**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

[Table of Contents 1](#_Toc368778285)

[Changelog 2](#_Toc368778286)

[Introduction 3](#_Toc368778287)

[Use Cases 3](#_Toc368778288)

[#1 View mensa list 3](#_Toc368778289)

[#2 View mensa details 4](#_Toc368778290)

[#3 Show upcoming menus 4](#_Toc368778291)

[#4 View Map 4](#_Toc368778292)

[#5 Find mensa 5](#_Toc368778293)

[#6 Set favourite mensa 6](#_Toc368778294)

[#7 Add Friends 6](#_Toc368778295)

[#8 Delete Friends 7](#_Toc368778296)

[#9 Notify Friends 8](#_Toc368778297)

[#10 Join friends for lunch 9](#_Toc368778298)

[#11 Rate a menu 9](#_Toc368778299)

[#12 Menu’s rating 10](#_Toc368778300)

[#13 Favourite food 10](#_Toc368778301)

[#14 Set default language 11](#_Toc368778302)

[#15 Switch Language 11](#_Toc368778303)

[Use Case Diagram 13](#_Toc368778304)

[Specific requirements 14](#_Toc368778305)

[Functional requirements: 14](#_Toc368778306)

[Non-functional requirements: 14](#_Toc368778307)

# 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