Functional Design Specification
for
eFlash

Final Version
Revision 1.0

Authors: David Chu, Sean Gabriel, Daniel Honegger, Darren Lam, Ying Tat Ng, Aretha Samuel, Kenneth Wong, Tony Wu, Anthony Yee

Date: May 4, 2006

PROJECT ABSTRACT

The eFlash team is designing a study-skills application designed to eliminate a student's need for paper flash cards. Our software intends to replace these vital learning tools while improving on the issues inherent in a pen-and-paper system.

DISTRIBUTION NOTES:

Review Deadline Date: _____
Review Meeting Date: _____

Comment(s):

_____
_____
_____
_____

Reviewer List:

Distribution List:

Document Revision History

Revision	Date	Author(s)	Comments
0.1	2006-02-29	Wu	Initial draft
0.2	2006-03-04	Lam, Ng	External Dependencies, Testing Strategy, Effect on Methodology
0.3	2006-03-05	Wu, Wong	System Architecture
0.4	2006-03-05	Gabriel	Intro to Requirements/Solutions
0.5	2006-03-06	Yee, Chu, Honegger	Features, Use Cases
0.6	2006-03-06	Samuel	System Metaphor, Database Tables
0.7	2006-03-06	Wu	Preliminary version with revisions
1.0	2006-04-10	Gabriel	All issues resolved

Product Overview

Purpose

This document describes a functional design solution to a particular problem as identified by a requirements specification. This document, the functional design specification, is a living document that reflects changes to the project. These changes generally take the form of issues that are unclear initially but become resolved after further research or experience.

Short Description

eFlash has been designed around the need of students from various academic disciplines to compile their subject knowledge into one easy-to-use repository. The natural inspiration for this tool was the flash card, a study-skills device that has withstood the test of time. But for all their uses, flash cards have many shortcomings that we believe can be improved through the application of modern technology.

To this end, we have incorporated an intuitive interface to model the already familiar world of flash cards, an easily searchable database of input terms and definitions, export functionality across a variety of file and print formats, heuristic analysis that challenges our users to really learn their weakest material, and collaborative support to encourage knowledge sharing between all users of eFlash. These features, drawn from the requirements of our requirements specification, will be delivered by an event-driven GUI application that is primarily user-centric in nature.

Authors

Although architecture has shaped this design document by architecting the software components that satisfy the requirements of our customers, all team members have played a vital role in translating this document into its current state today.

Name	Team Role
David Chu	Development
Sean Gabriel	Project Leader
Daniel Honegger	Development
Darren Lam	Backstop
Ying Tat Ng	Development
Aretha Samuel	Marketing
Ken Wong	Visionary
Tony Wu	Architecture
Anthony Yee	Development

Target Audience

This document is used by a variety of entities. The key audience is development, who should pay emphasis to the flowing parts of the document: System Architectures, Features – Compatibility, Features – User Interface, and Features – Other Interfaces. However, Marketing will also refer to this document both to guide their activities as well as to check that all is in accordance with their understanding of the project as a whole. Only rarely are customers directly involved with this document.

Specification Flow

The functional design specification is finalized after the requirements specification is approved and is the core document for the project. A particularly large development effort may have a set of functional specifications.

Review Considerations

The primary review objective of the Design Specification is to ensure that the details of the Requirements Specification are met.

Introduction

Introduction – Requirements

When designing eFlash, we were looking for all of the problems in students' current ways of studying, and primarily focused upon the difficulties of using flash cards as an exclusive study aid. Customer feedback pointed to several recurring ideas:

1. **Flexibility.** Flash cards are often a "one-time deal," and once created are difficult to edit and embellish without starting over again. Our customers appreciate the freedom of a blank canvas when laying out, coloring, and otherwise designing their flash cards, but recount the tedium in perfecting the cards for use. They also point to the wasted effort spent in correcting minor mistakes to a flash card.

2. **Self-testing.** The name of flash cards implies their use, i.e. a student will quickly view one side to see if he/she knows the corresponding answer on the other side. Our customers appreciate being able to control the pace of such an exercise, but also spend a great deal of time tracking their own progress and making mental notes, which detracts from their overall studies. Occasionally, there are also defects in the media (seeing through to the other side) that defeat the purpose of such quizzing.

3. **Mixed media support.** While flash cards are the preferred study tool for foreign language vocabulary and others sets of terms and definitions, they are impractical for storing visual information (art, diagrams, and some complex foreign alphabets) and impossible for storing audio information (music, pronunciation).

4. **Sharing and reusability.** Every semester, thousands of students across the country invariably end up making near-identical decks to study similar material for a common subject. The problem lies in the limitations of the medium – sharing a deck with a fellow student requires a physical meeting, and the creator ends up losing his or her copy of the deck as a result.

5. **Search.** Finding a card within a large deck is a fully manual and time-consuming process for a student. Generally, decks are built with categories in mind, but having only one piece of physical media makes it difficult to cross-reference cards across multiple decks without creating duplicates, again a time-consuming process. Changing one such card would also require a change to all of the duplicates.

Introduction – Solution

Note: the reader is advised to refer to the terminology section of this document for more information on the following discussion of product functionality.

Referring to the above section, we have broken up the functionality of eFlash into discrete components that embody all of the strengths of flash cards while overcoming the weaknesses described by our customers. Together, they form a complete solution for replacing paper-based flash cards as the primary study device for our customers.

1. **eFlash Creator.** This subcomponent of eFlash is responsible for creating flash card decks that can be used either in the Viewer or in another application (through our export functionality). It is compromised of the following functionality:

   - File import functionality designed to collect lists of flashcard data quickly (1).
   - Support for three types of objects on flashcards:
     1. Text – A single area of text that supports custom text formatting.
     2. Visual – A movable resizible image. (3)
     3. Sound – A sound player with control buttons (play and stop). (3)
   - A simple interface that allows users to build and/or edit flashcards quickly (1).
   - Formatting tools to allow customization of color, fonts, etc. (1, 3).
   - Multi-sided cards for multilingual students (1).
   - Export to PowerPoint presentation format (1, 4).
   - ~~Automatic print layouts, should physical flash cards be desired (1).~~

   Consider a French language student who wishes to create a new vocabulary deck for the current week's vocabulary. The user will simply access the Creator and begin typing pairs of words and their corresponding English definitions. Once the user has tweaked the formatting of the cards to his or her liking, the deck is ready for use in the Viewer.

2. **eFlash Viewer.** This subcomponent of eFlash handles the flash card decks that have been authored by the Creator, and provides a simple viewing interface as well as a self-test mode for users desiring to quiz themselves. It is compromised of the following functionality:

   - Simple pagination through a created deck (1, 3).
   - Quizzing mode that tracks self-imposed "correct" or "incorrect" answers when paginating through a deck (1, 2), with optional timing constraints.
   - Randomization of deck order, or heuristic-based ordering that tests a user on his weakest cards (2) after gathering information from previous quizzes.
   - Tracking of decks and quiz statistics through modularized user profiles (2, 4).
   - Integrated search based on card content (5).
   - Ability to cull cards from many sources into a new collection, facilitated through such search functionality (4, 5).

   For our French student who has just created a deck of the week's vocabulary, he or she may now view this deck in the Viewer. Making a quick pass through the list to verify all the terms have been listed, he or she now initiates a self-quiz and marks the answers he or she does not know. Our user can view these statistics in the user profile, or begin a learning quiz that will now repeatedly test him or her on the weakest words in the deck.

If later that week, the user needs to refer to the deck as a dictionary while writing up a homework exercise, he or she can bring up the deck and use the search functionality to find, either in English or French, the words he or she is looking for. During a quiz, however, he or she can always see the correct translation if desired, without having to stop and search through the deck.

3. **eFlash network connectivity.** This subcomponent of eFlash integrates with the Viewer to allow remote access to decks that have been registered with a centralized server, as well as allow the user to contribute to the network database with their own creations. It is compromised of the following functionality:

   - Access to a vast array of decks contributed by eFlash users (3, 4).
   - Search functionality across the network that replicates a local search (5).
   - Ability to download other users' decks into local user database and rank them(4).
   - Ability to upload own decks to the server (4).

   Our French student has studied the first week's worth of vocabulary, but is unwilling to wait for next week's words to be covered before learning more French. He or she can log in to the eFlash network and browse the French decks, perhaps downloading a new deck for various verb conjugations in literary past tense. Once done, he or she can see the new deck on his or her own computer, and begin learning anew through the Viewer.

   If the student does not feel comfortable uploading his or her week one vocabulary to the site for sharing, he or she is never forced to do so at any time while browsing the network.

References / Contacts / Terminology

References:

Requirements Specification

Contacts:

General team contact	Sean Gabriel
Architecture and vision	Kenneth Wong
	Tony Wu

Terminology:

- **Flashcard**
  - A set of 1 or more "sides"
  - Can have objects of type "text," "image," or "sound."
  - Each type of flashcard has its own layouts.
- **Deck**
  - A set of flashcards with a Category, Title, and Description.
  - Each and every flashcard belongs to at least one deck.
    - When flashcards are created, the user must specify which deck it belongs to
  - When accessing the eFlash network, users upload and download entire decks
    - ie. this is the basic unit of uploading/downloading

- When in local repository:
  - Remembers the local profile name of the creator or last editor
    - Null if is downloaded (not created locally)
    - Purpose: see point about uploading in "User Profile" below
- When in network repository:
  - Remembers network login name of uploader

- **User profile**
  - A profile for a user
  - Allows multiple users to use eFlash on a single machine
  - Each profile contains:
    - Optional network login name
    - ~~User's quiz statistics~~
  - Decks are **shared** between **all** user profiles
  - When uploading to the network, the user can only upload decks created/edited by the currently active profile

Limitations

This document describes the design of eFlash through functionality, not through implementation details. When getting down to specifics, some of the functionality may need to be altered to account for any unforeseen circumstances, and the reader is encouraged to view this specification as a living, breathing document that may change over the course of the development cycle.

With that said, one of the key invariants through this design process is our commitment to satisfying the needs of our customers. Functionality will not be compromised whenever a tangible customer need for it exists.

Feature Overview

System Metaphor

~~eFlash is a digital academic tutor that uses flashcards to aid in the user's memorization. It has four main features: flashcard creation, organization, quizzing, and retrieval. The creator is essentially a flashcard scribe. The user simply supplies it with a list of values (words, definitions, etc), and it creates decks of flashcards. The organizer works like a file cabinet for flashcards. The cards are grouped into decks, which are then further grouped into categories. The user can easily navigate to the desired deck by navigating through the category hierarchy. The quizzer is basically a simulator of physically reviewing the flashcards, minus the unnecessary overhead. However, it can also be thought of as a music player, but for flashcards. Each deck of flashcards is like a CD of songs, and each collection is like a play list. The quizzer is analogous to the player itself, which can play songs on a play list in some arbitrary order. It picks flashcards for the user at random, keeps track of time, user stats, etc. Finally, the retriever can be thought of as an online library of flashcards. The user can use simple search features to browse through huge libraries of flashcards.~~

eFlash is, in essence, a flashcard workstation.

A place where a user can go to work on creating new eFlash cards or simply a dedicated location to view and study a user's locally stored flashcards. In addition, just like a workstation, the user can effortlessly "look across his/her shoulder" to other eFlash users alike, and share their digitally mastered eFlash cards.

The aim of our product is to allow the user to simulate physical flashcards. In the creator, the user is given a white canvas that resembles a flashcard, and is allowed to put words and pictures on the canvas wherever he pleases. The created flashcards are then put into decks, just like real flashcards. In the viewer, the user can browse through decks of flashcards, flipping them over to view the other side whenever needed, just as he would if he were using physical flashcards.

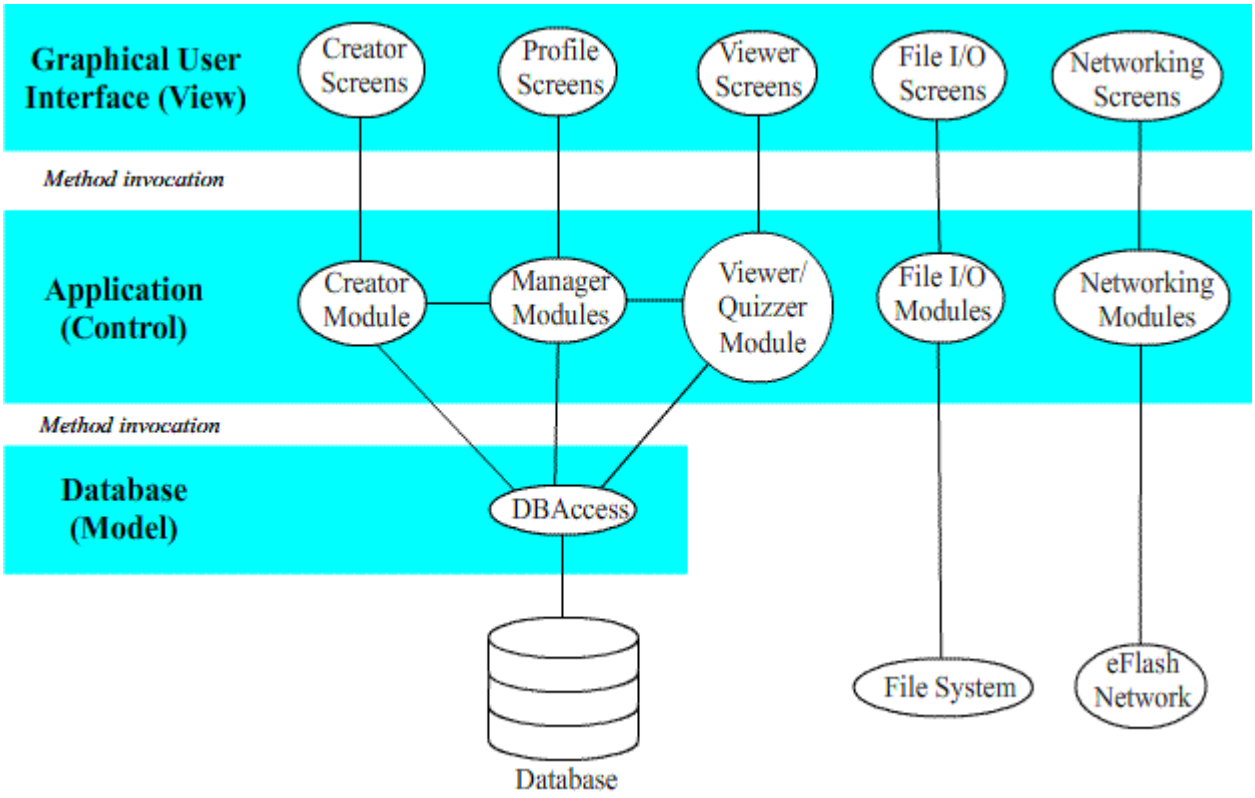
System Architecture *(deprecated)*

The system architecture is divided into three main layers: an event-driven Graphical User Interface (GUI) layer, an object-oriented Application layer, and a database access layer. The GUI layer interacts with the user through mouse and keyboard input. When GUI events are fired, the GUI layer makes one or more method calls to the Application layer, where the main application code resides. Finally, at the bottom is the Database layer, which provides methods to send/retrieve data to/from the database. The components of the top two layers can

be divided into four main groups: General, Creator, Viewer, and Networking.

