**SDLC Approach**

**Incremental Development**

**Taysser Project**

2/27/2022

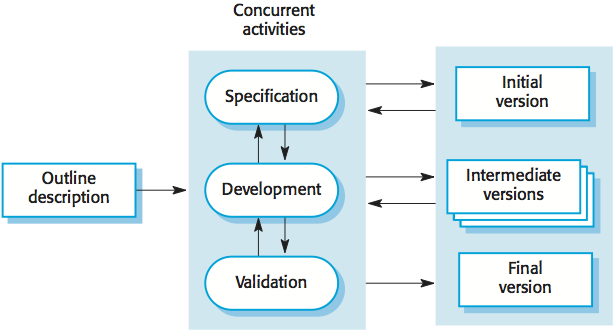
**Errors Team**

**1. Incremental Development Approach**

**1.1. Introduction**

When it comes to build a new software system, the project team starts to choose an appropriate SDLC approach depending on several factors such as time schedule, cost, clear user requirements, familiar technology, complexity, schedule visibility, advantages and disadvantages of each approach and how the approach can fit with the system nature. So accordingly, we have chosen incremental development approach.

**1.2. What is Incremental Development?**

Incremental development is a method of building software products in which a system is built piece-by-piece. The final requirement specification is clear from the beginning, and everyone knows the end result clearly. The system is broken down into small sub-systems which are designed, built and tested independently. This allows partial utilization of the product, but the full system isn't usable unless until the development is entirely done.

A good analogy to understand this model is looking at how a mason builds a wall. How the final wall should look like is already clear in their mind, and starting from zero they lay out the wall brick by brick. The wall becomes fully usable only when the construction is entirely done.

**1.3. Why Incremental Development?**

* We have chosen incremental approach as it is appropriate for a small-time range project and this in line with our project.
* Costs are reduced if we want to change something in the system according to customer requirements.
* We can keep up with customer easily to get his feedback on development work that has been done.
* It is possible to deliver and deploy useful software to the customer rapidly.
* It requires less analysis and documentation that has to be redone.
* Customers will be able to gain value and experience and use system early.
* Each increment is a prototype to help elicit for later increments.
* Lower risk of overall project failure.

**1.4. Incremental Approach Problems and Solutions**

* **Problem:** It is important to produce regular deliverables to measure progress.
* **Solution:** We intend to use version control system such as GitHub to keep information about each increment and to have the ability to produce deliverables regularly.
* **Problem:** System structure will be corrupted as new increments are added and software changes become increasingly difficult and costly.
* **Solution:** It is important to perform refactoring regularly to improve the software.

**2. Requirements Gathering**

After we identified the problem, and before working on it, we had to measure the extent of people's interest in solving it, and whether it was really feasible through this solution or not, and what requirements were desired to be implemented through our solution.

