Software Design Document (SDD)

TOEIC TEST SYSTEM

Date: 22/12/2018

Nguyễn Đinh Nhật Minh

Nguyễn Hữu Phúc

Võ Tr ần Huy Thông

Phạm Cao Nam Hải

Đỗ Đức Duy

Contents

The team for this project is,

Team Member	Student ID
Nguyễn Đinh Nhật Minh	51600049
Võ Trần Huy Thông	51600087
Nguyễn Hữu Phúc	51600049
Phạm Cao Nam Hải	51600023
Đỗ Đức Duy	51600020

Chapter 2

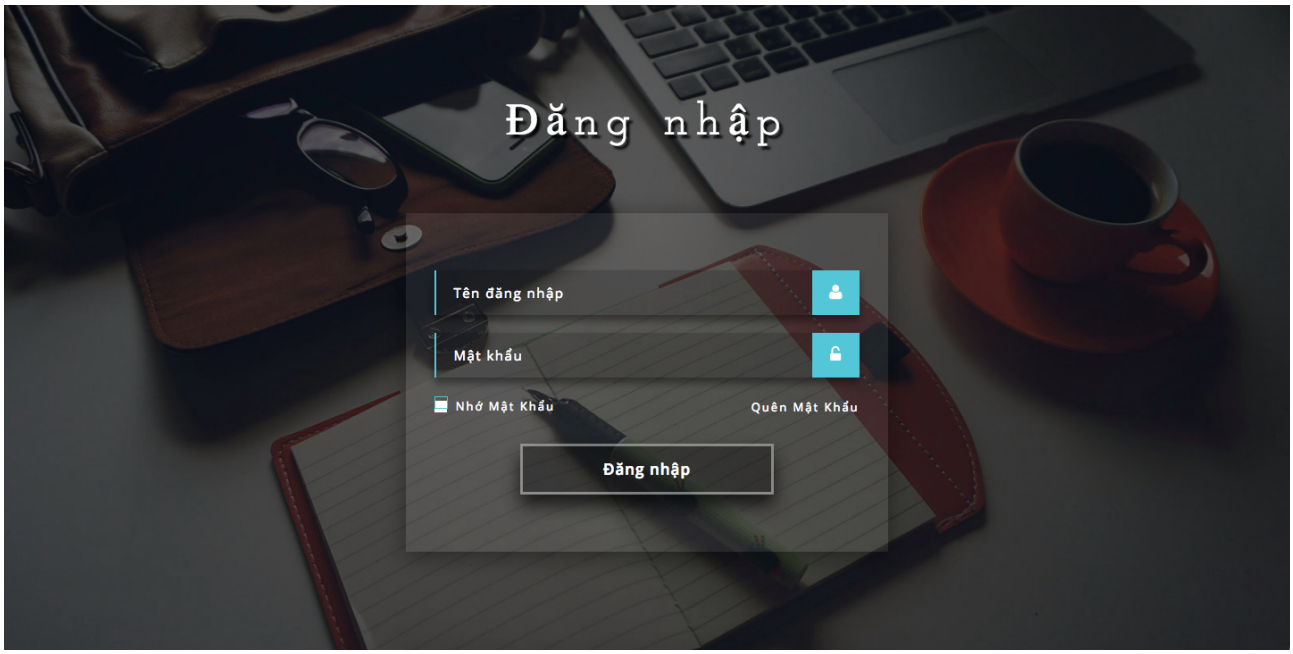
Module Design

The SDD module design of the Toeic Test System contains a detailed description of the objects defined. It will attempt to define methods, properties, accessors and, to a certain extent, provide algorithms or ways of approaching the coding process.

2.1 Class diagram

Figure 2.1 shows the class diagram for the system. This diagram uses UML to represent the classes. Consult a UML text for a data dictionary

