MINI PROJECT
(2019-20)

Design and Deploying of web Application in Docker Container

SYNOPSIS

Institute of Engineering & Technology

Team Members

Sahil Srivastava
(171500281)
Shruti Sharma
(171500332)

Supervised By:

Mr. Mandip Singh (Asst. Professor)
Department of Computer Engineering & Applications

Table of Contents

<Project Name>

1. Introduction

The purpose of this document is to present a detailed description of the project on Developing and Deploying a Web App on Docker Container . It will explain the purpose and features of this project, the interfaces of the project, what the web app will do, the constraints under which it must operate and how it will react to external stimuli. This document is intended for the supervisor as well as the developers

1.1 Purpose

Docker is used to build, ship, and run distributed applications. You can configure your service instances to run in Docker containers.
A Docker container allows an application to be packed together with its dependencies into a portable virtual package that can run with multi-platform support, isolation, and resource limits applied.

The main benefits of running a service instance in a Docker container are as follows:

Deployment

Application can be packed together to a portable Docker image that can run on different platforms.

Isolation

Containers isolate applications from each other and the underlying infrastructure while providing an added layer of protection for the application.

1.2 Scope

Now-a-days, docker is being used widely across the globe. Docker enables you to rapidly deploy server environments in “containers.” Now, you might question why use Docker rather than VMware or Oracle’s VirtualBox?

While Docker utilizes the virtualization technology in the Linux kernel, it does *not* create virtual machines (in fact, if you run Docker on MacOS or Windows, you’ll have to run it on a virtual machine).

Instead of abstracting the hardware, containers abstract the OS. Each container technology features an explicit purpose, limiting the scope of the technology. Docker’s runs Linux, whereas Citrix’s XenApp runs Windows Server. Every container shares the exact same OS, reducing the overhead to the host system. Recall each VM runs its own copy of the OS, adding overhead for each instance.Containers exist to run a single application. Like XenApp, every Docker container targets a specific application. Docker also incorporates container management solutions for easy scripting and automation (especially important when

considering containers' focus on reducing execution time). A Docker container focuses on only one application at a time, and the environment is chromed to prevent access outside the container's directory tree.

1.3 Definitions, Acronyms, and Abbreviations

Acronyms/Abbreviations	Definitions
Web	Website
App	Application

1.4 References

- https://docs.docker.com
- http://stackoverflow.com/questions/31448249/how-to-deploy-web-service-on-docker-container
- https://github.com/CoreyMSchafer/code
- https://github.com/CoreyMSchafer/code_snippets/tree/master/Django_Blog
- https://getbootstrap.com/docs/4.0/getting-started/introduction/#starter-template

1.5 Overview

This is a working document and, as such, is subject to change. In its initial form, it is incomplete by definition, and will require continuing refinement. Requirements may be modified and additional requirements may be added as development progresses and the system description becomes more refined. This information will serve as a framework for the current definition and future evolution of the University Academic Portal.

<Project Name>

2. General Description

- The web app is a blogging website where different users can like different posts or twitter posts.
- The web app contains an authentication system where a user or an admin can log in using his/her credentials and log out.
- The web app also contains a registration system where a new new user can register on the website.
- The login has a forgot password link where a user can reset his/her password by using his email.
- The users have also the ability to delete post.

Docker is an open source project that automates the deployment of applications inside software containers. Docker containers wrap up a piece of software in complete filesystem that contains everything it needs to run: code, runtime, system tools, and system libraries- anything you can install on a server. Docker provides an additional layer of abstraction and automation of operating system level virtualization on Windows and Linux. Docker uses the resource isolation features of the Linux kernel such as cgroups and kernel namespaces, and a union-capable file system such as OverlayFS and others to allow independent 'containers' to run within a single Linux Instance, avoiding overhead of starting and maintaining virtual machines

2.1 Product Perspective

(BLOG) is meant to serve as a common platform where management of everyday academic tasks can be carried out conveniently. Our goal is to develop a replacement to the academic portal used at GLA University making it more users friendly and to promote academic networking among the users.

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 System Interface

Apache will be used as web server. The user inputs data via the web server using HTML forms. The actual program that will perform the operations is written in PHP.

