Porting Open Source E-Learning Software to Mobile platform

Submitted by:

Jamael Tanveer 07110088

Md. Al- Mubin Sultan 07110013

Supervisor: Matin Saad Abdullah

Co-Supervisor: Dr. Mumit Khan

School of Engineering and Computer Science
August, 2010

BRAC University, Dhaka, Bangladesh

Acknowledgement

First of all we are grateful to Allah for giving us strength and wisdom throughout the whole life. We thank our family for their love, their moral and financial support they had given us.

We would like to give our gratitude for our thesis Advisor Matin Saad Abdullah, Senior Lecturer, Department of CSE, BRAC University, who had given us significant suggestions and inspections during the whole process of the thesis work. From the form of primary, we really acquire much knowledge about how to organize well-structured Software step by step.

We would also like to give our gratefulness to our Co advisor Dr. Mumit Khan, Professor and Chairperson, Department of CSE, BRAC University. He always stayed beside us, encouraged and sustained when we felt fatigued and puzzled.

Table of Content

Abstract

jMemorize is an open source learning software. It is very powerful, well designed, and user-friendly software, which is widely used all over the world. jMemorize is written in Java and uses Leitner flashcards to make memorizing facts not only more efficient but also more fun. As people spend a lot of time in mobile it is easier to learn something from mobile. So, porting this software to mobile platform will be very useful for the user of Jmemorize.

Objective

Our goal for this thesis is to port selected functionalities of jMemorize to mobile phones by understanding the system design and use cases of the software and to learn software design and development issues in mobile platforms. We also hope to contribute our work to the jMemorize open source community.

Designing issues of mobile platform

User Interface Design for Mobile Technologies

Interaction design is about finding the right ways to deploy the dazzling new technological capabilities to meet real user needs. Without effective interaction design, there will be negative emotional, economic and ethical impacts. What people want is how they are interacting with the system and how much are they satisfied with the system. And very straight forward answer is design of the interface with what user interacts and then comes the system feature and its uses. A much featured system can be used by fewer users because of its interactional design. Whether, a less featured system can be used by more users with its user friendly interactional design.

Interaction is a brand and its best example is APPLE. They had a reputation for producing technology with high design values and excellent usability. It was one of the first computer companies to see the value of making ease and pleasure of use a key part of it brand.

There are some golden rules for interface design

- o Strive for consistency
  - Enable frequent user to use shortcuts
  - Offer informative feedback
- o Strive for prevent error
  - allow user to undo

o Make user feel they are in control of the system o Reduce short term memory load

Some major factors regarding interaction designing

Interaction design is done in two levels

- Application level
- User Interface level

Application level

Ethnography and innovating new ideas are two major factors on application level designing. Ethnography comes from the word ethnicity. Ethnography is primarily form of reportage it consists of data about understanding users, developing prototype design and evaluation done on an ethnic group. From this report basically depending upon this report a system is designed starting from interaction design to the features the system will have.

Innovating new ideas are very important as users are waiting for innovative mobile services. But this approach should be 'User -Centered' because ultimate goal of building a new system is meeting real customer needs.

In designing issues technology should not be focused on experiences rather than it should be more focused on real customer needs and should be designed as their needs.

Innovating something is not just a innovation now it is more like 'Meeting the user needs with new ideas'.

User Interface level

Interface style and device types are factors that influence user interface level designing. Interface style is determined in terms of required functionality of the system as well as on access to the resources by the system. It is because depending upon the functionality and the resources accessed by the user the interface needs to be created for user convenience.

Device type is also another factor what certainly influences user behavior. For an example keypad of mobile phone is different for different brands as for different models as well. So it is not a good idea to stick on a rigid design for all type of device.

From some case study it is found that one handed phone user is quite common and people uses mobile by one hand when walking some people uses it by two hands when they are sitting somewhere.

Considering these factors as well as the system behavior interaction design needs to be designed.

Major Challenges of Mobile Platform

Desktop platform and mobile platform has very big structural difference. So the features existed in desktop version may not be the same in mobile version. While porting software from desktop to mobile, there will be difficulties and some challenges have to be faced. The developer should try to keep the core features as it is and change the other features as required. There are mainly four major challenges to face while porting a desktop application to a mobile platform:

- Form Factor
- User Interaction
- Hardware Constraint
- Software Development Constraint

Form Factor

Form factor is one of the major challenges for a developer to develop an application in mobile platform. Device type, size, keypad, etc these all things should be taken into consideration before developing a mobile application. Mobile device has limited screen space. So the many things have to be confined within this little screen. Moreover, the application in mobile platform may not have all the features like the desktop application due to this limitation.

