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

SOFTWARE DESIGN DESCRIPTION

for

Atom – Brain-Computer Interfacing using Electroencephalography

Version 1.0

By

Kinza Arshad **CIIT/FA16-BCS-108/ISB**

Muhammad Faizan Badar **CIIT/FA16-BCS-054/ISB**

Supervisor

Dr. Yasir Faheem

Bachelor of Science in Computer Science (2016-2020)

Contents

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 2
4. Design models 6
4.1 Use case models 6
4.2 Sequence Diagrams 10
4.3 Class Diagram 14
5. Data design 15
5.1 Data dictionary 18
6. Algorithm & Implementation 19
7. Software requirements traceability matrix 22
8. Human interface design 26
8.2 Screen objects and actions 26

Figures

Figure 1 Architectural Diagram for Atom as MVC 1
Figure 2 Activity Diagram for Account Handling module 2
Figure 3 Activity Diagram for User Analytics and Statistics 3
Figure 4 Activity Diagram for Specialized Control Training 4
Figure 5 Activity Diagram for Entertainment Incentivized Training 5
Figure 6: Use case model for Account Handling module 6
Figure 7 Use case model for User Analytics and Statistics module 7
Figure 8 Use case model for Specialized Control Training module 7
Figure 9 Use case model for Entertainment Incentivized Training module 8
Figure 10 Use case model for EEG Feature Extraction module 8
Figure 11 Use case model for Database Handling module 9
Figure 12 Sequence Diagram for Preliminary use cases 10
Figure 13 Sequence Diagram for User Analytics and Statistics use cases 11
Figure 14 Sequence Diagram for Specialized Control Training use cases 12
Figure 15 Sequence Diagram for Entertainment Incentivized Training use cases 13
Figure 16 Class Diagram for Atom 14
Figure 17 Screenshot from Atom for Register Activity 26
Figure 18 Screenshot from Atom for Dashboard Activity 28
Figure 19 Screenshot from Atom for Drawer Activity 29
Figure 20 Screenshot from Atom for Bookshelf Activity 30
Figure 21 Screenshot from Atom for Book Reader Activity 31
Figure 22 Screenshot from Atom for Game List Activity 32
Figure 23 Screenshot from Atom for Game Activity 33
Figure 24 Screenshot from Atom for Headset Settings Activity 34

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

Dr. Yasir Faheem

Signature_____

1. Introduction

Our project deals with the domain of Brain-Computer interface and Cognitive Electrophysiology. As the name suggests it uses brain to give input and reads its input by tapping into the electric mode pf communication that our neuron use to communicate all the thoughts and functionalities, we are able to perform. The field is growing, and the possibilities are endless. The fundamental idea is to use this EEG incorporated BCI to target issues relating to human activities, specifically enhancing the attention span to improve focus in daily activities such as reading and others with similar brain involvement. The methodology we've chosen to achieve said claim can be divided into two streams; entertainment incentivized training and specialized controlled training, achieved by mini-games and a book reader respectively, where-in both these utilities are taken use of by the BCI to be built

This document will specify the hardware and the software aspects while also discussing the compatibility of different platforms and the integration of different modules that come together to make the whole project.

2. 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.