2.2 The GUI login

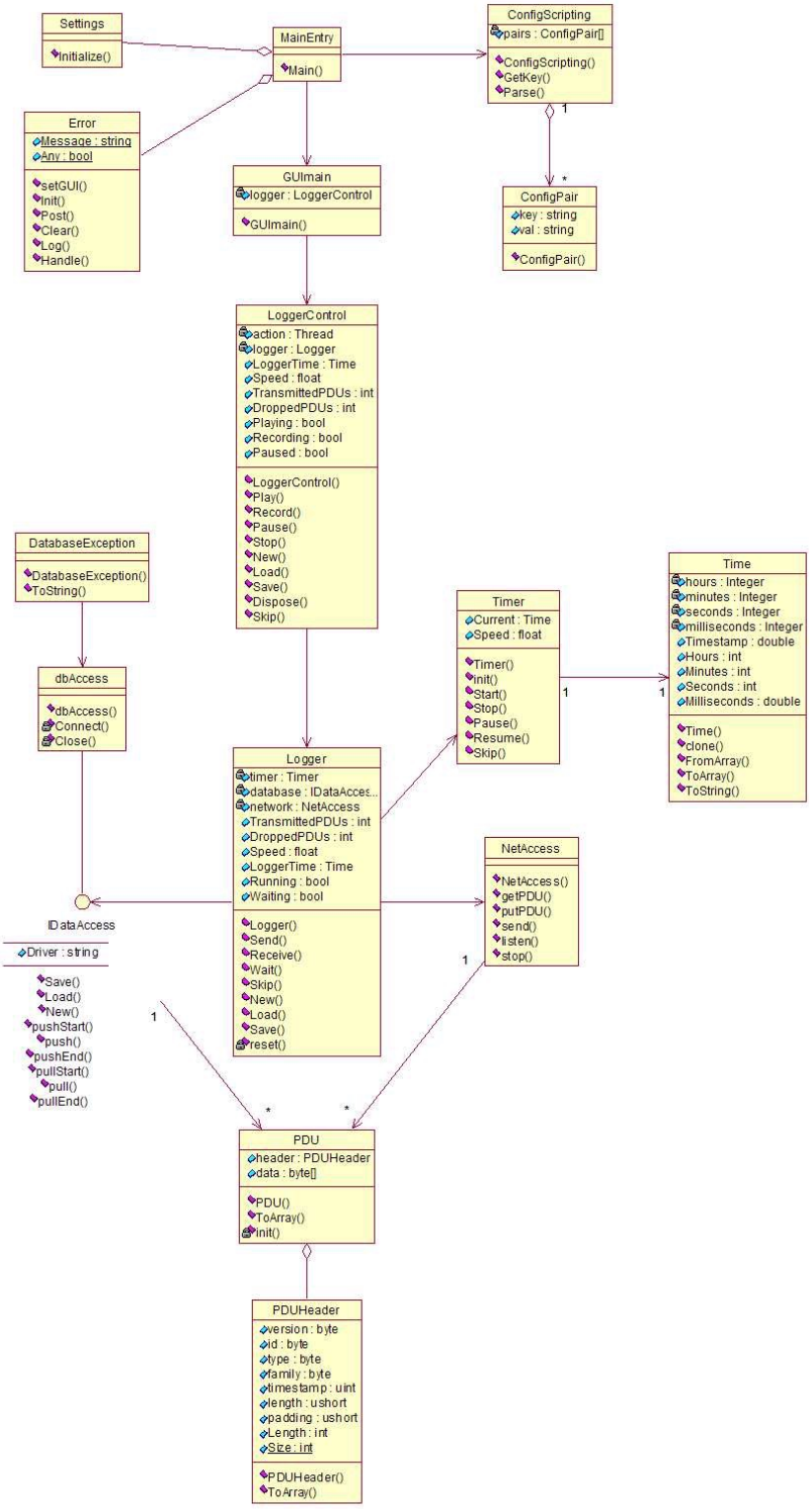


Đăng Nhập

Tên Đăng Nhập	Logo Login
Mật Khẩu	Logo Pw

☑ Nhớ Mật Khẩu Quên mật khẩu

Submit Button


Settings
Initialize()
MainEntry
Main()
ConfigScripting
pairs : ConfigPair[]
ConfigScripting()
GetKey()
Parse()
Error
Message : string
Any : bool
setGUI()
Init()
Post()
Clear()
Log()
Handle()
GUImain
logger : LoggerControl
GUImain()
ConfigPair
key : string
val : string
ConfigPair()
LoggerControl
action : Thread
logger : Logger
LoggerTime : Time
Speed : float
TransmittedPDUs : int
DroppedPDUs : int
Playing : bool
Recording : bool
Paused : bool
LoggerControl()
Play()
Record()
Pause()
Stop()
New()
Load()
Save()
Dispose()
Skip()
DatabaseException
DatabaseException()
ToString()
dbAccess
dbAccess()
Connect()
Close()
Timer
Current : Time
Speed : float
Timer()
init()
Start()
Stop()
Pause()
Resume()
Skip()
Time
hours : Integer
minutes : Integer
seconds : Integer
milliseconds : Integer
Timestamp : double
Hours : int
Minutes : int
Seconds : int
Milliseconds : double
Time()
clone()
FromArray()
ToArray()
ToString()
Logger
timer : Timer
database : IDataAcces...
network : NetAccess
TransmittedPDUs : int
DroppedPDUs : int
Speed : float
LoggerTime : Time
Running : bool
Waiting : bool
Logger()
Send()
Receive()
Wait()
Skip()
New()
Load()
Save()
reset()
NetAccess
NetAccess()
getPDU()
putPDU()
send()
listen()
stop()
IDataAccess
Driver : string
Save()
Load()
New()
pushStart()
push()
pushEnd()
pullStart()
pull()
pullEnd()
PDU
header : PDUHeader
data : byte[]
PDU()
ToArray()
init()
PDUHeader
version : byte
id : byte
type : byte
family : byte
timestamp : uint
length : ushort
padding : ushort
Length : int
Size : int
PDUHeader()
ToArray()
1
*

2.4 The Database Access Interface - IDataAccess.cs

Controls all access to the database used within the Toeic test system. Can be implemented using almost any database package

2.4.1 pushStart

Purpose: Create a new empty access database and initialize it (for example, create an empty table).

Prototype: public bool pushStart()
Inputs: None.
Outputs: Returns true if the method succeeded. Restrictions: None.

Called by: Logger.Receive() Calls: None.
Algorithm: –

• Initialize the database, ready for insertion.
• Algorithm is specific to the database package used.

2.4.2 push

Purpose: To insert a PDU and the current timestamp into the proper position in the database.

Prototype: public void push(Time time, ref PDU pdu)

Inputs: Time object specifying the timestamp for the data, and a reference to a PDU object to insert.

Outputs: None.
Restrictions: Assumes the PDU object is initialized correctly. Called by: Logger.Receive()
