Survey4All

Object Design

1

23.12.2019

Berkay Yılmaz

Gözde Gökyokuş

Umut Yıldız

Prepared for

SE301 Software Engineering



Table of Contents

[1. Introduction 1](#_Toc27834365)

[1.1. Object Design Trade-offs 1](#_Toc27834366)

[1.2. Interface Documentation Guidelines 1](#_Toc27834367)

[1.3. Definitions, Acronyms, and Abbreviations 1](#_Toc27834368)

[1.4. References 1](#_Toc27834369)

[2. Packages 1](#_Toc27834370)

[3. Class Interfaces 1](#_Toc27834371)

OBJECT DESIGN DOCUMENT

Object Design Document (ODD) describes object design trade-offs made by developers, guidelines they followed for subsystem interfaces, the decomposition of subsystems into packages and classes, and the class interfaces. The ODD is **used** to exchange interface information among teams and **as a reference during testing**. The audience for the ODD includes system architects (i.e., the developers who participate in the system design), developers who implement each subsystem, and testers.

Among three approaches to generate ODD, we follow “**ODD embedded into source code**” approach in SE301, since the other methods create many redundancies, inconsistencies.

The initial version of the ODD can be written soon after the subsystem decomposition is stable. Both packages and class interfaces can be generated from source code (comments!) by using a tool, which is named Javadoc. Keeping material for the ODD with the source code enables the developers to maintain consistency more easily and rapidly.

# Introduction

Describes the general trade-offs made by developers (e.g., buy vs. build, memory space vs. response time), guidelines and conventions (e.g., naming conventions, boundary cases, exception handling mechanisms), and an overview of the document. Interface documentation guidelines and coding conventions are the single most important factor that can improve communication between developers during object design. These include a list of rules that developers should use when designing and naming interfaces.

## Object Design Trade-offs

* + 1. Functionality vs Staffing

As our development group have lost a member throughout its development cycle, we had to cut some functionality from the project to deliver it by the beta test date. (Ex: Favorite Survey function)

* + 1. Use vs Build

Survey4All is a system which can be divided into modules. We found some of the modules at the market. Importing modules to your project accelerates you and increases the quality of it as well. With the contribution of them we accomplished to manage most of the functionality. Here are some of the examples we got from the market:

AngularFireModule.initializeApp(environment.firebaseConfig),

AngularFirestoreModule,

AngularFireAuthModule,

BrowserModule,

AppRoutingModule,

BrowserAnimationsModule,

MDBBootstrapModule.forRoot(),

FormsModule,

AngularFireDatabaseModule,

ReactiveFormsModule

* + 1. Delivery Time vs. Functionality

As mentioned in 1.1.1, due to member loss and our busy midterm weeks, we need to get some speed but our development is going successfully on schedule.

## Interface Documentation Guidelines

## Definitions, Acronyms, and Abbreviations

- ID : Identification

- Angular : Angular is a TypeScript-based open-source web application framework. Its goal is to augment browser-based applications with Model–View–Controller (MVC) capability and reduce the amount of JavaScript needed to make web applications functional.

- Firebase : Google’s Firebase is a mobile and web application development platform which has several services. Given examples are the services we use:

* Cloud Firestore: Stores data.
* Authentication: Authenticates users simply and securely. Registration and Login Services are easy to implement.
* Hosting: Delivers web assets with speed and securely.
* Cloud Storage: Stores and serves files at Google Cloud. (ex:Profile Picture uploading and editing)

- Angular MDB: is a UI Kit for building responsive apps and websites which has ready to use libraries including plugins, animations, icons, templates as well as modules for CSS and JS.

- Login: to get access to an operating system or application, usually in a computer.

- ODD : Object Design Document

- UI : User Interface

- Server: is a computer,provides services to other computers

- User: a person who use the system

- Exception: Represents errors that occur during application execution.

- Error: The condition of having incorrect or false knowledge.

## References

* <https://angular.io/>
* <https://firebase.google.com/>
* <https://mdbootstrap.com/docs/angular/>

# Packages

Describes the decomposition of subsystems into packages and the **file organization of the code.** This includes an overview of each package, its dependencies with other packages, and its expected usage.

# Class Interfaces

**Describes the classes and their public interfaces.** This includes an overview of each class, its dependencies with other classes and packages, its public attributes, operations, and the exceptions they can raise.