3. System overview

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

3.1 Architectural design

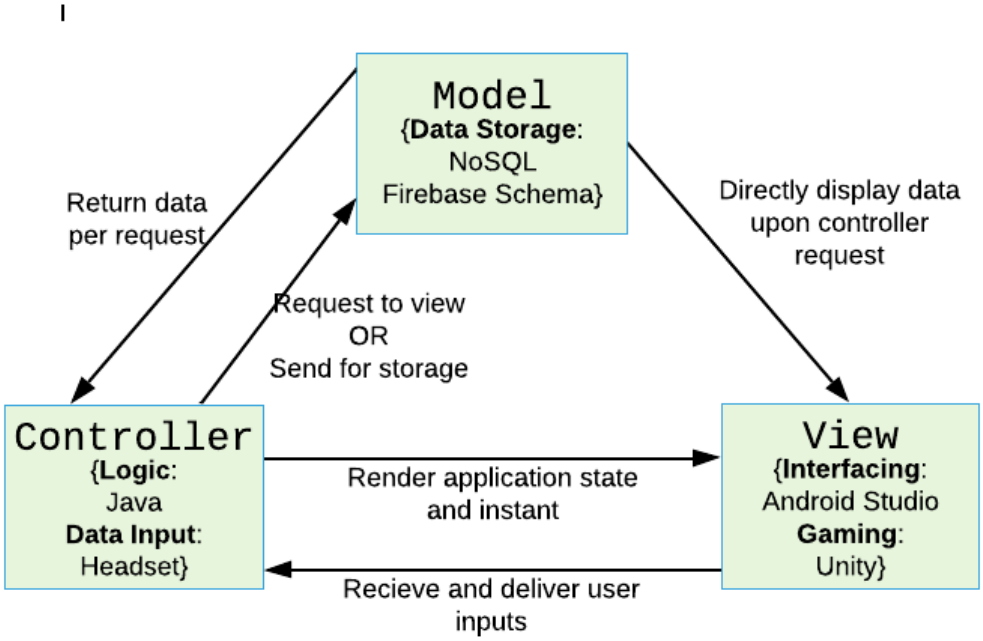

Figure 1 Architectural Diagram for Atom as MVC

3.2 Process flow/Representation

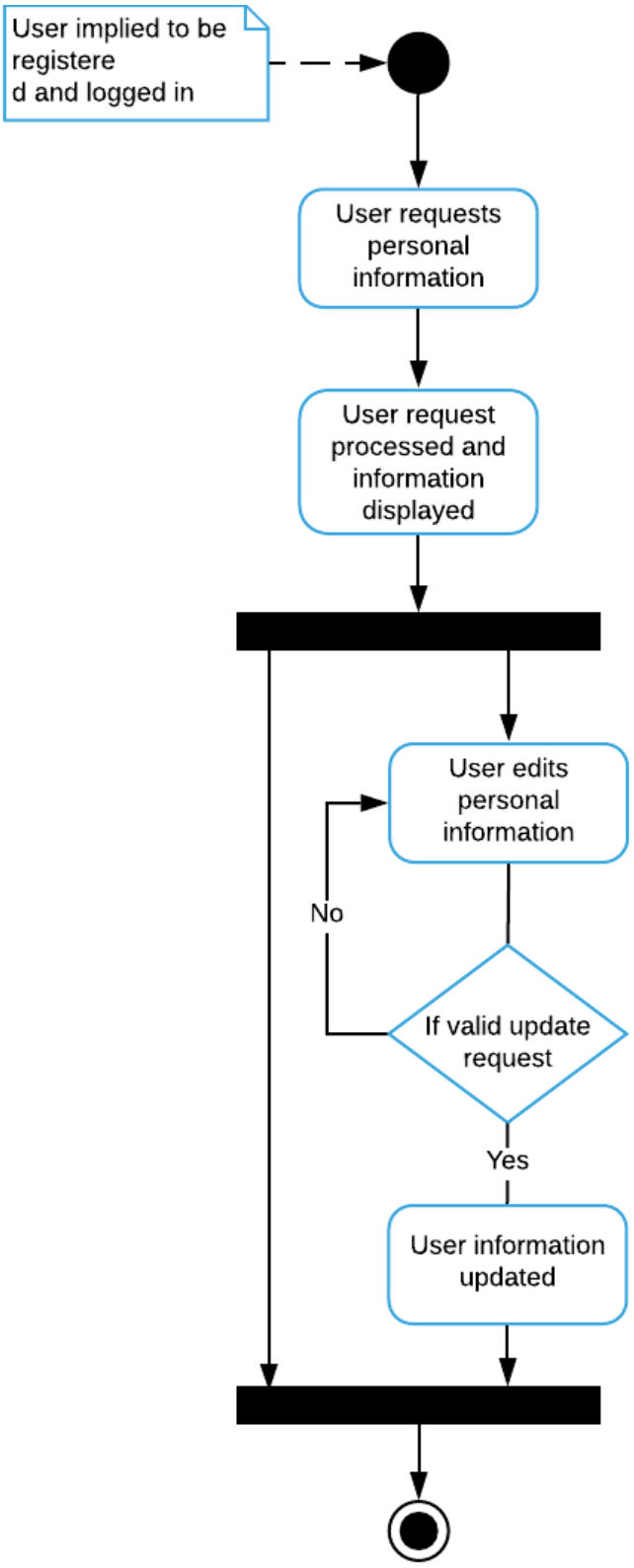

Figure 2 Activity Diagram for Account Handling module

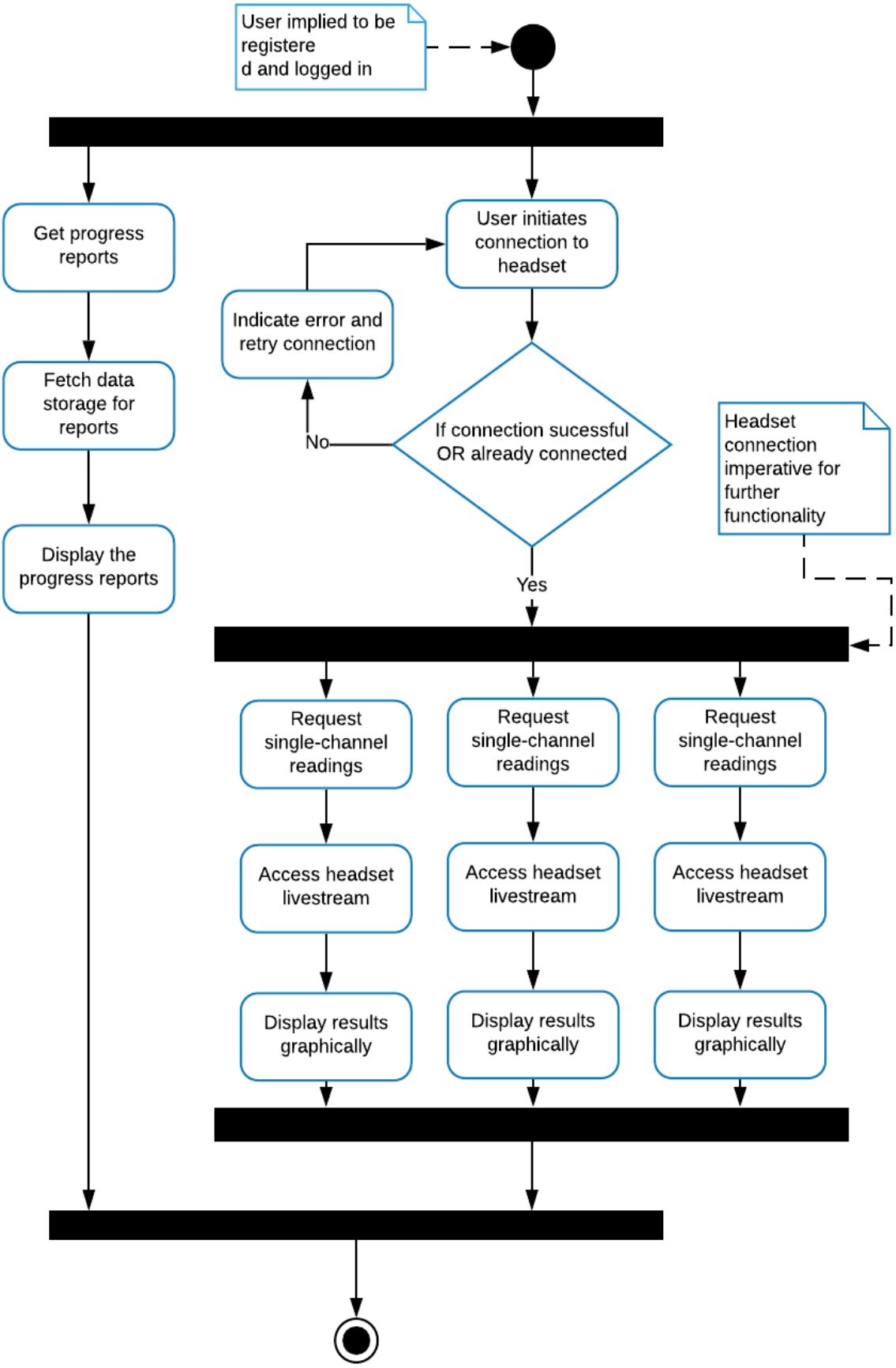

Figure 3 Activity Diagram for User Analytics and Statistics

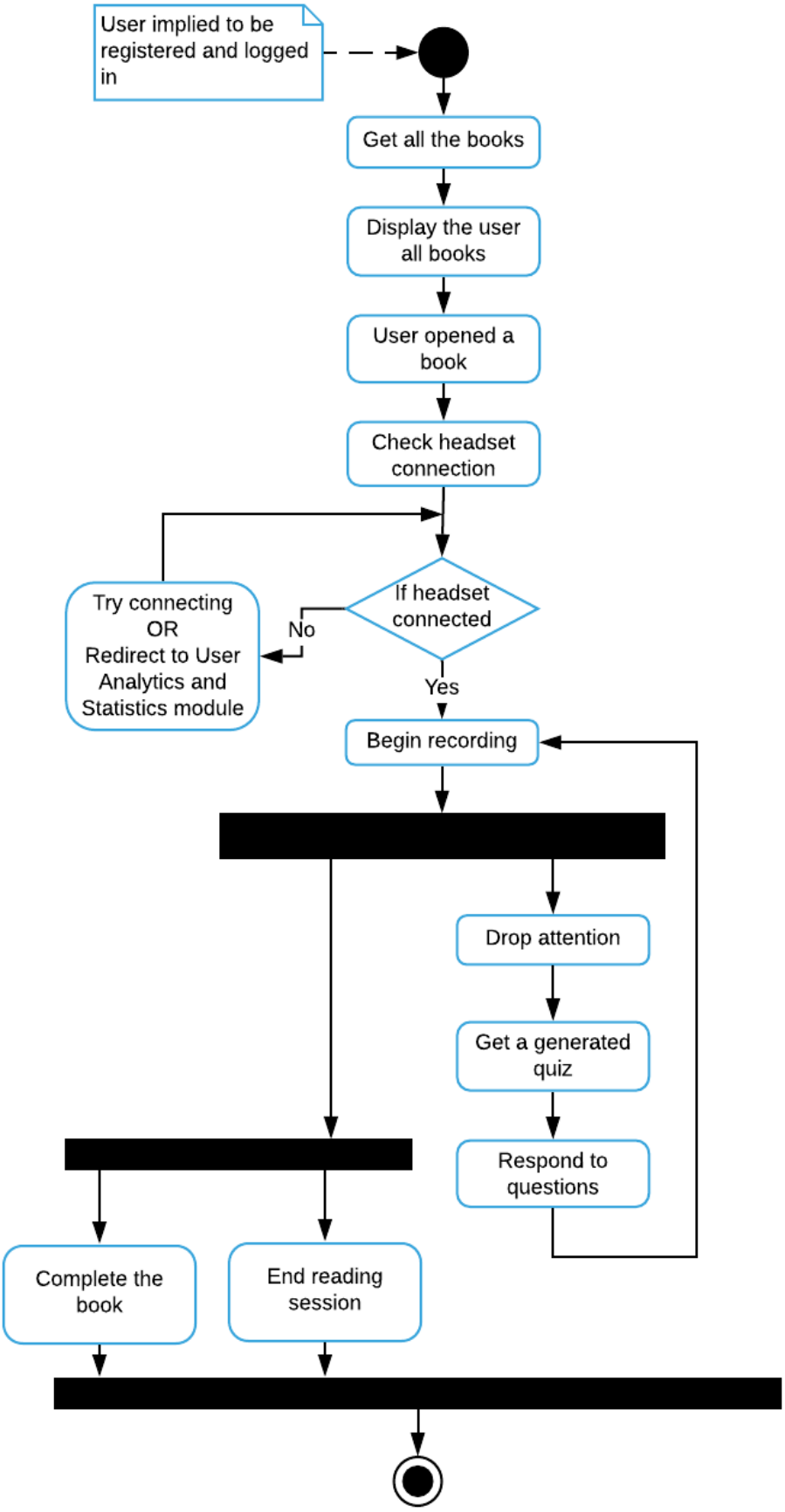

Figure 4 Activity Diagram for Specialized Control Training

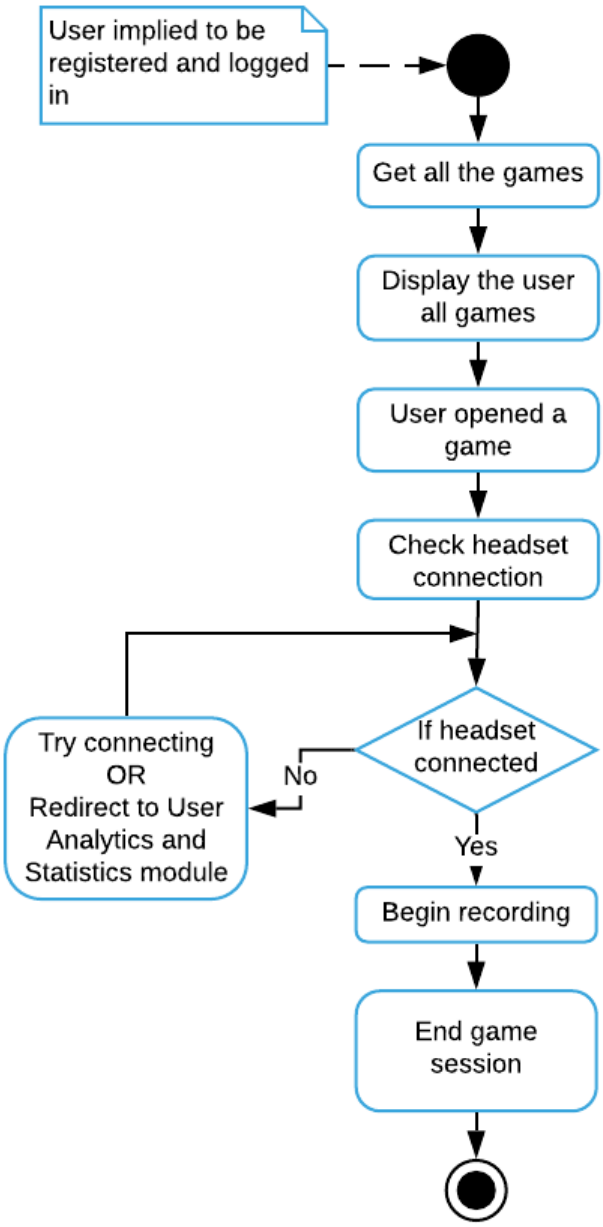

Figure 5 Activity Diagram for Entertainment Incentivized Training

4. Design models

4.1 Use case models

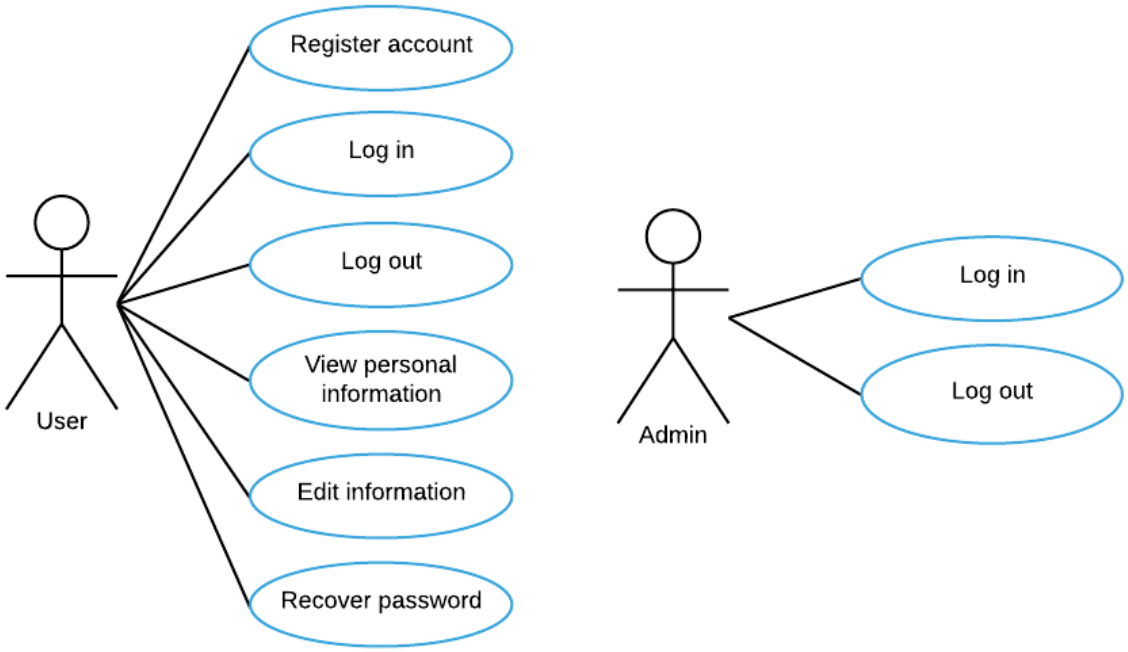

Figure 6: Use case model for Account Handling module

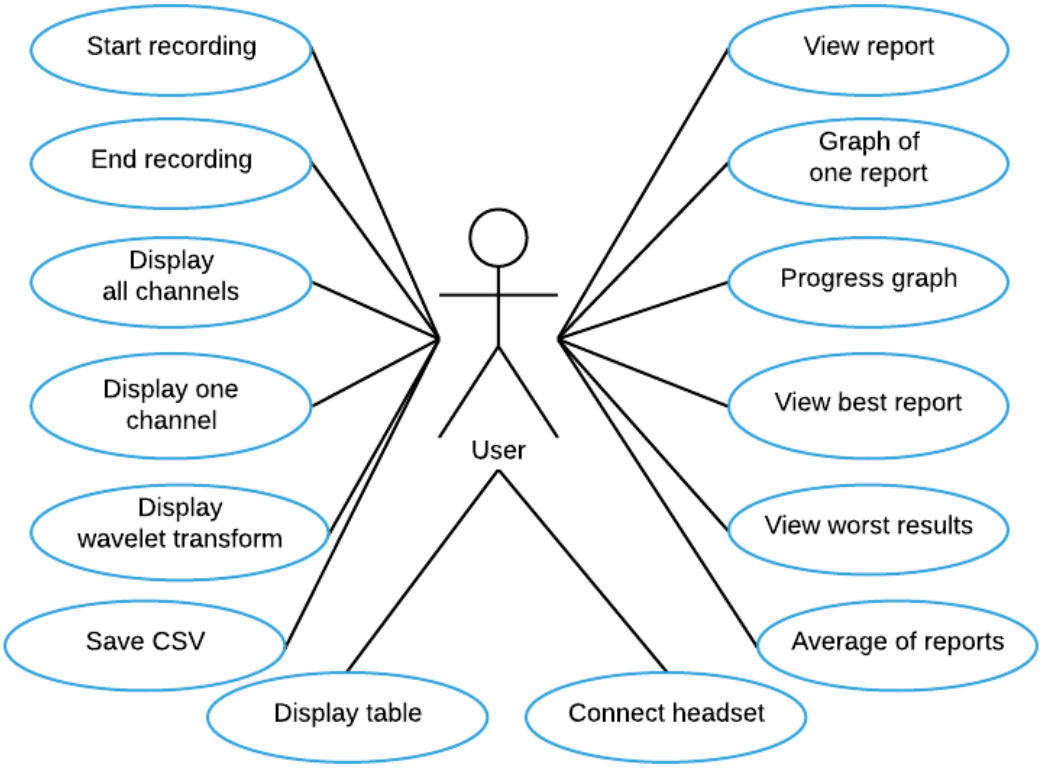

Figure 7 Use case model for User Analytics and Statistics module

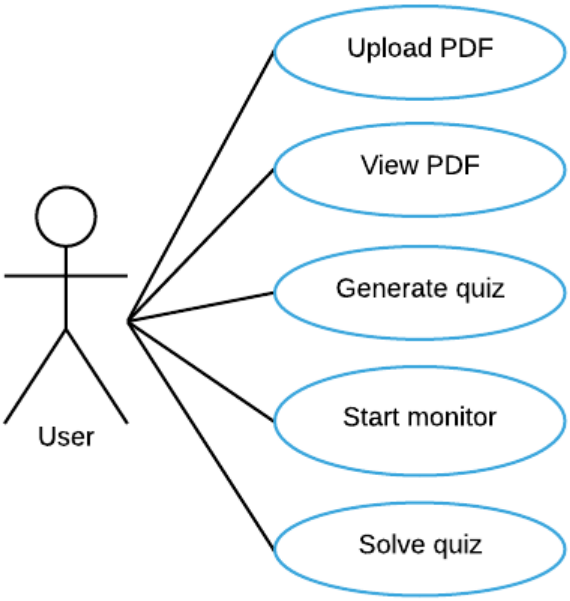

