<Photonest>

System Design

<1.0>

<Date>

<Dilara Ünbay

Nursena Karakulah

Feyzullah Berkay Danış

Özay Ezerceli

>

Prepared for

SE301 Software Engineering



Table of Contents

[1. Introduction 1](#_Toc433996772)

[1.1. Purpose of the System 1](#_Toc433996773)

[1.2. Design Goals 1](#_Toc433996774)

[1.3. Definitions, Acronyms, and Abbreviations 1](#_Toc433996775)

[1.4. References 1](#_Toc433996776)

[2. Current Software Architecture 1](#_Toc433996777)

[3. Proposed Software Architecture 1](#_Toc433996778)

[3.1. Overview 1](#_Toc433996779)

[3.2. System Decomposition 1](#_Toc433996780)

[3.3. Hardware Software Mapping 2](#_Toc433996781)

[3.4. Persistent Data Management 2](#_Toc433996782)

[3.5. Access Control and Security 2](#_Toc433996783)

[3.6. Global Software Control 2](#_Toc433996784)

[3.7. Boundary Conditions 2](#_Toc433996785)

[4. Subsystem Services 2](#_Toc433996786)

[5. References 2](#_Toc433996787)

SYSTEM DESIGN DOCUMENT [1]

The System Design Document (SDD) is written after the initial system decomposition is done and updated throughout the development. SDD describes the services provided by each subsystem. Although this section is usually empty or incomplete in the first versions of the SDD, this section serves as a reference for teams for the boundaries between their subsystems. The interface of each subsystem is derived from this section and detailed in the Object Design Document.

SDD is used to define interfaces between teams of developers and serve as a reference when architecture-level decisions need to be revisited. The audience for the SDD includes the project management, the system architects (i.e., the developers who participate in the system design), and the developers who design and implement each subsystem.

# Introduction

Provide a brief overview of the software architecture and the design goals. It also provides references to other documents and traceability information (e.g., related requirements analysis document, references to existing systems, constraints impacting the software architecture).

## Purpose of the System

Photonest is a social media application program in which users can share photos easily. The main purpose of Photonest is to share photos easily and for this purpose an easy interface is designed for users to use and remember how to use the application easily. Users can share photos, like photos shared by the other users, comment to photos, if they want, they can delete their photos, the comments done by themselves or the comments done by the other users under their posts. Also, users have the function of blocking other users. Overall, they can interact with the other people, the world easy and quick. This interaction will give enjoyable time to users.

## Design Goals

**Usability**

Users could see only other people’s posts if they registered to Photonest and if they are followers of those people.

The users should be able to reach their profiles with one click.

The users should be able to use the system without any training.

The system should be easy to navigate and understand such that a user should be able to successfully complete their profile within 10 minutes on their first time.

**Performance**

Users will be able to take feedback within a maximum of 10 seconds when they share a post, they register or login to Photonest.

The system should be able to create a new user profile within 3 seconds.

The feed page load time of the system should be no more than 6 seconds.

The system should be available 95% of the time.

The system should log in a user within 10 seconds.

The system should not lose any user data.

**Reliability**

Users’ password is not displayable for everyone.

The system should be able to back-up all data automatically every day.

When the system fails to share a post, the user should be able to retry sharing it without creating a new post.

The system should be able to prevent restarting on failures 85% of the time.

**Supportability**

The system should support the English language.

The system should be able to support 5% of concurrent usage without failing to satisfy the performance requirements.

The database of the system should be able to support 15% of growth without losing any performance.

**Implementation**

All related software associated with Photonest, will be written using Java. For the database management, Firebase Realtime Database will be used.

The system should be written in Android Studio.

The system should run on at least 87% of Android devices.

## Definitions, Acronyms, and Abbreviations

Photonest is designed with using JAVA and Firebase.

***SDD:*** System Design Document

***Photonest:*** An application which users can interact with each other by sharing photos, adding comment.

***Visitor:*** A user who is not registered into system.

***Java:*** Java is a programming language.

***Firebase Realtime Database:*** A cloud-hosted database provided by Google which allows to store and sync data.

***Subsystem****:* Subsystem is a collection of classes of the system that are closely related to each other.

***Post:*** An object that consists of an image, description in the form of text and a location selected from map.

## References

[www.instagram.com/](http://www.instagram.com/)

<https://tr.pinterest.com/>

# Current Software Architecture



Across the world, there are many different social media applications with different main purposes. These applications serve many functions, from sharing photos, videos, texts etc. An example of these applications is instagram.com with the nearly same main function as Photonest. This current system is based on MVC architecture and subsystems inside of these defined three subsystem such Model, View, Controller subsystems. The Model subsystems include only application data, such as user, photos, comment etc. and it does not describe how to present the data to a user. The View subsystems present the model’s data to the user. How to access the model’s data is under the View. The Controller subsystems are between the View and the Model. It listens to events triggered by the View and executes reaction.

