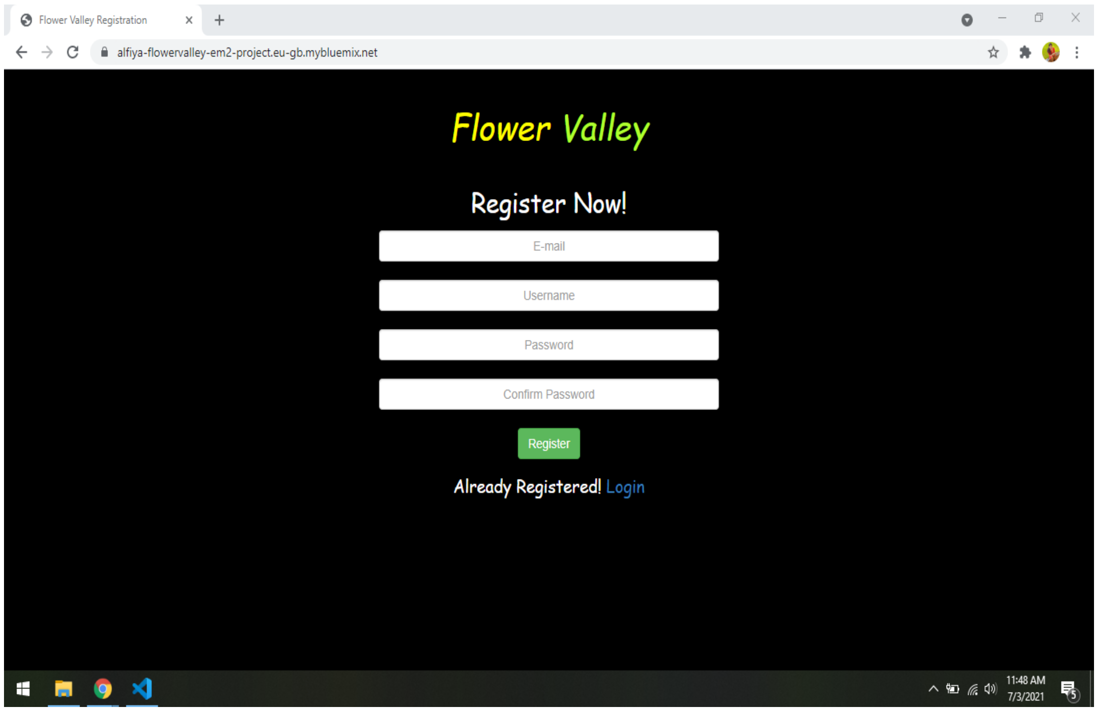
## 3. index.ejs

```
<!DOCTYPE html>
<html>
<head>
    <title>Flower Valley Registration</title>
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.cs
s">
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js">
    </script>
    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js">
</script>
    <link rel="stylesheet" href="./css/main.css">
    <script type="text/javascript">
        $(document).ready(function(){

            $( "#form1" ).submit(function(event) {
                event.preventDefault();

                $.ajax({
                    type: 'POST',
                    url: '/',
                    data: $('#form1').serialize(),
                    dataType: "json",
                    success: function(response){
                        //alert("a");
                        //console.log(response.Success);
                        $('#form1')[0].reset();


document.getElementById("check").innerHTML=response.Success;
                            //ADD THIS CODE
                            setTimeout(function(){

document.getElementById("check").innerHTML="";
                            },3000);
                            if (response.Success=="You are registered,You
can login now.") {
                                document.getElementById("aa").click();
                            };
                        },
                        error: function() {
                        }
                    })
            });
        });
    </script>
</head>
<body>

    <div class="col-md-4 col-md-offset-4">
        <h2><i><span class="yellow">Flower </span><span
class="green">Valley</span></i></h2>
```