Communication with the server will be piggybacked over http, and the functionality will be delivered transparently, as described. In the backend, we are considering implementing a thin XML-based communication scheme, and will flesh out the details in the upcoming implementation specifications.

This section gives an overview of how the modules are linked together. For further details about specific modules, please see the Features section below.



Below is a hierarchical list of the system architecture:

- **The Graphical User Interface layer**
  This layer is the primary means by which the user interacts with eFlash. It is comprised of screen objects that follow an event-driven architecture. It interacts with the Application layer via method calls to indirectly interact with the database.

  The GUI layer includes the following screens:

  - **General**
    - Profile Selector Popup
      - Description: Allows the user to select his/her user profile and/or create new ones.
      - GUI connections: Main Menu, New Profile Popup
      - Application connections: Profile Manager
      - Features implemented: User profiles
    - Main Menu
      - Description: This screen serves as a portal to the other GUI screens. It displays a tree hierarchy of decks in the local database, organized by categories. The user may select any deck to modify, export, view, or quiz. Additionally, he/she may create/import new decks.

- GUI connections: Profile Selector Popup, Layout Manager Screen, Viewer Screen, Network Welcome Screen, Import Screen, Export Screen
- Application connections: None
- Features implemented: None

- **Creation**
  - Layout Editor Screen
    - Description: This is the primary screen the user will use to create flashcards. It shows a white rectangular canvas, with a toolbar on the left to allow the user to add objects. These objects can then be repositioned and resized at will. An array of buttons at the top allows for arbitrary formatting of text.
    - GUI connections: Main Menu
    - Application connections: Deck Manager
    - Features implemented: Flashcard creating
  - Import Screen
    - Description: This is actually a series of screens that takes the user through the steps to read in a text file and convert it to a deck. The user can choose from a set of options to specify the exact formatting of the file.
    - GUI connections: Main Menu
    - Application connections: File Importer, Deck Manager, Flashcard Manager
    - Features implemented: File importing
- **Viewer**
  - Viewer Setup Screen
    - GUI connections: Quiz Setup Screen, Exporter Screen, Main menu
    - Application connections: Deck Manager
    - Features implemented: Flashcard viewing
  - Quiz Setup Screen
    - GUI connections: Quizzer Screen
    - Application connections: None
    - Features implemented: Multiple Choice, self Quiz, fill in the blank

- **Networking**
  - Network Welcome Screen
    - Description: This screen serves as the portal to the other three networking screens.
    - GUI connections Main Menu, Network Downloader, Network Uploader, Network Ranker
    - Application connections: None
    - Features implemented: None
  - Network Downloader
    - Description: This screen allows the user to browse the network decks and select which ones to download. For each deck, the user may download the preview image to preview the first side of the first side of the deck.
    - GUI connections Welcome Screen
    - Application connections: Networking
    - Features implemented: Networking
  - Network Uploader
    - Description: This screen allows the user to upload his/her decks to the online database
    - GUI connections Welcome Screen
    - Application connections: Networking
    - Features implemented: Networking
  - Network Ranker
    - Description: This screen allows the user to view rankings for each deck in the online database and cast his/her won vote.
    - GUI connections: Welcome Screen
    - Application connections: Networking
    - Features implemented: Networking

- **The Application layer**

This layer contains the main application code of eFlash. The modules in this layer consists of four manager modules, which deal with profiles, decks, flashcards, and collections, two modules that deal with file I/O, one to supply the logic to run quizzes, and one to handle the application's networking needs.

The Application layer includes the following modules:

- **General**
  - Profile Manager
    - This module provides all functions dealing with user profiles, including

profile management (creation, deletion, etc.), login authentication, and loading profile information.

- Receives requests from: Profile Selector Popup, New Profile Popup
- Sends requests to: DBAccess

- Deck Manager
  - This module handles all requests regarding flashcard decks. Its functions include creating and deleting decks in the database, retrieving a list of decks given a category ID, fetching a list of flashcards belonging to a given deck, etc.
  - Receives requests from: Deck Manager Screen, Deck Creation/Edit Screen
  - Sends requests to: DBAccess
- Flashcard Manager
  - This module provides all functions dealing with flashcards in the database, including searching for flashcards given a search string, creating/editing/deleting flashcards, fetching the contents of a flashcard given its ID number, etc.
  - Receives requests from: Flashcard Input Gatherer Screen, Quizzer Screen
  - Sends requests to: DBAccess
- Collection Manager
  - This module handles all requests relating to collections, including fetching a list of flashcards belonging to a specific collection, adding cards to a collection, creating/deleting collections, etc.
  - Receives requests from: Viewer Collection Setup Screen
  - Sends requests to: DBAccess

Creator

- File Importer
  - This module contains functions to read in data stored in text files and to convert it into the format used by eFlash. It does not actually store the data to the database. Rather, it simply returns a data structure containing the data, and it is the job of the caller to save it to the database, if desired.
  - Receives requests from: Import Screen

Viewer

- Quizzer
  - The main purpose of this module is to provide various functions needed when conducting quizzes, such as the logic to picking the next flashcard to display, loading/updating current user statistics, etc.
  - Receives requests from: Quizzer Screen
  - Sends requests to: Profile Manager, Collection Manager
- File Exporter
  - This module has functions to output flashcard and deck information stored in

the local database to various file formats. Currently, the primary file formats to export to are Powerpoint Presentations (.ppt), for user who wish to view flashcards independently from our application, and an eFlash file format, which can then be imported by other users.

- Receives requests from: Exporter Screen

- **Networking**
  - Networking
    - This module handles all of eFlash's networking needs. This includes server login creation and authentication, retrieving lists of decks and flashcards in the server repository, uploading and downloading decks to and from the online repository, etc.
    - Receives requests from: Network Welcome Screen, Network Browser, Ranker Popup

- **The Database layer**

  This layer contains a single module that provides generic database access functions. The goal of this layer is to abstract away the details of the database from the application code, so that the application code does not have to deal with formulating database-specific queries.

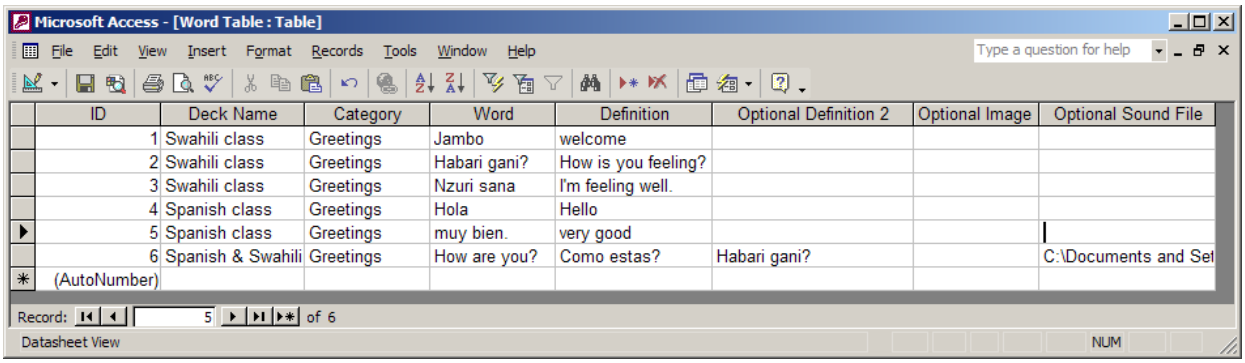
  The Database layer includes the following module:

  - DBAccess
    - This module provides all functions dealing with user profiles, including profile management (creation, deletion, etc.), login authentication, and loading profile information.
    - Receives requests from: Profile Manager, Deck Manager, Flashcard Manager, Collection Manager

Database Tables *(deprecated)*

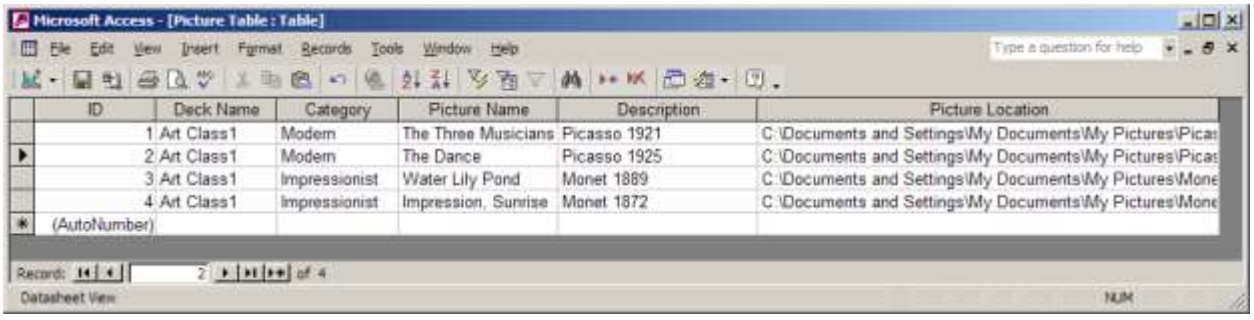
In its raw form, all the data categorizing each Deck and FlashCard will be stored in datatables located in the *eFlash* database that is created during the software installation phase. When a FlashCard object is created, FlashCard Creator will access the appropriate data in the database and consequently create the FlashCard. The *eFlash* database will hold 3 DataTables, one for holding all the word flashcards (**Word Table)**, one for holding all the picture flashcards (**Picture Table),** and one for holding all the sound flashcards (**Sound Table)**.

Data Table for Holding Words (Word Table):



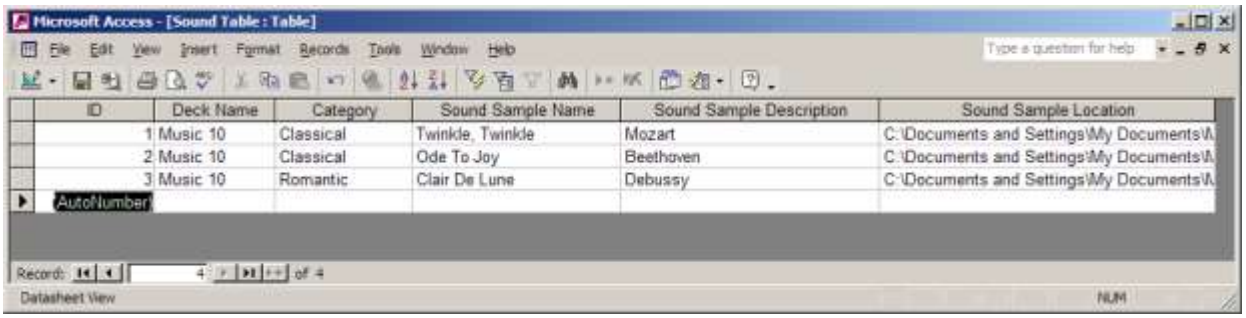
Description: This table holds all the words that are manually inputted by the user for each project. The fields that will describe each word are **ID**, **Deck Name**, **Category**, **Word**, and **Definition**. These fields are required; however the other fields are not. The user has the option of inputting another translation of the word, a picture, or a sound file. All of these fields except for ID will be specified by the user. The field **ID** will be the unique identifier for each Word FlashCard object and the **Word** field will be the name of each word flash card.

DataTable for Holding Pictures (Picture Table):



Description: This table as well holds all the pictures that are manually inputted by the user for each project. The fields that will describe each picture are **ID, Deck Name, Category, Picture Name, Description,** and the location of the picture in the user's computer, **Picture Location**. As in the case above, the **ID** field will be the unique identifier for each Painting FlashCard object, and the **Picture Name** field will be the name for each picture flash card.

DataTable for Holding Sound Samples (Sound Table):



Description: This table also holds entries manually entered by the user. This datatable could also be created automatically by grabbing the music tags of the music files and entering the locations and information for each record. The fields in this DataTable are **ID, Deck Name, Category, Sound Sample Name, Sound Sample Description,** and **Sound Sample Location**. The **ID** will serve to uniquely identify each music flashcard and the **Sound Sample Name** will suffice as the name of each music flashcard.

Invariants

1. Flashcards
   a. Every sound object must have a valid sound file.
   b. Every image object must have a valid image file.
   c. All image/sound file locations should be valid locations.
   d. Each flashcard must belong to at least one deck.
2. Networking
   a. The user must not be allowed to access the network unless proper authentication is provided.
   b. The user may only upload flashcard decks that he/she created/modified.
   c. Uploaded decks must be identical to the original.
   d. Downloaded decks must be identical to the original.

