



# CENG491 Initial Design Report

Buğra Şirin

Deniz Tav

Engin Fırat

Mert Özer

## BiBER





1. Introduction.....	5
1.1. Problem Definition .....	5
1.2. Purpose.....	5
1.3. Scope .....	5
1.4. Overview.....	5
1.5. Definitions, Acronyms and Abbreviations.....	6
1.6. References.....	6
2. System Overview .....	6
2.1. Core Module.....	7
2.2. BCID Module.....	7
2.3. Game Module.....	7
2.4. Graphical Interpreter Module .....	8
2.5. Database Module .....	8
2.6. Logger .....	8
3. Design Considerations.....	8
3.1. Design Assumptions, Dependencies and Constraints .....	8
3.1.1. Time Constraints.....	8
3.1.2. Resource Constraints.....	8
3.1.3. Performance Constraints.....	9
3.1.4. Software Constraints .....	9
3.1.5. Hardware Constraints.....	9
3.2. Design Goals .....	9
3.2.1. Performance.....	9
3.2.2. Reliability .....	10
3.2.3. Functionality .....	10
3.2.4. Usability.....	10
4. Data Design.....	10
4.1. Data Description.....	10
4.2. ER Design .....	11
4.3. Database Schemas.....	12
4.4. Class Diagrams.....	13
4.4.1. Model Package .....	13
4.4.2. Controller Package .....	14





4.5. Data Dictionary .....	15
5. System Architecture .....	16
5.1. Architectural design .....	16
5.2. Description of Components.....	17
5.2.1. Core Module.....	17
5.2.1.1. <i>Processing Narrative for Core Module</i> .....	17
5.2.1.2 <i>Interface Description for Core Module</i> .....	18
5.2.1.3 <i>Processing Detail for Core Module</i> .....	18
5.2.1.4 <i>Dynamic Behavior</i> .....	18
5.2.2. BCID Module.....	19
5.2.2.1. <i>Processing Narrative for BCID Module</i> .....	19
5.2.2.2 <i>Interface Description for BCID Module</i> .....	20
5.2.2.3 <i>Processing Detail for BCID Module</i> .....	20
5.2.2.4 <i>Dynamic Behavior</i> .....	20
5.2.3. Game Module.....	20
5.2.3.1. <i>Processing Narrative for Game Module</i> .....	21
5.2.3.2 <i>Interface Description for Game Module</i> .....	21
5.2.3.3 <i>Processing Detail for Game Module</i> .....	21
5.2.3.4 <i>Dynamic Behavior</i> .....	22
5.2.4. Graphical Interpreter Module .....	23
5.2.4.1. <i>Processing Narrative for Graphical Interpreter Module</i> .....	23
5.2.4.2. <i>Interface Description for Graphical Interpreter Module</i> .....	23
5.2.4.3. <i>Processing Detail for Graphical Interpreter Module</i> .....	23
5.2.4.4 <i>Dynamic Behavior</i> .....	24
5.2.5. Database Module .....	24
5.2.5.1. <i>Processing Narrative for Database Module</i> .....	24
5.2.5.2. <i>Interface Description for Database Module</i> .....	24
5.2.5.3. <i>Processing Detail for Database Module</i> .....	25
5.2.5.4. <i>Dynamic Behavior</i> .....	25
5.2.6. Logger Module .....	25
5.2.6.1. <i>Processing Narrative for Logger Module</i> .....	25
5.2.6.2. <i>Interface Description for Logger Module</i> .....	25
5.2.6.3. <i>Processing Detail for Logger Module</i> .....	26
5.2.6.4 <i>Dynamic Behavior</i> .....	26





5.3 Design Rationale .....	26
6. User Interface Design .....	27
6.1. Overview of User Interface .....	27
6.1.1. Login Screen.....	27
6.1.2. Main Screen.....	27
6.1.2.1. Administrator Case .....	27
6.1.2.2. Patient Case.....	28
6.1.3. Search Screen.....	29
6.1.4. Registration Screen.....	30
6.1.5. Graphic Data Screen .....	30
6.1.6. Game Screen.....	31
6.2. Screen Images.....	32
6.2.1. Login Screen.....	32
6.2.2. Game Screen.....	32
6.2.3. Registration Screen.....	33
6.3. Screen Objects and Actions.....	33
7. Time Planning (Gantt Chart) .....	33
7.1. Term 1 Gantt Chart.....	34
7.2. Term 2 Gantt Chart.....	35
8. Libraries and Tools .....	35
9. Conclusion .....	36



## 1. Introduction

### 1.1. Problem Definition

The problem that we attend to fix is rehabilitation of attention deficiency, performance disorders by using neurofeedbacks provided from Brain Computer Interface Devices (BCID). The product is going to help to students, sportsmen and businessman.

### 1.2. Purpose

This report provides the necessary definitions to conceptualize and further formalize the design of the software, of which its requirements and functionalities were summarized in the previous requirements analysis report. The aim is to provide a guide to a design that could be easily implemented by any designer reading this report.

### 1.3. Scope

Mindolog will have a hardware part and a software part. Hardware part is the BCID (Brain Computer Interface Device) and software is composed of a simple game which will be controlled by alpha and beta waves of the user, an alpha-beta interpreter and a statistical output graphic.

In this project, we are going to use a brain computer interface that receives alpha and beta waves from the prefrontal cortex of the brain. [1]

The software will save this information to the database and user will be able to see this information. User's brain activities will be displayed while he is playing the game. According to the improvement of the user after the game sessions, the statistical output will be displayed.

### 1.4. Overview

This document includes initial design report for Mindolog. First, an overview of the problem and the product are described. Then system overview and design considerations are presented to the audience. After declaring the data design of this project, system architecture will be clarified explicitly. Then user interface design and the detailed design of

the Mindolog are explained clearly. And finally, time planning of the project will be given by this document.

