Developer Contacts

Requirements Specification and Analysis

Version 1.0

17.11.2019

Batuhan Kurt

216SE2016L

Prepared for COMP4902

Software Engineering



Table of Contents

[1. Introduction 1](#_Toc498723792)

[1.1. Purpose of the System 1](#_Toc498723793)

[1.2. Scope of the System 1](#_Toc498723794)

[1.3. Objectives and Success Criteria of the Project 1](#_Toc498723795)

[1.4. Definitions, Acronyms, and Abbreviations 1](#_Toc498723796)

[1.5. Overview 2](#_Toc498723797)

[2. Current System 2](#_Toc498723798)

[3. Proposed System 2](#_Toc498723799)

[3.1. Overview 3](#_Toc498723800)

[3.2. Functional Requirements 3](#_Toc498723801)

[3.3. Nonfunctional Requirements 6](#_Toc498723802)

[Usability: 6](#_Toc498723803)

[Reliability: 6](#_Toc498723804)

[Performance: 6](#_Toc498723805)

[Supportability: 6](#_Toc498723806)

[Implementation: 6](#_Toc498723807)

[Interface: 7](#_Toc498723808)

[Packaging: 7](#_Toc498723809)

[Legal: 7](#_Toc498723810)

[3.4. System Models 7](#_Toc498723811)

[Scenarios 7](#_Toc498723812)

[Use Case Model 15](#_Toc498723813)

[Object model 29](#_Toc498723814)

[Dynamic model 30](#_Toc498723815)

[3.5. Project Schedule 32](#_Toc498723816)

[4. Glossary 38](#_Toc498723817)

[5. References 39](#_Toc498723818)

REQUIREMENTS ANALYSIS DOCUMENT

# Introduction

## Purpose of the System

This project is a cooperation platform where users can access anywhere and can share anything with each other. The main purpose of this project is to find solutions and answers to their projects, also they can share their CV’s and 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.

## Scope of the System

My system is called Developer Contacts. Developer Contacts can be used anywhere and anytime as it is a web-based application. It so simple to use, register to the system, create your profile and ask questions that are related to your project, instantly. This web-site is also a good information source for developers. Any person can register my application. This system collects the people who want to find solutions to his/her project or help to other people who are in the same roof.

## Objectives and Success Criteria of the Project

The success of the system depends on providing the given main set of arguments:

* Use GitHub repository effectively.
* The design of register and login systems.
* Association between database and system design.
* Make the general design of system in efficient way.
* The demo of system should ensure good success rate.
* Implementation should be understandable, clear, basic and efficient.

## 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: 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

## Overview

This document contains sections;

* Introduction
* Current System
* Functional Requirements
* Non-Functional Requirements
* Scenarios
* Use Case Model
* Object Model
* Dynamic Model
* User Interface
* Glossary
* References

In introduction, I have described a good explanation of scope purpose, criteria, definitions and other explanations which help other people who want to read document. In proposed system, I have described the advantages of Developer Contacts Application and what makes Developer Contacts Application to unique. In functional requirement, I have described that the functions are open to end users. Functional requirements states for explaining scope of the system.

In non-functional requirement section, I have described our system’s reliability, performance and others to make the system accessible to end users. In Scenarios section, I have described use case’s descriptions and explanations. In Use case model, I have stated or use case diagram. In Object model, I have described the programming objects which will implemented later on. In Dynamic model, I have described the flow of system in time with operations and function of actors. In glossary section, I have described dictionary and unknown words in RAD.

# Current System

My system is a very agile and useful system and, I designed the system for the web browsers. My system differs from other 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 platform is running synchronously, the host and system administrator are very quick to inform.

# Proposed System

Documents the requirements elicitation and the analysis model of the new system.

## Overview

Developer Contacts is a web site application that I make to benefit to people who wants to share information and help to the software developers. My system aims to help these people to find solutions and answers easily, for their projects. The Developer Contacts system interface has been designed so that it is understandable to everybody. All the portfolios, groups, posts and comments are in the system in a way that the user can distinguish them with the recognized images, and all the features of the system, and the descriptions are understandable.

Thanks to Developer Contacts, users can find the developer to hire for his/her projects or they can give a job to a developer who registered to my system. They can also share projects or social media accounts and, share the github repositories to other people. In addition, developers can send feedbacks with liking or disliking a post, so they can reach to correct answer as a result of discussion that exist on the posts page. Positive or negative comments are briefly evaluated by our system administrator. Obviously the Developer Contacts system has thought of everything it can do to please its developers. Developers will be at a higher level of satisfaction of obtaining correct information, thanks to my system.

When users enter the Developer Contacts system, they will be able to easily create a portfolio and share comments on the website. They can also show the blogs and comments of the projects that they are interested in. The admin of the Developer Contacts system can see the blogs, comments and portfolios so, he/she can delete the inappropriate entities which are existed in the system. In addition, developers can create private groups, so nobody except the members of this group can see any entry about this private group.

The Developer Contacts website always wants to provide correct and agile answers to developers. As a result, the Developer Contacts system is always designed to help to users and wants to improve developers.

## Functional Requirements

***Developer Functional Requirements***

* **Sign Up**

The developer registers on the system to the web-site.

* **Sign In**

The developer logs in to the system with the username and password from the homepage.

* **Create Group**

A developer can create a private group and other developers can join this group.

* **Create Profiles ( Portfolios )**

A developer can create his/her own profile according to his/her personal information.

* **Add Experience Information**

A developer can add his/her experience information into the profile.

* **Add Education Information**

A developer can add his/her education information into the profile.

* **Add Bio and Skill Set Information**

A developer can add his/her biography information and software skill sets into the profile.

* **Add Github Repository Information**

A developer can add his/her own github repository information into the profile.

* **Add Social Media Accounts Information**

A developer can add his/her social media accounts information such as facebook, twitter, linkedin and youtube into the profile.

* **View and Edit Dashboard**

A developer view to his/her own portfolios, he/she can edit ( insert or delete ) profile information.

* **Delete Account**

A developer can delete his/her profile.

* **Post Feed & Comment**

A developer can post an entry and he/she can comment an entry.

* **Like, Dislike and Delete Comments**

A developer can add like or dislike a comment. Also, he/she can delete his/her own comments.

* **Log Out**

Developer exit the system when they finished the operations on the system.

***Admin Functional Requirements***

* **Sign In;**

Admin part is different from Developer Sign in. Admin should enter the different UML. In this page, admin can login on system with username and password. Password is given by the system database.

* **Delete Accounts**

Admin can delete inappropriate accounts from the system, these accounts do not shown on the application anymore.

* **Delete Groups**

Admin can delete the any group that are created by a developer.

* **Delete Posts & Comments**

Admin can delete inappropriate posts and comments from the system, these entries do not shown on the application anymore.

* **Log Off**

Admin exit the system when they finished the operations on the system.

## Nonfunctional Requirements

Usability: Using the Developer Contacts website is quite simple, there is a design that everyone can understand and a global interface. Our interface is based on portfolios, blogs and comments, so my priority to use our system without any confusion. Developer Contacts continues to work to create a more effective interface.

Reliability: Developer Contacts system has almost all security requirements that is %90.

Performance: Developer Contacts uses Mongo DB as its database system, which has a permanent storage space. So, it reacts quite fast and at the same time about 5000 parallel users can enter my system.

Supportability: The maintenance of the Developer Contacts system is a secure portal that is open only for administrator.

Implementation: The Developer Contacts system is implemented in the JavaScript ( Node.js, Express.js and React.js ) programming language with excellent OOP features using Visual Studio Code and Mongo DB Cloud ( [cloud.mongodb.com](https://cloud.mongodb.com) ) programs and platforms. Developer Contacts uses Mongo DB database system, and Node.js environment in back-end, In-Front-End, Developer Contact uses React.js framework. You can access it via www.developercontacts.com.

