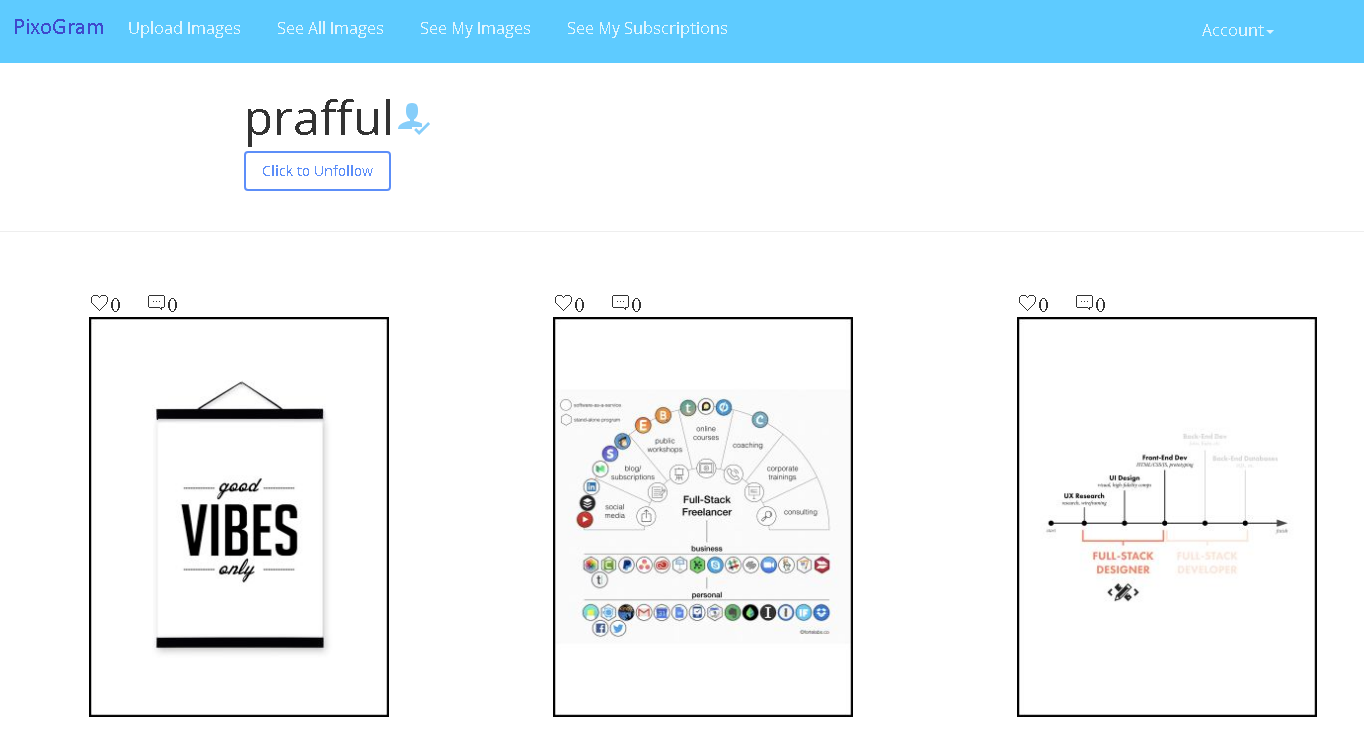
Deployment Microlayer

Prafful Daga  
IIHT

Micro Credential: Maven, GIT, Tomcat & Junit Microlayer

Duration: 4 to 8 Hour



CI & CD with Jenkins and Docker FOR SOCIAL PICTURE SHARING APPLICATION

This page is intentionally left blank.

Contents

[1 Important Instructions 3](#_Toc529802859)

[2 Business-Requirement: 4](#_Toc529802860)

[2.1 Problem Statement: 4](#_Toc529802861)

[3 Configure Jenkins and Docker for the project 5](#_Toc529802862)

[4 CI & CD 6](#_Toc529802863)

[5 Methodology 7](#_Toc529802864)

[5.1 Agile 7](#_Toc529802865)

[6 Technical Specification – Maven, GIT, Tomcat & Junit Microlayer Development Environment 8](#_Toc529802866)

[6.1 Spring Boot Layer 8](#_Toc529802867)

[6.2 Editors 8](#_Toc529802868)

