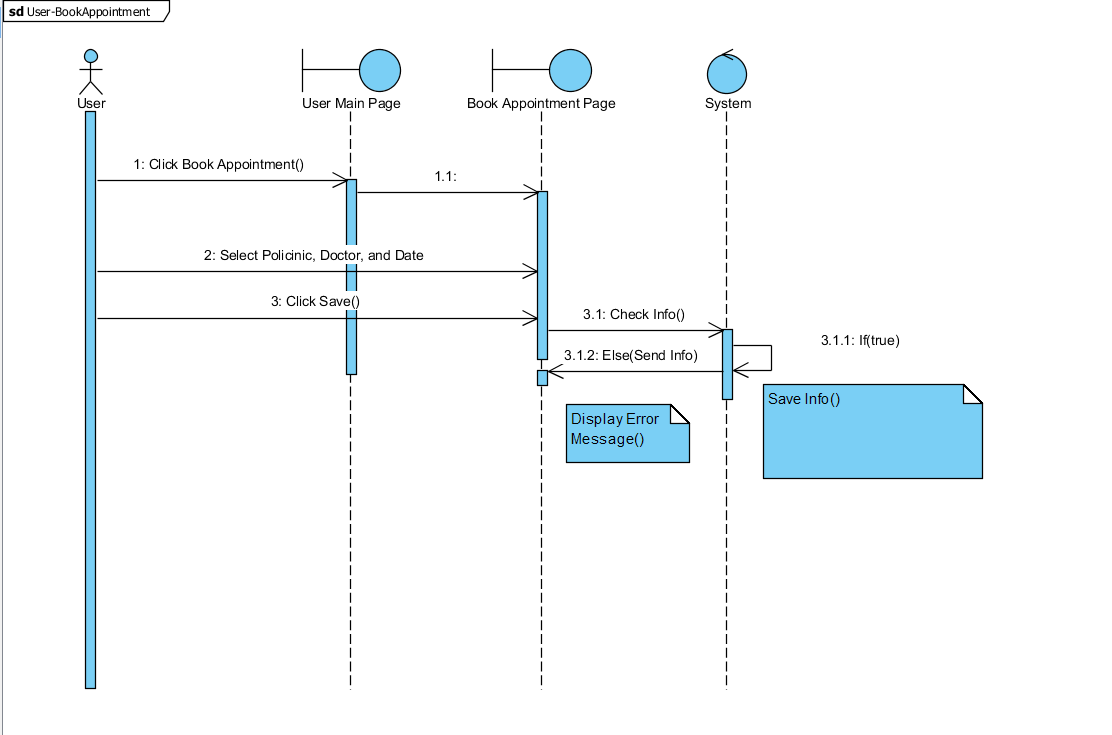


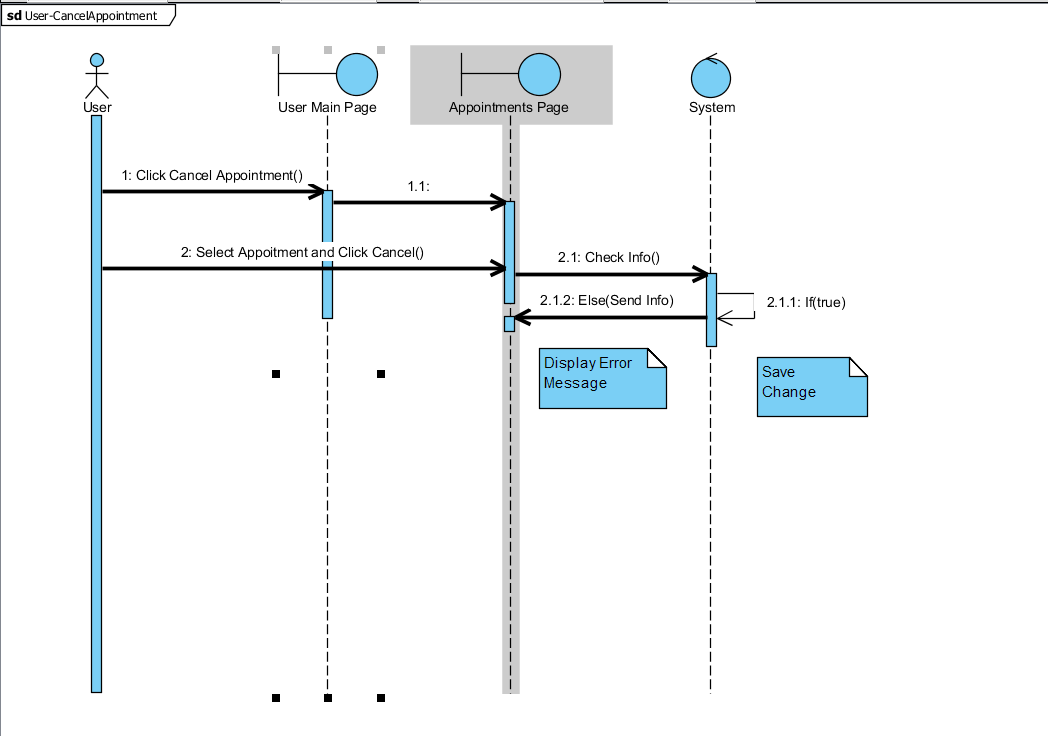
### **Dynamic model**

***Sequence Diagram***

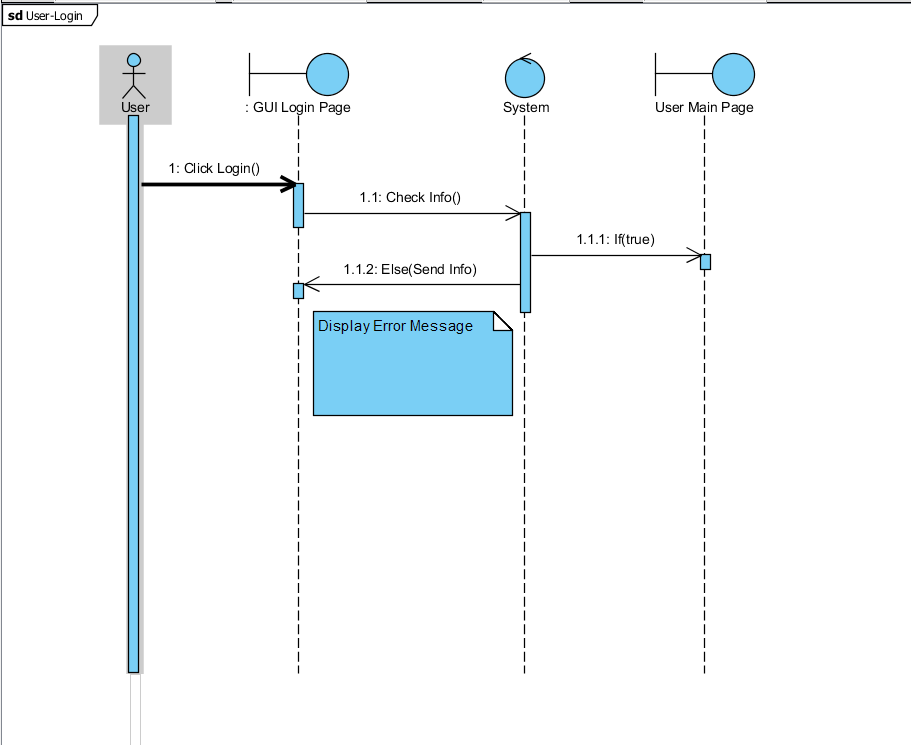
***User-Book Appointment***



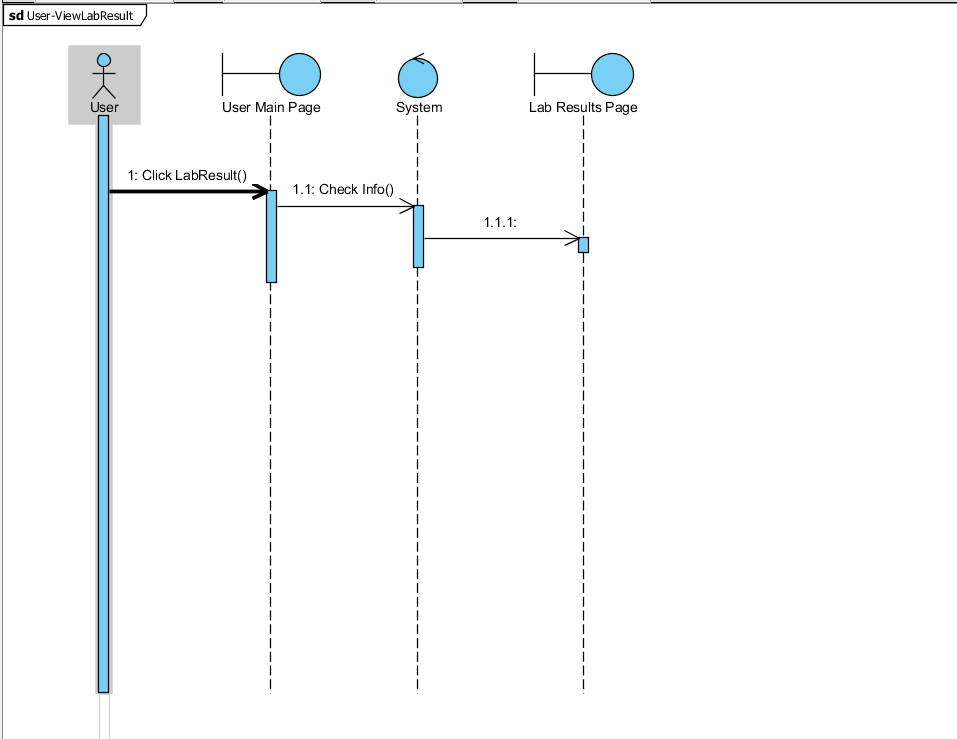
User-CancelAppointment



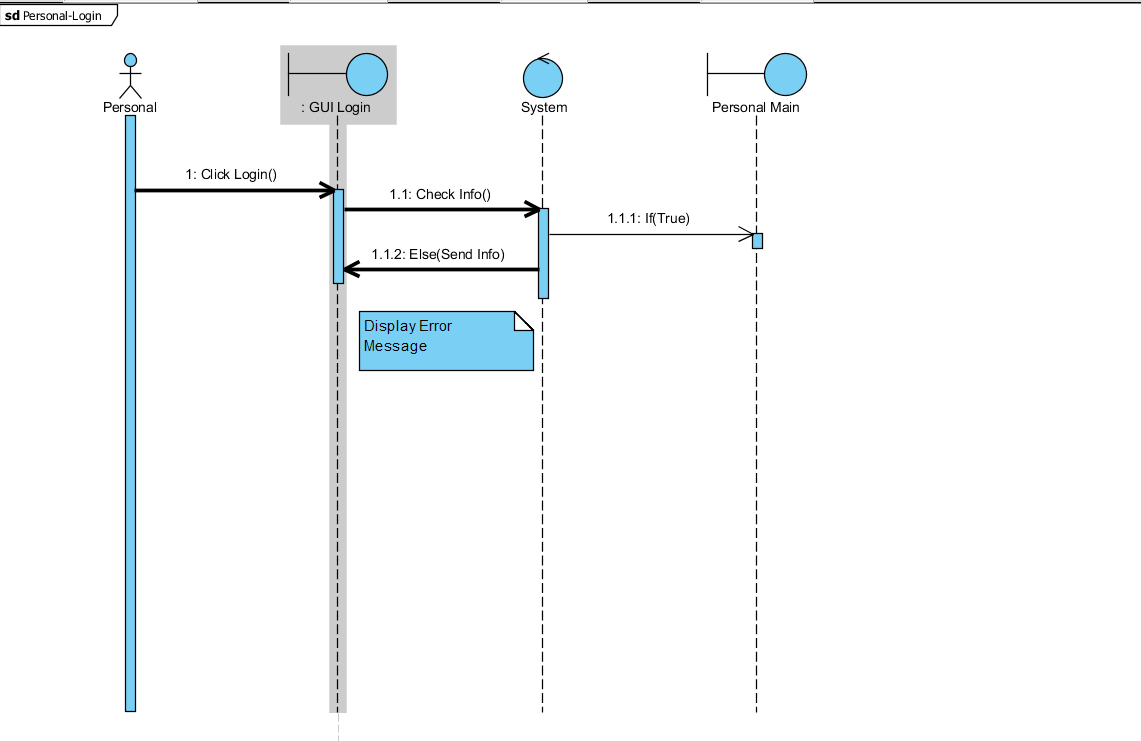
User-Login