The applying MVC architecture in Android application is slightly different from above explanation. The Activities, Fragments and Views should be the Views in the MVC world for android. The Controller subsystems should be separate classes which do not extend or use and Android class. The problem is about connection the Controller to the View. In the basic MVC model, the Controller needs a reference to the View. To do this in Android, is to have a BaseView interface which the Activity/Fragment/View would extend. Thus, the Controller would have a reference to the BaseView.

# Proposed Software Architecture

The proposed software architecture in Photonest is applying of MVC model to the Android. The Activities, Fragments and Views will be the Views. This architecture does not increase the testability of the code, but it makes it easier to extend and allowing an easy implementation of new features. Because of the Model classes do not have any reference to Android classes, the Model classes will be straightforward to unit test. Besides of this, the Controller does not extend or implement Android classes and it should have a reference to a View interface, by the way, unit testing is also possible for the Controller.

## Overview

MVC(Model-View-Controller) architecture are used for decomposing the system. There are 3 main subsystems: The Model Subsystems, The View Subsystems and The Controller Subsystems. The View Subsystems describes how the data presented to users. The Model Subsystems include data of application only. The View Subsystems execute reactions according to the events coming from View subsystems’ services.

System Decomposition, Hardware Software Mapping, Persistent Data Management, Access Control Security, Global Software Control and Boundary Conditions of the system will handle in the next parts.

## System Decomposition

To decompose the system, MVC(Model-View-Controller) as architectural style. Decomposed system has 3 level: Model Subsystems, View Subsystem and Controller Subsystems.

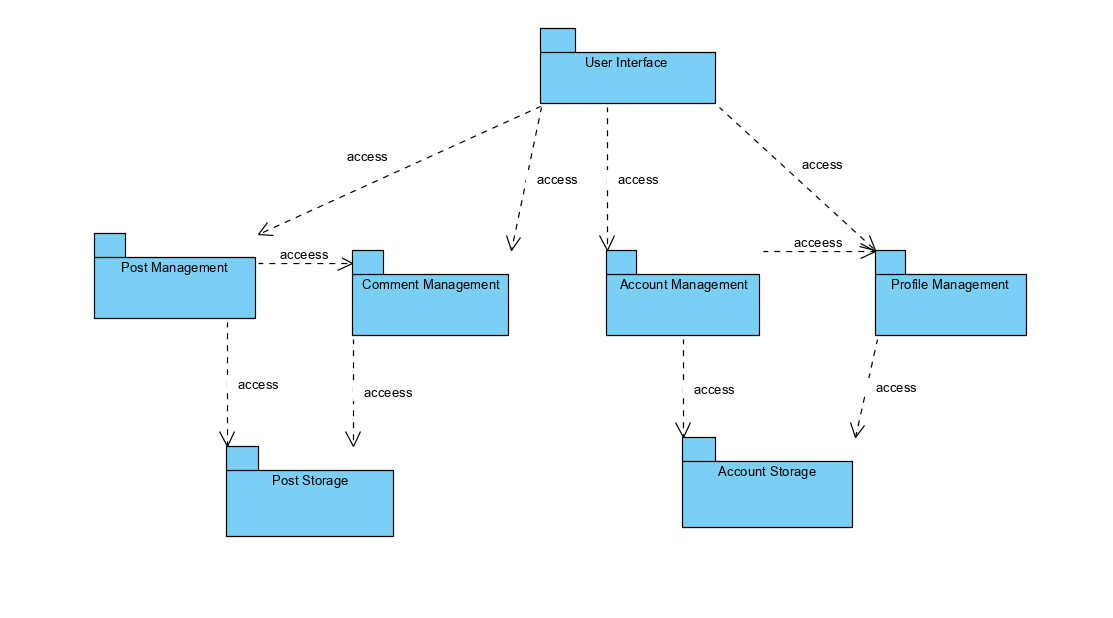
Model Subsystems compose the entity objects of the system. Model Subsystems are responsible of keeping the data of posts, such as location of the post, hashtags, number of like etc., and data of accounts.

* Post Storage subsystem, in addition to keeping the information of posts, it also includes the data about their comments.
* Account Storage subsystem is responsible of keeping data of accounts, such as username, email, password-are kept as encrypted-, full name, their posts etc.

View Subsystem is responsible to show reactions coming from Controller Subsystems by the user interface. Besides of this, it is responsible of presenting the model’s data to the user. To be listened to by the Controller Subsystems, it creates events triggered by users.

