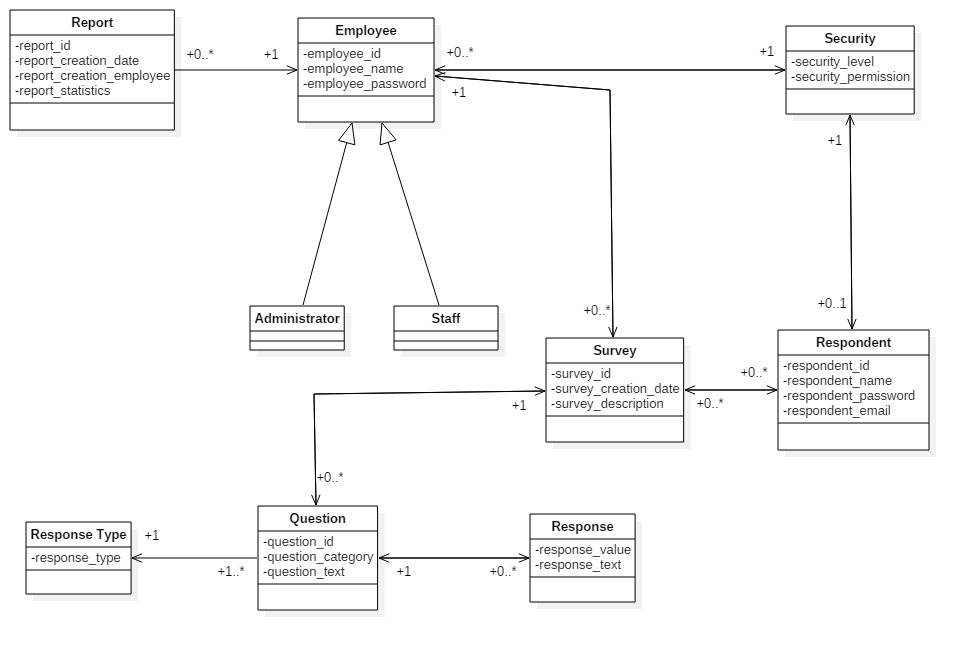
Survey Management

System

- Analysis and Design Document -

# Elaboration – Iteration 1.1

# Domain Model



**Association Definitions:**

Employee – Report Association

* Want to know who created the report and when.
* The system administrator can remove or update old reports.

Employee – Security Association

* Employees can have 0 or more security levels, depending on the type of employee.

Employee – Survey Association

* Each employee is able to create multiple surveys.
* Employees cannot contribute to the surveys that do not belong to themselves (except for administrators), thus, a survey has only one creator.

Respondent – Security Association

* Respondents can have 0 or 1 security levels, depending on their choice of taking the survey as an anonymous user or as a registered user.

Survey – Respondent Association

* Surveys can be created before respondents are identified.
* Respondents can be registered before their survey has been created.
* Respondents may respond to multiple surveys, if the respondent wants to retake a survey, the last answers of the respondent will be stored.

Survey – Question Association

* Surveys can be created without adding questions.
* Questions may be created only when a survey is opened for response.
* Questions cannot be reused by many surveys.

Question – Response Type Association

* Each question must have a single response type (e.g. vote type, text type). In the same time, the same response type can be used for multiple questions.
* Response types are predefined and used each time a question is created.

Question – Response Association

* Each question might have 0 or multiple responses.
* A response can belong only to a question, it is stored as a unique answer being independent of the type of response.

**Class Model Discussion:**

Employee Subclass Attributes:

* Administrator and Staff are not different other than their security levels, which are noted through security\_level in Security class.
* An Administrator may perform same functions as a Staff instance, and has some more features.

Respondent Attributes:

* Each respondent can have a single account, or no account at all.
* A registered respondent must have filled all the fields, meanwhile, for an anonymous user no field can be completed.