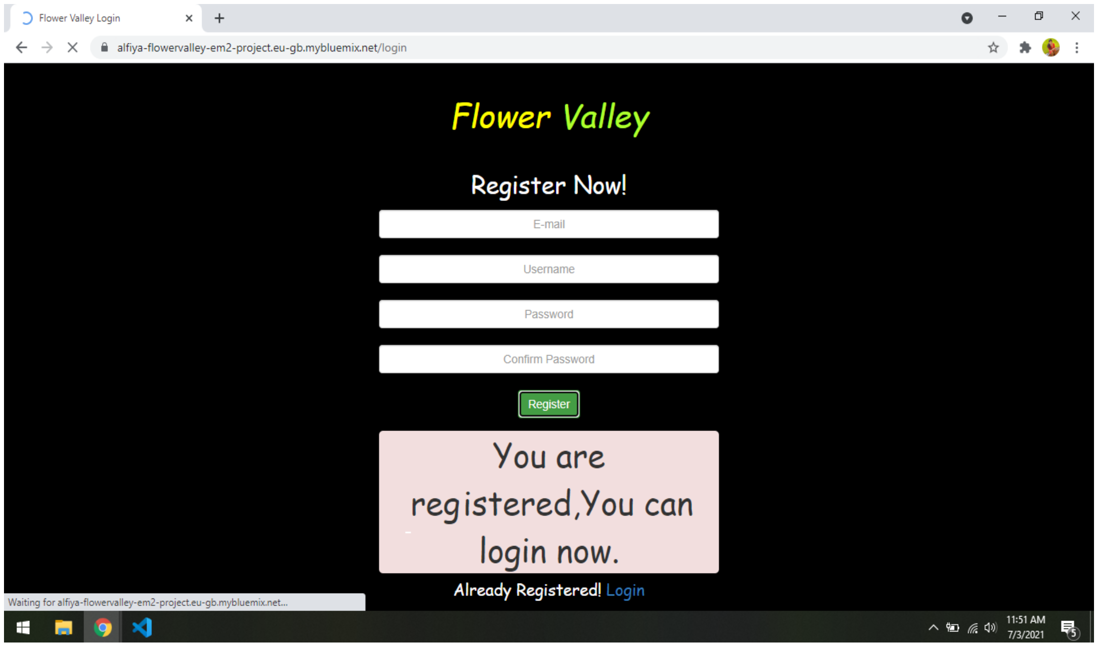
```
        <div>
            <p>Register Now!</p>
        </div>
        <div class="form-group">
            <form id="form1" method="post">
                <input type="email" name="email" placeholder="E-mail"
required="" class="form-control"><br/>
                <input type="text" name="username" placeholder="Username"
required="" class="form-control"><br/>
                <input type="password" name="password"
placeholder="Password" required="" class="form-control"><br/>
                <input type="password" name="passwordConf"
placeholder="Confirm Password" required="" class="form-control"><br/>
                <input type="submit" value="Register" class="btn
btn-success">
            </form>
        </div>

        <div class="mssg bg-danger">
            <span id="check"></span>
        </div>
        <div>
            <span id="span">Already Registered! <a
href="/login">Login</a></span>
        </div>
        <div id="LangTable"><a href="/login" id="aa"></a>
        </div>
    </div>
</body>
</html>
```

## 6.2 SCREENSHOTS

### 1. Flower Valley Registration


Flower Valley Registration
alfiya-flowervalley-em2-project.eu-gb.mybluemix.net
Flower Valley
Register Now!
E-mail
Username
Password
Confirm Password
Register
Already Registered! Login
11:48 AM
7/3/2021