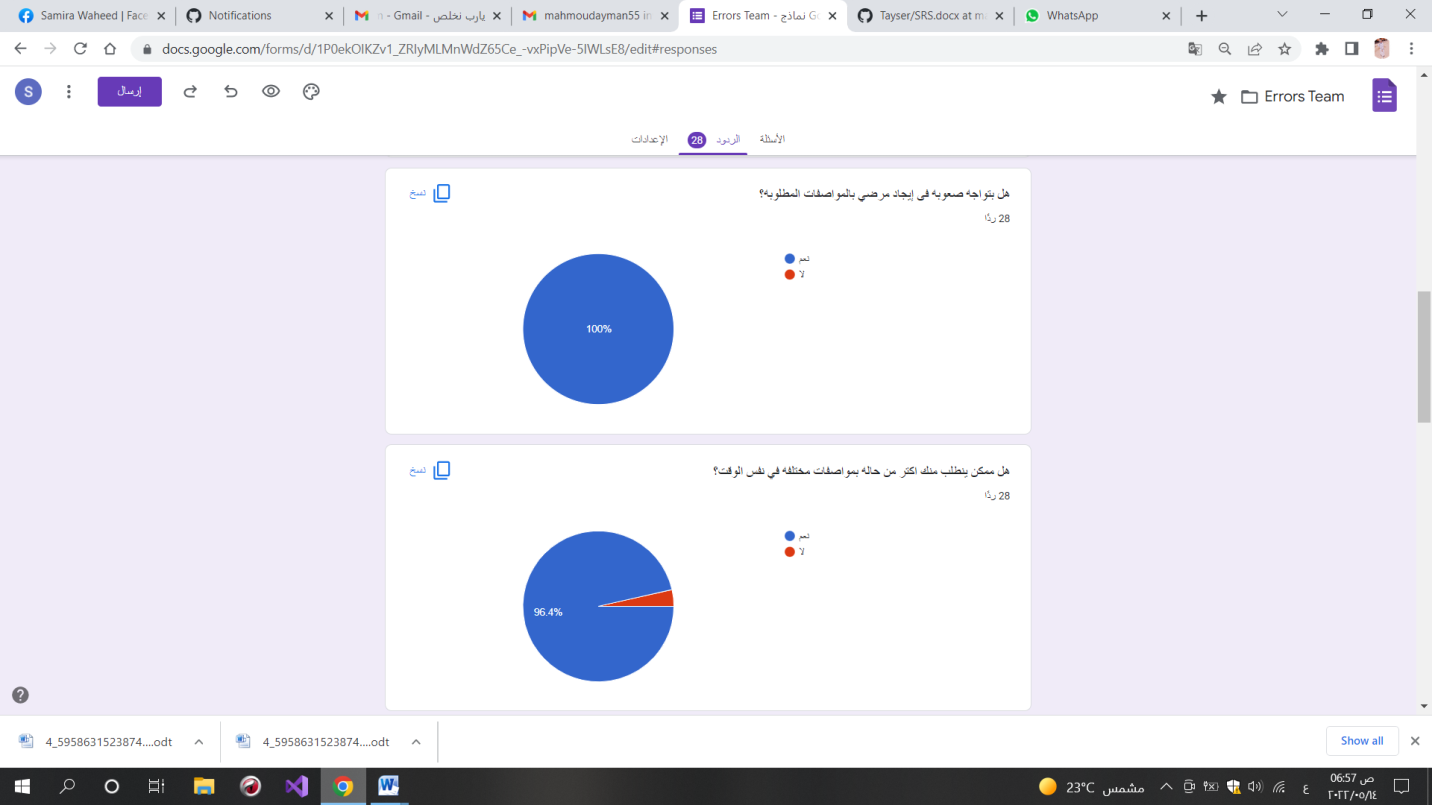
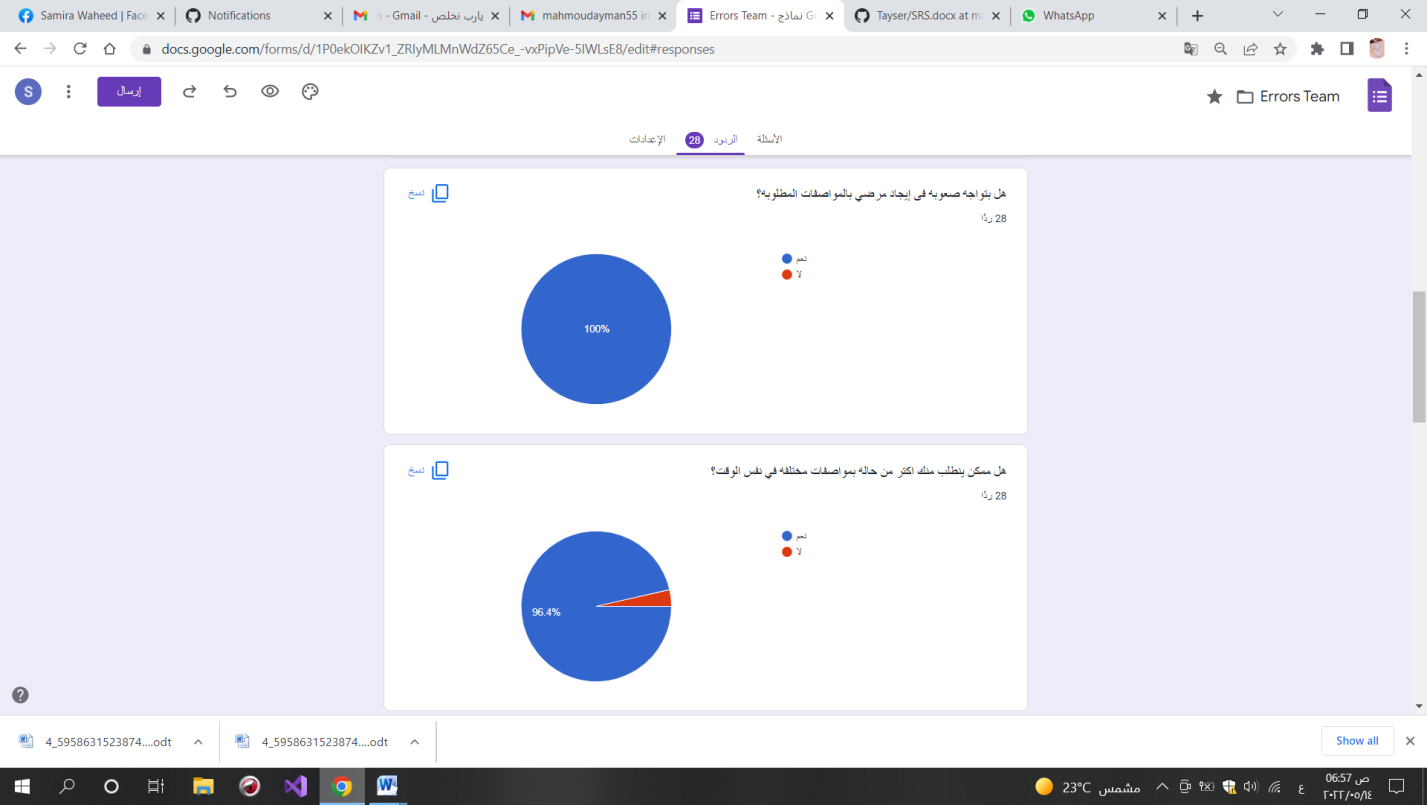
* **Questionnaire (**Electronic Questionnaire**)**