Survey Attributes:

* Each survey has a creation date and a description specifying the survey theme.

Question Attributes:

* Each question will belong to a category and will have the text body.

Response Attributes:

* A response can be given in many forms. This class’s fields will be optional, because a question can have only a response type. Also, a question has no default answer until the respondent offers an answer.

Security Attributes:

* As discussed above, each user has a security level which is used for deciding how the accounts are managed.
* Based on the security\_permission attribute, a user is able to perform more operations than other users.

Report Attributes:

* A report must have a creation date for monitoring the progress of the research.
* A report is generated by an employee, so the name of the employee has to be stored.
* The generated report is based on the statistics collected from the question answers. The statistics may be interpreted and represented in different ways.

# Architectural Design

## Conceptual Architecture

The system’s architectural style will be structured on layers so it can be decomposed into groups of subtasks, identifying and separating the single responsibility components. The motivation for this approach is because it is much easier to develop the application if each component is independent one of another, easier to bring modifications, easier to test the correctitude of the implemented operations.

The database layer will handle the storage of the information. The application will present multiple entities, namely report, survey, question, employee, respondent, response and survey\_respondent. Each of these entities will be identified by a unique number and have some features.

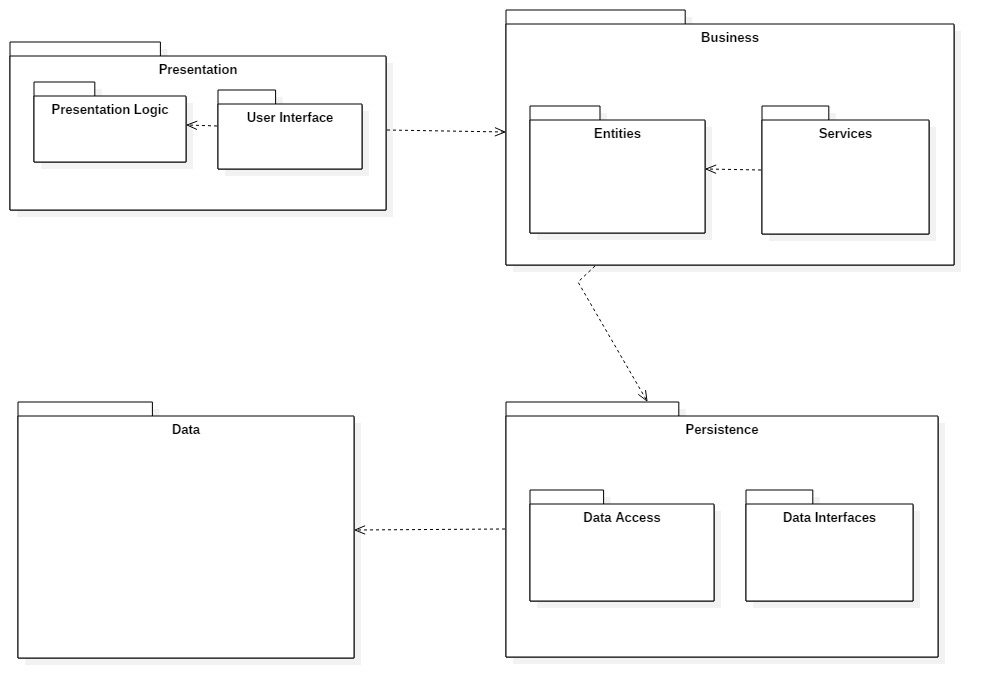
The persistence layer will be a bridge between database layer and business logic layer with the purpose of implementing the CRUD operations and retrieving other significant data which will be later used by the business logic layer.

The business logic will access the CRUD operations for each entity, will handle the generation of reports, filter surveys, manage the state of a survey, handle the operations managing the user accounts.

The presentation layer will have the information from business logic passed to it, or it will pass data to be processed to business logic. In the end the presentation layer will display the processed data to the users.

