**COMSATS University Islamabad,   
Park Road, Chak Shahzad, Islamabad Pakistan**

SOFTWARE DESIGN DESCRIPTION   
(SDD DOCUMENT)

for

**<PROJECT NAME>**  
Version 1.0

***By***

**Student Name 1 CIIT/SP09-BCS-xxx/ISB**

**Student Name 2 CIIT/SP09-BCS-xxx/ISB**

***Supervisor*Supervisor Name**

*Bachelor of Science in Computer Science (20xx-20xx)*

**Table of Contents**

Revision History iii

1. Introduction 1

2. Design Methodology and software process model 1

3. System Overview 1

3.1 Architectural Design 1

3.2 Process Flow/Representation 1

4. Design Models [along with descriptions] 1

5. Data Design 2

5.1 Data Dictionary 2

6. Algorithm & Implementation 2

7. Software Requirements Traceability Matrix 2

8. Human Interface Design 3

8.1 Screen Images 3

8.2 Screen Objects and Actions 3

9. Appendix I 3

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason for changes** | **Version** |
|  |  |  |  |
|  |  |  |  |

**Application Evaluation History**

|  |  |
| --- | --- |
| **Comments (by committee)**  **\*include the ones given at scope time both in doc and presentation** | **Action Taken** |
|  |  |
|  |  |

Supervised by

<Supervisor’s Name>

Signature\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Introduction

Briefly explain scope of the project covered till now including modules.

# Design methodology and software process model

Explain and justify the choice of design methodology being followed. (OOP or procedural). Also explain which process model are you following and why.

# System overview

Give a general description of the functionality, context and design of your project. Provide any background information if necessary.

## Architectural design

Develop a modular program structure and explain the relationships between the modules to achieve the complete functionality of the system. This is a high-level overview of how the system’s modules collaborate with each other in order to achieve the desired functionality.

Don’t go into too much detail about the individual subsystems. The main purpose is to gain a general understanding of how and why the system was decomposed, and how the individual parts work together.

Provide a diagram showing the major subsystems and their connections. **Use a simple Line-Box-Diagram for simpler systems and detailed diagrams (MVC, Client-Server, Layered, Multi-tiered) for complex systems.**

## Process flow/Representation

Provide a representation of the flow of **MAJOR processes** of your system in the form of an activity diagram. **DO NOT CREATE ACTIVITY DIAGRAMS FOR LOGIN OR SIGN-UP UNLESS THEY INVOLVE SIGNIFICANT COMPLEXITY**. Include only the major processes.

# Design models [along with descriptions]

**The applicable models may include:**

* Class Diagram
* Sequence Diagram
* State Transition Diagram
* Data Flow Diagram
* Schematic diagram (Hardware projects only)
* Timing diagram (Hardware projects only)

Insert ***applicable*** system models here.

You should be clear about all the concepts used in your diagrams for example for class diagram you should know about aggregation, composition, inheritance/generalization. Also ensure visibility of all diagrams.

Class diagram and associated models shall only be necessary for object oriented approach. In case of procedural, create a DFD. Data flow diagram should be extended to 2-3 levels. It should clearly list all processes, their sources/sinks and data stores.

**Note: System design should be complete in all aspects. Create any/all diagrams if you need to. A DFD can also be supplemented by a State Transition Diagram depending on the nature of the project.**

**Hardware projects can include Schematic diagram, System block diagram, timing diagram, Flow charts as replacement of sequence diagram/ Data flow diagram AFTER CONSULTATION WITH THEIR SUPERVISORS. Choice of models must be properly justified.**

# Data design

**{**

**{**

**“$schema” : “ “ ,**

**“$id” : ,**

**“Title” : “Users” ,**

**“description” : “all the accounts created”,**

**“Type” : “object”,**

**“Properties” :**

**{**

“Username” : {

“Description” : “name of the user”,

“Type” : “String”

}

“Email” : {

“Description” : “email of the user”,

“Type” : “String”

}

“Password” : {

“Description” : “password for the authentication”,

“Type” : “varchar”

}

“Contact” : {

“Description” : “phone number of the user”,

“Type” : “num”

}

“profileImg” : {

“Description” : “picture of the user”,

“Type” : “Jpg , png”

}

**},**

**“Required” : [“username”, “email”, “password” , “contact” ]**

**}**

**{**

**“$schema” : ,**

**“$id” : ,**

**“Title” : “Admin” ,**

**“description” : “all the accounts of admins”,**

**“Type” : “object”,**

**“Properties” :{**

“AdminID” : {

“Description” : “Id assigned by the system for admin access”,

“Type” : “String”