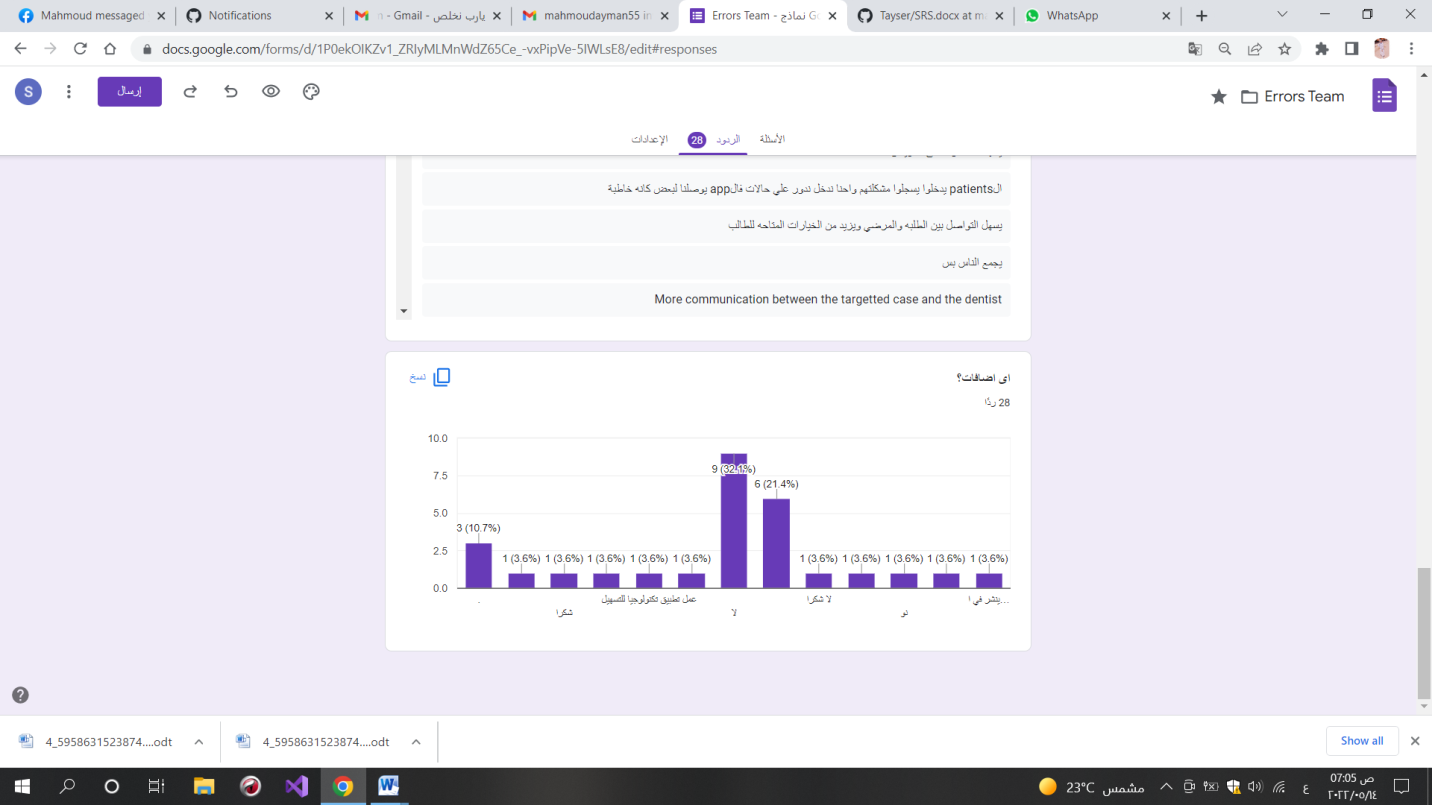
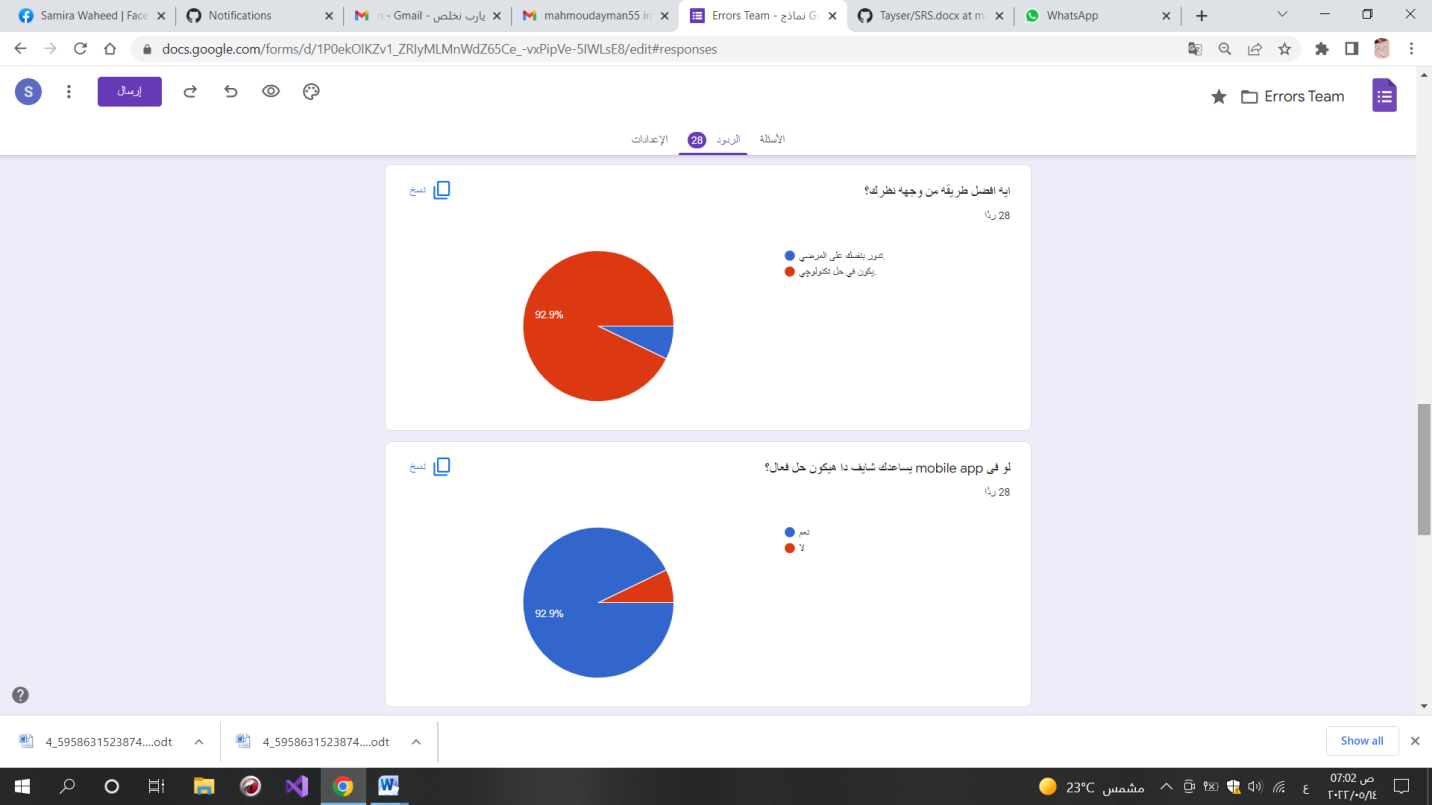
We chose to do a questionnaire with which we could collect as many opinions as possible with which we would measure the extent of the problem's impact on a large segment that cannot be ignored.