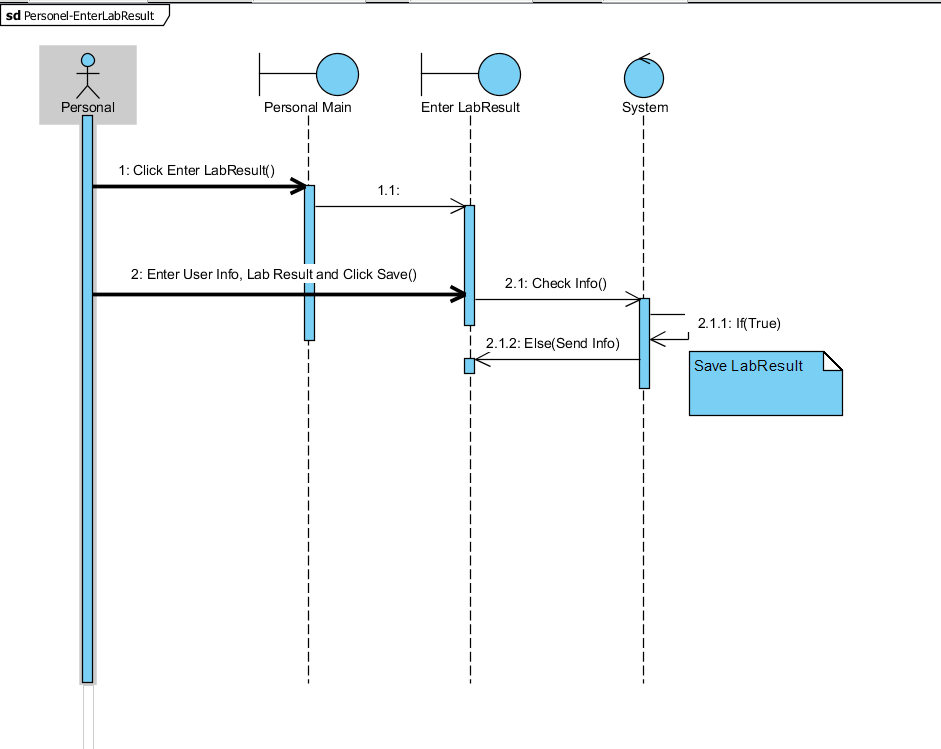
User-ViewLabResult



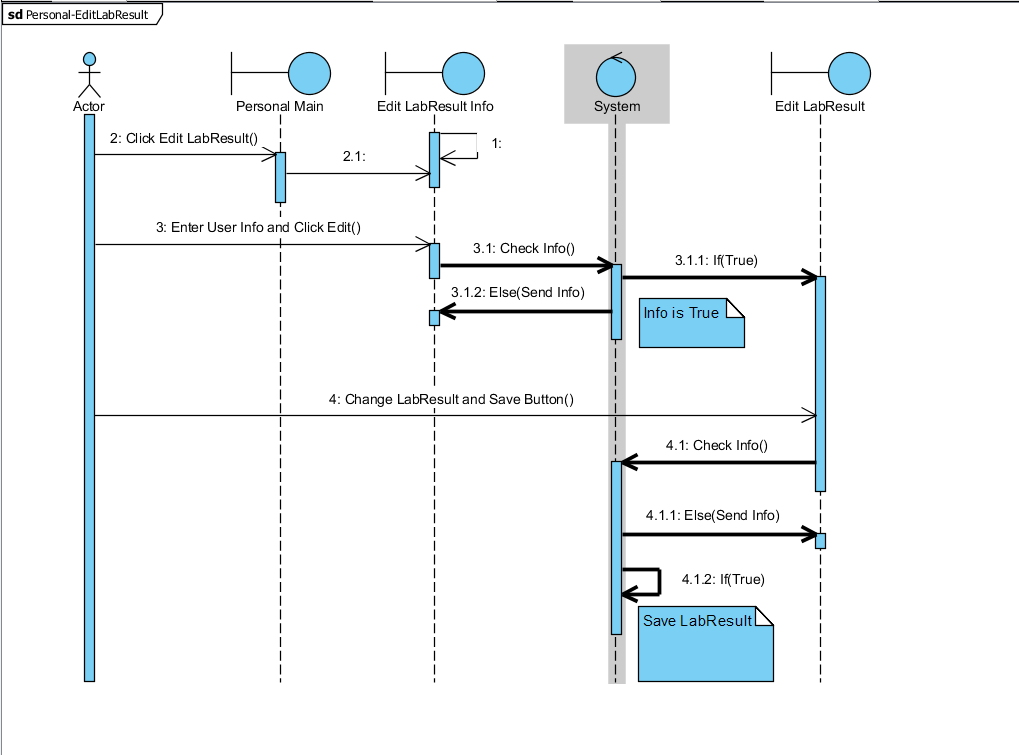
Personel Login



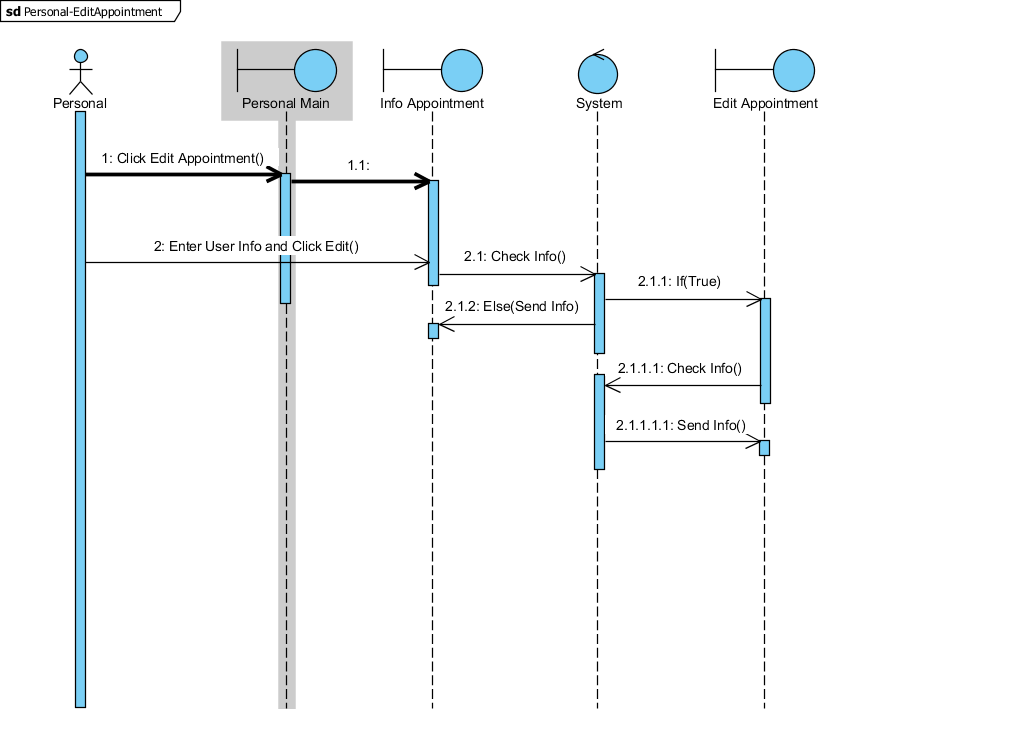
Personel-EnterLabResult



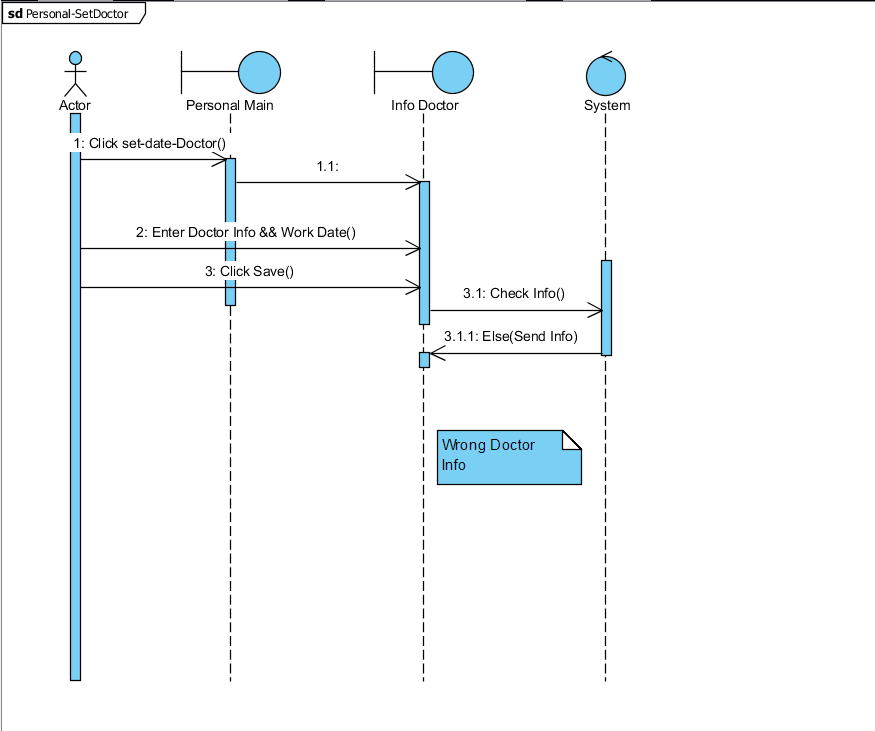
Personel-EditLabResult



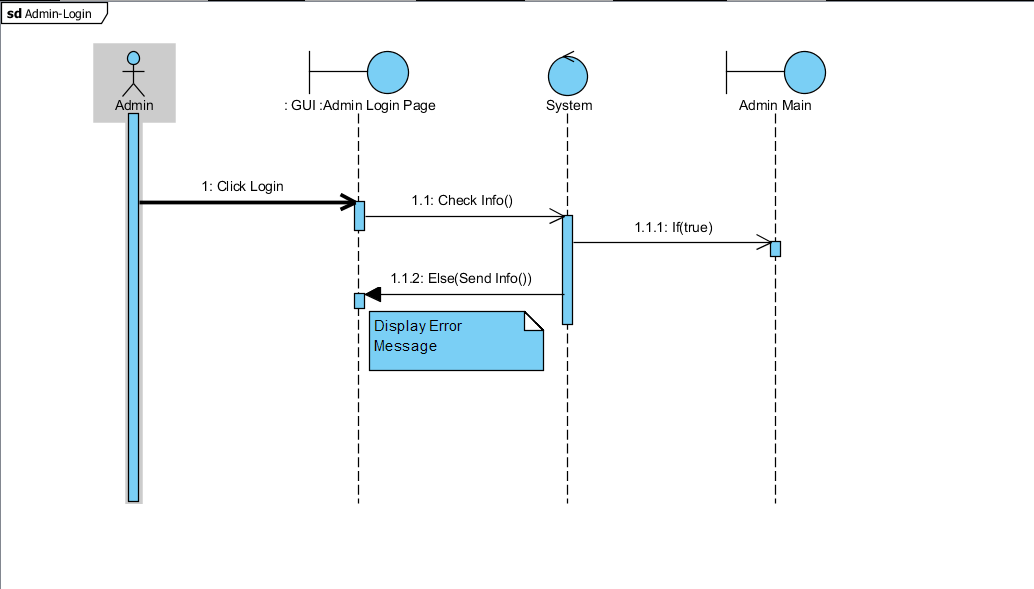
Personel-EditAppointment



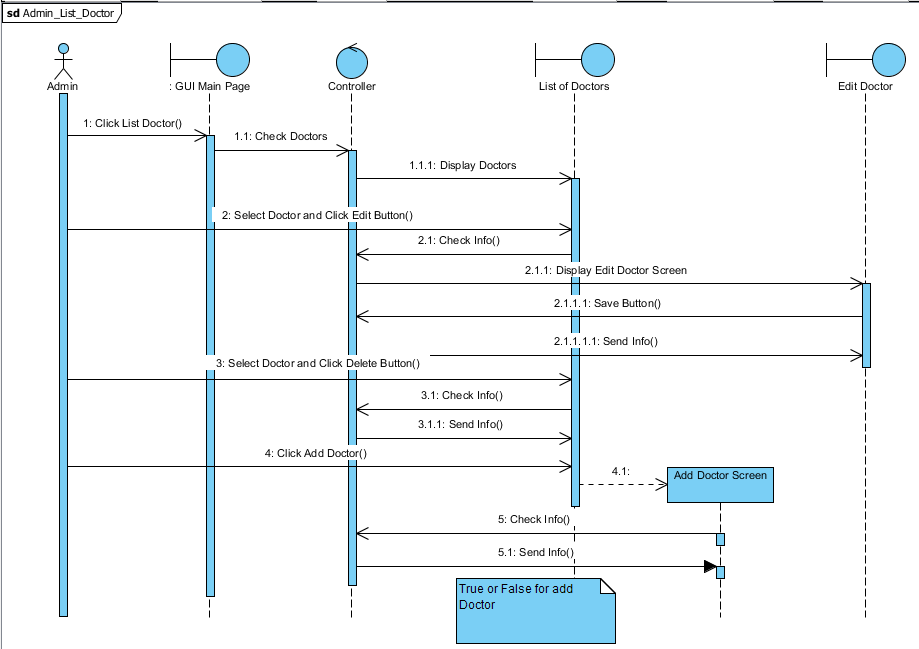