### *1.5. Definitions, Acronyms and Abbreviations*

IDR: Initial Design Report

BCID: Brain Computer Interface Devices

EEG: Electroencephalographic

IEEE STD 830-1998: IEEE Recommended Practice for Software Requirements Specifications

AX-CPT: type of test that wants respondent to memorize the sequence of characters.

GUI: Graphical User Interface

### *1.6. References*

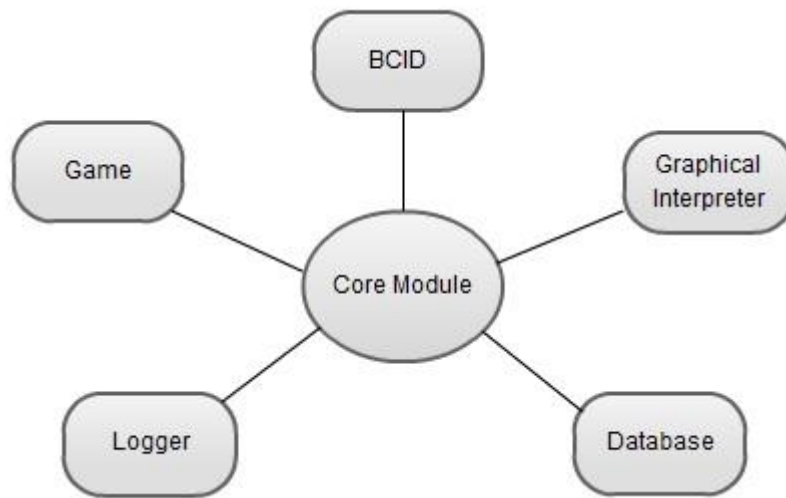
1. Wolpaw, J.R. & Birbaumer, N. & McForland, D.J. & Pfurtscheller, G. & Vaughn, T.M. (2002, March). Brain-Computer Interfaces for Communication and Control.
2. McForland, D.J. & Miner, L.A. & Vaughn, T.M. & Wolpaw, J.R. (2000, January). Mu and Beta Rhythm Topographies During Motor Imagery and Actual Movements.
3. <http://creately.com/>
4. [http://en.wikipedia.org/wiki/Flow-based\\_programming](http://en.wikipedia.org/wiki/Flow-based_programming)
5. <http://jpaulmorrison.com/fbp/>
6. [http://en.wikipedia.org/wiki/JMonkey\\_Engine](http://en.wikipedia.org/wiki/JMonkey_Engine)
7. <http://www.jmonkeyengine.com/>

## **2. System Overview**

This system runs according to the data which is provided by the EEG device. In the system the data is processed and modules in the system run appropriately. The system consists of modules hence the system has a modular property. There are six modules namely Core, BCID, Game, Graphical Interpreter, Database and Logger. These modules do not know the source



of the data coming from the core and do not know where the data go which they produce. The general system overview is depicted below:



**Figure #1** System Overview Diagram

The explanations of the modules are as follows:

### 2.1. Core Module

This module manages the system. Time management is handled in this module. Moreover all the data flows are accomplished in this module. In this manner other modules cannot communicate with each other. In other words, all the modules act like a black box which exchange data across predefined connections passing through the core module.

### 2.2. BCID Module

This module provides the data coming from the EEG device which is provided by BCID module consisting of two parts. First part is the third-party software provided by the company which produces the EEG device. The other part is a library provided by our sponsor company MINDER. These two software pieces process the meaningless raw data coming from the device and produces the meaningful raw data for our own use.

### 2.3. Game Module

Game uses the incoming data coming from the data flow managed by the core module. Game proceeds by these data and keyboard input.



### ***2.4. Graphical Interpreter Module***

This module uses the incoming data coming from the data flow managed by the core module and uses these data to produce output graph which shows the process of the patient.

### ***2.5. Database Module***

Saves data coming from the core module to the database and provides data to core module when asked.

### ***2.6. Logger***

According to the coming data from Core module, it saves the error message to the log file. The main purpose of this module is to provide technical data when an unexpected situation occurs.

## **3. Design Considerations**

The system is designed with an Object-Oriented paradigm hence each module is presented by a class. Since each class represents a module, all the module classes are implemented by a Singleton Design Pattern. Each module has to communicate only with the Core Module.

### ***3.1. Design Assumptions, Dependencies and Constraints***

#### ***3.1.1. Time Constraints***

The system has to run synchronously with the EEG device. This means that all the modules have to finish their job before EEG device produces new data. For example, if the EEG device produces data in a frequency of 50Hz, the slowest module of the program must accomplish its job in that frequency.

#### ***3.1.2. Resource Constraints***

Most of the services provided by the modules will be implemented by the help of third party software; therefore, if appropriate software could not be found we have to implement the appropriate one.





### 3.1.3. Performance Constraints

The system performance is strongly dependent on the EEG device performance because most of the data processed by the modules is provided by the EEG device. Moreover, the interaction between the modules and the core affects the overall performance of the system.

### 3.1.4. Software Constraints

The system will be implemented by using Java programming language.

BCID module will be provided by our sponsor MINDER.

The system consists of a game module. This module needs to render some computer generated graphics. We will use jMonkey graphics engine for this module.

Graphical Interpreter needs to render chart data graphics. To implement this module we will use jFreeChart library.

Database Module interacts with a PostgreSQL DBMS. JDBC will be used to provide appropriate connection between the system and the database.

Logger Module will be implemented by the primitive interfaces of the Java.

### 3.1.5. Hardware Constraints

There are 128 points in the brain that are suitable for receiving brain waves (alpha, beta, theta, and delta). However, the BCI that we use receives brain waves only from 16 of 128 points. This is the best available technology today. We will be using alpha and beta waves and the correctness of the measurements depends on this hardware.

## 3.2. Design Goals

### 3.2.1. Performance

The only part of the system that can be challenging in the sense of performance is the graphical interpreter because it must read and write data concurrently. Design and implementation will be accurate enough to reach the desired performance.



