



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

SOFTWARE DESIGN DESCRIPTION

for

**Atom – Brain-Computer Interfacing using
Electroencephalography**

Version 2.0

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)

Table of Contents

1. Introduction.....	1
2. Design Methodology and Software Process Model.....	1
2.1 Architectural Design.....	1
2.2 Process Flow/Representation.....	2
3. Design Models.....	7
3.1 Use Case Models	7
3.2 Sequence Diagrams	11
3.3 Class Diagram	15
4. Data Design.....	17
4.1 Data Dictionary	21
5. Algorithm & Implementation	22
6. Software requirements traceability matrix	26
7. Human interface design.....	32
8.2 Screen objects and actions	32
8. Conclusion	38

Table of Architecture Model(s)

Architectural Model 1 Atom in MVC	2
--	----------

Table of Activity Diagrams

Activity Diagram 1 Account Handling Module	3
Activity Diagram 2 User Analytics and Statistics.....	4
Activity Diagram 3 Specialized Control Training	5
Activity Diagram 4 Entertainment Incentivized Training.....	6

Table of Use case Models

Use case Model 1 Account Handling module	7
Use case Model 2 User Analytics and Statistics module	8
Use case Model 3 Specialized Control Training module	8
Use case Model 4 Entertainment Incentivized Training module	9
Use case Model 5 EEG Feature Extraction module	10
Use case Model 6 Database Handling module.....	10

Table of Sequence Diagrams

Sequence Diagram 1 Preliminary use cases.....	12
Sequence Diagram 2 User Analytics and Statistics use cases.....	14
Sequence Diagram 3 Specialized Control Training use cases	14
Sequence Diagram 4 Entertainment Incentivized Training use cases	15

Table of Screenshots

Interface 1 Register Activity.....	32
Interface 2 Dashboard Activity.....	33
Interface 3 Drawer Activity	34
Interface 4 Bookshelf Activity	35
Interface 5 Book Reader Activity	36

Revision History

Name	Date	Reason for changes	Version
Dr. Yasir Faheem (Supervisor)	4 th December, 2019	<ul style="list-style-type: none">- Inclusion of comments from previous evaluation- Image labelling according to type of Model/Diagram/Figure- Description of text covered under any particular heading- Add conclusion to the document	1.0

Application Evaluation History

Comments (by committee) *include the ones given at scope time both in doc and presentation	Action Taken
- Related system Analysis spacing	Added some space above the table
- Following modules lack specificity: <ul style="list-style-type: none">- Account Handling module- Specialized Control Training module- EEG Feature Extraction module- Database handling module	Added more specifics on each and every one of the modules
- Use case diagrams not according to convention	Followed Sommerville convention for the use case diagrams by keeping action lines outside the boundaries and limiting complex models
- Missing captions from use case tables	Added captions to use case tables
- Incomplete functional requirements	Added more definition to the requirements

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 of communication that our neurons 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

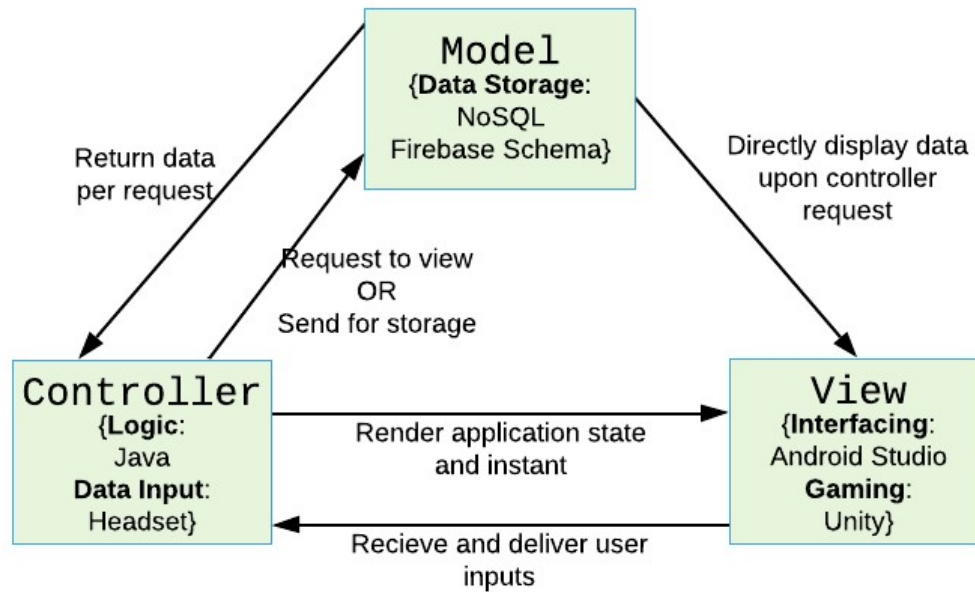
Since this project belongs to the same framework of research to the branches of Human-Computer Interaction like Social Networking, Augmented Reality, etc., our design methodology is primarily based around the focus of keeping the entire design user-experience-centric. It should pertain to accessibility, easy intractability, and minimalistic user interface provision. Regarding the process model, the software we are supposed to build is in close interaction with uncommon hardware and a rarely used peripheral device i.e. the brain, we are working in an Agile Development process model where our focus is on immediate prototype deployment and simultaneous self-induced testing and criticism.

System Overview

Since the primary choice to deploy Atom is to release an application or software on a platform which a mass majority of users are comfortable and familiar with, we've streamlined production on smartphone and due to technical barriers, the pre-dominant choice is Android. According to convention, Android targeted software are inherently followers of the Model-View-Controller paradigm, and in addition to that, the below diagram begins to concretize by providing further insight into each block. An implied improvisation on this architecture is the inclusion of the headset with the controller and not allocating another block for the peripheral to prevent breaking standard convention, although an accurate visualization might consider this as a separate block.

