Developer Contacts

System Design

Version 1.0

15.12.2019

Batuhan Kurt

216SE2016L

Prepared for COMP4902

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 2](#_Toc433996775)

[1.4. References 3](#_Toc433996776)

[2. Current Software Architecture 3](#_Toc433996777)

[3. Proposed Software Architecture 4](#_Toc433996778)

[3.1. Overview 5](#_Toc433996779)

[3.2. System Decomposition 8](#_Toc433996780)

[3.3. Hardware Software Mapping 9](#_Toc433996781)

[3.4. Persistent Data Management 10](#_Toc433996782)

[3.5. Access Control and Security 11](#_Toc433996783)

[3.6. Global Software Control 12](#_Toc433996784)

[3.7. Boundary Conditions 14](#_Toc433996785)

[4. Subsystem Services 15](#_Toc433996786)

[5. References 18](#_Toc433996787)

SYSTEM DESIGN DOCUMENT

# Introduction

My Developer Contacts system design is quite simple, understandable but very functional. This web-site provides a platform where users can access anywhere and can share anything with each other. Developers can easily find solutions and answers, according to their projects, also they can share their CV’s and find a job that related to their specifications with my functional system design. After the design stage, I can decrease the time required the implementation.

## Purpose of the System

As I said in RAD, the main aim of this project is to find ideal solutions to developers projects, also they can share their information and development skills to find a job that related to their specifications. This website, which contains several profiles of developers, can provide blogs and useful comments to developers. Developer can create an account, a profile and blogs. In addition, they can like or dislike their comments. Other specification of this project is that developers can look at other developer’s github repositories, or their social media accounts such as facebook, twitter, linkedin and youtube. Also, a developer can add software skills to his/her own profile. As a result, this system helps the users to achieve correct answer to their project and add a contact from software world.

## Design Goals

The purpose of my project is to fulfill the requirements of software engineering completely. It is to specify all functional and nonfunctional functions together. With the definition of the functions, I have defined all the requirements for my Developer Contacts project and have prepared an infrastructure for future versions or new projects. In my Developer Contacts system design, I provide to my users or my visitors easy access to my website, to find new jobs, to hear about updated news from software world, to reach correct information and to create groups for cooperation with each others. The features my system evaluates based on non-functional functions are as follows:

* **Dependability**

One of the most significant non-functional requirement is system security. The user security is at forefront in my Developer Contacts system. In addition to security and safety, I paid attention to robustness, reliability, availability and fault tolerance criteria to make a complex system.

* **Maintenance**

The maintenance of my Developer Contacts system is periodically performed by the system administrator. Of course, while I am creating to my system, shortcomings such as extensibility, modifiability, adaptability, portability and readability were taken into consideration.

* **End User Criteria**

On my Developer Contacts website, users and visitors can sign up, sign in, view their CV’s or other information, and create groups for cooperation. My system efficiently stores and retrieves user data in a dynamic manner. In addition, I noticed that utility and usability factors are important for me. The Developers system supports Microsoft and MacOS operating systems because it is a browser based application.

* **Performance**

My Developer Contacts system is responsive and it can accomplish a maximum number of tasks easily. The memory space of my system is available for speed optimizations. As I mentioned the response time, through memory criteria are significant for my system.

* **Cost**

I try to accomplish optimal level for cost of my system while I am developing it. In addition, this cost is not only for design considerations, but also executive as well. To maintain backward compatibility with a previous system, I can add to the development cost while reducing the transition cost. By the way, I handle the development cost, deployment cost, updating costs, maintenance cost and administration cost.

## Definitions, Acronyms, and Abbreviations

The abbreviations and definitions contained in the document are given below:

* Developer Contacts: Contacts of a developer that are added with thanks to my application
* Admin: Developer Contacts system admin
* User: Role of a developer
* Model: A schematic description of a system that accounts for its known or inferred properties.
* System: Any interacts by the application are considered to be done by the system
* Efficiency: The properties of an algorithm, which is the amount of computational resources used by the algorithm
* Service: Service is a keyword. Purpose of the service is to provide portfolio sharing among to developers
* Bottleneck: The capacity of an application severely limited due to having a single component crowded out.
* GUI: Graphical User Interface
* Interface: The entity allowing for the user to communicate with the system
* Resources: The entities that support the system which are limited e.g. space, data, time
* OOP: Object Oriented Programming
* POP: Procedural or Produce Oriented Programming
* MongoDB: Mongo Database
* HTML: Hypertext Markup Language
* CSS: Cascading Style Sheets
* MIT License: Massachusetts Institute of Technology License
* SDD: System Design Document.

## References

Requirements Analysis Document (15.11.2019)

<https://github.com/facebook/react>

<https://mlab.com/>

<https://github.com/expressjs/express>

<https://github.com/topics/mern-stack>

# Current Software Architecture

My Developer Contacts system is an intelligent and agile system, I have designed to my system on the web. My system differs from other web-sites, other sites may help to developers with a solution, however they do not provide CV sharing or showing github repositories or do not present details of social media accounts at the same time.