[7 Development Workflow 9](#_Toc529802869)

[8 Important Instructions 10](#_Toc529802870)

[9 Assessment Deliverables 10](#_Toc529802871)

[10 Other Full Stack Layers 11](#_Toc529802872)

[10.1 UI Layer (Not Applicable for Present Case Study) 11](#_Toc529802873)

[10.2 UX Layer (Not Applicable for Present Case Study) 11](#_Toc529802874)

[10.3 Back End Layer (Not Applicable for Present Case Study) 11](#_Toc529802875)

[10.4 Middle Tier Framework Layer (Not Applicable for Present Case Study) 11](#_Toc529802876)

[10.5 ORM & Integration Layer (Not Applicable for Present Case Study) 11](#_Toc529802877)

[10.6 Database Layer (Not Applicable for Present Case Study) 11](#_Toc529802878)

[10.7 Ancillary Layer(Not Applicable for Present Case Study) 11](#_Toc529802879)

[10.8 Deployment & Infra(Applicable for Present Case Study) 11](#_Toc529802880)

# Important Instructions

1. Follow the design specifications mentioned in the case study. You are free to improvise certain specifications mentioned in the case-study. But, for each such improvisation, you should keep the concerned POC informed. **POC will get in touch with concerned team at IIHT.**
2. You should stay **motivated** to initiate such and specific communications as it may have positive influence on the evaluation scores.
3. Please make sure that your code does not have any compilation errors while submitting your case study solution.
4. **Use h2 in-memory OR MySQL database where database functionality is required.**
5. Implement the code using best design standards for:
   1. Variable declarations
   2. Class names
   3. Package names
   4. Code Refactoring

# Business-Requirement:

## Problem Statement:

**The PixoGram (Spring Boot Web Application)** allows you to:

1. Register as a user
2. Login as a user
3. Retrieve/Change password
4. Manage your user account
5. Login/Logout to/from your account on PixoGram
6. Add Content
   1. Upload single/multiple pictures, caption and description
   2. Upload single/multiple videos, caption and description
7. Manage Content
   1. Organize Picture in Gallery
   2. Organize Videos in Playlists
   3. Rename Pictures and Videos
   4. Edit Caption, Description, Comment
8. Social Features
   1. Use emojis in comment
   2. Like or Unlike comment, pictures and videos of other users
   3. Follow/Unfollow other users
9. Edit Pictures
   1. Apply effects to pictures (sepia, grayscale, etc.)
10. Hide Pictures/Videos
11. Activity/Newsfeed
    1. View activity log of user-activity on the PixoGram
12. Offline Functionality:
    1. Certain parts of the application should be available in absence of connectivity.
    2. Relevant areas on the screen should display “Connectivity Not Available”

**In this micro layer you will configure Jenkins and create deployable for Docker. Use the application which you may have developed in either of the microlayer:**

* + - 1. **Spring Core and Spring MVC Microlayer**
      2. **Spring Boot & ORM Microlayer**

**It is recommended that you choose, Spring Boot & ORM solution if it is developed by you in one of the microlayer evaluations.**

# Configure Jenkins and Docker for the project

1. Import the project (as discussed above) in Spring Tool Suite and configure it locally to run it as Spring Boot App.
2. You may need to configure MySQL credentials and database name.
3. Execute the project locally and access the app at <http://localhost:portnumber>
4. Once, it is working fine in local development environment; do the following as a part of deploying the app to docker as container:
   1. Push the app source in internal GIT server. Pl. ask your mentor for the Internal GIT server URL.
   2. Configure Jenkins locally to pull the source from internal GIT repository
   3. Jenkins should build the project and create the deployable (war/jar). It should run the unit tests created in ”Maven, GIT, Junit, Tomcat Micro Layer for PixoGram”
   4. Create a docker file and docker-compose.yaml such that, when you run the following command:
      1. docker-compose up
      2. It should deploy the attached project at <http://localhost>
   5. Hints:
      1. Docker Compose file should:
         1. Create ngix container
         2. Create mysql container
         3. Configure attached “app” to run in docker containers

# CI & CD

Make few changes in the project (source code)

Make it sure that project is running locally in development environment without errors.

If it running locally without errors, push the changes to the internal GIT repository which was connected in 3.4.a

If 3.4.b was done correctly, Jenkins will automatically pull the code updates from internal GIT repo and build and deploy the project with updated code.