Figure 8 Use case model for Specialized Control Training module

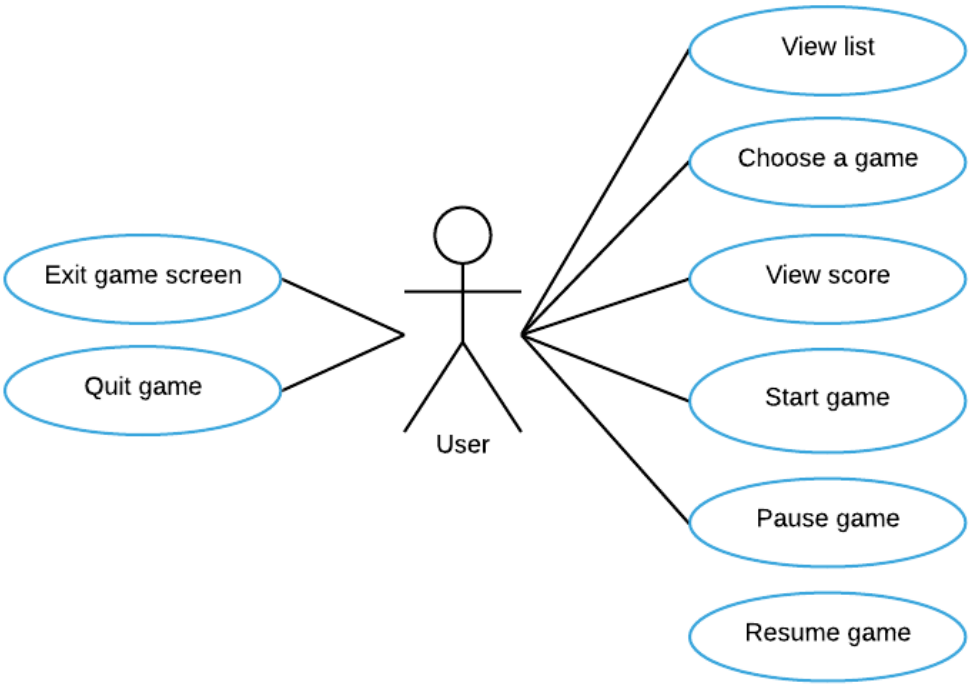

Figure 9 Use case model for Entertainment Incentivized Training module

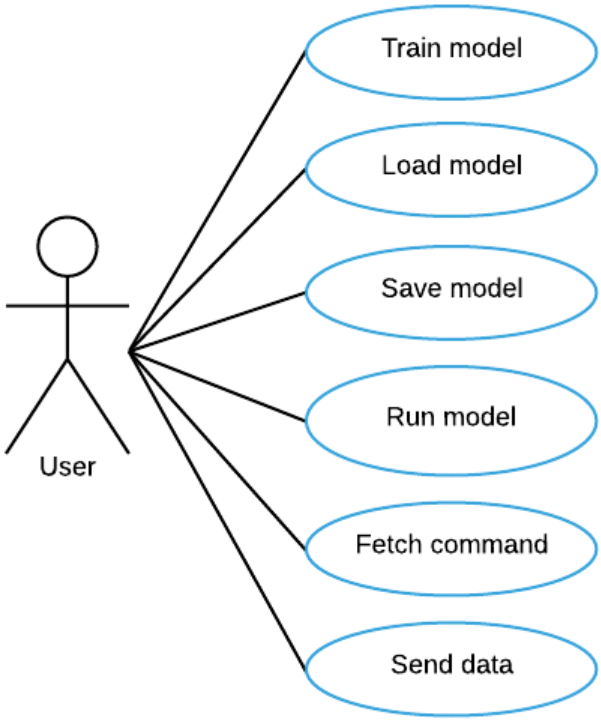

Figure 10 Use case model for EEG Feature Extraction module

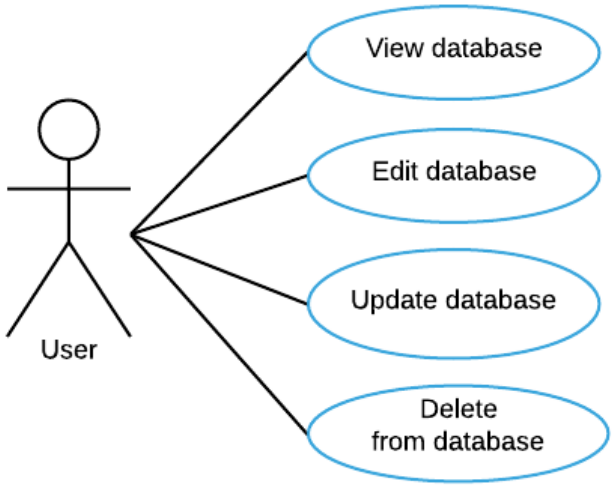

Figure 11 Use case model for Database Handling module

4.2 Sequence Diagrams

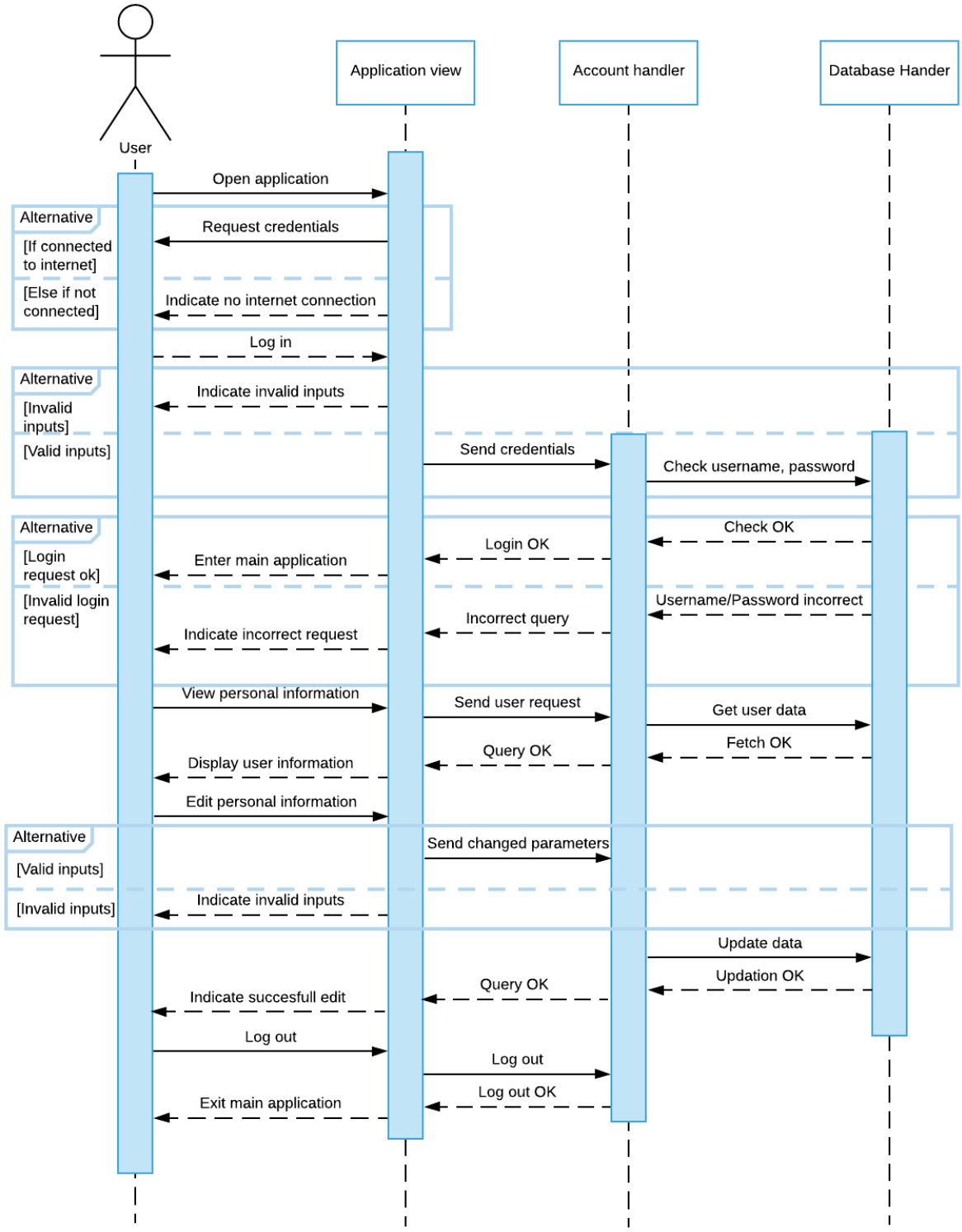

Figure 12 Sequence Diagram for Preliminary use cases

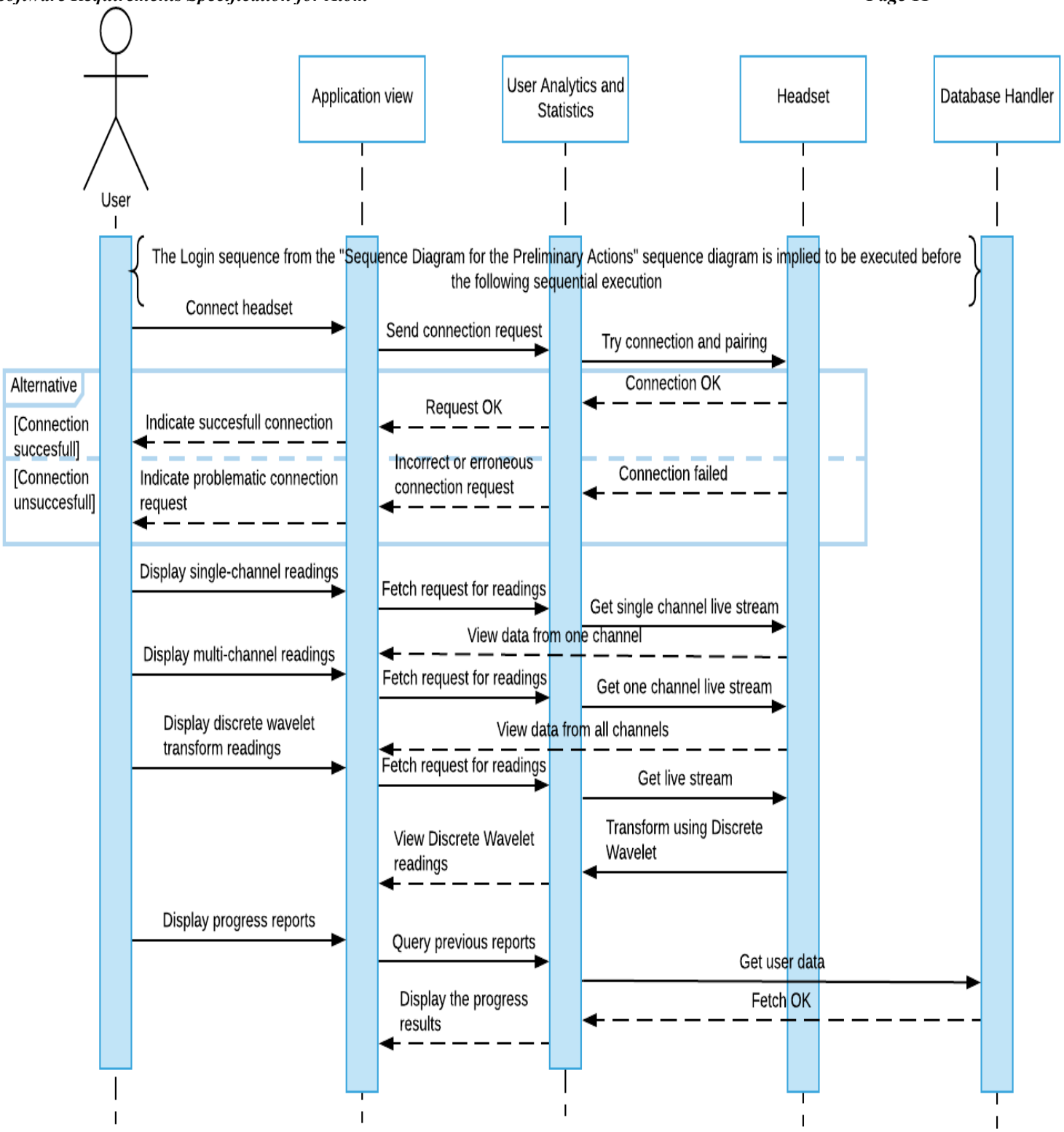

Figure 13 Sequence Diagram for User Analytics and Statistics use cases

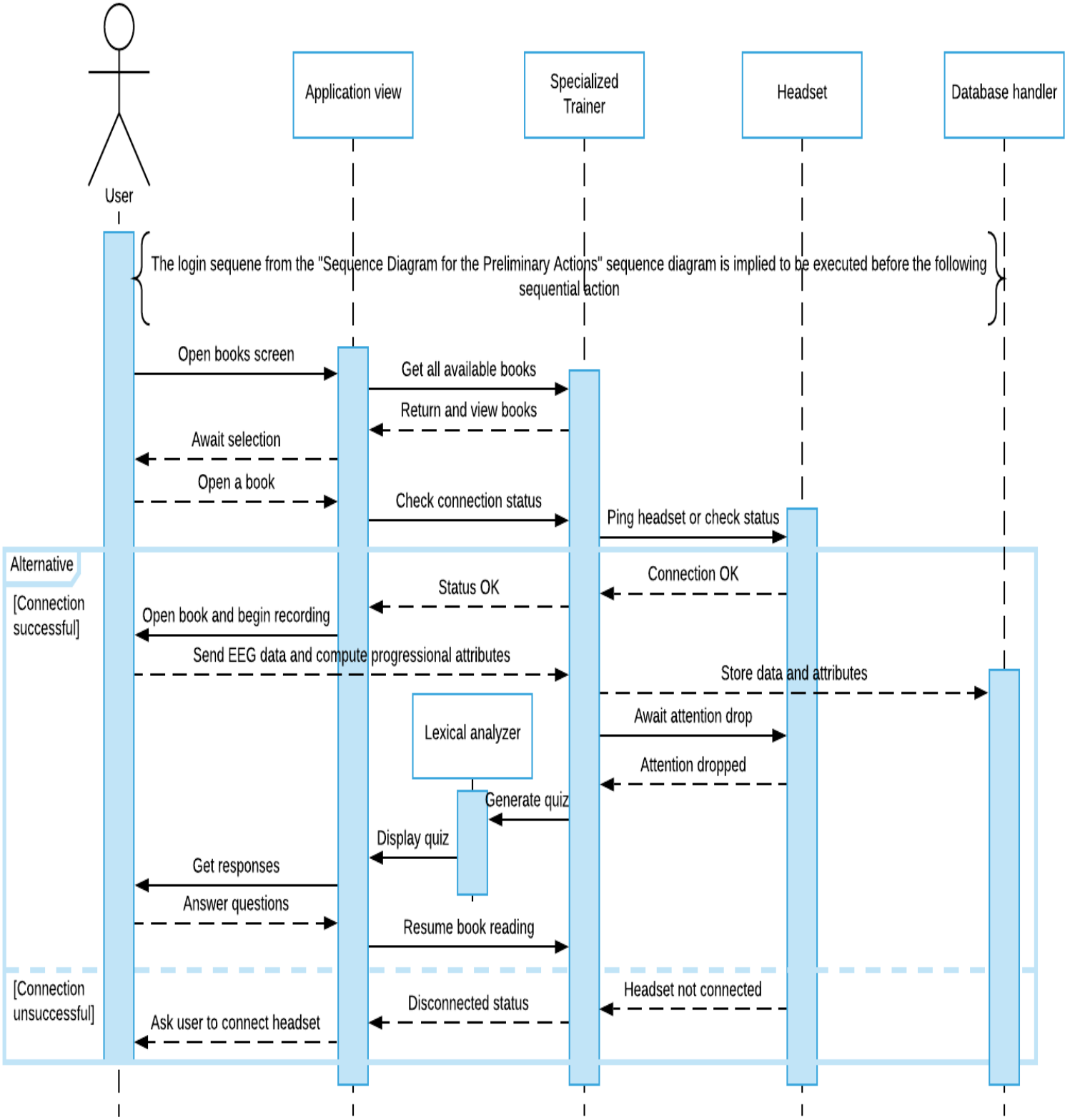

Figure 14 Sequence Diagram for Specialized Control Training use cases

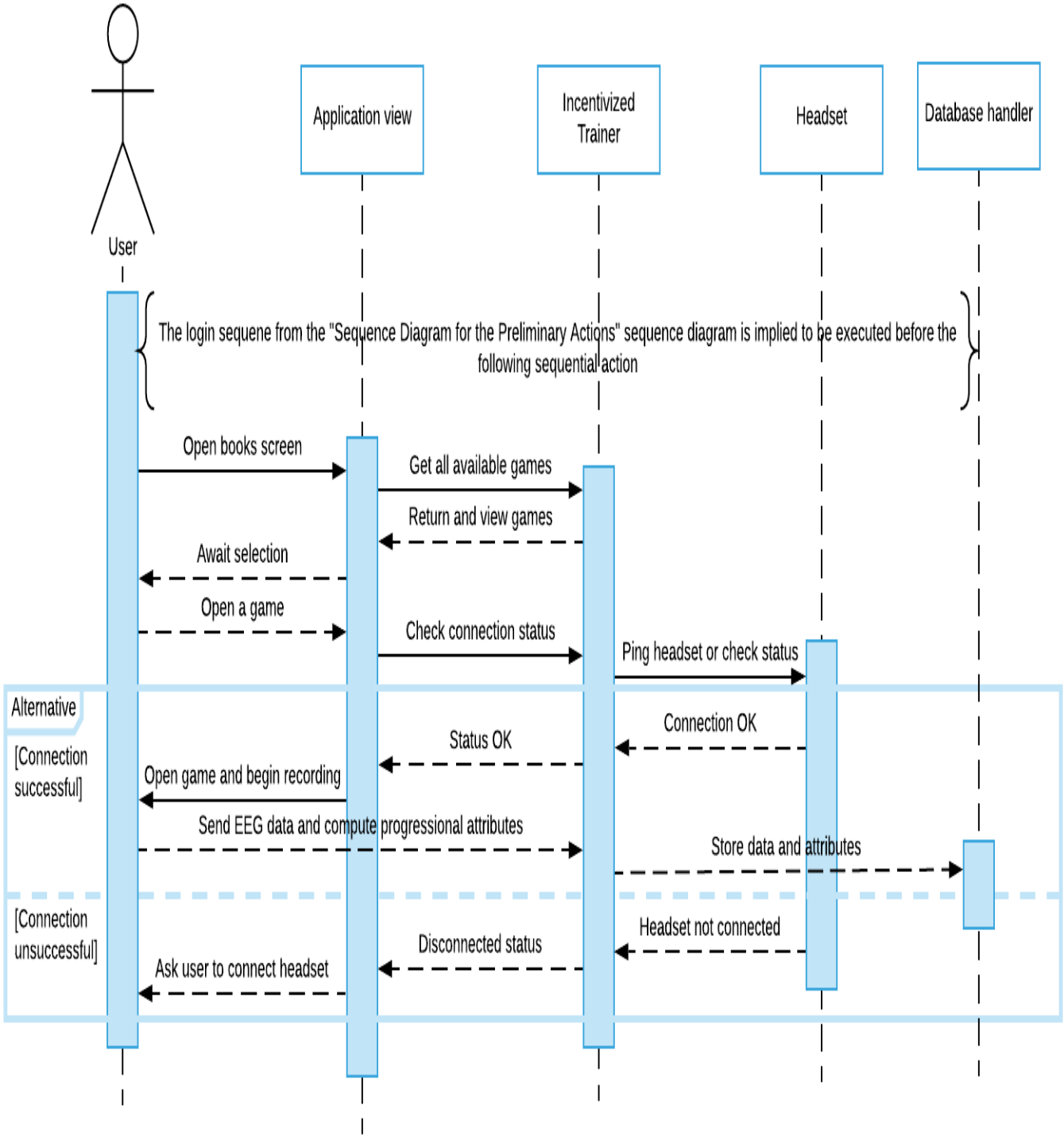

Figure 15 Sequence Diagram for Entertainment Incentivized Training use cases

4.3 Class Diagram

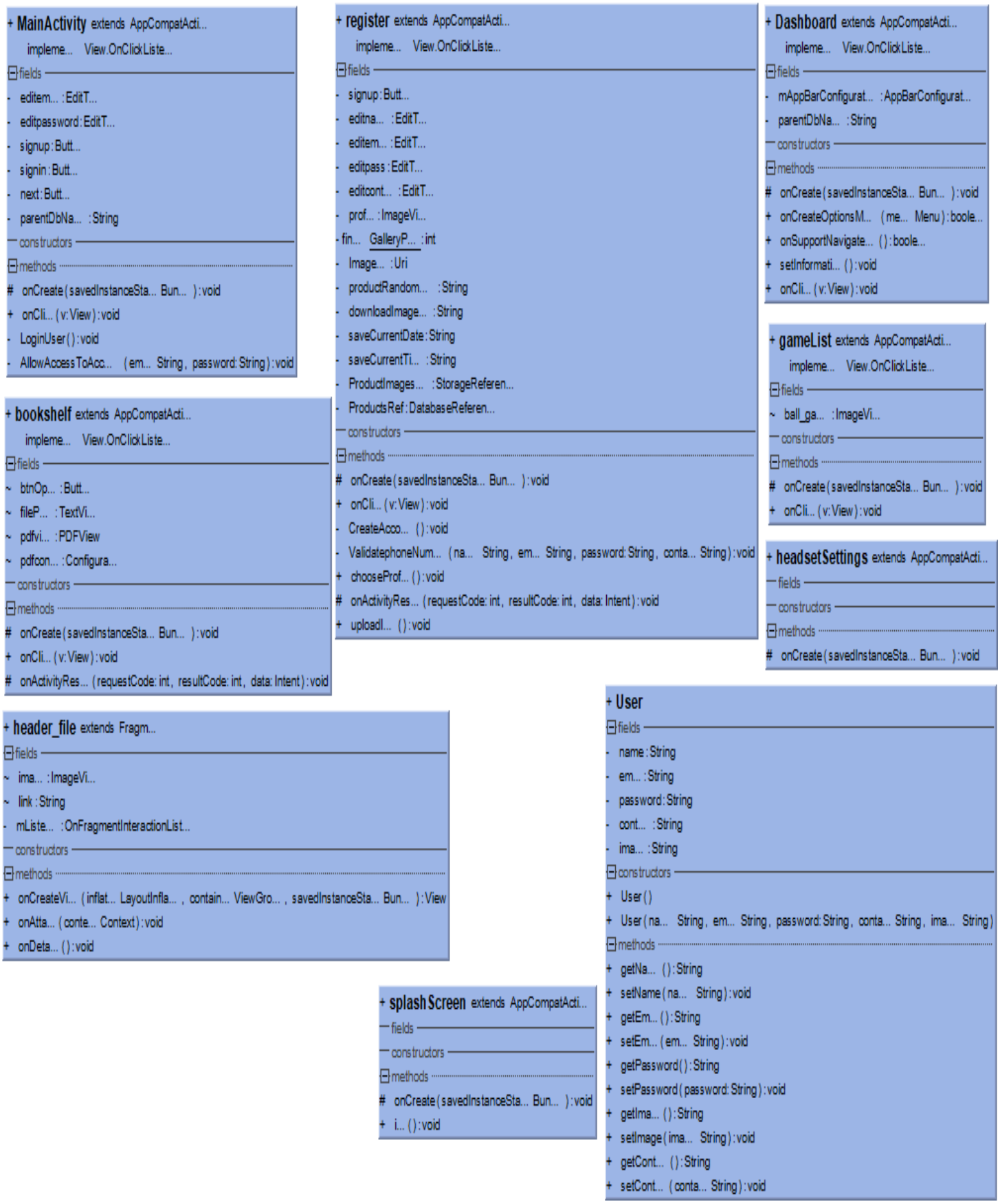