Interface: The interface of the Developer Contacts system is designed with using React.js components and required CSS files for interface implementation. In addition, despite the use of Interfaces, third party applications and services used in the Developer Contacts system and disclosed to users may not be used. There are no inherited projects that present services or interfaces.

Packaging:The system packaging will be with future versions.

Legal:Developer Contacts is licensed with MIT License. It is provided "as is" without warranty of any kind, including, but not limited to, marketability, manufacturability, software, certain fitness for purpose, and any infringement of rights. In any unusual circumstances, administrators shall not be liable for any damage or other liability arising out of the author, authors or copyright holders, from a software contract, tort or otherwise.

## System Models

Here, I show the high level functionalities of my system and how the users of my system interact with the system. For the object models I depict our classes using UML diagrams and show the relation between them. In the dynamic model, I showed interactions between objects within a use case using sequence diagrams and showed how state machines are used to show the behaviour of that single object.

### Scenarios

**Scenario 1**

|  |
| --- |
| **Scenario name:** Sign up |
| **Participant actor instances:**  Batuhan: Admin, Burak: Developer |
| **Flow of events:**   * Firstly, if a visitor enters to Developer Contacts web-site. * Then, he wants to register Developer Contacts web site. * He clicks to "Sign Up" button on the main page. * He fills the all fields and clicks Sign Up. * If information are correct, and the passwords that are entered by him matches,   he can access his account.   * Finally, user account is created. |
|  |

**Scenario 2**

|  |
| --- |
| **Scenario name:** Log In |
| |  | | --- | | **Participant actor instances:**  Batuhan: Admin, Burak: Developer | |
| **Flow of events:**   * A visitor enters to Developer Contacts web-site. * Then, if he is a developer, he wants to create a profile or, if he is an admin, he wants to delete an account. * So, He clicks to "Log In" button on the main page. * He fills the username and password fields, then clicks sign in. * If information are correct, he will sign in, but if the password is not correct, he canclicks to "Remember Password" button to reset his password. * Finally, he signs in and if he wants to log out, he clicks to "Log out" button to log on main page. |

**Scenario 3**

|  |
| --- |
| **Scenario name:** View Developers List |
| |  | | --- | | **Participant actor instances:**  Batuhan: Admin, Burak: Developer | |
| **Flow of events:**   * A visitor enters to Developer Contacts web-site. * Then, if he is a developer or an admin, he can see the “Developers” button on the main page. * So, He clicks to "Developers" button on the main page. * He will see the list of developers who registered to Developer Contacts System. * He can look at the name, surname, title and software skills of the registered developers. |

**Scenario 4**

|  |
| --- |
| **Scenario name:** View Developers Profiles |
| |  | | --- | | **Participant actor instances:**  Batuhan: Admin, Burak: Developer | |
| **Flow of events:**   * A visitor enters to Developer Contacts web-site. * Then, if he is a developer or an admin, he can see the “Developers” button on the main page. * So, He clicks to "Developers" button on the main page. * Then, he wants to look at a developer profile. So, he clicks the “View Profile” button which belongs to a developer who registered to system. * Finally, he can see the “profile” page which belongs to a developer. On the profile page, he can see the avatar picture, biography, education, experience, skill sets, github repositories and social media accounts of related developer. |

**Scenario 5**

|  |
| --- |
| **Scenario name:** View Dashboard |
| |  | | --- | | **Participant actor instances:**  Burak: Developer | |
| **Flow of events:**   * A visitor enters to Developer Contacts web-site. * Then, he can see the “Dashboard” button on the main page. * So, He clicks to "Dashboard" button on the main page. * Then, he can see his developer profile. * Also, he can edit his profile with “edit profile” button, add an experience with “add experience” button. |

**Scenario 6**

|  |
| --- |
| **Scenario name:** Dashboard Edit Profile |
| |  | | --- | | **Participant actor instances:**  Burak: Developer | |
| **Flow of events:**   * A visitor enters to Developer Contacts web-site. * Then, he can see the “Dashboard” button on the main page. * So, He clicks to "Dashboard" button on the main page. * Then, he can edit his profile, he clicks the “edit profile” button. * Finally, he can update his information with filling the related field which are on the edit profile page. Also, he can update the social media accounts information on this page. |

**Scenario 7**

|  |
| --- |
| **Scenario name:** Dashboard Add Experience |
| |  | | --- | | **Participant actor instances:**  Burak: Developer | |
| **Flow of events:**   * A visitor enters to Developer Contacts web-site. * Then, he can see the “Dashboard” button on the main page. * So, He clicks to "Dashboard" button on the main page. * Then, he can add an experience to his profile, he clicks the “add experience” button. * He can add an experience with filling the “Company”, “Job Title”, “Location”, “From Date”, “To Date” and “Job Description” fields. * Finally, he can clicks “Submit” button to adding experience. |

**Scenario 8**

|  |
| --- |
| **Scenario name:** Dashboard Add Education |
| |  | | --- | | **Participant actor instances:**  Burak: Developer | |
| **Flow of events:**   * A visitor enters to Developer Contacts web-site. * Then, he can see the “Dashboard” button on the main page. * So, He clicks to "Dashboard" button on the main page. * Then, he can add an education information to his profile, he clicks the “add education” button. * He can add an education information with filling the “School”, “Degree or Certification”, “Field of Study”, “From Date”, “To Date” and “Program Description” fields. * Finally, he can clicks “Submit” button to adding experience. |

**Scenario 9**

|  |
| --- |
| **Scenario name:** Delete My Account |
| |  | | --- | | **Participant actor instances:**  Burak: Developer | |
| **Flow of events:**   * A visitor enters to Developer Contacts web-site. * Then, he can see the “Dashboard” button on the main page. * So, He clicks to "Dashboard" button on the main page. * Then, he can delete his account with clicking the “delete my account” button. * Finally, he can clicks “delete my account” button to delete his account. |

**Scenario 10**

|  |
| --- |
| **Scenario name:** Delete Experience and Education Information |
| |  | | --- | | **Participant actor instances:**  Burak: Developer | |
| **Flow of events:**   * A visitor enters to Developer Contacts web-site. * Then, he can see the “Dashboard” button on the main page. * So, He clicks to "Dashboard" button on the main page. * Then, he can delete his experience information with clicking the “delete experience” ,   and education information with clicking the “delete education” button. |

**Scenario 11**

|  |
| --- |
| **Scenario name:** Post Feed |
| |  | | --- | | **Participant actor instances:**  Burak: Developer | |
| **Flow of events:**   * A visitor enters to Developer Contacts web-site. * Then, he can see the “Post Feed” button on the main page. * So, He clicks to "Post Feed" button on the main page. * Then, he can post anything on this page with filling the post field. * Finally, he clicks the “Submit” button to create post. |
|  |

**Scenario 12**

|  |
| --- |
| **Scenario name:** Post Comment |
| |  | | --- | | **Participant actor instances:**  Burak: Developer | |
| **Flow of events:**   * A visitor enters to Developer Contacts web-site. * Then, he can see the “Post Feed” button on the main page. * So, He clicks to "Post Feed" button on the main page. * Then, he can see the all posts on this page. He can comment to post with choosing a post and clicking “Comments” button. * Finally, he fills the comment field and clicks the “Submit” button to comment. |
|  |