Other difference of my system is that developers can create very detailed portfolio, they can add their software skills and create a group with their contacts. The communication between the system administrators and users (developers) works in a synchronous manner. The platform is running synchronously, the host and system administrator are very quick to inform. As a result, I used the MVC (Model, View, Controller) architecture style for my Developer Contacts system. Because, MVC is well suited for interactive systems, especially when multiple views of the same model are needed.

In addition, I was implemented with the client-server architecture to this application. In this system, there is a client which is the app, and there is a server. The application makes API calls to this server and returns the results back to server. At the end of all this, there exists a database which is on the cloud. The database retrieves any user related data. The client sends request to the server and the server returns results.

# Proposed Software Architecture

Documents the requirements elicitation and the analysis model of the new system. The Developer Contacts system is web based. My project will be very useful for developers who are looking for a new jobs or projects.

I am going to use the MVC architecture for the new system. The Model View Controller (MVC) design pattern requires that an application is made up of a data model, presentation information, and control information. Each of these need to be separated into different models. The MVC mostly associates to the UI / interaction layer of an application.

I still need to have a business logic layer, and some service layer and data access layer. The model contains only the pure application data, it contains no logic describing how to present the data to a user.

The View displays the model’s data to the user. The view knows how to access the model’s data, but it does not know what this data means or what the user (developer) can do to change it. The controller exists between the view and the model. It listens to events triggered by the view and executes the appropriate response to the events. In most cases, the response is to call a method on the model. So, the view and the model are connected through a notification mechanism, the result of this action is then automatically displayed in the view.

## Overview

In my Developer Contacts system, I have designed subsystems based on software engineering requirements, working more efficiently, rapidly and working together. This provides me a coherence. I divided to my system some subsystems that are; create groups interface, developer interface, create portfolio interface, create post interface, the web-site main page, the main-page without sign in home page interfaces.

MVC stands for Model, View and Controller. MVC separates application into three components ; Model, View and Controller. It is well suited of applications with a user interface which is my developer interface.

Model maintains the data of an application, it also represents shape of the data and business logic. Model objects are responsible for retrieving and storing model state in a database.

The view represents the user interface. It is where commands are taken and triggers events. It views the display data to the user also allows them to modify the data.

Controller serves as the link between the client and the system. Client requests are handled by the controller. Usually, client interacts with the View which will then create appropriate URL request, this request is handled by a controller.

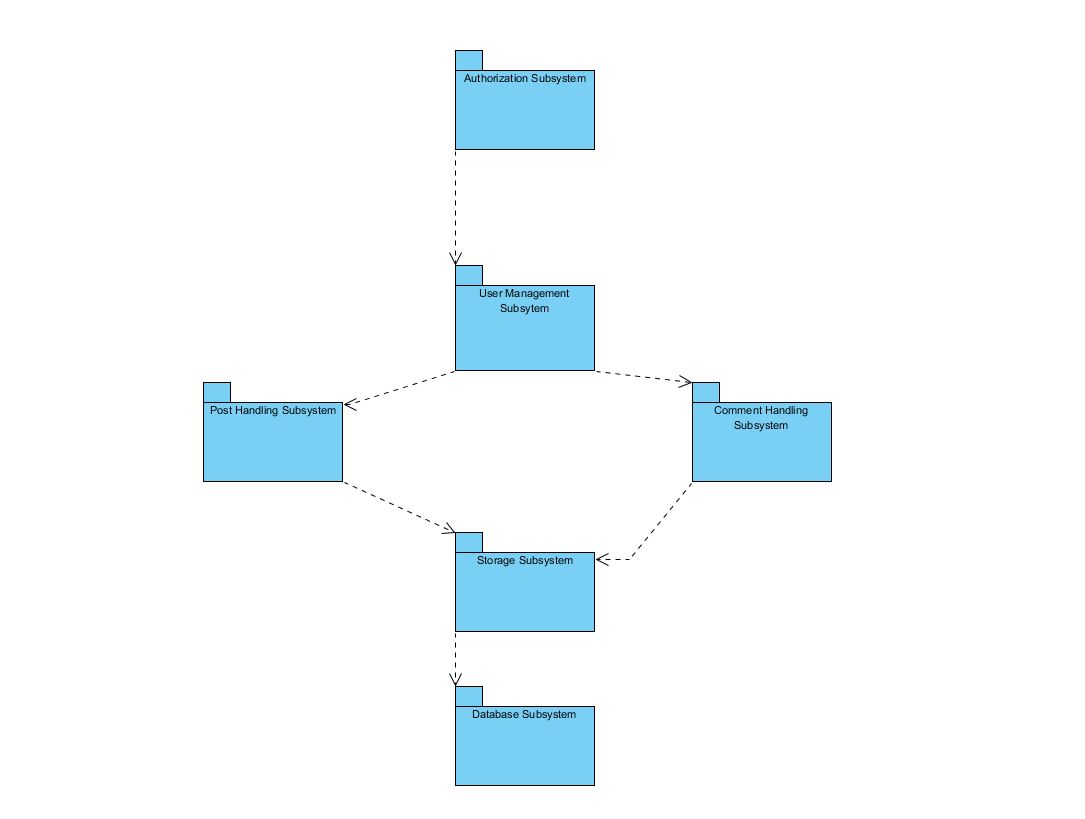
Advantages of using an MVC include:

* Ability to provide multiple views
* Modification doesn’t affect the entire system
* Faster development process
* Support for asynchronous technique

Disadvantages of using an MVC include:

* Increased complexity
* Performance bottleneck
* Inefficiency data access in view.

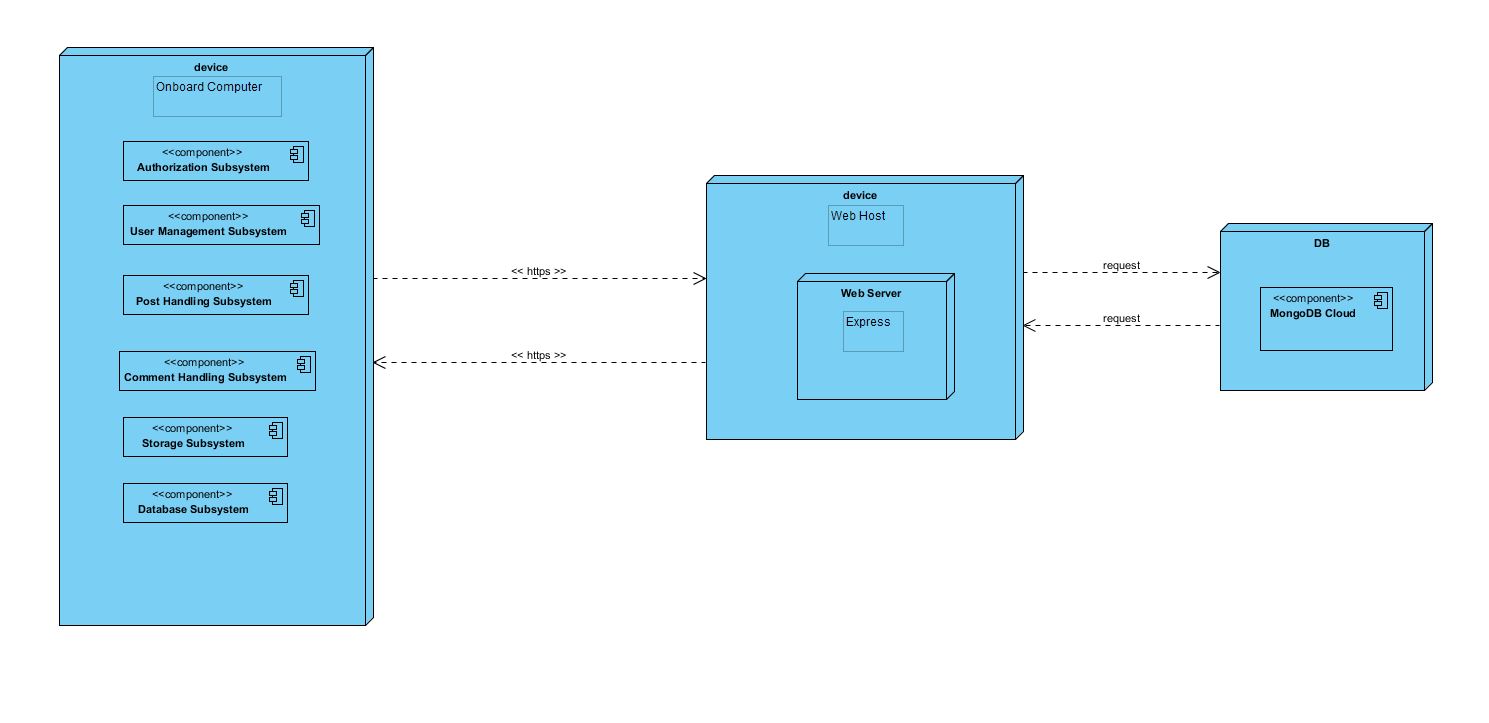
## System Decomposition



I have six subsystems in my system. Each one has a unique set of features and functionalities. I achieved this by decomposing my domain model and using the object model. Then, I filtered which were most crucial in achieving the tasks my system requires. My subsystems are as follows:

* *Authentication Subsystem:* This subsystem is fully responsible for user login and register. Authorized persons can access the system through this subsystem. Developers can register with the system through this system.
* *User Management Subsystem:* This subsystem manages user (developer) interactions with the system. It is responsible for allowing the user to create profile, edit profile, delete profile and share portfolio. It is also responsible for adding experience information, deleting experience information, adding education information, deleting education information and creating private developer groups.
* *Post Handling Subsystem:* This subsystem manages user (developer) posts interactions with the system. It is responsible for allowing the user to add post, delete post, like post and dislike post.
* *Comment Handling Subsystem:* This subsystem manages user (developer) comments interactions with the system. It is responsible for allowing the user to add comment, delete comment, like comment and dislike comment.
* *Database Subsystem:* This subsystem is responsible for storing all my data. It is setup in such a way that a storage accesses the database and all other subsystems contact the storage for any modifications or additions to data assets. This is done in an effort to reduce dependency on the database and for security breach issues.
* *Storage Subsystem:* This subsystem is the bridge between the other subsystems and the database. If any subsystem wants to access database for data retrieval or update, the storage provides the service and then it accesses the database itself.

## Hardware Software Mapping



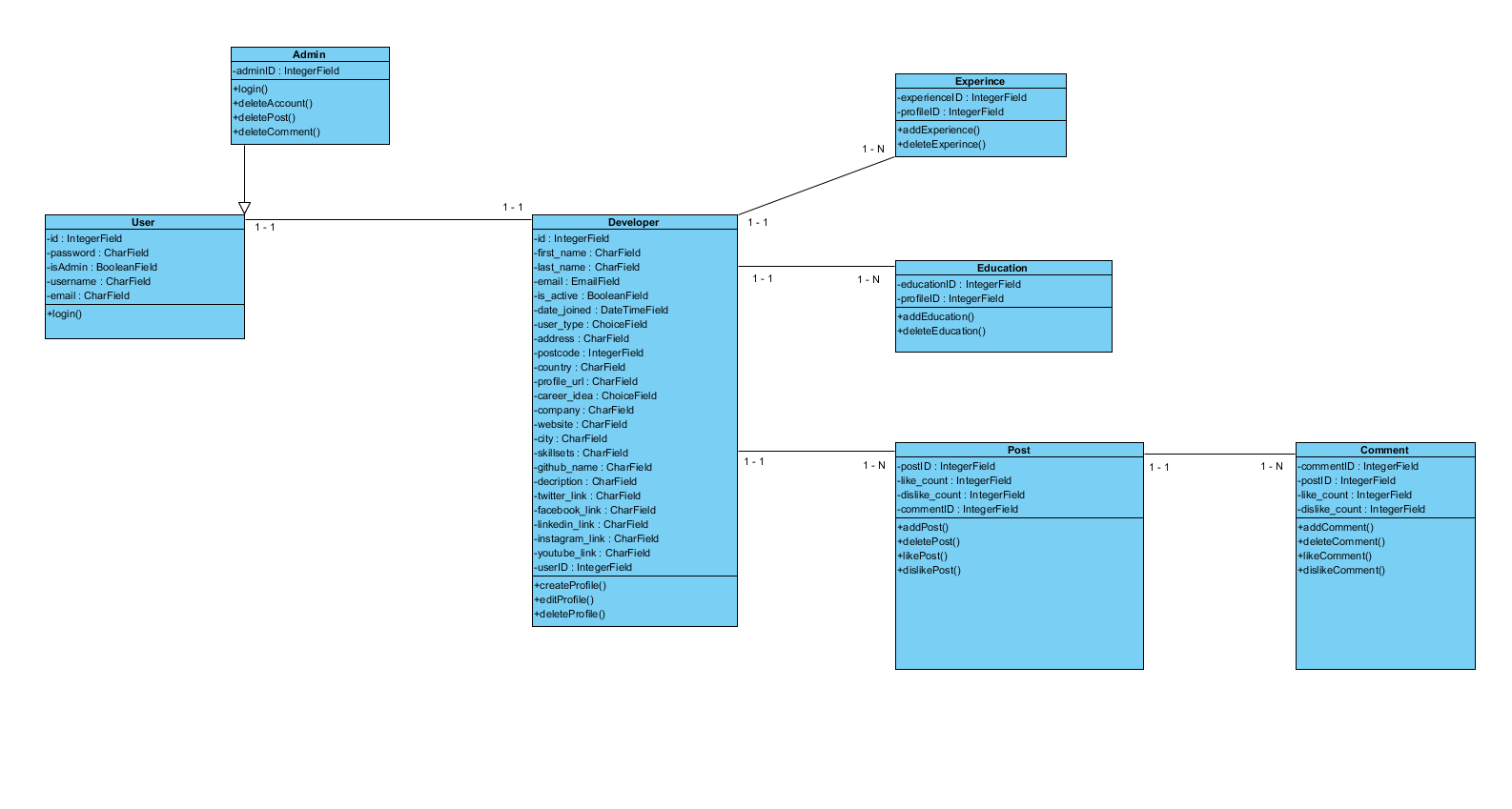
I described how subsystems are assigned to hardware and off-the-shelf components. It also lists the issues introduced by multiple nodes and software reuse.