Figure 16 Class Diagram for Atom

5. 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” ]
      }
}
```

5.1 Data dictionary

FieldName	DataType	DataFormat	FieldSize	Description	Example
username	string	text	Not specified	Full name of the user	“Kinza arshad”
email	string	text	Not specified	Email of the user	“maida@gmail”
password	VarChar	text	Not specified	Password for user authentication	“mustafa”
contact	num	text	Not specified	Phone number of the user	“03321576652”
profileImg	String	text	Not specified	Link of the profile uploaded on the database storage	“ “
adminId	String	text	Not specified	Id assigned by the developers to the admins	“kinza@gmail”
adminPass	VarChar	text	Not specified	Admin password assigned to adminID for authentication	13718847262”
gameName	String	text	Not specified	Name of all the games in the application	“ball_jump”
gameID	num	text	Not specified	Game ID assigned to the game to identify it	“01’
score	num	text	Not specified	Score to keep track of progress of the users	“20”
recordingID	num	text	Not specified	RandomID assigned by the system to the recording file	“0318487101”
recordingLink	String	text	Not specified	link of the file In the storage	“ “

6. 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
```

7. 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
FR0X	**Class Diagram**	**ClassName**	**FunctionName(s)**
FR01	**Sign-in**	**register**	**onClick()**
FR02	**Sign-up**	**register**	**createAccount()** **validateInformation()**
FR03	**Name**	**register**	**oncreate()** **createAccount()**
FR04	**Email**	**register**	**oncreate()** **createAccount()**
FR05	**Password**	**register**	**oncreate()** **createAccount()**
FR06	**Contact**	**register**	**oncreate()** **createAccount()**
	Profile	**register**	**oncreate()** **uploadImg()**
FR07	**Database Input**	**register**	**validateInformation()**
FR08	**Success-Register**	**register**	**validateInformation()**
FR09	**Failure-Register**	**register**	**validateInformation()**
FR10	**Logout**	**HomeFragment**	**onCreate()** **onClick()**
FR11	**Edit**	**header_file**	**onCreate()** **onClick()**
FR12	**Name-Edit**	**edit**	**onCreate()**
FR13	**Email-Edit**	**edit**	**onCreate()**
FR14	**Password-Edit**	**edit**	**onCreate()**
FR15	**Contact-Edit**	**edit**	**onCreate()**
FR16	**Confirm-Edit**	**edit**	**onCreate()**
FR17	**Database-update**	**edit**	**onCreate()** **editInformation()**
FR18	**Success-Edit**	**edit**	**onCreate()** **editInformation()**
FR19	**Failure-Edit**	**edit**	**onCreate()** **editInformation()**
FR20	**View**	**HomeFragment**	**onCreate()** **onClick()**
FR21	**List-View**	**userAnalytics**	**onCrete()** **onClick()**
FR22	**View-Button**	**userAnalytics**	**onCreate()** **onClick()**