Mapping keys appropriately is another major concern in developing a mobile application. Interactive keys should be mapped in a way that a user doesn't face any difficulties in using the software.

New technology is coming everyday. A developer should be aware about it. The software should be developed according to the technological change.

User Interaction

The way a user interacts with the mobile phone is another concern in developing a mobile application. If a user can not use the software interactively, then it's a failure for a developer. So the software should work in potentially distracting usage environment. For example, while a user is running an application, if he gets an incoming call, the software should be paused and should be kept as it is until the call ends. But if the software hangs at that moment, that user will never use that software. So these things should be considered to develop interactive software.

Surrounding environment is another factor for a mobile application. For an example, light intensity inside a building is different from that of outside. So the screen illumination should be adjusted according to the light of the environment.

User movement and mode of operation should also be taken into consideration. A user may walk or sit while using the software. He may use one hand or both hands. The usability should be according to the requirements so that, the user does not face any difficulty in using the software.

Hardware Constraint

The vital difference between desktop and mobile platform is the hardware. There is a big hardware limitation in mobile device. Mobile device has very low memory space and slow processing time. If a developer wants to make a big software for mobile, that is not going to work. Whatever a developer makes, the system requirement should be as low as the mobile configuration. So, while porting from big software to mobile platform, some features those take up higher memory should not be included or should be included with minimal configuration.

Software Development Constraint

There are also some designing limitations in mobile platform. There are lots of desktop libraries that mobile platform does not support. For example, JAVA has built in XML parser. But J2ME does not support XML. To get the support, a third party library should be imported.

Since the mobile device has very limited memory space, so the application should not be very big. So many features of the desktop application may not be included in the mobile version.

Introduction to E-learning Software

When people started memorizing they were used to memorize things by reading it out loudly again and again by looking at it and try to remember it not looking at it. But now everything is changing and shifting to e-media. People now prefer more e-media than hard copy for learning. For an example reading from a book or from hard copy of documentation is sometimes helpful but reading from computer or from soft copy is much easier and helpful because you can access multiple resources at a time. By the changing of time e-learning software has become popular for memorizing. Software that helps people to learn and memorize things is e-learning software for memorization.

Overview of the software (Both Mobile and Desktop Platform)

jMemorize is a Java application that manages your flashcards by the famous Leitner system and makes memorizing facts not only more efficient but also more fun. jMemorize manages your whole learning progress and features statistics, categories and a visually appealing and intuitive interface.

The basic idea is to divide the cards into different decks depending on the difficulty they present to you. This is done by repetitive quizzes in which you try to answer the question out of your mind. Every time you know the correct answer to a card, it is put on the next higher card deck. If you fail at a card, it is put back to the starting deck.

This system is combined with time schedules. Cards that have been known are considered to be learned until a specific expiration date has passed. The higher the deck, the more far away the expiration date is set.

For example, a card that has been successfully checked for the first time is scheduled to be relearned one day later again, while a card that has been correctly answered three times in a row might be considered as learned for about week. As long as a card is considered learned, it will not appear in learn sessions.

As a whole, this system manages your personal learn sessions and allows you to focus on learning, while it automatically decides which facts should be learned right now to make the most out of your time.

Here are some key features of "jMemorize":

- Create, edit, remove and reset cards.
- Arrange cards by creating card categories.
- Have a visual representation of your current learn situation.

- Customize your learn session setting in nearly every way imaginable. This includes the abilities (among others) to
  - Choose which category to learn.
  - Choose to learn only new cards, expired cards or all cards.
  - Choose card/time limits.
  - Choose whether you want learn cards in normal mode, with flipped card sides or randomly flipped card sides.
  - Choose from different preset time schedules or enter you own custom time schedule.
  - Search through your cards with the find tool.