**Scenario 13**

|  |
| --- |
| **Scenario name:** Like Comment |
| |  | | --- | | **Participant actor instances:**  Burak: Developer | |
| **Flow of events:**   * A visitor enters to Developer Contacts web-site. * Then, he can see the “Post Feed” button on the main page. * So, He clicks to "Post Feed" button on the main page. * Then, he can see the all posts on this page. * He can like the comments, if he thinks the post is correct. * Then, he clicks to like image. |
|  |

**Scenario 14**

|  |
| --- |
| **Scenario name:** Dislike Comment |
| |  | | --- | | **Participant actor instances:**  Burak: Developer | |
| **Flow of events:**   * A visitor enters to Developer Contacts web-site. * Then, he can see the “Post Feed” button on the main page. * So, He clicks to "Post Feed" button on the main page. * Then, he can see the all posts on this page. * He can dislike the comments, if he thinks the post is incorrect. * Then, he clicks to dislike image. |
|  |

**Scenario 15**

|  |
| --- |
| **Scenario name:** Delete Comment (Developer) |
| |  | | --- | | **Participant actor instances:**  Burak: Developer | |
| **Flow of events:**   * A visitor enters to Developer Contacts web-site. * Then, he can see the “Post Feed” button on the main page. * So, He clicks to "Post Feed" button on the main page. * Then, he can see the all posts and comments on this page. * He can delete his own comment with clicking to cancel image. |
|  |

**Scenario 16**

|  |
| --- |
| **Scenario name:** Delete Accounts |
| |  | | --- | | **Participant actor instances:**  Batuhan: Admin | |
| **Flow of events:**   * A visitor enters to Developer Contacts web-site. * He log in to the system as an admin. * Then, he can see the “Developers” button on the main page. * So, He clicks to "Developers" button on the main page. * Then, he can delete any account with clicking to “delete account” button. If he thinks this account is inappropriate. * Finally, the deleted account is removed from the system. |

**Scenario 17**

|  |
| --- |
| **Scenario name:** Delete Comments and Posts ( Admin ) |
| |  | | --- | | **Participant actor instances:**  Batuhan: Admin | |
| **Flow of events:**   * A visitor enters to Developer Contacts web-site. * He log in to the system as an admin. * Then, he can see the “Post Feed” button on the main page. * So, He clicks to "Post Feed" button on the main page. * Then, he can delete any post with clicking to “delete post” button or delete any comment with clicking to “delete comment” button. If he thinks this post or comment is inappropriate. * Finally, the deleted post or comment is removed from the system. |

**Scenario 18**

|  |
| --- |
| **Scenario name:** Log Off |
| |  | | --- | | **Participant actor instances:**  Batuhan: Admin, Burak: Developer | |
| **Flow of events:**   * A visitor enters to Developer Contacts web-site, and log in the web-site with his username and password. * Then, he wants to exit Developer Contacts web-site. * He clicks to “Log off” button. * Then system immediately logs him out and redirects to him on the home page. |

### Use Case Model

**Use case 1**

|  |
| --- |
| **Use case name:** Sign Up |
| **Participant actors:** Admin, Developer |
| **Flow of events:**   * Actor enters to the site. * Actor clicks "Sign Up" button. * Site responds by displaying the Sign-up screen. * Actor enters all required fields. * Actor clicks "Sign Up" button. * System inserts the information to the database which is provided by User. * System redirecting actor to login page. |
| **Entry Condition:** Actor visits the URL page of the system. |
| **Exit Condition:** Actor clicks "Sign-up" button and entered information should be correct. |
| **Quality Requirements:**  1. If Actor leaves one or more fields empty, system displays a warning message, like "This area cannot be empty.”  2. The information that user entered is checking and verified by the system in real-time. |

**Use case 2**

|  |
| --- |
| **Use case name:** Log In |
| **Participant actors:** Admin, Developer |
| **Flow of events:**   * Actor goes on sign-in page. * System responds by displaying the login screen of the application. * Actor enters username and password to the login fields. * Actor clicks "Log in" button. * If the username and password match with the username and password which is stored in database. User authenticates. * System fetches the dashboard of user. |
| **Entry Condition:** Actor visits the URL page of the system. |
| **Exit Condition:** Actor clicks "Log In" button and entered information should be correct. |
| **Quality Requirements:**  1. If Actor leaves one or more fields empty, system displays a warning message, like "This area cannot be empty.”  2. If the information’s checked from database are not true, system displays a warning message, like "Wrong username or password, please retry.” |

**Use Case 3**

|  |
| --- |
| **Use Case Name:** Create Developer Profile |
| **Participant actor instances:** Developer |
| **Flow of events:**   * This event starts with Developer’s clicking Dashboard page. * User can create developer profile with clicking “Edit Profile” button. * System shows to related field such as name, surname, career information, city, software skills, additional descriptions. * Developer clicks any field and types into it, then developer clicks to submit button. * System creates the profile in database. |
| **Entry Condition:** Developer should log in to the system. |
| **Exit Condition:** Profile created. |
| **Quality Requirement:**  **1.** System response to the actor arrives in a few seconds. |

**Use Case 4**

|  |
| --- |
| **Use Case Name:** View Developer Lists |
| **Participant actor instances:** Developer, Admin |
| **Flow of events:**   * This event starts with actors enters to Developer Contacts web-site. * Then, if actor is a developer or an admin, actor can see the “Developers” button on the main page, and actor click to "Developers" button on the main page. * Actor will see the list of developers who registered to Developer Contacts System. * Actor can look at the name, surname, title and software skills of the registered developers. |
| **Entry Condition:** Actor should log in to system. |
| **Exit Condition:** Developer profiles listed. |
| **Quality Requirement:**  **1.** System response to the actor arrives in a few seconds. |

**Use Case 5**

|  |
| --- |
| **Use Case Name:** View Developer Profiles |
| **Participant actor instances:** Developer, Admin |
| **Flow of events:**   * This event starts with actors enters to Developer Contacts web-site. * Then, if actor is a developer or an admin, he can see the “Developers” button on the main page. * So, actor clicks to "Developers" button on the main page. * Then, actor wants to look at a developer profile. So, he clicks the “View Profile” button which belongs to a developer who registered to system. * Actor can see the “profile” page which belongs to a developer. On the profile page, actor can see the avatar picture, biography, education, experience, skill sets, github repositories and social media accounts of related developer. |
| **Entry Condition:** Actor should be logged. |
| **Quality Requirement:**   * System response to the actor arrives in a few seconds. |
| **Exit Condition:** Developer profile is viewed. |

**Use Case 6**

|  |
| --- |
| **Use Case Name:** View Dashboard |
| **Participant actor instances:** Developer |
| **Flow of events:**   * This flow starts with clicking to "Dashboard" button on the main page. * Developer can see his/her developer profile. * Also, developer can edit his profile with “edit profile” button, add an experience with “add experience” button and add an education information with “add education” button. |
| **Entry Condition:** Developer should be logged. |

**Use Case 7**

|  |
| --- |
| **Use Case Name:** Delete Account |
| **Participant actor instances:** Developer |
| **Flow of events:**   * This event starts with clicking to "Dashboard" button on the main page. * Then, developer can delete his account with clicking the “delete my account” button. * Developer can clicks “delete my account” button to delete his account. |
| **Entry Condition:** User should have a developer account. |
| **Exit Condition:** Developer account is deleted. |
|  |

**Use Case 8**

|  |
| --- |
| **Use Case Name:** Delete Experience and Education Information |
| **Participant actor instances:** Developer |
| **Flow of events:**   * This event starts with clicking to "Dashboard" button on the main page. * Developer can delete his experience information with clicking the “delete experience” and education information with clicking the “delete education” button. |
| **Entry Condition:** Developer should have an experience or education information. |
| **Exit Condition:** Experience or education is deleted. |
| **Quality Requirement:**  **1.** System response to the actor arrives in a few seconds. |