FR23	**Display-Reports**	**userAnalytics**	**onCreate() onClick() displayReports()**
FR24	**Sign-in**	**MainActivity**	**onCreate() onClick()**
FR25	**Sign-up**	**MainActivity**	**onCreate() onClick()**
FR26	**Email**	**MainActivity**	**onCreate()**
FR27	**Password**	**MainActivity**	**onCreate()**
FR28	**Success-Login**	**MainActivity**	**onCreate() LoginUser() AllowAccessToAccount()**
FR29	**Failure-Login**	**MainActivity**	**onCreate() LoginUser() AllowAccessToAccount()**
FR30	**Slide profile**	**Dashboard()**	**onCreate() setInformation()**
FR31	**Forgot password**	**MainActivity**	**onCreate() onClick()**
FR32	**Recovery email**	**recovery**	**onCreate()**
FR33	**New password**	**recovery**	**onCreate()**
FR34	**Confirm-password**	**recovery**	**onCreate() generateNewPassword()**
FR35	**graph**	**userAnalytics**	**GenerateGraph()**
FR36	**Progress graph**	**userAnalytics**	**generateProgressGraph()**
FR37	**Display-EEG**	**userAnalytics**	**displayEEG()**
FR38	**View-worst**	**reports**	**onCreate() onClick()**
FR39	**Average-reports**	**reports**	**onCreate() onClick()**
FR40	**Display-Channels**	**userAnalytics**	**onCreate() onClick()**
FR41	**Channels**	**userAnalytics**	**displayChannels()**
FR42	**Start-Recording**	**recordingEEG**	**startRecord()**
FR43	**Pause-Recording**	**recordingEEG**	**pauseRecord()**
FR44	**End-Recording**	**recordingEEG**	**endRecord()**
FR45	**DWT**	**userAnalytics**	**onCreate() onClick()**
FR46	**Save-csv**	**recordingEEG**	**Save()**
FR47	**Success-save**	**recordingEEG**	**Save()**
FR48	**Failure-save**	**recordingEEG**	**Save()**
FR49	**Table**	**userAnalytics**	**generateTable()**
FR50	**Connect**	**headsetSettings**	**Connect()**
FR51	**Dis-connect**	**headsetSettings**	**Dis-connect()**
FR52	**Success-headset**	**headsetSettings**	**Connect()**
FR53	**Failure-connect**	**headsetSettings**	**Connect()**
FR54	**Start-exploring**	**bookshelf**	**onCreate() onClick()**

FR55	**Open-gallery**	**bookshelf**	**onCreate()** **onClick()**
FR56	**Click-file**	**bookshelf**	**onCreate()** **onClick()**
FR57	**Scrollable-pdf-display**	**bookshelf**	**onActivityResult()**
FR58	**Generate-quiz**	**bookshelf**	**onCreate()** **onClick()**
FR59	**Display-quiz**	**quiz**	**generateQuiz()**
FR60	**Monitor**	**bookshelf**	**onActivityResult()**
FR61	**Success-monitor**	**bookshelf**	**onActivityResult()**
FR62	**Failure-monitor**	**bookshelf**	**onActivityResult()**
FR63	**Solve-quiz**	**DisplayQuiz**	**onCreate()** **onClick()**
FR64	**Display-question**	**DisplayQuiz**	**onCreate()** **onClick()**
FR65	**Choose-quiz-option**	**DisplayQuiz**	**onCreate()** **onItemClickListener()**
FR66	**done**	**DisplayQuiz**	**onCreate()** **onClick()**
FR67	**Display-scores**	**DisplayQuiz**	**onCreate()** **onClick()** **displayScores()**
FR68	**Save-score**	**quiz**	**onCreate()** **onClick()** **saveScore()**
FR69	**View-list**	**HomeFragment**	**onCreate()** **onClick()**
FR70	**Display-gamelist**	**games**	**diaplayGames()**
FR71	**View-score**	**unityActivity**	**getLatestScore()**
FR72	**Display-gamescore**	**games**	**onCreate()** **onClick()** **getScores()**
FR73	**Pause-game**	**player**	**Update()** **Pause()**
FR74	**Start-game**	**player**	**Update()** **Start()**
FR75	**Touch-Input**	**unityActivity**	**Update()**
FR76	**Headset-Input**	**player**	**Update()** **getData()** **makePeices()**
FR77	**Resume-game**	**player**	**Update()** **Resume()**
FR78	**Quit-game**	**player**	**Update()** **Quit()**
FR79	**Exit-game-screen**	**player**	**Update()** **Back()**
FR80			

FR81	**Delete-database**	**adminHome**	**onItemClickListener() delete()**
FR82	**Success-database-delete**	**adminHome**	**onItemClickListener() delete()**
FR83	**Failure-database-delete**	**adminHome**	**onItemClickListener() delete()**
FR84	**View-database**	**adminHome**	**getData() display()**
FR85	**Failure-database-view**	**adminHome**	**getData() display()**
FR86	**Update-database**	**adminHome**	**onItemClickListener() update()**
FR87	**Success-database-update**	**adminHome**	**update()**
FR88	**Failure-database-update**	**adminHome**	**update()**
FR89	**Edit-database**	**adminHome**	**onItemClickListener() edit()**
FR90	**Success-database-edit**	**adminHome**	**Edit()e**
FR91	**Failure-database-edit**	**adminHome**	**Edit()**

8. Human interface design

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.

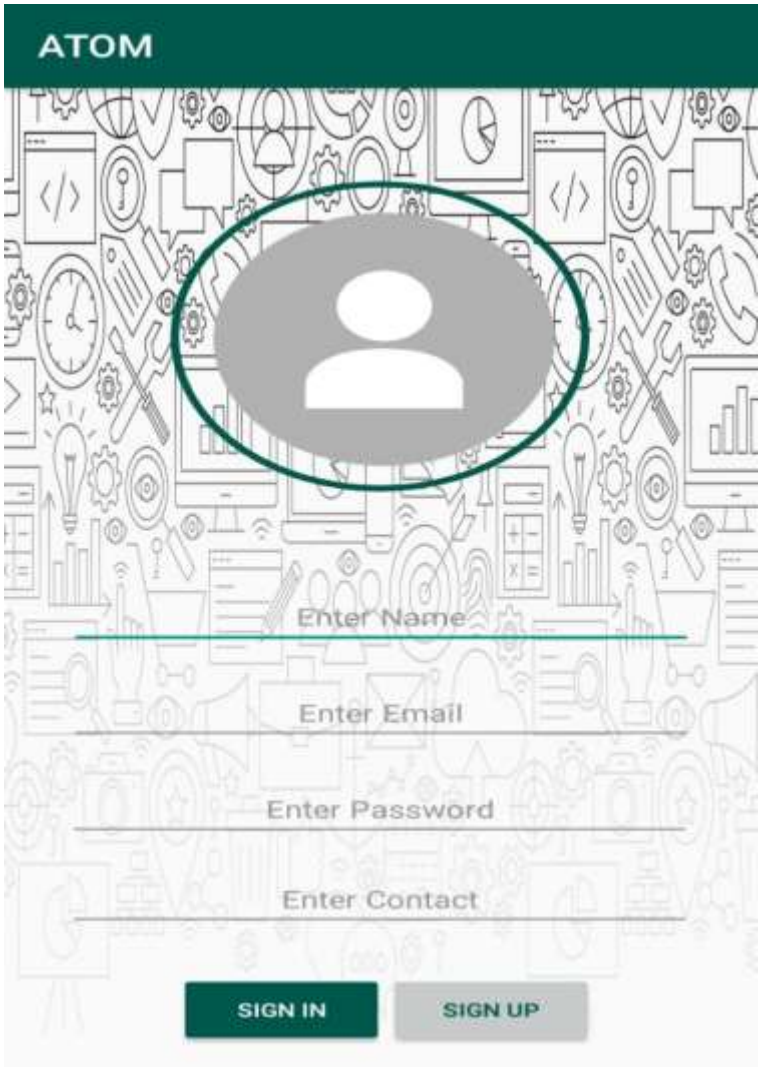

Figure 17 Screenshot from Atom for Register Activity

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.