### 3.2.2. Reliability

Since Mindolog will be a healthcare system and will be keeping data for several patients, we have to design it with minimum faults. Another issue about the reliability is that the data that will be used and produced must be error-free. We will use various testing strategies at the milestones that we reach while designing the project in order to improve the performance and decrease the number of errors that will occur.

### 3.2.3. Functionality

Functionality is another important issue that should be considered. We should keep the interface and features simple for the game part of the project in order to users can play the game easily. However, these games must be appropriate enough to measure the brain waves of the patients.

### 3.2.4. Usability

One of the most important design considerations is usability for our project because this product will be used by doctors and patients who can be among any part of the society. Therefore, we will design a simple and understandable interface.

## 4. Data Design

### 4.1. Data Description

There will be 4 types of data objects in the system namely; user objects, alpha-beta objects, game objects and graph objects.

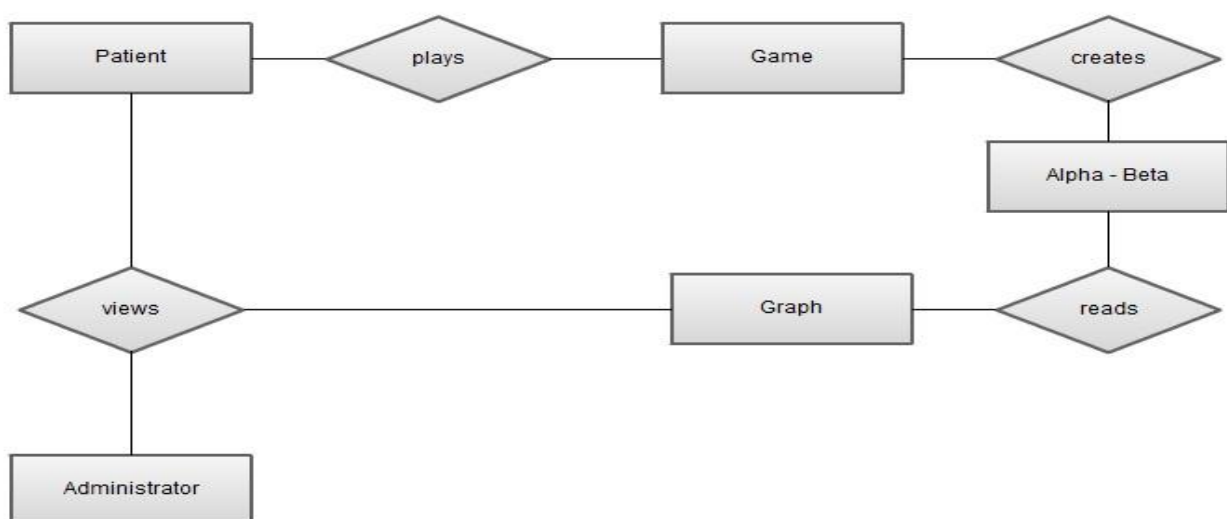
- *User Objects:* This object will hold user id, password, name, surname, birth date, gender, e-mail and phone information of a patient. Other objects will inherit user information from this data object.
- *Alpha beta objects:* This object will hold amplitude values of each brain wave (alpha and beta) of a patient and the exact time of the measurements. These objects will help to determine the session interval with time. For each object there will be also user id attribute.

- *Game Objects:* This object will hold level number to determine the state of the game when user starts to play. That gives a chance to user to continue from where he/she left the game. User id will also be kept.
- *Graph Objects:* This object will hold session times and user information to show session separately alpha beta values for specified user. While alpha beta objects keep all the history of a patient, these objects are created only when a specific session data is desired to draw the graphs separately.

All of the patients, their alpha and beta values and game information are stored in database. Graph information is not hold in the database because it is used only when drawing the graph according to information given. Admin can add patients to database and also can delete. While admin can reach all information of all the patients, every patient can only see their own information.

Alpha-beta objects need user objects by inheriting to store user information. Alpha beta objects are created by game objects. Graph object need user information and alpha beta objects to draw the graph so it needs user objects and alpha beta objects.

#### 4.2. ER Design



**Figure #2 Entity Relationship Diagram**

### 4.3. Database Schemas

- *Administrator Table:*

Field	Type	Null	Foreign Key	References
<b>name</b>	Varchar(30)	No	No	-
<b>password</b>	Varchar(8)	No	No	-

Administrator Table holds information about the administrator user of Mindolog. It holds the basic attributes of the administrator entity which are name and password.

- *Patient Table:*

Field	Type	Null	Foreign Key	References
<b><u>patientID (P.K.)</u></b>	Integer	No	No	-
<b>name</b>	Varchar(30)	No	No	-
<b>password</b>	Varchar(8)	No	No	-
<b>birthDate</b>	Date	No	No	-
<b>gender</b>	Varchar(6)	No	No	-
<b>email</b>	Varchar(30)	No	No	-
<b>phone</b>	Varchar(30)	No	No	-

Patient Table holds information about the registered user of Mindolog. It holds the basic attributes of the patient entity such as name, password, email etc. Primary key of the patient table is patientID.

- *Alpha-Beta Table:*

Field	Type	Null	Foreign Key	References
<b>patientID</b>	Integer	No	Yes	Patient
<b>alphaAmplitude</b>	Float	No	No	-
<b>betaAmplitude</b>	Float	No	No	-
<b>measurementTime</b>	Varchar(30)	No	No	-

Alpha-Beta Table holds information about the brain wave values of the registered users' of Mindolog. It holds the basic attributes which are alpha amplitude, beta amplitude and measurement time. patientID is a foreign key which comes from Patient entity.

