Deliverable #2 Template
NatureOptix

SE 3A04: Software Design II – Large System Design

Kevin Hardy-Cooper 1312836
Nareshkumar Maheshkumar 1320375
Athidya Raveenthranehru 1316204
Radhika Rani Sharma 1150430
Mario Calce 1304792
Abishek Mukherjee 1151803

1 Introduction

1.1 Purpose

The following document will outline the user interactions, architecture, and classes of the application, NatureOptix. The intended audience for this document will be the teaching assistants as well as Dr. Ridha Kedhri. It is the goal of this document to provide insight to the intended audience into the internal mechanisms of NatureOptix. After reading this document, the audience should have a good understanding of the way in which a user can interact with the system, the architecture that the system uses, and the classes that comprise the system.

1.2 System Description

The NatureOptix application will allow the user to identify a natural phenomena. The application will ask the user a series of questions and the user will be able to specify an answer for each question from a drop down menu. The application will use a combination of Repository and Blackboard Architecture. It will be comprised of 9 entities; 3 experts (a physicist, a field expert, and a meteorologist), a location expert, a database corresponding to attributes identified by each one of the experts for every optical natural phenomena, and a controller. Each database will contain information about each phenomena specific to what it's expert knows. Each expert will act as an algorithm that compares characteristics of a phenomena in the database to the user input. The physicist will compare the visual attributes, the field expert will compare the physical attributes, and the meteorologist will compare the weather condition attributes required for each phenomena to occur. Each expert will create a list of phenomena that has characteristics that match the user input. The three lists will then be sent to the controller. The controller will determine what phenomenon is common to all three lists and will present this phenomenon to the user. In some boundary cases, the applications shall provide the users with an aggregated list of closest match results to their query. The experts will not be able to communicate to each other and will only be communicating with the database and controller.

1.3 Overview

The user interactions will be shown using a use case diagram, along with descriptions of each interaction. The architecture for the system will be presented using descriptions along with justifications for decisions. Furthermore, the classes of the application will be outlined using an analysis class diagram as well as a class responsibility collaboration.