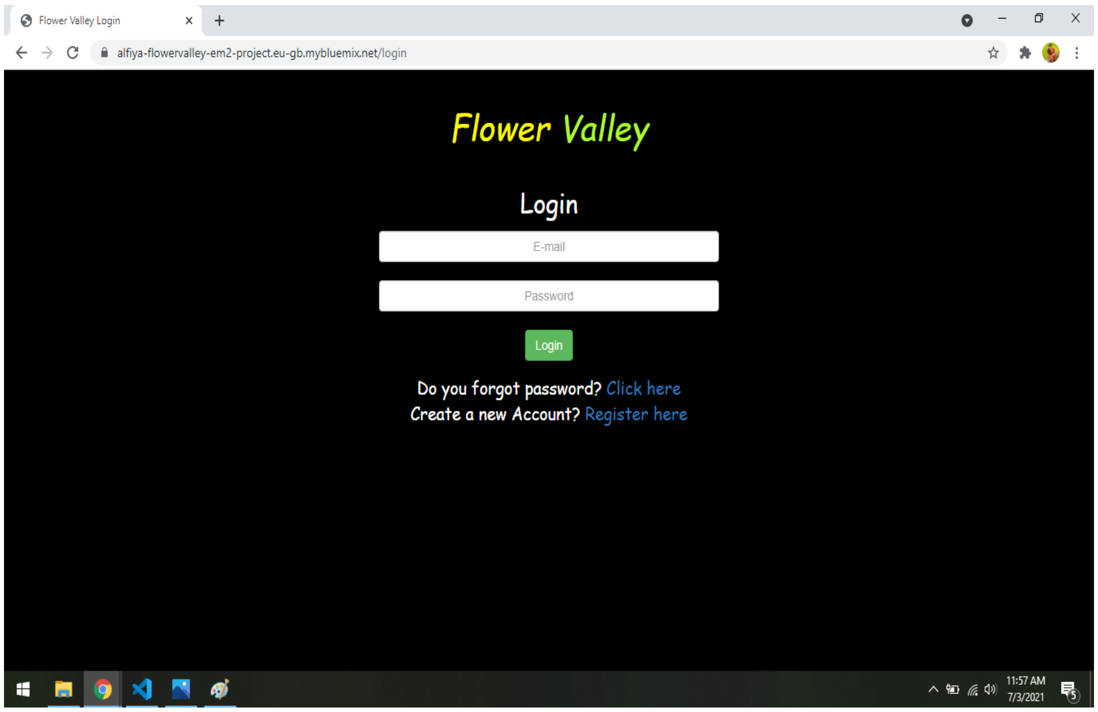
### 2. Registration Successful


Flower Valley Login
alfiya-flowervalley-em2-project.eu-gb.mybluemix.net/login
Flower Valley
Register Now!
E-mail
Username
Password
Confirm Password
Register

You are registered,You can login now.

Already Registered! Login
Waiting for alfiya-flowervalley-em2-project.eu-gb.mybluemix.net...
11:51 AM
7/3/2021