Developer Contacts is a web-based online cooperation system. Developer Contacts is connecting to web server when user (developer) tries to visit and login or register to the system. Developer Contacts system connects to web host with http protocol. Multiple users can access web server simultaneously. Web host have web server. The system connects to database by using JavaScript (Express.js) library.

The Users who is a developer can create portfolio with using the Developer Contacts system. These users communicate with Developer Contacts system with using https protocol in their personal computers web browser. In addition, Developer Contacts uses MongoDB database. This database is running on the cloud. MongoDB is a non-relational database management system. MongoDB is a popular choice for local / client storage applications such as web browsers. It is arguably one of the most preferred deployed database engine, as it is used today by several widespread browsers, operating systems, and embedded systems.

In addition, for the User Interface, I used React.js because it is very popular, agile and open source JavaScript library.

## Persistent Data Management

My system uses the Express and Mongoose libraries. These frameworks help to my system to communicate to the MongoDB database. This will allow the database to be easily integrated with and accessed by the rest of the system. The database will retain user (developer) information for functions such as viewing career information of a developer. My database structure is seen below with entity field’s relations etc.

## Access Control and Security

My system is a multiuser application so it consists of 2 types of users which are developer and administrator. Because of this, the application will provide different interfaces for each user type.

First, the administrator can connect to the system with the membership interface, and will do the administrator's duties, such as adding, updating, editing, deleting etc. with using MongoDB admin interface. As a summary, the administrator does not have to register because he/she is already registered in the Mongo DB database.

The system will store all the information in the database. Then, it will use them by collecting data from the database. The information in the database will use for the confirmation of user types. All types of users must log in to the system with their username and password.

During registration, field filling does not require access to the database, while completion of the process requires the data to be written to the database, which requires read and write access to the database. In that case, the required database fields will be blocked and simultaneous access of multiple users will be denied.

For some situation like updating or deleting an information, it is necessary to update one of the tables in the database in its phase of completion. Therefore, this case must be handled with more care since, several users can be the cause of updating the table at the same time. This will also be avoided by blocking.

Finally, viewing the information or lists again requires read-only access to the database. Therefore, multi-user access does not impose problems and new restrictions.

As a result, the usernames and passwords of users (developers) will be stored in the user table. No one else except the administrator can have access to this information. Authentication interfaces are different for each type of user and will be directed to their own main pages after the login process.

|  |  |  |  |
| --- | --- | --- | --- |
| Actors/  Classes | Admin | Developer | DB Connection |
| Admin | deleteAccount()  deleteComment()  deletePost()  deleteGroups()  viewProfile()  deleteProfile() |  | login()  logout() |
| Developer |  | register()  createProfile()  createGroup()  post()  comment()  like()  dislike()  deleteAccount()  deletePost()  deleteComment()  addExperince()  addEducationInformation() | login()  logout() |

## Global Software Control

My system has MVC (Model - View - Controller) software architecture. Developer Contacts is thread safety but also multithreaded program either because my system must provide many users at the same time to cooperation among the developers.

I have decided to go with event-driven control due to the requirements and nature of my system. The sequencing of actions in my system are directed by an external factor or event generated by an actor to achieve a goal. This makes change in my control structure easy to change and well suited for my system. Because many users can access change and access data in the system and resources are shared, this may creates concurrency issues.

