**Universität Bern**

Introduction to Software Engineering

ShopNote

Software Review Document

v1.0

Last modified on 15.11.2013

**Customer**  
Team 8

**Project**

Shopping-List

**Autors**  
Raul Bolaños, Nicolas Kessler, Theodor Truffer, Lukas Zahnd

Table of Contents

1. Introduction

1.1 Purpose

1.2 Stakeholders

1.3 Definitions

1.3.1 Shopping list

1.3.2 Android/iOS/Windows Mobile

1.4 System overview

1.4.1 Basic functions

1.4.2 Planned extensions

1.5 References

1.5.1 Our references

1.5.2 Program references

2. Use cases

2.1 Diagram

2.2 Use cases

2.2.1 Create new shopping list

2.2.2 Update shopping list

2.2.3 Delete shopping list

2.2.4 Add items to a shopping list

2.2.5 Manage items

2.2.6 Share with friends with app

2.2.7 Share with friends without app

2.2.8 Find a shop

2.2.9 Get shop location notification

2.2.10 Access archive function

2.2.11 See statistics

2.2.12 Access database

2.2.13 Set appearance

3. Specific requirements

3.1 Functional requirements

3.2 Nonfunctional

requirements

# Assignment

The assignment consists of 3 points:

* Write a constructive and useful review of the project assigned to your team using the checklist template reported below. The review should serve as guideline to improve both code and design.
* Analyze and describe the strategy used to persist data. You can use a UML digram to be more clear in your description. Split the adopted strategy in multiple steps and criticize the overall design. If a specific step of strategy is not well design, provide details on how it can be improved.
* Pick an activity of choice and analyze its code. Does the class have too many responsibilities ? Is there some logic that should be moved to another class ? If so, why ?

The Review must be saved in the assigned repository (NOT yours) at the following location: /Review/

Accepted file types: txt, doc, pdf (diagrams can be embedded or saved as image files)

Design

* Violation of MVC layers
* Usage of helper objects between view and model
* Rich OO domain model
* Clear responsibilities
* Sound invariants
* Overall code organization & reuse, e.g. views

Coding style

* Consistency
* Intention-revealing names
* Do not repeat yourself
* Exception, testing null values
* Encapsulation
* Assertion, contracts, invariant checks
* Utility methods

Documentation

* Understandable
* Intention-revealing
* Describe responsibilities
* Match a consistent domain vocabulary

Test

* Clear and distinct test cases
* Number/coverage of test cases
* Easy to understand the case that is tested
* Well crafted set of test data
* Readability