## 3. Login Form


Flower Valley Login

alfiya-flowervalley-em2-project.eu-gb.mybluemix.net/login

Flower Valley

Login

E-mail

Password

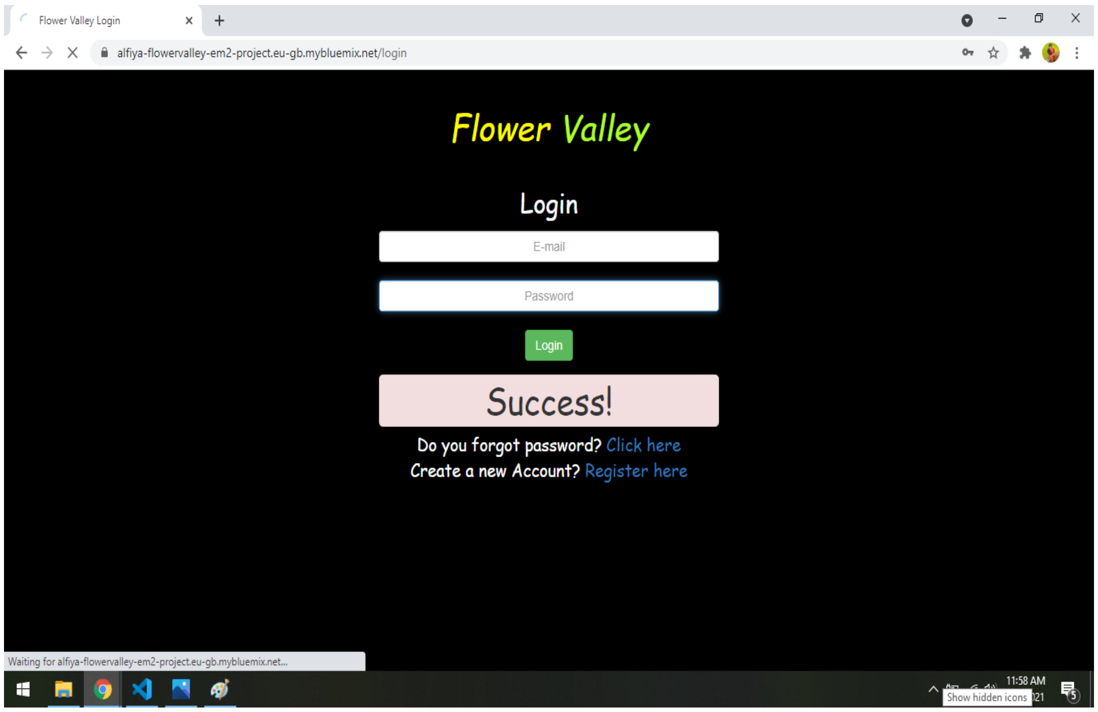
Login

Do you forgot password? Click here

Create a new Account? Register here

11:57 AM
7/3/2021

## 4. Login Successful


Flower Valley Login

alfiya-flowervalley-em2-project.eu-gb.mybluemix.net/login

Flower Valley

Login

E-mail

Password

Login

Success!

Do you forgot password? Click here

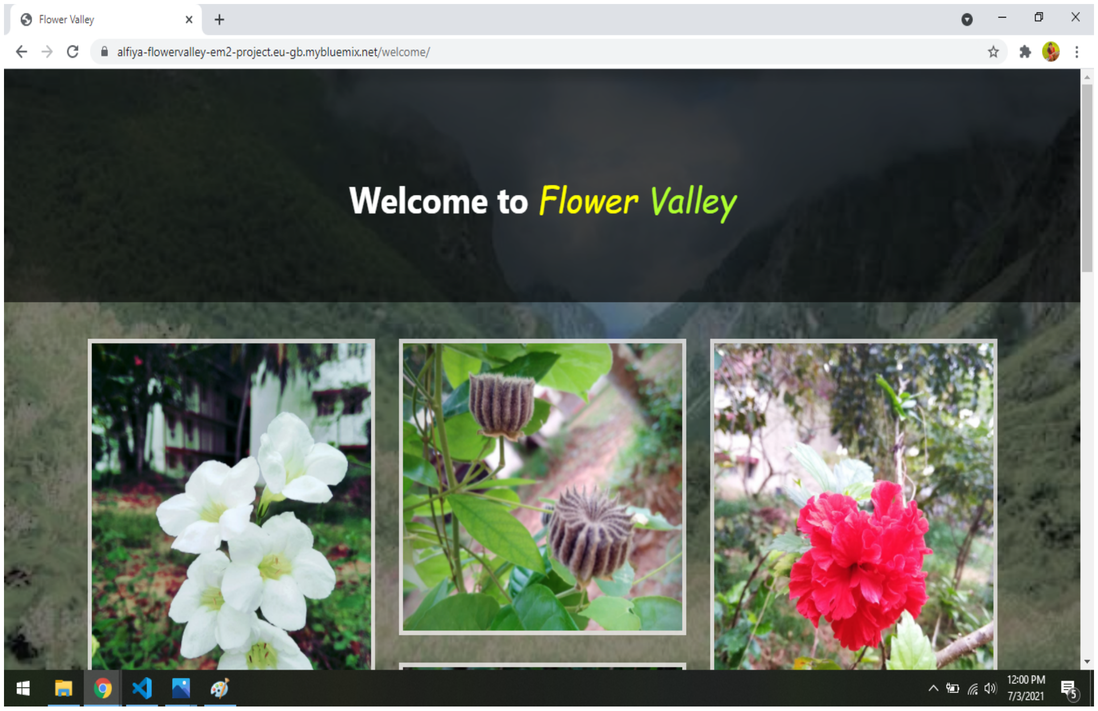
Create a new Account? Register here

Waiting for alfiya-flowervalley-em2-project.eu-gb.mybluemix.net...

