CHAPTER 4: REQUIREMENTS ANALYSIS AND DESIGN

This chapter highlights requirements analysis and system design of ML-Kit. Requirements analysis aims to elicit requirements of the users, modeling and analysing these requirements to be documented as a basic design for the coding phase.

4.1 Requirements Analysis

Requirements analysis is the process of understanding and analysing the dependencies for realisation of a proposed system and is a well-defined stage in the Software Development Life Cycle model. Requirements describe how a system should behave and does not provide technical information of how to implement the system properties. It can also be a statement of ‘what’ an application is expected to do (Hull, Jackson, and Dick, 2004).

Software requirements specification is a complete description about the system to be developed. The software requirements analysis process covers the task of documenting requirements from users in the form of use cases that describe all of the interactions that the users will have with the software. The software requirements specification contains functional requirements and nonfunctional requirements in addition to use cases.

4.1.1 Functional requirements

A functional software requirement is usually the best way to match a user’s needs and describes what the developer is supposed to build (Wiegers, 2005 b).

Functional requirements provide the blue print of the system. It includes details of how the system functions based on well defined functionalities such as the input, actions and output. The actions include technical details, data manipulation and processing related to the specific functionality.
Most of the functional requirements can be described using the use case diagrams. ML-Kit is divided into four main modules which are lessons, exercises, games and past year exam papers as illustrated in the use case diagram as shown in Figure 4.1.

### 4.1.2 Use Case Diagrams

Use cases describe the system behaviour from functionality initiators point of view. Use cases is represented as a sequence of simple steps and describes the interaction between the initiator and the system (Rosenberg and Stephens, 2007). The Standard Four students are the initiators in ML-Kit.

The use case diagrams can be used to show all of its available functionality. It is also used to capture a system's behavioral requirements by detailing scenario-driven threads through the functional requirements (Rosenberg and Stephens, 2007).

A ‘use case’ was used during the design of ML-Kit to explain the interaction between users and the system when carrying out an operation. Figure 4.1 displays the use case diagram of selecting a learning module. The use case shows four main operations ‘select chapter,’ ‘select exercises,’ ‘select games,’ and ‘select past year exam papers.’
Figure 4.1: Use Case Diagram for ML-Kit
a) **Select Chapters**

This function allows user to select any of the 15 chapters from the module. Once a chapter is selected, animation of the selected chapter will be loaded to allow the user to learn at his or her own pace.

b) **Select Exercises**

This function allows user to practice Mathematics questions at their own pace.
c) Select Games
This function allows user to play games after doing exercises. One game is included for each chapter to make the learning interesting and to help the user to remember the formulas.

d) Select Past Year Exam Papers
Past year Mathematics examination papers for 2008 are included in this module. The user has to answer both the objective and subjective type of questions.

4.1.3 Class Diagram of ML-Kit

Figure 4.2 shows the class diagram of ML-Kit. There are six classes - user, main menu, lessons, games, exercises and past year exam papers. There is also one sub-class which is inherited from lessons which is animations and one sub-class inherited from lessons, games, exercises and past year examination papers.

A user can access to one main menu, which are denoted by 1 to 1 relationship. Besides that, user can access to one exercise, games and past year exam papers class, which is denoted by 1 to 1 relationship.
4.1.4 Sequence Diagrams of ML-Kit

A sequence diagram depicts the sequence of actions that occur in a system. Sequence diagrams are organized according to time and this makes the diagram a useful tool to represent the dynamic behaviour of a system (Rosenberg and Stephens, 2007). The sequence diagrams of the four main functions of ML-Kit are shown below.
a) **Sequence Diagram of the Lesson Module**

The lesson module allows a user to choose lessons from any of the 15 chapters. Lessons options can be selected from menu page or other pages in this system. Figure 4.3 shows the sequence diagram of Lesson Module.

