Flashcard App

Architecture/Design Document

**Table of Contents**

[**1**](#_gjdgxs) **INTRODUCTION 3**

[**2**](#_30j0zll) **DESIGN GOALS 4**

[**3**](#_1fob9te) **SYSTEM BEHAVIOR 4**

[**4**](#_3znysh7) **LOGICAL VIEW 5**

[**4.1**](#_2et92p0) **High-Level Design (Architecture) 5**

[**4.2**](#_1t3h5sf) **Detailed Class Design 5**

[**5**](#_4d34og8) **PROCESS VIEW 6**

[**7**](#_2s8eyo1) **PHYSICAL VIEW 6**

[**8**](#_17dp8vu) **USE CASE VIEW 7**

Change History

**Version:** 1.0

**Modifier:** Adrian Ridder

**Date:** 04/05/2019

**Description of Change:** Document creation, UML diagrams created.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Introduction

This document describes the architecture and design for the Flashcard application being developed for University of Missouri—Kansas City (UMKC). The Flashcard application is a study tool that allows users to create, edit, and study virtual flashcard decks via their web browser. The system tracks each user’s progress, and the user will be able to see their associated statistics for each deck.

The purpose of this document is to describe the architecture and design of the Flashcard application in a way that addresses the interests and concerns of all major stakeholders. For this application the major stakeholders are:

* Users and the customer – they want assurances that the architecture will provide for system functionality and exhibit desirable non-functional quality requirements such as usability, reliability, etc.
* Developers – they want an architecture that will minimize complexity and development effort.
* Project Manager – the project manager is responsible for assigning tasks and coordinating development work. He or she wants an architecture that divides the system into components of roughly equal size and complexity that can be developed simultaneously with minimal dependencies. For this to happen, the modules need well-defined interfaces. Also, because most individuals specialize in a particular skill or technology, modules should be designed around specific expertise. For example, all UI logic might be encapsulated in one module. Another might have all business logic.
* Maintenance Programmers – they want assurance that the system will be easy to evolve and maintain on into the future.

The architecture and design for a software system is complex and individual stakeholders often have specialized interests. There is no one diagram or model that can easily express a system’s architecture and design. For this reason, software architecture and design is often presented in terms of multiple views or perspectives [IEEE Std. 1471]. Here the architecture of the Flashcard application is described from 4 different perspectives [1995 Krutchen]:

1. Logical View – major components, their attributes and operations. This view also includes relationships between components and their interactions. When doing OO design, class diagrams and sequence diagrams are often used to express the logical view.
2. Process View – the threads of control and processes used to execute the operations identified in the logical view.
3. Development View – how system modules map to development organization.
4. Use Case View – the use case view is used to both motivate and validate design activity. At the start of design the requirements define the functional objectives for the design. Use cases are also used to validate suggested designs. It should be possible to walk through a use case scenario and follow the interaction between high-level components. The components should have all the necessary behavior to conceptually execute a use case.

# Design Goals

There is no absolute measure for distinguishing between good and bad design. The value of a design depends on stakeholder priorities. For example, depending on the circumstances, an efficient design might be better than a maintainable one, or vise versa. Therefore, before presenting a design it is good practice to state the design priorities. The design that is offered will be judged according to how well it satisfies the stated priorities.

The design priorities for the Flashcard application are:

* The design should minimize complexity and development effort.
* The design should maximize productivity by promoting clear divisions of labor. The database side should be independent of the backend, frontend should be separate from the backend, etc.

# System Behavior

The use case view is used to both drive the design phase and validate the output of the design phase. The architecture description presented here starts with a review of the expect system behavior in order to set the stage for the architecture description that follows. For a more detailed account of software requirements, see the requirements document.

When the user starts the system, the system should present to the user a list of their card decks, the option to create a card deck, and the option to study a card deck. The study session should quiz the user in a typical flashcard manner. At the end of the study session, the user should be presented with their progress, then take the user back to the home screen.

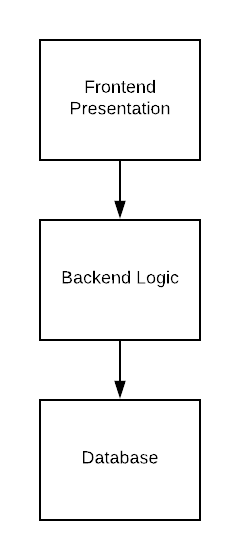
# Logical View

The logical view describes the main functional components of the system. This includes modules, the static relationships between modules, and their dynamic patterns of interaction.

In this section the modules of the system are first expressed in terms of high level components (architecture) and progressively refined into more detailed components and eventually classes with specific attributes and operations.