3.1.2 User interface

The new system shall provide a very intuitive and simple interface to the user and the administrator, so that the user can easily navigate through pages, assignments, groups and sub- groups, start discussion threads, blogs, survey, upload assignments, share data, old papers sharing and the administrator can easily manage groups and revoke user

3.1.3 HARDWARE REQUIREMENTS

The web application will be hosted on a web server which is listening on the web standard port, port 80.

b) Client side

Monitor screen – the software shall display information to the user via the monitor screen

Mouse – the software shall interact with the movement of the mouse and the mouse buttons. The mouse shall activate areas for data input, command buttons and select options from menus.

Keyboard – the software shall interact with the keystrokes of the keyboard. The keyboard will input data into the active area of the database.

3.1.4 Software Interface

a) Server side
An Apache web server will accept all requests from the client and forward it accordingly. A database will be hosted centrally using MySQL.

b) Client side
An OS which is capable of running a modern web browser which supports JavaScript and HTML5 and Django.

3.1.5 Communication Interfaces

The HTPP or HTTPS protocol(s) will be used to facilitate communication between the client and server.

3.1.6 Memory Constraints

Memory constraints will come into play when the size of MySQL grows to a considerable size.

3.2 Functional Requirements

This section describes specific features of the software project. If desired, some requirements may be specified in the use-case format and listed in the Use Cases Section.

3.3 Use Cases

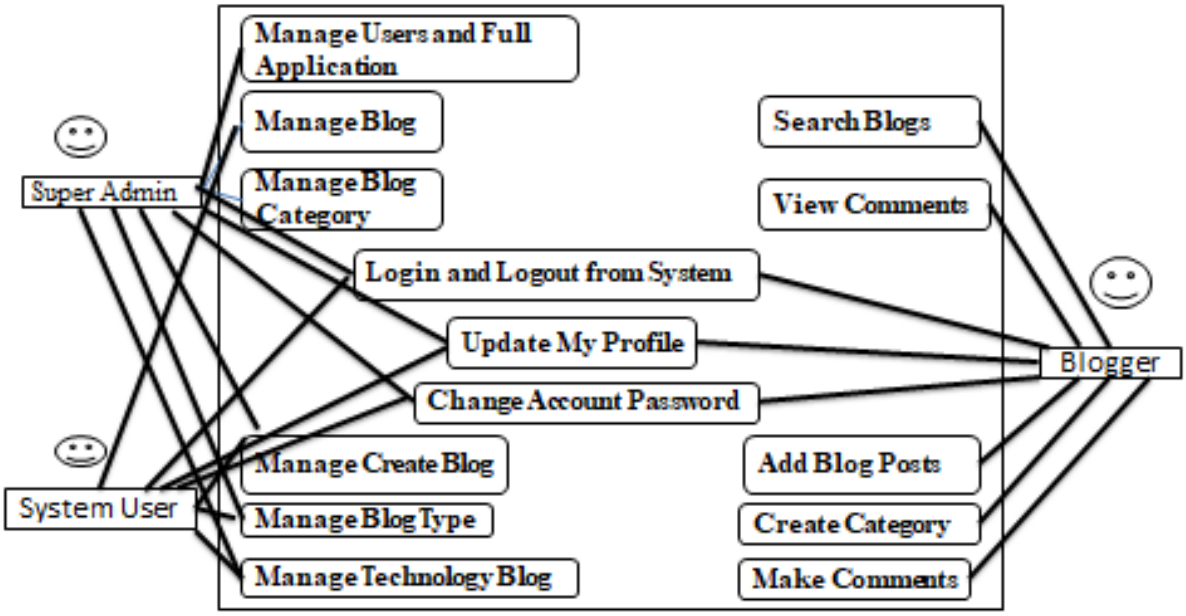
- Graphic depiction of the interactions among the elements of the Blogging System

- Represents the methodology used in system to identify, clarify, and organize system requirements of Blogging System

<Project Name>

- Main actors are:-

Super Admin, System User, Blogger, Anonymous Users who perform the different type of use cases such as Manage Blog, Manage Blog Category, Manage Create Blog, Manage Blog Type, Manage Comment, Manage Technology, Manage Web Page



3.4 Non-Functional Requirements

Often these requirements must be achieved at a system-wide level rather than at a unit level. The following requirements in the following sections in measurable terms. Non-functional requirements may exist as the following attributes

3.4.1 Performance

Docker is now everywhere. Over the past few years, a lot of modern-day software has now moved to become packaged in a Docker container, and with good reason. One of the biggest benefits touted about Docker containers is their speed. You don't get lightning-fast performance out of the box without Docker performance tuning.