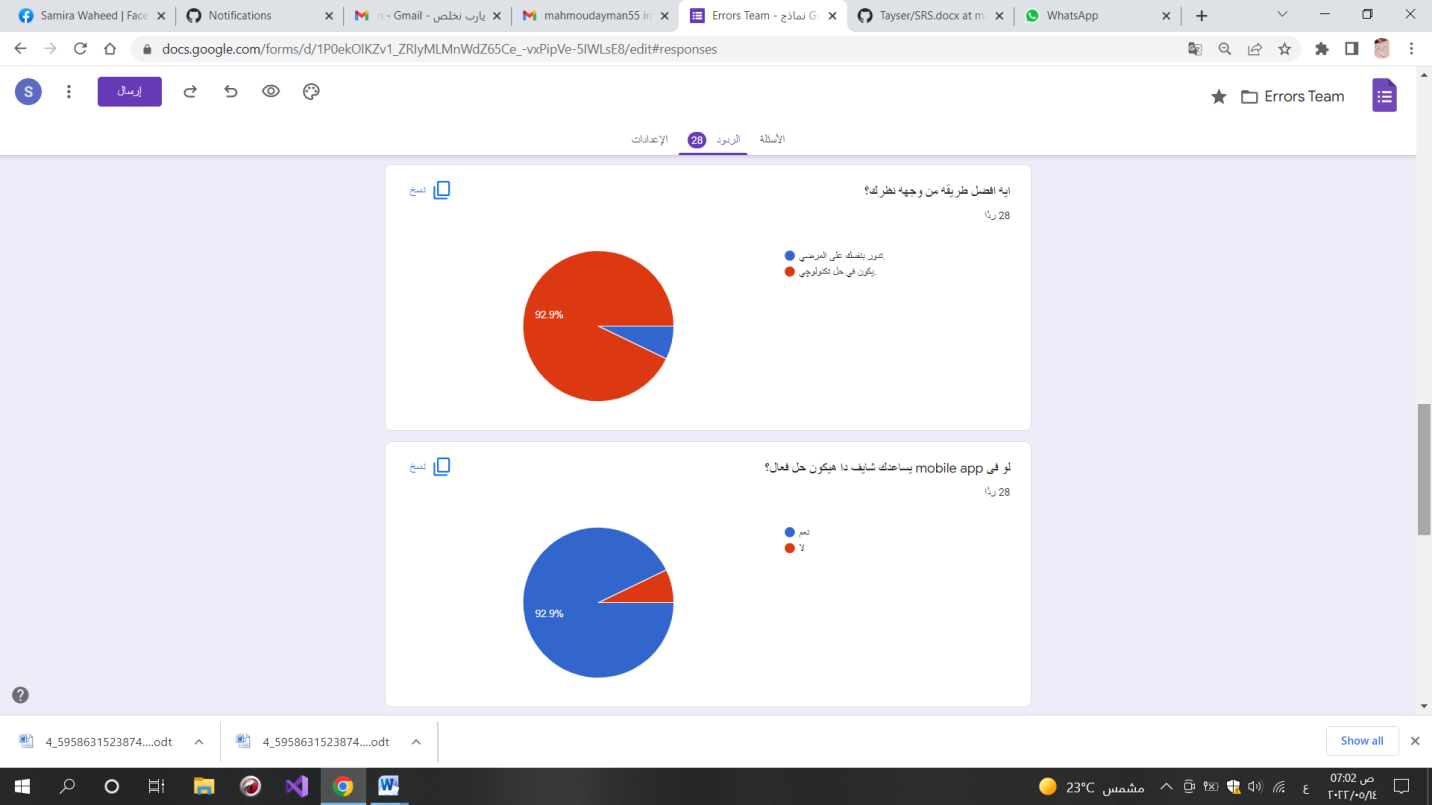
Questionnaires are well-designed and coherent questions, which aim to collect the data we want in a research; to achieve effective results; That is, the questionnaire is one of the methods of data collection, and it is applied to a sample taken from a particular community or environment...etc. The data we obtain from applying the questionnaire to the samples is representative of the community as a whole, or of the environment as a whole.

* **Questionnaire Features:**
* It leads to the collection of important information on the subject in question, at a low cost, and in a short time.
* Objectivity and impartiality in data collection.
* The ease of preparing the questionnaire, compared to other tools used in scientific research, such as interviews.
* Get highly accurate answers.
* Sample is given complete freedom, as members, aren’t mention their names.
* It gives us a huge amount of information.
* Not specifying a specific time for the answer, which keeps the sample from worrying.
* It can be applied to wide geographical areas.
* **Questionnaire defects and their solution:**
* Sometimes some members of the sample are unable to accurately choose the answers to the questionnaire questions, due to the complexity of the questions, the difficulty of their formulation, or the strangeness of their words.
* Therefore, the questions were developed with great care and a great deal of simplicity, so it is easy to understand and answer them.
* Provoking boringness to the sample if the questions are too many or too long.
* Therefore, the questions were most of the optional type, directed to facilitate dealing with them.

The questions were summarized in knowing the problems faced by the students and their personal suggestion for solutions to help us identify and reduce these problems by mobile application. The questions and responses to them were as follows:





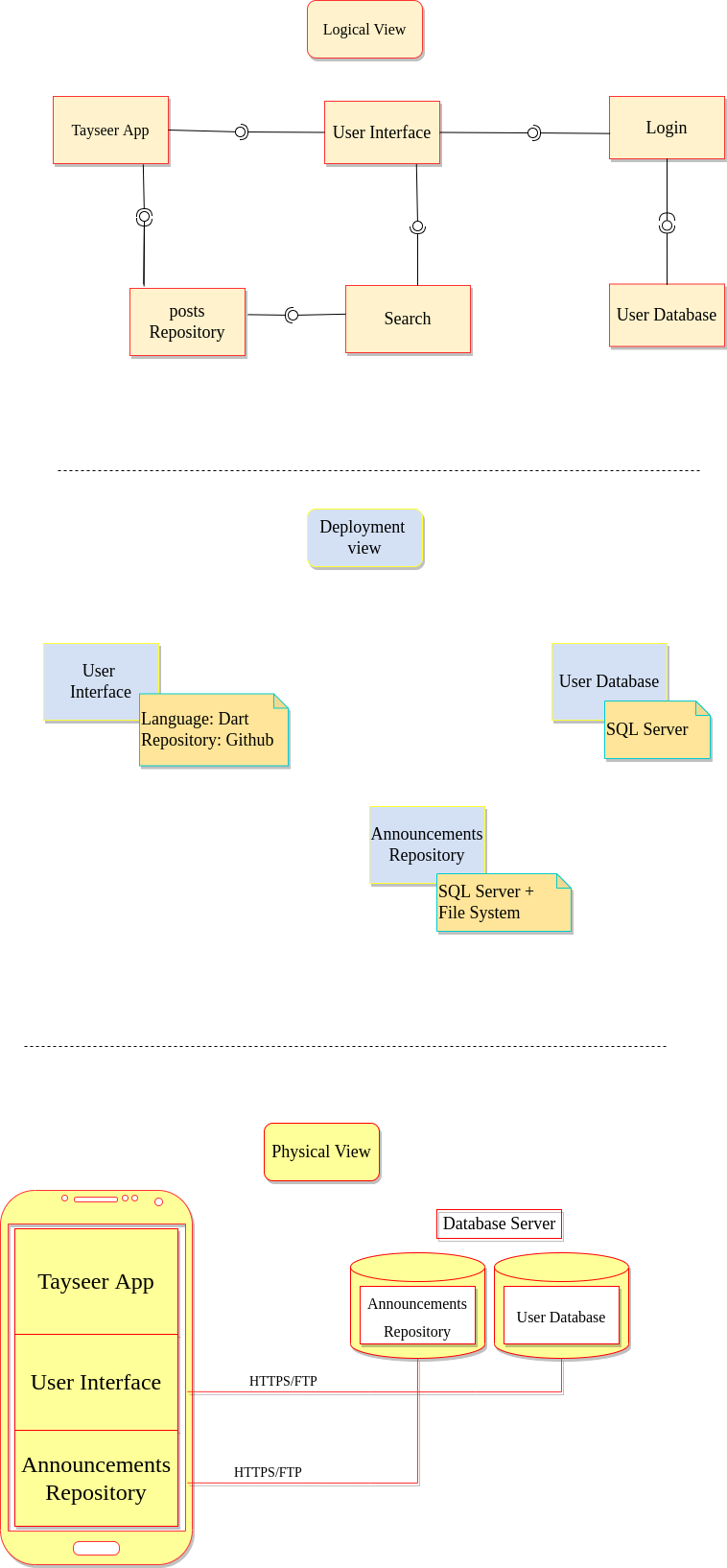
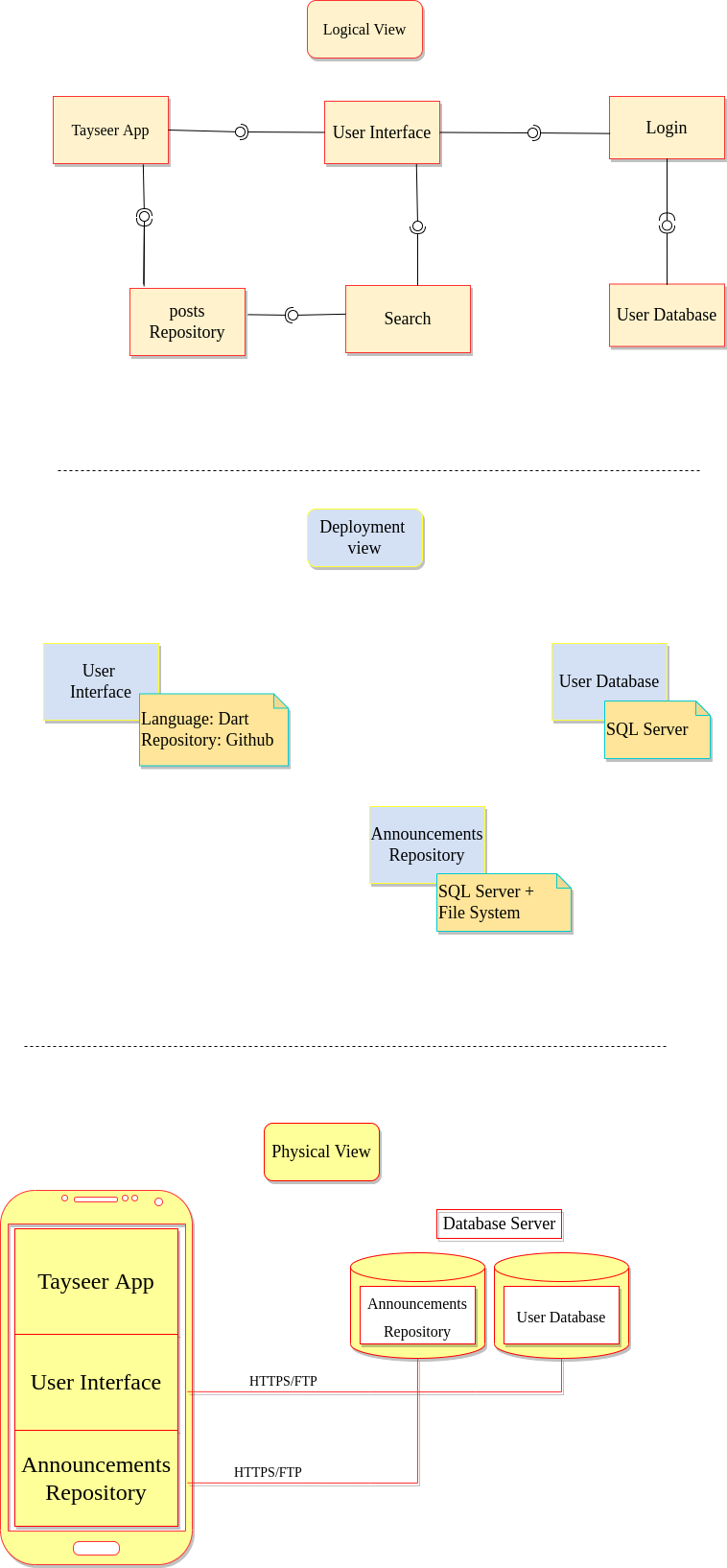
