<NeYesek>

<Muhammet Şeramet - 150115069>

<Mahmut Aktaş - 150115010>

<Ali Beytullah Özkan - 150114043>

**1. INTRODUCTION**

**1.1) Purpose**

This document will outline in detail the software architecture and design for the NeYesek. This document will provide several views of the system's design in order to facilitate communication and understanding of the system. It intends to capture and convey the significant architectural and design decisions that have been made for the NeYesek.

**1.2) Scope**

The scope of this document is to provide the architecture and design of the NeYesek. This document will show how the design will accomplish the functional and non-functional requirements in the NeYesek SRS document.

**1.3**) **Definitions, Acronyms and Abbreviations**

**SRS:** Software Requirements Specification

**AS:** Assumption

**CO:** Consideration

**1.4) References**

* <http://www.se.rit.edu/~vdkrit/design/VDK-RIT_SDS.doc>
* NeYesek Software Requirements Specification Document

**2. DESIGN CONSIDERATIONS**

**2.1) Design Assumptions**

See NeYesek Software Requirements Specification Section 2.2 for details.

**2.2) Considerations**

See NeYesek Software Requirements Specification Section 2 for details.

**2.3) Environment**

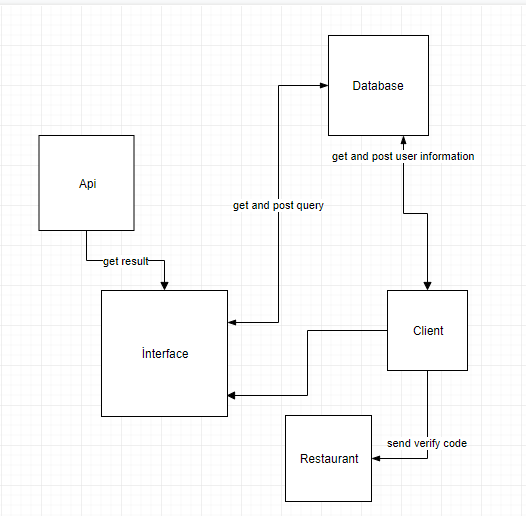
See NeYesek Software Requirements Specification Section 2 for details.

**3. SYSTEM DESIGN**

**3.1) Architectural Strategies**

* The design shall use Object Oriented Principles (OOP). The trade-off of increased code overhead and object message passing is considered justified by the gain of modularization of functionality, data encapsulation, communication through interfaces and re-use through polymorphism. In addition, the entire team is familiar with this paradigm and a design of this type will facilitate communication amongst the developers.
* Authentication will be limited to password checking on initial login and a session ID subsequently. This is considered sufficient to the low risk nature of the data.
* Application needs past restaurants for suggestions, so we save user’s restaurant history on DB. This action is goods for memory using.
* NeYesek, will connect to Google Map APIs. Thus, the results of the restaurant will be instantaneous and we will not have to save the results in a DB.
* Level systems; when people use a suggestion, they will take point and have a level. These user information’s save DB, so user information’s are protected for changing.
* Discount codes valid for a certain period of time show user and send restaurant. This action is on client, when user click button, a code created and send.

**3.2) Architecture Design**



**3.2.2 Sub Components**

• API; Google Maps Search API.

