**Software Engineering**

**System modeling**

**&**

**Architectural design**

****

**TEAM 4**

|  |  |  |  |
| --- | --- | --- | --- |
| **제출일** | **2018. 04. 29.** | **전공** | **소프트웨어학과** |
| **과목** | **Software Engineering** | **이름** | **문지환 201333186**  **윤지은 201533658**  **장해웅 201635846** |

**[ INDEX ]**

**1. System modeling**

1.1 Context models

1.2 Interaction models

1.3 Structure models

1.4 Behavior models

**2. Architectural design**

2.1 Architecture design

2.2 Architectural patterns – **MVC Pattern**

(END)

**1. System modeling**

**1.1 Context models**

**- Project Management System :**

Multiple projects can be managed independently.

**- Viewing Calendar System :**

You can use the calendar to manage your schedule.

**- Alarm Setting System :** You can set an alarm to go through the project periodically.

**- Camera Composition Guide System :**

When you take a picture, you can get a guide to the composition.

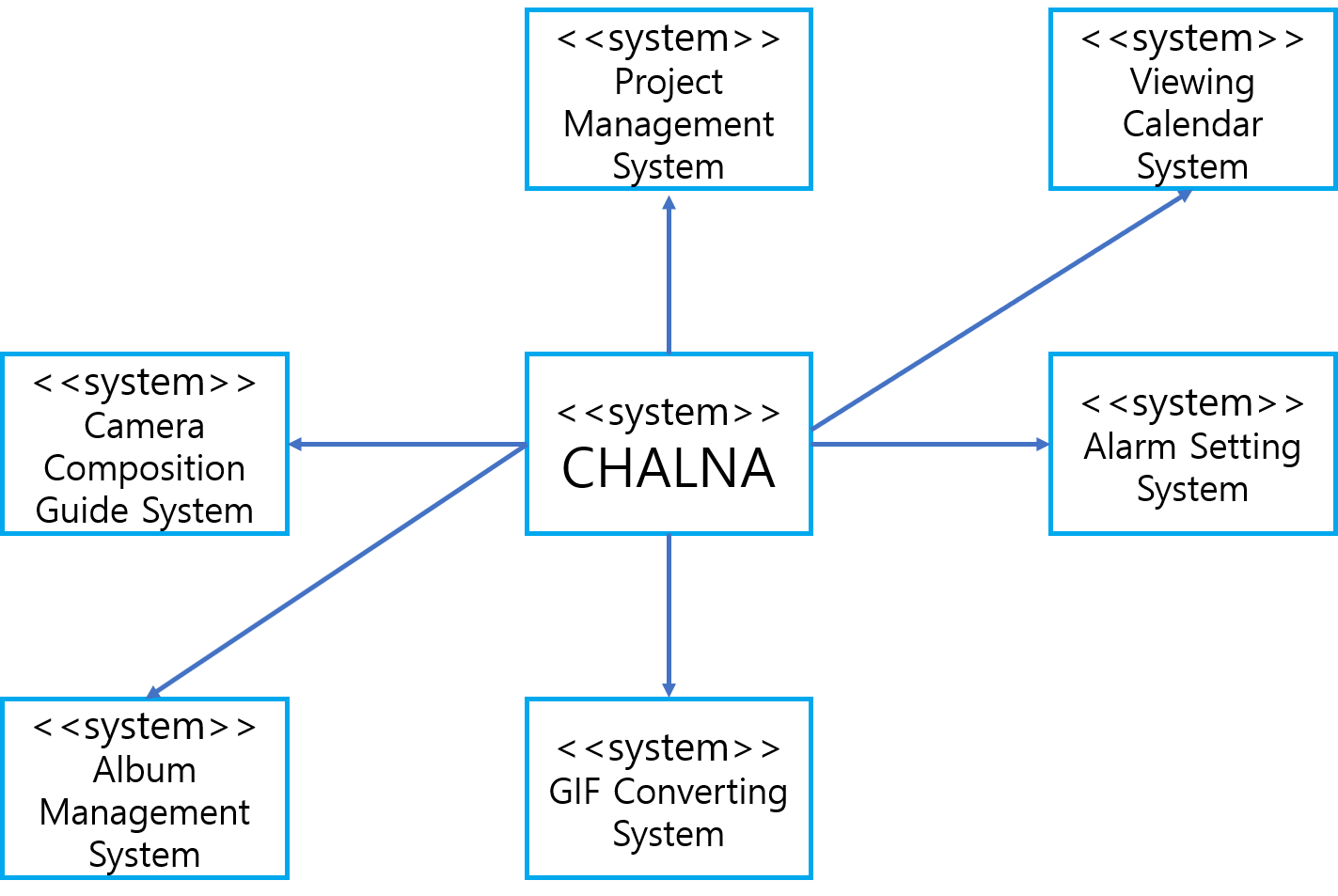
**- Album Management System :**

You can manage the album of the project in progress.