<Project Name>

Part 1
Optimizing the speed of containers before we ship (build-time configuration):
Docker file optimization
Docker context relevancy
Push/pull latency cost

Part 2
Optimizing your containers in production:
Host/container relationship
Container performance data
Leveraging APMs for easier performance data

3.4.2 Reliability

Docker containers can be used to back up and restore data efficiently, but they come with some requirements that should be taken into account before implementing containerization.
The original goal of containerization, as exemplified by Docker, was to provide an environment for short-lived, temporary instantiations of applications. The benefit was the ability to develop, deploy and implement applications without the overhead and maintenance of physical servers or virtual machines. While Docker container backup can rapidly deploy code and be more responsive to a business, emerging adoption has shown that persistent data and persistent containers are still requirements when backing up with Docker.

3.4.3 Availability

Docker Universal Control Plane is designed for high availability (HA). You can join multiple manager nodes to the cluster, so that if one manager node fails, another can automatically take its place without impact to the cluster.

Having multiple manager nodes in your cluster allows you to:

Handle manager node failures,
Load-balance user requests across all manager nodes.
Size your deployment
To make the cluster tolerant to more failures, add additional replica nodes to your cluster.

Manager nodes	Failures tolerated
1	0
3	1
5	2

For production-grade deployments, follow these rules of thumb:

For high availability with minimal network overhead, the recommended number of manager nodes is 3. The recommended maximum number of manager nodes is 5. Adding too many manager nodes to the cluster can lead to performance degradation, because changes to configurations must be replicated across all manager nodes.

When a manager node fails, the number of failures tolerated by your cluster decreases. Don't leave that node offline for too long.
You should distribute your manager nodes across different availability zones. This way your cluster can continue working even if an entire availability zone goes down.

3.4.4 Security

Docker allows you to use containers from untrusted public repositories, which increases the need to scrutinize whether the container was created securely and whether it is free of any corrupt or malicious files. For this, use a multi-purpose security tool that gives extensive dev-to-production security controls.
There are four major areas to consider when reviewing Docker security:

the intrinsic security of the kernel and its support for namespaces and cgroups;
the attack surface of the Docker daemon itself;
loopholes in the container configuration profile, either by default, or when customized by users.
the "hardening" security features of the kernel and how they interact with containers.

3.4.5 Maintainability

Eliminate the "it works on my machine" problem once and for all. One of the benefits that the entire team will appreciate is parity. Parity, in terms of Docker, means that your images run the same no matter which server or whose laptop they are running on. For your developers, this means less time spent setting up environments, debugging environment-specific issues, and a more portable and easy-to-set-up codebase. Parity also means your production infrastructure will be more reliable and easier to maintain.

3.4.6 Portability

In one sense, Docker containers offer a lot of portability. With containers, you can build an application once, place it inside a container image (or series of images, if the app is composed of multiple services, each running in a separate container) and run it on any host environment that supports Docker and runs on the same family of operating system. Porting an application from one Linux distribution to another using Docker is much simpler than doing it in another way. Before Docker, your best bet for moving an application from Ubuntu to RHEL was to build a new package for it, because you couldn't install a Debian package (the kind of installation package Ubuntu uses) on an RHEL server (which uses RPM packages). (Well, you technically could do this using a tool like Alien, but this is messy and not practical for production.)

.

3.5 Design Constraints

1. The communication between the portal software and the database will be in SQL.
2. The portal layout will be produced with HTML/CSS.
3. The product will be written in PHP.
4. The output must be compatible with W3C XHTML 1.0
5. The source code must follow the coding conventions of PHP.
6. System administrators must have access to comprehensive documentation.

