<Survey Horse>

Object Design

<1.0>

<28.12.2016>

Osman Çiçek

Ali Buğra Kanburoğlu

Emre Şaşmaz

Burak Sağlam

Prepared for

SE301 Software Engineering



Table of Contents

[1. Introduction 1](#_Toc436772639)

[1.1. Object Design Trade-offs 1](#_Toc436772640)

[1.2. Interface Documentation Guidelines 1](#_Toc436772641)

[1.3. Definitions, Acronyms, and Abbreviations 1](#_Toc436772642)

[1.4. References 1](#_Toc436772643)

[2. Packages 2](#_Toc436772644)

[3. Class Interfaces 3](#_Toc436772645)

OBJECT DESIGN DOCUMENT

# Introduction

## Object Design Trade-offs

Our trade off was about the answer option of the survey. We thought the design of answer options should be constricted because other ways it would be too difficult to implement and we have limited time. Finally, we determined 5 options for each question because it would make our model implementation simple and efficient. Our system can be added only 5 answer optioned questions.

## Interface Documentation Guidelines

System interface mainly consists of three parts and they all stay in view layer by communicating controller and model layers. First of these parts is for Surveyor. We provide an interface for the surveyor to make his/her fill the survey. Secondly, we have “Survey Manager” Interface which has create, edit and show results options. Finally, we have System Admin Interface which also has create, edit survey and show results Interfaces. Additionally, system admin has extra interface to edit the users’ information.

## Definitions, Acronyms, and Abbreviations

*Model* A schematic description of a system that accounts for its known or inferred properties

*View*  A visual representation of a model which might

*Domain* Name and address ofwebsite.

*HTTP*  is a protocol for secure communication over a computer network which is widely used on the Internet.

*GUI* Graphical User Interface

*Surveyor*  Any user can interact with the system

*Survey*  has questions about title that decided by survey manager.

*Controller*  The controller translates interactions with the view into actions to be performed by the model.

*Survey Manager: Is an actor who can add, edit the surveys and can see the survey results.*

*Unregistered Surveyor: Is an actor who can only fill the survey.*

*Registered Surveyor: Is an actor who can register and fill the survey.*

*System Admin: Is an actor who can add, edit the surveys and can edit, rename, delete user.*