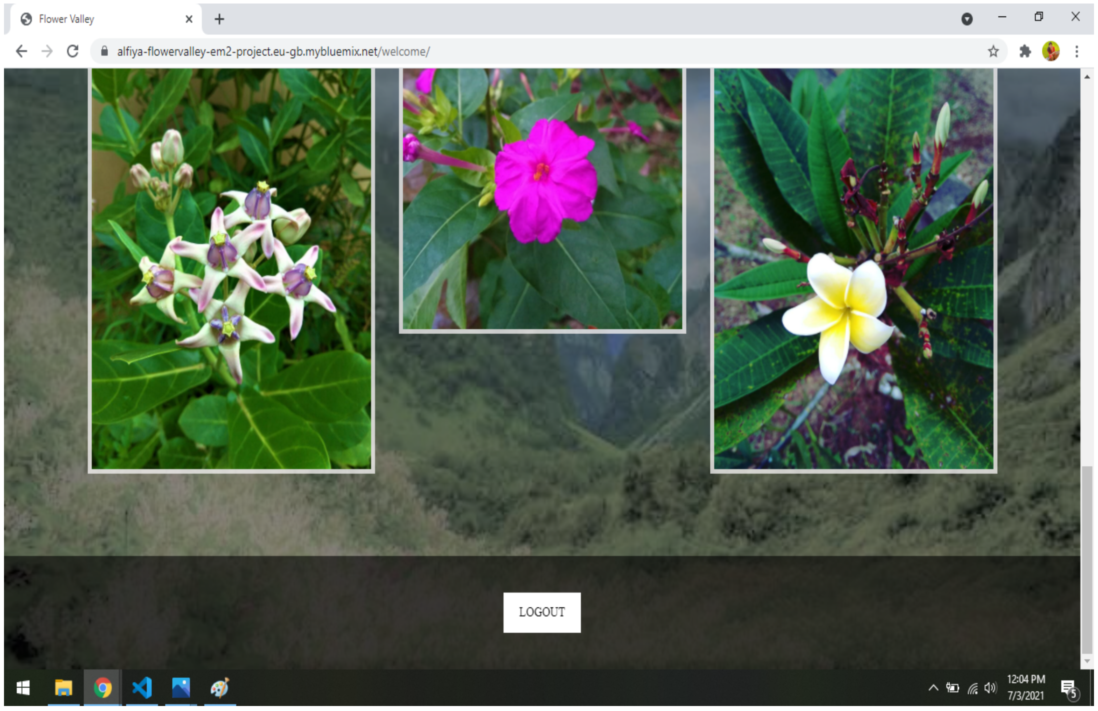
11:58 AM

Show hidden icons

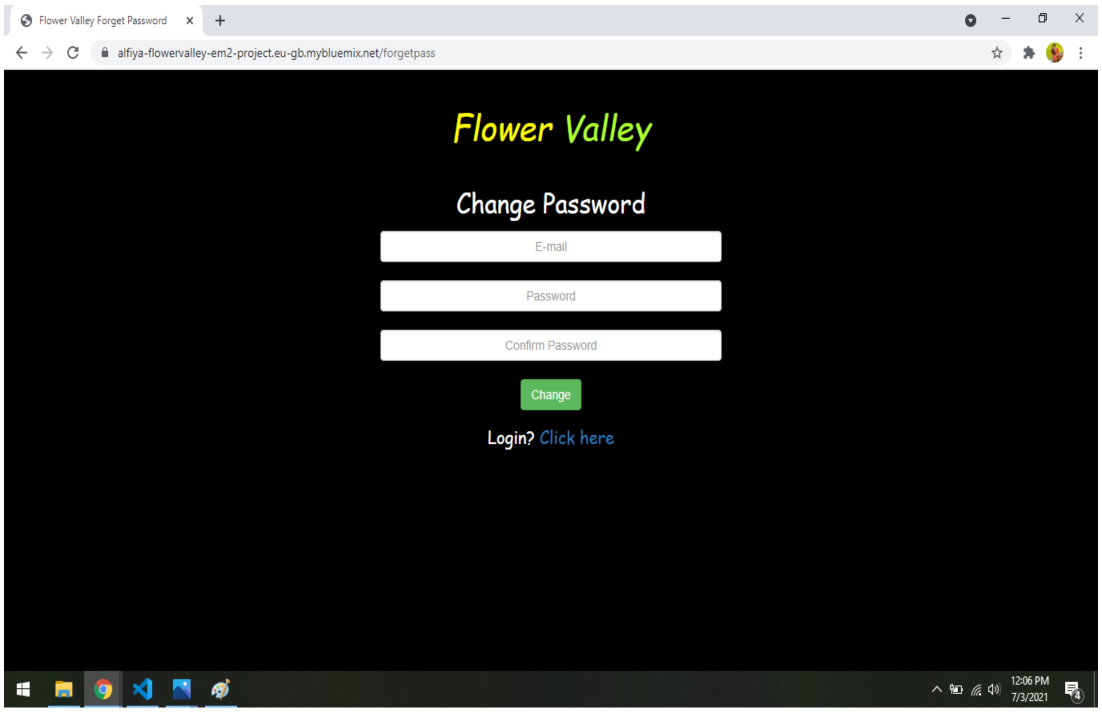
## 5. Flower Valley Homepage


Flower Valley
alfiya-flowervalley-em2-project.eu-gb.mybluemix.net/welcome/
Welcome to Flower Valley
12:00 PM
7/3/2021