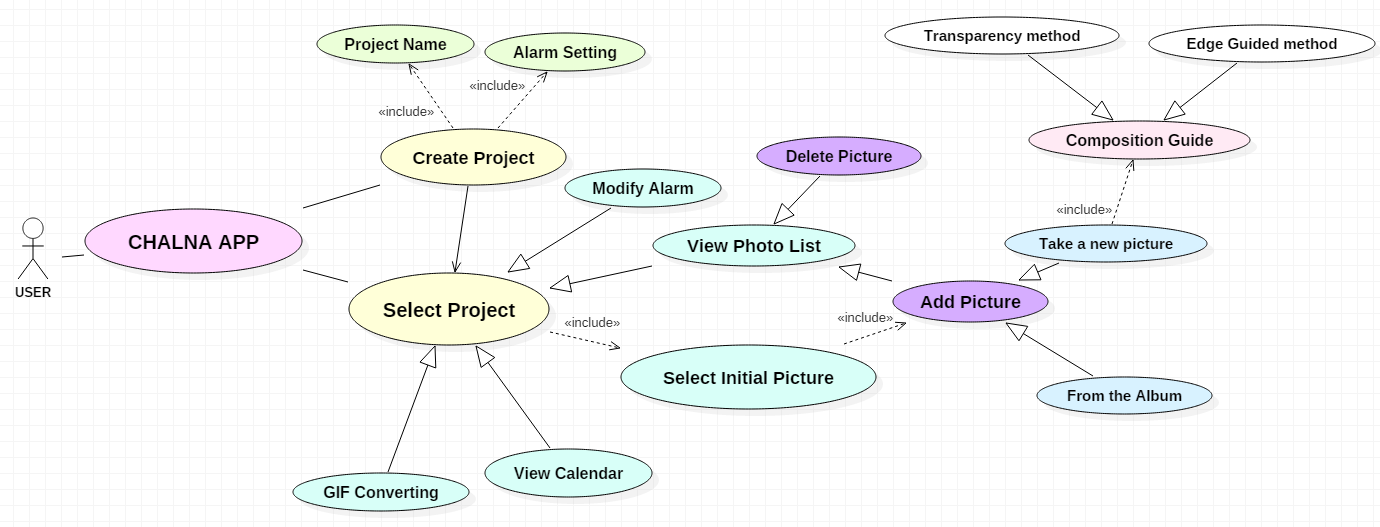
**- GIF Converting System :**

Whether complete or not, The process up to the desired moment can be converted to GIF.