2 Use Case Diagram

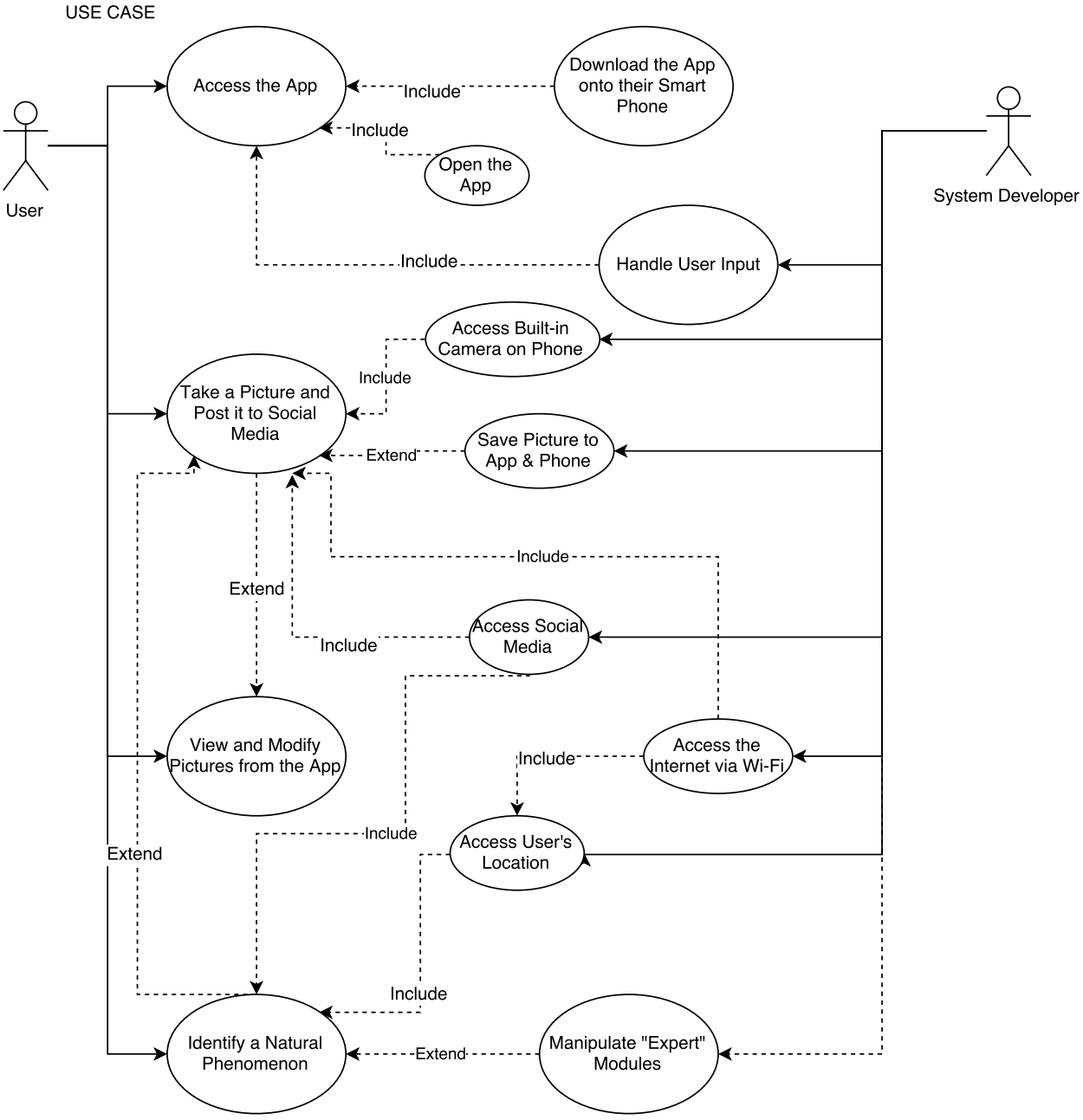

Figure 1: Use Case Diagram

1. User wishes to access the app. The user shall be able to download the app onto their Android smart phone in order to access it. This downloading action will be handled by the Android operating system. User shall be able to open the app. The system developer needs to ensure that the application shall be able to handle user input, so that the user can access and interact with the app.

2. User wishes to take a picture and post it to social media (Instagram). User can then access the built-in camera on phone and it's functionality from the app. User can also save the picture to the app and the phone if they so desire. User shall be able to post to social media directly from the app. The system

developer shall ensure that the application can access the internet via wireless connection from their Android smart phone, the built-in camera on the smart phone, and social media. The application shall be able to save pictures directly to the phone.

3. User wishes to view and modify pictures from the app. User shall be able to view saved pictures through the app and the phone. User shall be able to delete pictures from the app and the phone. System Developer shall ensure that the application can display requested pictures to the user. The application shall be able to delete pictures directly from the phone.

4. User wishes to identify a natural phenomena. User shall be able to post a picture on social media through the app. The system developer shall ensure that the app can access the user's location using Google maps services. App shall have access to the internet via wireless connection from smart phone. App shall be able to switch "expert" modules in order to identify the natural phenomenon.