}

“AdminPass” : {

“Description” : “password for admin authentication”,

“Type” : “varchar”

}

**},**

**“Required” : [“adminID”, “adminPass” ]**

**}**

**{**

**“$schema” : “ “ ,**

**“$id” : ,**

**“Title” : “Games” ,**

**“description” : “all the games in the application”,**

**“Type” : “object”,**

**“Properties” :{**

“gameName” : {

“Description” : “name of the game”,

“Type” : “string”

}

“gameID” : {

“Description” : “random ID assigned to the game”,

“Type” : “num”

}

**},**

**“Required” : [“gameName”, “gameID” ]**

**}**

**{**

**“$schema” : “ “ ,**

**“$id” : ,**

**“Title” : “Scores” ,**

**“description” : “scores recorded of all the users ”,**

**“Type” : “object”,**

**“Properties” :**

**{**

**“**Username” : {

“Description” : “name of the user”,

“Type” : “string”

}

“gameID” : {

“Description” : “ID of the game ”,

“Type” : “num”

}

“Score” : {

“Description” : “score of the user ”,

“Type” : “num”

}

**},**

**“Required” : [“username”, “gameID” , “score” ]**

**}**

**{**

**“$schema” : “ “ ,**

**“$id” : ,**

**“Title” : “Recordings” ,**

**“description” : “all the recording files”,**

**“Type” : “object”,**

**“Properties” :{**

“userName” : {

“Description” : “name of the user”,

“Type” : “string”

}

“recordingID : {

“Description” : “random ID assigned to the file by system”,

“Type” : “num”

}

“recordingLink” : {

“Description” : “link of the file uploaded in the database storage”,

“Type” : “string”

}

**},**

**“Required” : [“username”, “recordingID” , “recordingLink” ]**

**}**

**}**

## Data dictionary

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **FieldName** | **DataType** | **DataFormat** | **FieldSize** | **Description** | **Example** |
| username | string |  |  | Full name of the user | “Kinza arshad” |
| email | string |  |  | Email of the user | “maida@gmail” |
| password | VarChar |  |  | Password for user authentication | “mustafa” |
| contact | num |  |  | Phone number of the user | “03321576652” |
| profileImg | String |  |  | Link of the profile uploaded on the database storage |  |
| adminId | String |  |  | Id assigned by the developers to the admins | “kinza@gmail” |
| adminPass | VarChar |  |  | Admin password assigned to adminID for authentication | 13718847262” |
| gameName | String |  |  | Name of all the games in the application | “ball\_jump” |
| gameID | num |  |  | Game ID assigned to the game to identify it | “01’ |
| score | num |  |  | Score to keep track of progress of the users | “20” |
| recordingID | num |  |  | RandomID assigned by the system to the recording file | “0318487101” |
| recordingLink | String |  |  | link of the file  In the storage |  |

# Algorithm & Implementation

**KNN:**

Classify(X,Y,x)

X= training data

Y= class labels of X

x= unknown sample

For i =1 to m

Compute Distance d(Xi , x)

Compute set I containing indices for the k smallest distances d(Xi , x)

Return majority label for {Yi where i belongs to I)

**DWT:**

Public static int[ ] discreteWaveletTransform( int[ ] input){

//this function assumes that input.length= 2^n , n>1

Int[ ] output = new int[ input.length ];

For (int length = input.length / 2 & length= length/2){

//length is the current length of the working area of the output array

//length starts at half of the array size and every iteration is halved until it is 1

For (int i=0 ; i<length ; ++i){

Int sum= input[ i\*2 ] + input[ i\*2+1 ];

Int difference= input[ i\*2 ] - input[ i\*2+1 ];

Output[ i ]= sum;

Output[ length+i ]= difference;

}

if(length == 1){

Return output;

}

system.arraycopy(output,0,input , 0, length);

}

}

For i in range (X):

coeffs= discreteWaveletTransform(X)

cA1 , cD1= coeffs

coeffs2= discreteWaveletTransform( cA1)

cA2, cD2 = coeffs2

coeffs3= discreteWaveletTransform( cA2 )

cA3 , cD3= coeffs3

coeffs4= discreteWaveletTransform(X)

cA4 , cD4= coeffs4

coeffs5= discreteWaveletTransform( cA4 )

cA5 cD5= coeffs5

For j in range(16):

Processed [ i ] [ j ] [ 0 ] = cA5[ j ]

Processed [ i ] [ j ] [ 1 ] = cD1[ j ]

Processed [ i ] [ j ] [ 2 ] = cD2[ j ]

Processed [ i ] [ j ] [ 3 ] = cD3[ j ]