Features

Login Screen

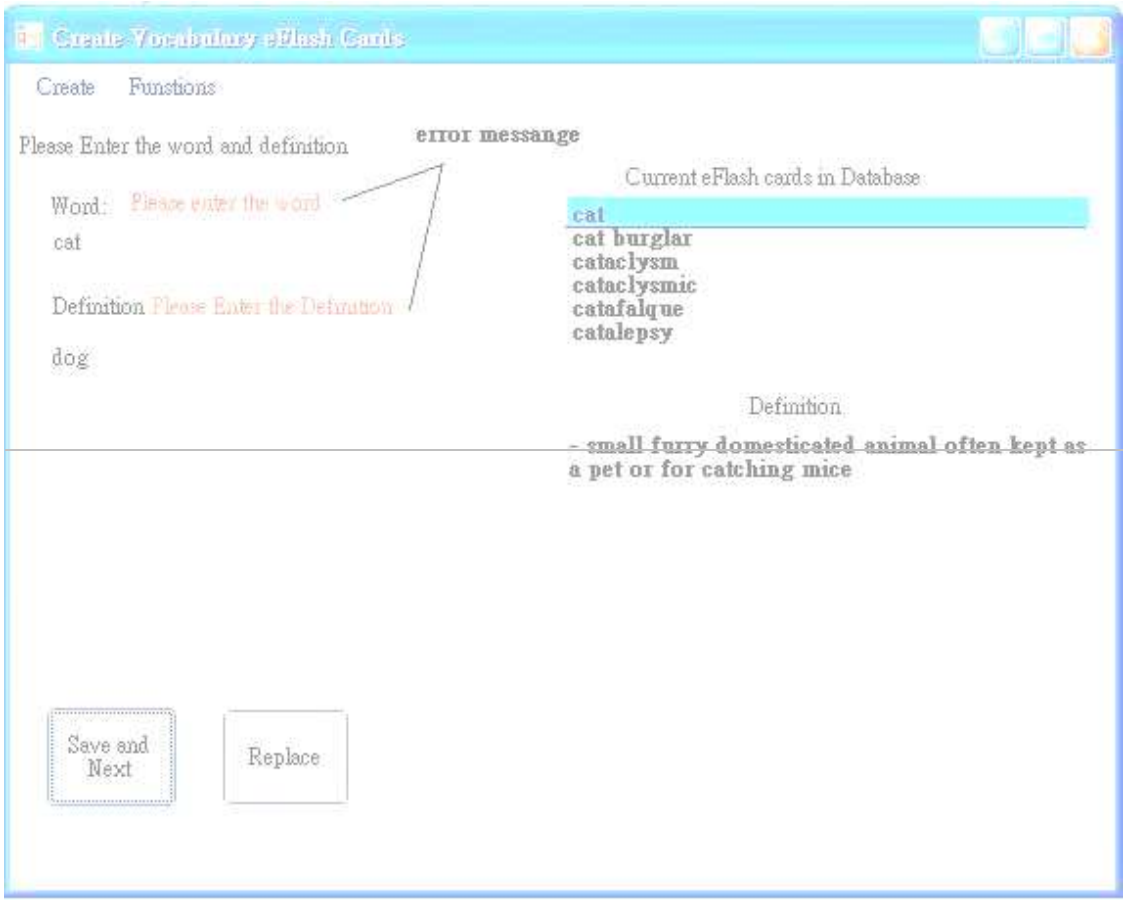
- **User profiles**
  - Choose from a list of existing profiles, or may create their own.
  - Password field is optional.
  - If password exists, it can be saved, and is then automatically filled in each time the profile is selected (if the user wishes).
  - Has a checkbox to let user specify "auto-login," which automatically logs in to the currently selected profile each time eFlash runs, bypassing the login popup.
    - Only one profile may be designated "auto-login."
  - Relevant architecture elements: Profile Selector Popup, New Profile Popup
- **Deck management**

- Displays a list of all decks in the user's local repository in a expandable hierarchal manner, sorted by Category, then Title
  - For instance:
    - - Biology
      - Week 1
      - Week 2
    - + History
    - - Swahili
      - Nouns
      - Verbs
- Selecting a deck's title displays its description
- Has dedicated buttons for creating, editing, and deleting decks

Relevant architecture elements: Deck Manager Screen, Deck Creation/Edit Screen

- **Flashcard creating**
  - A list displays all of the currently existing flashcards of the selected type in the current deck.
    - After the user type every character in the Word/Link field, the system will do a real time search the database and refresh the list to display the best match as the top-most word and selects it.
    - Clicking on entries in this list fills the textboxes with the contents of the selected entry.
  - Additional options:
    - Save and Next
      - Saves the current flashcard and clears the objects.

- ~~**Flashcard types** *(deprecated)*~~



Vocabulary flashcard creation screen

- Vocabulary:
  - User can input "word" and "definition".
  - A textbox displays the definition of the currently selected word in the currently existing list.
  - Error 1: Empty Word
    - Give warning: Please Enter the Word on the screen.
  - Error 2: Empty Definition
    - Give warning: Please Enter the definition on the screen.
  - Error 3: The Word already exists
    - The "Save and Next" button is disabled, and "Replace" is enabled.

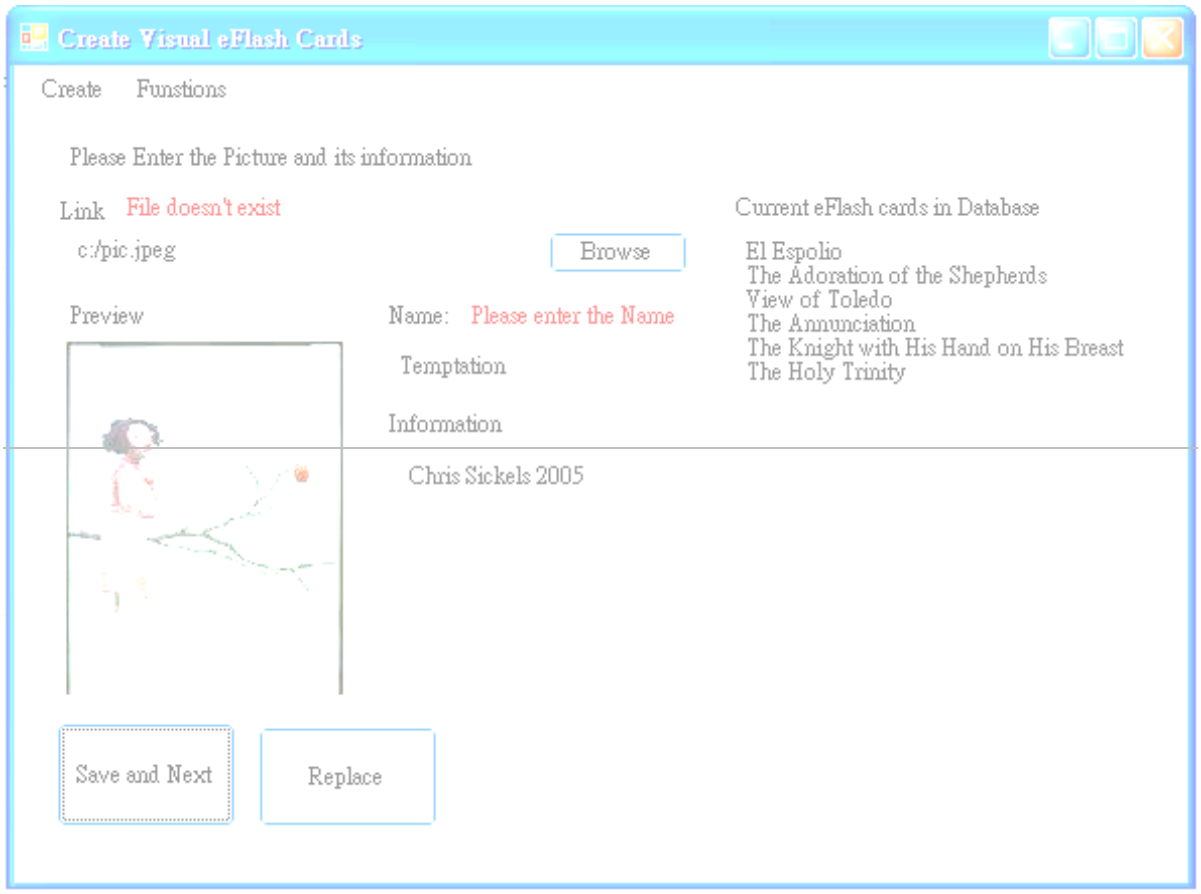


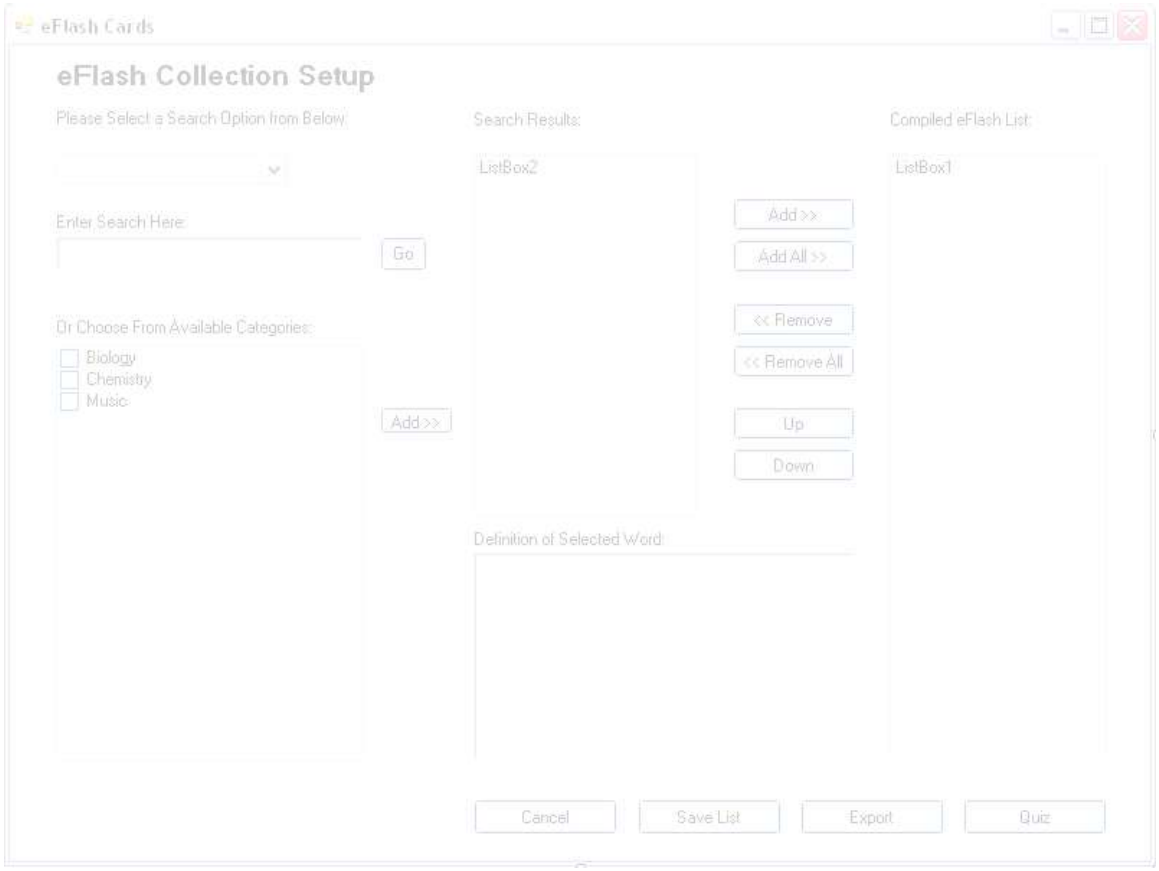
Image Flashcard creation

- Visual:
  - User can either directly input image path or browse for it in the local file system.
  - User can input optional Name and Information fields.
  - Displays a preview of the image specified by Link.
  - Error 1: Empty Link
    - Give warning: Please Enter the link/ File doesn't exist on the screen.
  - Error 2: There already exists a flashcard with the given image.
    - "Save and Next" disabled, "Replace" enabled.

Music flashcard creation

- Music:
  - User can either directly input sound file path or browse for it in the local file system.
  - User can input optional Name and Information fields.
  - Displays a slider bar and sound control buttons to let user preview the audio file specified by Link at any position.
    - Play
      - Play the sound from the current position.
    - Pause
      - Stop playing the sound.
    - Stop
      - Stop playing the sound and set current position to beginning of file.
  - User can use the slider bar to indicate a segment of the audio file to use for the flashcard (instead of using the entire length of the file).
  - Error 1: Empty Link
    - Give warning: Please Enter the link/ File doesn't exist on the screen.
  - Error 2: There already exists a flashcard with the given audio file.
    - "Save and Next" disabled, "Replace" enabled.

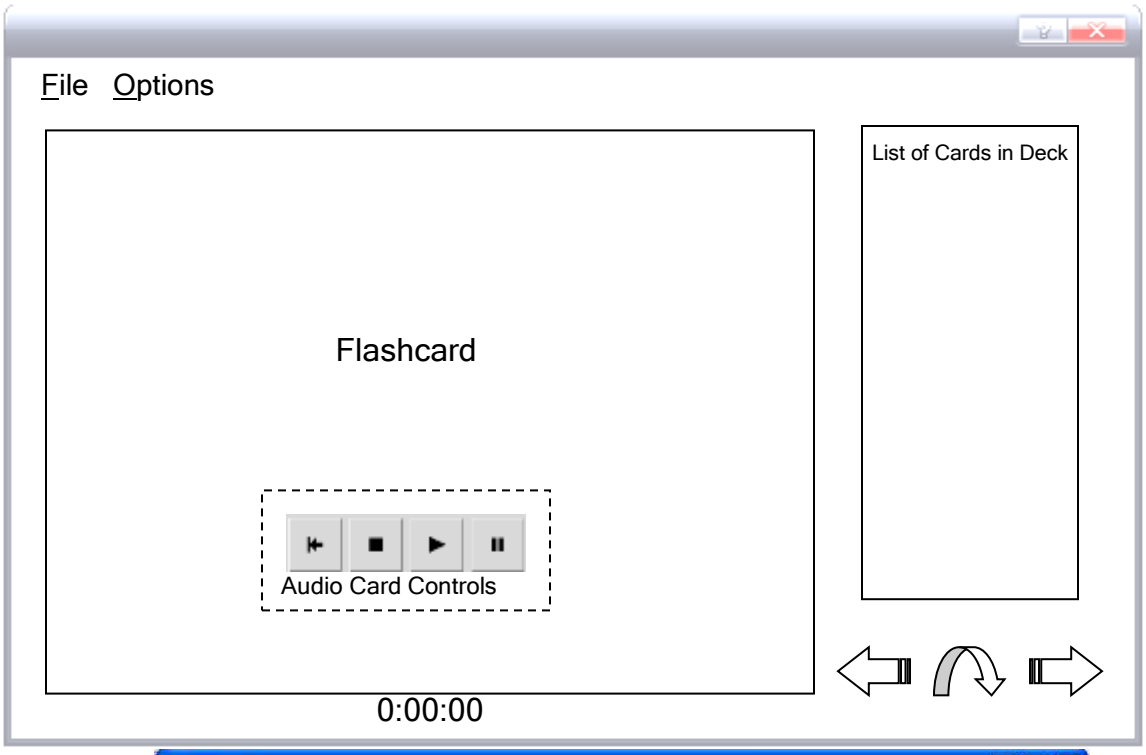
Relevant architecture elements: Flashcard Input Gatherer Screen