2.1 Architectural Design

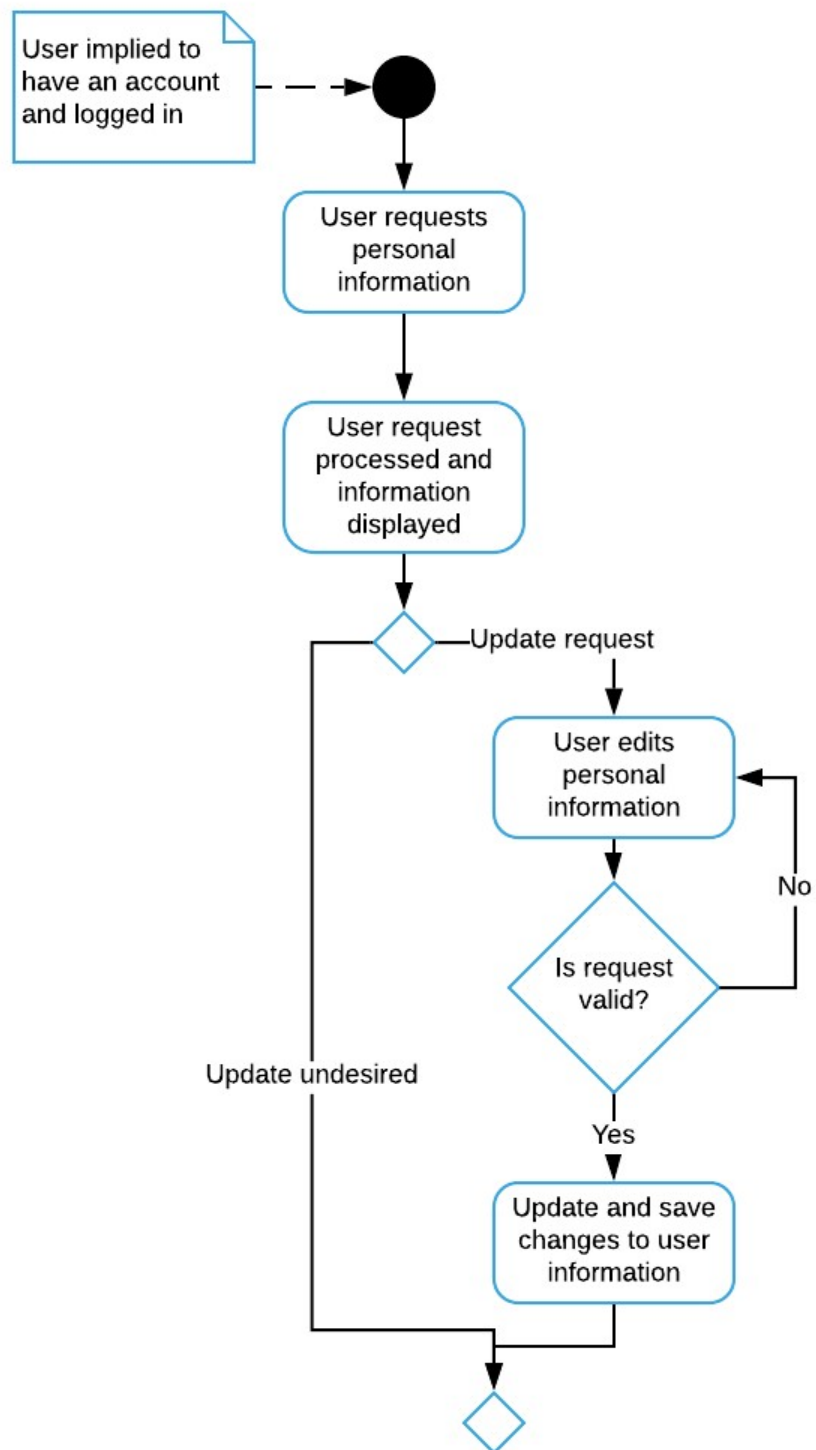
The following presents the block diagram of Atom:



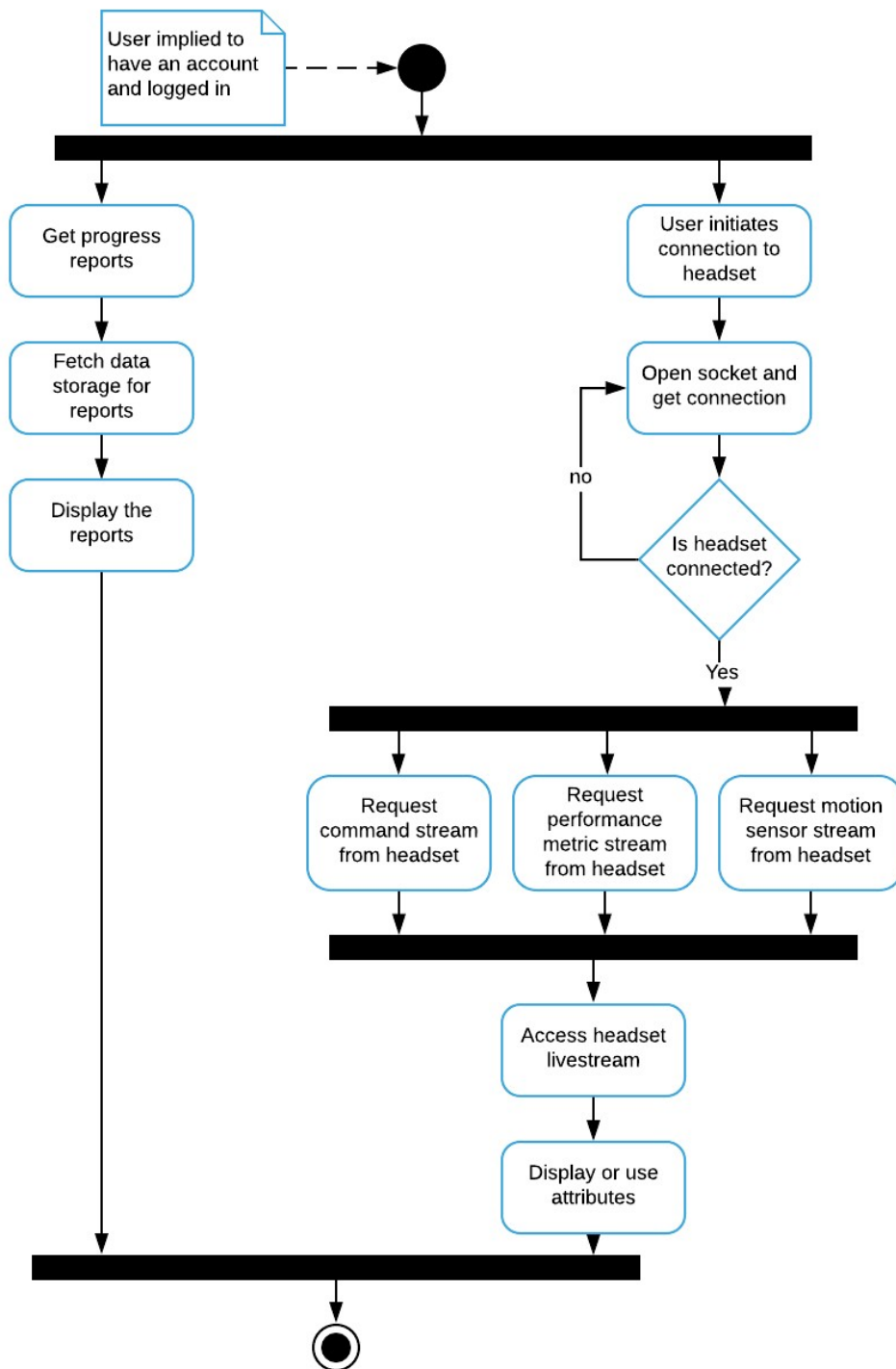
Architectural Model 1 Atom in MVC

2.2 Process Flow/Representation

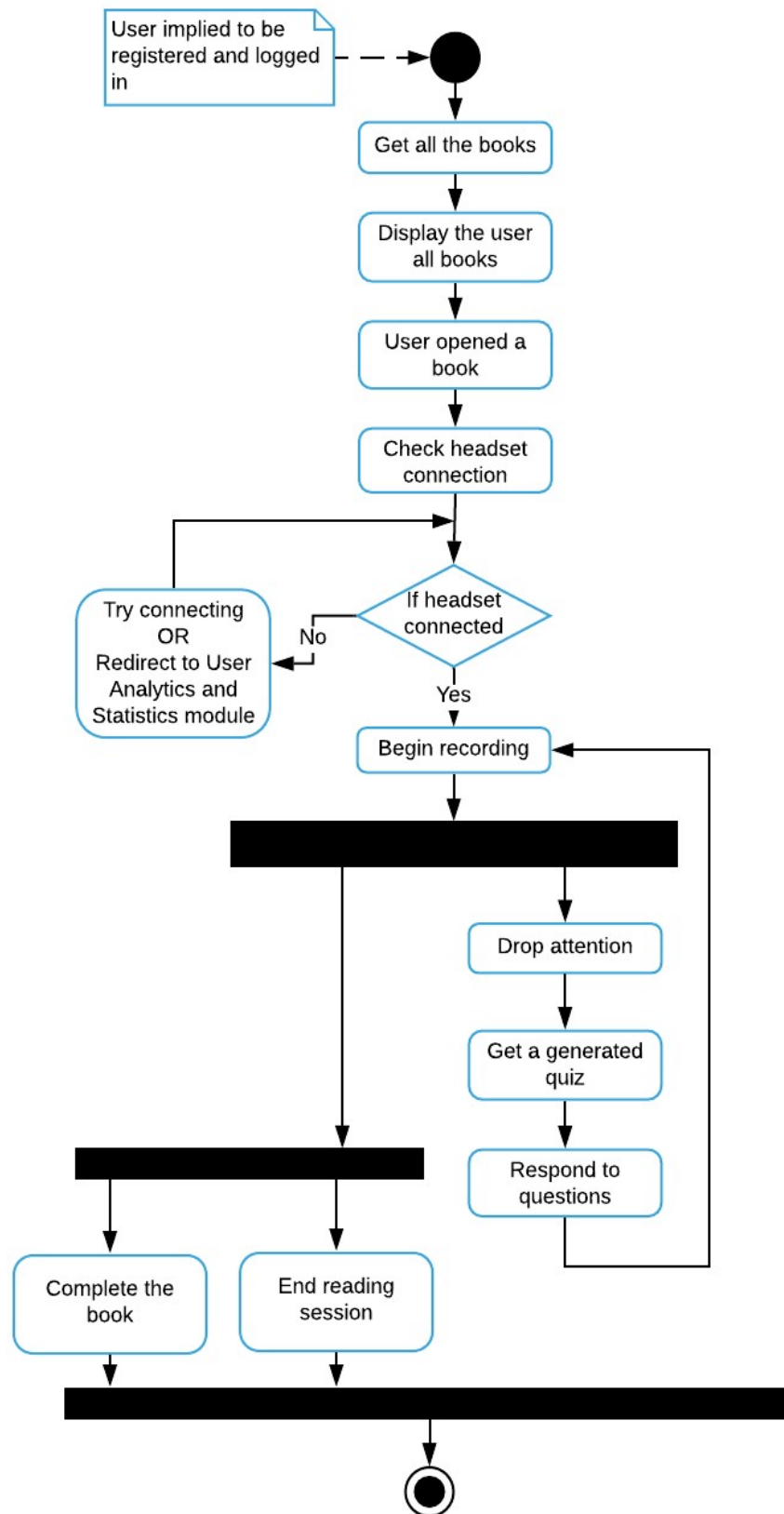
The following section will provide Activity Diagrams for specific actions that the user can conduct based on the specific modules:



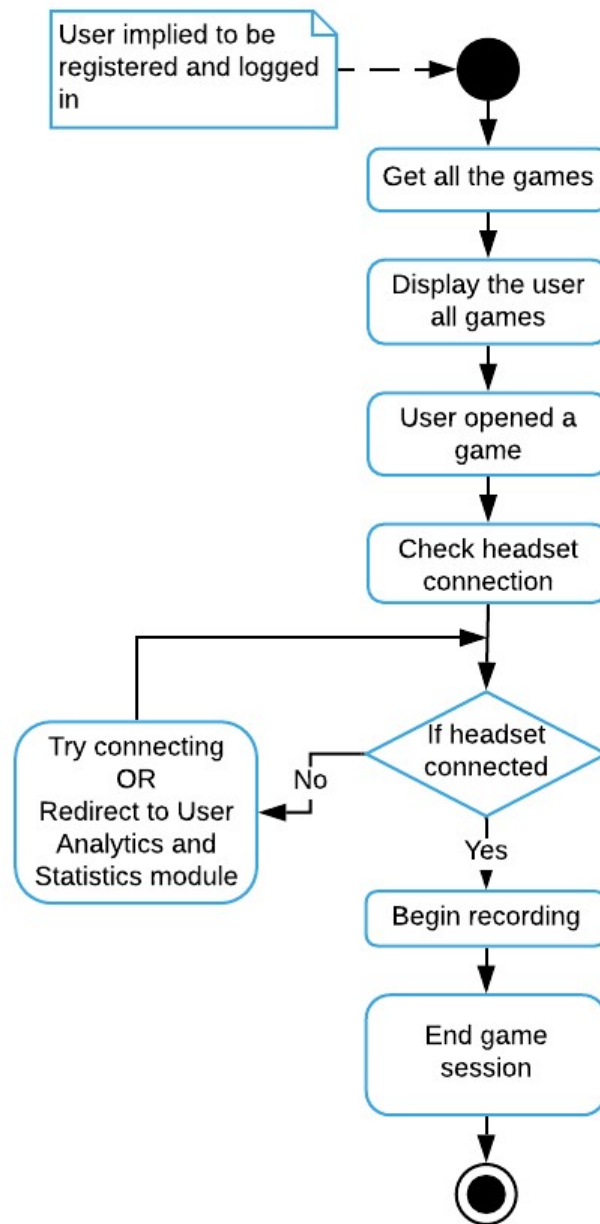
Activity Diagram 1 Account Handling Module



Activity Diagram 2 User Analytics and Statistics



Activity Diagram 3 Specialized Control Training

**Activity Diagram 4** Entertainment Incentivized Training

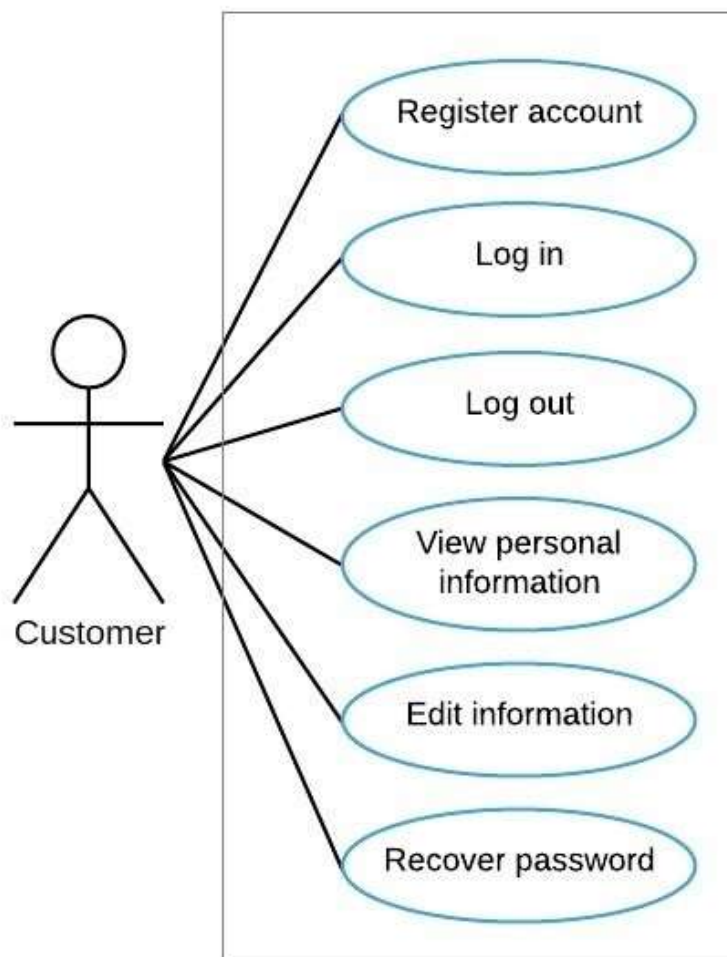
3. Design Models

In consideration of the previous models and to further clear away the design abstraction, the following section presents:

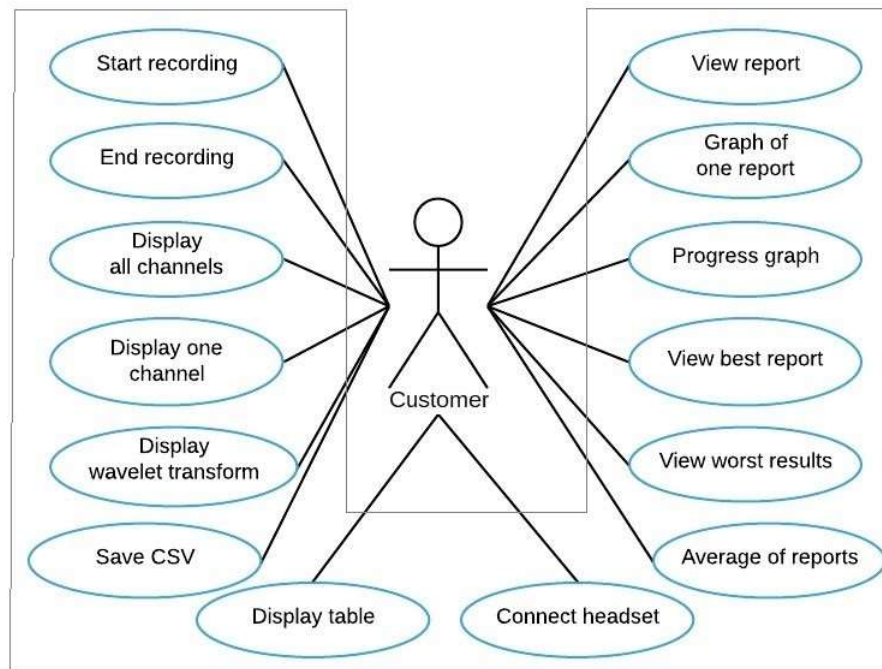
- Use case Models
- Sequence Diagrams
- Class Diagram

3.1 Use Case Models

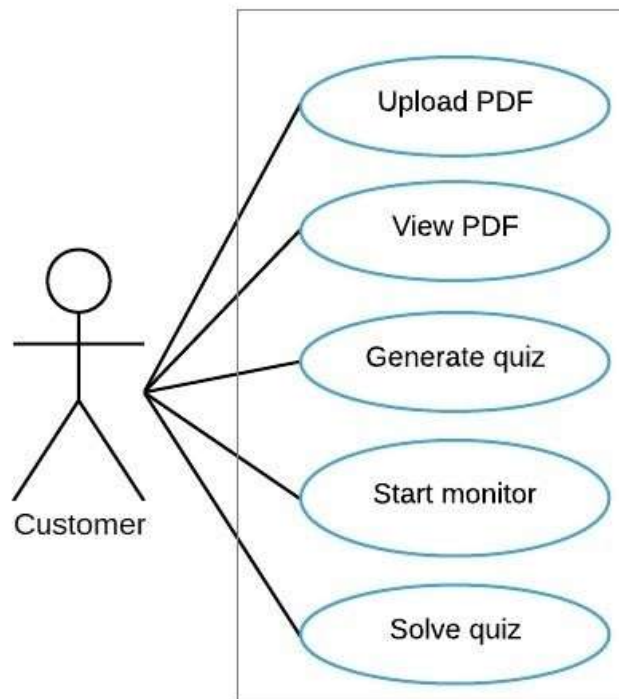
The use case models of Atom:



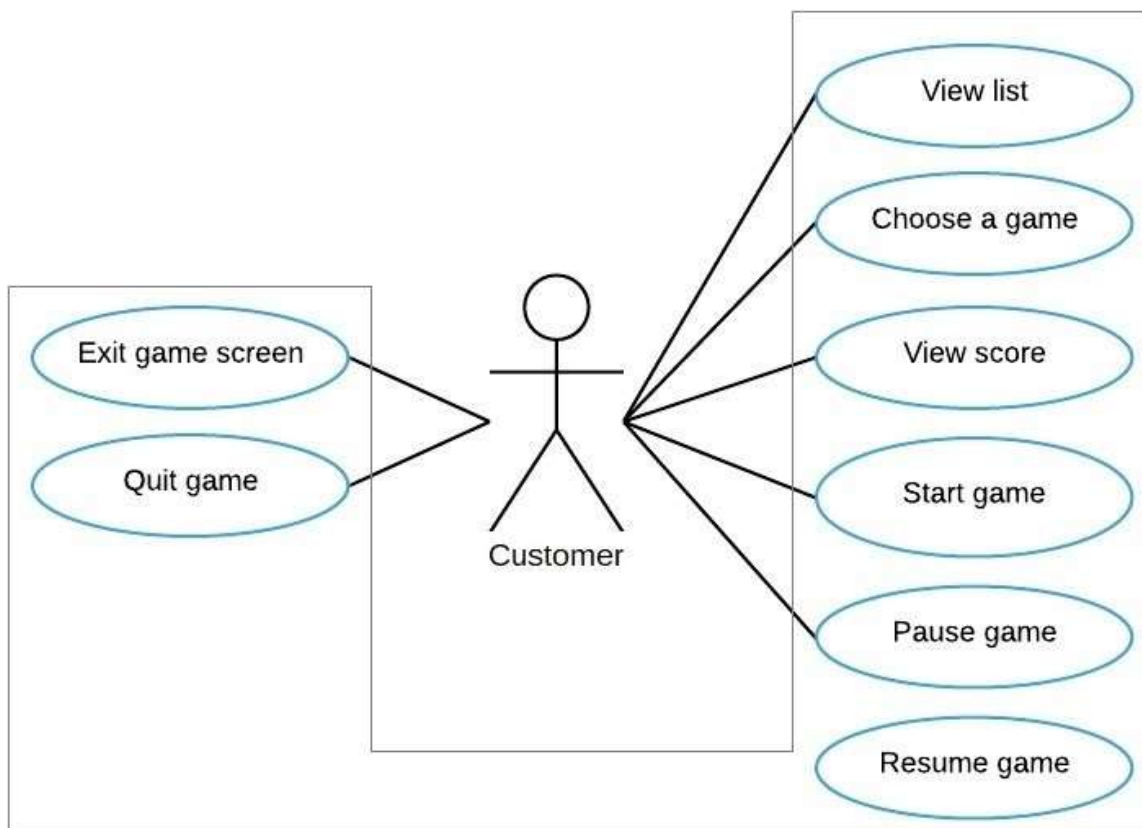
Use case Model 1 Account Handling module