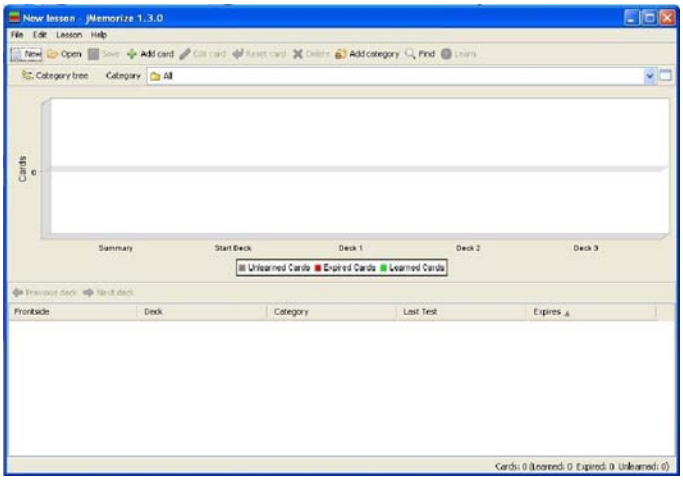


Fig: jMemorize Main Form

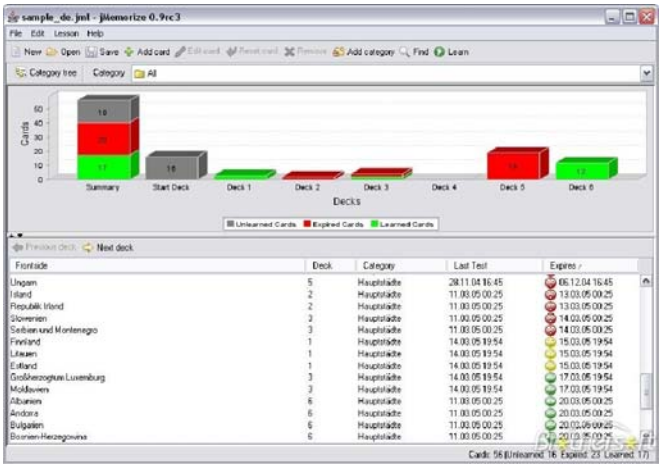


Fig: jMemorize Learn Statistics

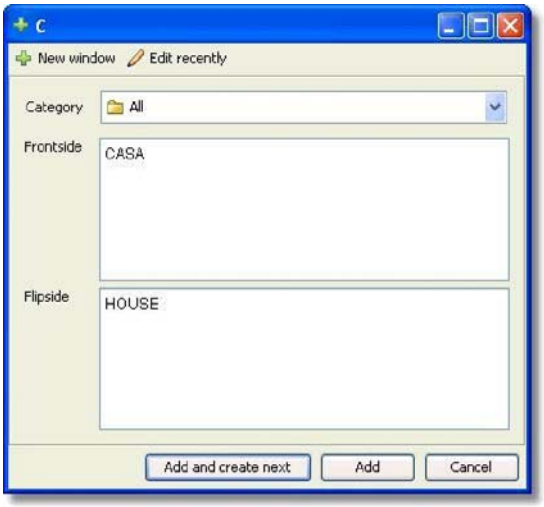


Fig: Card Add/Edit

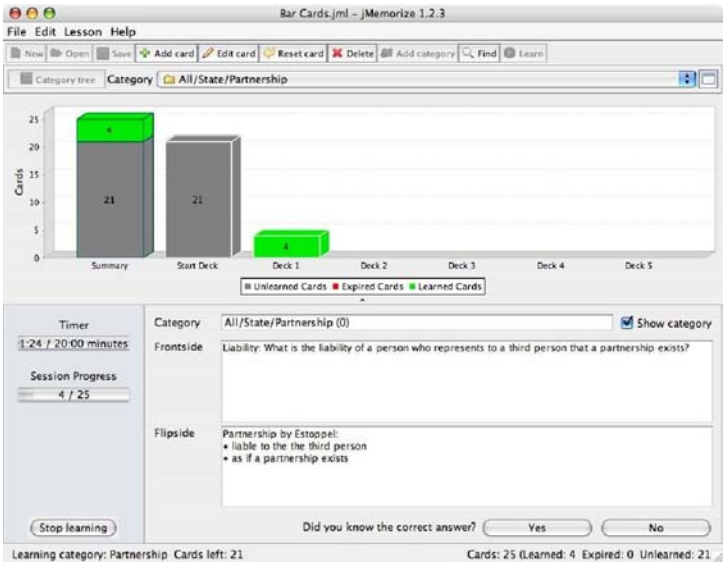


Fig: Learn Session

Use Case of jMemorize(Desktop Version)

Before developing the mobile version, we prepared these following use cases that helped us to design the mobile application in well organized way.