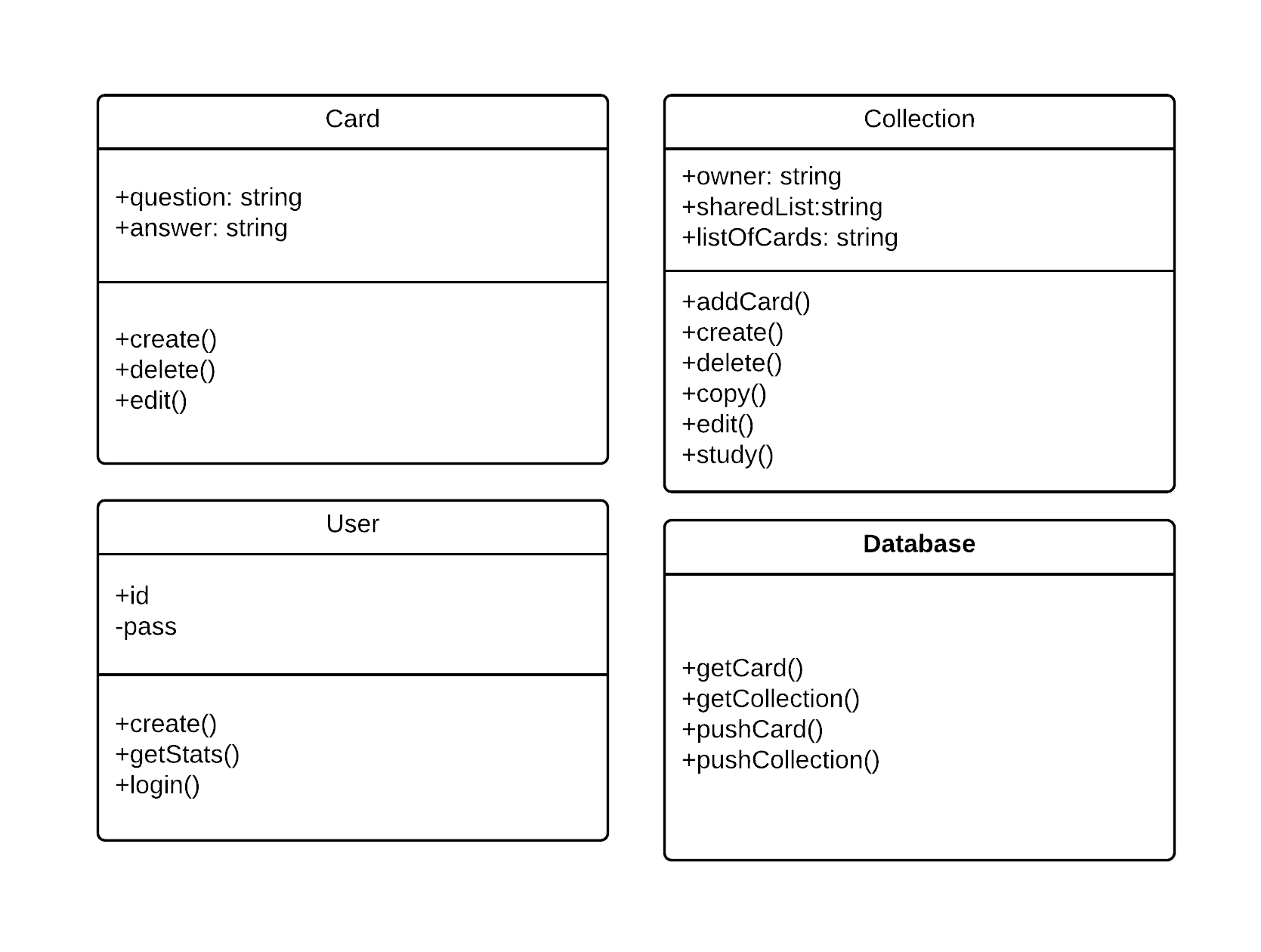
## High-Level Design (Architecture)

The high-level view or architecture consists of 3 major components:

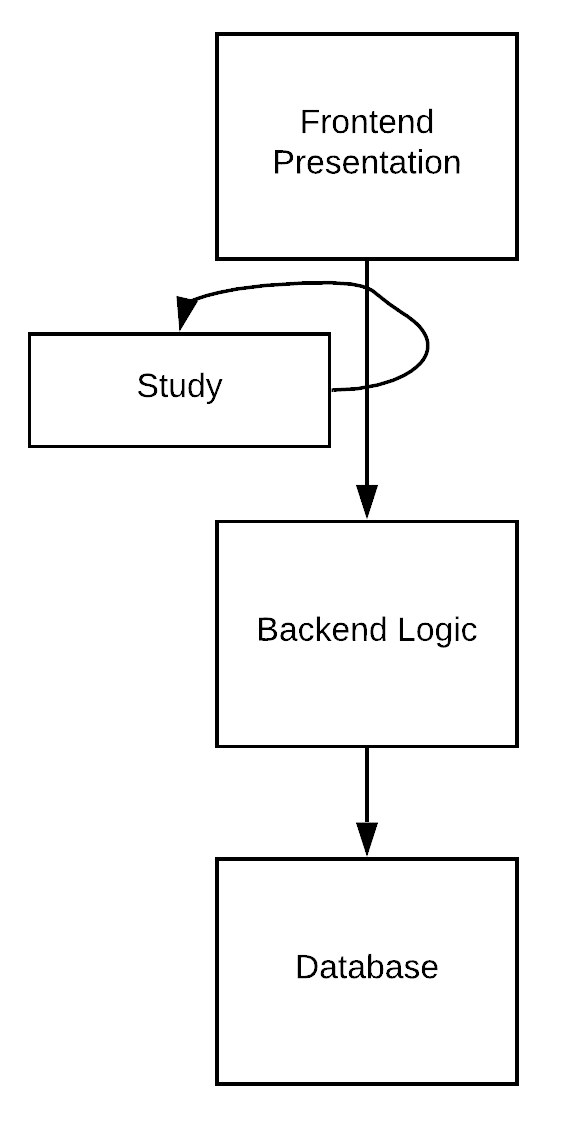


* **Frontend Presentation**: the frontend presentation consists of the HTML, CSS, and Javascript that will be used to display the flashcards.
* **Backend Logic:** the backend logic consists of the C# code used to code the program.
* **Database:** the database consists of the user’s login information, their associated flashcard decks, the flashcards within each deck, and the user’s statistical information.

## Detailed Class Design



# Process View



# Physical View

[TBD]

# Use Case View