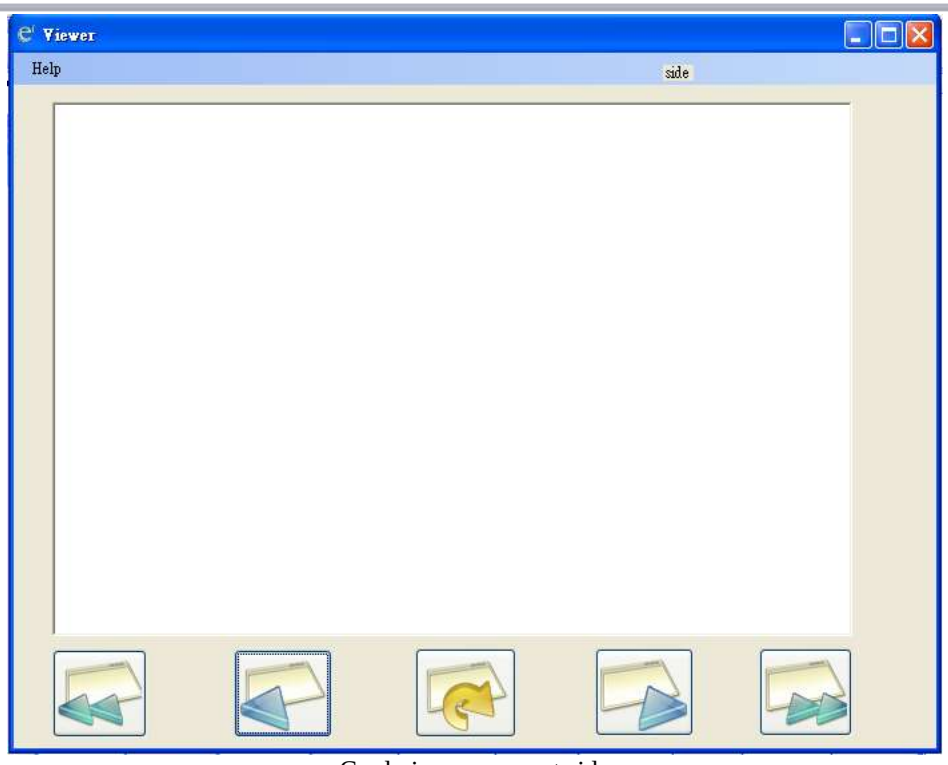


~~Collection Manager Screen~~

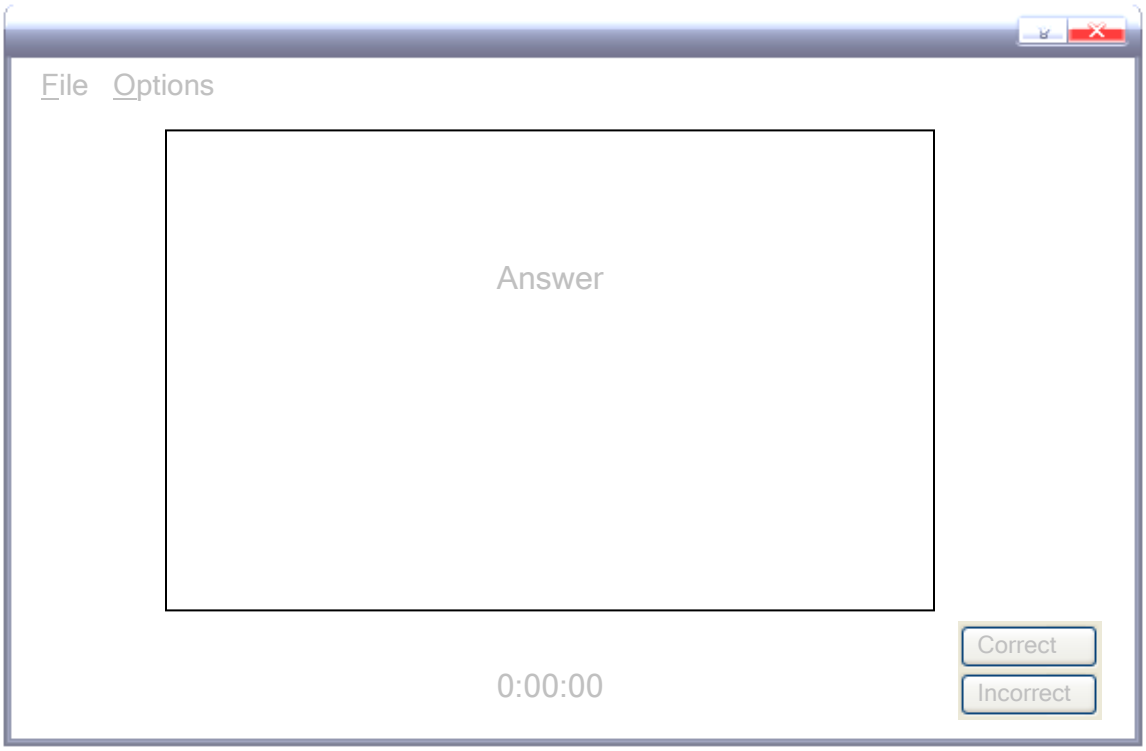
- ~~**Collection management**~~ *(replaced by **Main Menu**)*
  - ~~Display:~~
    - ~~List of existing collections.~~
    - ~~Hierarchical list of categories and decks.~~
    - ~~Hierarchical list of search result decks and their flashcards.~~
    - ~~Definition/image/sound of the selected flashcard.~~
    - ~~List of flashcards in the current collection.~~
  - ~~Deck search~~
    - ~~Searches all decks and displays the results in the appropriate list.~~
  - ~~Options:~~
    - ~~Run search.~~
    - ~~Select flashcard to display its contents in the preview box.~~
    - ~~Add selected flashcard to collection.~~
    - ~~Remove selected flashcard from collection.~~
    - ~~Save the current collection.~~
    - ~~Begin quiz on the current collection.~~

Relevant architecture elements: Viewer Collection Setup Screen

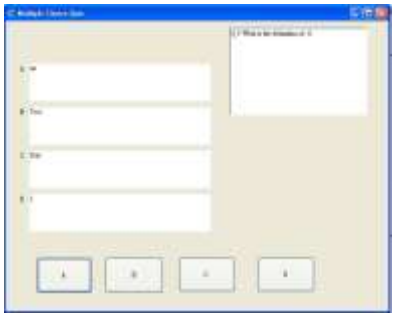


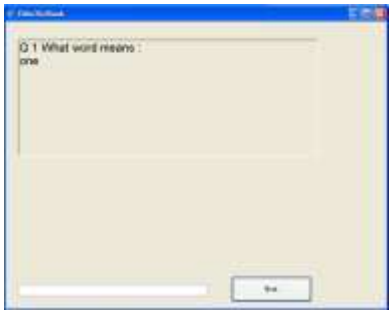


Card viewer, prompt side



Card viewer, answer side





- **Flashcard viewing**
  - Displays each side of flashcard.
  - In the case of audio objects, display sound control buttons Play and Stop.
  - ~~Show timer to keep track of time elapsed.~~
  - ~~Show list of flashcards in current collection.~~
    - ~~Clicking on a flashcard in the list jumps to that flashcard.~~
  - Options (each option has a corresponding keyboard key):
    - Previous flashcard.
    - Last flashcard in the Deck
    - First flashcard in the Deck
    - Skip to next flashcard (without flipping).
    - When viewing front side: flip to next side of flashcard

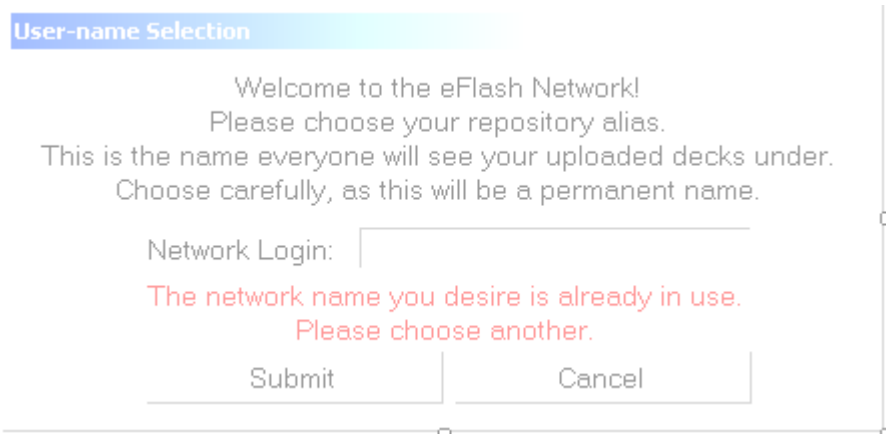
- When viewing back side: indicate whether the user got it correct or not.
- Exit quiz.

Relevant architecture elements: Quizzer Screen, Quiz Setup Screen

- **File Importing**
  - User can select file by directly typing in the file path or by browsing for it in the local file system.
  - May import into a new empty deck.
  - Supported file types include:
    - Text files formatted according to eFlash standards.
    - ~~eFlash deck file format.~~

  Relevant architecture elements: Import Screen
- **File Exporting**
  - User can export a deck to a file.
  - Supported file types include:
    - Powerpoint Presentation.
    - ~~eFlash deck file format.~~

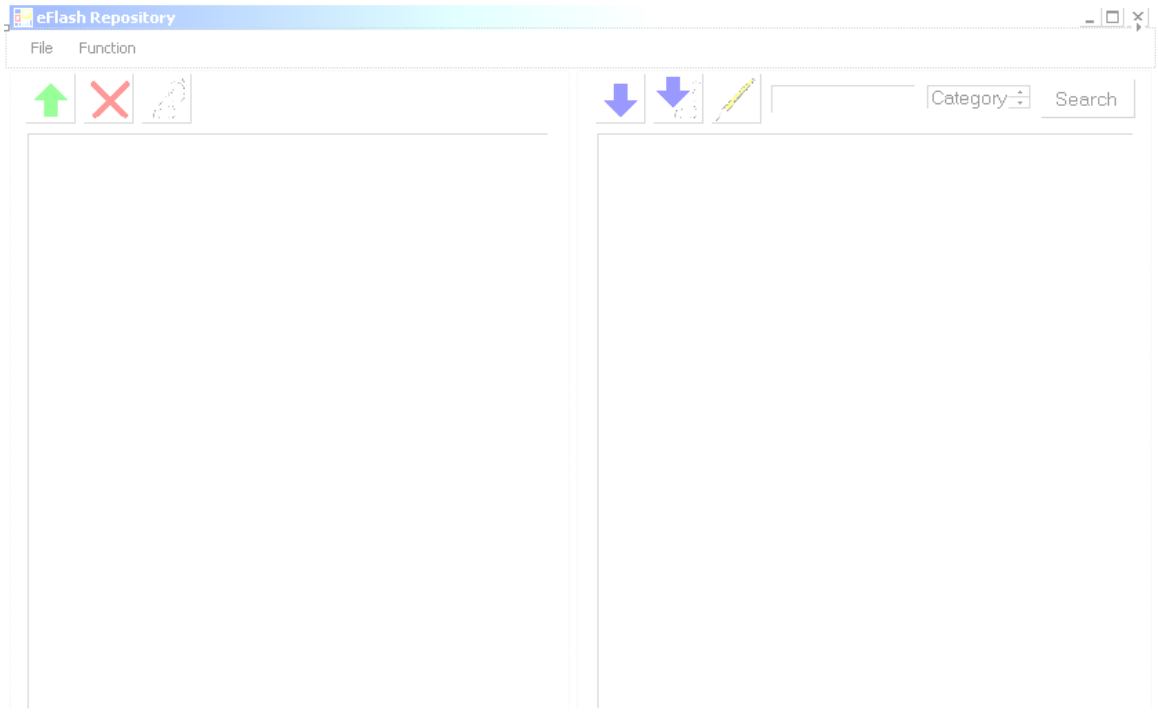
  Relevant architecture elements: Exporter Screen
- **Networking**



Network Login Screen

(This screen has been revamped)

- Displays welcome to eFlash networking options
- Only shown if the current user profile does not have an associated network login name.
- ~~Prompts user for a network login name.~~
- ~~Submit~~
  - ~~Contact server database. Make sure requested login name is unique.~~
    - ~~If unique: proceed to network browser.~~
    - ~~If not unique: display error message and prompt user again.~~
- Exit
  - Return to Main Menu Screen



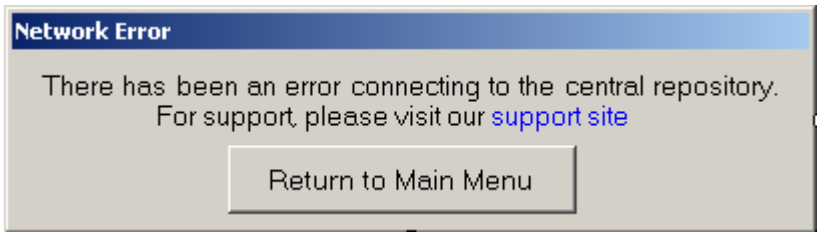
Network Deck Management screen
(This screen has been revamped)

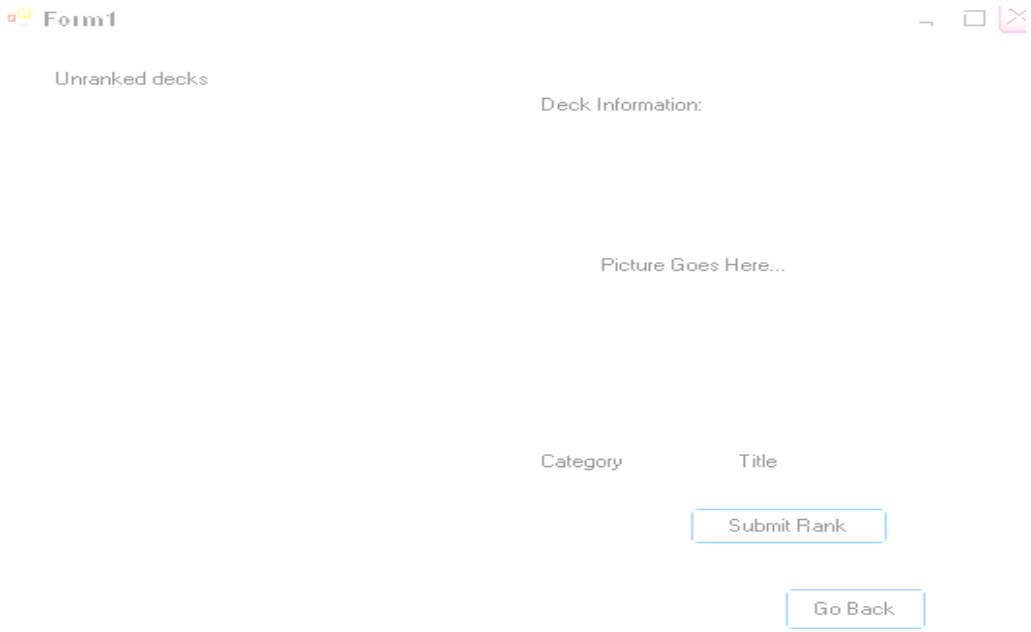
- eFlash network browser:
  - Displays two file-hierarchy panels
  - ~~Left panel shows local flashcards and upload-related options.~~
    - Upload
      - Upload selected file if new file.
      - Upload selected file (overwrites) if previously uploaded.
      - Error: Cannot connect to server.
    - Delete
      - Remove uploaded file from server.
      - Error: Cannot connect to server.
    - View ratings
      - Expand description window below file name in hierarchy listing – shows number of downloads, number of raters, and average rating
      - Error: Cannot connect to server.
  - ~~Right panel shows server flashcards and download-related options.~~
    - Download
      - Downloads selected file, updates view (appears in Downloaded and not yet rated category).
      - Error: Cannot connect to server.
    - Download and view
      - Downloads selected file, transitions to deck viewer.
      - Error: Cannot connect to server.
    - Search

- Searches server for keyword, displays all results in categorical hierarchy in browse hierarchy.
- Error: Cannot connect to server.

- Return to browsing
  - Reverts right panel back to browse deck hierarchy.
  - Error: Cannot connect to server.

- Shows error message popup if encounters connection problems.





