**Software Design Specification**

**Gayathri - U101116FCS039**

**Arnav Ajay – U101116FCS014**

**Neeraj Pandey – U101116FCS076**

**Ashwin Jawahar – U101116FCS017**

**Hrithik Raj – U101116FCS047**

**1. Introduction**

* 1. **Purpose of this document**

This Software Design Specification Document is made with the purpose of outlining the software architecture and design of the Online Examination Portal in detail. This document will provide developers an insight in meeting the client’s needs efficiently and effectively. Moreover, the document facilitates communication and understanding of the system by providing sequence diagrams, collaboration models, state diagrams and other required documents of the system design. This document will demonstrate how the design will accomplish the functional and non-functional requirements captured in the Software Requirement Specification Document. The document will provide a framework to the programmers through describing the architecture, sub-systems, interfaces, database design and functions.

**1.2 Scope of the development project**

The scope of this project is very broad as compared to manually taken examinations:

1. This portal is not only limited to educational institutes but also to the corporate world
2. There is no restriction as to the presence of an examiner during the period of the exam
3. Less time consumption, thereby increasing the efficiency
   1. **Definitions, acronyms, and abbreviations**

IEEE: Institute of Electrical and Electronics Engineers

SDS: Software Design Specification

**1.4 References**

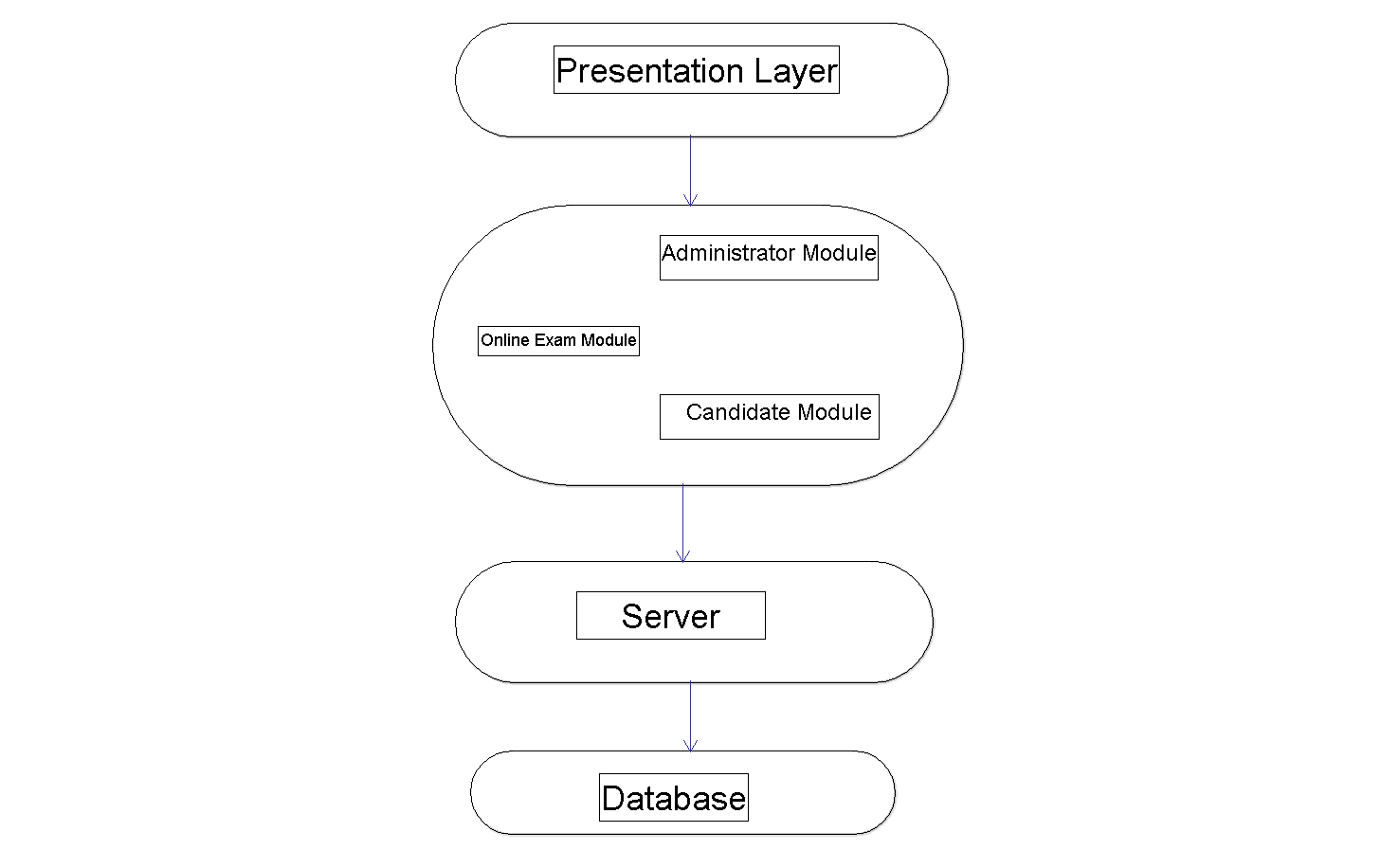
1. R. S. Pressman, Software Engineering: A Practioner’s Approach, 5th Ed, McGraw-Hill, 2001.
2. IEEE SDS template

**1.5 Overview of document**

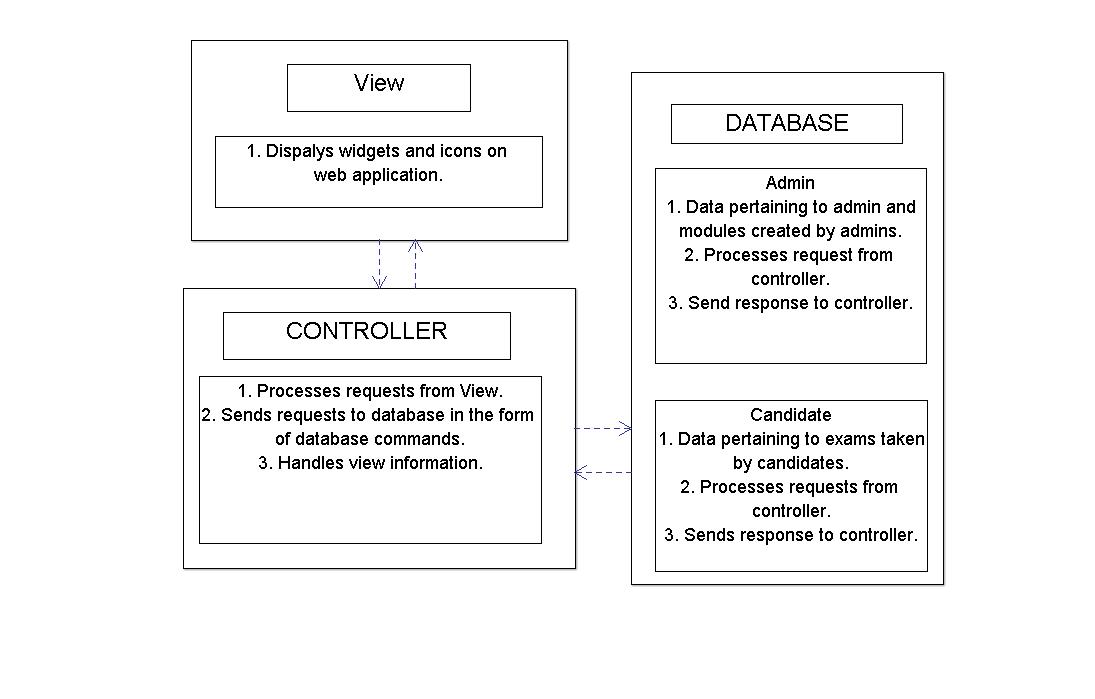
The SDS document is divided into the following sub-sections:

1. Introduction: It mainly describes the purpose of the document and scope of the development project.
2. Conceptual Architecture/Architecture Diagram: The intent of conceptual architecture is to direct attention at an appropriate decomposition of the system without delving into the details of interface specification.
3. Logical Architecture: Logical architecture is a structural design that gives as much detail as possible without constraining the architecture to a particular technology or environment.
4. Execution Architecture: The execution architecture determines largely the realtime and performance behavior of a system. Concepts such as latency, response time and throughput are illustrated.
5. Design Decisions and Trade-offs: It describes the decisions taken throughout the system design and why they were taken instead of their alternatives.
6. Pseudocode: Pseudocode/algorithms implemented in the system.
7. Appendices: Subsidiary matters, if any.

