Cairo University

Faculty of Engineering

Computer Department

CUFE Students

Software Design Specification (SDS)

Version: 1.0

CM Identifier: sds_v1.0

Team 2

Names:

- Mohamed Sayed
- Mohamed Taha
- Riad Adel
- Ahmed Salah
- Abdallah Mustafa

Revision History

Sl. No.	Prepared/ Modified by	E-mail	Version	Date	Approved by	Descriptions/ Remarks
1.	Mohamed Taha	Mohamed.taha.177@gmail.com	1.0	April,10th, 2018		Added section 5, 3.1.
2	Mohamed Sayed	m.assr299@gmail.com	1.0	April, 10th, 2018		Added section 1.
3	Abdallah Mustafa	callmenssar@gmail.com	1.0	April, 10th, 2018		Final Revision.
4	Riad Adel	Riad.adel22@gmail.com	1.0	April, 11th, 2018		Added section 3.2, 3.3.
5	Ahmed Salah	Ahmedkin3030@gmail.com	1.0	April, 11th, 2018		Added section 2, 4.

Distribution list

Name	E-mail	Notes
Eng. Ali Elseddeek	alielseddeek@gmail.com	Response is required

Table of Contents

List of Tables

List of Figures

1. Introduction

This document illustrates the Software Design Specification (SDS) process.

The general purpose of an SDS document is to pave the way towards the implementation stage to make life easy for developers and provide a visual interface for the rest of the team members so that open discussions are made and communication becomes easy between members with different perspectives.

1.1 Purpose of this Document

This document will define the design of "CUFE Students" application. It contains specific information about the expected input, output, and functions. The interaction between classes to meet the desired requirements are outlined in detailed figures.

1.2 Scope

At this iteration, the app allow the user to login to their account which they can
-See their personal Info
-See information about the departments in the faculty
-See their last results
-See their term classwork
-See any warning sent to them by the faculty
The user can also logout from the app and get his password if he/she forgot it.

1.3 Table of Acronyms and Definitions

Term	*1.3.1 Definition*
APP	APPLICATION

1.4 References

1.5 Overview of Document

The remainder of this document is divided into 5 chapters

chapter 2 provides information about the system architecture, as it decomposes the system into subsystems.

Chapter 3 goes over the design patterns used in the project, class diagrams modeling the relationships between each class of the system, and sequence diagrams showing how the developer would go about implementing a certain feature.

Chapter 4 shows an ER diagram of the database used by the app .

chapter 5 models how the system will be deployed using deployment diagrams.

Finally , Chapter 6 shows a traceability matrix mapping design elements to user requirements from the SRS document <Gmn_SE02_v1.0>