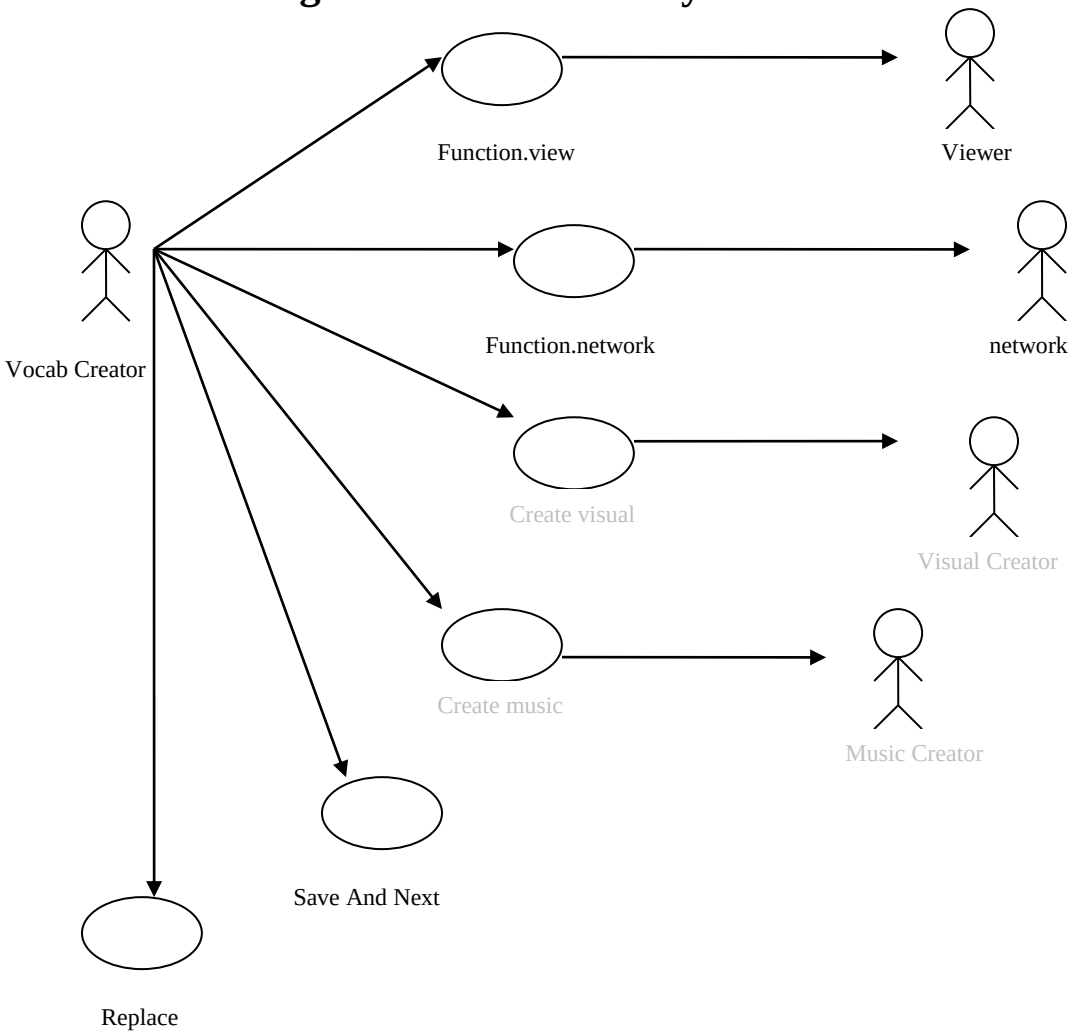
(This screen has been revamped)

- **eFlash Network Ranker**
  - Divided up into a left and right section.
  - Left panel shows local decks that were downloaded and unranked.
    - Preview
      - Download preview and display information about deck.
      - Error: Cannot connect to server.
    - Submit Rank
      - Sends insertion rank query to remote database
        Calculates network user's new average ranking

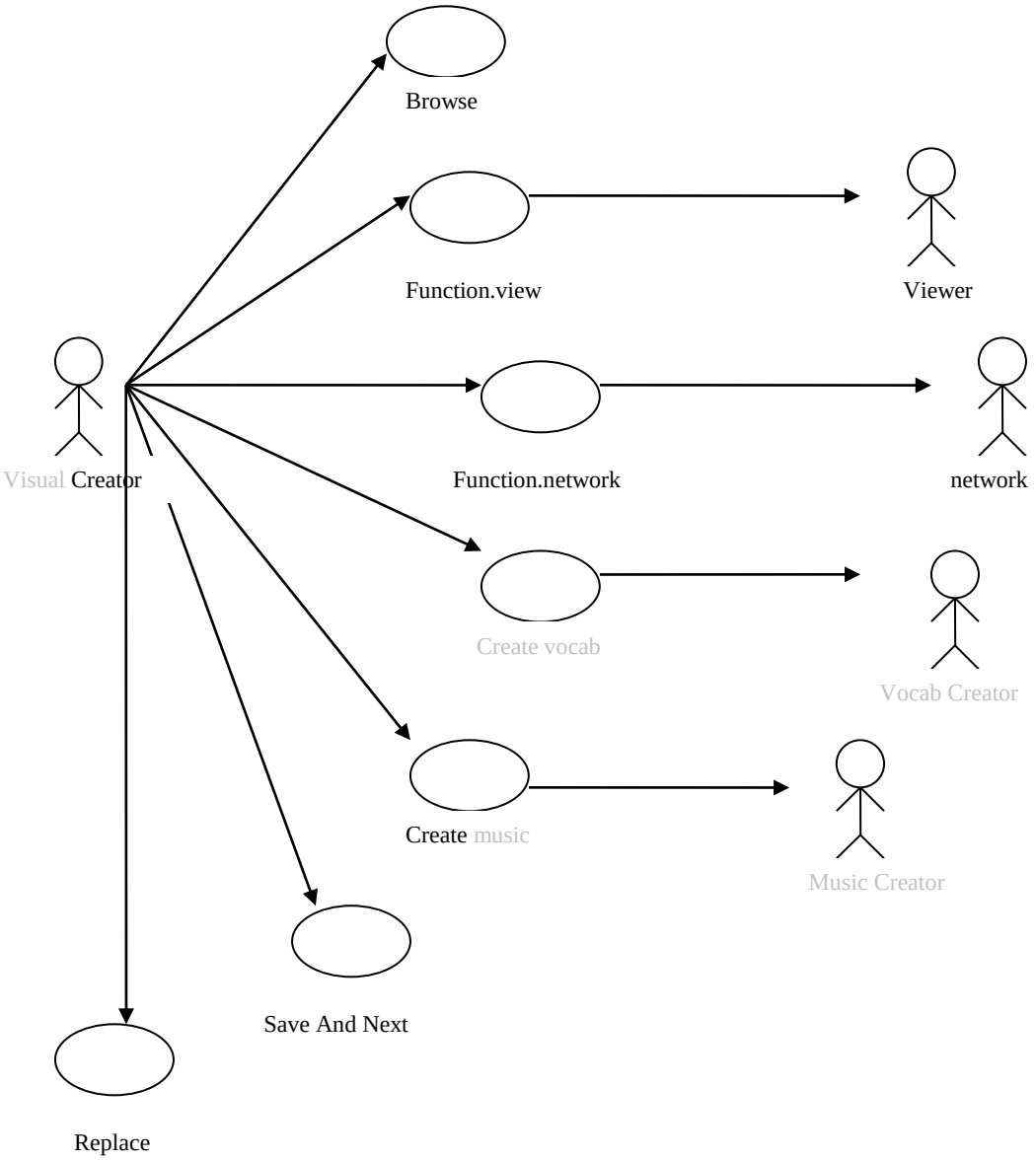
Relevant architecture elements: Welcome Screen, Network Uploader/Downloader, Network Ranker

Use Cases

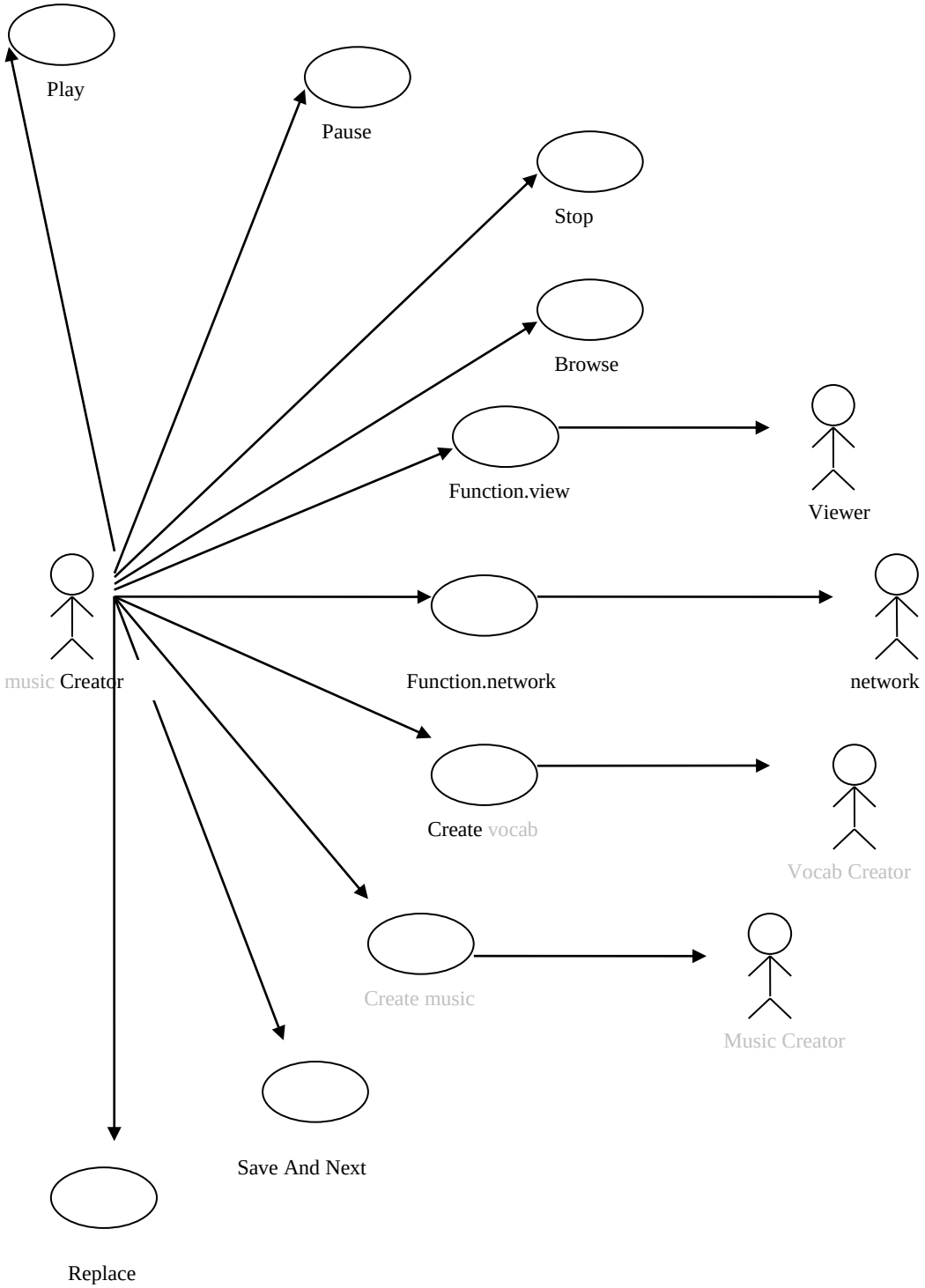
Use case Diagrams for Vocabulary Flashcard Creator



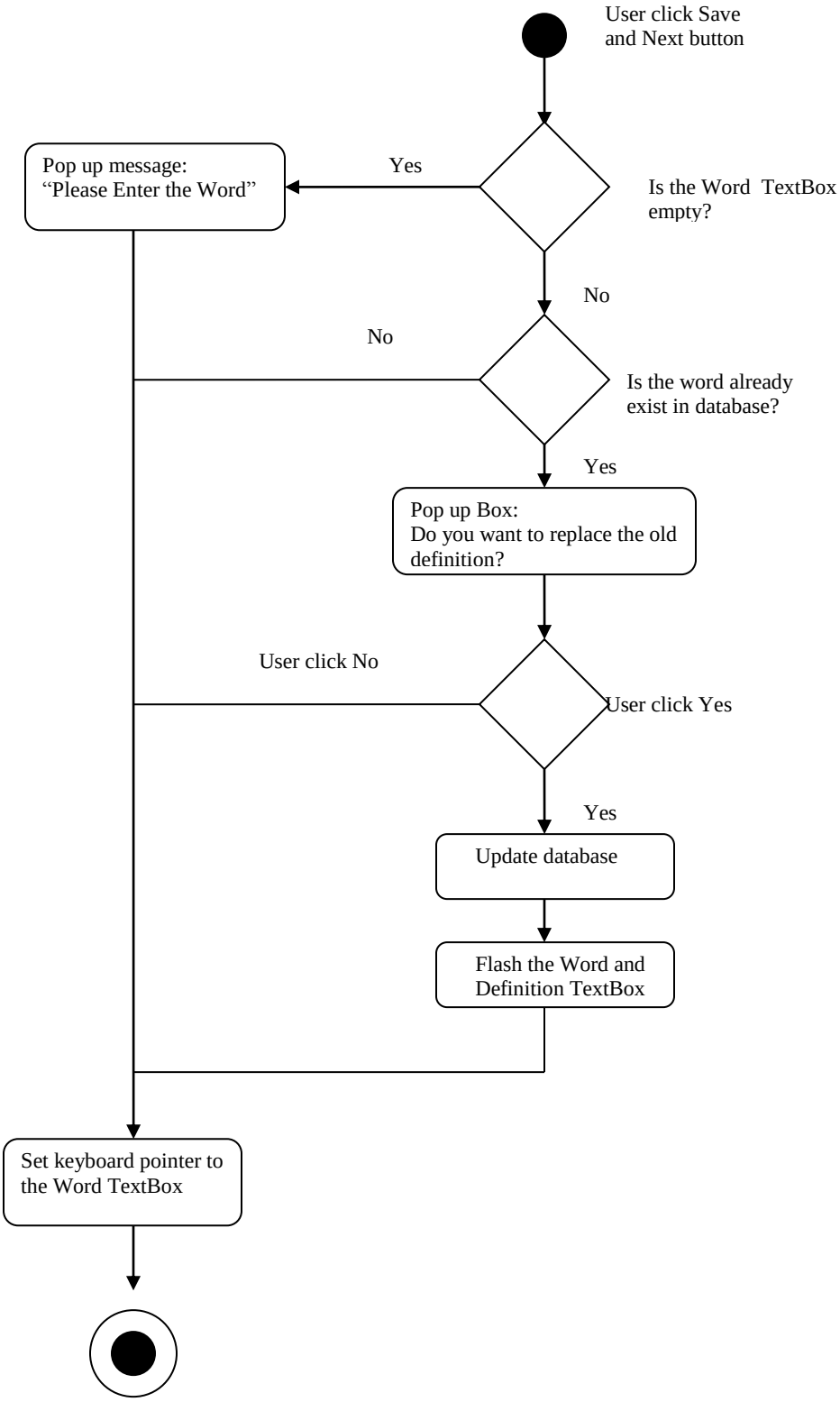
Use case Diagrams for Visual Flashcard Creator