**- The context of ‘CHALNA App’**

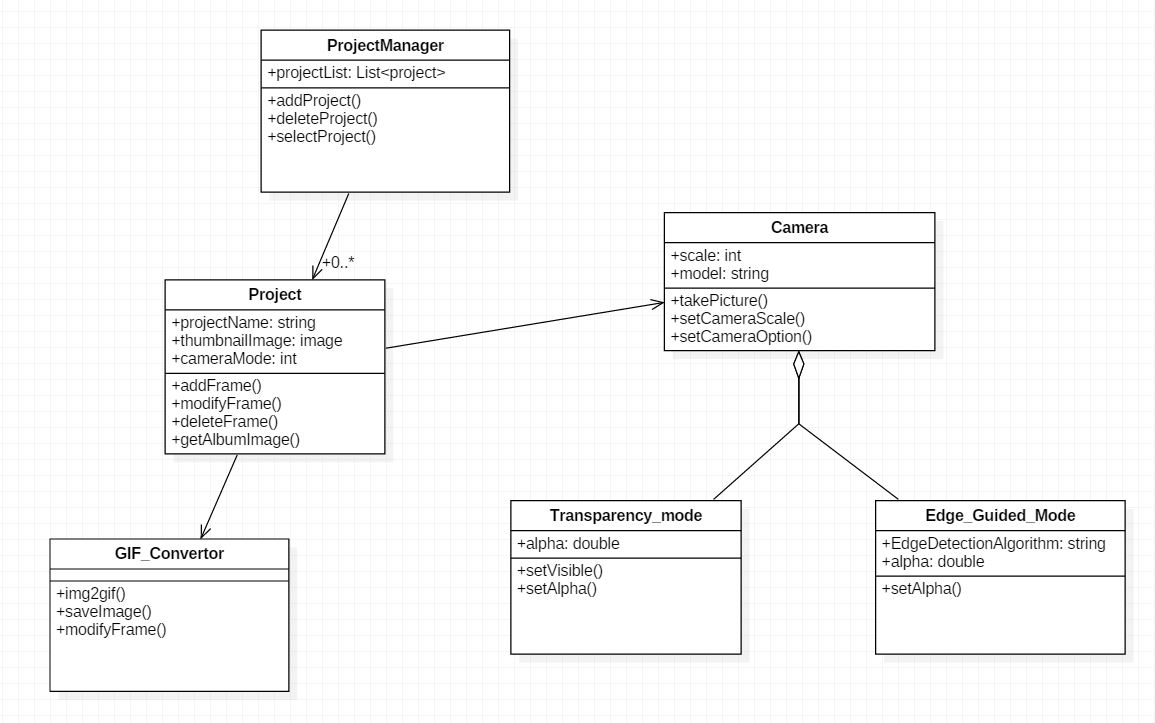


**1.2 Interaction models**

**< Use Case Diagram >**

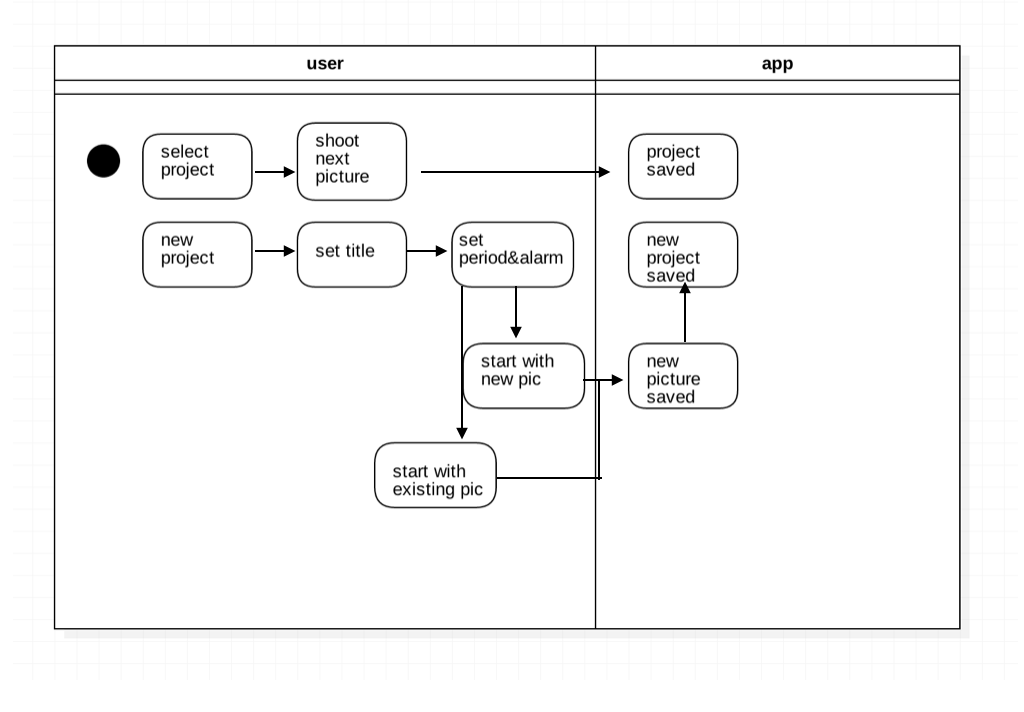
To understand our application it is important to see the interaction between the user and the system. We describe the interaction between each user and the system using the Use Case Diagram. The main system is divided into APPLICATION and related systems for generation, selection and modification of the corresponding project.

**1.3 Structure models**

**< Class Diagram >**

The Structure Model can be roughly divided into three class components.   
The camera class is responsible for taking pictures. And the project works independently.   
Each manager is responsible for several independent projects and interacts with the GIF Converter separately. The camera model has two modes, the transparency mode and the edge guided filter mode.