Actor: Administrator

Use Case Name: Importing file.

Goal: Importing cards from different file format (.csv, .tsv file) or from a whole jMemorize lesson.

Summary: This is a very useful part of jMemorize. .csv or .tsv file format stores number of cards in it. If .csv or .tsv file is imported in an existing lesson, all the cards in the file will be added to the list of cards of the existing lesson. Incase of importing jMemorize lesson, the expiration status will also be imported along with the cards.

Basic Flow:

1. Clicks on the "File→Import" from Menu bar.
2. Selects any of the three options (.csv, .tsv, jMemorize lesson).
3. An Open dialogue box appears.
4. Goes to desired directory.
5. Selects the desired file and clicks "Open".
6. Cards from the file are added to the existing list.

Alternate Flow:

1. Imports unsupported or corrupted file.
    - A message box appears notifying the occurrence of error while loading file.
    - Shows debug information.

Use Case Name: Exporting file.

Goal: Exporting cards to different file format (.csv, .tsv file) or to a jMemorize lesson.

Summary: This is also a very useful part of jMemorize. The cards can be exported to .pdf of .rtf to keep it as a document. Cards can also be exported to .csv file format that stores all the cards in a single file. This file can be further used in other jMemorize lessons. Incase of exporting to jMemorize lesson, the expiration status will also be exported along with the cards.

Basic Flow:

1. Clicks on the "File→Export" from Menu bar.
2. Selects any of the four options (.pdf, .rtf, .csv, jMemorize lesson without personal data).
3. A Save dialogue box appears.
4. Goes to desired directory.
5. Selects the desired file name and clicks "Save".
6. Cards will be exported in desired file format.

Alternate Flow:

2. Exporting an empty list to a file.
   - The file will have no data in it.
3. Exports with a file name and format that already exists.
   - A message box appears asking for confirmation for overwriting.
   - Overwrites if Yes. Discards if No.

Use Case Name: Open and Save lesson.

Goal: Opening any saved lesson from Hard disk and saving any running lesson to Hard disk.

Summary: Open and Save plays a very important role in jMemorize. Learning process takes a huge amount of time. It is impossible to learn everything in one instance. When a part of a lesson is learned, a user can save the session up to that time. Then again open that session any later time. It helps the user to manage the time, since time is a big factor in jMemorize.

Basic flow:

Open:

1. Clicks on the "File→Open" from Menu bar.
2. An Open dialogue box appears.
3. Goes to desired directory.
4. Selects the desired jMemorize lesson file and clicks "Open".
5. The last saved session is then opened.

Save:

1. Clicks on the "File→Save" from toolbar.
2. A Save dialogue box appears.
3. Goes to desired directory.
4. Selects the desired file name and clicks "Save".
5. The session is then saved in current state.

Alternate Flow:

4. Opens unsupported or corrupted file.
    - A message box appears notifying the occurrence of error while loading file.
    - Shows debug information.
5. Saves with a file name and format that already exists.
    - A message box appears asking for confirmation for overwriting.
    - Overwrites if Yes. Discards if No.

Use Case Name: Setting preferences.

Goal: Setting the user interface according to the user requirement.

Summary: Setting preferences helps the user make the interface understandable by setting preferred language and font. It simply helps the user to change the working environment.

Basic Flow:

1. Clicks on the "File→Preferences" from Menu bar.
2. A dialogue box with few options appears.
3. Selects the preferred language from language list.
4. Selects whether the lesson should be compressed in GZIP format or not.
5. Selects the preferred font, size and alignment for different type of object like Frontside, Flipside for different environment like learning environment and card list environment.

Actor: Learner

Use Case Name: Review Stack

Goal: To see the summary of learning and to check the learned, unlearned or expired cards.

Summary: In the main interface, there are deck of cards and visual representation of the deck. It mainly shows the summary of learning of a user. The summary deck shows the overall result and list of all cards. Other decks represent the particular status of the cards.