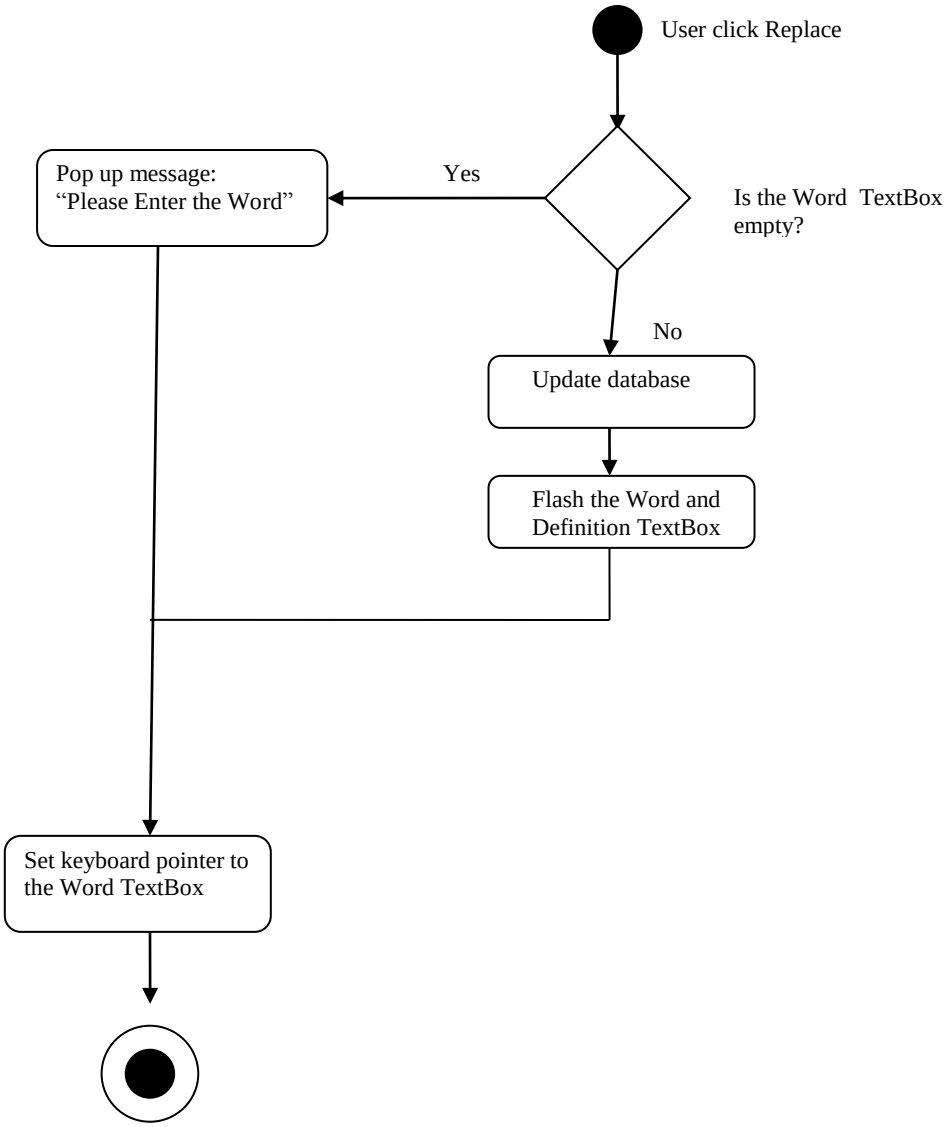
Use case Diagrams for Visual Flashcard Creator



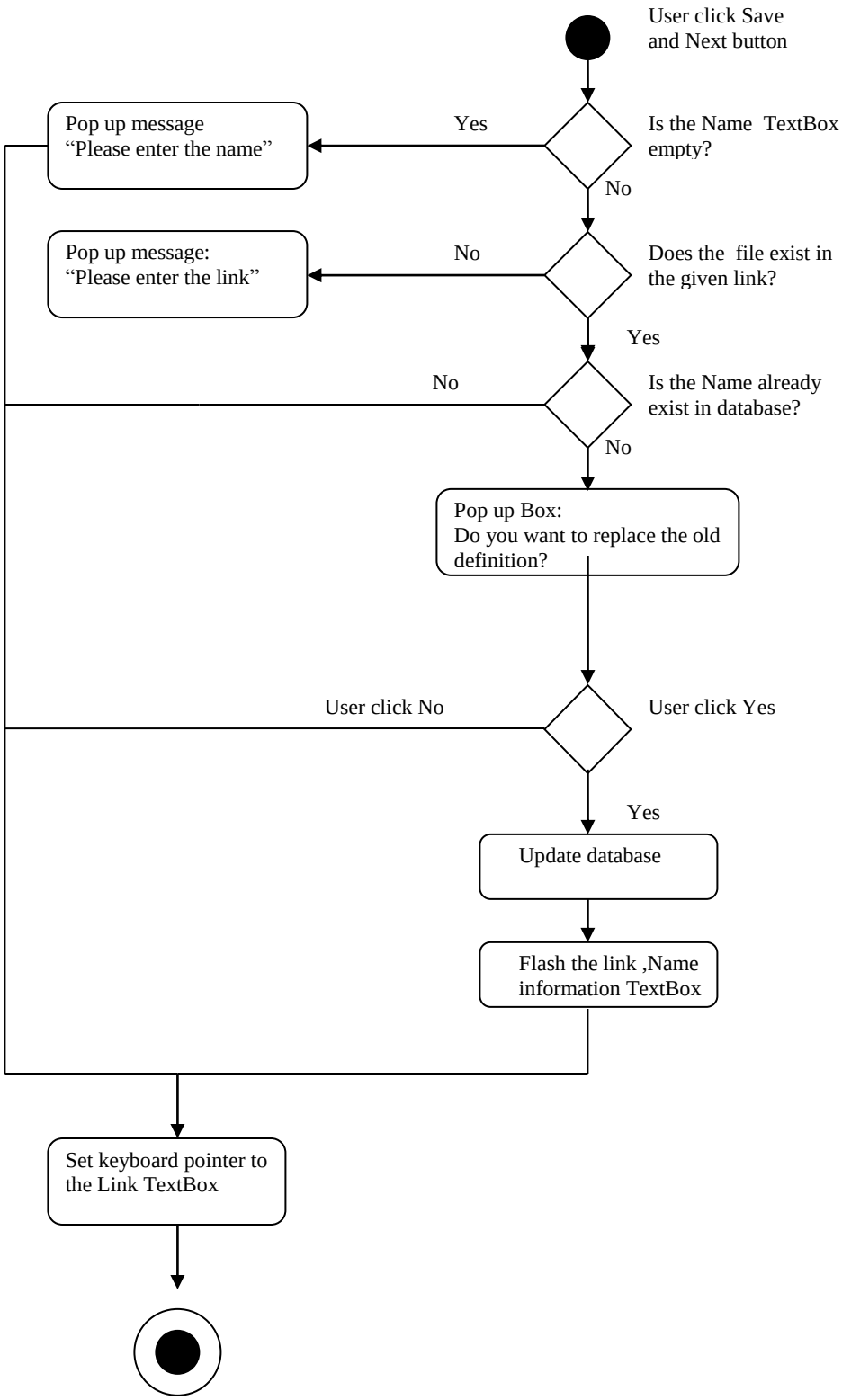
Activity Diagram :Save and Next (Vocabulary Flashcard Creator)



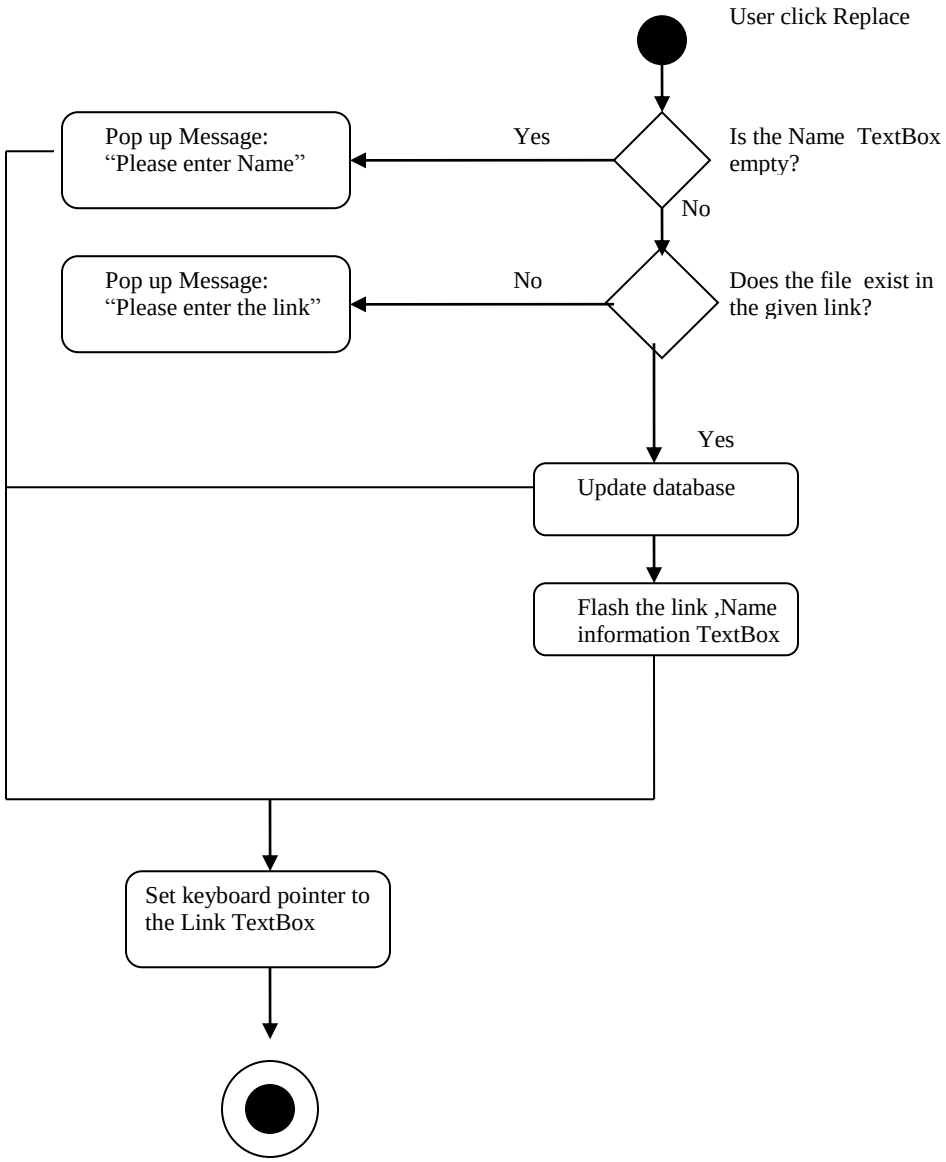
Activity Diagram :replace(vocabulary flashcard creator)



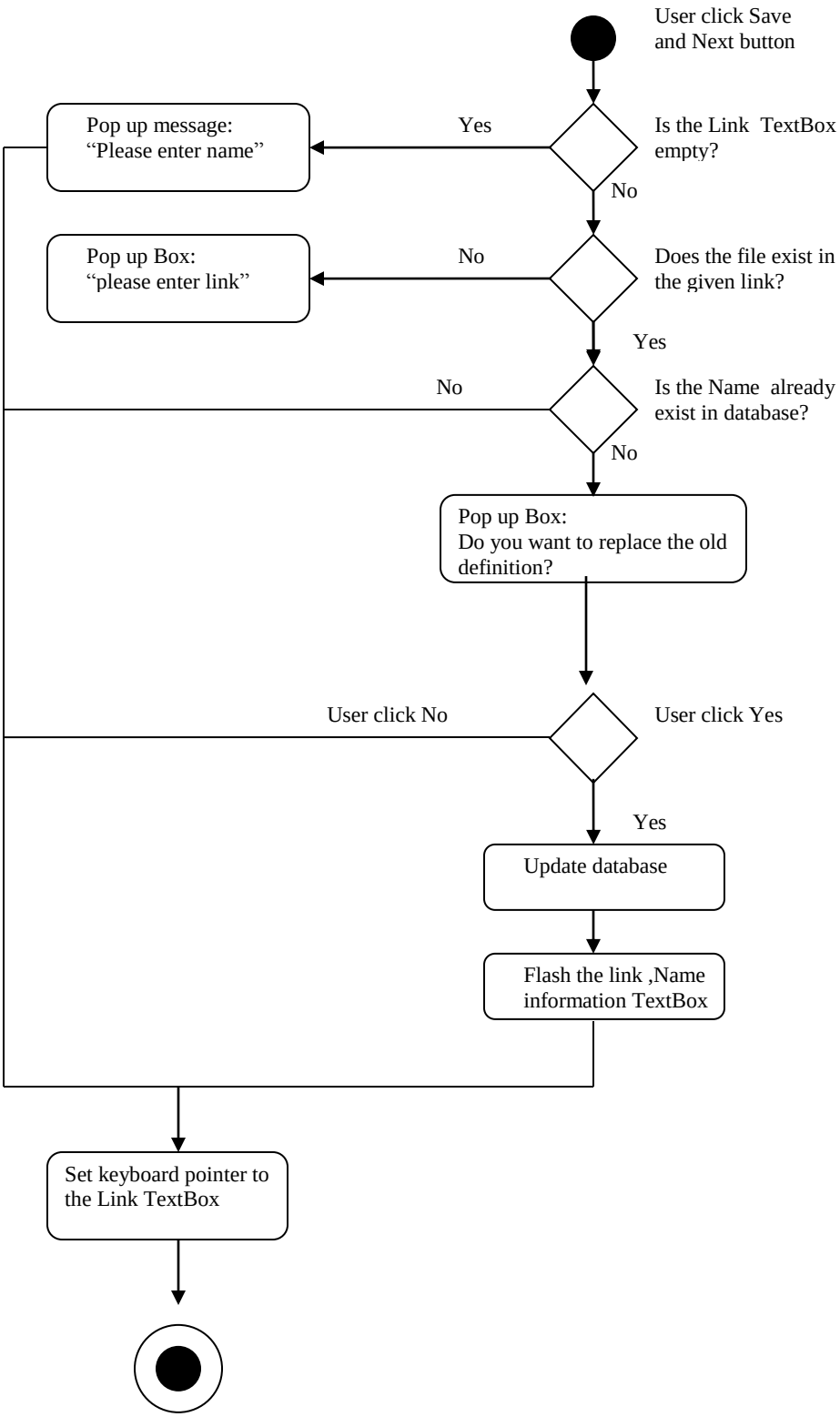
Activity Diagram :Save and Next (music creator)



