<A BIDV based smart banking app>

Software Architecture Document

Version <1.0>

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| <20/12/2022> | <1.0> | <Analysis and write usecase> | <Đàm Thị Linh> |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1. Introduction 2

1.1 Purpose 2

1.2 Scope 2

1.3 Definitions, Acronyms, and Abbreviations 2

1.4 References 2

1.5 Overview 2

2. Architectural Representation 2

3. Architectural Goals and Constraints 2

4. Use-Case View 2

4.1 Use-Case Realizations 2

5. Logical View 2

5.1 Overview 2

5.2 Architecturally Significant Design Packages 2

6. Process View 2

7. Deployment View 2

8. Implementation View 2

8.1 Overview 2

8.2 Layers 2

9. Data View (optional) 2

10. Size and Performance 2

11. Quality 2

Software Architecture Document

# Introduction

## A smart banking app is an application that helps you make transfers and bill payments quickly and safely on all operating systems and devices such as phones, tablets and laptops. Granting a user an account, smart banking provides services used conveniently as an app that can nearly be replaced for the real bank. The system can produce statistical data for the records. The technology enables the establishment of a transaction repository that is later used to generate documents, allowing for greater flexibility in the holding period and cash flow of the transaction history. Additionally, it is capable of performing an automatic balance check.

This system design is divided into several viewpoints and parts, each of which is described in detail. The 5 view model serves as the reference model for this essay.

## Purpose

This document provides a comprehensive architectural overview of the system, illustrating several system aspects from many architectural angles. Its goal is to describe and document the key architectural choices made for the system. This paper includes descriptions of the system architecture from five different angles. Both the static and dynamic behavior of the system are covered in this article. The descriptions of all the necessary diagrams are included in this paper.

The software can be represented as exactly as feasible using a five-view model. It makes it possible for a variety of stakeholders to find the data they need in the architecture document.

## Scope

The system's static and dynamic components are both covered in the software architecture document. In order to make the document complete and consistent, the five view model is used as a reference model. It contains many various viewpoints of the system.

The class diagram, package diagrams, and other static architecture designs are examined in the context of the system's static behavior. The dynamic properties of the system are explained via use case realizations and system sequence diagrams.

## Definitions, Acronyms, and Abbreviations

## References

## Overview

In this paper, the smart banking architecture will be in-depthly investigated. In the portions that follow, the project's architecture representation—including architectural representation, architectural ambitions and restrictions, and use case realizations—is discussed in more detail. The following sections discuss the particulars of the system's four main views (logical view, process view, development view, and implementation view). The document also talks about data views, performance, and quality views.

# Architectural Representation

This section details using the views defined in the 5 models to build the architecture: use case view, process view, deployment view, and implementation view. The following are the views used to document the smart banking application:

## 2.1. Use case view

Audience:

All the stakeholders of the system, including the end-user.

Area:

Describes a number of use examples and/or scenarios that illustrate some of the most crucial and fundamental aspects of the system. Describe the actors and cases in the system. Along with the basic workflow, the article also covers exception cases, exception outputs, and other relevant use cases.

## 2.2. Logical view

Audience:

Designers, programmers, testing staff

Area:

Functional requirements, object hierarchy, system layers

Describes the object model's design. The subsystems and their interactions are also described.

## 2.3. Process view

Audience:

Integrators, programmers

Area:

Non-functional requirement

Describes the concurrency and synchronization features of the design. Define the system's behavior at runtime.

## 2.4. Deployment view

Audience:

Programmers, code testers

Area:

Software component: Describes the modules and subsystem divisions of the system.

## 2.5. Implementation view

Audience:

Database administrators, system engineers, development managers

Area:

Persistence: Describes the data model's architecturally significant persistent element. The mapping of software to hardware is described, as well as the distributed features of the systems.

# Architectural Goals and Constraints

## 3.1 Application side

Online smart banking will be hosted on a JPS server. This underlying client OS can be any mobile operating system (OS, android) because this is a mobile application. MySQL will be used as the core database server. All client communication must conform to public HTTPS and TCP/IP communication protocol standards.

## 3.2. Client side

Users can not access online smart banking without the internet. This will be hosted as an independent application allowing users to access the functionality through the main system. To receive the full experience, clients/users are expected to update to the latest version of the app.

**3.3** **Security**

The main security measure is under the control of institute officials. They will have access to both database and application functions. All user accounts for transactions must be created by the admin team. After being added to the system, users will receive a default password that they can alter. All passwords are encrypted to provide a higher level of security. The user's comments and pledges are only visible to the authorized administrator and instructor.