3.6 Logical Database Requirements

All data will be saved in the database: user accounts and profiles, discussion data, messages etc. (except files which are stored on the disk.) The database allows concurrent access and will be kept consistent at all times, requiring a good database design.

4. Analysis Models

4.1 Sequence Diagrams

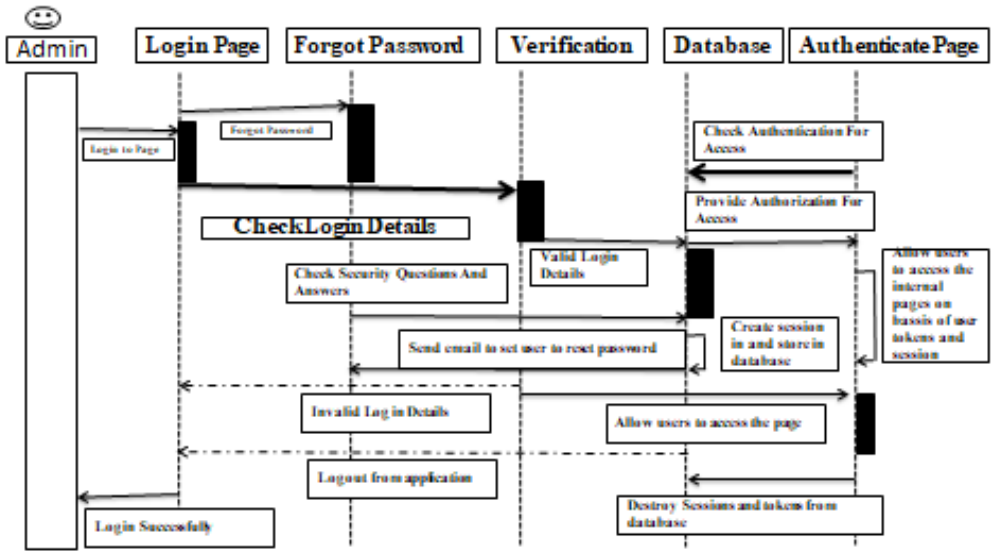
- ✓ UML sequence diagram of Blogging System which shows the interaction between the objects of Blog, Web Page, Blog Type, Category, and Comment.

- ✓ Demonstrates how the login page works in a blogging system

- ✓ Instance of class objects involved in this UML Sequence Diagram of Blogging System are as follows:-

- Blog Object
- Web Page Object
- Blog Type Object
- Category Object
- Comment Object

<Project Name>



4.2 Data Flow Diagrams (DFD)

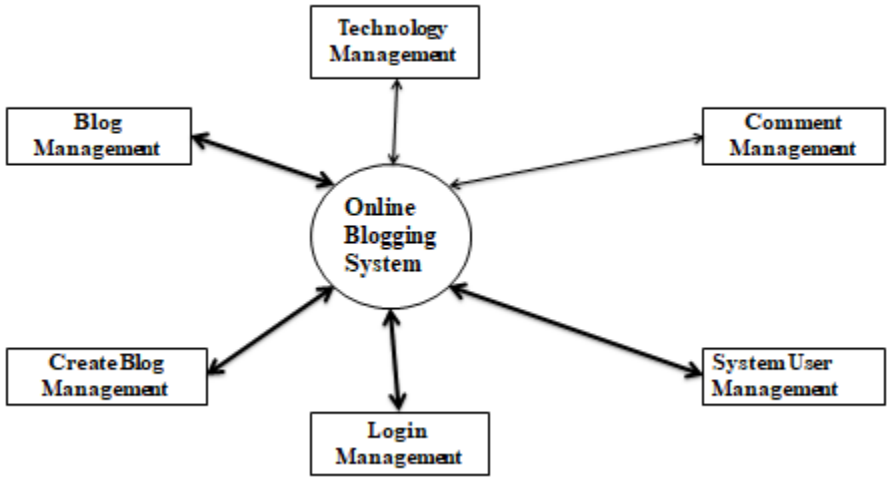
4.2.1 Zero Level DFD of BLOG

✓ Basic overview of the whole online Blog system where processes being analysed or modelled

✓ Designed to be an at-a-glance view of Comment, Technology, Blog and Web page showing the system as a single high-level-process