Basic Flow:

1. By default, Summary deck is shown. The bar shows the percentage of learned and unlearned cards. The list shows all the cards in the deck and also their deck, category, last test, expires.
2. Clicks the “Next deck”
    - Start deck is shown.
    - List of unlearned cards are shown.

3. Clicks the “Next deck”
    - Deck 1 is shown
    - List of learned cards are shown.

4. Learn again the learned card
    - If correct, Card is moved to next deck
    - If wrong, Card is moved to previous deck

Use Case Name: Changing Learn Settings.

Goal: To make an easy environment for the learner, so that he/she can learn his/her lessons efficiently in his/her preferred way.

Summary: This part of jMemorize helps the learner to change different option according to his/her way. Learner can choose specific category, change the occurrence of the cards, sets the time limit or card limit, sorting the card order, change the appearance of the cards and schedule the delay of cards for different level.

Basic Flow:

1. Selects “Learn” from Tool bar.
2. A dialogue box containing different types of options appears.
3. Sets the General option.
    - Sets the category of cards from category list.
    - Selects whether to learn only unlearned cards or all unlearned and expired cards or only expired cards or some selected cards.
    - Sets the delimiters. Such as time limit, card limit and repetition of failed cards.
4. Sets the advanced option.
    - Sets the card order. Sets the percentage of cards that should not be sorted by their deck level.
    - Sets the grouping of the cards according to category. Under this, either chooses natural order or random order.
    - Sets the side mode. Either to choose front side, flipside or the random side.
5. Sets the Scheduling mode.

- Sets the delay of the cards to be expired for each level. Either chooses from presets or enters the time manually for each level.
- Chooses a fixed time of expiration of cards (optional).

6. Clicks “Apply” to save the settings.

Alternate Flow:

1. Clicks “Cancel” button.
    - Change of settings is discarded and gets back to the normal settings.

Use Case Name: Learn Session.

Goal: to learn the cards for that particular session under particular learn settings.

Summary: This is actual session where the learner starts learning the cards. The result is processed according to the learner’s response for a particular card. When a card is learned it is transferred to the next deck and marked as learned. At the end of the session the system shows the pie chart of the percentage of learning.

Basic Flow:

1. Clicks “Start Session” from Learn Settings dialogue box.
2. Timer starts.
3. Cards appear according to learn settings.
4. Clicks “Show Answer”
    - System asks whether the learner knew the correct answer or not.
    - If Yes, then card is transferred to the next deck and marked as learned.
    - If No, then card is transferred to the previous deck and marked as unlearned.

- Next card appears.

5. Clicks “Skip Card”
    - Next Card appears.

6. At the end of the session, the result appears. A pie chart is shown.
7. In the card deck, the status of the cards is changed, i.e. learned.

Alternate flow:

1. Clicks “Exit” during learning session.
    - A message box appears asking for saving the modified session.
    - Yes: Session is saved
    - No: Session is discarded
    - Cancel: Back to previous state

2. Clicks “Stop learning”
    - Session is stopped at that instance.
    - Cards and result is updated up to that instance.

Actor: Card Writer

Use case name: Create new category

Goals: Adding new category to the software to learn

Summary: By adding category it is ensures that different categories are distinguishable

Basic Flows:

- Clicks on add category
- A text box appears
- Enter the name of category
- new category is inserted

Alternate Flows:

- Clicks the cancel button, goes out from the text box
- checks add button without entering name, a message box entered that this is not valid

Use case name: Add new card

Goals: Adding new card to the software to learn

Summary: By adding card user learn it in next time and it is not deleted until user don't delete it.

Basic Flows:

- Clicks on add new card
- A dialogue box appears
- Select the category from list
- Enter information on both side of card

- Select add and create new card (adds and prompt again to create new card) or select on add(it only create new card and don't prompt)

Alternate Flows:

- Clicks the cancel button, exit from add card box
- checks add button without entering card information, a message box entered that this is not valid

Use case name: Edit card

Goals: Edit existing card

Summary: By editing card user edh information's saved on card

Precondition: There must be a card on the list

Basic Flows:

- Right clicks on a card
- Click on edit card
- A dialogue box appears
- Edit information on both side or on one side of card
- Finishes by clicking OK

Alternate Flows:

- Clicks the cancel button, exit from edit option

- checks add button without entering card information on a side or both, a message box entered that this is not valid