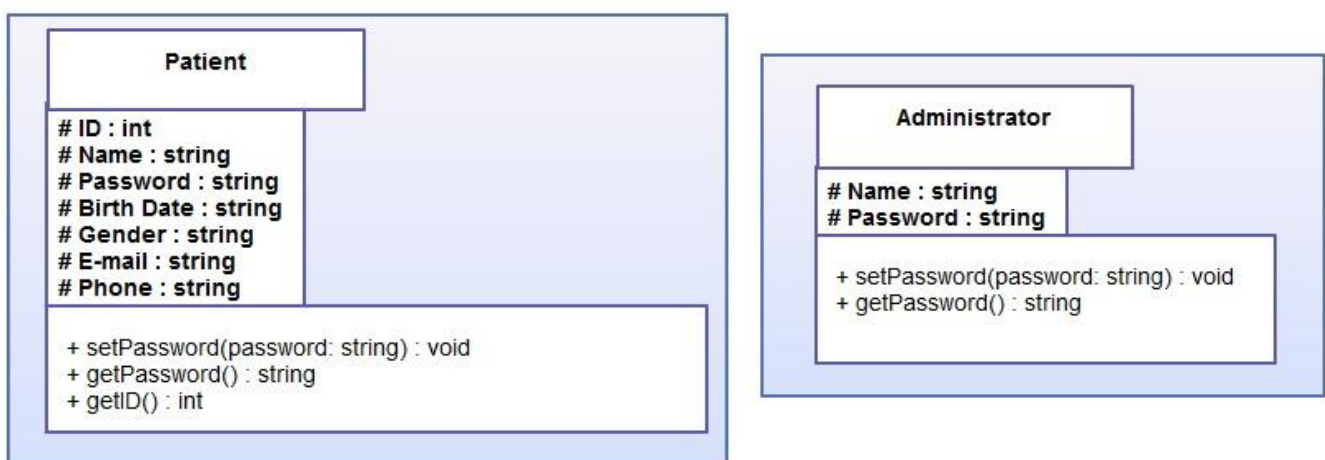
- *Game Table:*

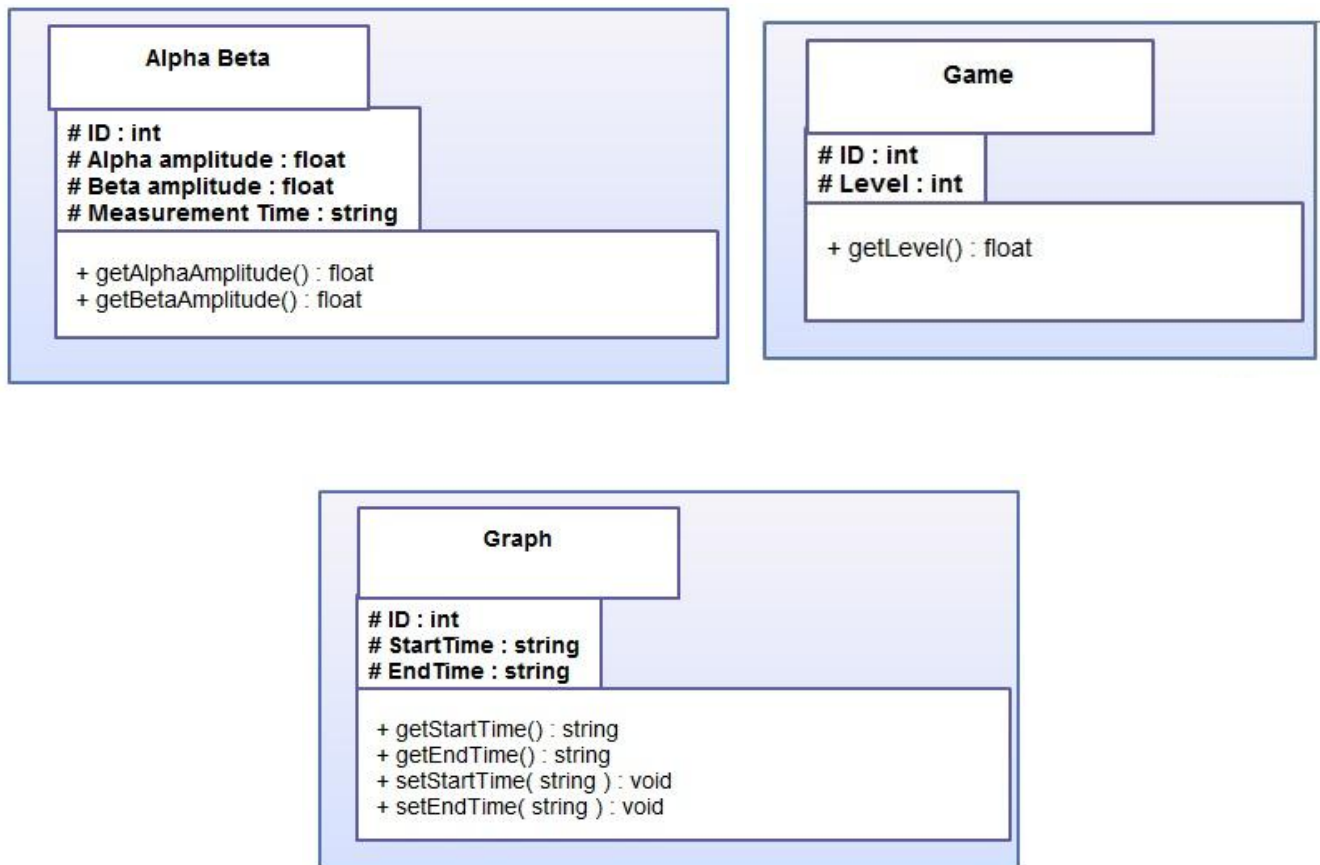
Field	Type	Null	Foreign Key	References
<b>patientID</b>	Integer	No	Yes	Patient
<b>level</b>	Integer	No	No	-

Game Table holds information about the users' game level. patientID is a foreign key which comes from Patient entity.