**2. Conceptual Architecture/Architecture Diagram**

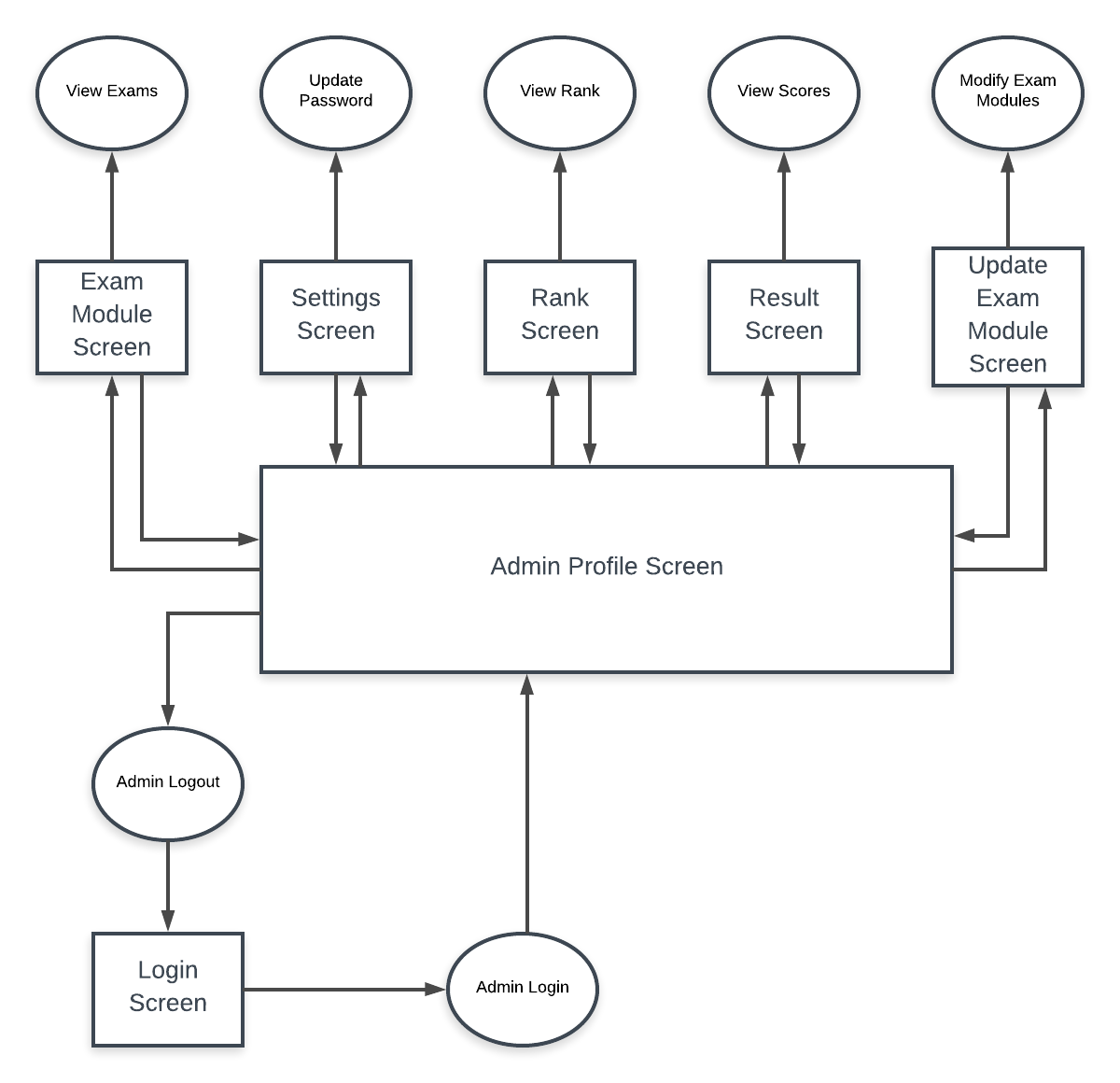


**2.1 Overview of modules / components**

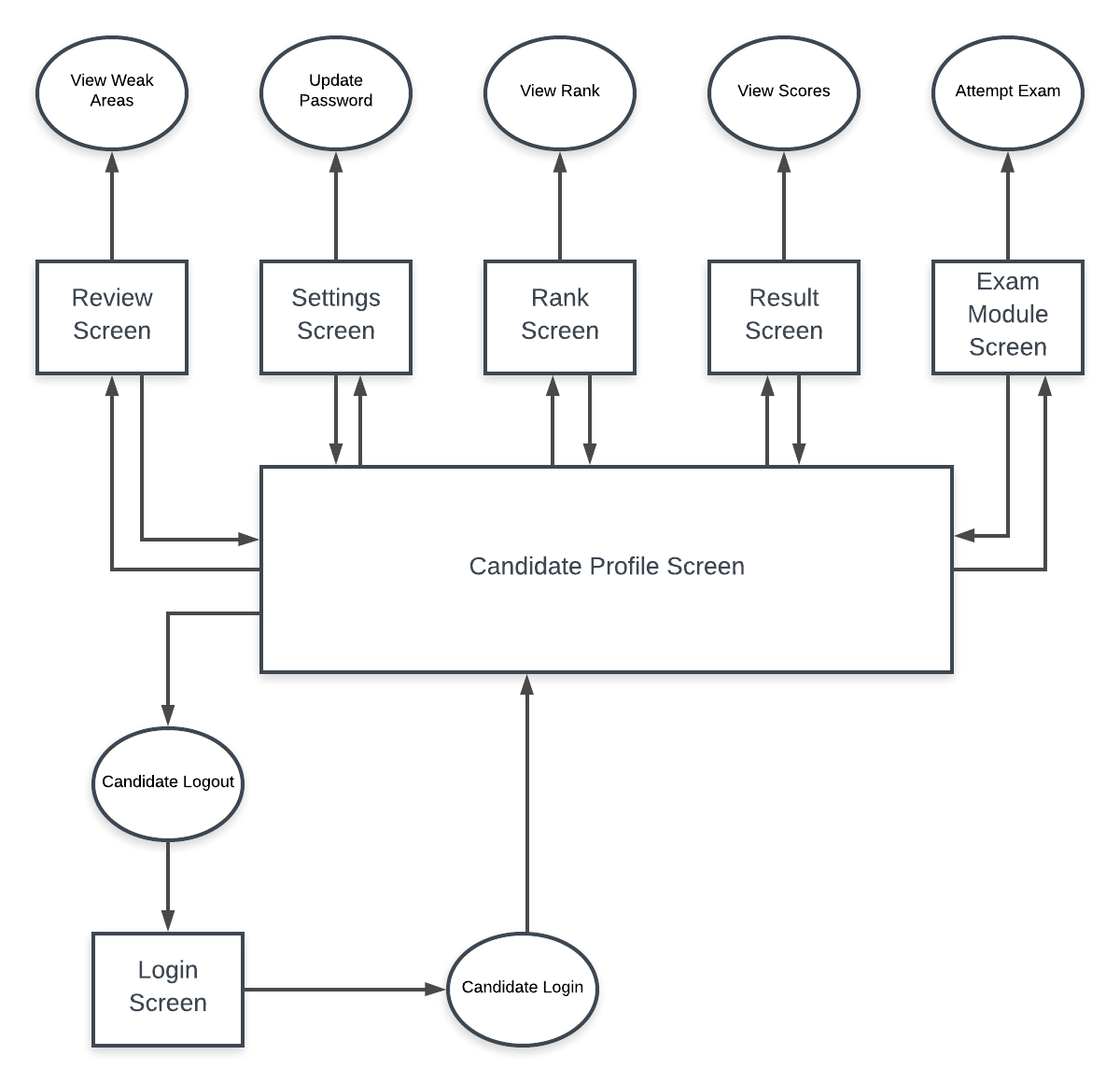


**2.2 Structure and relationships**

* + 1. Administrator



* + 1. Candidate

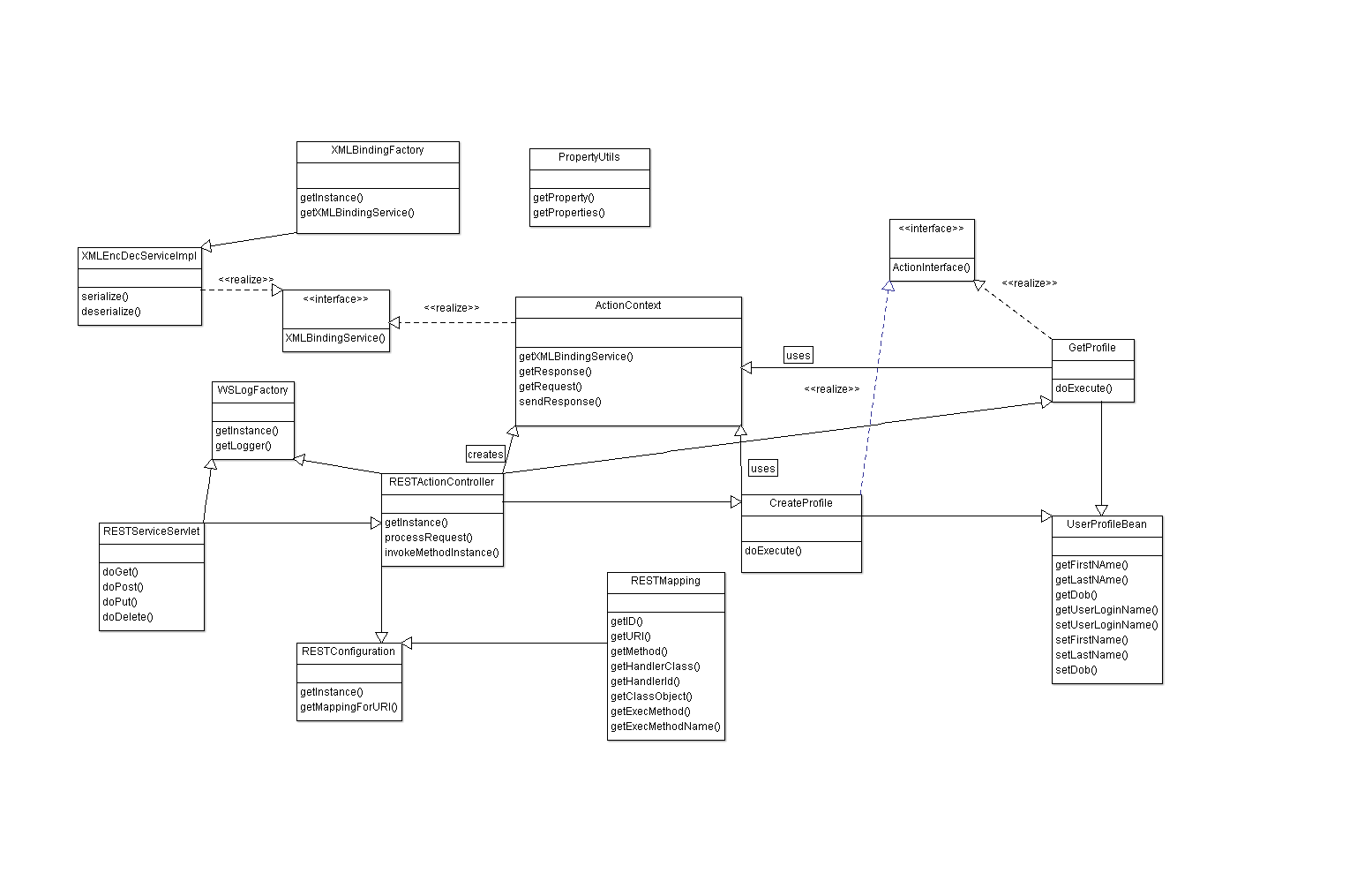


**2.3 User interface issues**

Application will be accessed through a Browser Interface. The interface would be viewed best using 1024 x 768 and 800 x 600 pixels resolution setting. No user would be able to access any part of the application without logging on to the system. User Interface Issues for the possible users of the system are:

1. User A is a 23-year-old student or corporate candidate who is looking to take a mock online examination using this system. As the user is young and technologically proficient in using computer applications and systems, the user interface will be of fairly common conventions on the candidates’ screen.
2. User B is a 35-year-old person who is an administrator of the online system with responsibilities to update exam modules on the site by modifying the questions. Since, user B might not be completely well-versed in using online portals, directives will be given for ease of use.