Calls: None.
Algorithm: –

• Inserts the data into the database.
• Algorithm is specific to the database package used.

2.4.3 pushEnd

Purpose: Close the connection to the database once insertion is complete.
Prototype: public void pushEnd()
Inputs: None.
Outputs: None.

Restriction: None.
Called by: Logger.Receive() Calls: None.
Algorithm: –

• Shuts down the database after insertion is complete. • Algorithm is specific to the database package used.

2.4.4 pullStart

Purpose: Connect to an existing database and prepare to read data from it.
Prototype: public bool pullStart(Time time)
Inputs: The timestamp at which to start reading data from.
Outputs: Returns true if the method succeeded (and there was data to read).

Restrictions: None. Called by: –

• Logger.Send() • Logger.Skip()

Calls: None. Algorithm: –

- Initialize the database, ready for selection.
- Read the first entry from the database (if there is one).
- Return true if there was an entry to read, and the method succeeded. Otherwise return false.
- Algorithm is specific to the database package used.

2.4.5 New

Purpose: To initialize a new database.
Prototype: public bool New()
Inputs: None.

Outputs: true if clearing the database was successful, false otherwise.
Restrictions: None.

Called by: Logger.New() Calls: None.
Algorithm: –

- Restore default values of settings relating to file and database.
- Algorithm is not necessarily database specific.

2.4.6 Save

Purpose: To save a database with a given name.
Prototype: public bool Save(string fullname)
Inputs: Full path name of file to save this database as.
Outputs: true if successful, false otherwise. Also outputs (to the filesystem) the file to be saved. Restrictions: Filename should be a valid Windows path and file name combination.

Called by: Logger.Save().
Calls: None.
Algorithm: A suggested algorithm is: –

• Copy current database to new database name.
• Delete old database if it was previously 'temporary' (that is, if the user didn't previously

create it and was created by EDDIS in a temporary directory with a temporary name).

2.4.7 Load

Purpose: To load an existing database and ready it for replay. Prototype: public bool Load(string fullname) Inputs: Full path name of file (database) to load. Outputs: true if successful, false otherwise.

Restrictions: File must exist. Called by: Logger.Load(). Calls: –

- Connect()
- Close()

Algorithm: –

• Find file.
• Connect to database file.
• Find the length of the exercise (last row's timestamp). • Close the database file.
• Set up any settings relating to file and database.