Personel-SetDoctor



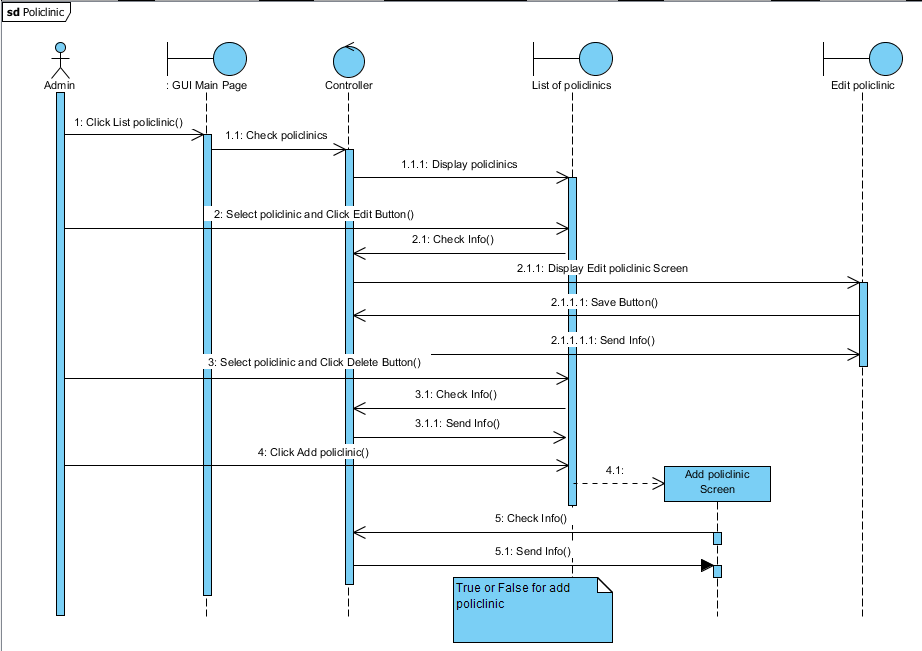
Admin-Login



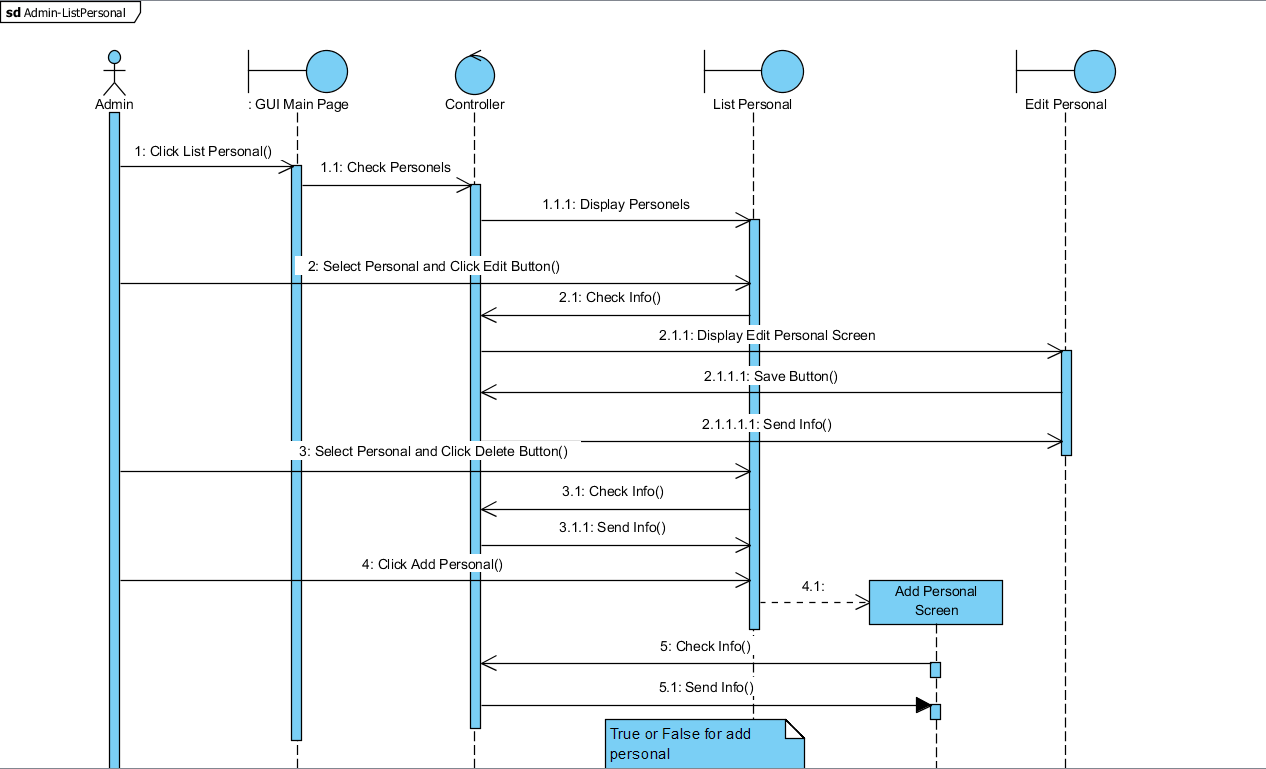
Admin­-List\_doctor



Admin policlinic



Admin-ListPersonel



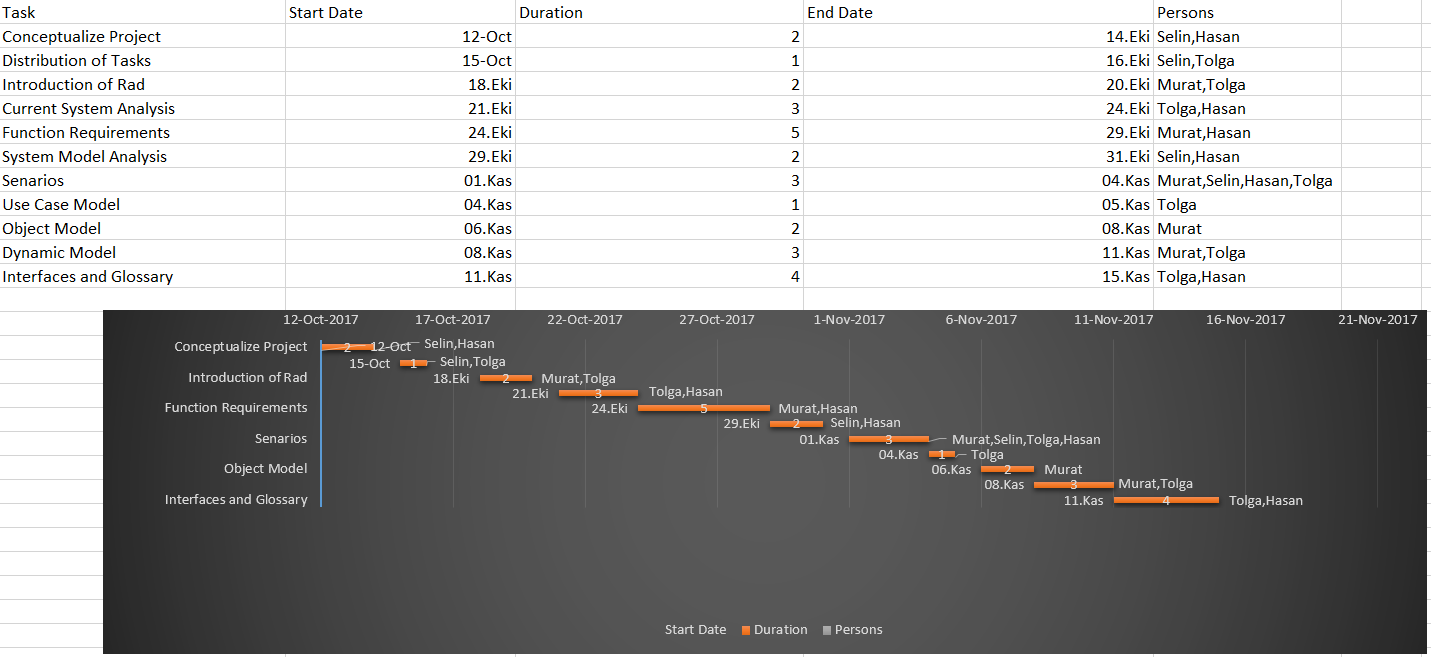
***Interfaces***

*Personnel UI*

*Doctor&Patient UI*

*Visitor UI*

*3.5 Project Schedule*

**

# Glossary

**Appointment**: Operation or order which is neccesary to doctor can examine the patient.

**Actor:** External entity that needs to exchange information with the system. An actor can represent either a user role or another system. In other words; a person, software system, or hardware device that interacts with a system to achieve a useful goal.

**Analysis:** An activity during which developers ensure that the system requirements are correct, complete, consistent, unambiguous, and realistic.

**Authentication:** The process of associating a person with access rights.

**Class diagram:** UML notation representing the structure of the system in terms of objects, classes, attributes, operations, and associations. Class diagrams are used to represent object models during development.

**Functional Requirement:** An area of functionality the system must support. The functional requirements describe the interactions between actors and the system independent of the realization of the system.

**Login:** Procedure used to get access to an operating system, or application, usually in a remote computer.

**Nonfunctional Requirement:** A user visible constraint on the system. Nonfunctional requirements describe user visible aspects of the system that are not directly related with the functionality of the system.