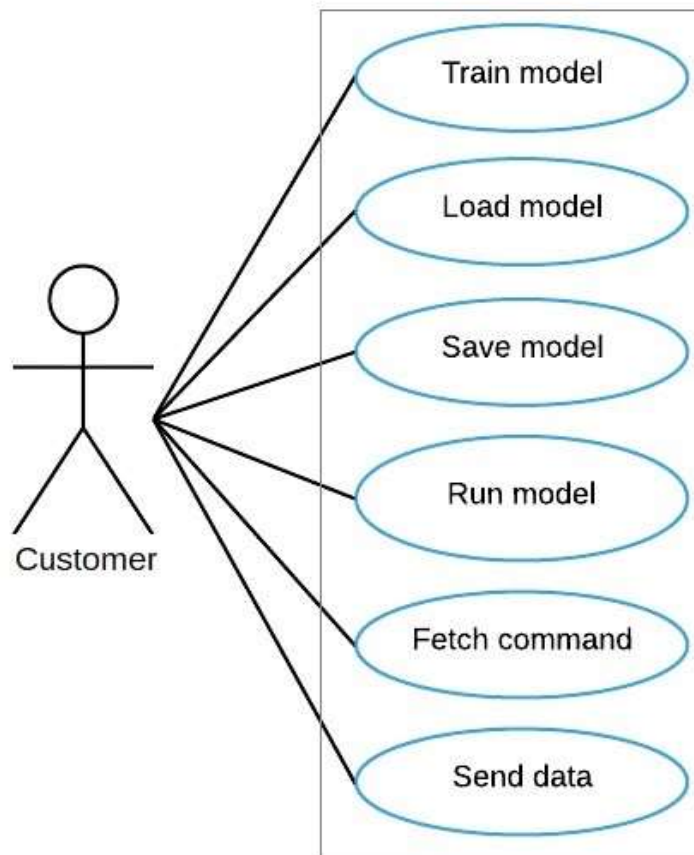
Use case Model 2 User Analytics and Statistics module



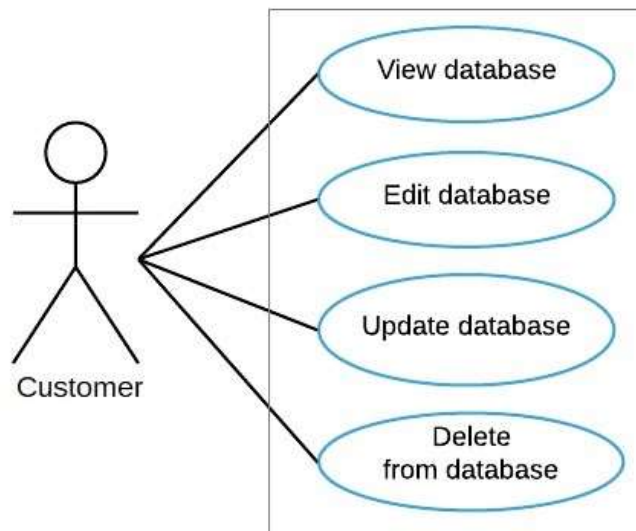
Use case Model 3 Specialized Control Training module



Use case Model 4 Entertainment Incentivized Training module



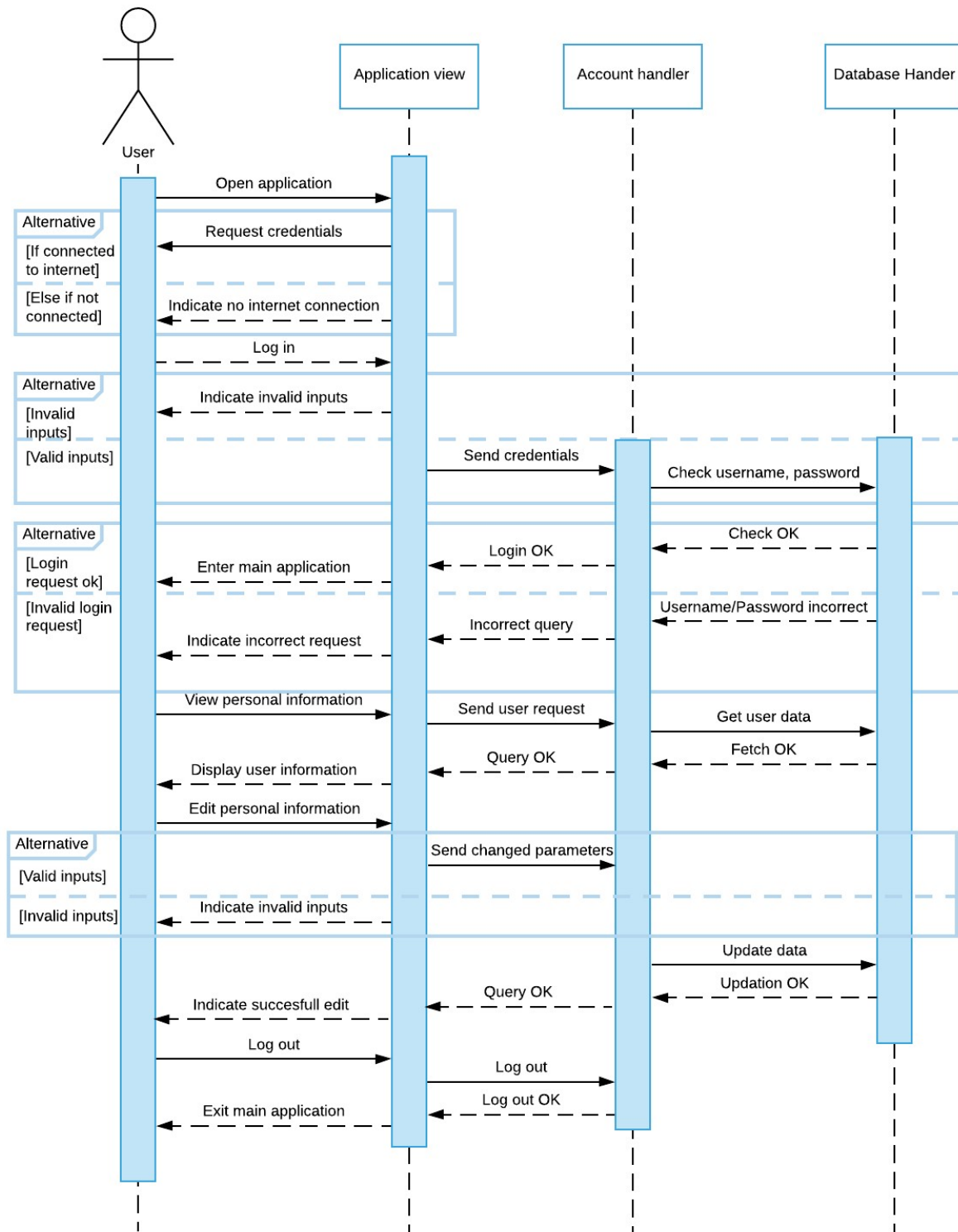
Use case Model 5 EEG Feature Extraction module



