FlashPig
SDD

Jesper Bergquist Wendy Pau Madeleine Xia
Salvija Zelvyte

October 16, 2020

Contents

1 Introduction

Flashpig is an application that provides the opportunity to create flashcards and use them to study and learn in three different ways.

The first way is "Flashcard" which consists of two sided cards, where one side has a question while the other has an answer, both sides can be appeared as picture or text. Before starting this mini game the user is offered to choose which side of the card will be shown first. After each card the user has to decide if the card

Användaren får chansen att välja om de vill se framsidan först av ett kort och därefter baksidan eller tvärtom. Efter varje kort ska användaren välja svårighetsgraden (easy, medium eller hard), utifrån dessa kommer en "flashcard progress" byggas upp och vara tillgänglig för att visas för användaren i startsidan.

De andra två sätten är minispelen "Memory" och "Pair up". Syftet med "Memory" är att användaren ska para ihop fram-och baksida genom det klassiska memoryspelet. "Pair up" ger däremot användaren fler svarsalternativ för att para ihop med rätt framsida av respektive kort. Användarens inlärning kan följas på flashcard spelet via information om vilka kort som användaren har enkelt, medel och svårt med.

The other two ways are the mini-games "Memory" and "Pair up". In Memory the users goal is to pair the front and back of a flashcard through the classic memory game style. "Pair up" is similar to "Memory" but in this game all the cards are revealed and the user has to pair up the right front and backside.

The application is designed for students and other users that want to learn new information in a quick, effective and fun way.

Applikationen riktar sig mot studenter och andra användare som vill lära sig information snabbt, effektivt och roligt. Med de nya inlärningstekniker applikationen ger ämnar vi att skapa ett bestående komplement till de inlärningstekniker som redan finns på marknaden idag.

1.1 Definitions, acronyms, and abbreviations

2 System architecture

The most overall, top level description of your application. If your application uses multiple components (such as servers, databases, etc.), describe their responsibilities here and show how they are dependent on each other and how they communicate (which protocols etc.) You will to describe the 'flow' of the application at a high level. What happens if the application is started (and later stopped) and what the normal flow of operation is. Relate this to the different components (if any) in your application.

2.1 Dependency analysis

3 System Design

Draw an UML package diagram for the top level for all components that you have identified above (which can be just one if you develop a standalone application). Describe the interfaces and dependencies between the packages. Describe how you have implemented the MVC design pattern. Create an UML class diagram for every package. One of the packages will contain the model of your application. This will be the design model of your application, describe in detail the relation between your domain model and your design model. There should be a clear and logical relation between the two. Make sure that these models stay in 'sync' during the development of your application. Describe which (if any) design patterns you have used. The above describes the static design of your application. It may sometimes be necessary to describe the dynamic design of your application as well. You can use an UML sequence diagram to show the different parts of your application communicate an in what order.

3.1 MVVM

Insert UML here

Model

ViewModel

View

3.2 Design Patterns

3.2.1 Adapter pattern

3.2.2 Observer pattern

3.2.3 Singleton pattern

3.3 UML sequence diagram

4 Persistent data management

If your application makes use of persistent data (for example stores user profiles etc.), then explain how you store data (and other resources such as icons, images, audio, etc.).

5 Quality

Describe how you test your application and where to find these tests. If applicable, give a link to your continuous integration. List all known issues. Run analytical tools on your software and show the results. Use for example: – Dependencies: STAN or similar. – Quality tool reports, like PMD. NOTE: Each Java, XML, etc. file should have a header comment: Author, responsibility, used by ..., uses ..., etc.

5.1 Access control and security

If you applications has some kind of access control, for example a login, of has dif- ferent user roles (admin, standard, etc.), then explain how you application manages this.

NA

6 References

List all references to external tools, platforms, libraries, papers, etc. The purpose is that the reader can find additional information quickly and use this to understand how your application works.