Controller Subsystems includes Post Management, Profile Management, Account Management and Comment Management components. Controller subsystems mainly are responsible of managing the data or event coming from the user. It is a bridge between the Model Subsystems and the View Subsystems.

* Post Management process requests coming by users and manages the user data through the Model Subsystems.
* Comment Management also related with the Post Management. The processes made on the comments by users are managed by the Comment Management again through the Model Subsystems.
* Account Management handles authentication processes by sending the data coming from user through the View Subsystem, and receiving requests coming from the Model Subsystems.
* Profile Management is related with the Account Management. It processes the data related with the users by the Model Subsystems.



## Hardware Software Mapping

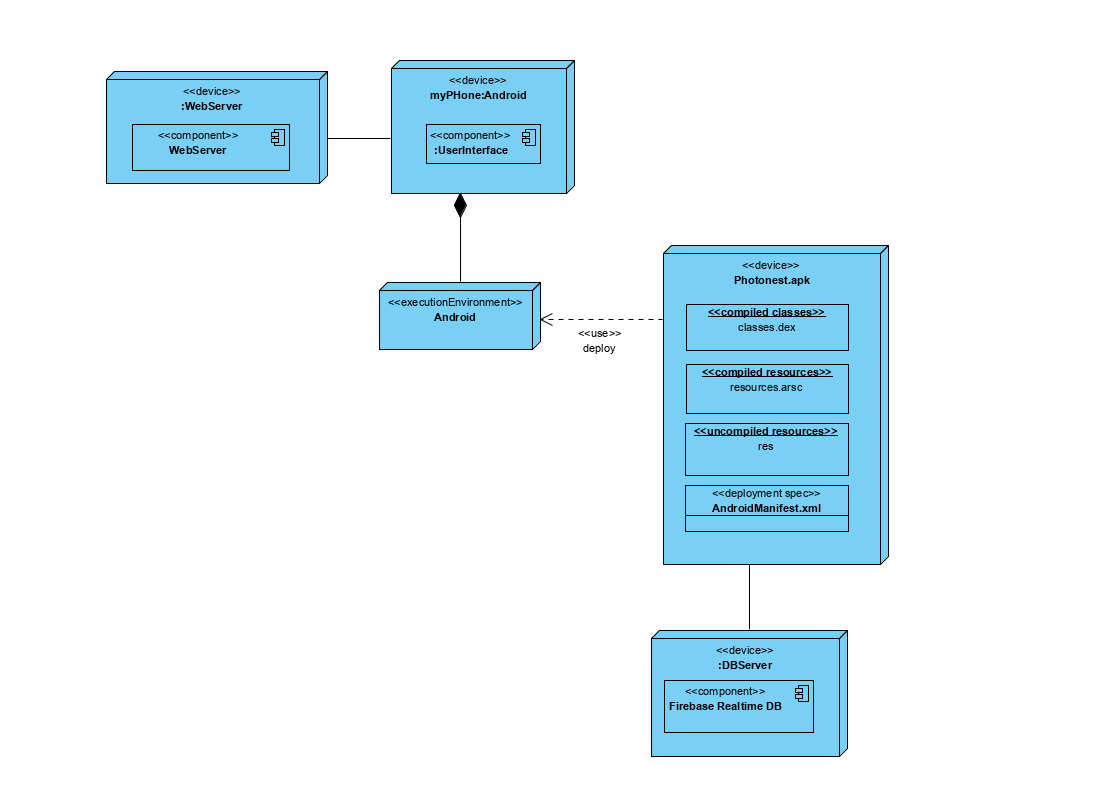
The .apk file, Photonest.apk, represents Android application to be deployed to the Android mobile device, myPhone.

The AndroidManifest.xml file defines some requirements, for example the minimum version of Android required to deploy the application, hardware components etc.

myPhone: Android layer corresponds to the View layer, UserInterface component are responsible of processing user requests.

Photonest.apk layer corresponds to the Controller layer, it deployed execution environment specified by Android, receives the requests coming from View layer, myPhone: Android, and to update the data at the Model layer handles with the data.

As the Database Management System, Firebase Realtime DB is used, the DBServer layer corresponds to that. It performs as the Model layer in where the data received from the Controller by the View, updates.



## Persistent Data Management

Describe the persistent data stored by the system and the data management infrastructure required for it. This section typically includes the description of **data schemes, the selection of a database, and the description of the encapsulation of the database**.

## Access Control and Security

Describe the user model of the system in terms of an access matrix. This section also describes security issues, such as the selection of an authentication mechanism, the use of encryption, and the management of keys.