#### 4.4. Class Diagrams

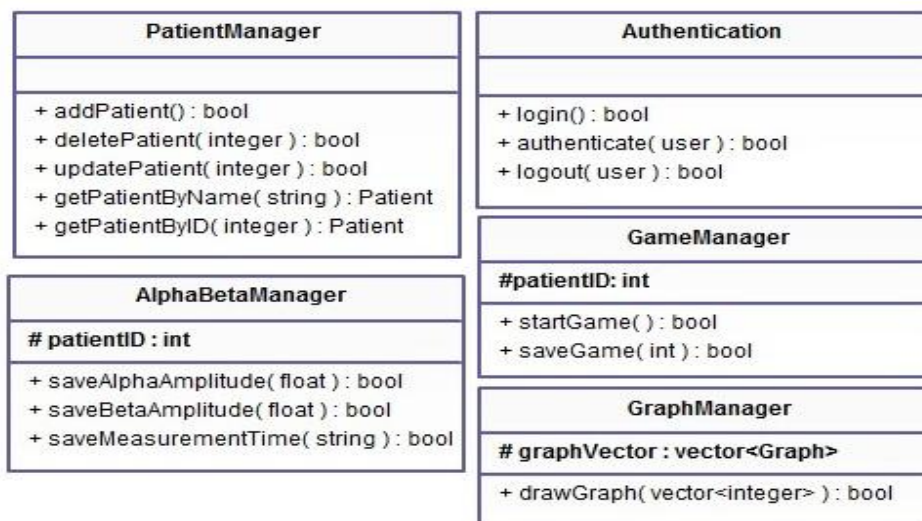
##### 4.4.1. Model Package





*Figure #3 Class Diagrams*

#### 4.4.2. Controller Package



*Figure #4 Class Diagrams of Controller Package*



In this package the first two classes will be used to manage the users of the system. Here “Patient Manager” contains the actions that are related to users which will use the system as patients. This class is intended for the use of the admin, i.e. admin will use the methods of this class, so patient management is done by the administrator. “Authentication Manager” will be used to organize the authentications of the users to the system.

The other three classes seen above will be used for interacting with the database. Alpha-Beta Manager handles storing the wave information of the patient with patientID. The last two classes will also act like an API to publish the data on the user interface. Game Manager starts the game, saves game level for the user with patientID. Graph Manager draws the graphs in the Graph vector.

#### 4.5. Data Dictionary

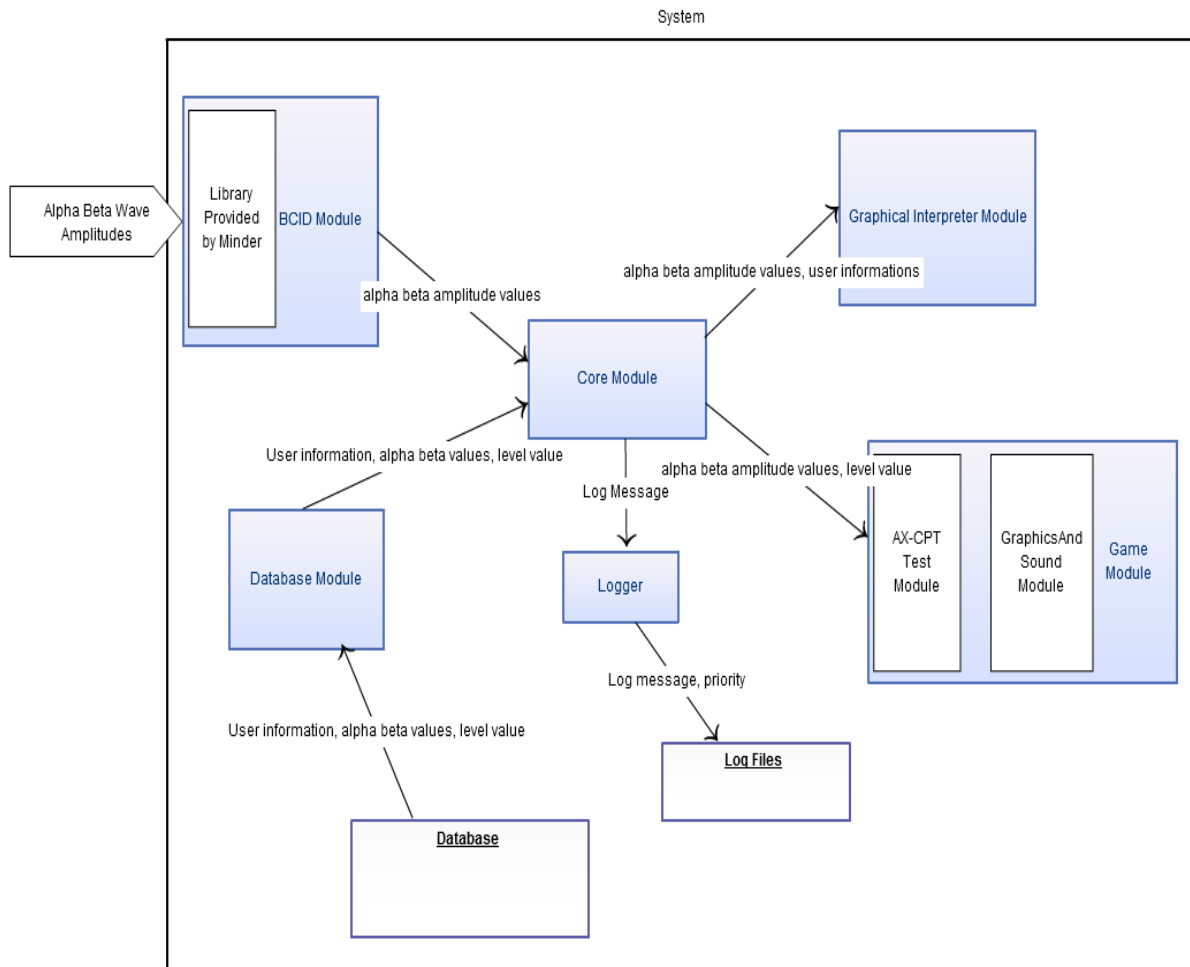
DATA	TYPE	DESCRIPTION
<b>alpha amplitude</b>	float	instantly measured alpha value of the user
<b>beta amplitude</b>	float	instantly measured beta value of the user
<b>birth date</b>	string	birth date of the user
<b>email</b>	string	email of the user
<b>gender</b>	string	gender of the user
<b>level</b>	integer	level of the game lastly reached by the user
<b>measurementTime</b>	string	measurement time of the instantly measured alpha or beta value
<b>name</b>	string	name of the user
<b>password</b>	string	password to login the system
<b>patientID</b>	integer	every patient has a unique id and this data is used to identify patients.
<b>phone</b>	string	phone number of the user