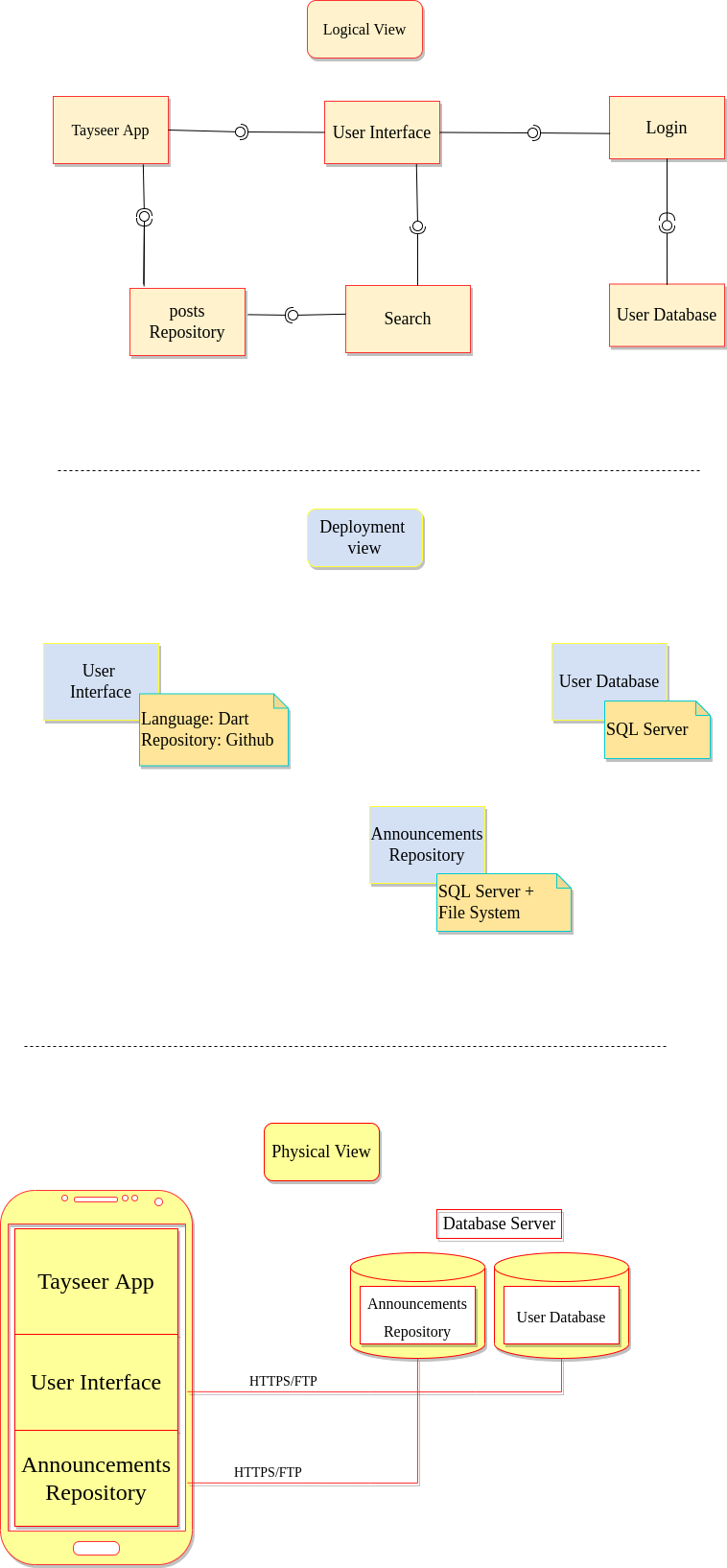