**Use Case 9**

|  |
| --- |
| **Use case name:** Post Feed |
| **Participant actors:** Developer |
| **Flow of events:**   * Developer clicks to the "Post Feed" button on the main page. * Then, developer can post anything on this page with filling the post field. * Developer clicks the “Submit” button to create post. |
| **Entry Condition:** Developer should signed-in. |
| **Exit Condition:** Post is created. |
|  |

**Use Case 10**

|  |
| --- |
| **Use Case Name:** Delete Comment |
| **Participant actor instances:** Developer |
| **Flow of events:**   * This event starts with clicking to "Post Feed" button on the main page. * Then, Developer can see the all posts and comments on this page. * Developer can delete his own comment with clicking to cancel image. |
| **Entry Condition:** Developer should have a comment before. |
| **Exit Condition:** Comment is deleted. |
| **Quality Requirement:**  **1.** System response to the actor arrives in a few seconds. |

**Use Case 11**

|  |
| --- |
| **Use Case Name:** Delete Accounts |
| **Participant actor instances:** Admin |
| **Flow of events:**   * This event starts with clicking to "Developers" button on the main page. * Then, Admin can delete any account with clicking to “delete account” button. If admin thinks this account is inappropriate. * The deleted account is removed from the system. |
| **Entry Condition:** There should be an account in the system. |
| **Exit Condition:** Account is deleted. |
| **Quality Requirement:**  **1.** System response to the actor arrives in a few seconds. |

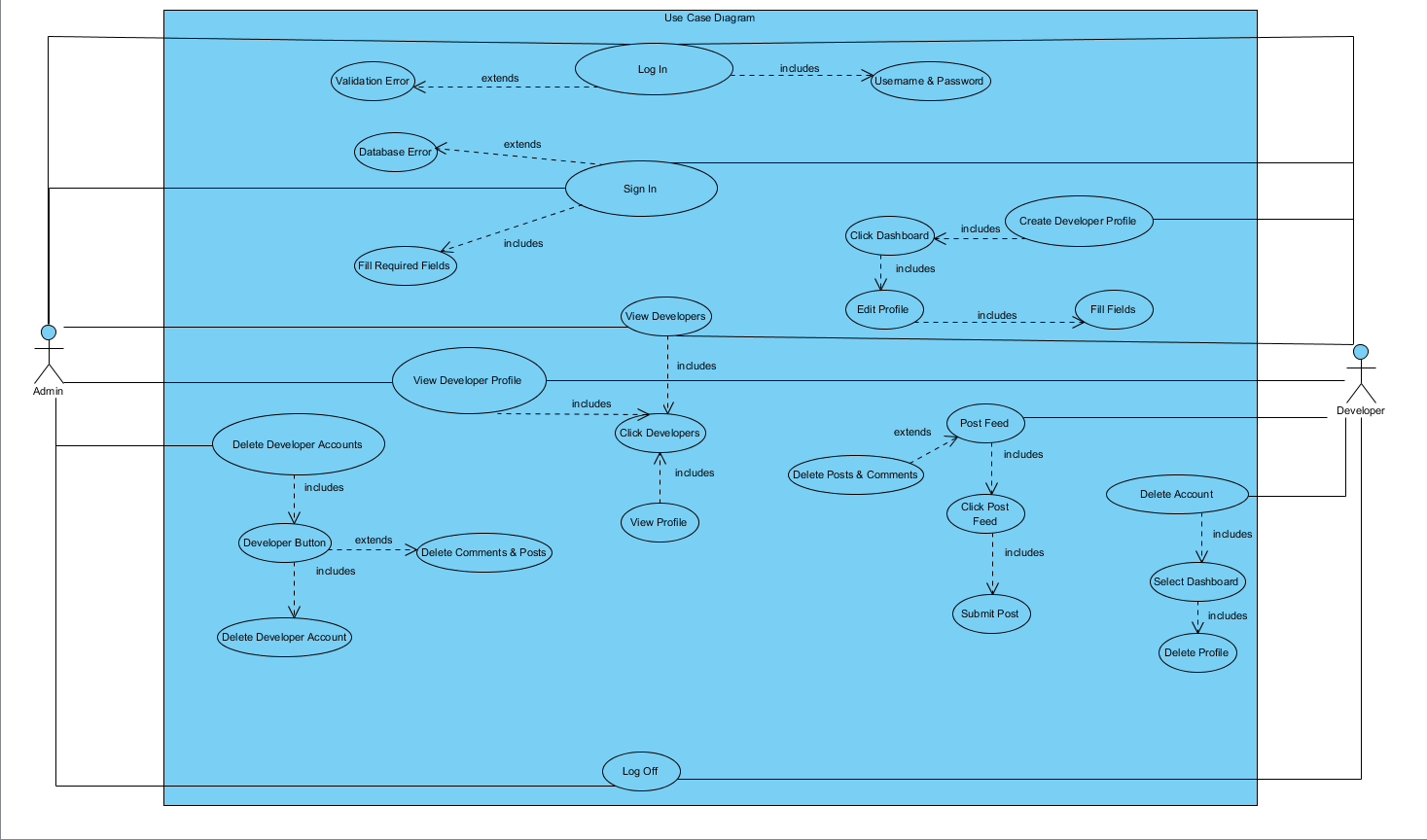
**Use Case 12**

|  |
| --- |
| **Use Case Name:** Delete Comments and Posts |
| **Participant actor instances:** Admin |
| **Flow of events:**   * This event starts with clicking to "Post Feed" button on the main page. * Admin can delete any post with clicking to “delete post” button or delete any comment with clicking to “delete comment” button. If Admin thinks this post or comment is inappropriate. * The deleted post or comment is removed from the system. |
| **Entry Condition:** There should be comments and posts in the system. |
| **Exit Condition:** Comments or posts are deleted. |
| **Quality Requirement:**  **1.** System response to the actor arrives in a few seconds. |