## 5. System Architecture

### 5.1. Architectural design

System will be designed with idea of flow based programming. So we have core module at the center of the system. Core module will be in contact with each other module. There will be 6 modules namely core, BCID, graphical interpreter, database, game and logger. BCID module is responsible from getting alpha beta amplitude values from user's brain waves. It has sub module that is provided us by the company which generates meaningful values from raw data of alpha beta brain waves. Graphical interpreter module is responsible from creating graphs to users see their alpha beta value progress. It uses the alpha beta values from database to create graph. Database module is responsible from connection of database. It will store and get data from database when core module wants. Database module is important because user will want to see his/her progress session by session to decide whether he/she is going well or not. Therefore all alpha beta values should be saved to the database. Game module will create game environment. It needs level value from database, to create environment according to that level value. Game module also have sub module namely info panel. Info panel will show the concurrent alpha beta values graphically. Logger module is responsible from maintenance of the system. All warning/error messages will be saved to the log files. When one wants to see where the problem of the system is, he/she simply looks to the log files to identify the type of problem. Core module is in the middle of the modules that are described. Core module is responsible from the communication of each module to each other. Here is the overview of the modules;





**Figure #5 UML Diagram**

## 5.2. Description of Components

### 5.2.2. Core Module

This module is the most important module of the whole system. All the modules of the system is registered to this module. This module provides a communication interface between the other modules of the system.

Moreover this module includes a tick generator to tick the whole system.

#### 5.2.2.1. Processing Narrative for Core Module

This module ticks all the other modules. The tick generator in it, watches the time and it ticks all the modules at the appropriate time. Tick generator also gets data from the Emotiv EPOC in appropriate time intervals.



This module also provides an interface between the modules. For example when X module wants to call the Y module's method, X calls Y's method through the core module. In addition to this, the data is transferred through the Core Module.

When designing this module we have inspired from a method called Flow-Based Programming. [4][5].

#### *5.2.2.2 Interface Description for Core Module*

The interface of the module provides following properties:

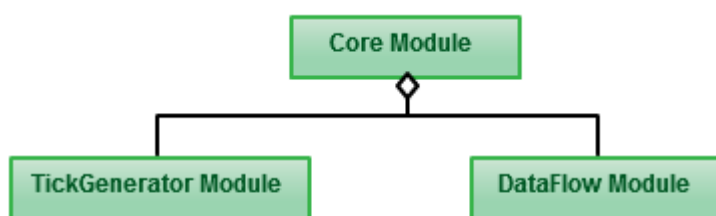
**Time interface:** This interface provides time information to the system. Time interface ticks all the system at the appropriate times.

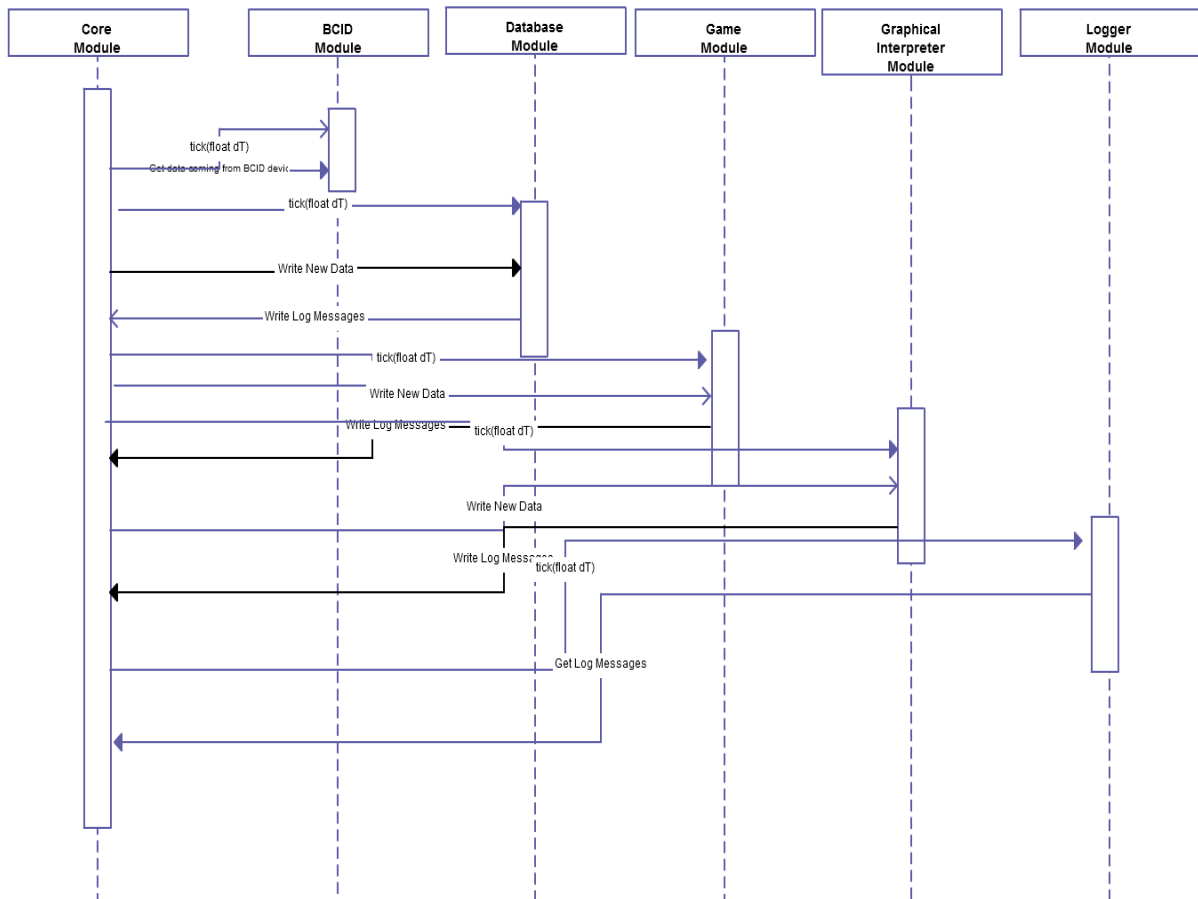
**Data Interface:** This interface provides an interface to the system to carry data between the modules. All the data through the system will flow over this interface.