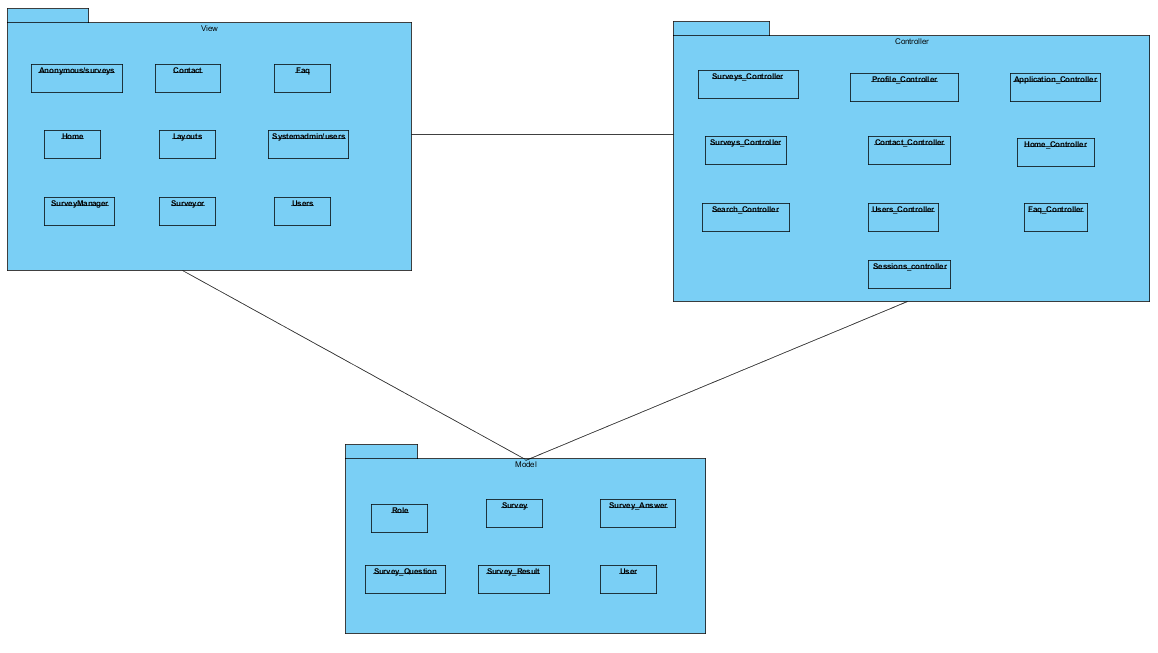
## References

1. RAD
2. SDD

# Packages

We have three subsystems these are Model, View and Controller. They are packages of the system. We use rails gems and bundles to use MVC pattern.

**Note:** The diagram below has added as .png file to the BETA folder.



# Class Interfaces

**Class: SystemAdmin**

**Class Attributes:** count (int)

**Object Attributes: users** (list)

**Method: show**Item

**Arguments: users** (User),index(int)

**Return Value: allInfo(String)**

This method show the register user abour name,email,type but except admin in the system.

**Method: editUser(list),type(role)**

**Arguments: users(User),index(int)**

**Return Value:** none

This method edit the users information that want to edit but admin can edit name,email and password.

**Method: deleteUserList**

**Arguments: user**s(User)

**Return Value: none**

This method delete the register users in the register user.

**Method:** getAllTextSurvey

**Arguments:** none

**Return Value:** allText (String)

This method gathers survey title, number of questions and link to fill by unregister.

**Method: editSurveyTitle**

**Arguments: Survey(title,id)**

**Return Value: edit** (title)

This method edit the title in the system.

**Method: deleteSurvey**

**Arguments: Survey(list),Survey(index)**

**Return Value: none**

This method delete the survey in the system..

**2.Class: SurveyManager**

**Class Attributes: id**

**Object Attributes: Survey** (list), results (list),User(id)

**Method: editProfile**

**Arguments: user**(User)

**Return Value:** none

This method edit the information about their individual information these email, password.

**Method: showSurveyResults**

**Arguments: surveys**(Survey)

**Return Value:** list (list)

This method show survey results table that contains Survey title, username that answered the questions,question and answer.

**Method: createSurvey**

**Arguments: newSurvey** (Survey)

**Return Value:** none

This method adds the given feature extractor into the SurveyList

**Method: showSurveyList**

**Arguments: listSurvey** (Survey)

**Return Value: list(Survey),link(string)**

This method show survey`s attributes surveyTitle,noq,link in the system.

**Method: editSurvey**

**Arguments: Survey** (Survey)

**Return Value: survey(Survey)**

This method edit only survey title.

**3.Class: Surveyor**

**Class Attributes: id**

**Object Attributes: searchSurvey(Survey),**

**Method: searchSurvey**

**Arguments: searchSurvey(SurveyTitle),**

**Return Value: Survey**

**This method take a title that have survey and search in the system.**

**Method: editProfileSurveyor**

**Arguments: User** (id)

**Return Value: none**

**This method edit surveyor profile information.**

**Method: fillSurvey**

**Arguments: survey**(survey\_id)

**Return Value: surveyAnswers**

**This method answer the question by the surveyor.**

**4.Class: Anonymous**

**Arguments: surveyLink()**

**Return Value: surveyAnswers**

**This method answer the question by the anonymous person.**

**5.Class: UserRegister**

**Class Attributes:id(int)** name(string),email(string),role(int),password(string)

**Object Attributes: surveyor(User),surveyManager(User)**

**Method: registerUser**

**Arguments:surveyor(User),surveyManager(User)**

**Return Value: allInfo(String)**

This method create a new user two different types that is surveyor and surveymanager.