**3. Logical Architecture (Class Diagram, Sequence Diagram, State Diagram)**

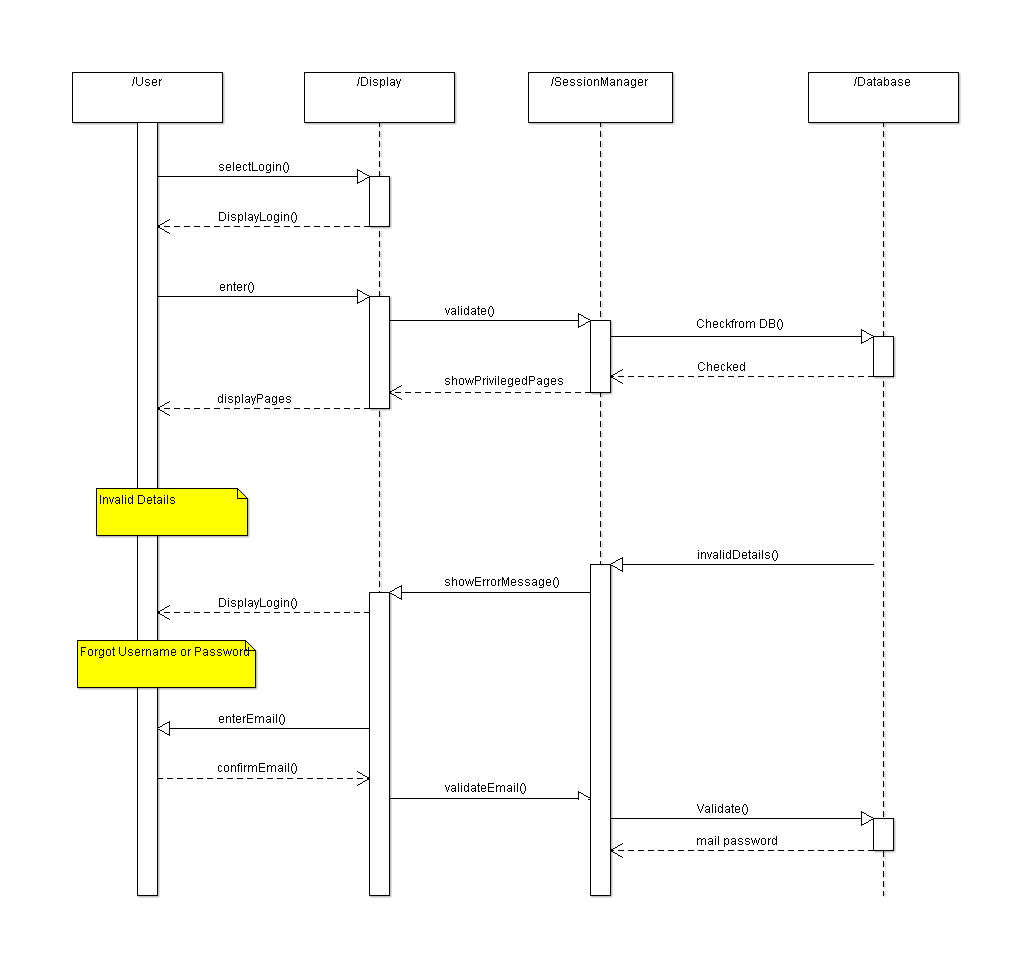


**3.1 Logical Architecture Description**

Representational State Transfer is a software architectural style that defines a set of constraints to be used for creating web services. Web services that conform to the REST architectural style, or RESTful web services, provide interoperability between computer systems on the Internet. One of the key advantages of REST APIs is that they provide a great deal of flexibility. Data is not tied to resources or methods, so REST can handle multiple types of calls, return different data formats and even change structurally with the correct implementation of hypermedia. This explains the class structure in Java. The rest of the functionalities use bootstrap and typescript, the architecture of which is given below.