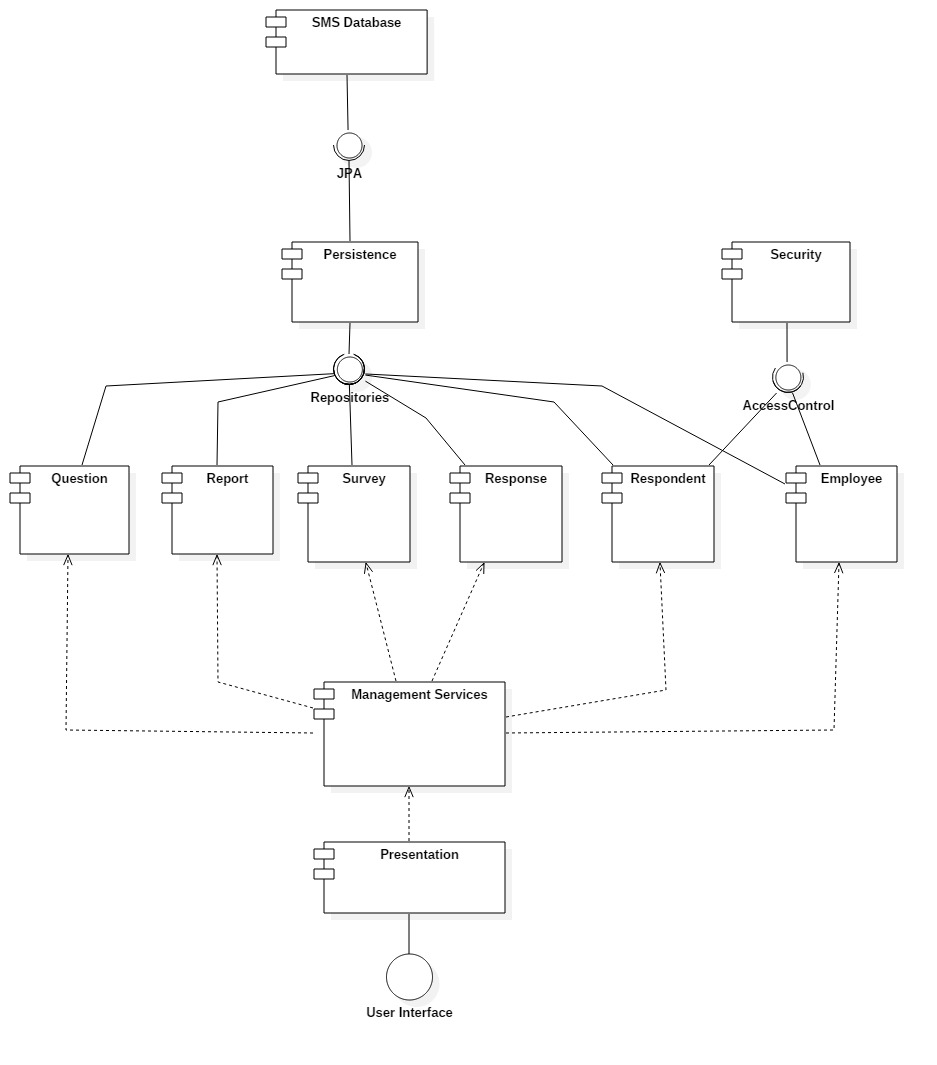
The intended architectural style to be used is REST because the application will present multiple states so it has to aim for performance. The motivations to use this style are because of the modifiability of components to meet changing needs and the simplicity of a uniform interface.