![Sequence Diagram of Lesson Module](image)

**Figure 4.4: Sequence Diagram of Lesson Module**

b) **Sequence Diagram of Exercises Module**

This function allows user to enhance their learning by solving exercises. There are ten questions in each chapter. Each objective question displays 4 answers for the user to choose. A check button is included for the user to refer if he or she is not sure of the answer. Figure 4.4 shows the sequence diagram of Select Exercises Module.
c) **Sequence Diagram of Games Module**

This function allows user to select games. Each chapter has one game to enhance the understanding and make the learning interesting. Figure 4.5 shows the sequence diagram of Games Module.
e) **Sequence Diagram of Past Year Exam Papers Module**

This function allows user to select past year exam papers. Two sets of exam papers with objective and subjective questions are included. Figure 4.6 shows the sequence diagram of Past Year Exam Papers Module.
4.1.5 Activity Diagrams of ML-Kit

An activity diagram focuses on the flow of activities involved in a single process. In the Unified Modeling Language (UML), activity diagrams can be used to depict the business and operational workflows in a system step-by-step and in a sequence. An activity diagram shows the overall flow of control (Rosenberg and Stephens, 2007).

a) Activity Diagram of Selecting Lessons Module

Figure 4.7 shows the activity diagram of selecting lessons. A user can choose any of the 15 chapters from the main menu. The user then selects the topic and then the lessons will be displayed.
Figure 4.8: Activity Diagram of Selecting Lessons

b) Activity Diagram of Exercises

Figure 4.8 shows the activity diagram of Exercises module. When a user selects exercises from the menu, the exercises are displayed. There are 10 questions in the exercises module and each question displays 4 answers for the user to choose.
c) **Activity Diagram of Games**

Figure 4.9 shows the activity diagram of Games module. When a user selects games from the menu, the images and texts are displayed. User has the option to click on the help button to learn how to play the game.
d) **Activity Diagram on Exam Papers**

Figure 4.10 shows the activity diagram of past year exam papers. Users can choose either objective or subjective type of questions from the menu.
Figure 4.11: Activity Diagram of Past Year Exam Papers
4.1.6 Non-functional requirements

Non-functional requirements define constraints - design or implementation such as performance requirements, quality goals and quality of service (Wiegers, 2005 b). The non-functional requirements of ML-Kit include:

i. Usefulness
ML-Kit can help students to improve their Mathematics by observing the examples illustrated using animations. The exercises module helps the user to practice what they have learnt in school and in the lessons module. The games module enables the user to test their knowledge by playing the games that are related to the contents of each chapter.

ii. Performance
All the modules in the ML-Kit will be able to load fast. ML-Kit is fully built on Flash and uses some Extensible Markup Language (XML). XML is used to store all the questions and answers for the exercises module, hence it loads fast. When performance testing was conducted, the response time to load each and every module was fast and there was no delay. Answers for the exercises can be retrieved and updated within 5 seconds by a few clicks of the mouse.

iii. Maintainability
Maintainability is the ability to enhance and repair the software easily (Perry, 2006). To modify ML-Kit, the programmer needs to understand the programming codes of all function. Suitable names are given to the variables, functions, and procedures when constructing the system. Therefore, it is important to understand the naming convention to do enhancements or bug-fixing (Perry, 2006). ML-Kit uses a standard naming convention for the variables, which consist of components such as text box, buttons, check box, etc. ML-Kit uses similar programming flow in the Action Scripts by using the reuse functions and procedures for many parts of the system. This facilitates maintenance process.
iv. User Friendliness

User friendliness is an important issue in system design. A lot of software available in the market is useful, but fails to achieve ease of use. Ease of use is a combination of how easy it is for users to learn the system and how easy it is to understand what the system does (Quesenbery, 2006). ML-Kit incorporates such features as Standard Four students can use this system with minimum training.

The non-functional requirements described above are tested and are further explained in system coding and testing in Chapter 5.

4.2 Hardware Requirements

The hardware requirements used to develop ML-Kit are shown in Table 4.1.