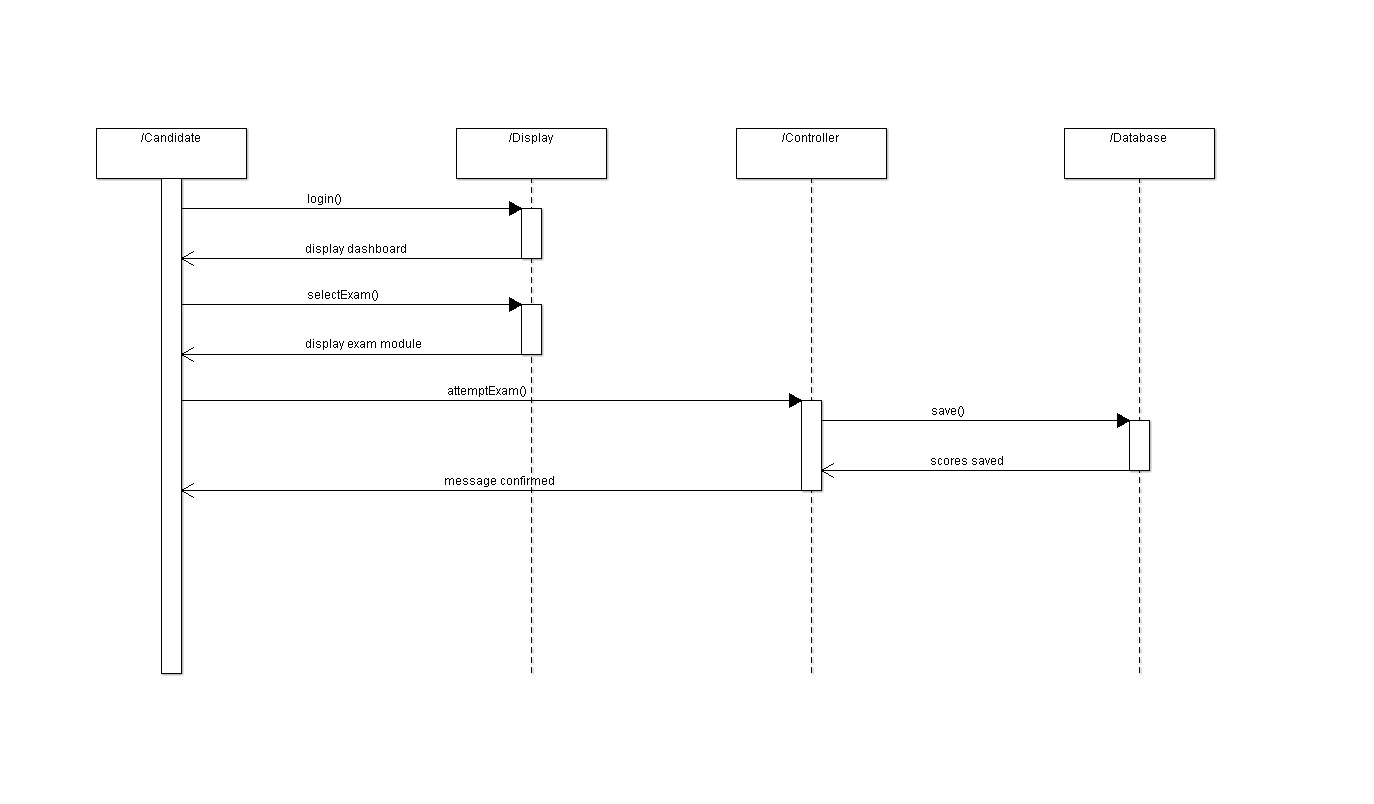
**3.2 Login**

3.2.1 Sequence Diagram

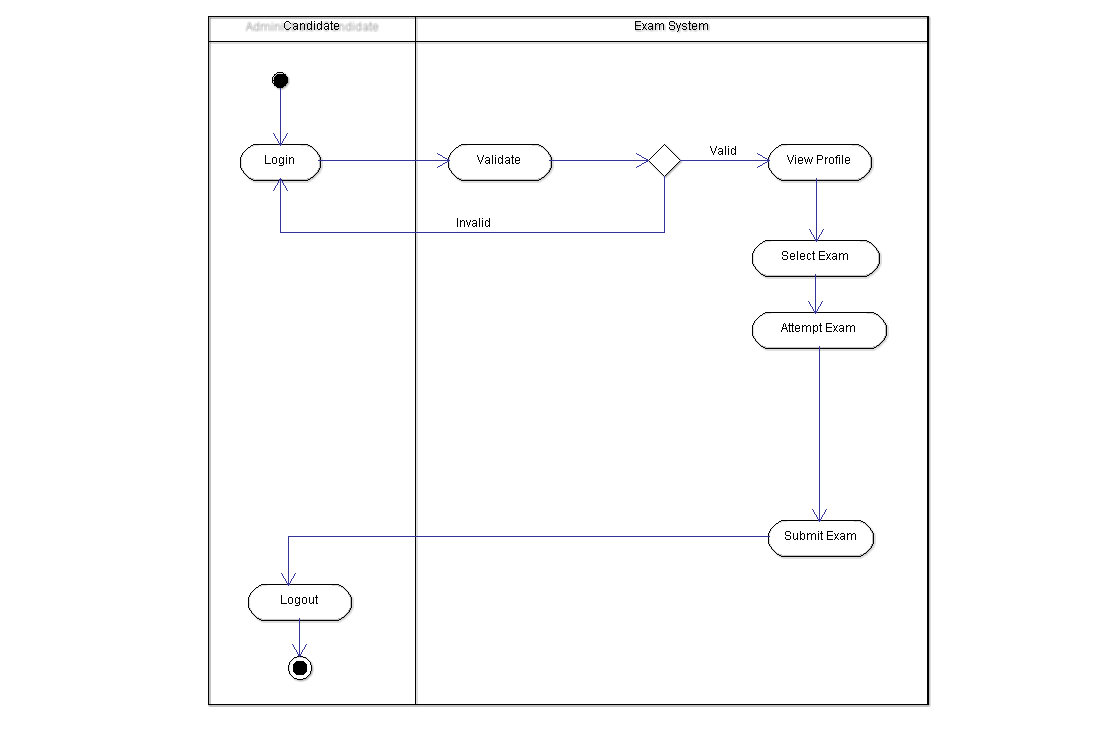


**3.3 Attempt Exam**

* + 1. Sequence Diagram

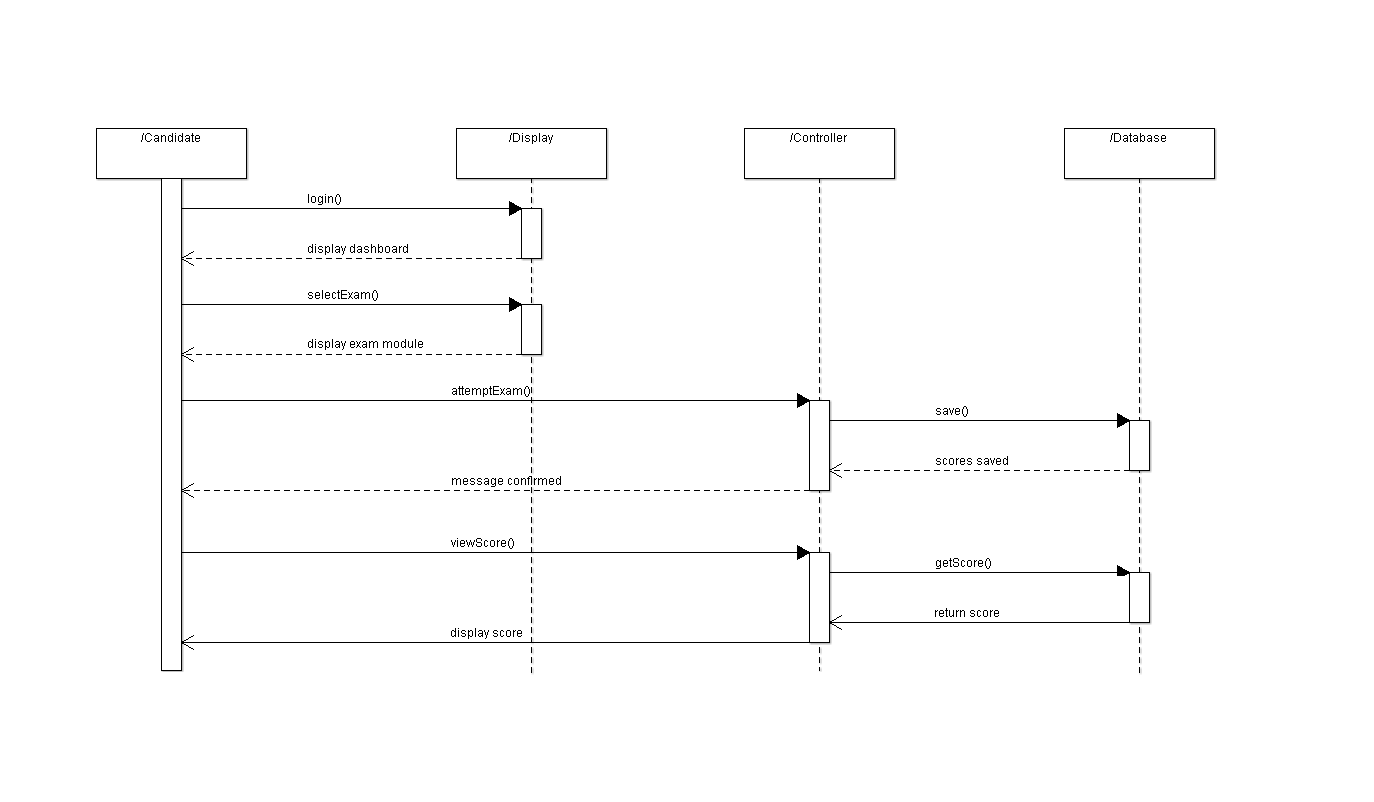


3.3.2 Activity Diagram

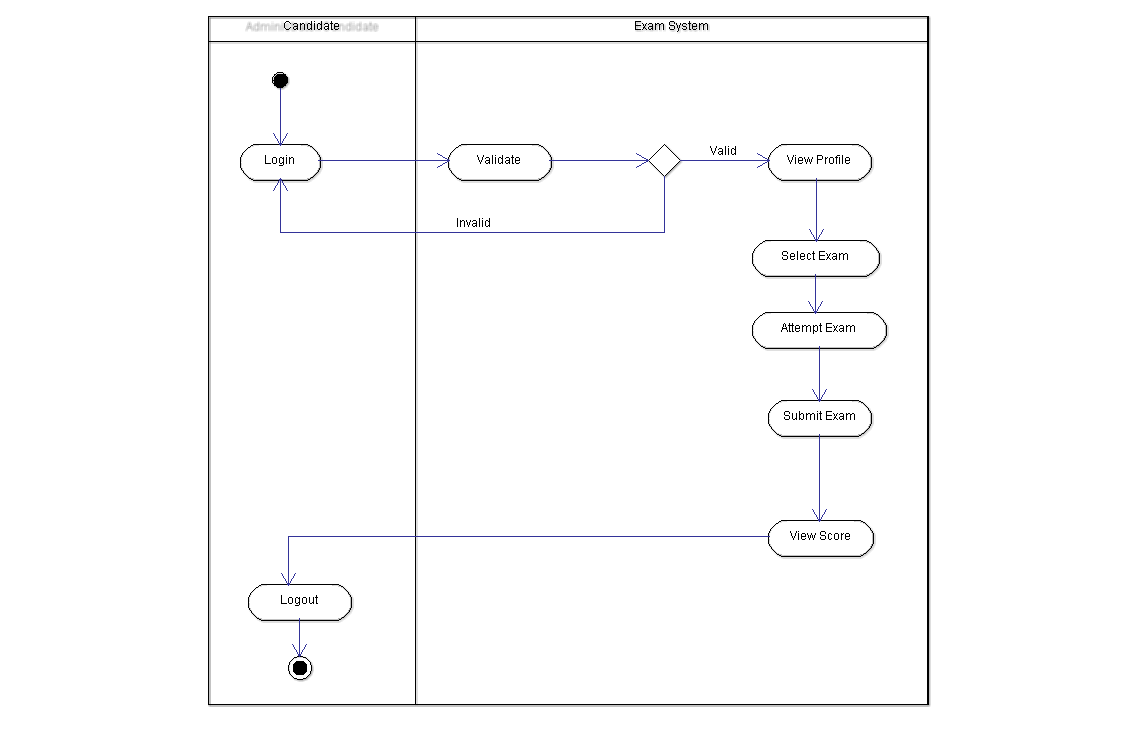


**3.4 View Score**

3.4.1 Sequence Diagram

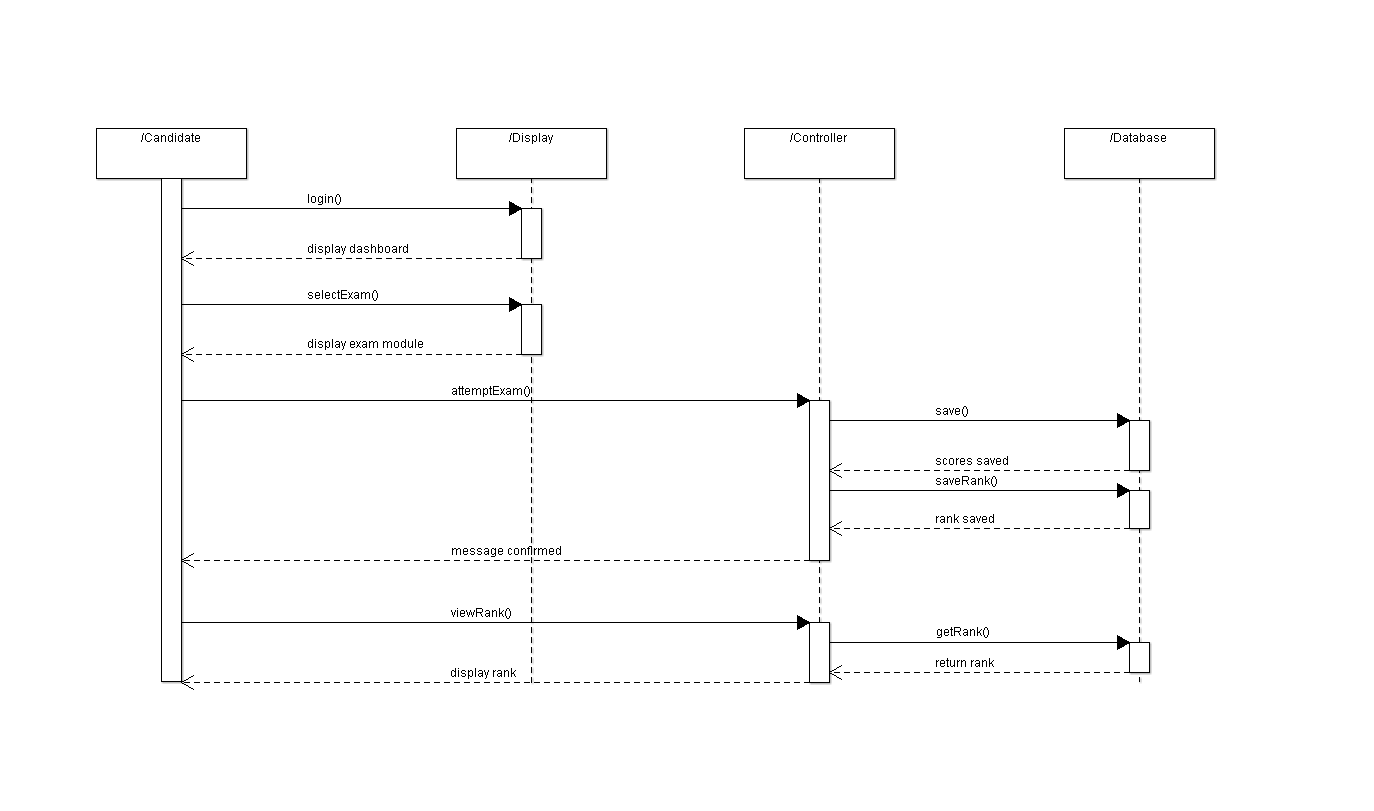


3.4.2 Activity Diagram

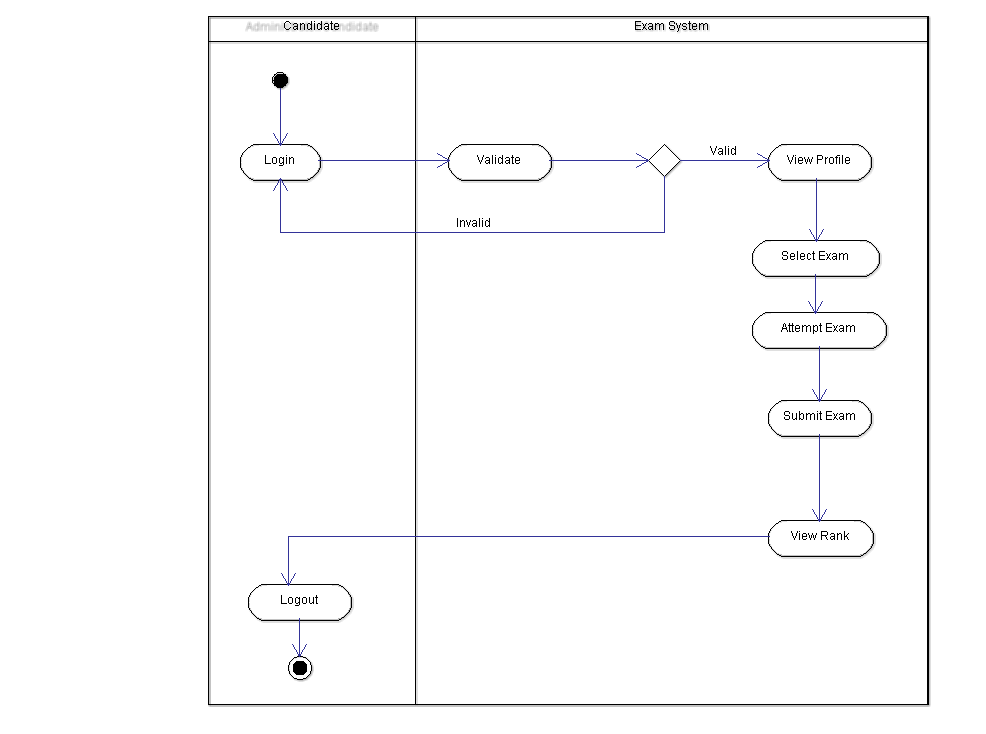


**3.5 View Rank**

3.5.1 Sequence Diagram

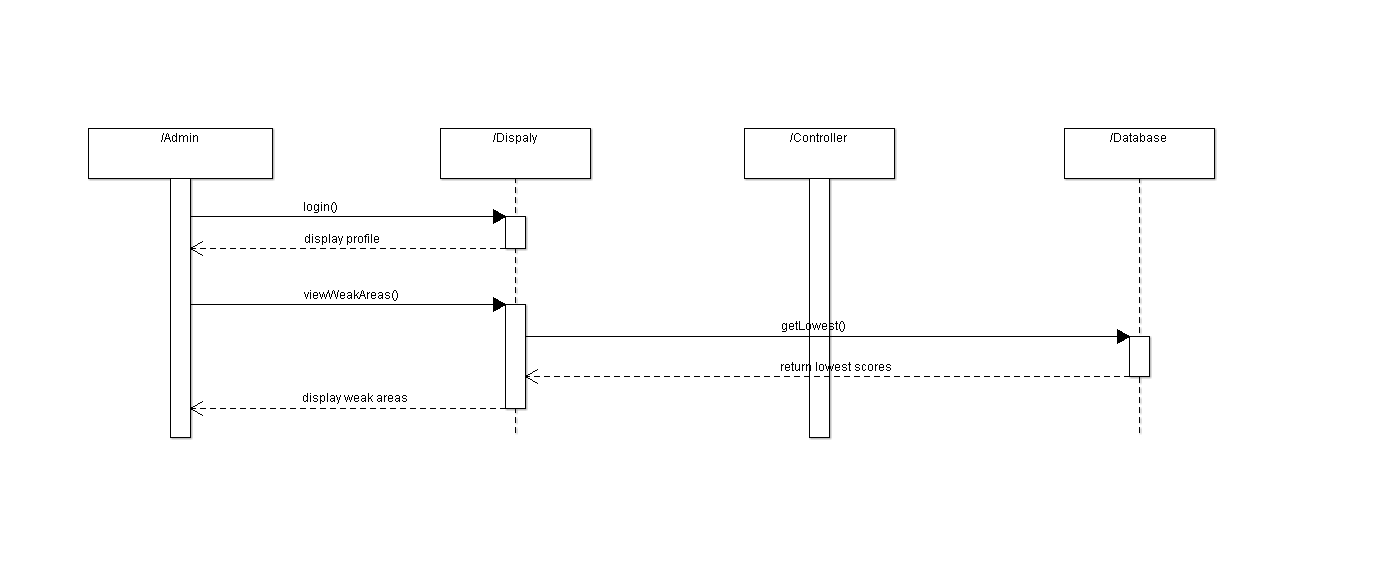


3.5.2 Activity Diagram

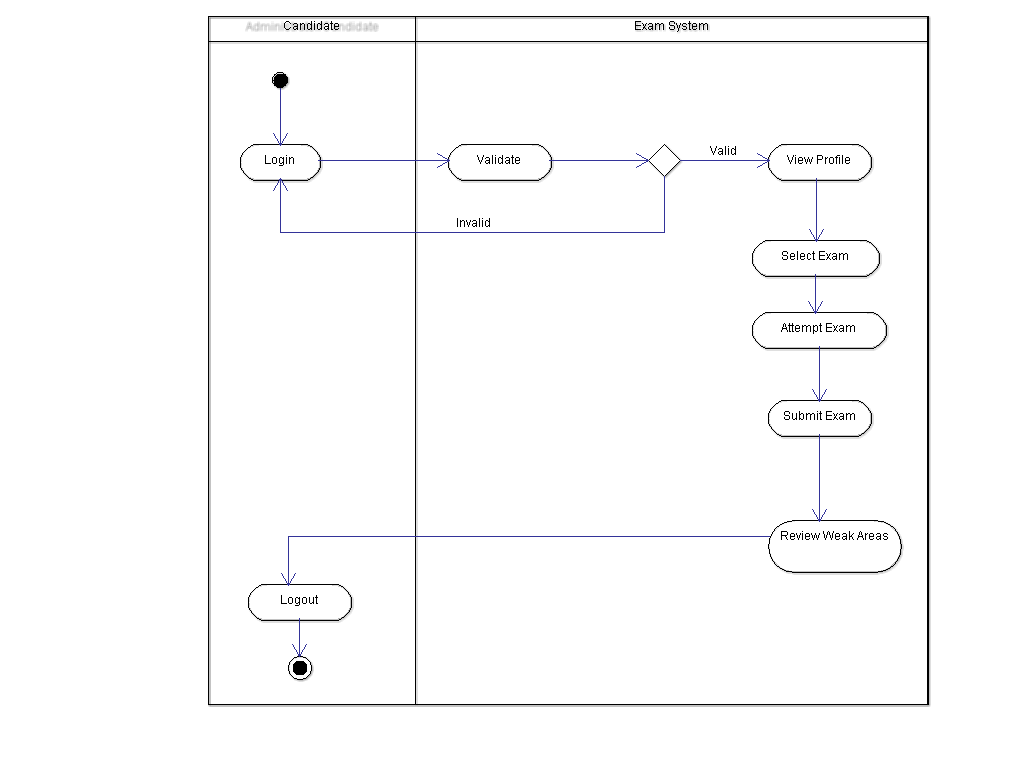