✓ Elaborated the high level process of Online Blogging

✓ Shows the relationships of Comment, Technology, Blog and Web Page to external entities of Blog, Blog Category and Create Blog



<Project Name>

High Level Entities and process flow of online blogging system:

Managing all the Blog

Managing all the Blog Category

Managing all the Create Blog

Managing all the Blog Type

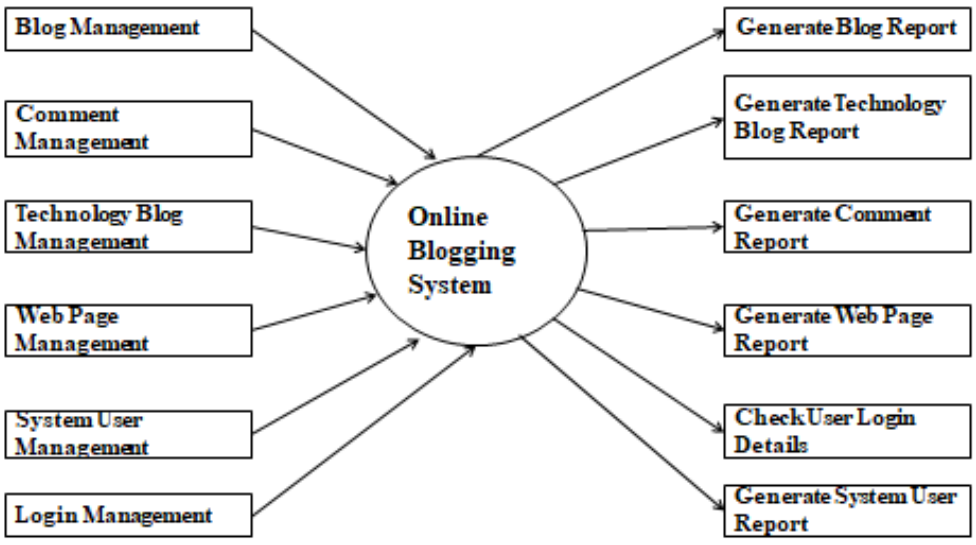
Managing all the Comment

Managing all the Technology Blog

Managing all the Web Page

4.2.2 First Level DFD of Blog

- Shows how the system is divided into sub-systems, each of which deals with one or more of the data flows to or from an external agent, and which together provide all of the functionality of the Online Blogging System as a whole.
- Identifies internal data stores of Web Page, Technology Blog, Comment, Blog Type, Create Blog that must be present in order for the online blogging system to do its job
- Shows the flow of data between various parts of Blog, Create Blog, Technology Blog, Web Page
- Provides a more detailed breakout of pieces