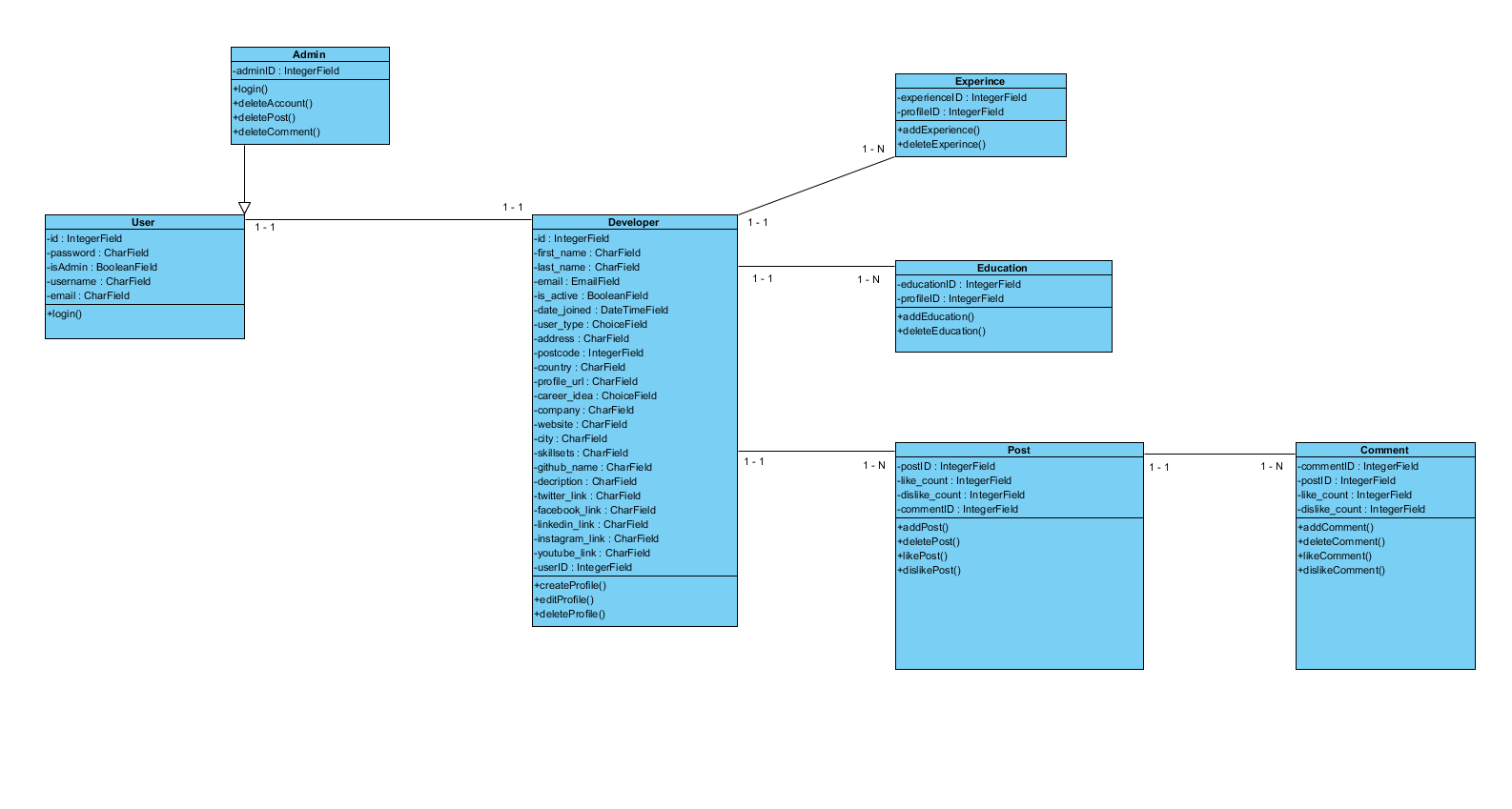
**Use Case 13**

|  |
| --- |
| **Use Case Name:** Log off |
| **Participant actor instances:** Developer, Admin |
| **Flow of events:**   * This event starts with entering to Developer Contacts web-site, and log in the web-site with his username and password. * Actors want to exit Developer Contacts web-site. * Actors click to “Log off” button. * Then system immediately logs them out and redirects to them on the home page. |
| **Entry Condition:** Actors should be logged in the system. |
| **Exit Condition:** Actors redirect to home page. |
| **Quality Requirement:**  **1.** System response to the actor arrives in a few seconds. |