## Boundary Conditions

**Initialization**

Startup: Go to system URL and login

Shut Down: Click log out and close browser

**System Failure**

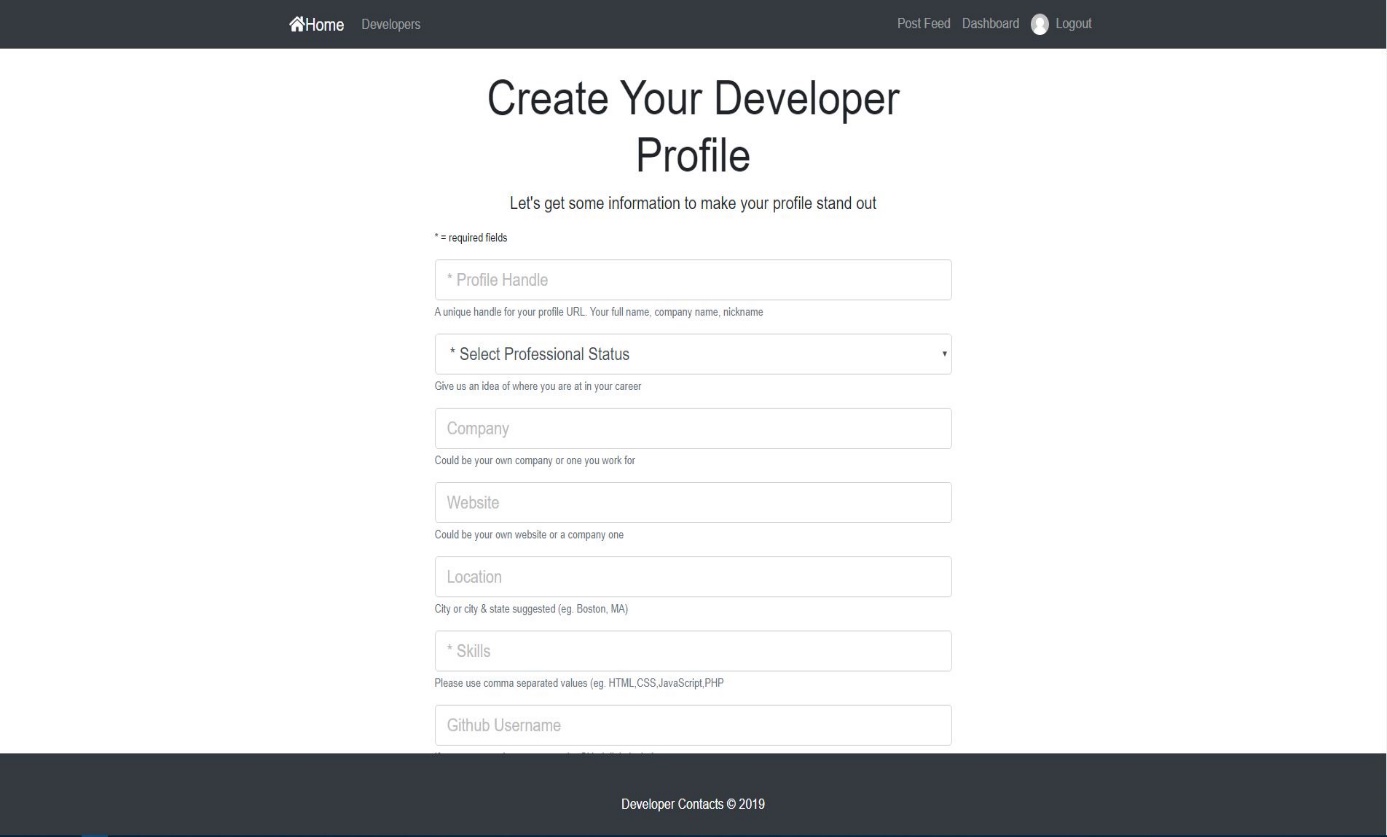
* If database connection is lost, there is a reconnect mechanism which attempts to reestablish the connection to the database.
* Checkpoint system can be applied to prevent database connection problem. System is connected to database and required data is written regularly. If a problem occurs, system will be returned to its previous errorless state.