**3.6 View Weak Areas**

3.6.1 Sequence Diagram

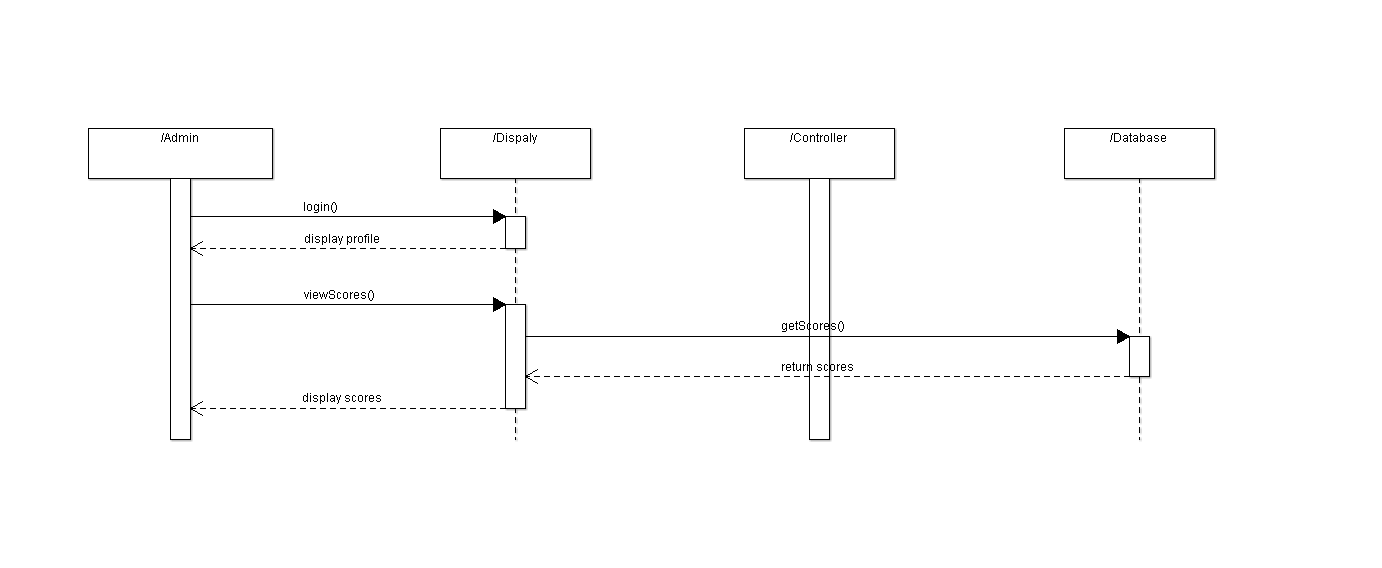


3.6.2 Activity Diagram

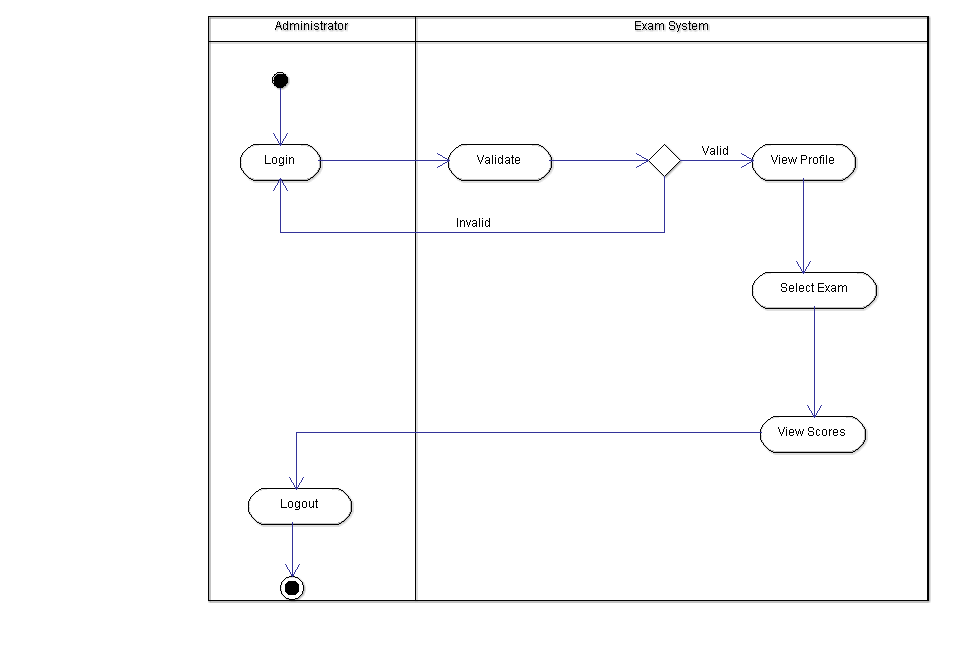


**3.7 Admin- View Scores**

3.7.1 Sequence Diagram

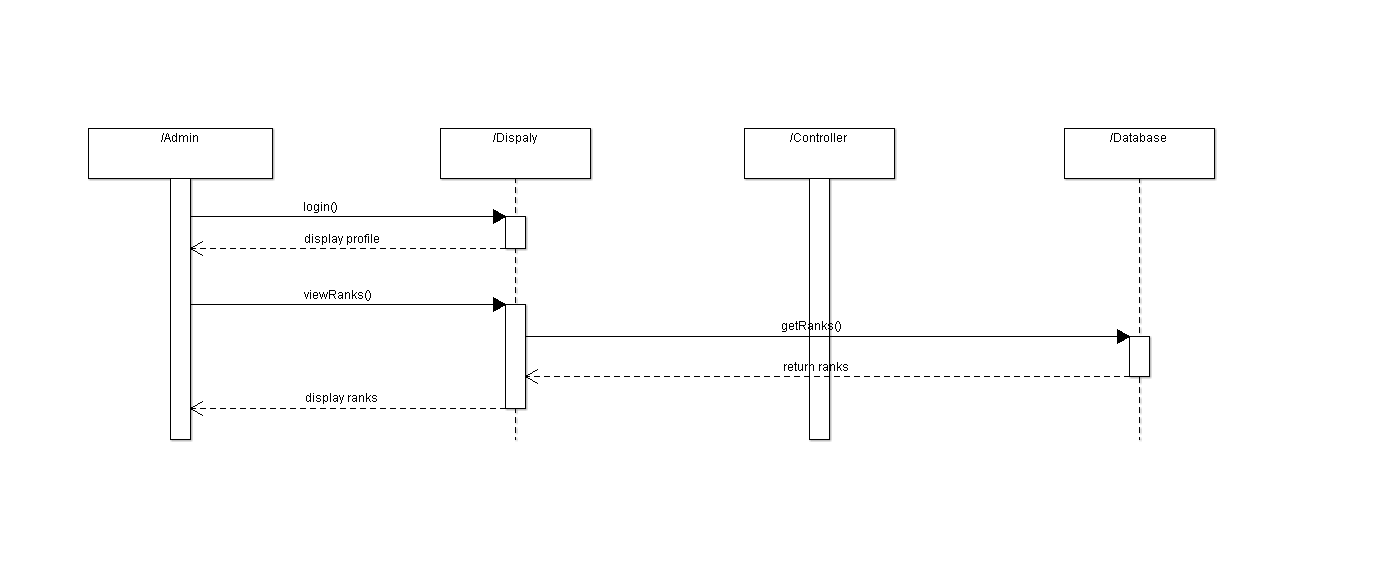


3.7.2 Activity Diagram

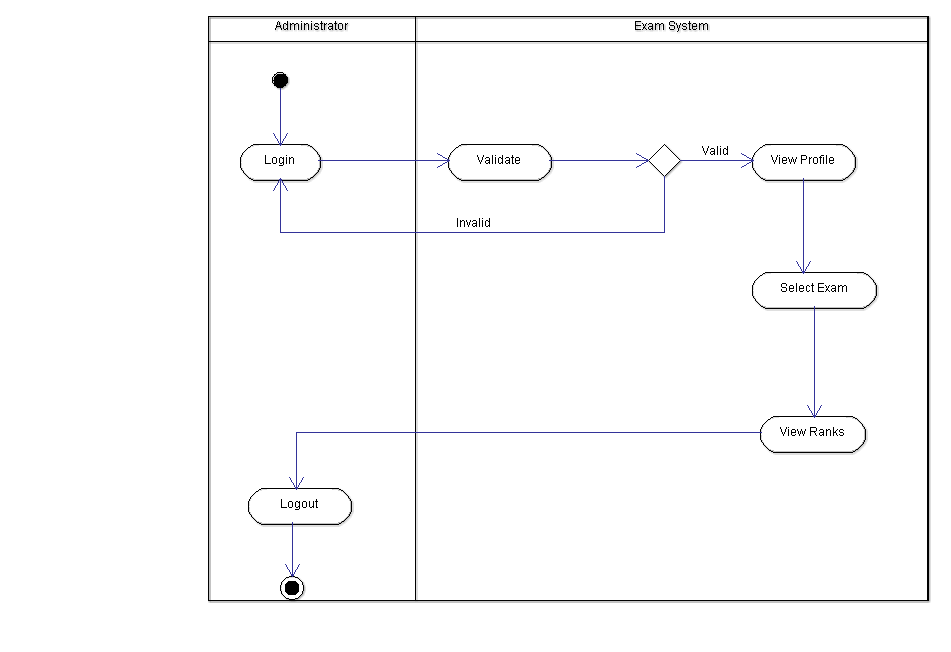


**3.8 Admin- View Ranks**

3.8.1 Sequence Diagram

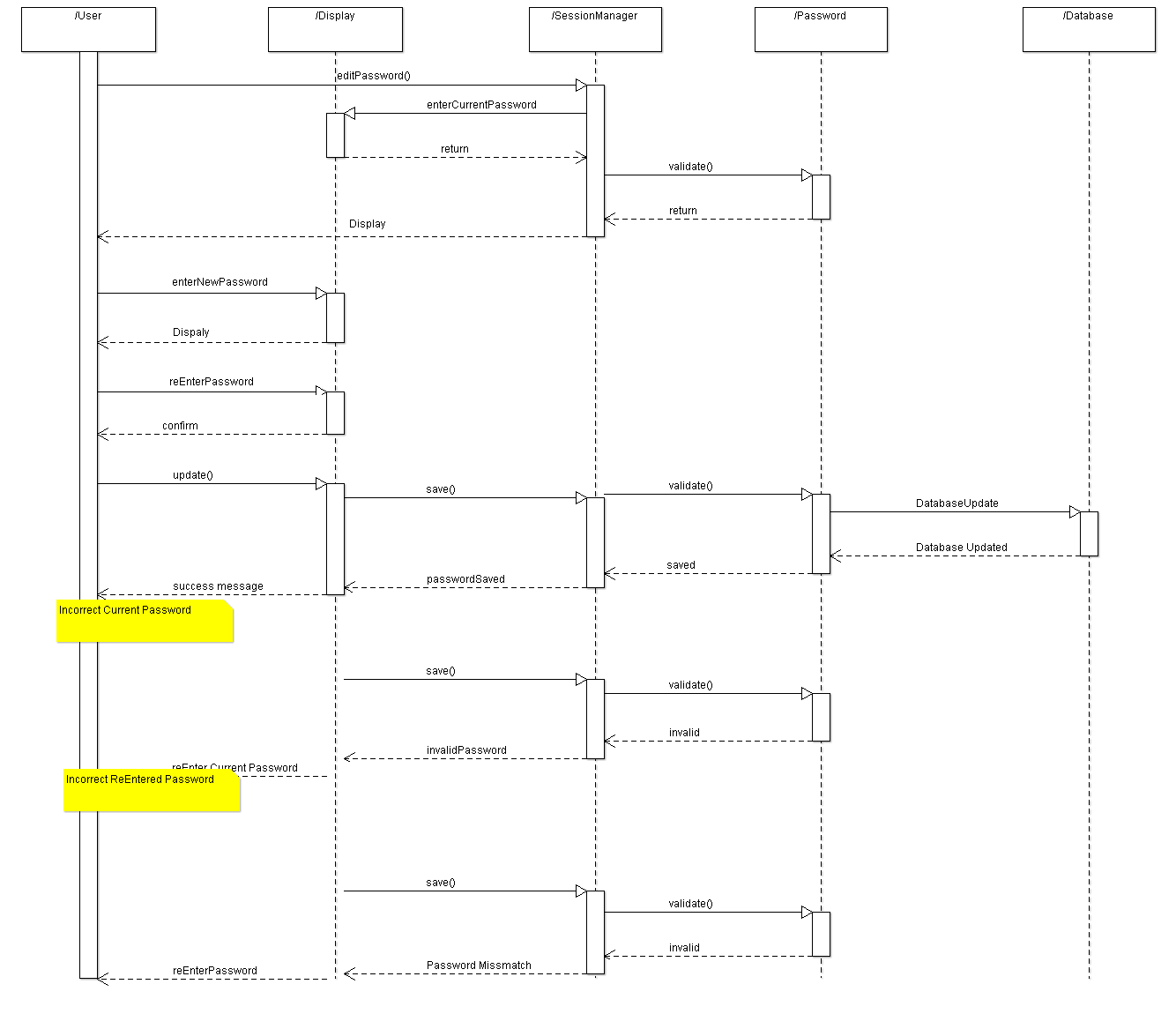


3.8.2 Activity Diagram

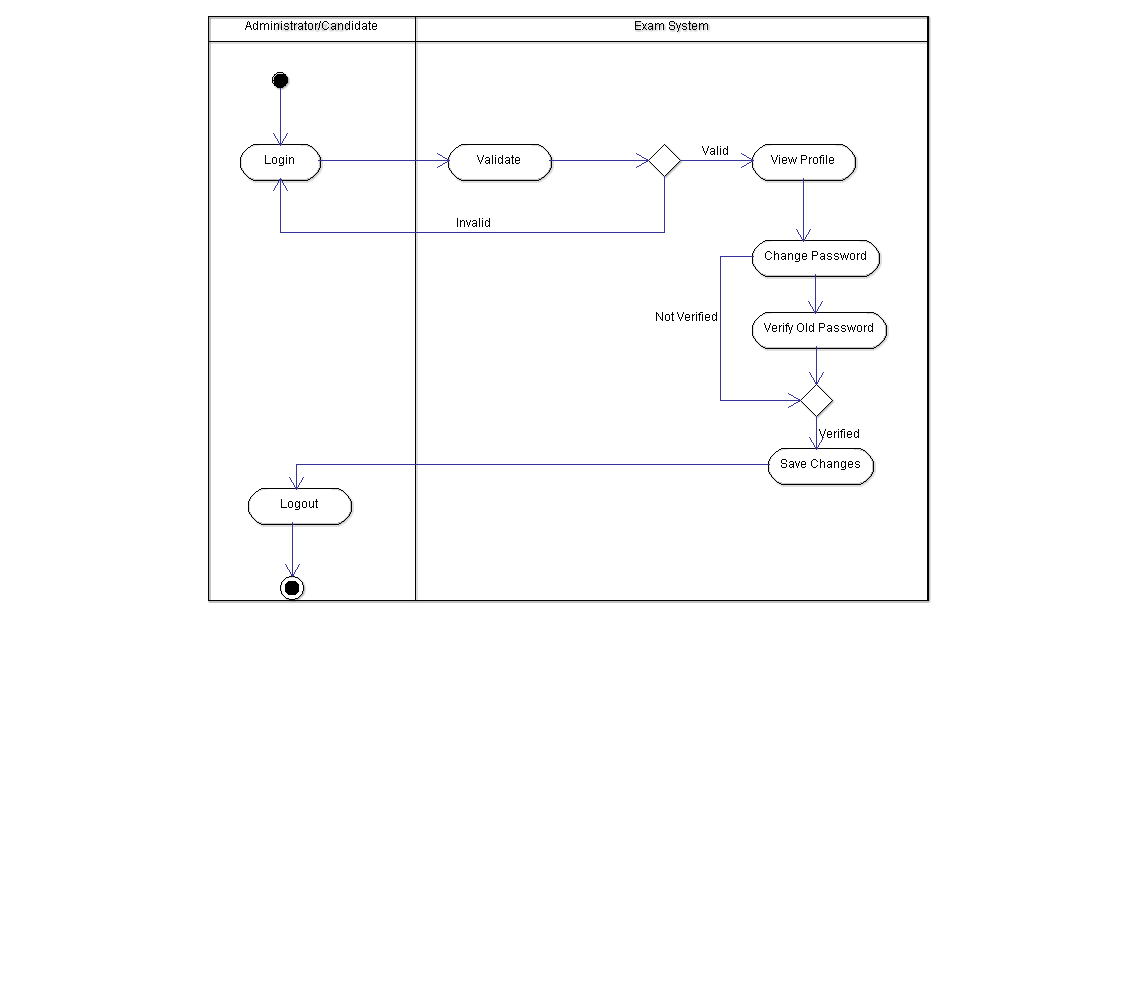


**3.9 Admin/Candidate- Change Password**

3.9.1 Sequence Diagram