2.5 The Error class – Error.java

The Error class is used to report and log all errors that occur in the Toeic test system. All methods and properties of the Error class should be declared static, so they can be accessed anywhere within the system.

The Error class defines the following public accessor properties:

Message Gets the last message generated within the Error class. Resets the internal message string to empty. Read-only.

Any Gets if there are any messages currently waiting to be handled. Returns true if there are, false otherwise. Read-only.

2.5.1 Init

Purpose: Overloaded. To initialize the error reporting class. Prototype: –

1. public static void Error(Form main)
2. public static void Error(Form main, string log)
3. public static void Error(Form main, string log, bool debug)

Inputs: –

1. Takes a Windows form for displaying error dialogs onto. This should usually be the GUImain object instantiated in MainEntry.

2. As above, but also takes a filename to save log information to.

3. As above, but also takes true if debugging information should be shown (extra data about errors).

Outputs: None.
Restrictions: Log-file must be a valid filename. Called by: MainEntry.Main()
Calls: setGUI()
Algorithm:

- First two definitions calls the third definition with default values (log-file and debugging info should be set to settings stored in Settings).
- Sets all internal variables, and sets the destination form to main.

Chapter 3

Database Design

This section describes the basic design of the database that will store the event data derived from the PDUs, which is retreived from the network.

3.1 Table Design

Table 3.1 lists the structure of an individual entry (row) in the database. It shows the field name of each column, the MySQL type.