* *Use Case Diagram*

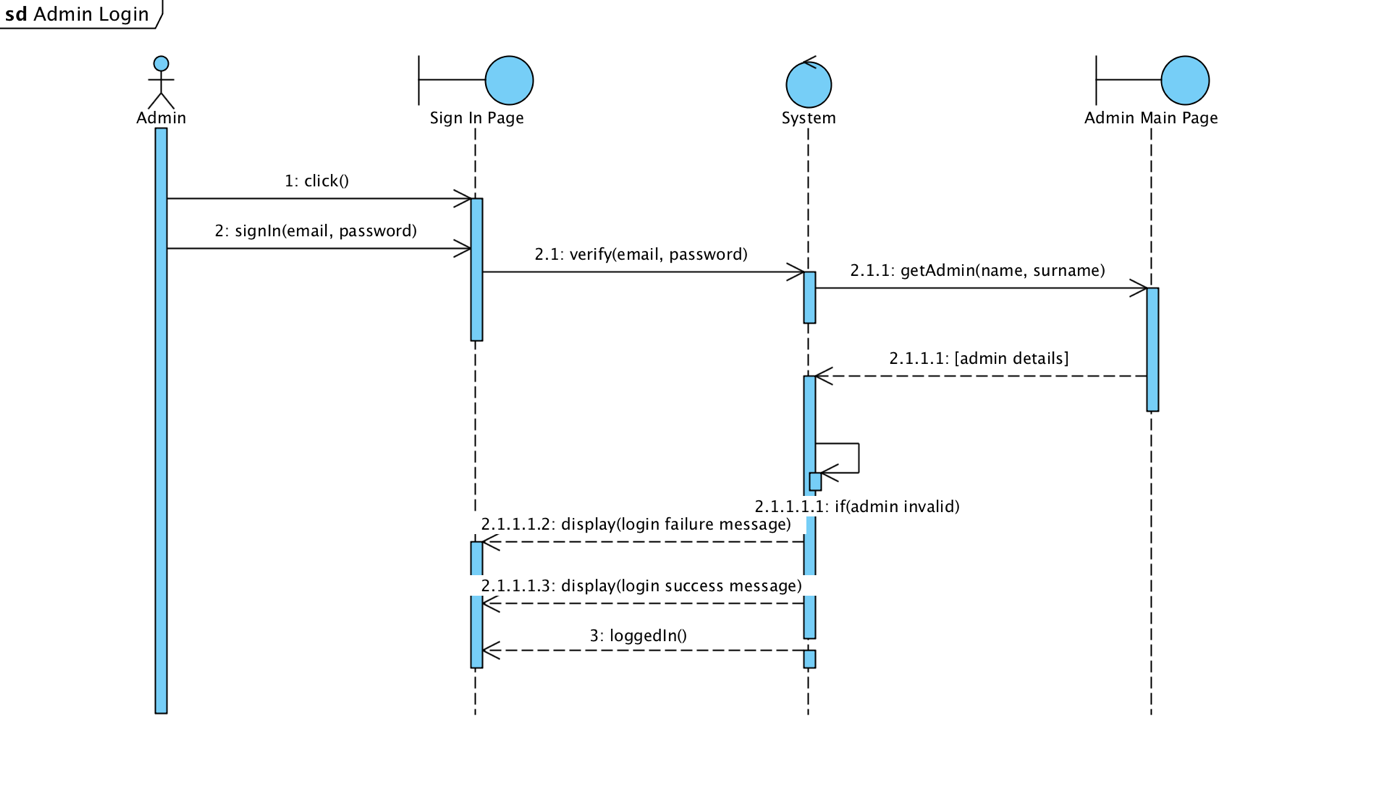


* *Object Model*



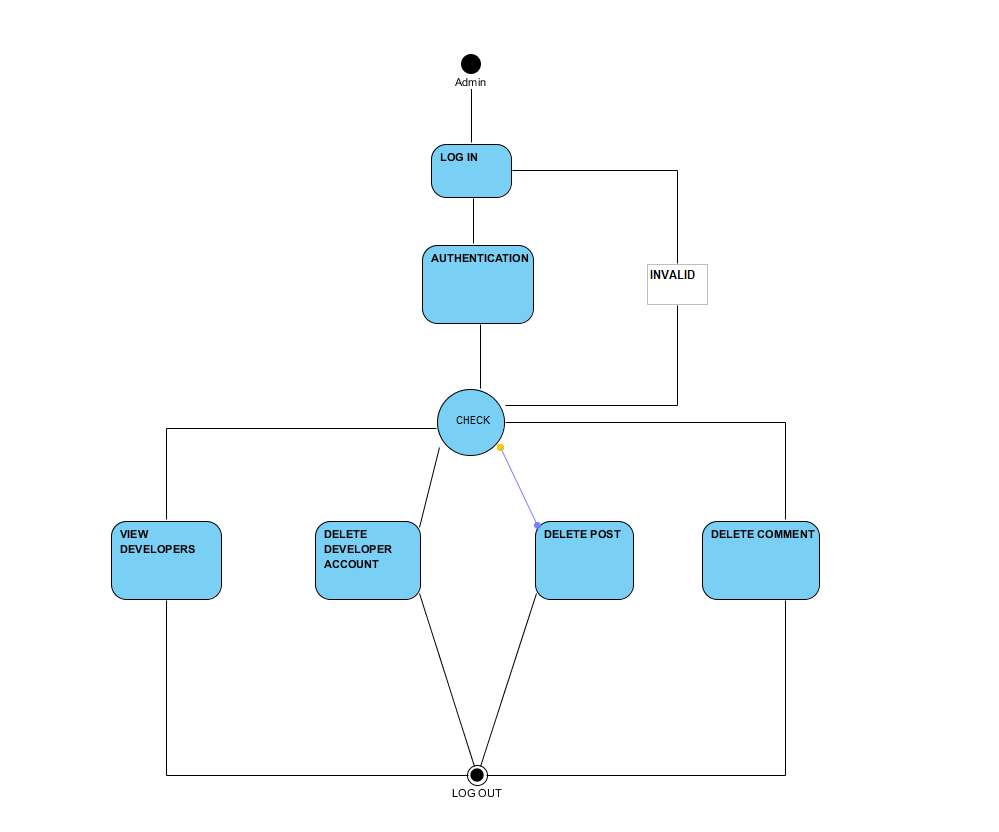
* *Dynamic Model*

Admin Dynamic Model

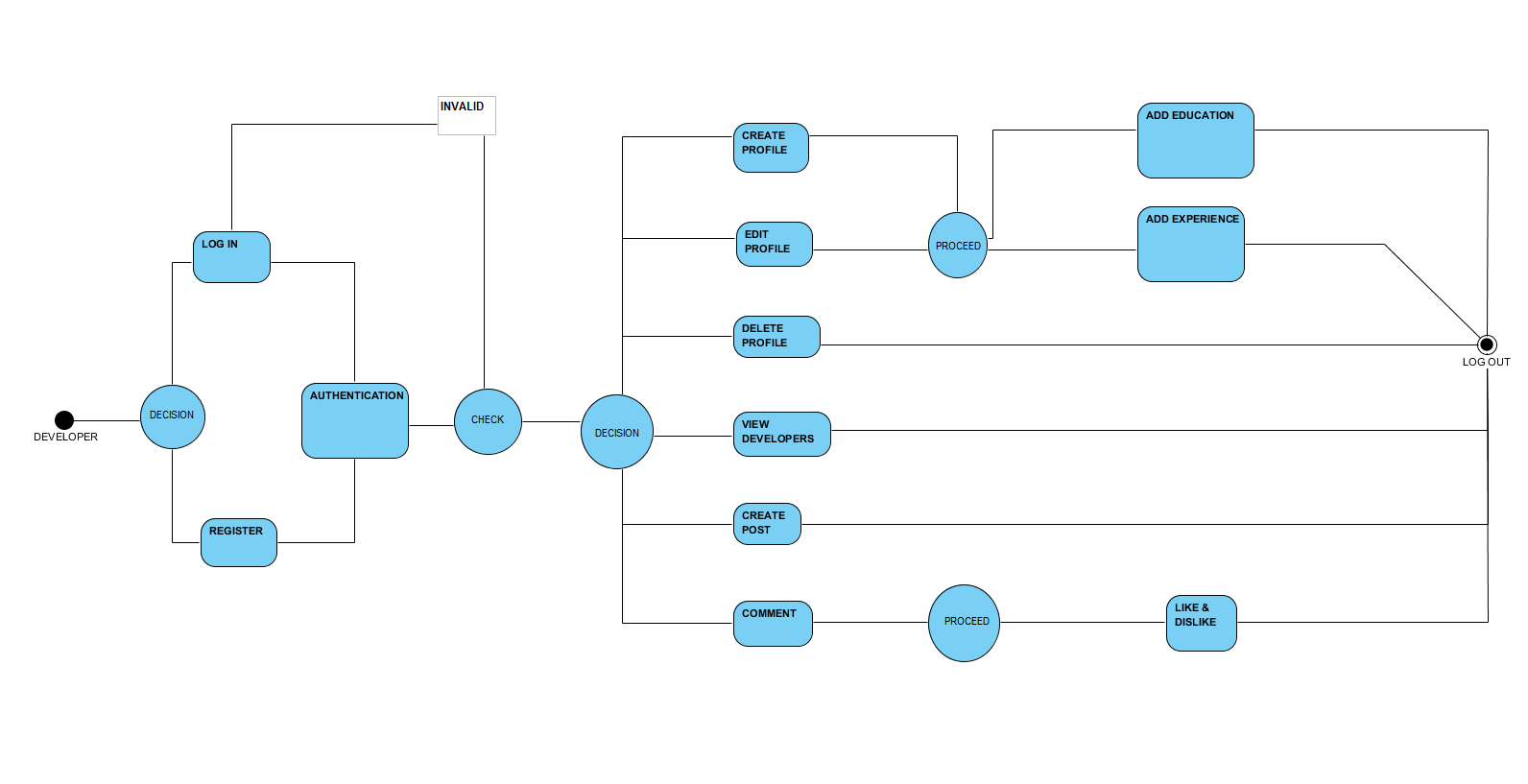


* *Activity Diagrams*

Admin Activity Diagram

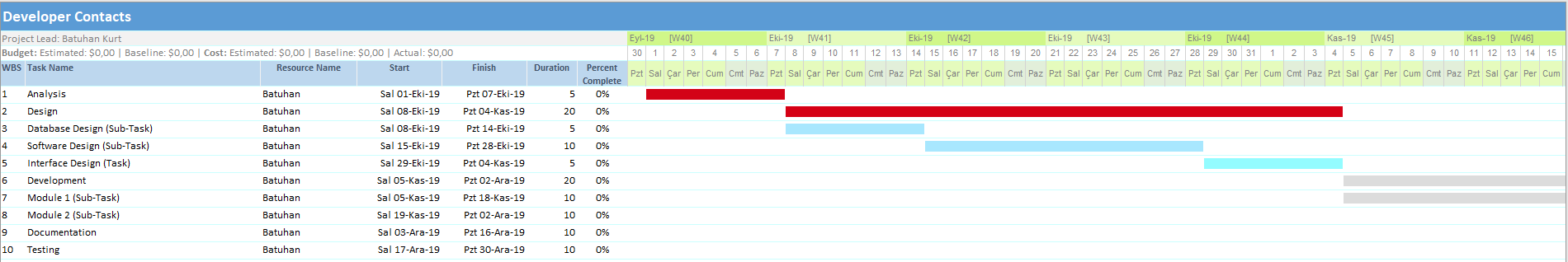


Developer Activity Diagram



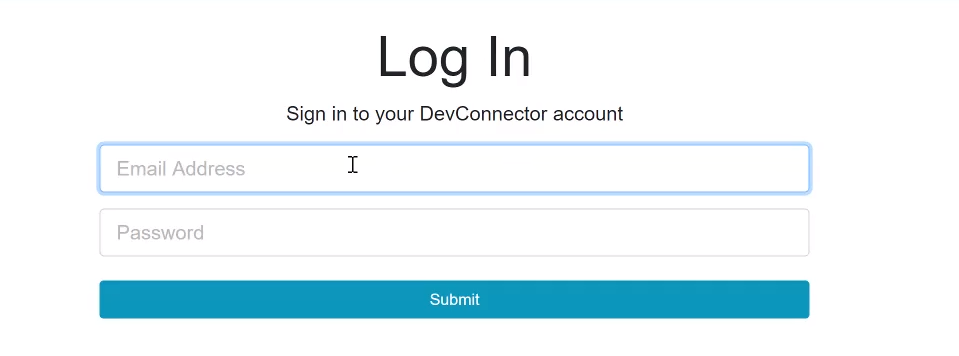
## Project Schedule

Project Gantt Chart

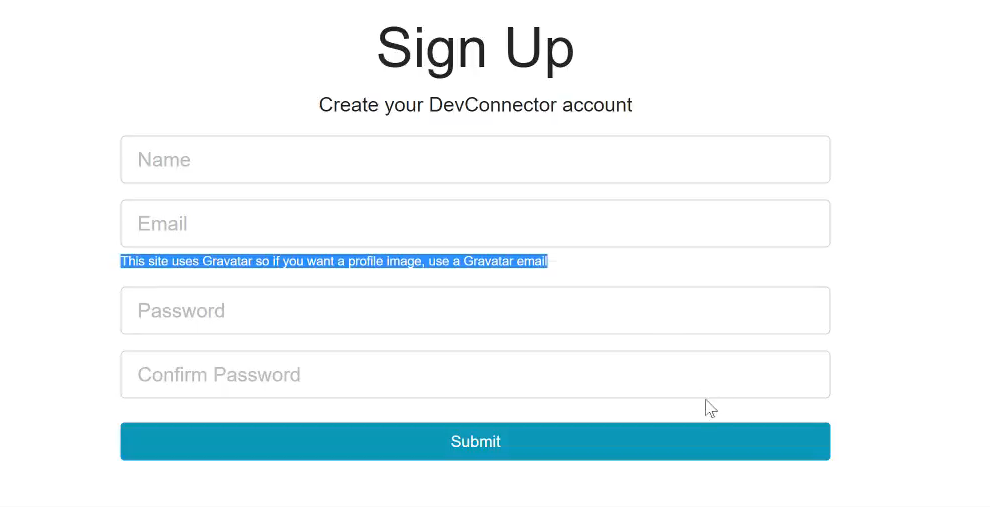


**User Interfaces**

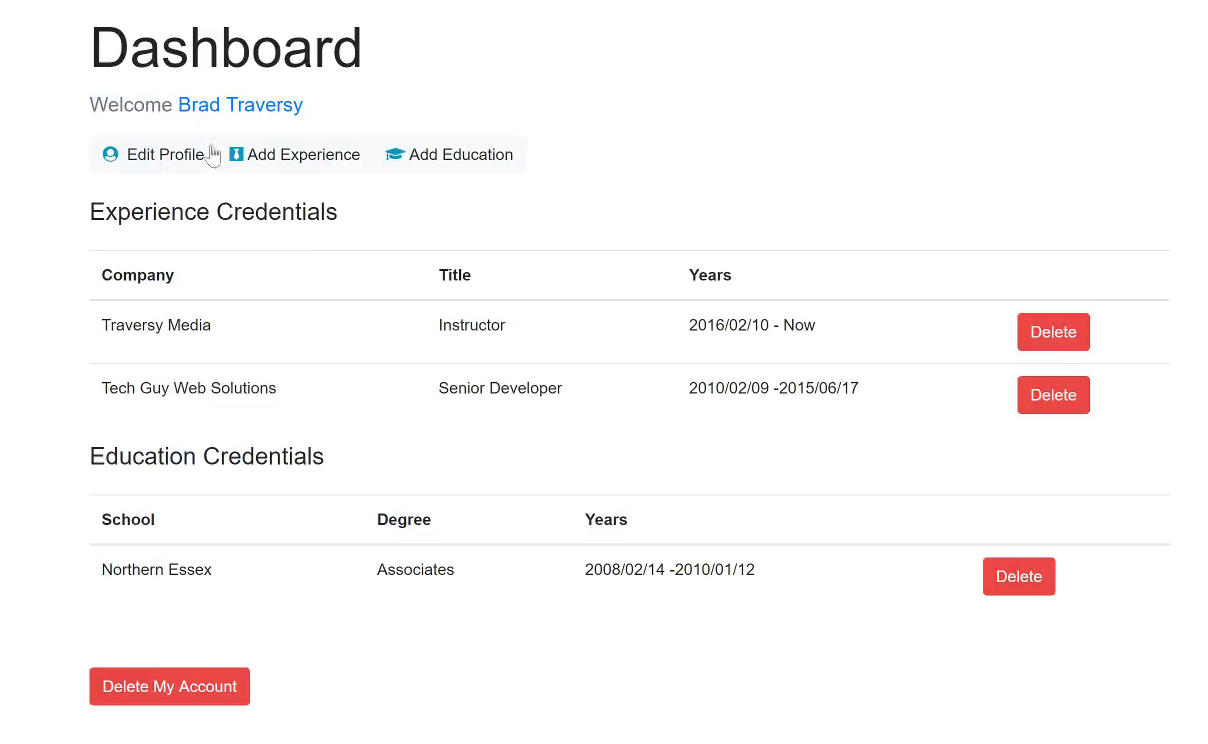
*Developer & Admin Login*



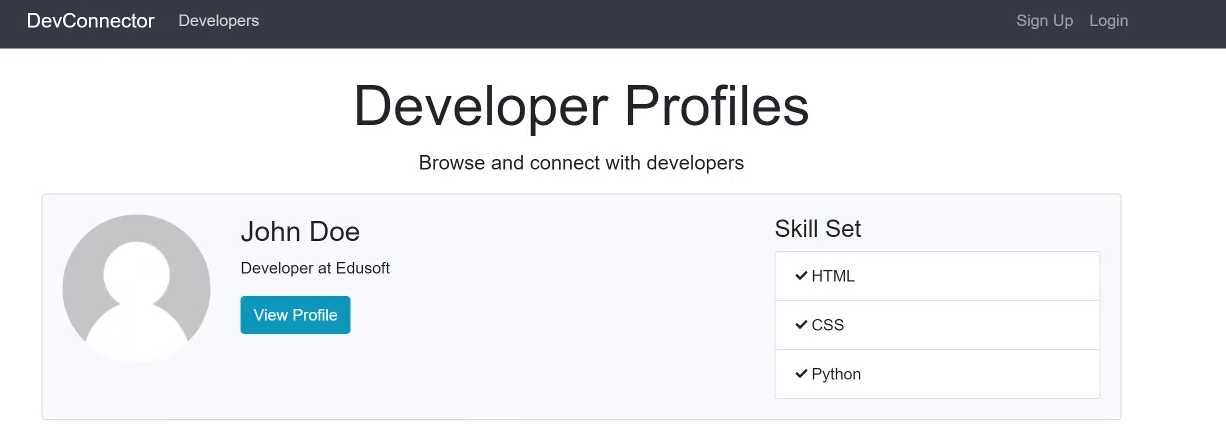
*Developer Register*

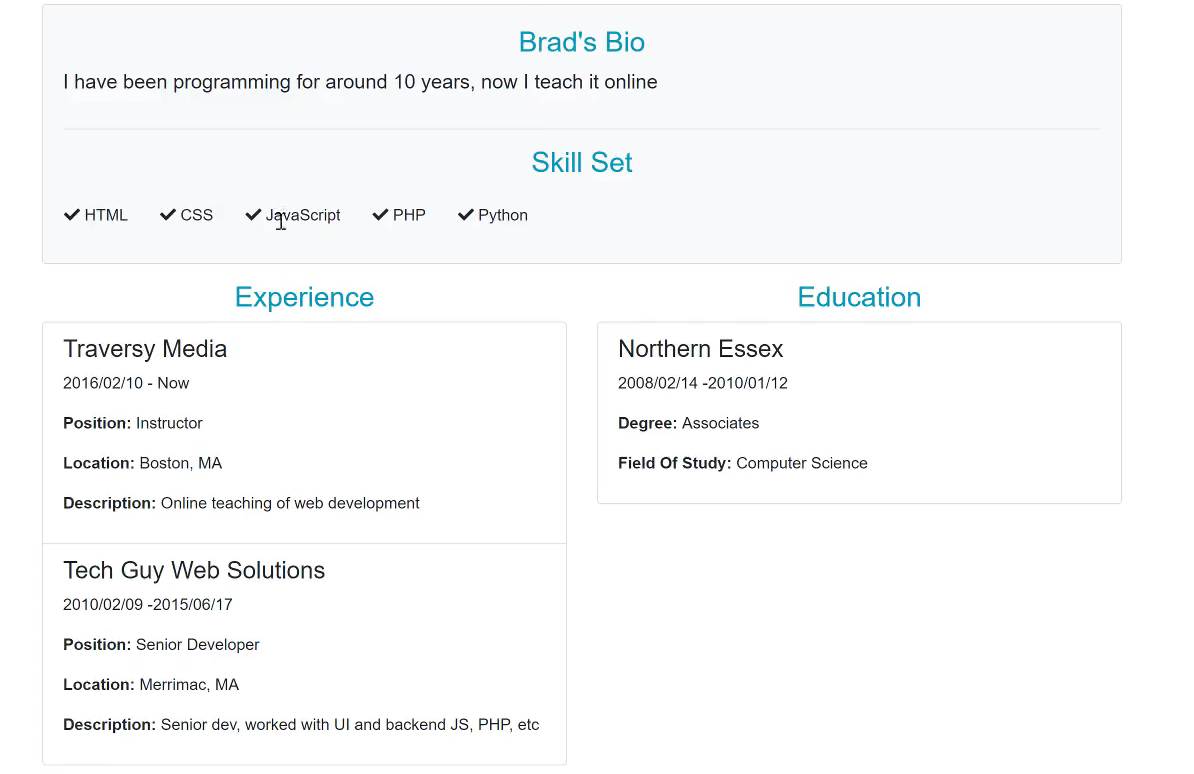
**

*Developer Dashboard Page*

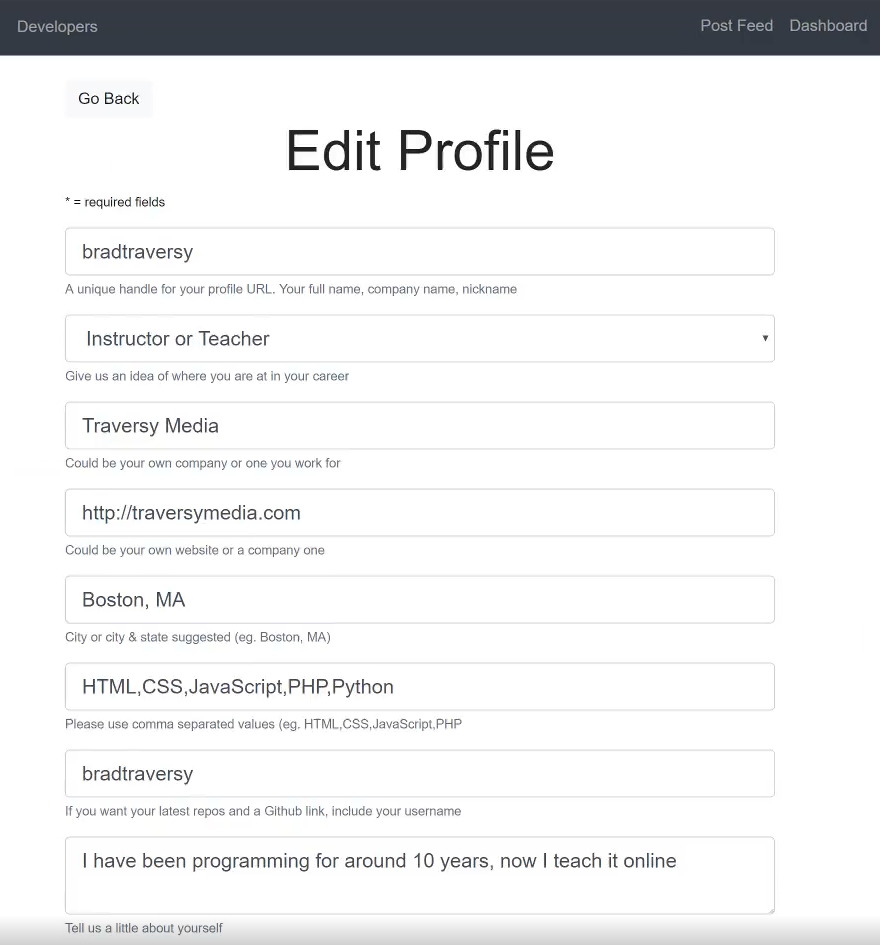
**

*View Developer Profile Page*

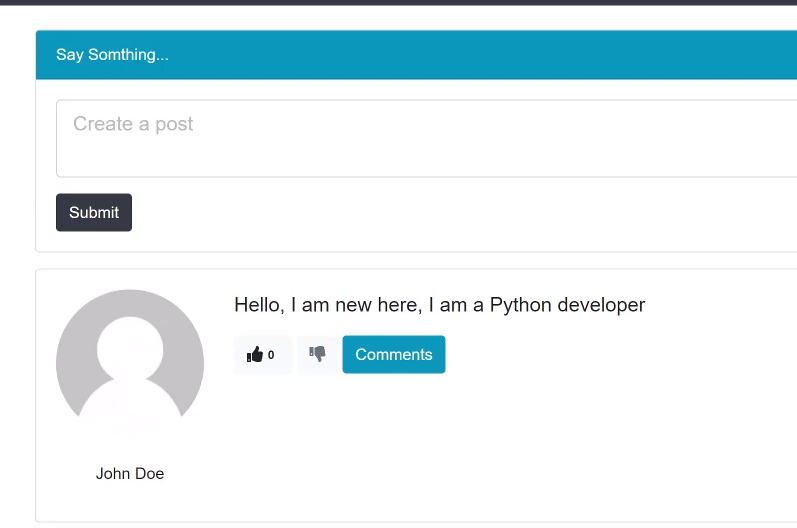


**

*Developer Edit Profile Page*