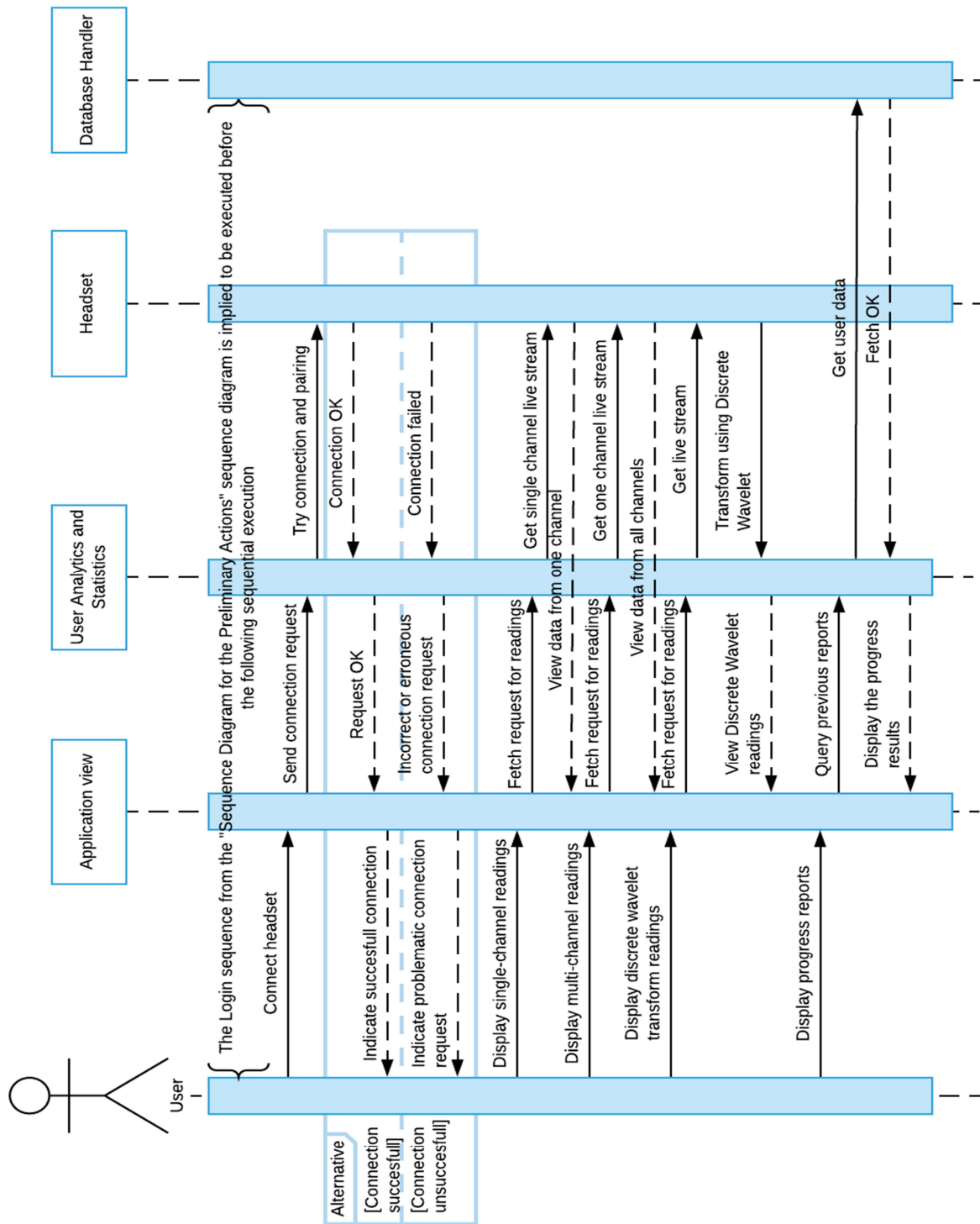
Use case Model 6 Database Handling module

3.2 Sequence Diagrams

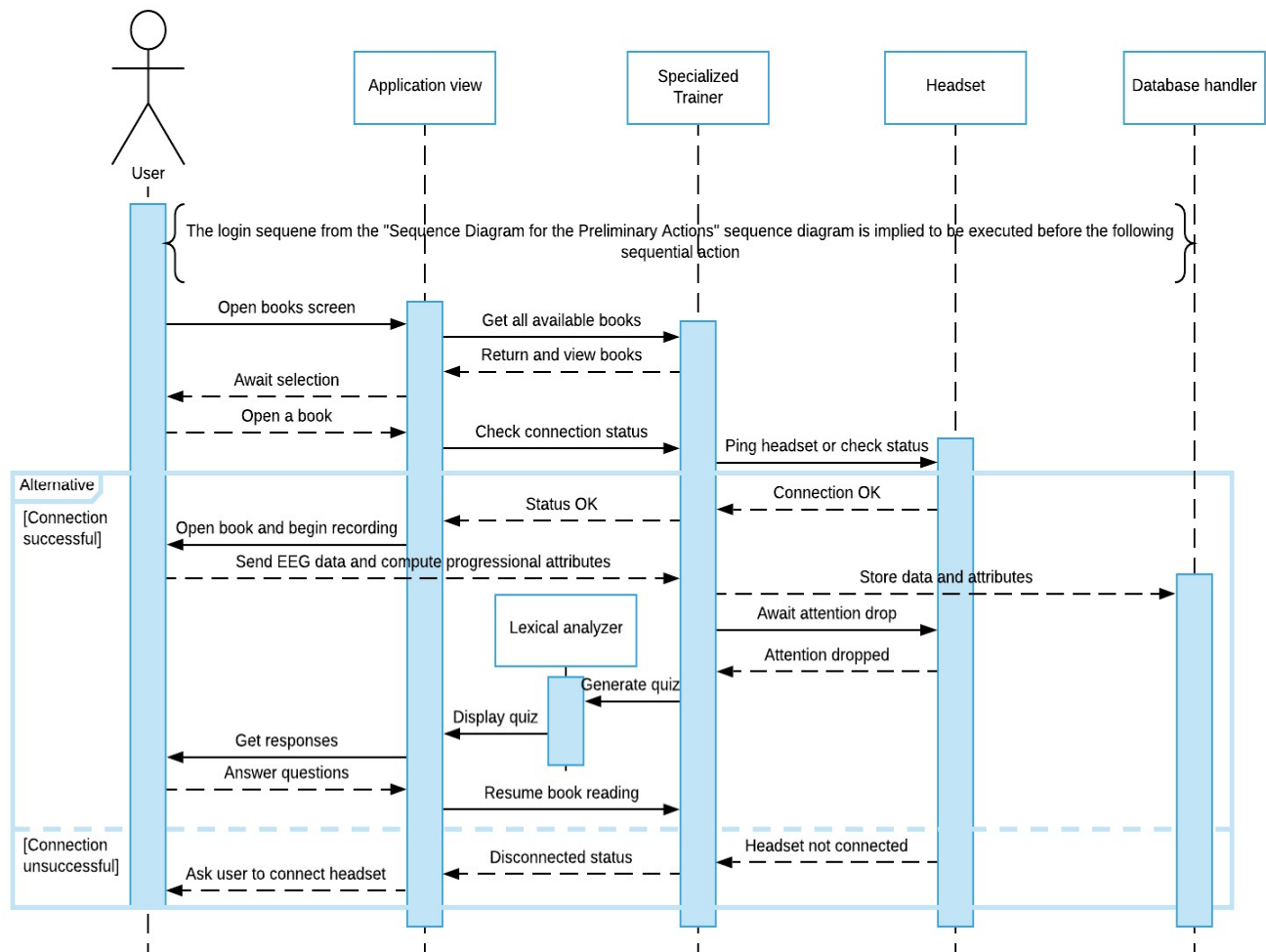
The sequence diagrams of Atom:



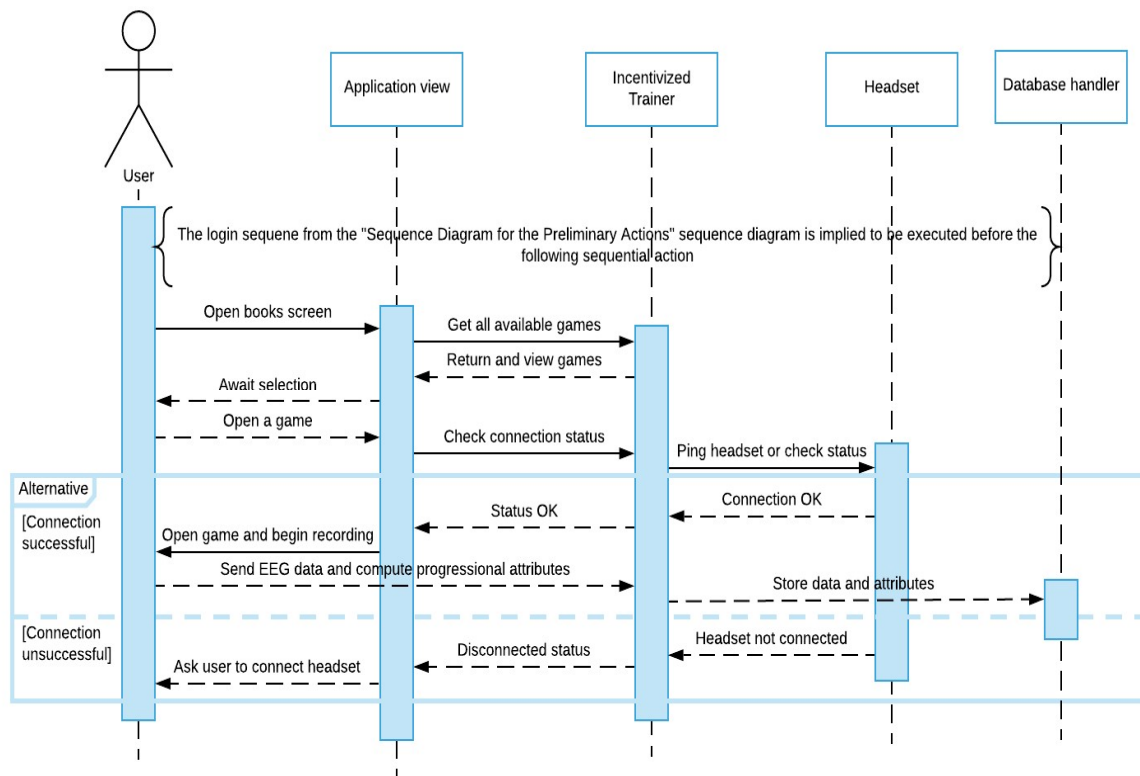
Sequence Diagram 1 Preliminary use cases



Sequence Diagram 2 User Analytics and Statistics use cases



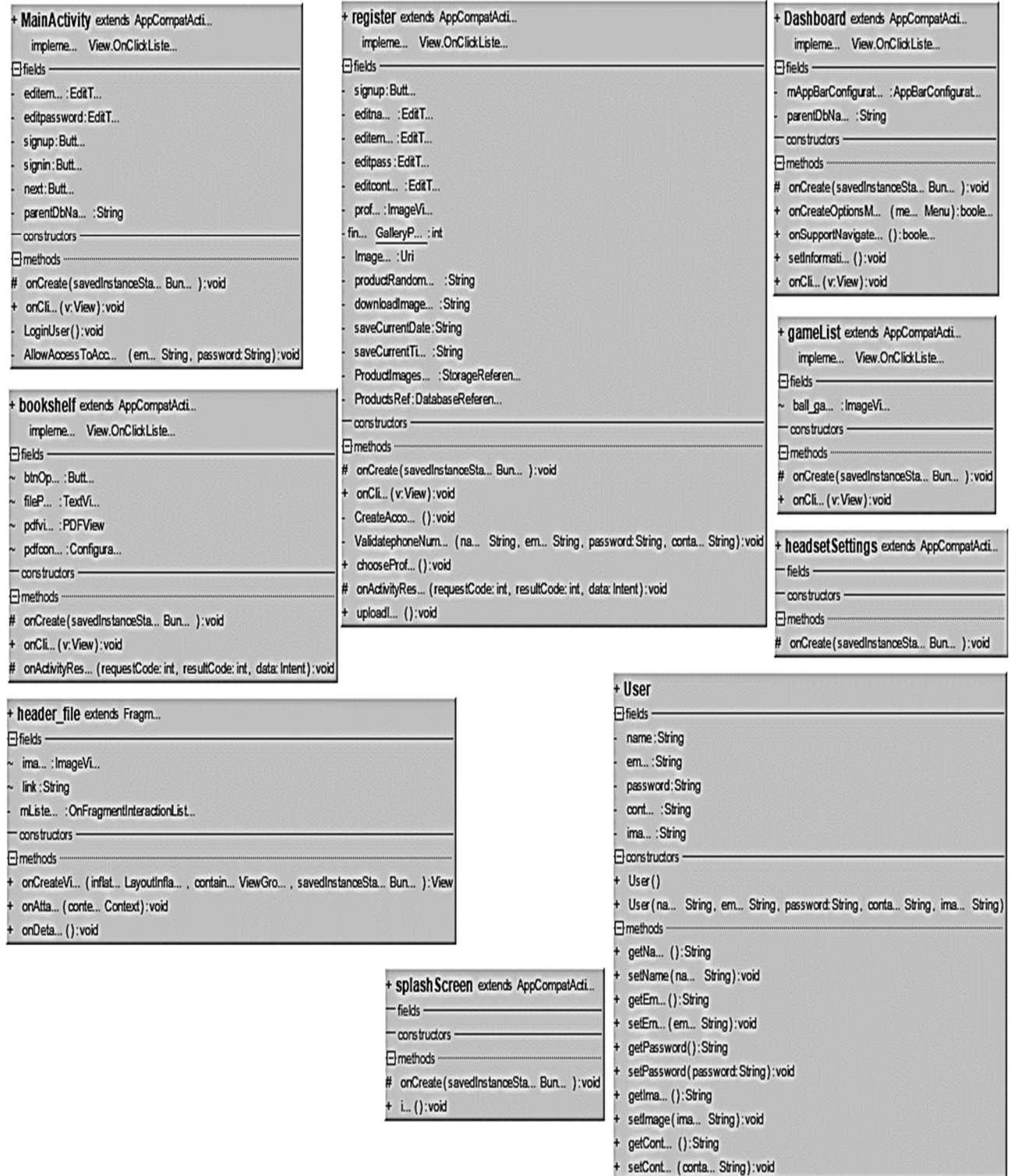
Sequence Diagram 3 Specialized Control Training use cases