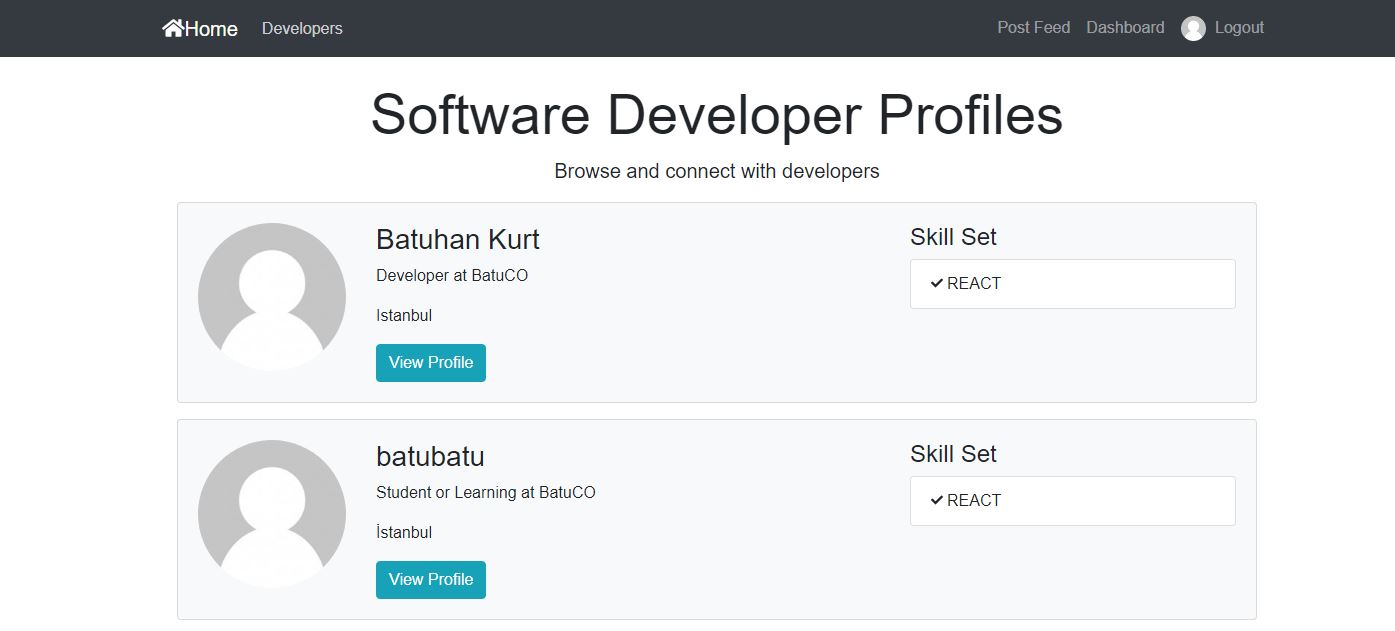
**Error Conditions**

* Logging in:
* Username or password field is blank.
* Password is not 6 characters long or more.
* Password and username don’t match.
* Username is wrong or does not exist.
* Welcome screen does not appear after logging in.
* Developers Data
* Developer information can not be not exist.
* Developers information can not be blank.

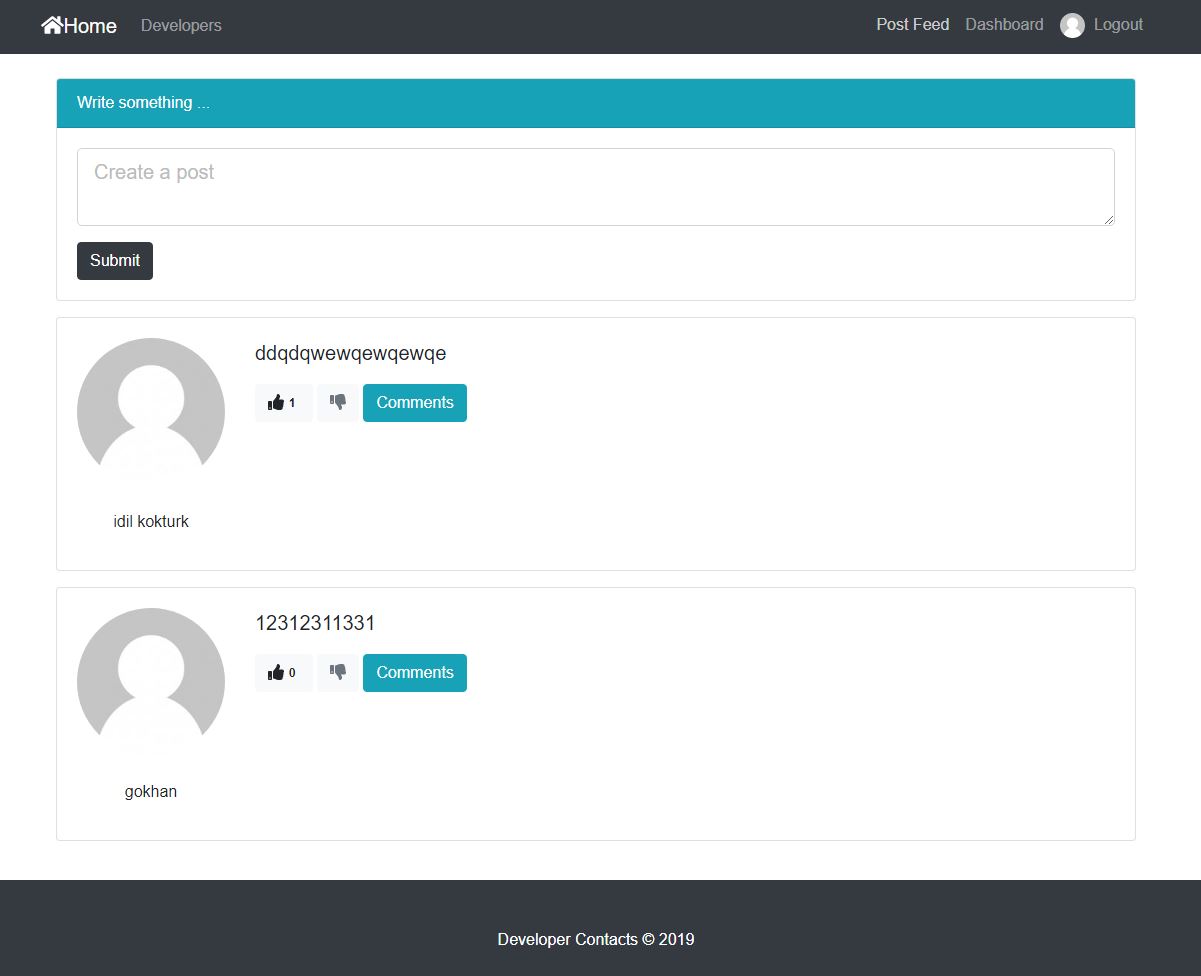
# Subsystem Services

I preferred to divide my system to subsystems to adapt each other themselves and, provide the whole system functionalities.