2. System Architecture

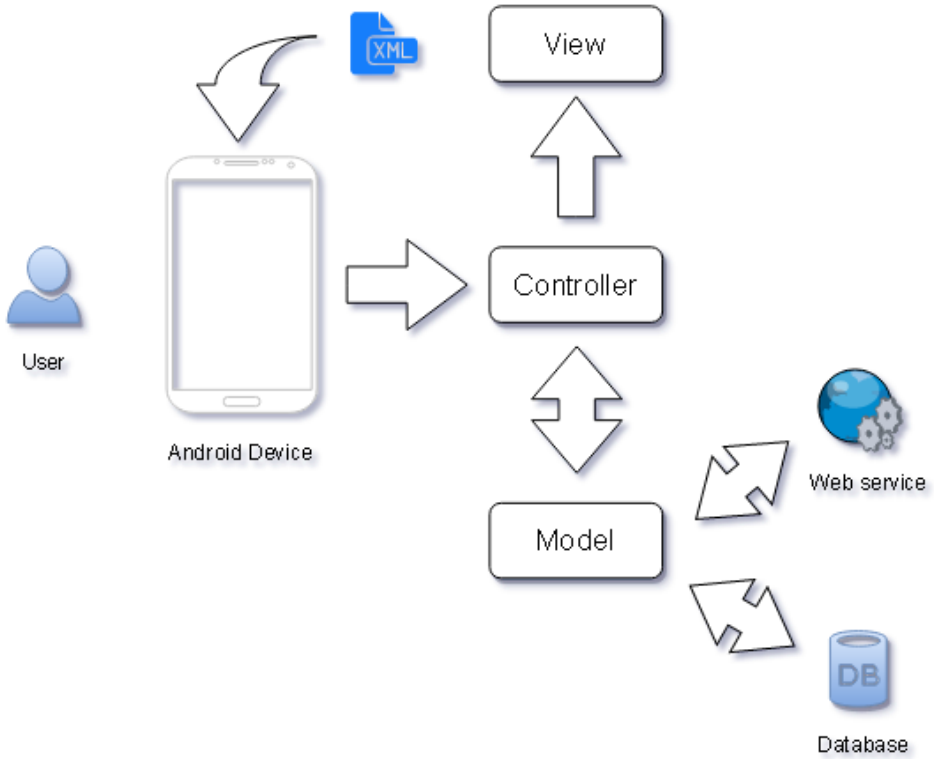

Figure 1.1 – System Architecture

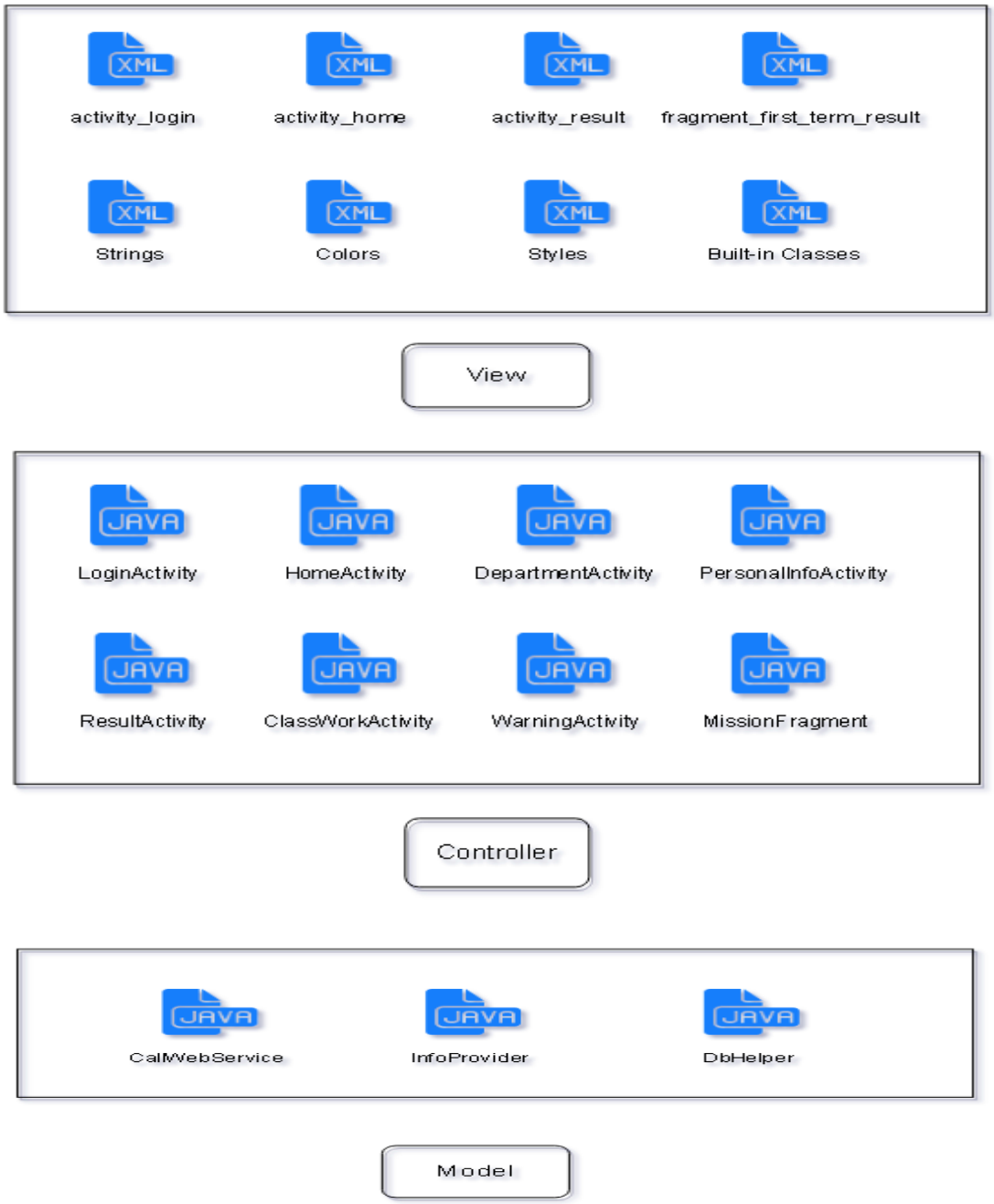

Figure 1.2 – Detailed System Architecture

The view is the xml files which displays user-interface on the android device; it includes activities layout, Strings, Colors, Styles and built-in classes like buttons… Etc.
The Controller is a set of java classes that handles the user actions; it includes activities and fragments classes.
The Model is a set of java classes that calls the web service and access the database.