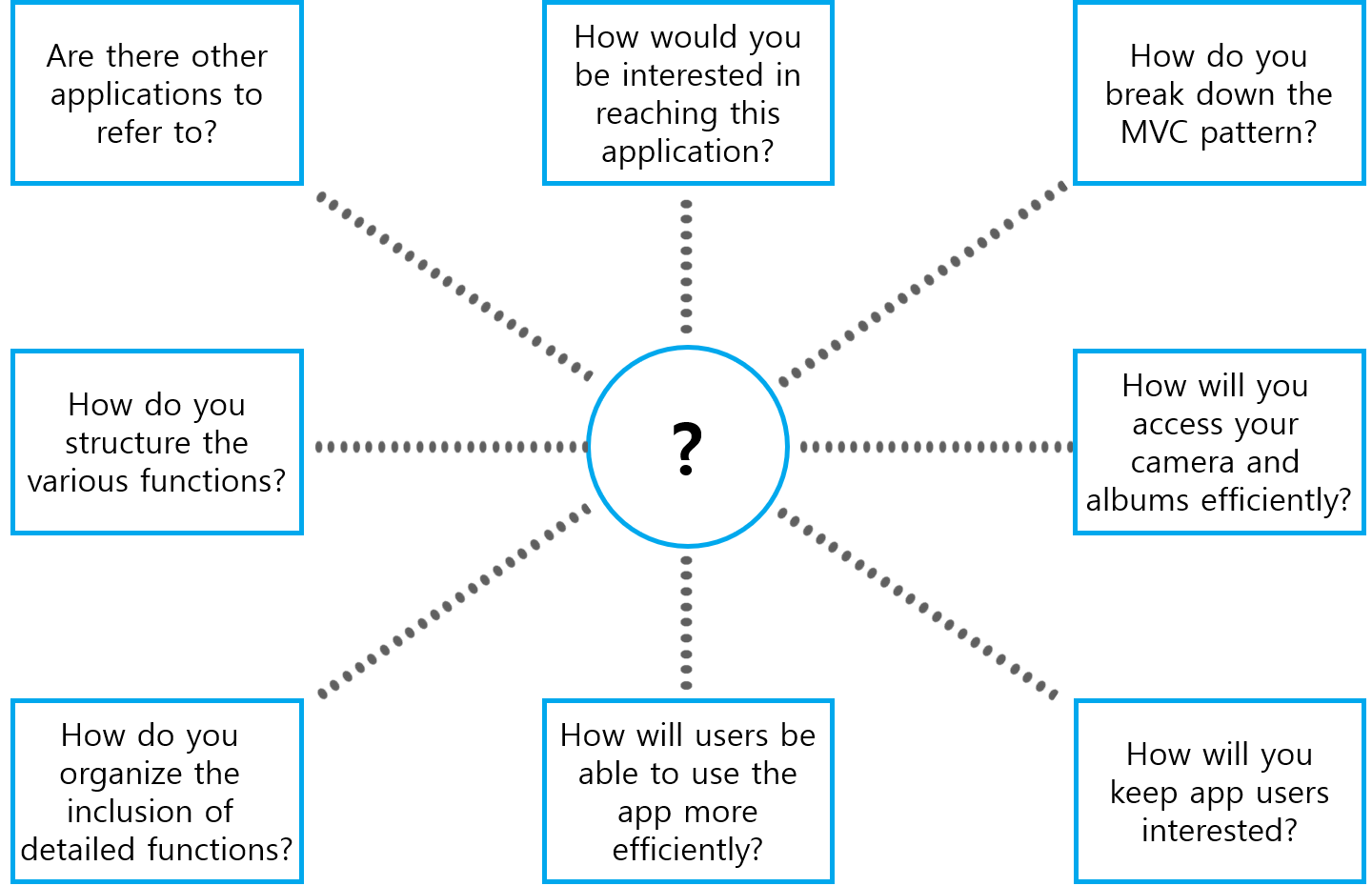
**1.4 Behavior models**

**< Activity Diagram >**

The behavior model is simple. And the interaction between the user and the app. Here, we separated the project selection, new project creation, and revision. The separation of projects is a hallmark of our application.