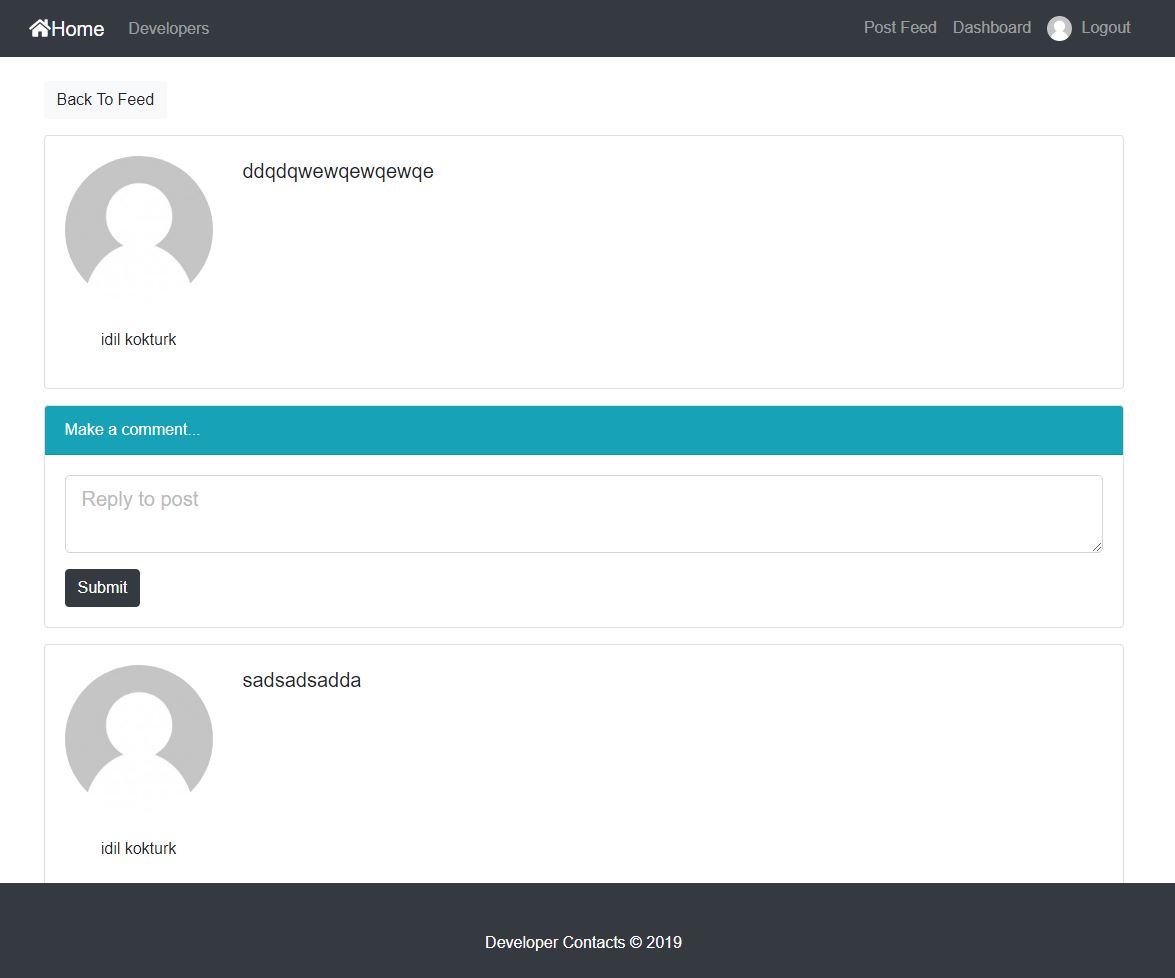
* **Create Developer Profile Interface**
* **View Developers Interface**



* **Posts and Like & Dislike Interface**



* **Comments Interface**



# References

1. Bruegge B. & Dutoit A.H.. (2010). *Object-Oriented Software Engineering Using UML, Patterns, and Java*, Prentice Hall, 3rd ed.
2. <https://www.uml-diagrams.org/composite-structure-diagrams.html>
3. https://www.geeksforgeeks.org/mvc-design-pattern/
4. Enes Taylan, Huseyin Guler, Alperen Eraslan, Omer Durmus (2009). *Billiard Project, Object Oriented Software Engineering.*
5. <http://www.tutorialsteacher.com/mvc/mvc-architecture>
6. <https://mindmajix.com/react-js-tutorial#component-api>