3. Design Models

3.1 Design Patterns Description

3.1.1 ***Singleton:*** *is a software design pattern that restricts the instantiation of a class to one object. This is useful when exactly one object is needed to coordinate actions across the system. The concept is sometimes generalized to systems that operate more efficiently when only one object exists, or that restrict the instantiation to a certain number of objects. Class Student is our Singleton, it holds the user's data through the whole log in time.*

3.1.2 ***Factory Method:*** *Define an interface for creating an object, but let subclasses decide which class to instantiate. Factory Method lets a class defer instantiation to subclasses. Class getStudentFactory creates an object of SemesterStudent or CreditStudent based on the user's department.*

3.1.3 ***Adapter:*** *is a software design pattern (also known as Wrapper, an alternative naming shared with the Decorator pattern) that allows the interface of an existing class to be used as another interface. It is often used to make existing classes work with others without modifying their source code. Multiple adapter classes are used to in application.*

3.2 Class Diagrams

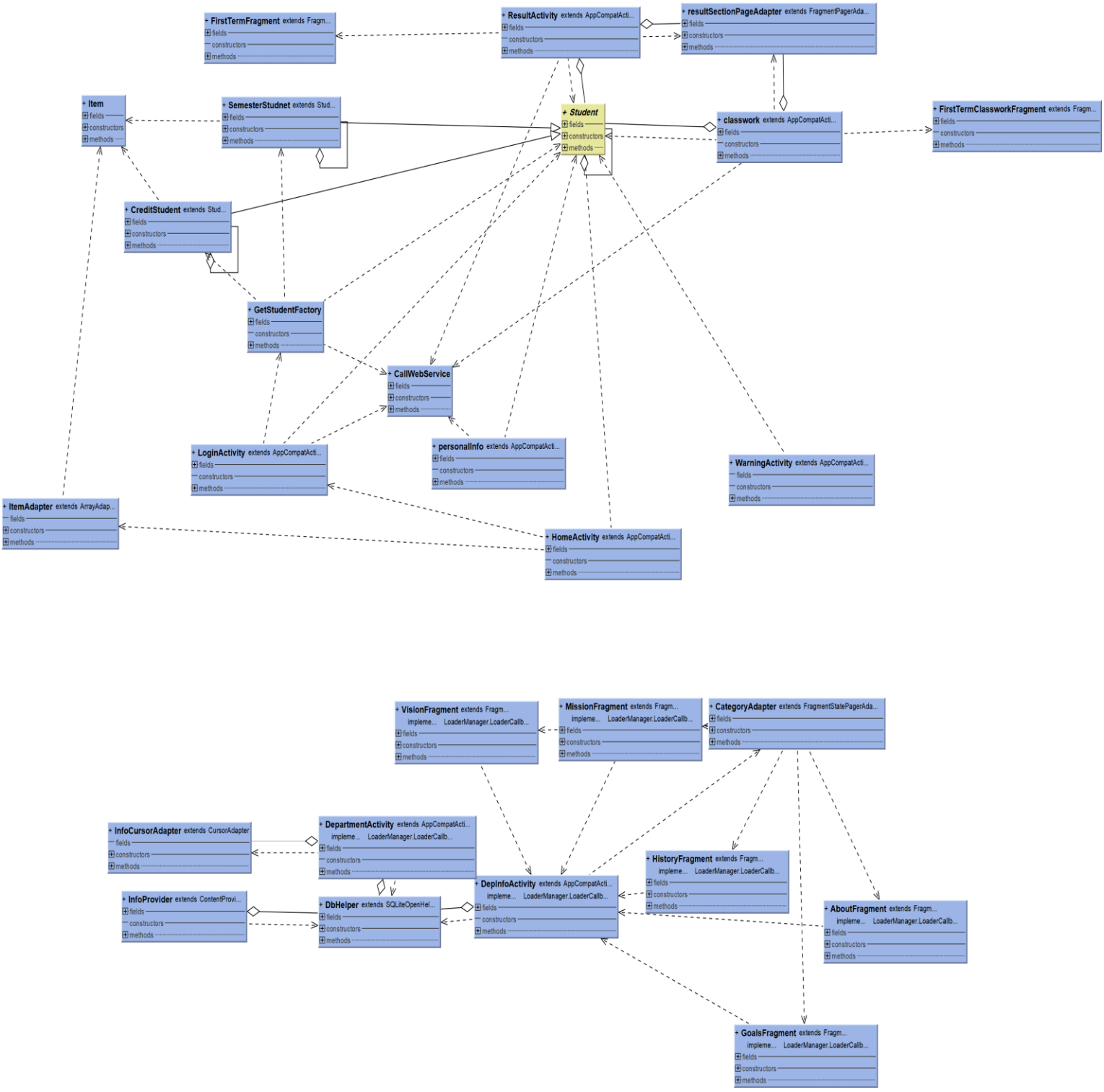

Figure 2 – Class Diagrams

3.3 Interaction Diagrams

SD1:

Figure 3– LoginActivity

SD2:

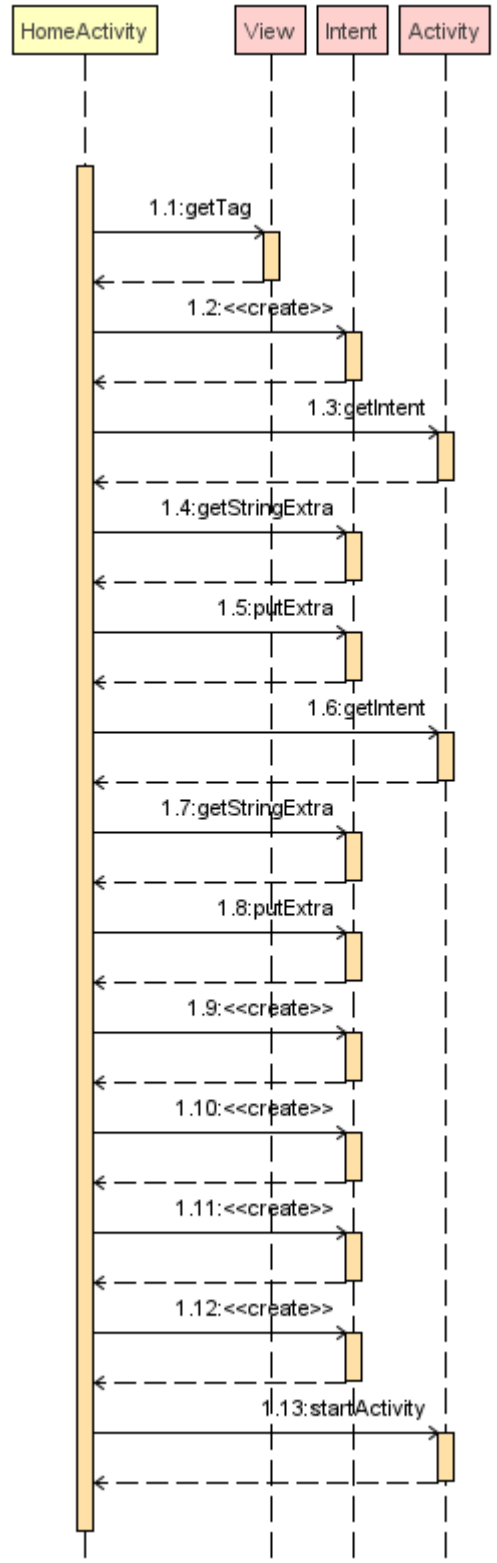

Figure 4 – HomeActivity

SD3:

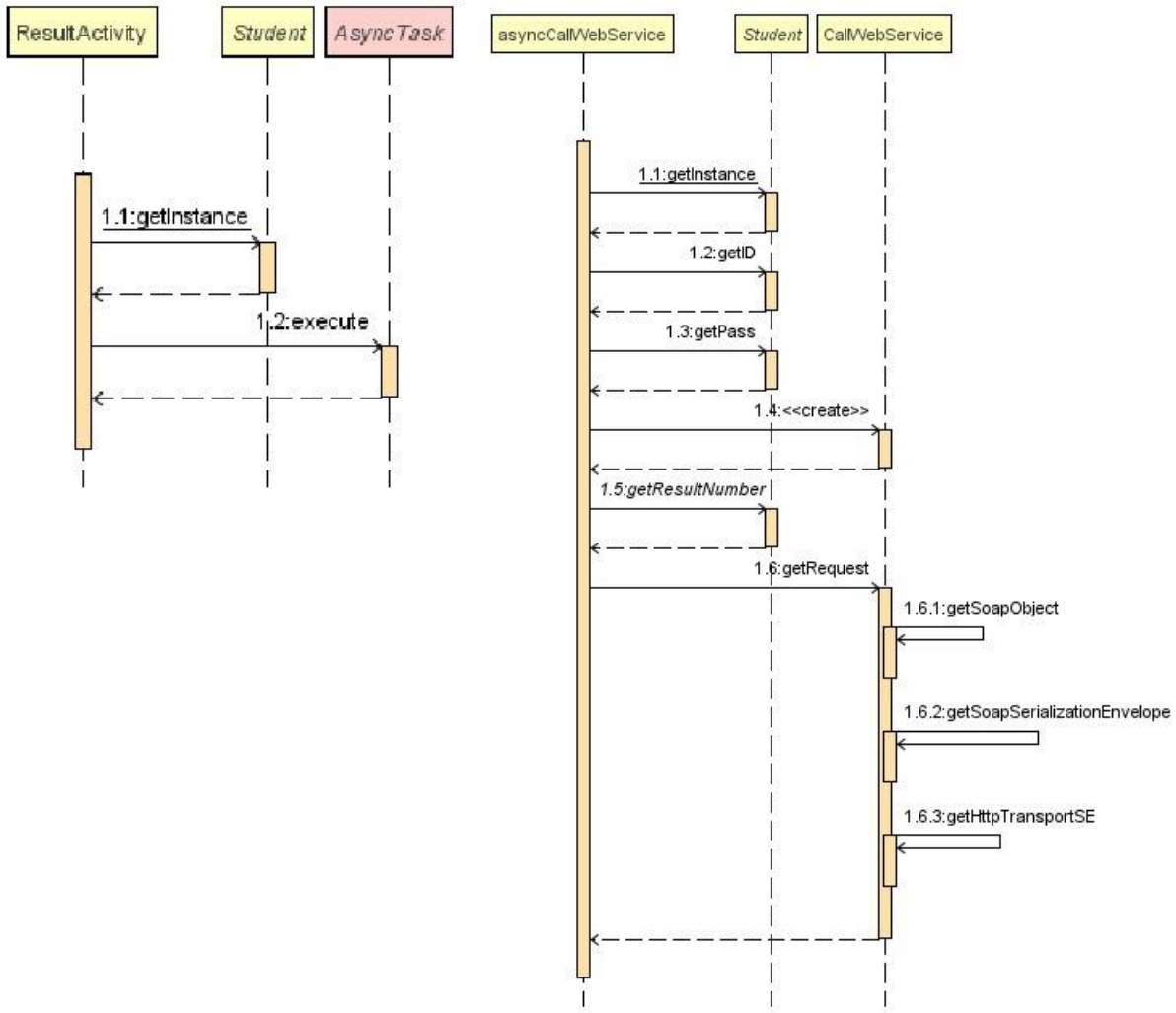

Figure 5 – ResultActivity

SD4:

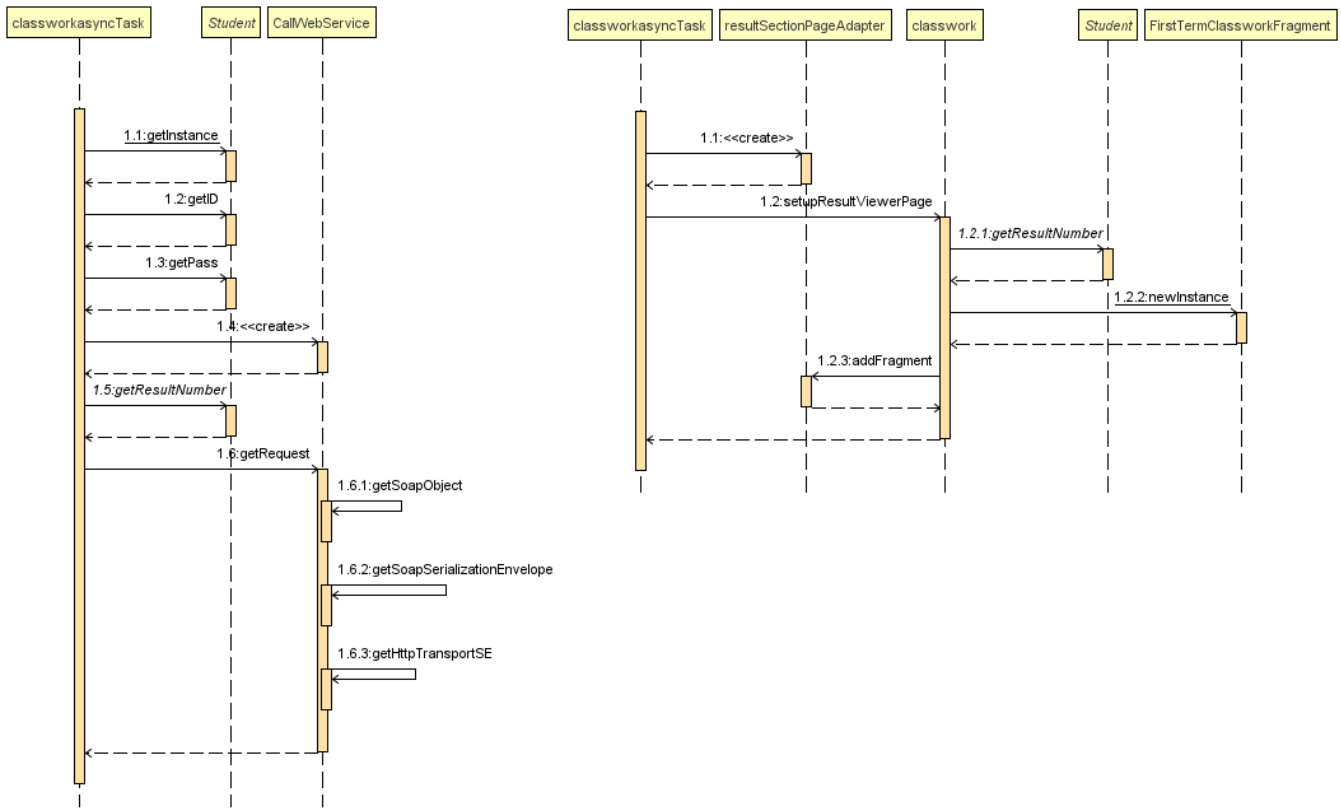

Figure 6 – classworkActivity

SD5:

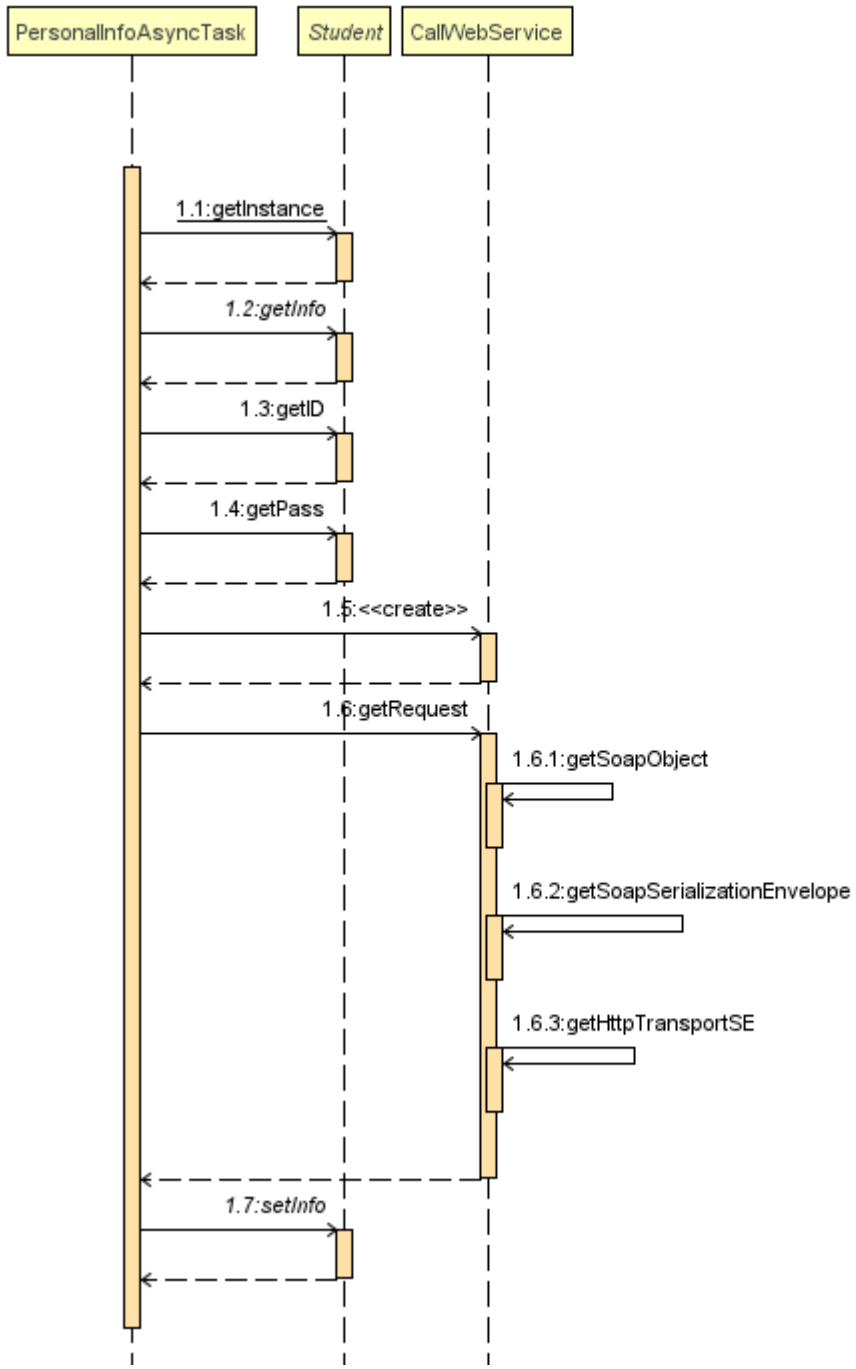

Figure 7 – PersonalInfoActivity

SD6:

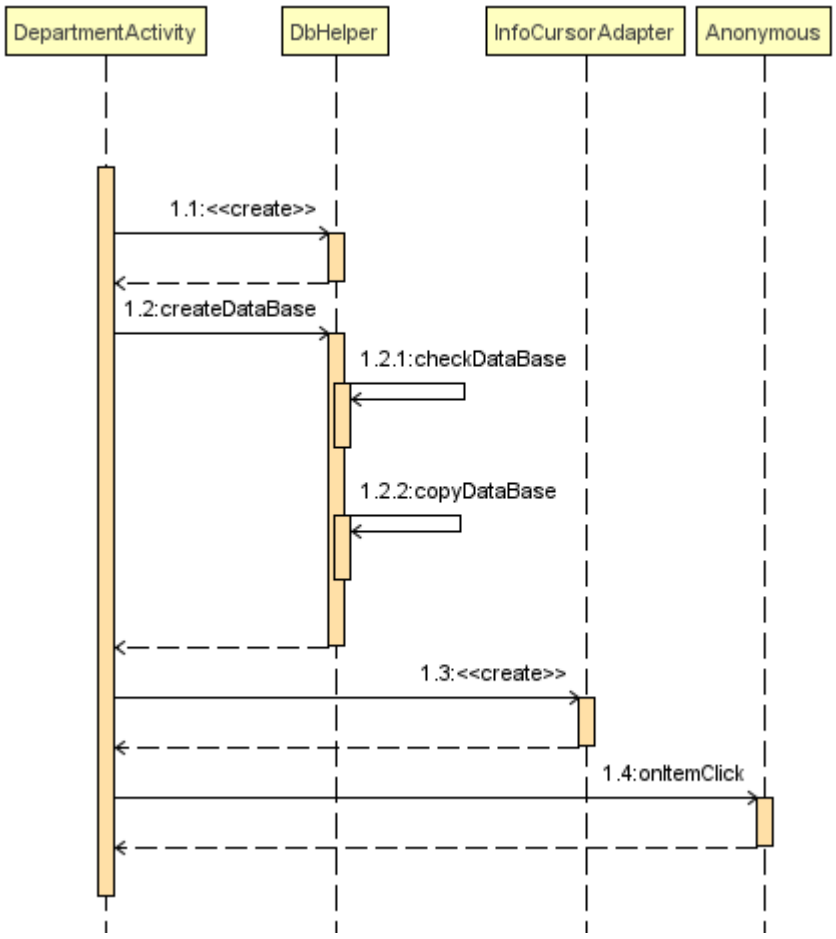

Figure 8 – DepartmentActivity

SD7:

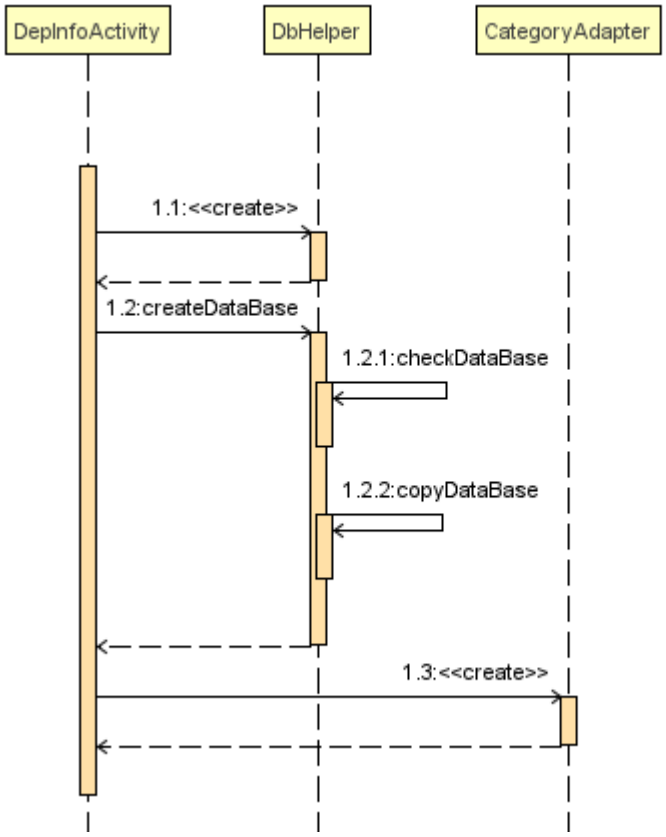

Figure 9 – DepInfoActivity

4. Data Models

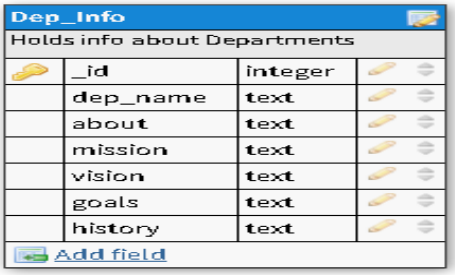

Figure 10 – Data Model

5. System Deployment

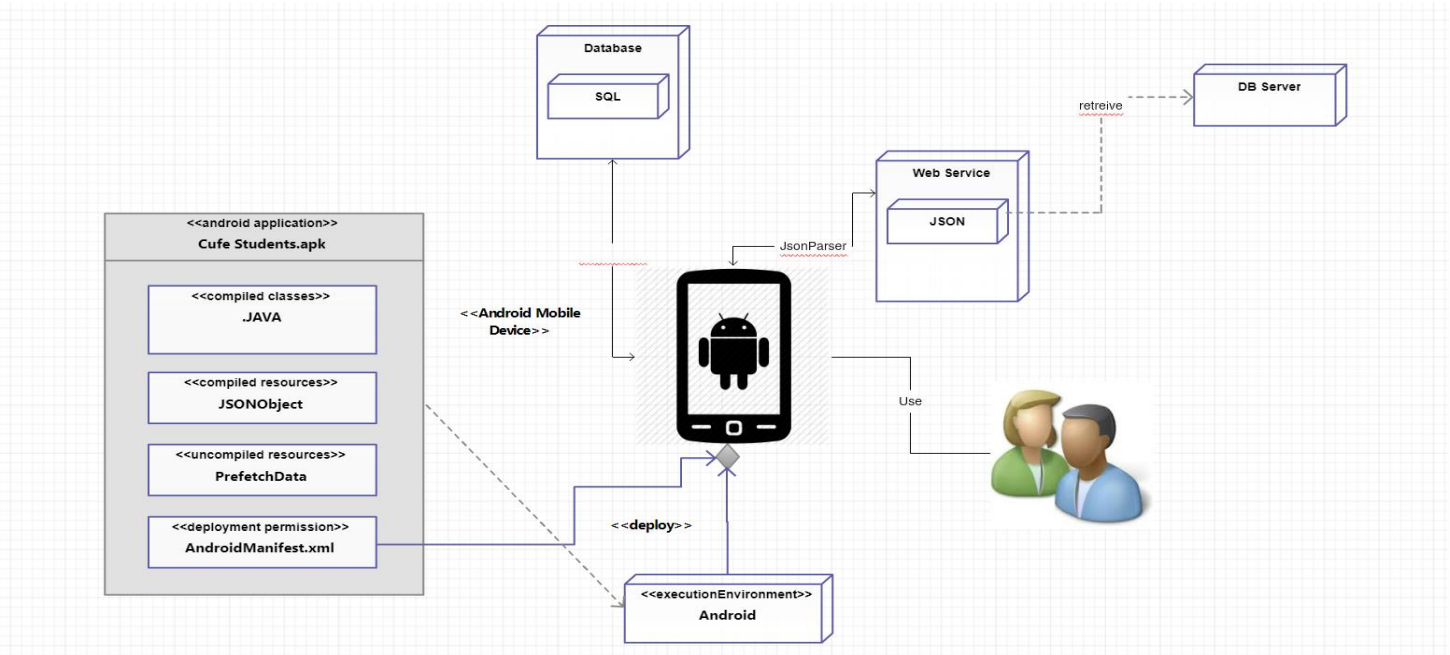

Figure 11 – System Deployment

6. Traceability to Requirements

SD1(*LoginActivity***)**	**UC9 (***Login***)**
SD2(*homeActivity***)**	**-**
SD3(*ResultActivity***)**	**UC1 (***Result***)**
SD4(*ClassWorkActivity***)**	**UC1 (***Result***)**
SD5(*PersonalInfoActivity***)**	**UC4 (***PersonalInfo***)**
SD6(*DepartmentActivity***)**	**UC8(***Departments***)**
SD6(*DepInfoActivity***)**	**UC8 (***Departments***)**

Table 1 – Traceability to Requirements