3 Analysis Class Diagram

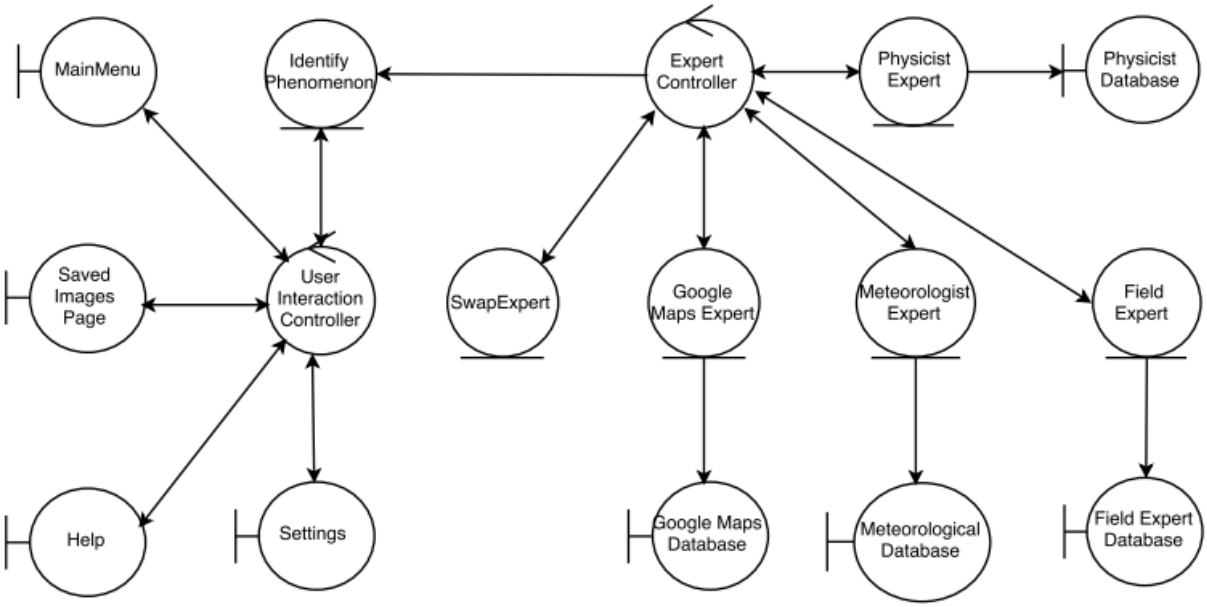

Figure 2: Analysis Class Diagram

4 Architectural Design

This section provides an overview of the overall architectural design of the application NatureOptix. The overall architecture followed by the division of the system into various subsystems is discussed.

4.1 System Architecture

1. We are using the Blackboard Architecture with some elements of the Repository Architecture (that have been discussed in the course) for our system.

2. The Data Centered quality of the architecture helps distinguish and separate the modules, which in our case are the experts, from one another as we do not intend for them to be communicating with one another and be implemented as standalone entities. This approach is useful in creating modularity within the program. We are looking at this from 2 viewpoints:

2.1 Our application is centered around displaying information to the user after they provide us with an initial input; the client, in this case the expert(s) (which is represented by the corresponding algorithm) will be focused on fetching the data from the data center in the form of a batch transaction request. Hence, the element of a Repository architecture seems useful.

2.2 However, the fetching and display of data will be instructed and controlled by the data center, which is active and is the one responsible for delegating tasks to the different experts (algorithm) and hence the main architecture is Blackboard. This approach also provides for a highly cohesive yet lowly coupled structures, as at any point in time we can get rid of any one of the experts without making severe changes to the codebase, while the cohesiveness comes from the data center running as a single, unified and non dependant on the agents kind of an entity.

4.2 Subsystems

NatureOptix will be broken down into ten subsystems. These subsystems are:

1. UserInteractionController
2. ExpertController
3. Experts (4),
4. Database (4).

The UserInnteractionController will ensure that the proper actions occur after the user has selected them. This means that this class will faciliate user experience. The UserInterfaceController will inciate the process of identifying the phenomenn. Once the user has selected that they wish to identify a phenomeon, this controller will send information to the IdentifyPhenomenon module. It will also recieve the relevant information from the IdentifyPhenomenon class and pass it along to the user.

The ExpertController will control the experts. It is responsible for passing data to the experts and to pass information to the IdentifyPhenomenon module. The ExpertController is also reponsible for facilating the process of swaping experts. This is done with a back and forth with the SwapExpertClass

Three of the four Experts will be implemented as algorithms. These three experts are the Physicist, the FieldExpert, and the Meterologist. The experts will answer questions that pertain to their field. The Physicist will handle all physical aspects of natural phenomena. This includes colour, shape, and size. The FieldExpert will be the expert responsible for identification based on attributes such as temperature and humidity. Finally the Meterologist will attempt to identify phenomeon based on the current weather at the location. The location expert will be the Google Maps API.