**2. Architectural design**

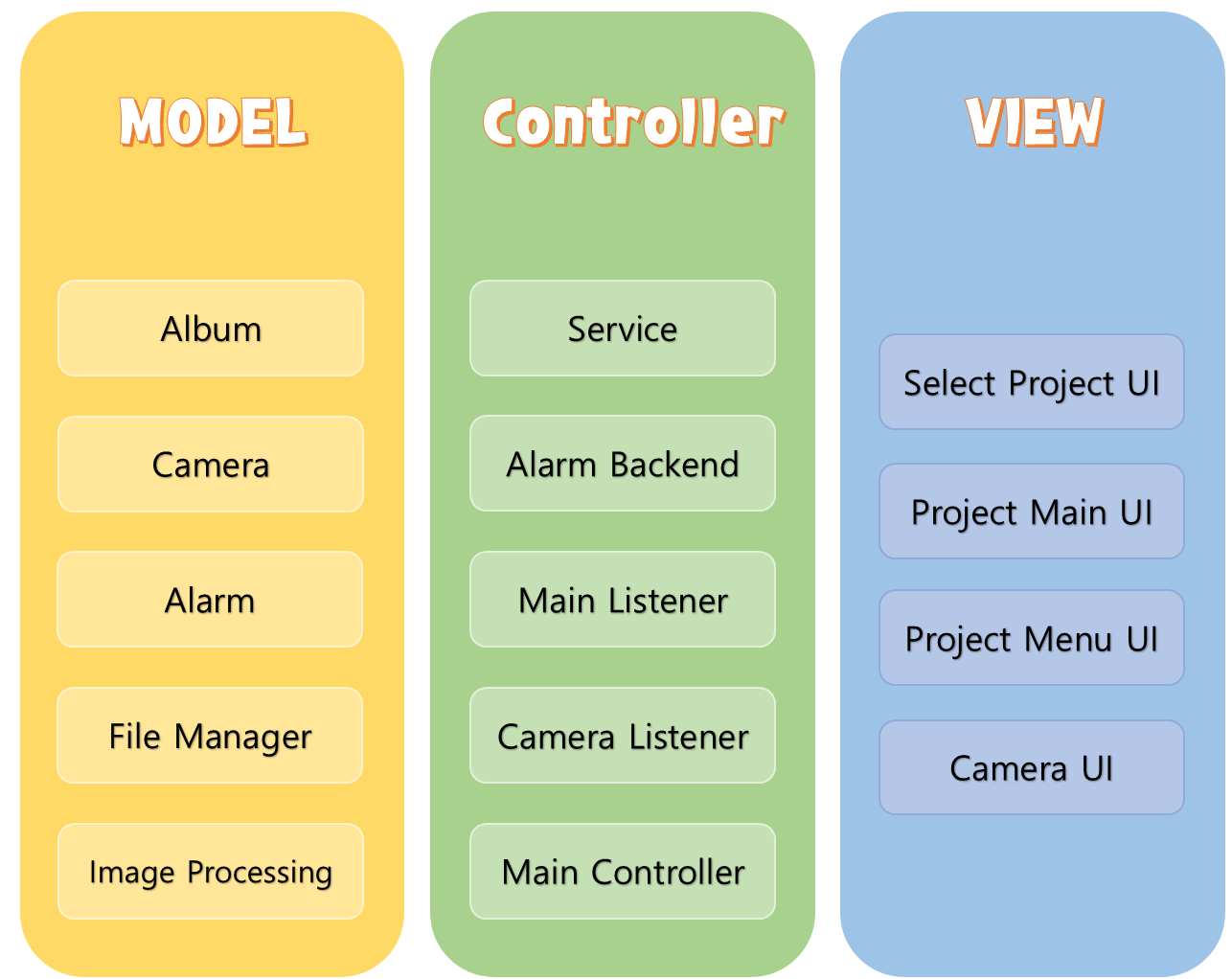
**2.1 Architecture design**



Through architectural design decisions, you can consider factors that can affect non-functional features. If we have a generic app, we can refer to it or compare it. Then, we can think more about various decisions. For example, the component should be composed of which subcomponent, What is the architecture pattern, what patterns are appropriate, What kind of approach should be used to determine the system structure? And What 'structure' to select to achieve a non-functional requirement**.**

**2.2 Architectural patterns**

**< MVC pattern >**



We divided it into a Model View Controller structure. MVP and MVVM patterns are under consideration depending on the production stage. The model is responsible for all the computation processes. Internal system calls, such as calls to cameras, albums, and file managers, work independently of the Model. We can interleave several models by modifying only the controller (all views are independent of controller and model). The controller is responsible for connecting the model to the view. The view does not know the existence of the model, and the controller takes the view's interaction as a callback, processes it in the model, and updates the UI. In this process, an update of the model also occurs.