Sequence Diagram 4 Entertainment Incentivized Training use cases

3.3 Class Diagram

The class diagram for the Android application for Atom:



Class Diagram 1 Atom Class Diagram

4. Data Design

This section contains the JSON schema as an alternative to ERD Diagram since the storage mechanism that will be used is a NoSQL Storage Methodology:

```
{
  {
    "$schema" : " ",
    "$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" ]
}
}
```

4.1 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	“20”

				progress of the users	
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	“ “

5. Algorithm & Implementation

The major algorithms in form of pseudocode:

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:

```
//update 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
```

6. 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	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	onCreate() 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()

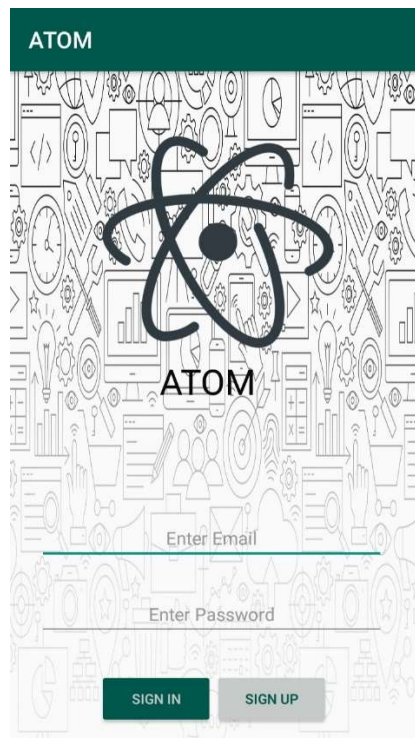
7. Human interface design

This section is a view into the primary build of the working application interfaces:

8.2 Screen objects and actions

registerActivity:

This activity has inputs in the form 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.



Interface 1 Register Activity

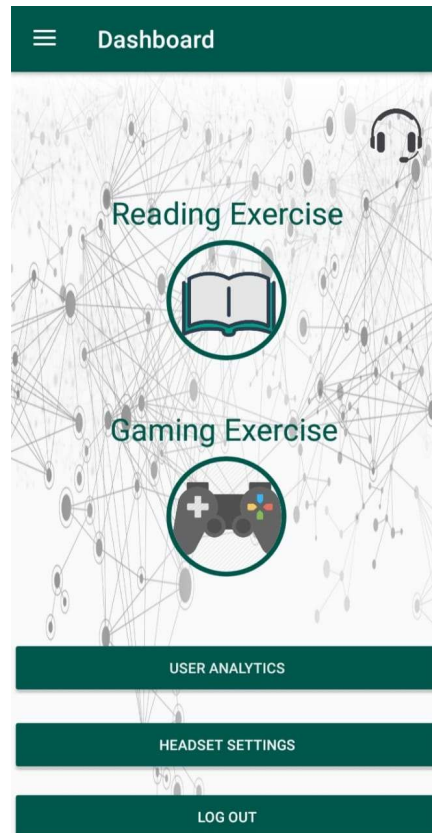
sign-in Activity:

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.

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

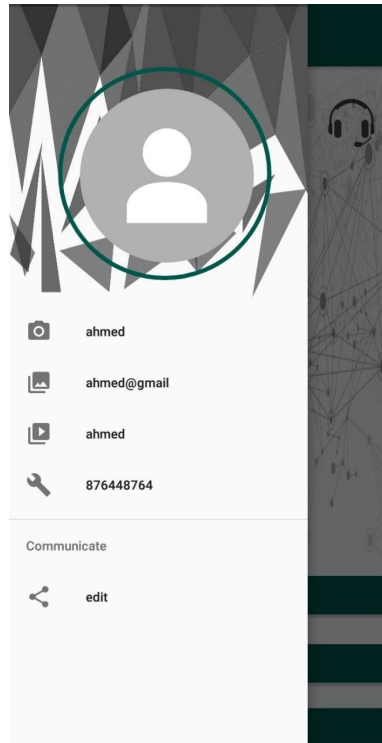
the headset is connected or not . The buttons on the bottom are User Analytics , Headset Settings , Log-out . The User Analytics button takes us to an activity which lets us view the data in different formats . The Headset Settings opens an activity that lets us see the connection status and signal strength with our headset. The logout buttons logs the system out and deletes data from the paper.



Interface 2 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.



Interface 3 Drawer Activity

Bookshelf Activity:

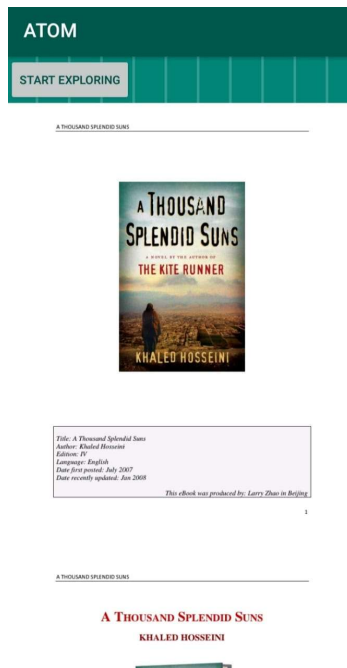
This activity has one button that calls the default choose file action to let the user choose a .pdf file.



Interface 4 Bookshelf Activity

PDFViewer Activity:

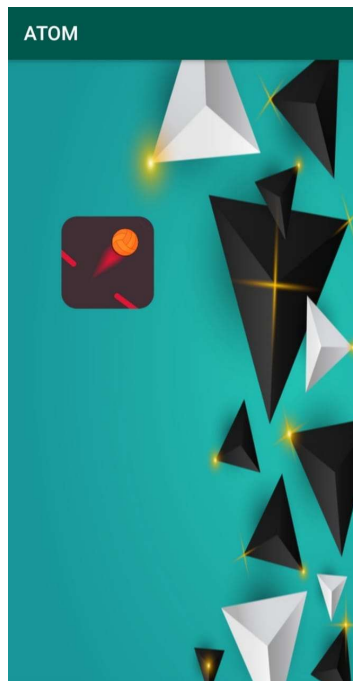
This activity loads the pdf from the page one and lets the user scroll the pdf file .



Interface 5 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.



Interface 6 Game List Activity

Game:

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



Interface 7 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.



Interface 8 Headset Settings Activity

8. Conclusion

The document tends to present and define the design specification of Atom. It defines the high-level Architecture of the application using the Architectural block diagram up to the low level sequences and activities using the corresponding diagrams and concretizes code using the class diagram presented and the algorithms and implementation techniques presented above. This document will be used as a standpoint for further development into the project leading into a stable and compelling application.