This is not only all the responses, but all of them are attached to the Excel file companion to project files. Based on all the responses that reached us through the questionnaire, we identified all the project requirements that serve this sample and more, and for an innovative and effective solution, all of this is included under the title Functional and nonfunctional requirements. We recorded all of this in “IEEE Template” in points 4 and 5, also companion to project files you can see it now.

**3. Architecture Views**

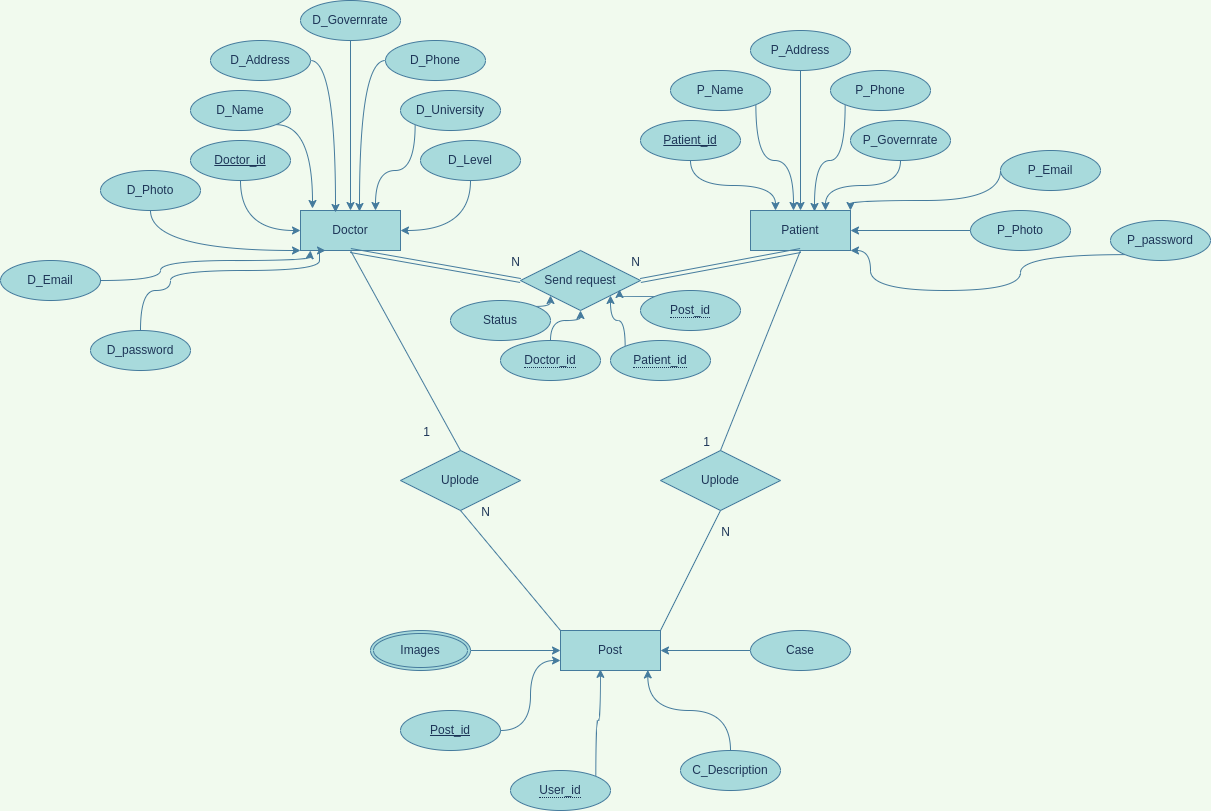
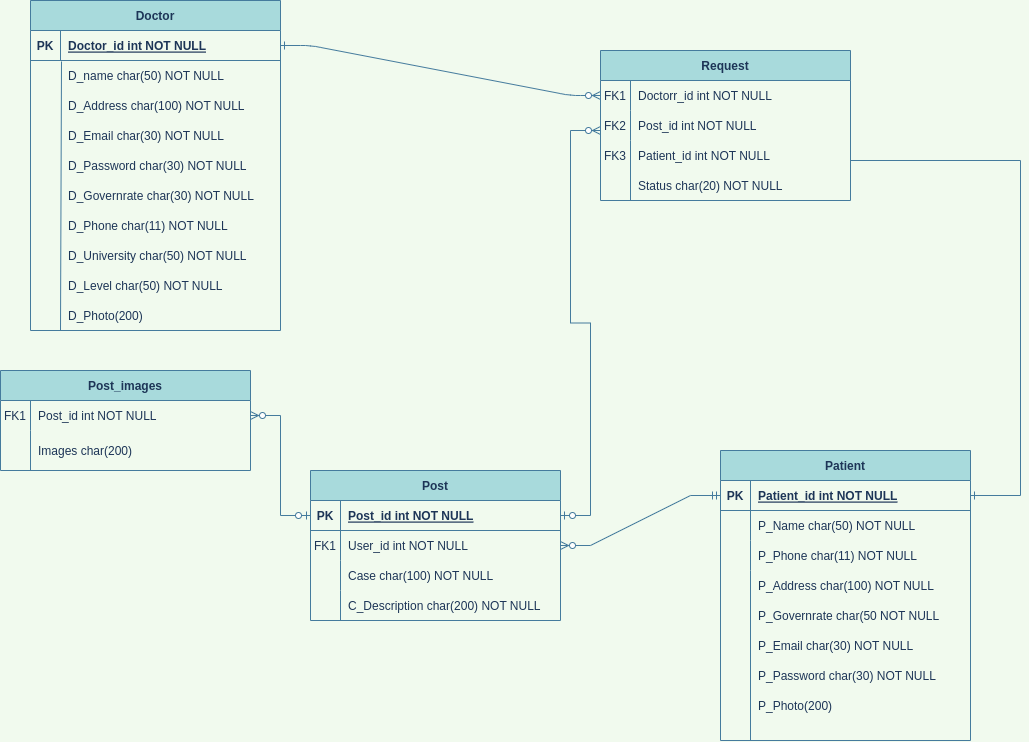
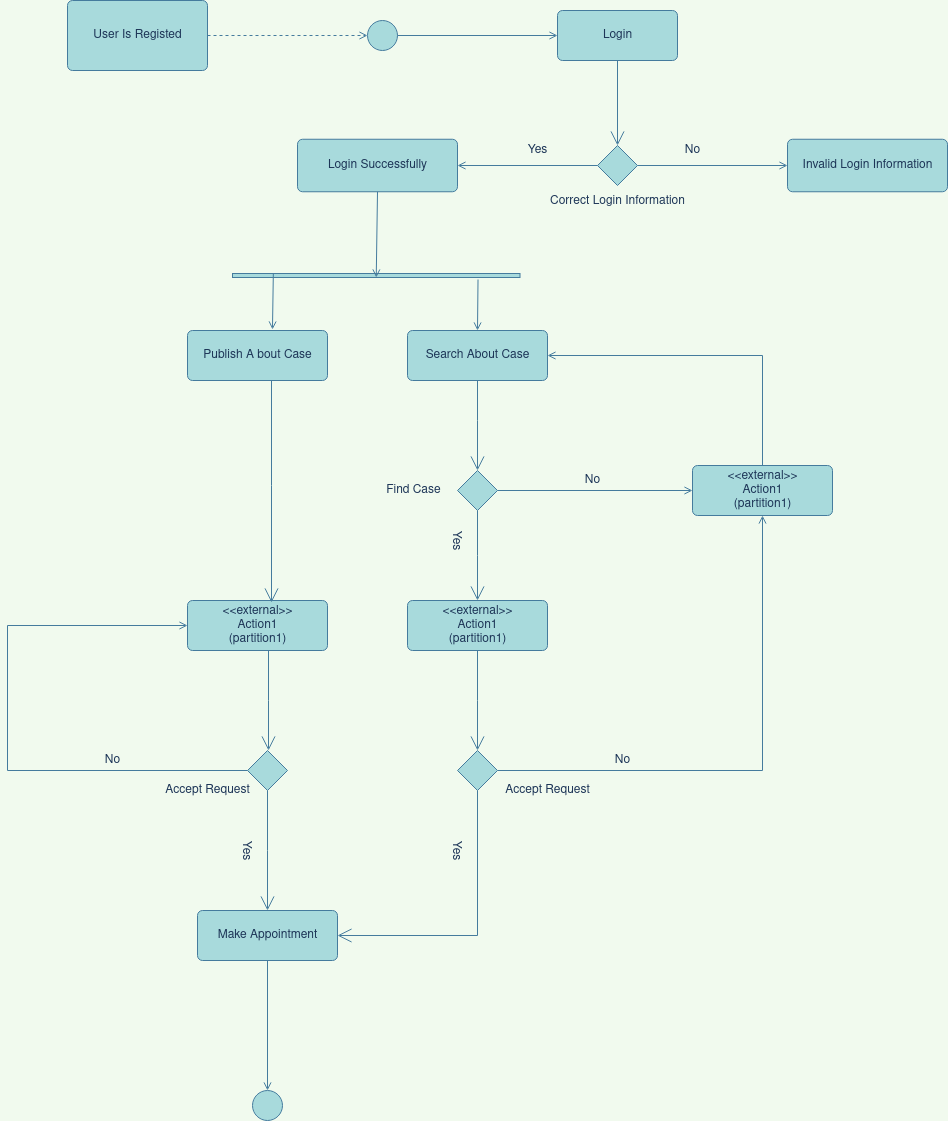
It is an architecture verification method for studying and documenting software architecture design and covers all the aspects of software architecture for all stakeholders. It provides there essential views −