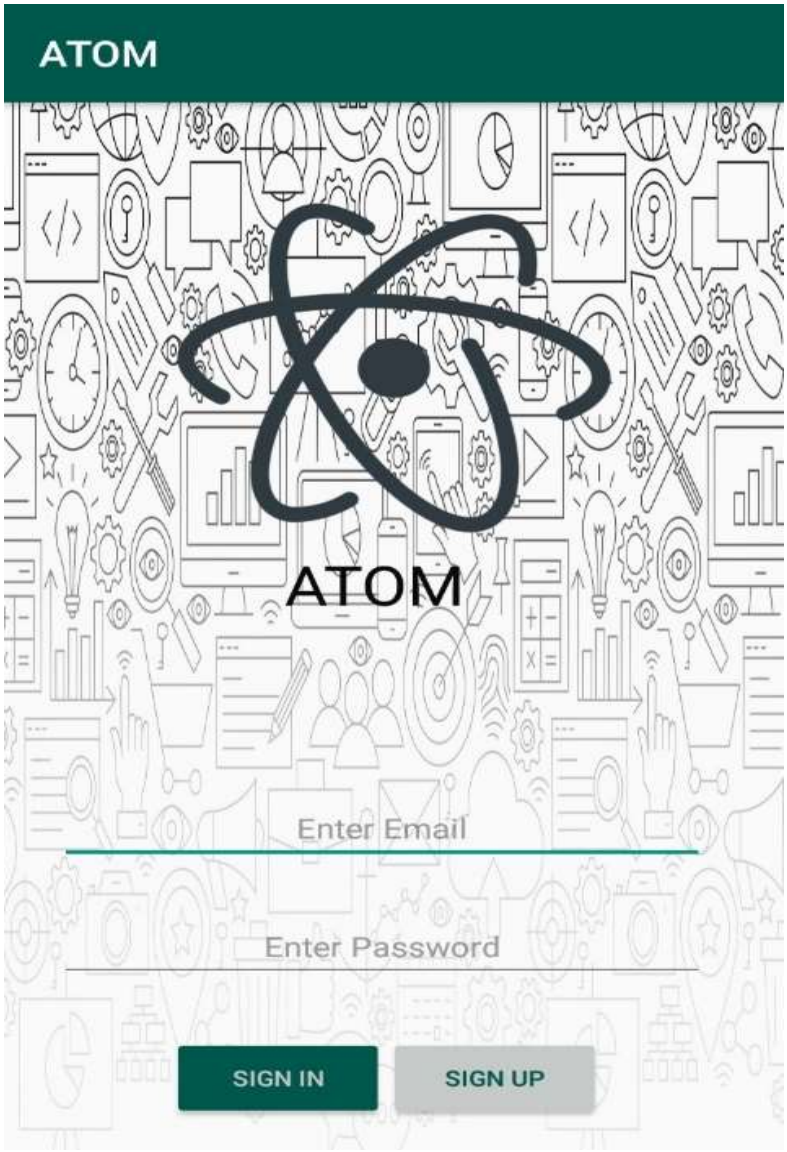

Figure 18 Screenshot from Atom for sign-in Activity

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.

Figure 19 Screenshot from Atom for Dashboard Activity

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.

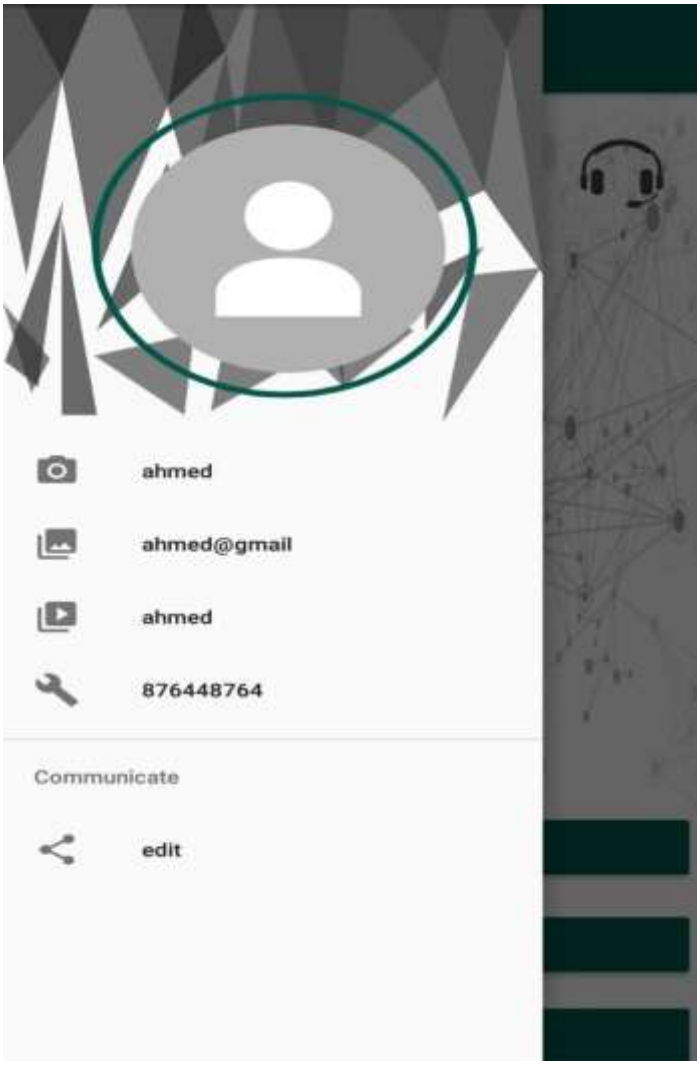

Figure 20 Screenshot from Atom for Drawer Activity

Bookshelf Activity:

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

Figure 21 Screenshot from Atom for Bookshelf Activity

PDFViewer Activity:

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

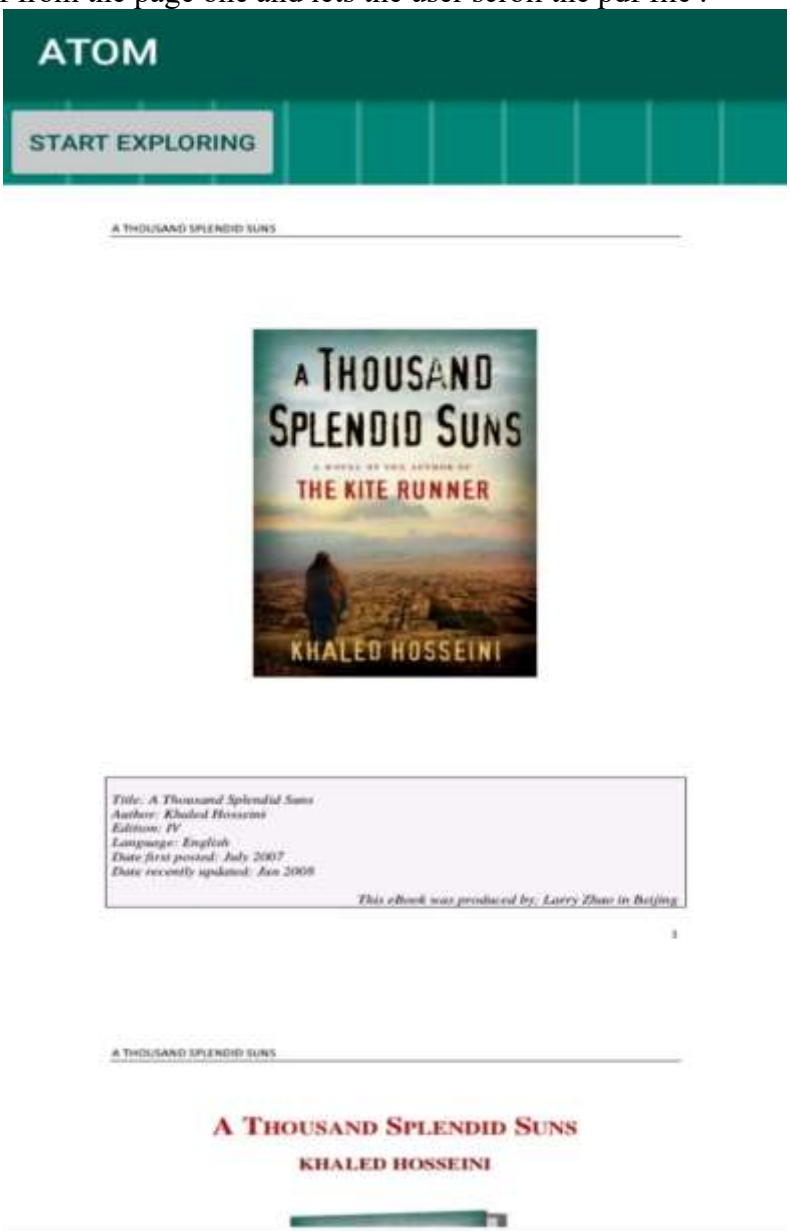

Figure 22 Screenshot from Atom for Book Reader Activity

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.

Figure 23 Screenshot from Atom for Game List Activity

Game:

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

Figure 24 Screenshot from Atom for Game Activity

Headset Settings:

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

Figure 25 Screenshot from Atom for Headset Settings Activity