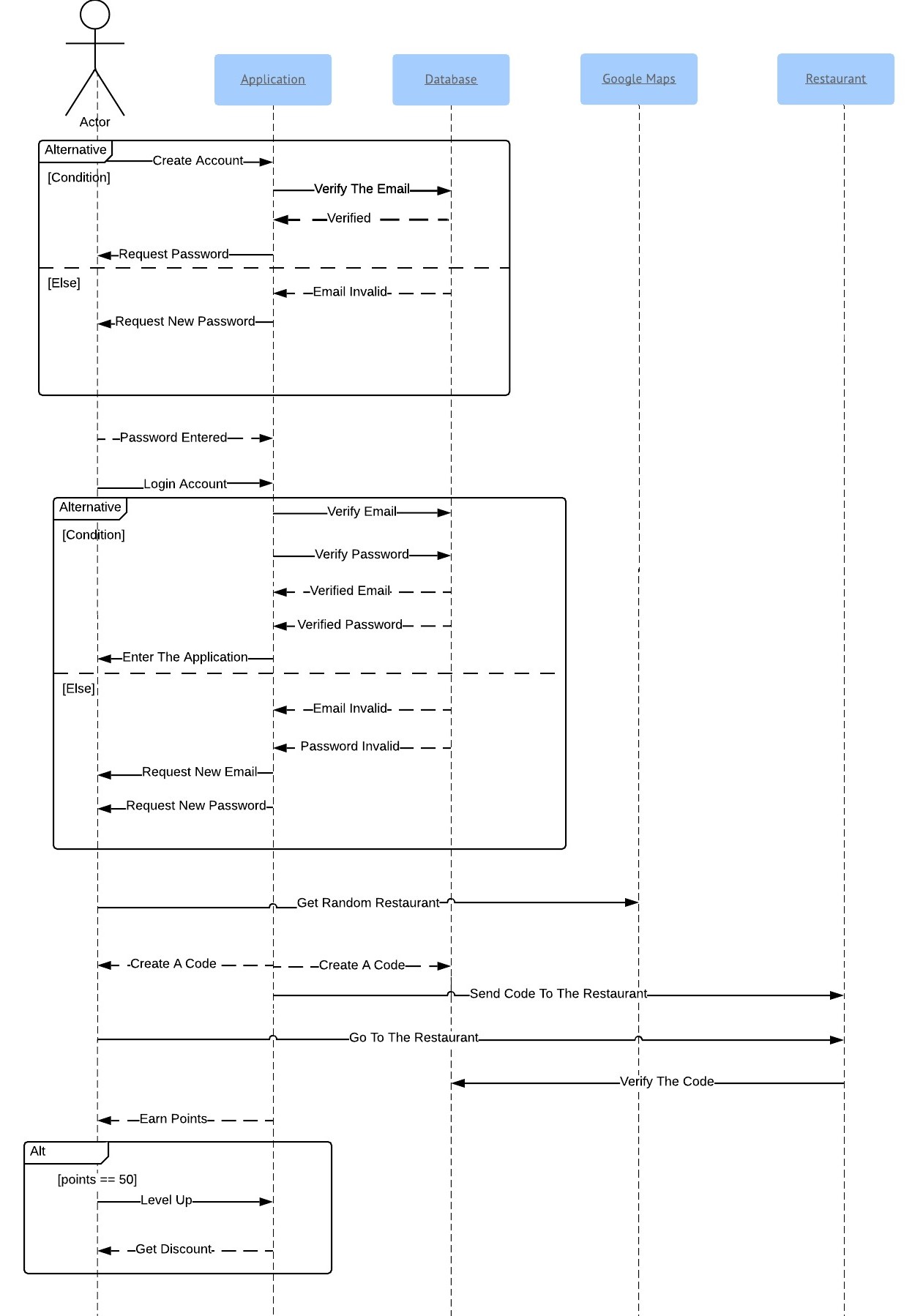
• Database; LiteSQL.

• Client; Users’ Android Phones

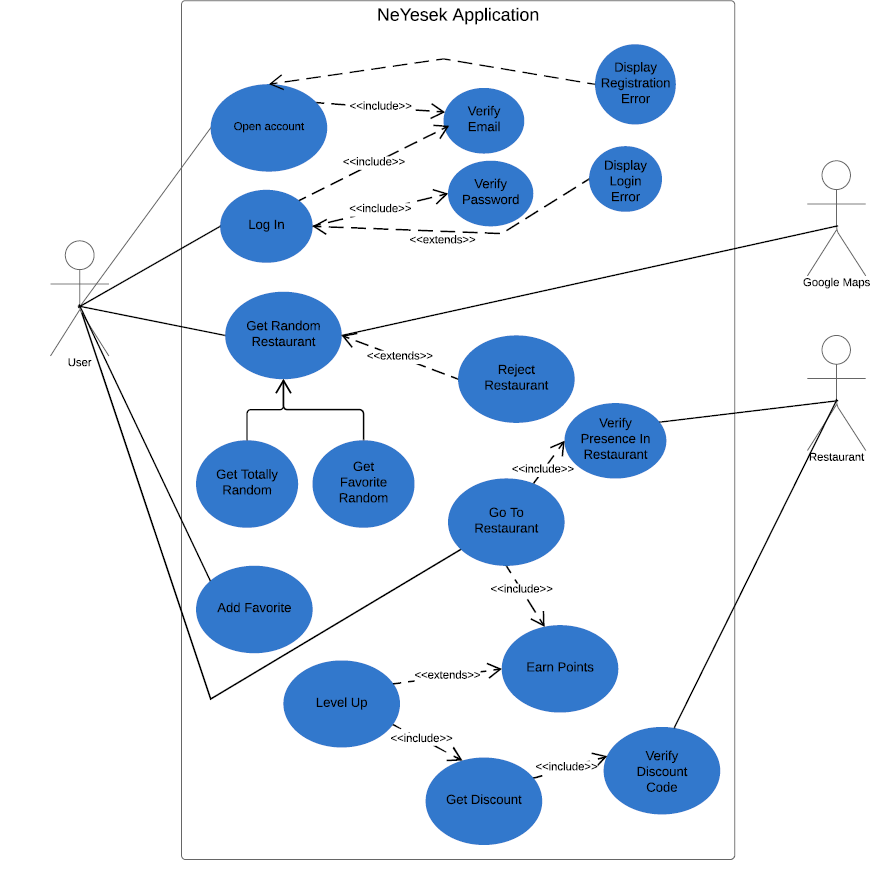
• Restaurant; given discount restaurant

**CLASS DIAGRAMS**

**Sequence Diagram**



**Use-Case Diagram**



**CONCLUSION**

In this document we tried to give information about software architecture and system design of NeYesek as much as we can. We will use this class diagrams and design strategies while implementing the code. This document will enable us to code clearer and well disciplined. We will be using this document a lot while developing application.

**Participation to the document**