The final subsystems are the Databases. Each expert will be assigned a Database. This ensures that the experts are easily swappable. The Database will store all the researched data about the natural phenomenon. Each expert will access the assigned database when needed. Each database will have it's own schema. This makes having multiple databases necessary.

5 Class Responsibility Collaboration (CRC) Cards

Class Name: Physicist Database	
Responsibility:	**Collaborators:**
Has information on all the attributes of each phenomenon that the physicist would be looking for. This includes shape, colour, opacity, size, angle, elevation and brightness.	Physicist Expert

Class Name: Field Expert Database	
Responsibility:	**Collaborators:**
Has information on all the attributes of each phenomenon that the field expert would be looking for. This includes temperature, density and moistness.	Field Expert

Class Name: Meterologist Database	
Responsibility:	**Collaborators:**
Has information on all the attributes of each phenomenon that the meterologist would be looking for. This includes sun, temperature and precipitation.	Meteorologist Expert

Class Name: Google Maps Database	
Responsibility:	**Collaborators:**
Has information on the locations of each phenomenon in order for the Google expert to use.	Google Maps Expert

Class Name: Physicist Expert	
Responsibility:	**Collaborators:**
For attributes shape, colour, opacity, size, angle, elevation and brightness it will compare the attribute specifications given by the user to the phenomenons in the database and create a list of phenomenons that match the user input.	Physicist Database Expert Controller

Class Name: Field Expert	
Responsibility:	**Collaborators:**
For attributes temperature, density and moistness it will compare the attribute specifications given by the user to the phenomenons in the database and create a list of phenomenons that match the user input.	Field Expert Database Expert Controller

Class Name: Meterologist Expert	
Responsibility:	**Collaborators:**
For attributes sun, temperature and precipitation it will compare the attribute specifications given by the user to the phenomenons in the database and create a list of phenomenons that match the user input.	Meterologist Database Expert Controller

Class Name: Google Maps Expert	
Responsibility:	**Collaborators:**
For location attribute it will compare the user's location to the phenomenons in the google maps database and create a list of phenomenons that could be in the user's area.	Google Maps Database Expert Controller

Class Name: Swap Expert	
Responsibility:	**Collaborators:**
Switches experts for expert controller to analyse each experts results.	Google Maps Database Expert Controller

Class Name: Expert Controller	
Responsibility:	**Collaborators:**
Gives experts information from user and gives the user information complied by experts to identify phenomenon. Also, chooses phenomenons that appear in all lists given by experts and sends to user as identified phenomenon.	Identify Phenomenon SwapExpert Google Maps Expert Meterologist Expert Field Expert Physicist Expert

Class Name: Identify Phenomenon	
Responsibility:	**Collaborators:**
Holds all attributes specified by the user for the Expert Controller. Expert Controller then returns phenomenon identified by experts to this class to be shown to user.	User Interaction Controller Expert Controller

Class Name: User Interaction Controller	
Responsibility:	**Collaborators:**
Coordinates all user interaction with system.	Identify Phenomenon MainMenu Saved Images Page Help Settings

Class Name: MainMenu	
Responsibility:	**Collaborators:**
Main graphical interface that user will identify phenomenon's attributes with.	User Interaction Controller

Class Name: Saved Images Page	
Responsibility:	**Collaborators:**
Graphical interface where user will be able to see all images saved through this application and have the option to share it to their social media.	User Interaction Controller

Class Name: Help	
Responsibility:	**Collaborators:**
Graphical interface where user will be able to see intructions on how to naviagate and use this application.	User Interaction Controller

Class Name: Settings	
Responsibility:	**Collaborators:**
Graphical interface where user will be able to change the settings of the application.	User Interaction Controller

6 Division of Labour

Kevin John Hardy-Cooper - Use Case Diagram
Nareshkumar Maheshkumar - Analysis Class Diagram
Athidya Raveenthranehru - Class Responsibility Collaboration Tables
Radhika Rani Sharma - Introduction
Mario Calce - Architecture Subsystems
Abhishek Mukherjee - Architectural Design