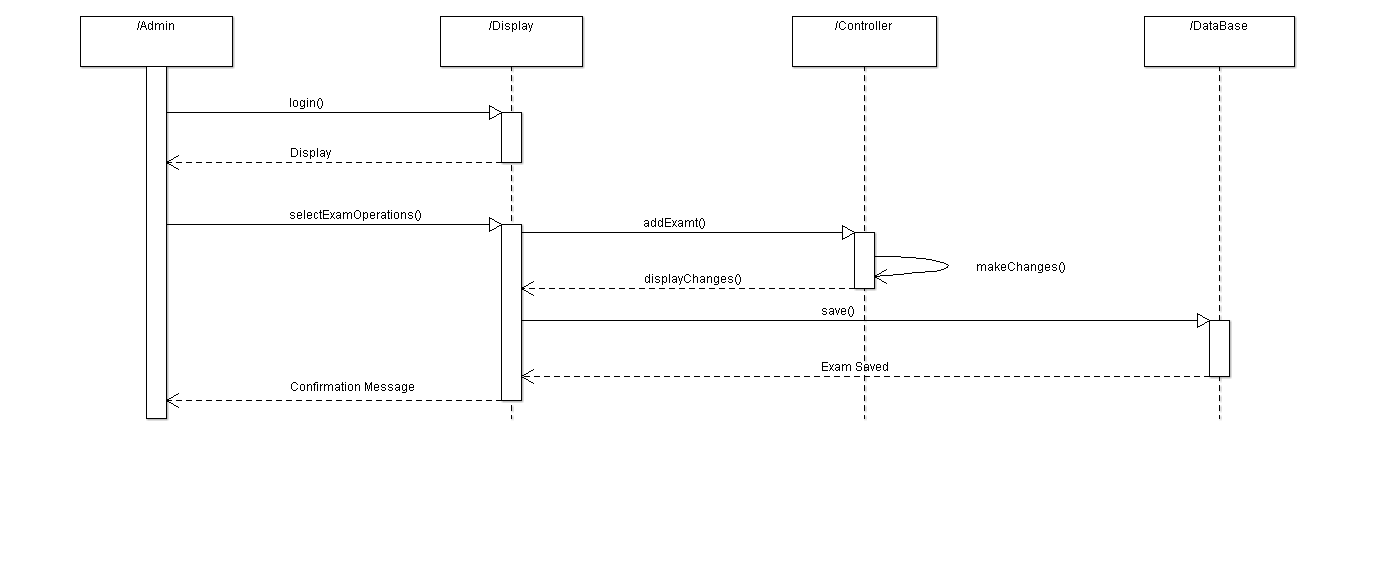


3.9.2 Activity Diagram

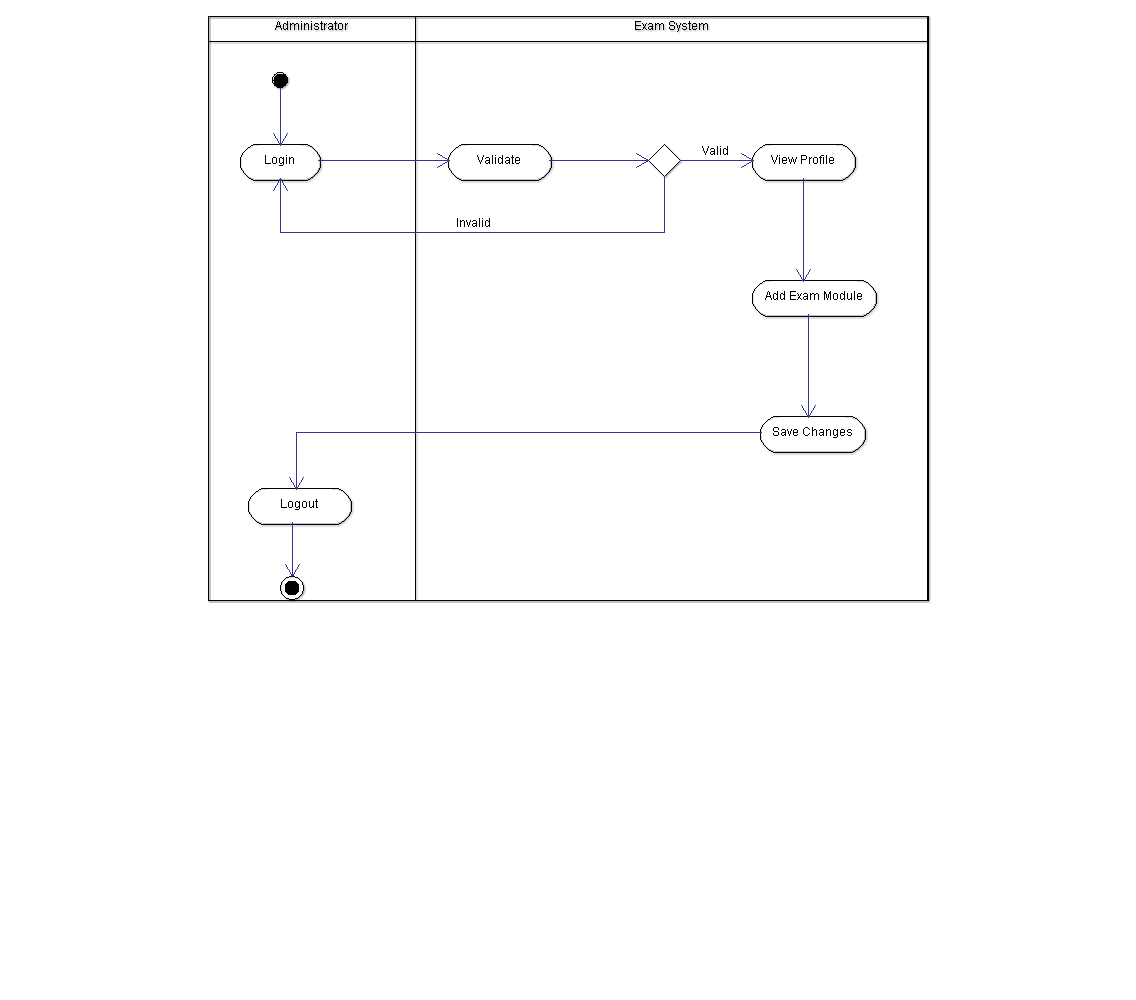


**3.10 Add Exam Module**

3.10.1 Sequence Diagram

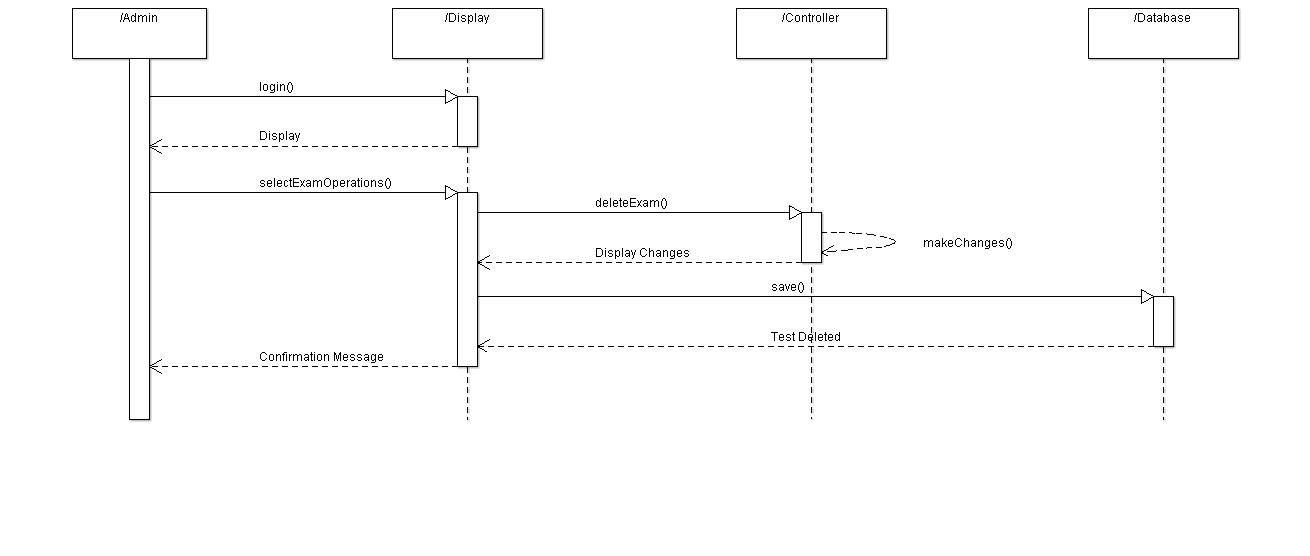


3.10.2 Activity Diagram

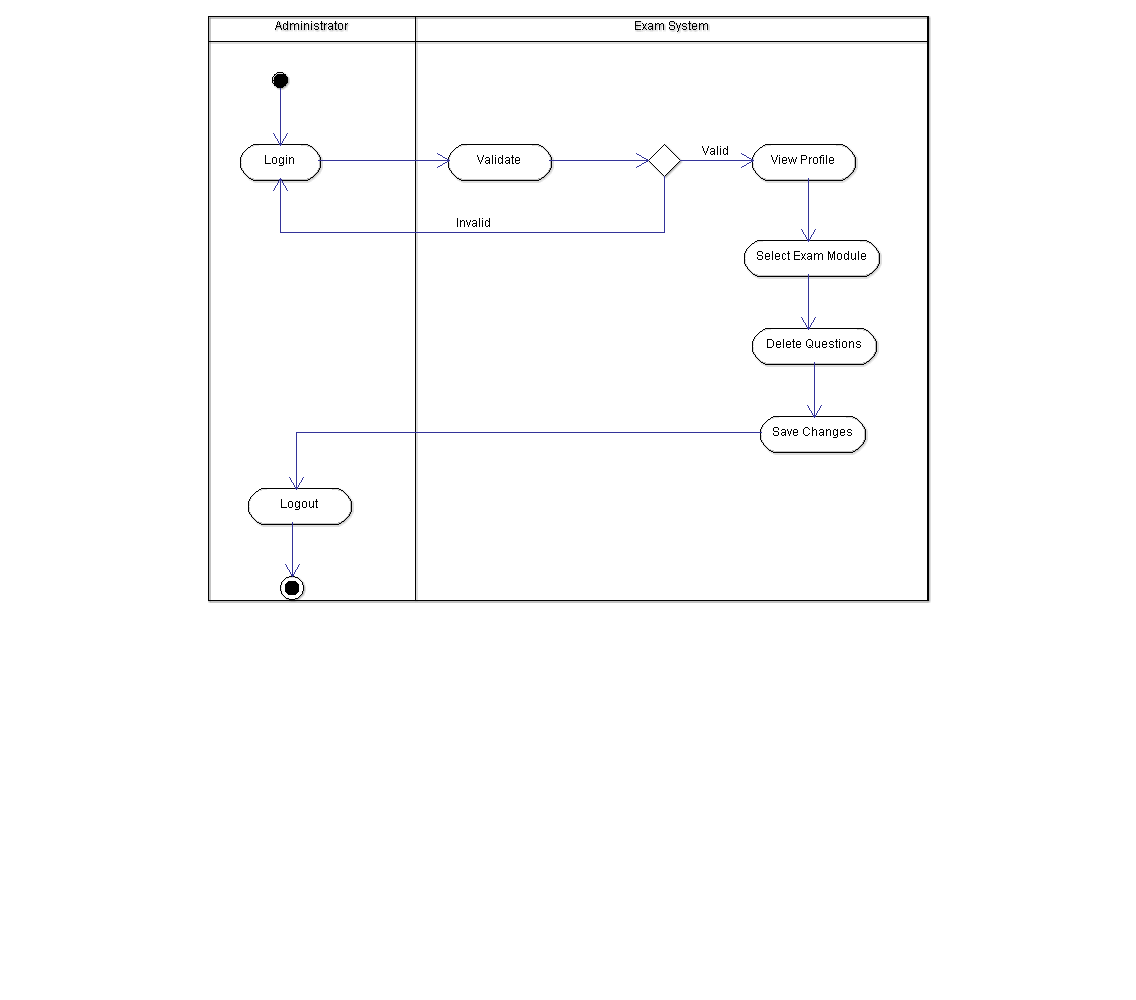


**3.11 Delete Exam Module**

3.11.1 Sequence Diagram

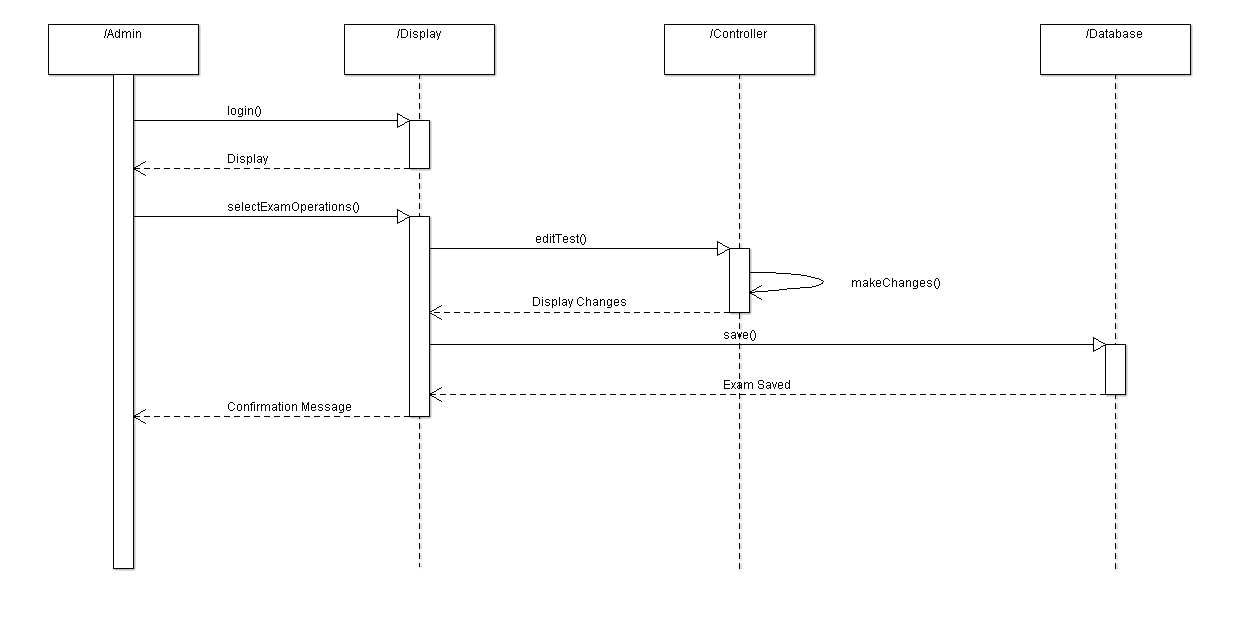


3.11.2 Activity Diagram

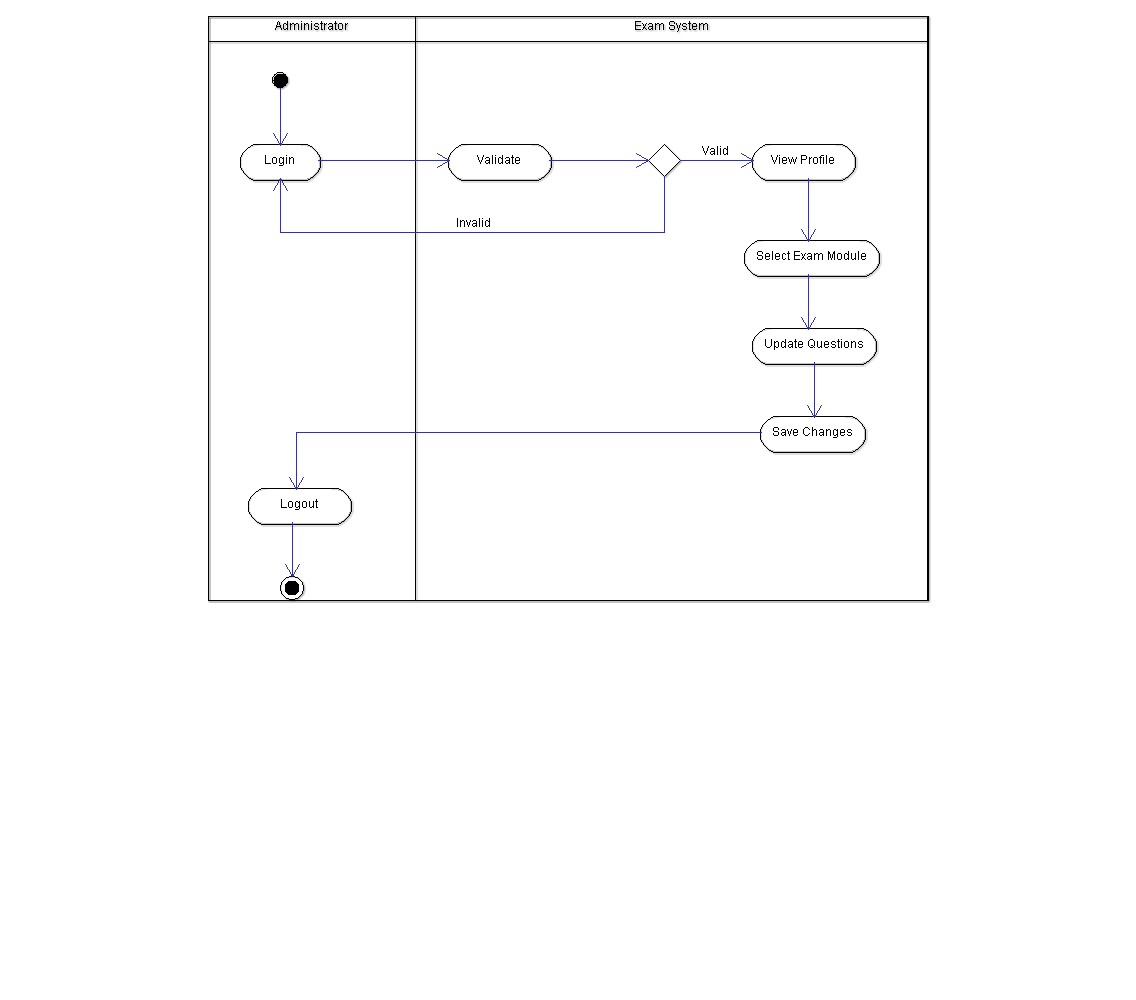


**3.12 Update Exam Module**

3.12.1 Sequence Diagram



3.12.2 Activity Diagram



**4. Execution Architecture**

Runtime environment required for the Online Exam Portal is any device with a web browser and an internet connection. The deployment platform used is Angular.

**4.1 Reuse and relationships to other products**

N/A

**5. Design decisions and tradeoffs**

By using trade-offs we want to give small software projects a simple way to have an efficient decision making process. This is an important criteria for us, since we want to use methods that are well known, and easy and intuitive to understand and use. We limited the number of functionalities as we made a trade-off between “sooner-but-worse” and “later-but-better”. AngularJS was used because it is a structural framework for dynamic web apps that makes much of the code you would otherwise have to write completely redundant. It is easy to integrate third party features with AngularJS as Angular integration comes pre-built into frameworks.

The decision to use SpringBoot was because it completely uses  new development model to make Java Development very easy by avoiding some tedious development steps and boilerplate code and configuration and increases productivity.

**6. Pseudocode for components**

**7. Appendices (if any)**

N/A