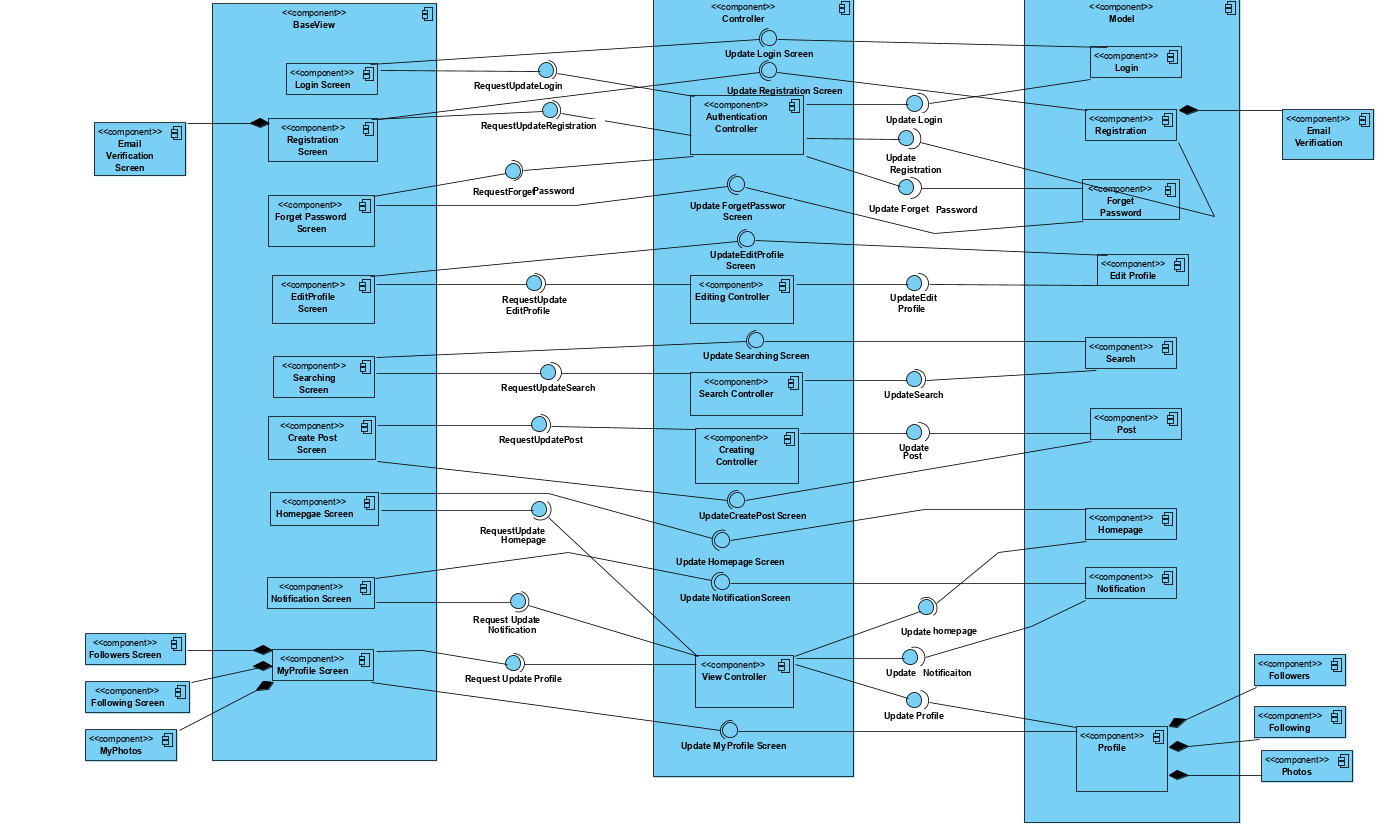
## Global Software Control

## Boundary Conditions

Describe the start-up, shutdown, and error behavior of the system. (If new use cases are discovered for system administration, these should be included in the requirements analysis document, not in this section.)

# Subsystem Services

Describe the **services provided by each subsystem**. Although this section is usually empty or incomplete in the first versions of the SDD, this section serves as a reference for teams for the boundaries between their subsystems. The interface of each subsystem is derived from this section and detailed in the Object Design Document.



# References

1. Bruegge B. & Dutoit A.H. (2010). Object-Oriented Software Engineering Using UML, Patterns, and Java, Prentice Hall, 3rd ed.
2. “Gnu.org.” The GNU General Public License v3.0 - GNU Project - Free Software Foundation, www.gnu.org/licenses/gpl-3.0.en.html.
3. <https://www.tutorialspoint.com/design_pattern/observer_pattern.htm>
4. Android Architecture Patterns Part:1 Model-Viwe-Controller, Florina Muntenescu,2016

<https://medium.com/upday-devs/android-architecture-patterns-part-1-model-view-controller-3baecef5f2b6>