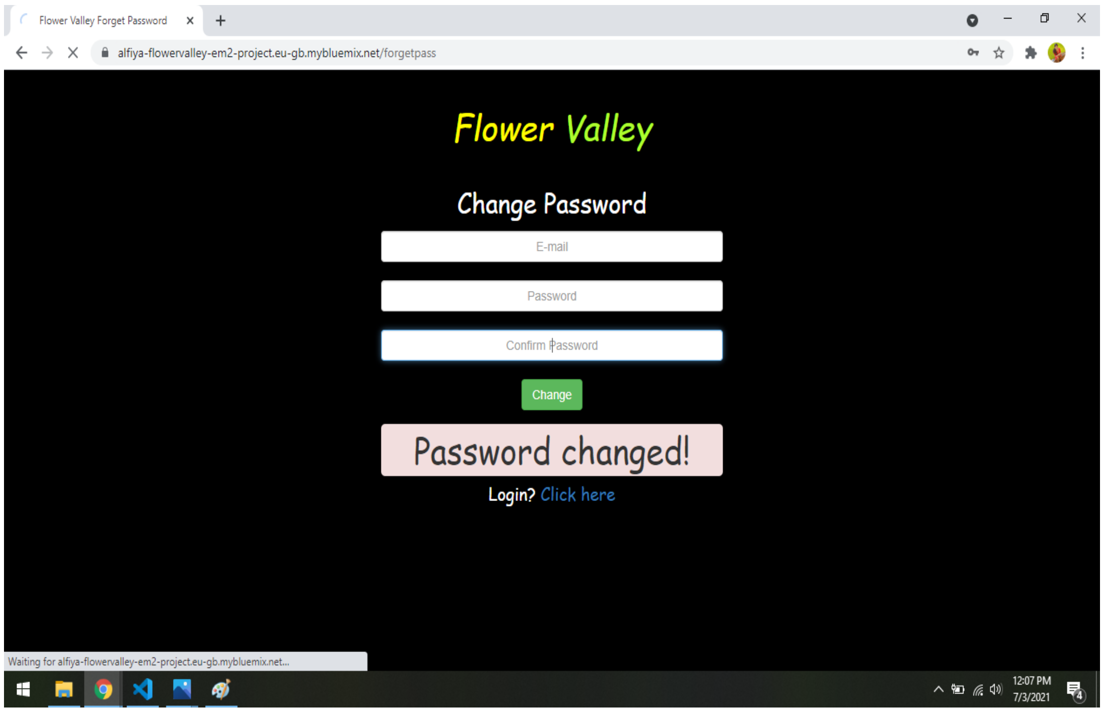
## 6. Logout


Flower Valley
alfiya-flowervalley-em2-project.eu-gb.mybluemix.net/welcome/
LOGOUT
12:04 PM
7/3/2021

## 7. Change Password



## 8. Password Changed


Flower Valley Forget Password
alfiya-flowervalley-em2-project.eu-gb.mybluemix.net/forgetpass
Flower Valley
Change Password
E-mail
Password
Confirm Password
Change

Password changed!

Login? Click here
Waiting for alfiya-flowervalley-em2-project.eu-gb.mybluemix.net...
12:07 PM
7/3/2021

# 7. CONCLUSION

The goal of this project was to create an easily accessible online gallery of floral photographs. Users must first register and then log in to view the flower gallery. The application uses the Node.js runtime from IBM Cloud Foundry and saves user credentials in MongoDB Atlas, a worldwide cloud database service. Users who have access to the internet can easily use the 'Flower Valley' web application using this link.

## 8. REFERENCE

1. https://cloud.ibm.com/docs/
2. https://www.mongodb.com/cloud/atlas/