## Package Design

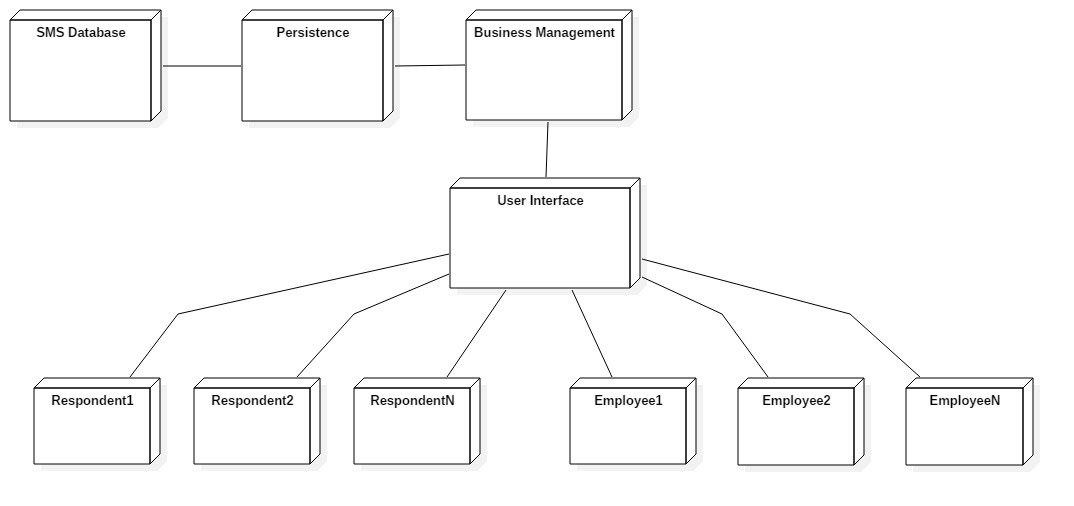


## Component and Deployment Diagrams

**Component Diagram:**