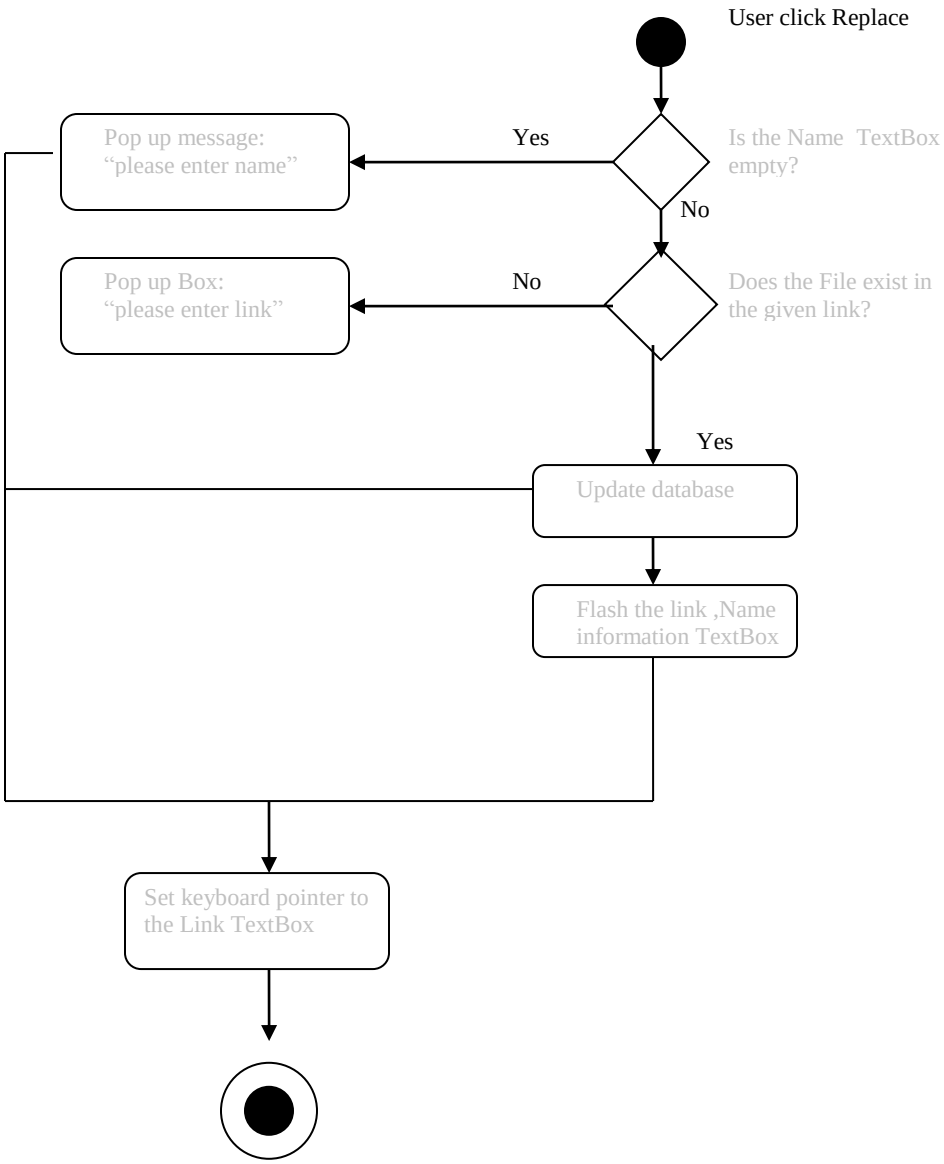
Activity Diagram :Replace (music creator)



Activity Diagram :Save and Next (visual flashcard creator)

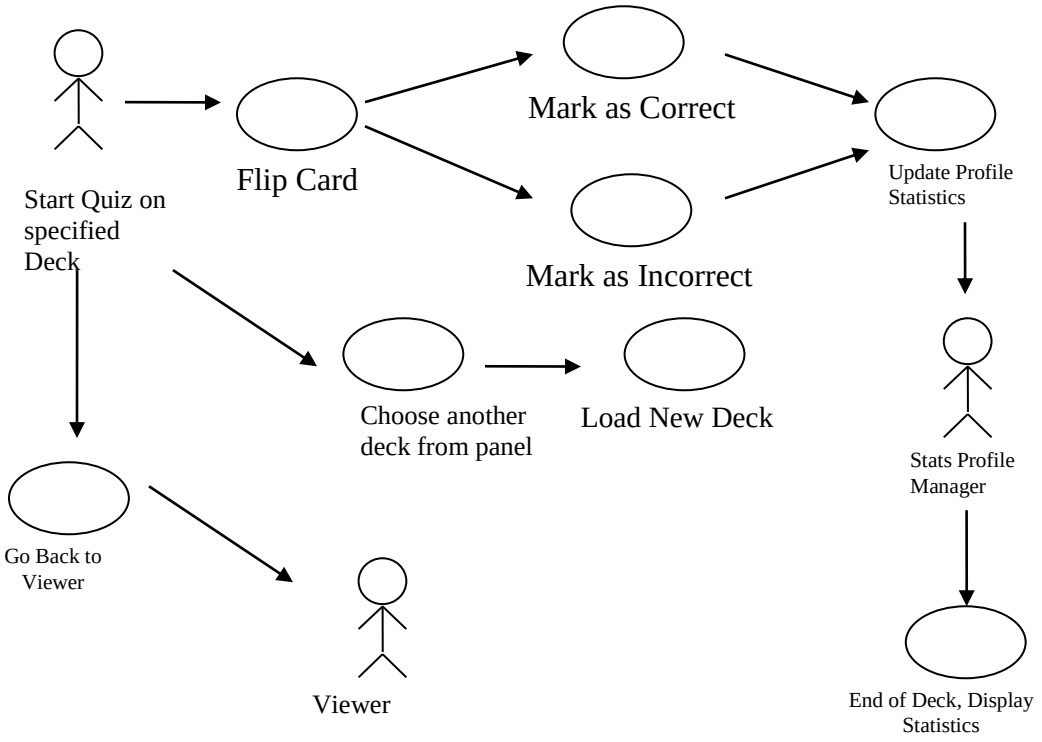


Activity Diagram :Replace (visual flashcard creator)

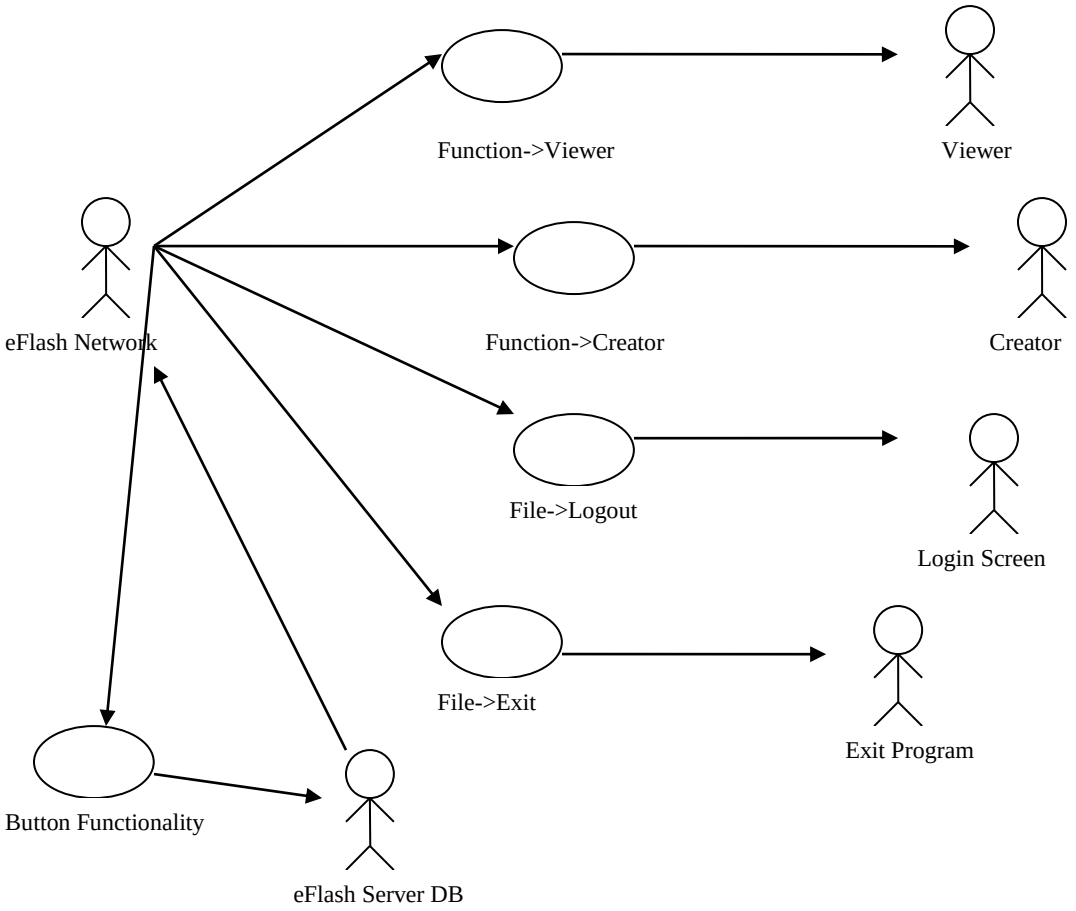


Error!

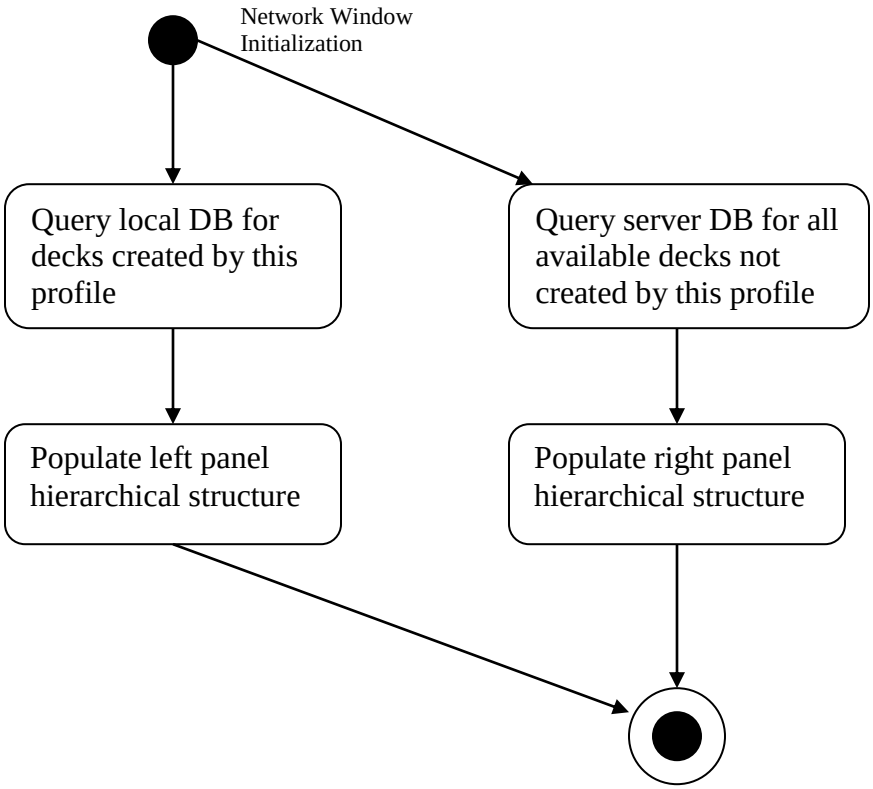
Use case Diagrams for Quizzer



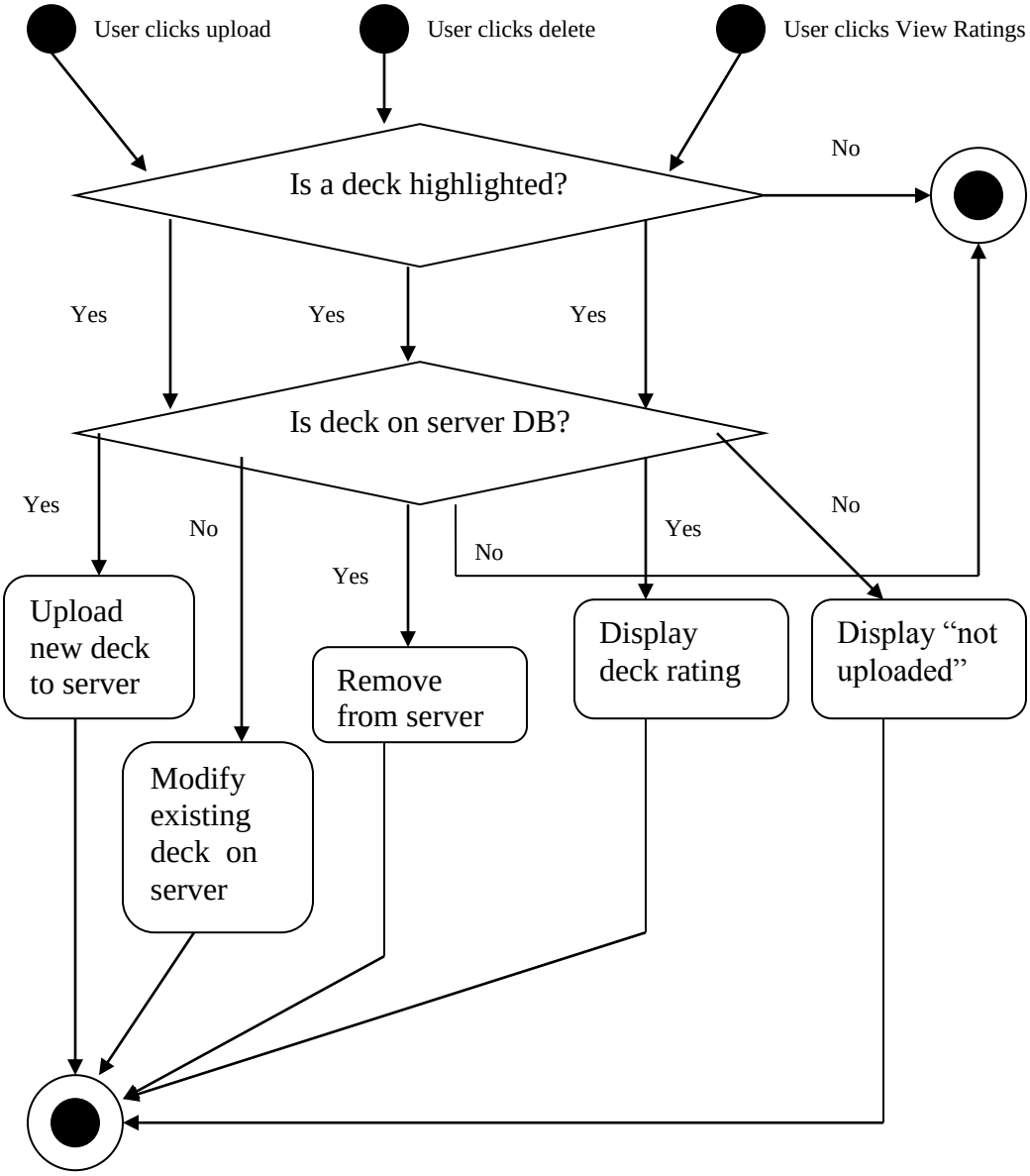
Use case Diagrams for eFlash Network Menus