* **The logical view**: It describes the object model of the design.
* **The deployment view**: It describes the activities of the system, captures the concurrency and synchronization aspects of the design.
* **The physical view**: It describes the mapping of software onto hardware and reflects its distributed aspect.

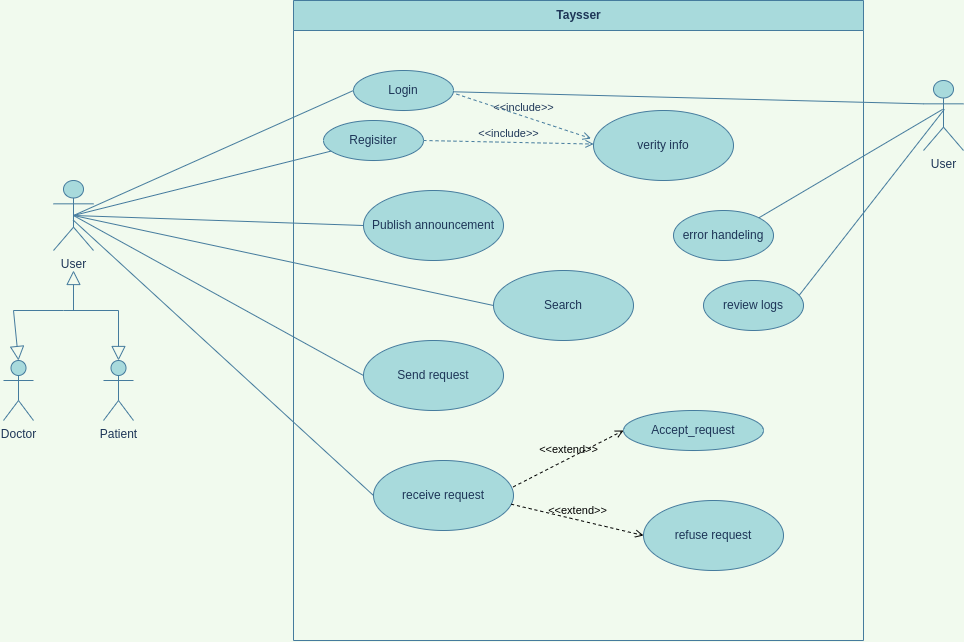
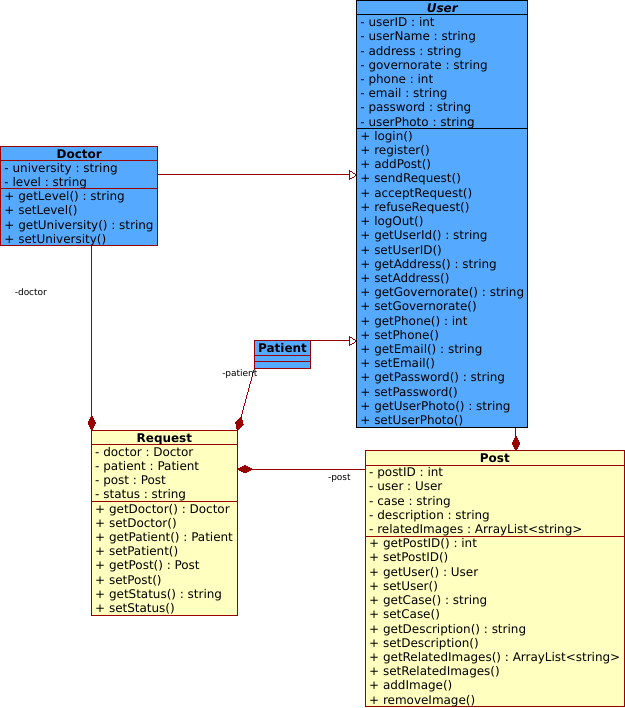
1. **Logical view**
2. **Deployment view**
3. **Physical view**

**4. UML Diagrams**

UML is a way of visualizing a software program using a collection of diagrams. The current UML standards call for different types of diagrams,These diagrams are organized into two distinct groups:

* **Behavioral UML diagrams**
* **Eintity relationship diagrams (ERD)**
* **Schema diagram**
* **Activity diagram**
* **Sinariou diagram**

|  |  |  |
| --- | --- | --- |
| Use Case Name: Publishing Announcement | ID: 01 | Priority: High |
| Actor: Doctor - Patient | | |
| Description: After logging to the system, every doctor and patient post about their case. | | |
| Trigger: The doctor and the patient login to the system to start using the features of the system.  Type: 🗹 External ☐ Temporal | | |
| Preconditions:  1. The user logs to the system.  2. User information must be verified. | | |
| Steps:  1. The user whether doctor or patient browse the main page.  2. The user starts to write a post about his/her case.  3. The user uploads needed attachments to his post.  4. The user confirms his post to display it to other users. | | |
| Post conditions:  - The doctor now will be able to see patients’ announcements and vice versa. | | |
| Exceptions:  - The user might have entered wrong information about his/her case.  - User’s personal information might be incorrect. | | |
| Assumptions:  - User has successfully created an account and managed to enter correct personal information and correct information about his/her case. | | |

* **Use case diagram**
* **Sequence diagram**
* **Structural UML diagrams**
* **Class diagram**