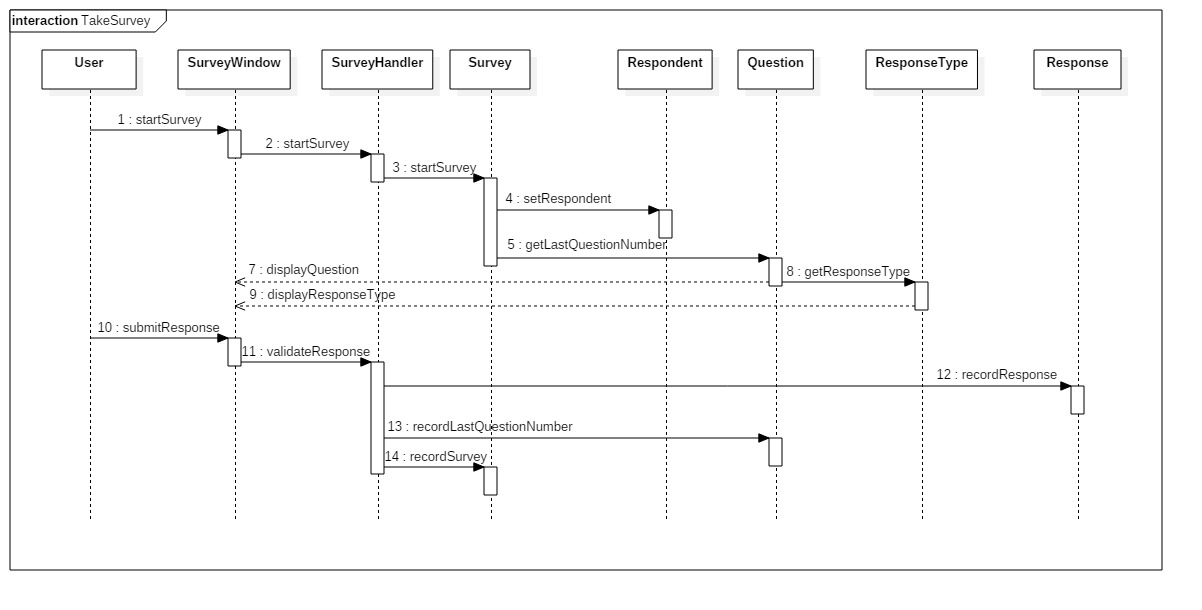
**Deployment Diagram:**



# Elaboration – Iteration 1.2

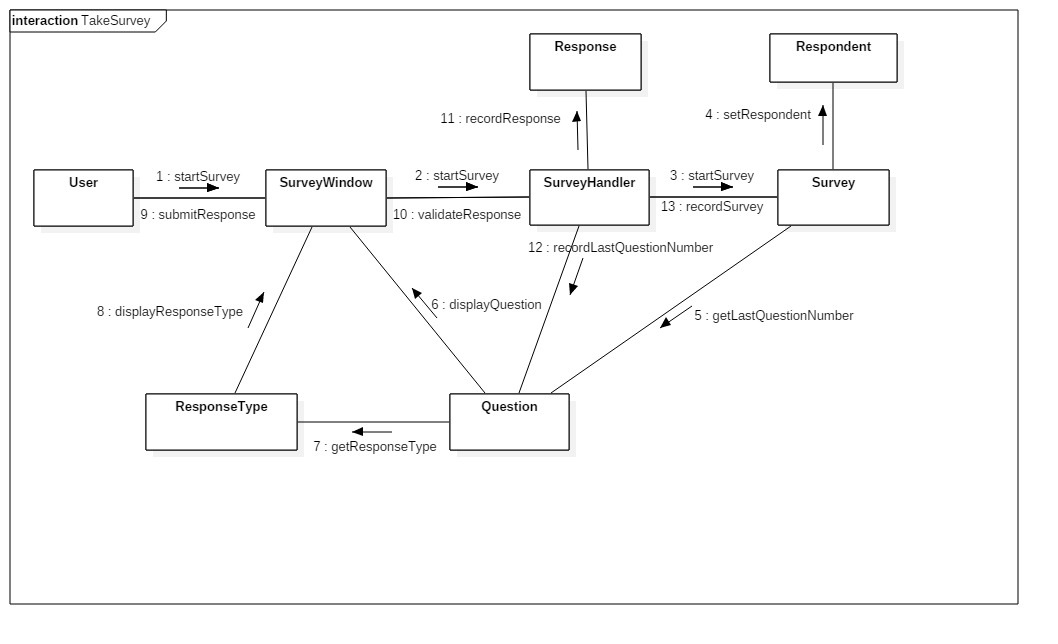