Use case name: Delete card

Goals: Delete existing card

Summary: By deleting card user delete the card not required anymore

Precondition: There must be a card on the list

Basic Flows:

- Select card from the list
- Right clicks on it
- Click on delete card
- Finishes by clicking OK

Alternate Flows:

- Clicks the cancel button, exit from edit option
- checks add button without entering card information on a side or both, a message box entered that this is not valid

Actor: Time

Use case name: Len System

Goals: To maintain time of memorization

Summary: By deleting card user delete the card not required anymore

Basic Flows:

- Expires card automatically even if the machine was switched off
- It triggers time and expires card automatically
- It is done automatically when the software is started

Synchronizing Mobile Application with Desktop Application

The desktop platform supports .JML file which is jMemorize Lesson file. Our mobile platform also supports it. At this time, synchronizing is done by direct copying and pasting the .JML file to the mobile platform. But data integrity is ensured.

How Our Application Works

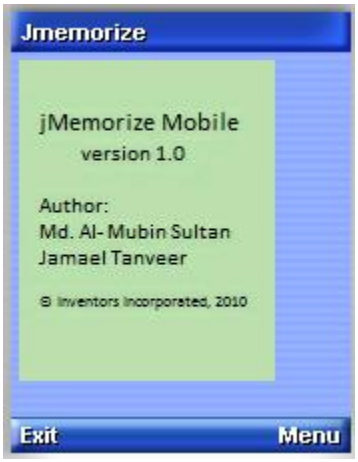
Requirement:

Firstly, create a .jml file from the jMemorize desktop version. That .jml file should NOT be saved in compressed (GZIP) format. To change this preference, go to File→Preferences. Uncheck the “Compress lessons in GZIP format”

After creating the .jml file, copy the file anywhere in the mobile phones gallery.

Procedure:

Open the Application.

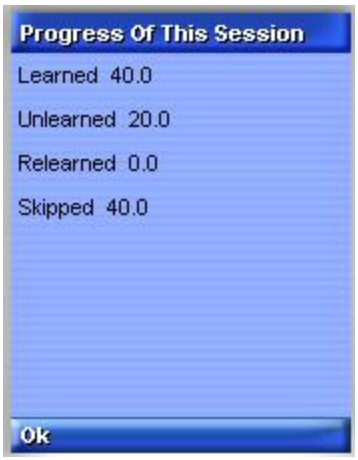


Select a .jml file with file browser.

Then the current status (the decks with the number of cards) will be displayed.

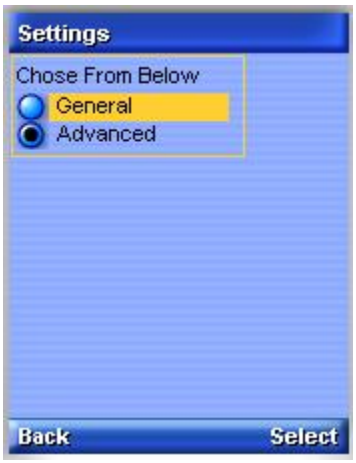
To start learning, select Menu→Learn. Then the cards will appear one by one.

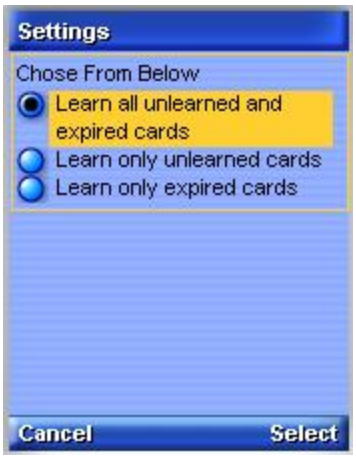
To see the progress of the session, select Menu→Progress of the Session.

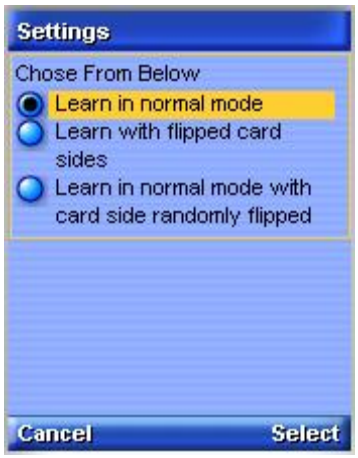


After learning is finished, Current Status will be displayed again.