#### *5.2.2.3 Processing Detail for Core Module*

This module sees all the other modules as blackboxes and this module. All the communications is done over this module. In other words the modules cannot communicate with each other they only communicate with the core module. All the information that needs to be transmitted somewhere is hold in the core module. So when a module wants data to be transmitted somewhere it writes the data to the appropriate port of the core module. Then core module sends the data to the appropriate module.

#### *5.2.2.4 Dynamic Behavior*





**Figure #6** Core Module Sequence Diagram

### 5.2.2. BCID Module

BCID module is designed to get alpha and beta values in to the system. With this module user's alpha and beta waves amplitudes are implemented to the system. In other words, with this module system becomes a system that gets inputs from user's brain. This part of the system constitutes the most powerful property of the system.

#### 5.2.2.1. Processing Narrative for BCID Module

BCID module is responsible from getting input from user's brain. It should give double values that have meaning in terms of alpha beta waves amplitudes. These values should be passed to the system once in 250ms. This module passes the incoming alpha beta amplitude values to the core module.



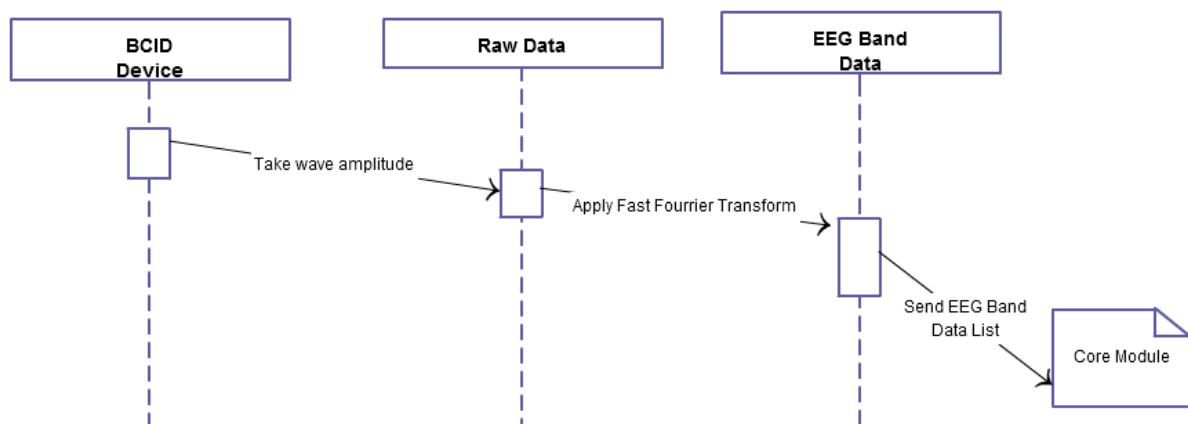
### 5.2.2.2 Interface Description for BCID Module

BCID module has interface with core module and BCID device. BCID module sends alpha beta amplitude values to the core module. Moreover it gets alpha and beta amplitude values from BCID device. This device generates alpha beta theta and delta values from user's brain waves. EEGBandData type of structure holds all four values in terms of double type. Layer between the device and our module (which is provided by the company "Minder"), creates these data values from device.

### 5.2.2.3 Processing Detail for BCID Module

BCID module should get alpha and beta values from device. Our module is subscribed to the library that is provided by the company "Minder". It gets the data for once in 250ms (since library implemented requires at least this time to get data). It gets a list contains 36 elements consists of alpha beta amplitude values. These values are passed to the core module. With this implementation, concurrent data flow is established.

### 5.2.2.4 Dynamic Behavior



**Figure #7 BCID Module Sequence Diagram**

### 5.2.3. Game Module

This module provides the required routines for the game part of the project. It renders a basic 3D world according to the inputs coming from the keyboard and the EEG device.

This module consists of two sub modules. Digit-span test module and GraphicsAndSounds module.

AX-CPT Test Module handles the AX-CPT test related tasks. It keeps the test details and provides appropriate information to the GraphicsAndSounds module.

GraphicsAndSounds module uses jMonkey Game Engine to handle some important tasks, such as rendering, playing sounds, a GUI for the player, etc.[6][7]

#### *5.2.3.1. Processing Narrative for Game Module*

This module achieves its tasks whenever it is ticked by the Core Module. In every tick of this module, it calculates the position of the game object according to the coming inputs - alpha and beta waves, keyboard inputs - to the module. It plays appropriate sound according to the game state. Moreover an AX-CPT test is also running concurrently.

#### *5.2.3.2 Interface Description for Game Module*

The interface of the module provides following properties:

Tick Interface: This interface ticks the whole module. The whole module needs differential time information to do all the calculations correctly so that the passed time after the met, for example position of the game object. This method ticks all the sub modules accordingly.

Alpha Wave Interface: This interface sets the alpha wave information for the current tick.

Beta Wave Interface: This interface sets the beta wave information for the current tick.

Level Interface: This method saves the level of the user to the database.