# Design Model

## Dynamic Behavior

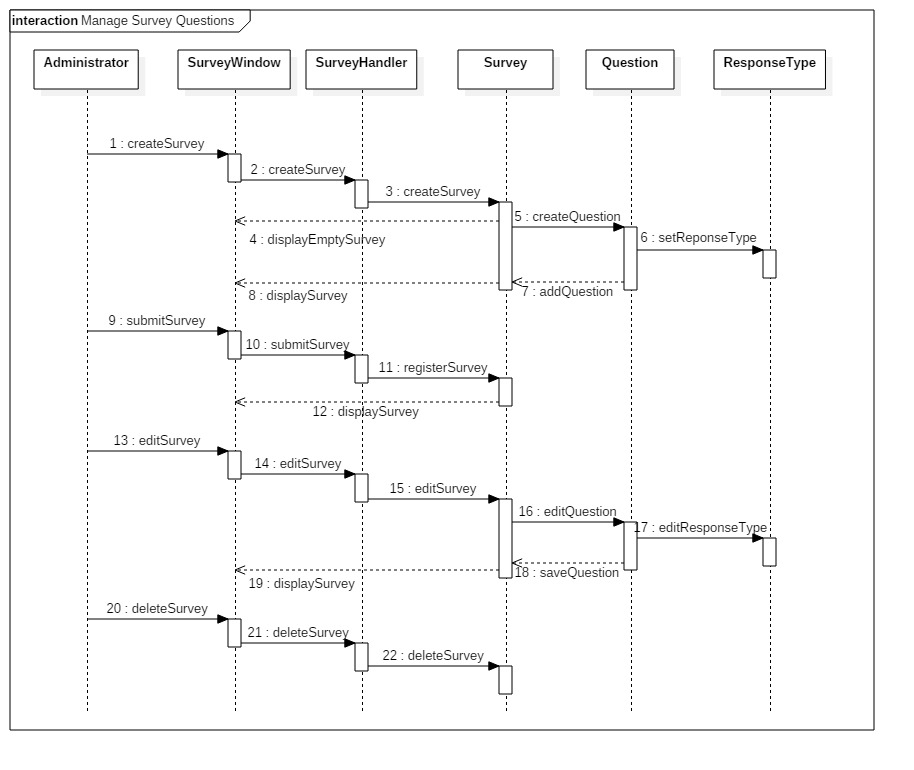


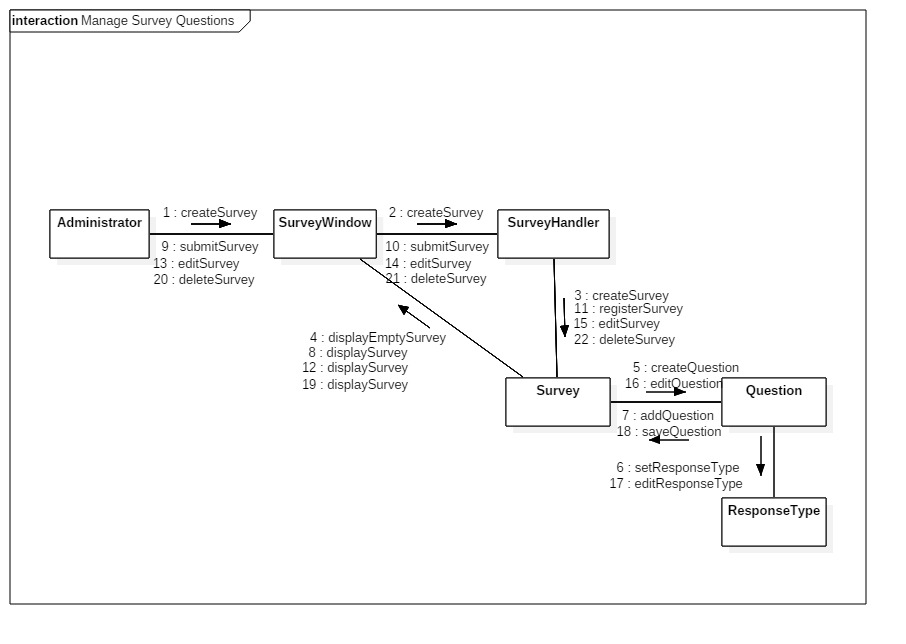
**Sequence Diagram: Take Survey Scenario**