Now, when you visit <http://localhost>; you should see the changes in the browser window

# Methodology

## Agile

1. Mentor will ask you about progress as you start implementing CI & CD Layer.
2. Communicate with your mentor via email; ideally daily as you develop the Spring framework Layer.
3. Scope of discussion with your mentor:
   1. Q/A
   2. New Ideas and New feature implementations
   3. Any development related challenges
   4. Skill Gaps
   5. Another pointers key to Spring Development

# Technical Specification – Maven, GIT, Tomcat & Junit Microlayer Development Environment

## Spring Boot Layer

|  |  |
| --- | --- |
| **Framework(s)/SDK/Libraries** | **Version** |
| Spring Boot, Hibernate, Spring Data | - |
| Maven | - |
| Junit | - |
| GIT | - |
| Jenkins | - |
| Docker | - |

## Editors

|  |  |
| --- | --- |
| **Name** | **Version** |
| STS | - |

# Development Workflow

1. You must follow following process while deploying the project
   1. Feel free to take approval at each stage:
      1. Approval includes:
         1. Communicating with mentor via email to showcase the progress.
         2. Progress must be shared with mentor as each CI/CD layer is crossed.
         3. It is recommended to avoid moving to next stage until feedback from the mentor is received.
         4. Pushing assets/code to GIT repository.

# Important Instructions

1. Follow the design specifications mentioned in the case study. You are free to improvise certain specifications mentioned in the case-study. But, for each such improvisation, you should keep the concerned POC informed. **POC will get in touch with concerned team at IIHT.**
2. You should stay **motivated** to initiate such and specific communications as it may have positive influence on the evaluation scores.
3. Please make sure that your code does not have any compilation errors while submitting your case study solution.
5. **Use h2 in-memory OR MySQL database where database functionality is required.**
6. Implement the code using best design standards for:
   1. Variable declarations
   2. Class names
   3. Package names
   4. Code Refactoring

# Assessment Deliverables

1. Dump of command “git log”
2. Jenkins report about number of times it pulled the code from GIT
3. Working POM.xml for the project.
4. Jenkins/build report and code coverage report from JaCoCo

# Other Full Stack Layers

## UI Layer (Not Applicable for Present Case Study)

|  |  |
| --- | --- |
| HTML5 | - |
| CSS3 | - |
| Bootstrap/Material | - |
| Typescript | - |

## UX Layer (Not Applicable for Present Case Study)

|  |  |
| --- | --- |
| Angular | 6 |
| Javascript & JQuery | - |
| Typescript | - |

## Back End Layer (Not Applicable for Present Case Study)

|  |  |
| --- | --- |
| JDK | 1.8 or above |

## Middle Tier Framework Layer (Not Applicable for Present Case Study)

|  |  |  |
| --- | --- | --- |
| **Technology** | **Framework(s)/SDK/Libraries** | **Version** |
| Spring Framework | Spring MVC | 5.0 or above |

## ORM & Integration Layer (Not Applicable for Present Case Study)

|  |  |  |
| --- | --- | --- |
| **Technology** | **Framework(s)/SDK/Libraries** | **Version** |
| Spring | Spring Boot | 2.0 |
| Java JPA | Hibernate | 5.0 or above |
| Spring Data | 2.0 |

## Database Layer (Not Applicable for Present Case Study)

|  |  |  |
| --- | --- | --- |
| MySQL | MySQL | 7.x + |

## Ancillary Layer(Not Applicable for Present Case Study)

|  |  |  |
| --- | --- | --- |
| **wTechnology** | **Framework(s)/SDK/Libraries** | **Version** |
| Source Code Management Tool | GIT | 2.18 |
| Build Tool/JAVA Stack | Maven | 3.5.x |
| Testing Tool/JAVA Stack | Junit/Spring Test | 4.x/5.x |
| Javascript Dependency Management Tool | NPM | 6.x.x |

## Deployment & Infra(Applicable for Present Case Study)

|  |  |  |
| --- | --- | --- |
| **Technology** | **Framework(s)/SDK/Libraries** | **Version** |
| Docker | - | 17.06.2 |
| Apache Tomcat | - | 9.0 |
| Jenkins | - | 2.121.2 |