Field	MySQL
ID	int
Name Account	varchar
Password	varchar

3.1.1 Database Schema

The database schema is:

```
table(
   ID,
   account,
   password
  )
```

Chapter 4

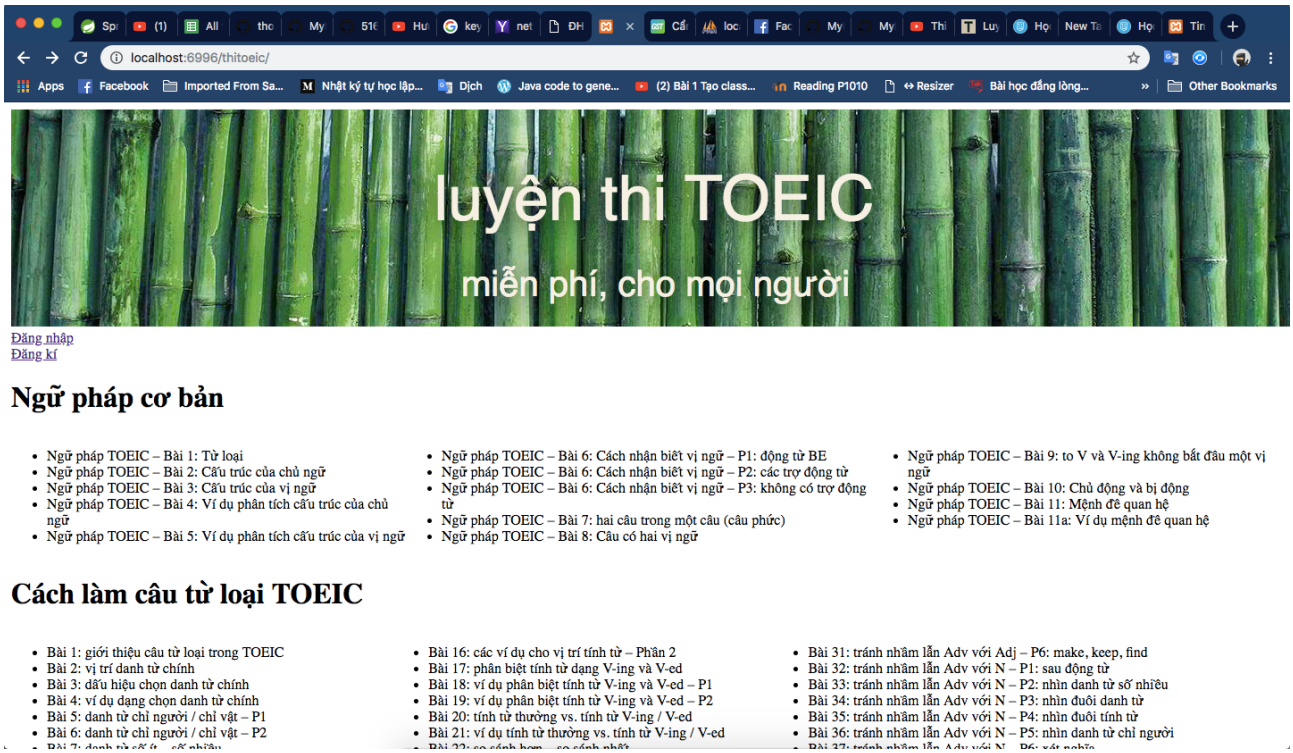
User Interface

The User Interface is a crucial aspect of the system in terms of both what the client wants and needs. For this reason there is an overview of the User Interface in the Software Requirements Specification (SRS). This section will detail all aspects of the UI and its design, and thus will be more extensive than the SRS section. These documents also have different target audiences and aims and thus different User Interface sections are presented.

The Graphical User Interface (GUI), for the purpose of this description, has been broken up into three main sections. These are:

- The Login
- The Registration
- The Toeic test

4.1 The Menu



4.2 The Registration

Đăng Ký

Tên Tài Khoản : Tên Tải Khoản

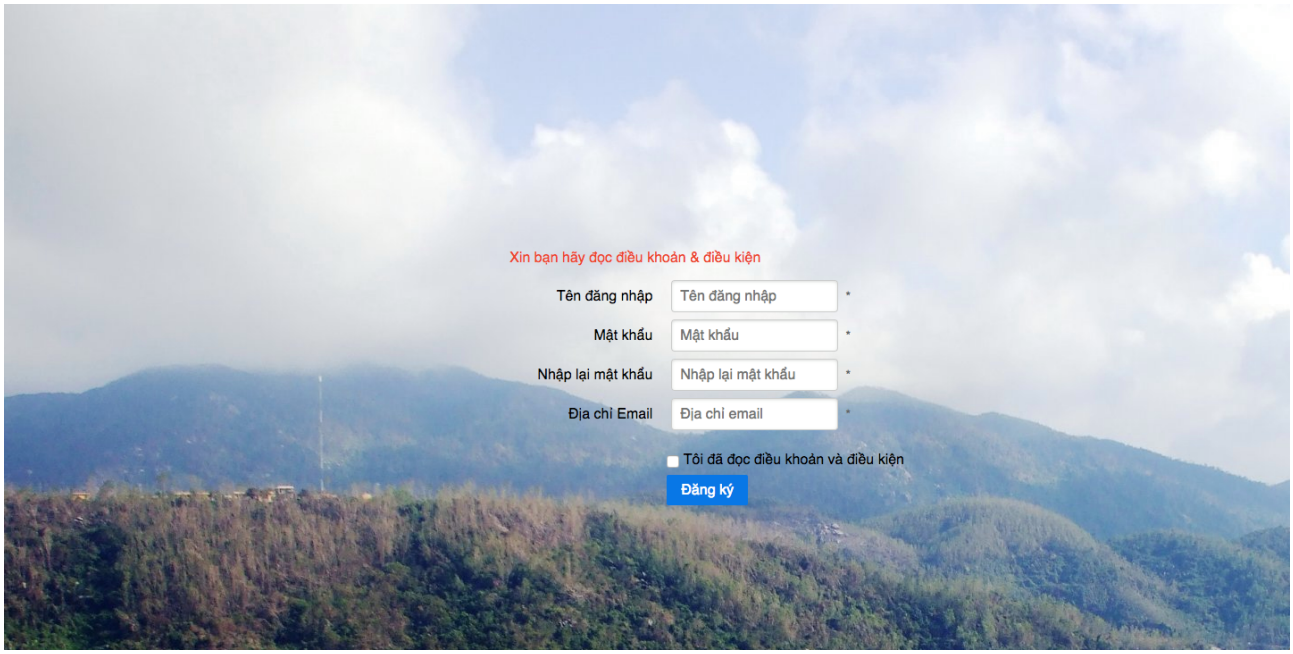
Mật Khẩu : Mật Khẩu

Nhập Lại Mật Khẩu : Nhập Lại Mật Khẩu

Email : Địa Chỉ Email

☑ Tôi đã đọc điều kiện và điều khoản

Button Submit



Glossary

GUI - Graphical User Interface The interface through which the user interacts with all aspects of the program.

UDP - User Datagram Protocol UDP is a simple protocol that exchanges datagrams without ac-knowledgments or guaranteed delivery, requiring that error processing and retransmission be handled by other protocols. UDP is defined in RFC 768. UDP is limited to a packet size of 65507 bytes.