**Communication Diagram: Take Survey Scenario**



**Sequence Diagram: Manage Survey Questions Scenario**



**Communication Diagram: Manage Survey Questions Scenario**

## 

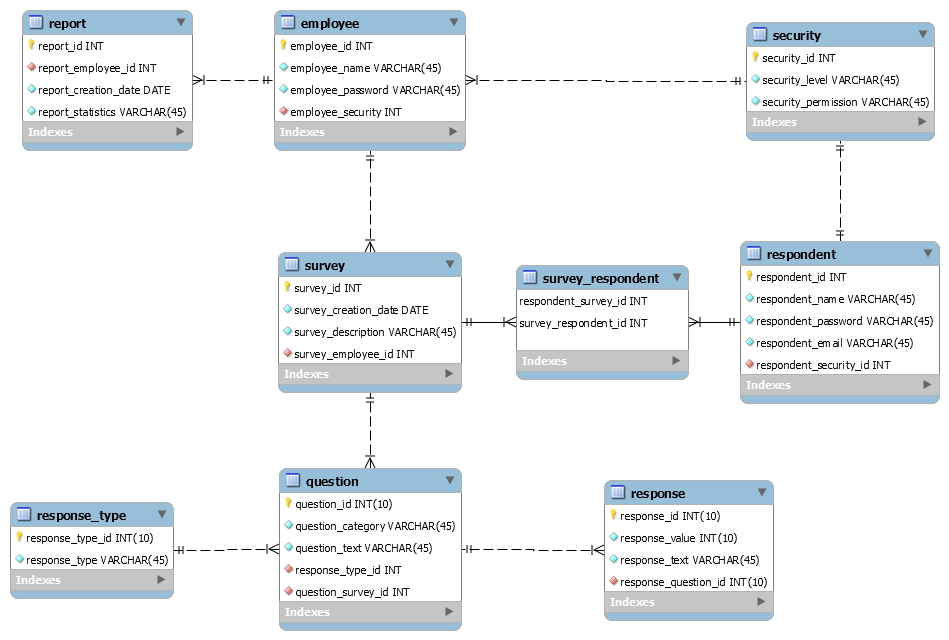
## C:\Users\flyfonfun98\AppData\Local\Microsoft\Windows\INetCache\Content.Word\ClassDiagramSMS2.jpgClass Design

The design patterns which will be applied to the system are Abstract Factory Pattern and Observer Pattern.

Abstract Factory Pattern will be used to control the instantiation of classes. As the components of the system are closely related and they use the same type of methods (add, create, edit, find, delete), this design pattern is the most suited to group the related objects and determine the selection of concrete factory classes at run-time.

The Observer Pattern will be used for the interface of the system where components need to be updated when changes occur, such as the creation of a new question, the modifications brought to questions’/surveys’ texts.

# Data Model



# Test Strategy

The types of tests performed by the system will be Unit Testing and Integration Testing.

|  |  |  |  |
| --- | --- | --- | --- |
| **Testing Method** | **Input Data** | **Test Scenario** | **Expected Result** |
| securityAccess() | The username and the password of a user | The system validates the username and password of a user. Based on the access level of a user different operations will try to be performed . | The test should fail if the user performs an action on which has restricted access, and the test should succeed if the user has permission. |
| setRespondent() | The username of a user and the survey to be completed | The application sets the respondent for the current survey, recording that a survey is on-going. Will be tested if the respondent attends to a survey. | A user will be associated with a survey |
| getLastQuestionNumber() | The on-going survey and the user which takes the survey | The system retrieves the last question from a survey the user has started. Based on the question number, the system will display the next question to the user. In case the user starts a new survey, the first question will be shown. | User gets the last unanswered question from a survey |
| getResponseType() | A question from a survey | System will test if the question exists and will retrieve and test its response type. | User is able to see the response type of the question |
| recordResponse() | A response given by the user to a question | The application validates the response of the user to match with the response type and adds it to the survey’s question. | The survey’s question will have its response field filled with data |
| recordLastQuestionNumber() | The on-going survey and the current question the user is at. | The system will test that the last answered question is recorded to the on-going survey. Tests if the next question will be shown. | Application saves the last answered question from a survey |
| recordSurvey() | The on-going survey and its respondent | Tests if after the survey has been completed, the survey has retained all the expected data. | Survey will be saved and displayed to the user |

# Elaboration – Iteration 2

# Architectural Design Refinement

*[Refine the architectural design: conceptual architecture, package design (consider package design principles), component and deployment diagrams. Motivate the changes that have been made.]*

# Design Model Refinement

## *[Refine the UML class diagram by applying class design principles and GRASP; motivate your choices. Deliver the updated class diagrams.]*

# Construction and Transition

# System Testing

*[Describe how you applied integration testing and present the associated test case scenarios.]*

# Future improvements

*[Present future improvements for the system]*