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Intel Pentium 4, 1.6 GHz</td>
</tr>
<tr>
<td>Memory</td>
<td>2 GB</td>
</tr>
<tr>
<td>Hard Disk Space</td>
<td>320 GB</td>
</tr>
<tr>
<td>CD / DVD-ROM</td>
<td>CD-RW or DVD-RW</td>
</tr>
<tr>
<td>Monitor</td>
<td>VGA or Flat Screen</td>
</tr>
<tr>
<td>Graphic Card</td>
<td>32 bit</td>
</tr>
<tr>
<td>Keyboard</td>
<td>Logitech Keyboard</td>
</tr>
<tr>
<td>Mouse</td>
<td>Any Optical Mouse</td>
</tr>
<tr>
<td>Speaker</td>
<td>Sonic Gear Speakers</td>
</tr>
</tbody>
</table>
4.3 Software Requirements

A number of software tools were used to develop ML-Kit. Table 4.2 shows the software tools used and their respective applications in system development.

Table 4.2: Software Tools for ML-Kit Development

<table>
<thead>
<tr>
<th>Software</th>
<th>Application</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Windows XP</td>
<td>System Requirements</td>
<td>Operating system.</td>
</tr>
<tr>
<td>Adobe Flash</td>
<td>Interface and Contents Design</td>
<td>To create animations and to convert them into movies.</td>
</tr>
<tr>
<td>Action Scripts 3.0</td>
<td>Programming Language</td>
<td>To control objects used in animations and to integrate with XML</td>
</tr>
<tr>
<td>XML</td>
<td>Programming Language</td>
<td>To retrieve questions and answers stored in Adobe Flash</td>
</tr>
</tbody>
</table>

4.4 System Design

The system design of ML-Kit includes architectural, animation, audio and user interface designs. These designs are described below.

4.4.1 Architectural Design

ML-Kit has three components – user interface, program modules, animation and audio files. The User interface communicates with program module. The program module then communicates to animation file and audio files. ML-Kit will retrieve data from the XML document which creates a link with animation files and audio files. It was not necessary to use database as XML is capable of storing data.
Figure 4.12 shows the architectural design of ML-Kit and how the components communicate with each other.’

![Diagram of architectural design of ML-Kit]

**Figure 4.12: Architectural Design of ML-Kit**

### 4.4.2 Animations

Animation is a simulation of movement created by displaying a series of pictures through frames (Peters, 2007). Animations of each frame for the lessons module had to be adjusted separately to be in sync with the audio.

### 4.4.3 Audio Design

ML-Kit has incorporated pre-recorded voices to pronounce the numbers, and words of each chapter. If there are amendments made to the texts, an audio file has to be re-created.

### 4.4.4 User Interface Design

A good user interface design has to have a simple design based on users’ requirements. The better the user interface design, the easier people can use the system without any assistance required (Stone, Jarrett, Woodroffe and Minocha, 2005). The user interface design of ML-Kit was based on the requirements given by the participants in the questionnaire survey. Samples of user interface design of ML-Kit are shown below.

Figure 4.13 shows the main menu of the user ML-Kit. Users can choose the chapters that they would like to learn from this menu. After choosing a chapter, the system will automatically display lessons on the screen together with the audio and animations. Figure 4.14 shows the chapter 1 lessons.
Figure 4.13: Main Menu Design

Figure 4.14: Screen Design of Lessons

Numbers and Place Values

The following beads shows that there are 3 ten thousands, 9 thousands, 4 hundreds 2 tens and 8 ones. So, altogether it comes to 30 428 or thirty nine thousand, four hundred and twenty eight.
Figure 4.15 shows the user interface design of the Exercises module. After clicking the exercise link on the main menu, exercises will be automatically displayed on the screen. Users have to choose one answer from the four choices given. The total number of questions answered correctly is shown on the screen. Scores are awarded to the users based on the correct answers.

![Image of Exercise Screen Design](image.png)

Figure 4.15: Screen Design of Exercises

Figure 4.16 shows the user interface design of the Games module. Users have to solve the problem given within a specific time. Games are incorporated as they make learning fun to the users.
4.5 Summary

This chapter discusses the user requirements which include the functional requirements and non-functional requirements of ML-Kit. The various UML diagrams for the functional requirement and system designs are discussed in this chapter.