## 3.4 Persistence

A single server will function as the central repository for all the data. This is a third rational database that is implemented in normal form. To maintain ACID (Atomicity, Confidentiality, Integrity, and Durability), some measures have been implemented, like encrypting passwords and employing transactions for all database commits, among other things.

## 3.5 Reliability

The system will go through many testing procedures (unit testing, integration testing, and system testing) before being deployed to make sure it is dependable. The MYSQL database server can accommodate many clients at once while preserving the consistency and integrity of the data.

## 3.6 Performance

Within 3 seconds of the standard database and web server script timeouts, the system responds to any request. The hardware, network, and internet connection capabilities that are available also have an impact on system speed. Particularly lengthy tasks that include the generation of statistical data are possible. As a result, real performance can only be assessed following the development and testing of the system.

## 3.7 Portability and reuse

Online smart banking is intended to operate independently, however it can be modified to communicate with other payment systems like Momo, Zalopay... To maintain reusability, each capability is very carefully stacked and structured. FUP best practices are adhered to throughout the project, and OOP concepts are meticulously upheld.

## 3.8 Development tools

The project requires many development tools.

Programming: Intellij IDEA, visual studio code

Database: XAMPP

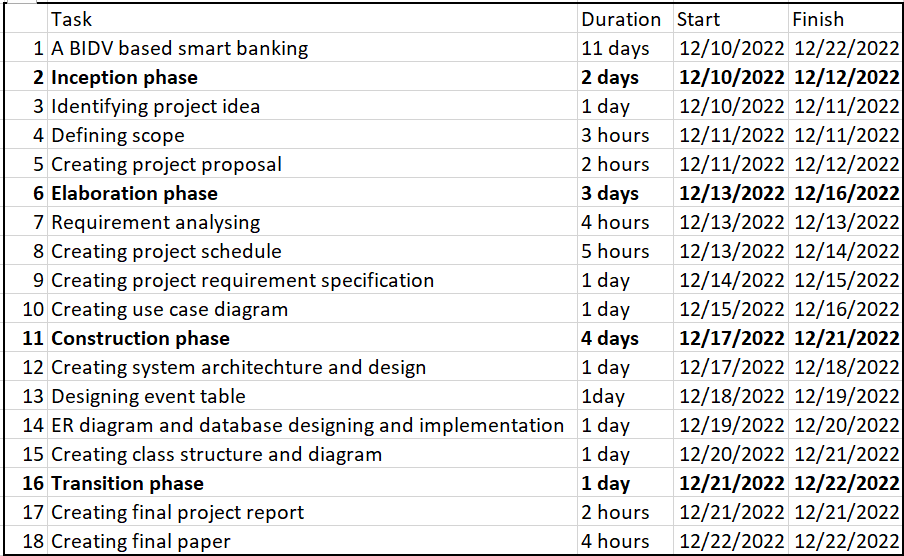
Diagrams: visual diagram

Database connection: MYSQL connector

Schedule: Microsoft Project

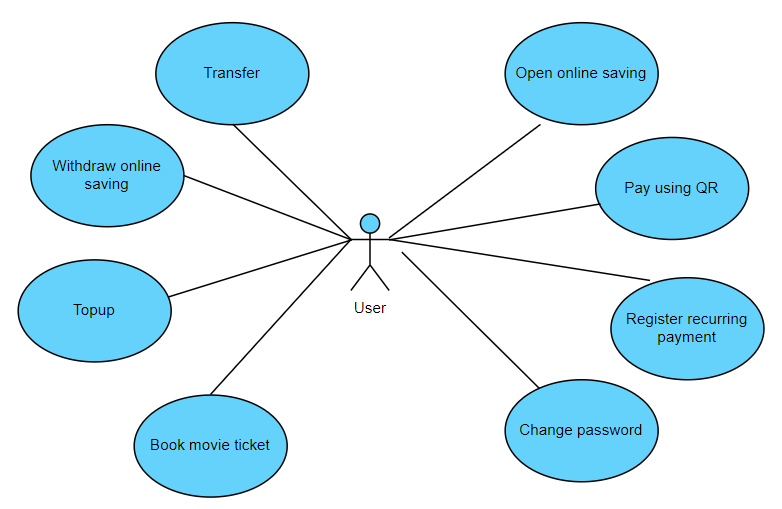
## 3.9 Schedule

The project is being developed using RUP methodology. The four categories of operations are inception, elaboration, construction, and transition. All deliverables and related deadlines are analyzed in this job breakdown structure and met.



# Use-Case View

The following diagram depicts 8 use cases in the following system. All the detailed descriptions of each use case are represented in the use case specification and use case realization docs.