Processed [ i ] [ j ] [ 4 ] = cD4[ j ]

Processed [ i ] [ j ] [ 5 ] = cD5[ j ]

**GAME:**

//updata is called once per frame

Void update(){

readData();

//makePieces();

for( int i=0 ; i<Input.touchCount ; i++){

if(Input.GetTouch(i).phase == TouchPhase.Began){

//construct a ray from current touch coordinates

transform.Translate( 0, 2, 0);

}

}

}

Void readData(){

//read data from the port

}

Int makePieces(){

// make the pieces in to 500 rows to make small samples

Int r= callModel(tempArray);

Return r;

}

Int callModel(Array tempArray){

//call the model and get input

Return 1;

}

**PDFViewer:**

//Declare buttons

//open default ACTION\_GET\_CONTENT from android to select pdf

//create chooser

//get result code and check if it is OK

//load pdf

**Sign-in:**

//initialize the buttons

//initialize Paper(remembers username and password) library

//set up the onClicks on buttons

//get the text from the EditTexts

//check if the information user entered is null

//remember the username and password on Paper

//initialize firebase database

//check if the table Users exists

//check if the email exists

//check if the password is correct

**Sign-up :**

//initialize the buttons

//initialize Paper(remembers username and password) library

//set up the onClicks on buttons

//get the text from the EditTexts

//check if the information user entered is null

//initialize firebase database

//check if the table Users exists

//check if the email exists

//start the default ACTION\_GET\_CONTENT for GalleryPick

//if pic upload is successful then upload it to database storage

//create a HashMap of all the data

//upload the data on database

//If upload is successful then start activity login

//remember the username and password on Paper

# Software requirements traceability matrix

This section should contain a table that summarises how each software requirement has been met in this document. The tabular format permits one-to-one and one-to-many relationships to be shown.

Table 1 Requirements Traceability Matrix

|  |  |  |  |
| --- | --- | --- | --- |
| **Req. Number** | **Ref. Item** | **Design Component** | **Component Items** |
| FR01 | Class Diagram | ClassName | FunctionName(s) |
| FR01 |  |  |  |
|  |  |  |  |

# Human interface design

**8.2 Screen images**

## 8.2 Screen objects and actions

**registerActivity:**

This activity has inputs in the from of EditTexts and ImageView .It allows you to pick an image from the gallery and add name, email, password, contact . when sign-up is pressed a new user is created in the database and login activity is opened. If the user already has an account he/she can click sign-in button and go back to sign-in activity.

**sign-in Activity:**

This activity has inputs in the form of EditTexts to enter an already existing account. When the user presses sign-in , the system authenticates the username and password from the database and if the authentication is successful takes the user to the Dashboard . If the user doesn’t have an account he/she can click sign-up and go to the sign-up page to register.

**Dashboard Activity:**

This activity has two ImageView buttons that take us to the reading exercise and the gaming exercise which are the core features of our application .The headset icon on the top right shows if the headset is connected or not . The buttons on the bottom are User Analytics , Headset Settings , Log-out . The User Analytics button takes us to an activity which lets us view the data in different formats . The Headset Settings opens an activity that lets us see the connection status and signal strength with our headset. The logout buttons logs the system out and deletes data from the paper.

**Drawer Activity:**

This activity has all the Profile information . It gets all the data from from the firebase database in realtime against the username that is logged-in.

**Bookshelf Activity:**

This activity has one button that calls the default choose file action to let the user choose a .pdf file.

**PDFViewer Activity:**

This activity loads the pdf from the page one and lets the user scroll the pdf file .

**GameList Activity:**

This activity has the Icons of all the games in the listformat . on click the icon takes the user to a unity activity so he/she can play the game.

**Game:**

This activity has the game with a ball that can jump.

**Headset Settings:**

This activity shows the connection status and signal strength . It also provides with two buttons connect and disconnect from the headset.

# Appendix I

* How to design using UML (OOP): For guidance please follow the instructions mentioned in the link: http://agilemodeling.com/artifacts/
* How and when to design ER diagrams: For guidance please follow the instructions mentioned in the link:

<http://people.inf.elte.hu/nikovits/DB2/Ullman_The_Complete_Book.pdf>

* Data flow diagrams: For guidance please follow the instructions mentioned in the link and book:
  + http://www.agilemodeling.com/artifacts/dataFlowDiagram.htm
  + Software Engineering –A Practitioner’s approach by Roger Pressman
* Architecture diagram: For guidance please follow the instructions mentioned in the link and book:
  + Ian Sommerville – Software Engineering 9th Edition– Chapter 6