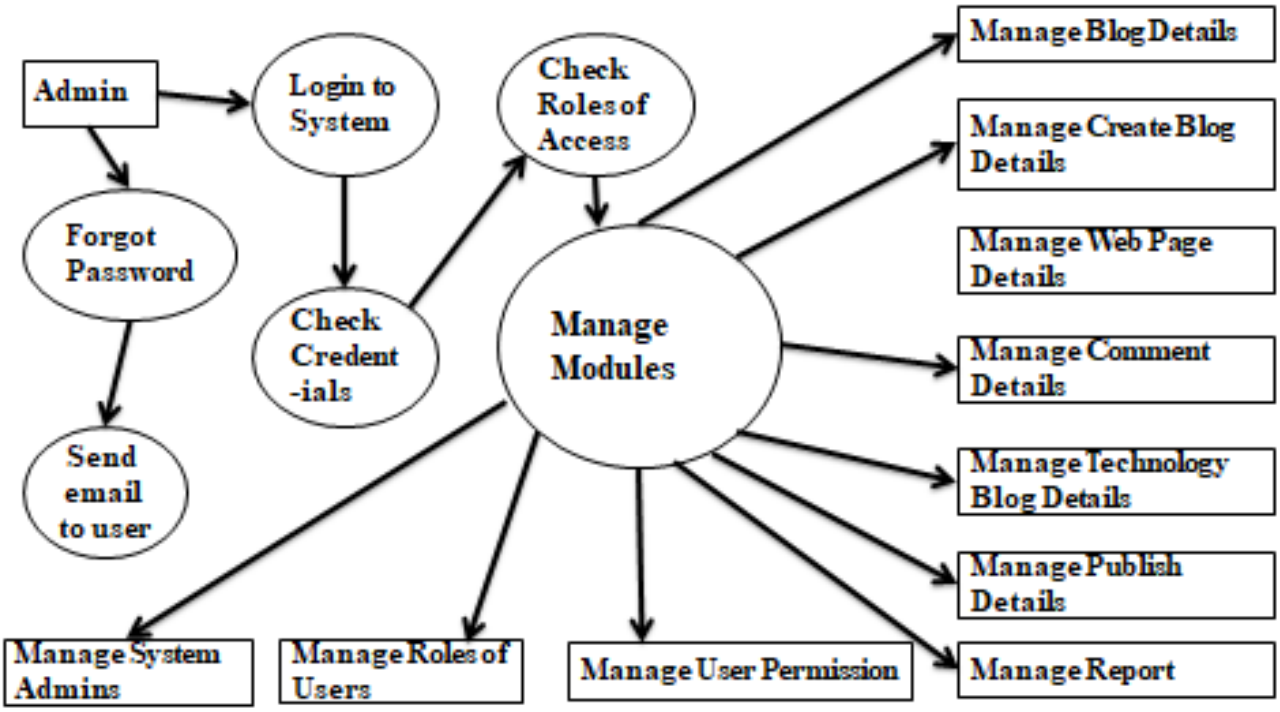
<Project Name>

Main entities and output of First Level DFD

- Processing Blog records and generate report of all Blogs
- Processing Blog Category records and generate report of all Blog Category
- Processing Create Blog records and generate report of all Create Blog
- Processing Blog Type records and generate report of all Blog Type
- Processing Comment records and generate report of all Comments
- Processing Technology Blog records and generate report of all Technology Blog
- Processing Web Page records and generate report of all Web Pages

4.2.3 Second Level DFD of BLOG

- ✓ One step deeper into parts of level 1 of online blogging system
- ✓ Contains more details of Web Pages, Technology Blog, Comment, Blog Type, Create Blog, Blog Category and Blog
- ✓ Requires more functionality of online blogging to dive into the Second Level DFD



Low Level Functionalities of Online Blogging System

- ✓ Admin logins to the system and manage all the functionalities of Online Blogging System
- ✓ Admin can add, edit, delete and view the records of Blog, Create Blog, Comment, Web Page
- ✓ Admin can manage all the details of Blog Category, Blog Type, Technology Blog
- ✓ Admin can also generate reports of Blog, Blog Category, Create Blog, Blog Type, Comment, Technology Blog
- ✓ Admin can search the details of Blog Category, Comment, Technology Blog
- ✓ Admin can apply different level of filters on report of Blog, Blog Type, Comment
- ✓ Admin can track the detailed information of Blog Category, Create Blog, Blog Type, Comment

4.3 Entity Relationship Diagrams (ERD)

Blogging System entity and their attributes:

- ✓ ***Blog Entity:-***
  blog_id, blog_user_id, blog_title, blog_type, blog_content, blog_description
- ✓ ***Blog Category Entity:-***
  blog_category_id, blog_category, blog_category_title, blog_category_type, blog_category_content, blog_category_description
- ✓ ***Create Blog Entity:-***
  blog_id, blog_user_id, blog_title, blog_type, blog_content, blog_description
- ✓ ***Comment Entity:-***
  comment_id, comment_user_id, comment_type, comment_title, comment_description
- ✓ ***Blog Type Entity:-***
  blog_type_id, blog_type_name, blog_type_description

Description of Database

- ✓ The details of Blog is store into the Blog tables respective with all tables.
- ✓ Each entity(Technology Blog, Create Blog, Comment, Blog Category, Blog) contains primary key and unique keys

- ✓ The entity Create Blog, Comment has binded with Blog, Blog Category entities with foreign key
- ✓ There is one-to-one and one one-to-many relationships available between Comment, Blog Type, Technology Blog, Blog
- ✓ All the entities Blog, Comment, Create Blog, Technology, Blog are normalized and reduce duplicacy of records