# Logical View

## Overview

For flexible programming languages, frameworks and developer’s technical skills specialized in, the online smart banking system is divided into three main subsystems: Presentation, Application and Network. Additionally, as an executed system, it has the interaction from the user via Browser to access Presentation and a Database for storing all the data of the system, which is used by Application.

5.1.1 Presentation

This is the front-end part of the system which contains all the views of the system. The purpose of this subsystem is to display UI to users and perform the functions that do not need to access the database.

5.1.2 Application

This is the back-end part of the system which performs the functions that need to work with the database. It can be divided into Controller, Repository and Domain packages for specific functions

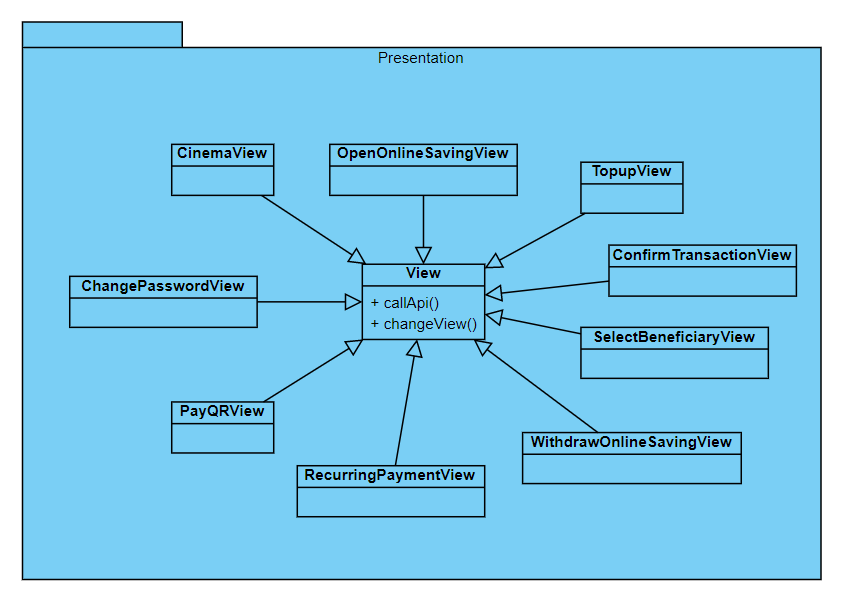
5.1.3 Network

This is the connection to connect Presentation and Application with the user via Browser. The user’s experience, whether good or not, partly depends on this subsystem. It conforms to the TCP/IP model, which consists of four layers: Transport layer, Application layer, Internet layer and Network access layer.

## Architecturally Significant Design Packages

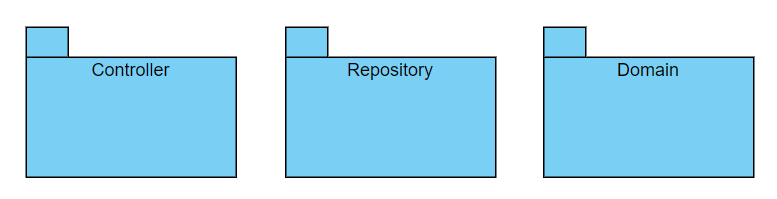
5.2.1 Presentation

Although the following view does not cover all the whole system, it offers some most basic view of the smart banking system and will be used by the Application in the use cases presented.