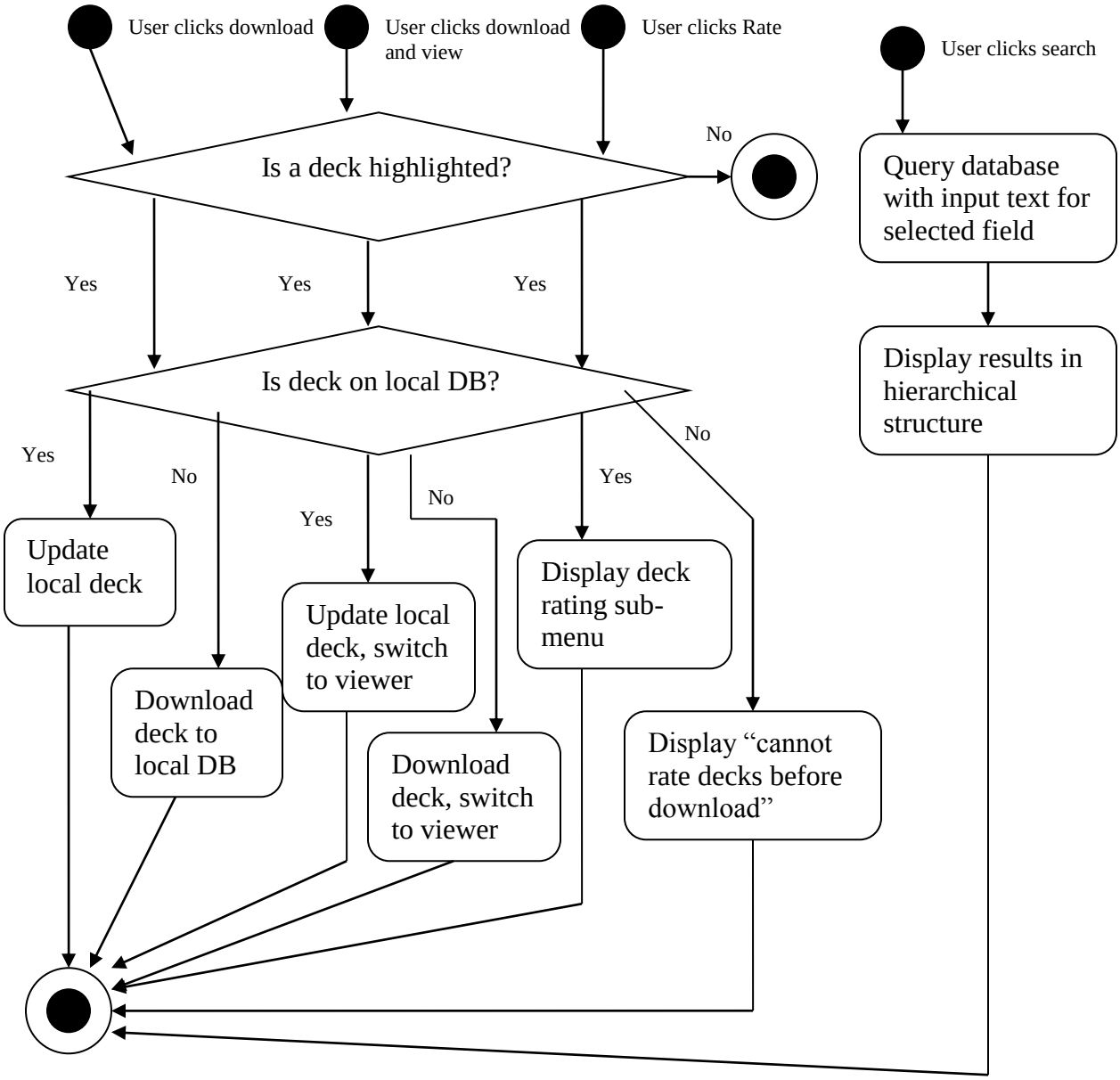
Activity Diagram: Network Window Initialization



Activity Diagram: Networking Left Panel



Activity Diagram: Networking Right Panel



Features – Compatibility

eFlash faces several compatibility issues. First, our biggest issue is that because our software uses Microsoft's .NET Framework, .NET must be installed on the user's machine for eFlash to run at all. We will create an installer to automate this task, but as listed in our minimum system requirements, no official support is provided for non-XP installations.

The user's machine's file system must also be compatible with our application. Any machine using the Windows file system will be fine. Additionally, there is a compatibility issue with the user interface. We will try to make it as intuitive as we can and emulate the Windows user interface, which the user is most likely already familiar with.

Features – User Interface

See System Architecture and Features sections above.

Features – Other Interfaces

We will be utilizing the Microsoft PowerPoint interface to export flashcards to PowerPoint Presentation format.

External Dependencies

- Microsoft Windows XP - eFlash is developed on windows platform and therefore can only be used on Microsoft Windows platforms.
- .NET Framework – eFlash is developed through Microsoft Visual Studio 2005. Therefore we utilize .NET Framework's features to enhance this part of the application.
- MySQL Database Server – for managing and storing flash cards.
- Music file – eFlash can automatically generate titles and descriptions for the music flashcards through the ID3 tag of mp3 files. Therefore, eFlash currently supports mp3 format only.
- (Optional) Internet connection – eFlash allows users create and view their flashcards locally. However, internet connection is required if users want to share their flashcards through eFlash network.

Features Limitations

Due to time constraints, the design describe in this document does not allow the user as much control over the layout of the flashcards as we hoped. If time allows, we will implement a fully customizable layout that allows the user to drag and drop objects such as textboxes, sound sliders, and images to arbitrary locations onto a blank canvas.

Testing Strategy

Testing will be a continual process in the implementation of this product. We will use a bottom-up testing strategy, taking into account unit, integration, and regression testing.

Unit Testing

The testing process begins with each component of the system going through unit tests. Each unit is expected to pass without error before integration testing.

Profile Selector Popup

Testing on Profile selector popup will involve making sure that it displays profiles correct and allows users to choose the profiles; make sure it prompt for password if it is required for the profile. Also, we have to make sure that its GUI links to *New Profile Popup, Main Menu*, and *Profile Manage* correctly.

New Profile Popup

Testing on New profile popup will involve making sure that it prompt user for profile name and the optional profile password at the beginning. If the checkbox for enabling network access is checked, we have to make sure it checks the uniqueness of the network login name. Similar to Profile selector popup, we have to make sure that its GUI links to Profile Selector Popup and Profile Manager correctly.

Main Menu

Testing on Main Menu will involve making sure that a short description is display when users place the cursor on the items in the main menu. Moreover, we have to make sure that its GUI links to Profile selector popup, Deck Manager Screen, Viewer Collection Setup, and Network login correctly.

Deck Manager Screen

Testing on Desk Manager Screen will involve making sure that it display a list of all deck in the local repository in an expandable hierarchical manner. We also have to check whether it is first sorted by category then by title of the deck. Moreover, we have to make sure that the buttons for creating, editing, deleting, and importing decks are working properly. Finally, we have to make sure that its GUI links to Main Menu, Deck Creation/Edit Screen, and Import Screen correctly.

Deck Creation/Edit Screen

Testing on Deck Creation/Edit Screen will involve making sure that it displays a list of deck's flashcards' primary fields; its textboxes for Category/Title/Description function correctly. Also, we have to check new, edit, and delete buttons for flashcard function accordingly. Finally, we have to make sure that its GUI links to Deck Manager Screen and Flashcard Creator correctly.

Import Screen

Testing on Import Screen will involve making sure the file importer can import different text file formats suitably. Moreover, we have to make sure that its GUI links to Deck Manager Screen appropriately.

Flashcard Input Gatherer Screen

Testing on Flashcard Input Gatherer Screen will involve making sure we have textbox for each object in the flashcard and allowing the user to navigate to other flashcards without going back to the Deck Creation/Edit Screen. We have to also test the correct connections between Flashcard Input Gatherer Screen, Flashcard Creator and Deck Creation/Edit Screen.

Viewer Collection Setup

Testing on Viewer Collection Setup will involve making sure correct GUI connections between Quiz setup, Exporter and Main Menu and users are able to create quizzes and export flashcard documents.

Network Welcome Screen

Testing on Network Welcome will involve making sure a user with a unique network account stored in user profile can be grated access to his/her File Management Screen after matching the password stored on the server as a hashed entry. We also have to ensure a correct GUI connection between Network Login and Network File Management Screen.

Network Uploader/Downloader Screen

Testing on Network File Management Screen will involve making sure users can upload/download/remove their files on local/remote repository. We also have to ensure file sharing between different users can be performed securely and conveniently; Remote File listing can display correct flashcard titles/category/author etc.

Network Ranker

Testing on Network Ranker will involve making sure that users can only rank once, and that the rank is stored correctly in the remote database. In addition, it must recalculate the network user's average rank.

Integration Testing

System Integration test will be performed after the successful completion of the Unit Testing stage. Since eFlash is basically an Object Oriented System the integration testing will be predominantly done in a bottom-top manner. We will need to ensure that all the module interfaces are utilized correctly by other modules. Those tests will be run with every new build to ensure correctness.

Regression Testing

After a major bug fixed or source code changes that may have inadvertently introduced errors, regression testing will be performed. We will try to verify that the newly added features/bug fixed will not affect previously working functions.

Support

The purpose of eFlash is to provide an intuitive stream-lined alternative to the analog task of flash card creation and maintenance. As such, the GUI will be as simple and self-explanatory as possible. Helpful features will include pre-built card set templates for specific uses (i.e. a foreign language template). Step by step instructions for creating decks will also be displayed online at the eFlash™ website.

Licensing

Given that our product's primary use is to stream-line an already widely practiced analog experience, our software will be distributed freely under a public license agreement allowing dissemination and use by anyone. The source code (and hence any further modifications and updates) will be reserved solely for eFlash™.

Benchmarks and Metrics

The benchmarks and metrics of our product will be based upon user feedbacks and customer surveys. Users will be asked to answer a list of questions regarding to the functionality, user-friendliness and their general satisfactions of our product for each beta release. Some of the questions will be score-based, for which a numerical score will be assigned by the users.

Sample questions:

1. What is your satisfaction on user interface design? (score)
2. Is it easy to import or export flash card documents? If not, please specify
3. Is it fast to connect to our server? (score)
4. Do you satisfy with the way we categorize the flash cards? If not, how do you want them to be categorized?
5. Do you like the way flash cards are shared among users? If not, how do you want us to improve?

After collecting users' feedbacks, we will do a series of tests and modifications to improve the overall quality of our product. Moreover, we will record the time needed for common operations like exporting or importing documents, retrieving document list from server and opening document templates. We will then compare the time needed for all these operations with the previous beta release, to make sure newly added or modified operations take reasonable time.

Effect on Methodology

eFlash will revolutionize the use of flashcards by eliminating the annoyances and limitations of physical flashcards while providing a wealth of additional features. Time-consuming organization and management processes will be automated by providing templates that allows users directly drag-and-drop the items into flash cards. Additionally, the functionality of flashcards is vastly enhanced, expanding upon current flashcard technology as well as extending out to fields never before possible; music flashcard can be easily created with the music flashcard template, which contains a sound-file slider that allows users to seek a segment of the music file. Moreover, eFlash's export feature, which allows flashcards to be exported as Microsoft PowerPoint (ppt) format, makes flashcards can be easily viewed and printed. With eFlash, students across countless fields will be able to enjoy the benefits of flashcards without the associated headaches.

Issues

There are no **open** or **resolved** issues at this time.

Checklist

The following checklist is provided to help the reviewers (and author) prepare for the review by providing a set of questions for the reviewer to answer about the specification document. If the answer to any question is "no", that item should be identified as an issue at the review. The checklist is only a guideline; it should not be solely relied upon for a complete review. Reviewers may want to add their own questions to the checklist.

Y	N	
		CONTENT
___	___	If they exist, have the Requirement and Methodology Specifications been reviewed and accepted by the appropriate reviewers? (This is a prerequisite).
___	___	Are all the interfaces to the system specified in detail?
___	___	Are acceptable solution alternatives and their trade-offs specified?
___	___	Are you satisfied with all parts of the document?
___	___	Do you believe all parts are possible to implement?
___	___	Are all parts of the document in agreement with the product requirements?
___	___	Is each part of the document in agreement with all other parts?
		COMPLETENESS
___	___	If requirements are included, are they clear and complete?
___	___	Where information isn't available before review, are the areas of incompleteness specified?
___	___	Are all potential changes to the Functional Specification specified, including the likelihood of each change?
___	___	Are all sections from the document template included?
___	___	Are you willing to accept the specification with the items in the open issues section unresolved?
		CLARITY
___	___	Does the functional solution avoid specifying the low-level implementation?
___	___	Is the solution at a fairly consistent and appropriate level of detail?
___	___	Is the solution clear enough to be turned over to an independent group for implementation and still be understood?
___	___	Is each requirement (or feature list item) measurable (will it be possible for independent testing to determine whether each requirement has been satisfied)?
___	___	Are all items clear and not ambiguous? (Minor document readability issues should be handled off-line, not in the review, e.g. spelling, grammar, and organization).

Appendix: Methodologies Employed to Create the Functional Design Specification

Original features discussion notes:

- Layout of categories:
  - Problem: there are many types of categories (by date, by subject, user-defined, etc.)
    - How to make it all fit together?
  - Allow subcategories?
    - Could make categories on network messy
  - Final decision:
    - Each flashcard has a Category, a Title, and a Description
      - Category (subject): Swahili, Biology, etc.
      - Title: "Week 1 stuff", "Difficult words", etc. Input by user
      - Description: optional description input by the user
- Network category structure problem:
  - How to lay out the category structure on the network?
  - How to resolve category-naming conflicts?
  - Decision:
    - Each deck of cards uploaded will have the Category, Title, Description, and the upload user's login.
- Allow videos in flashcards?
  - No
- Is network function a standalone website? Or is it integrated into our application?
  - Integrated into application for now
- Profile and network login:
  - Problem:
    - Users need a unique login name for network access
    - Users who don't want to use the network should be able to use any name they want (not checked for uniqueness)
    - But having two user names and passwords is inconvenient for user
  - Decision:
    - Each user has a non-unique profile name and an optional password
    - If user wants network access, he must create a unique login name
      - That login is stored in his user profile
      - But doesn't need a separate network password
      - Because network login is done through our application, logging in to the user profile can serve as network login validation