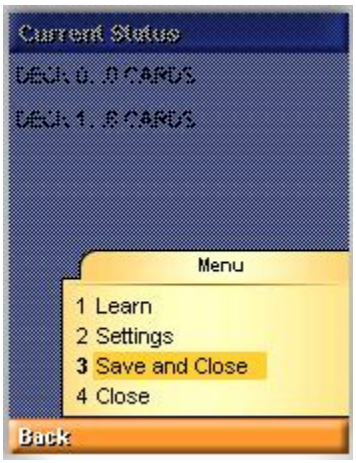
To change the learn settings, go to Menu→Settings. Here you can change the repetition of cards and learn mode.







To save and close the .jml file, go to Menu→save and close. To close without saving, go to Menu→close. It will take the user to the Open file page.



Difficulties during Development

We were aware of the limitations and the hardware and software constraints. But during the development of the software, these problems came out much severely than expected. The major problems we have faced are:

- Limited string length
- Unavailability of libraries
- Inconsistency between the device and the emulator
- All the forms are loaded in backend
- Problem in deck calculation

Limited string length

this problem occurred when we wrote the code for .jml file writing. The writer class can not write more than 34 characters at an instance. So we hade to break up the strings in to several lines, where as it could be done in a single line. This problem also occurred in some other classes as well. Moreover, because of this reason, we could not write our toString() method, which could have made our life a bit easier. But we had to choose another way to do that job.

Unavailability of libraries

J2ME has got very limited amount of libraries. We had to write the required libraries by ourselves. For example, J2ME does not support Date, which is highly required in our project. Fortunately, J2ME supports Calendar class. We made DateFormation class with the help of Calendar class. We had to write some other libraries like LinkedList, StringFormation and Time etc.

Inconsistency between the device and the emulator

One of the major difficulties we have faced is the inconsistency between the mobile device and the emulator. An application which ran well in the emulator, threw exception in the original hardware. This problem made our work difficult. For example, limited string problem does not occur in emulator, but occurs in mobile phone. For this reason we had to rewrite some classes like WriteFile. Another problem with mobile phone is the exceptions are thrown even if those are caught. So we had no option but to change lots of functionalities regarding exception.

All the forms are loaded in backend: Another major drawback of J2ME is it loads all the forms in the backend along with the active form. Some forms require some values from other forms. As a result, some forms do not get the required values and throws null pointer exception. So we had to individually restrict all the forms from loading.

Problem in deck calculation

when we were almost at the end of our project, we faced another problem with deck calculation. We did not have the clear idea about the deck calculation. The .jml file does not have any value regarding deck. Moreover to it, we did not have enough documentation of the desktop version of jMemorize. The situation made us really frustrated. Things were not actually working without calculating deck properly. Then we kept ourselves calm and analyzed a .jml file very precisely and found the solution for it. But for this new problem space, we had to rewrite some classes like Card, WriteFile etc.

Future Works

In future we will implement the following things:

- Add/Delete/Edit Card
- Reset Card
- Scheduling setting

We have implemented a simple GUI in this project. The decks and summary are shown by text. But in the later versions, we will update the GUI, so that the application gets a better look and feel. After developing the GUI, we will upload the application in sourceforge.

Conclusion

The major achievement of our project is the consistency between Desktop version and Mobile version. A user can easily switch a same .jml file between desktop and mobile version. In the end, we can say that it was a successful project. We could meet up all the requirements as targeted.

Resource

The .JAR and .JAD file and the Source code is available in the following link:
http://student.bracu.ac.bd/~u07110013

System Requirement:
MIDP 2.0 or Above

Book References

IGI Global Handbook of Research on User Interface Design and Evaluation for Mobile Technology

Written by:
Joanna Lumsden National Research Council of
Canada Institute for Information Technology –
e-Business, Canada

MOBILE INTERACTION DESIGN

Written by:
MATT JONES
University of Wales Swansea
GARY MARSDEN
University of Cape Town

References

Flashcard:

http://en.wikipedia.org/wiki/Flashcard

Leitner system:

http://en.wikipedia.org/wiki/Leitner_sytem

Leitner Cardfile System:

http://www.flashcardexchange.com/docs/leitner

jMemorize(Sourceforge):

http://sourceforge.net/scm/?type=cvs&group_id=121967