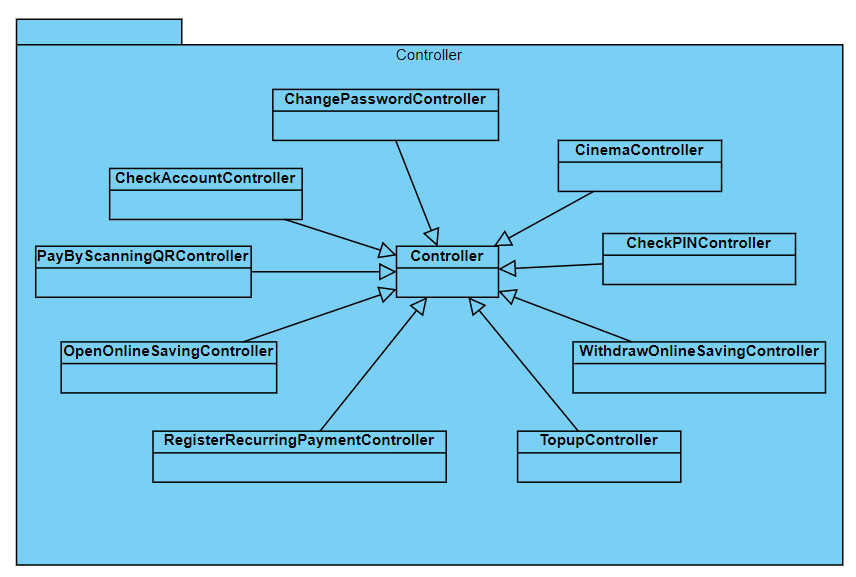


5.2.2 Application

The back-end part of the system includes a Controller, which is called by some of the views in Presentation, then executes the corresponding function to manage data and return data if needed to the view, a Repository which works with data and is called by some controllers in controller package and Domain which maps necessary objects, used by both Controller and Repository.



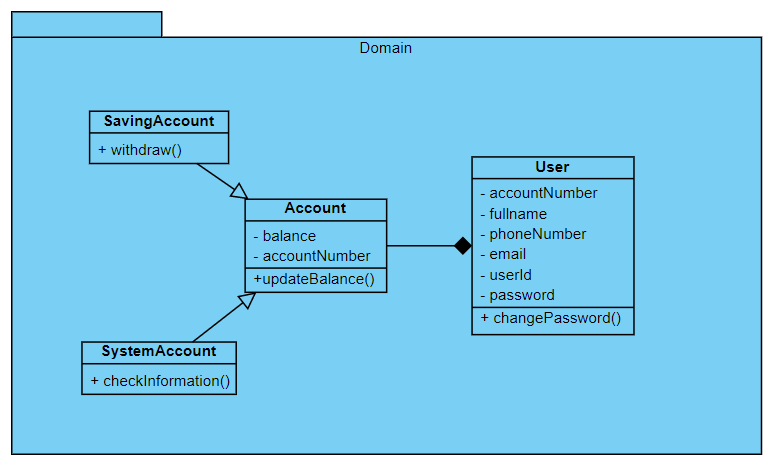
5.2.2.1 Controller package



5.2.2.2 Repository package

5.2.2.3 Domain package

It is clearly seen that there are an amount of domain models in this system and especially applying in 8 use cases offered above. But all of them will work with User, make some changes to User domain (change password) and its Account (transfer, topup, pay by scanning QR use cases…) and subclass of Account which is SavingAccount (open online saving and withdraw online saving use cases). As illustrated on the diagram, a User must have at least one Account.

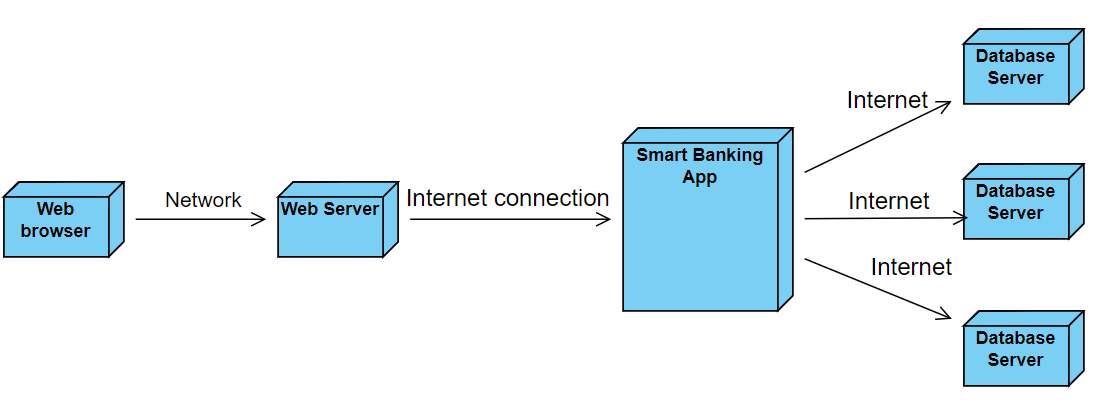


5.2.3 Network

# Process View (optional)

# Deployment View

With any mobile device is capable of running web browser with the internet provided, when the user call any api to web server, which has smart banking app deployed on, then the user will get a response from the system right after the smart banking app executed (may use Database server with the internet connection). The database will be hosted in some other hosting space.



# Implementation View

# There are no considerable changes when implementing the system. Then, this view will be ignored in this document.

# Data View (optional)

# Size and Performance

# The software as designed will support 100,000 concurrent users.

Data storage under 10MB per user (including history transaction).

With maximum load all transactions well under standard server script / database connection timeout – 30 seconds.

# Quality

The software as described above supports the existing online smart banking graphical standards, interfaces with the existing online smart banking server, and provides a self-describing user interface.