**

*Post & Feed Page*



# Glossary

**Developer:** Developer who register to system.

**Admin:** The system administrator.

**Create Profile:** Developer can create profile with this functionality.

**Create Group:** It is where the developer can create private group.

**Like & Unlike Comment:** It is where the developer like or dislike a comment.

**Database:** A collection of specific data.

**View and Edit Dashboard:** It is where the developer can view and edit his/her portfolio.

**Delete Account:** It is where the admin can delete any account, or a developer can delete his/her own account.

**Post Feed & Comment:** It is where the developer can share posts or comment to posts.

**Delete Groups:** It is where the admin can delete any private group.

**Functionality:** An action or control supported by our system for any of the users to carry out a task.

**Log in:** Being entered to the Developer Contacts System with a specific registered account.

**Log out:** Being exited from the Glass Booking System with a specific registered account.

**Use case:** A methodology used in system analysis to identify, clarify, and organize system requirements.

**Registered User:** A user who has an account on the Developer Contacts System.

**Password:** A secret word or phrase that must be choose of all developers when they are register.

**Profile:** A section that includes the developer’s information.

# 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/class-diagrams-overview.html>
3. <https://www.uml-diagrams.org/class-reference.html>
4. <https://www.ibm.com/developerworks/architecture/library/ar-analpat/> (IBM)