**Class:** A description of a set of objects having common properties and behaviors, which typically correspond to real-world items (persons, places, or things) in the business or problem domain.

**Exception:** A condition that prevents a use case from successfully concluding. The use case’s post conditions are not reached and the actor’s goal is not satisfied.

**Scenario:** A description of a specific interaction between a user and a system to accomplish some goal. An instance of usage of the system. Often presented in the form of a story.

**Sequence Diagram:** An analysis model that shows the order in which messages pass in a system or the chronological sequence of steps that take place in an activity and the entities or classes involved in the activity.

**Unified Modeling Language:** UML is a modeling language used to define a system prior to construction, much like a blueprint is used prior to building a house. It allows the project team to specify, visualize, and document an application, including its structure and design, in a way that meets all of the user requirements.

**Use Case:** A description of an interaction between an actor and a system that results in an outcome that provides value to the actor.

**Use Case Diagram:** An analysis model that identifies the actors who can interact with a system to accomplish valuable goals and the various use cases that each actor will perform.

**Life Hospital System**: Patients use the system to get an appointment and they go to doctor from the system.

**Hospital Personel:** Hospital Personel take a monthly program to doctor and them record these informations on the system.Also they can upload the information if it is neccesary.

**Admin:** Admin can reach all lists which are doctors, hospital personels and policlinics.And add, edit and delete their information.

**Patient:** Person who gets an appointment to go to doctor to be treated.

# References

1. UML Use Case Diagrams: Tips and FAQ

* <https://www.andrew.cmu.edu/course/90-754/umlucdfaq.html>

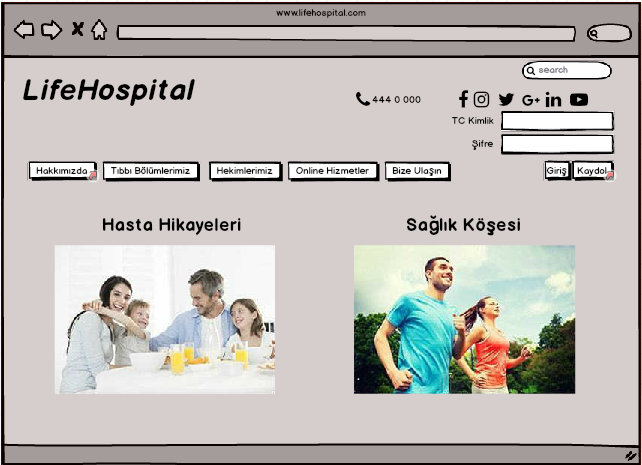
1. Introduction to Software Engineering/UML

* <https://en.wikibooks.org/wiki/Introduction_to_Software_Engineering/UML>

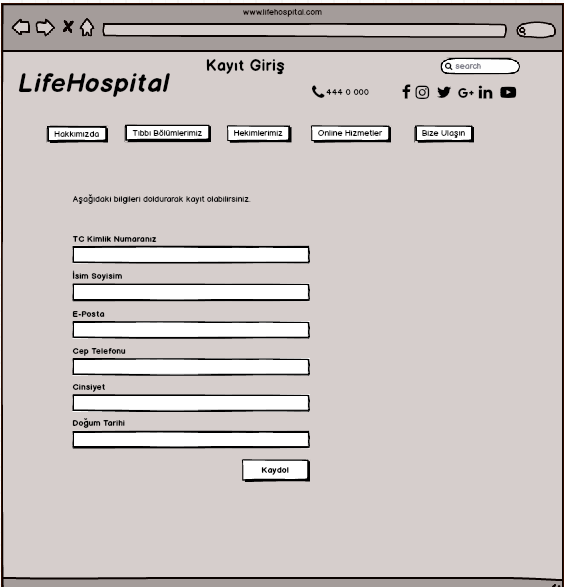
1. Bruegel B. & Dutoit A.H.. (2010). Object-Oriented Software Engineering Using UML, Patterns, and Java, Prentice Hall, 3rd ed.

# Mockups

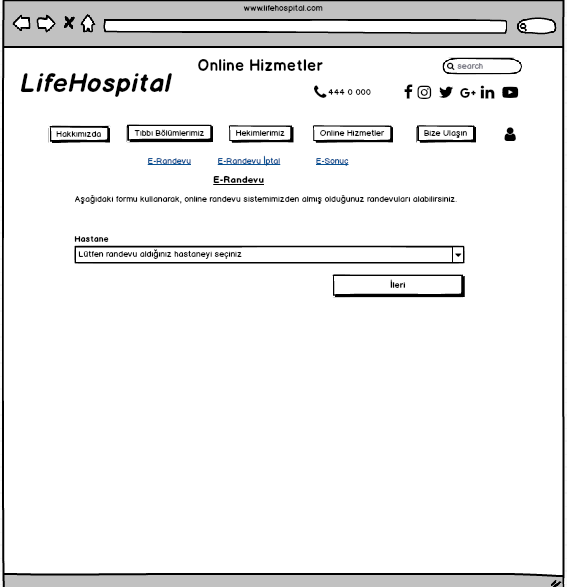
**Mockup 1**



**Mockup 2**



**Mockup 3**



**Mockup 4**