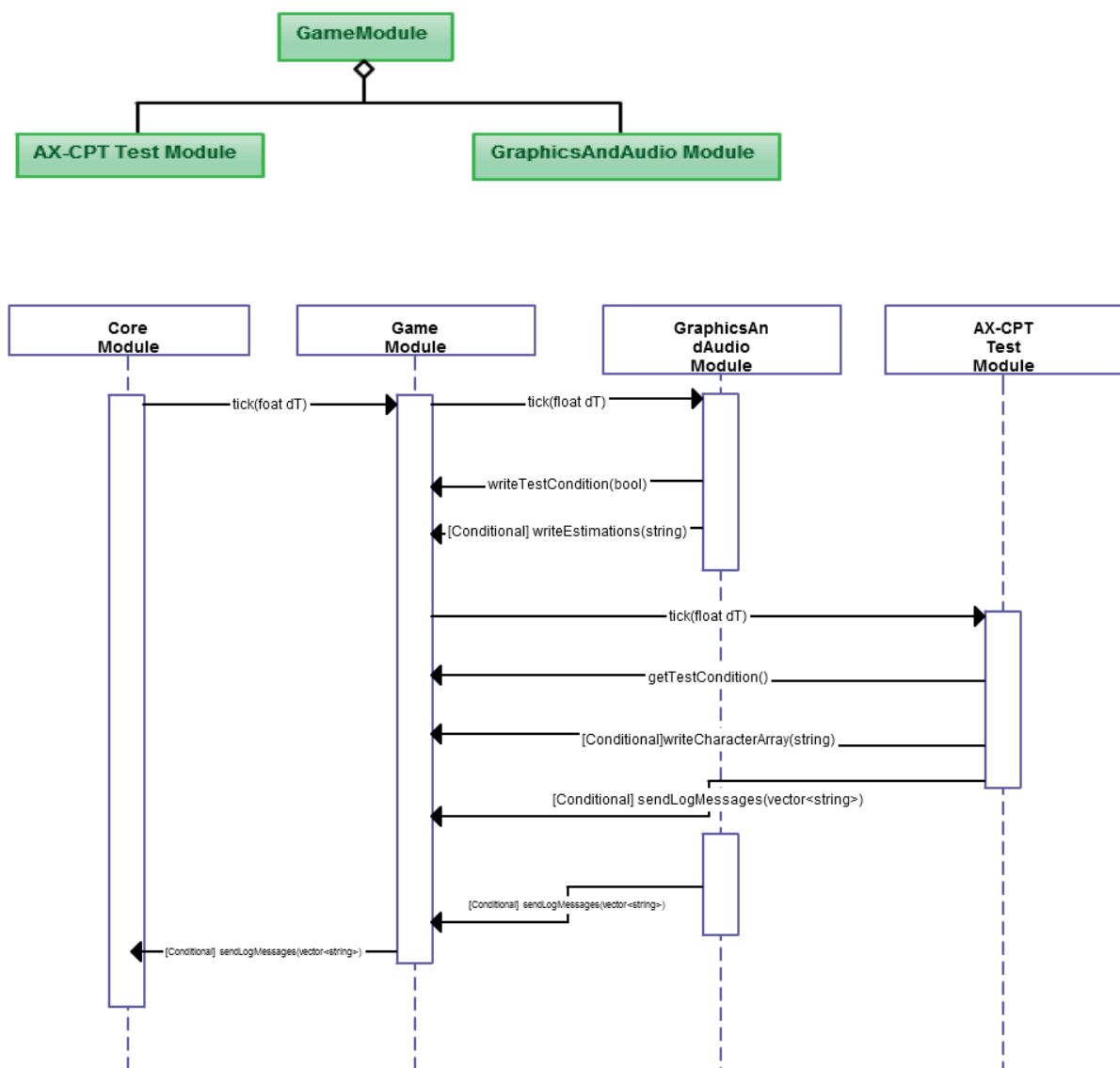
#### *5.2.3.3 Processing Detail for Game Module*

Whenever a tick comes the whole module starts to accomplish required tasks. The tick method of the main modules ticks the two sub modules accordingly.

AX-CPT test module generates an character array in every X seconds. X is determined according to the level of the user. Whenever an array is generated the whole array is sent to the GraphicsAndSounds module to be rendered. After all the estimations done by the user, the estimation array is sent to AX-CPT test module and required calculations are done. According to the estimations the module calculates the results of the user.

GraphicsAndAudio module handles the much of the tasks of the Game Module. jMonkey graphics engine is the most important part of this module. In this module there is a list of the objects which can be both rendered and played. In every tick all the objects in the list rendered and played accordingly and a check operation is performed to determine whether the AX-CPT test array is ended. If it is ended the estimations will be sent to the AX-CPT test module else the number estimated by the user is pushed back to the estimations array.

#### 5.2.3.4 Dynamic Behavior



**Figure #8** Game Module Sequence Diagram

#### *5.2.4. Graphical Interpreter Module*

Graphical Interpreter module is designed give users a chance to see their alpha/beta values session by session. Users can interpret their data with line charts. By looking to the graph, one can see his/her progress in terms of alpha beta values which mean progress in attention deficiency. Moreover with this module administrators (doctors in our project) can see all patients' alpha/beta values and compare them. Since in the game screen, there is concurrent alpha beta values are shown this module is also drawing concurrent alpha/beta values.

##### *5.2.4.1. Processing Narrative for Graphical Interpreter Module*

Graphical Interpreter module is responsible from the functionality of showing data to the patient of the system and drawing concurrent alpha/beta values. It should give alpha/beta values distinctly by time intervals. Therefore one can easily see whether he/she is progressing or regressing. Administrators can compare patients' progress with more than 1 couple of line charts. Moreover values got from BCID module should be given to the this module to draw graph of concurrent alpha and beta values in game screen.

##### *5.2.4.2. Interface Description for Graphical Interpreter Module*

Graphical Interpreter module has interface to get alpha, beta values for the patient or patients. Module wants the data for user with user id. Because it needs data to draw line graph of the patient, it should reach the database to get alpha beta values of each patient. Moreover it needs all data got from BCID module to draw line chart of concurrent alpha and beta values dynamically. Since our product will be based on Flow Based Programming, all interactions with database are done through core module.

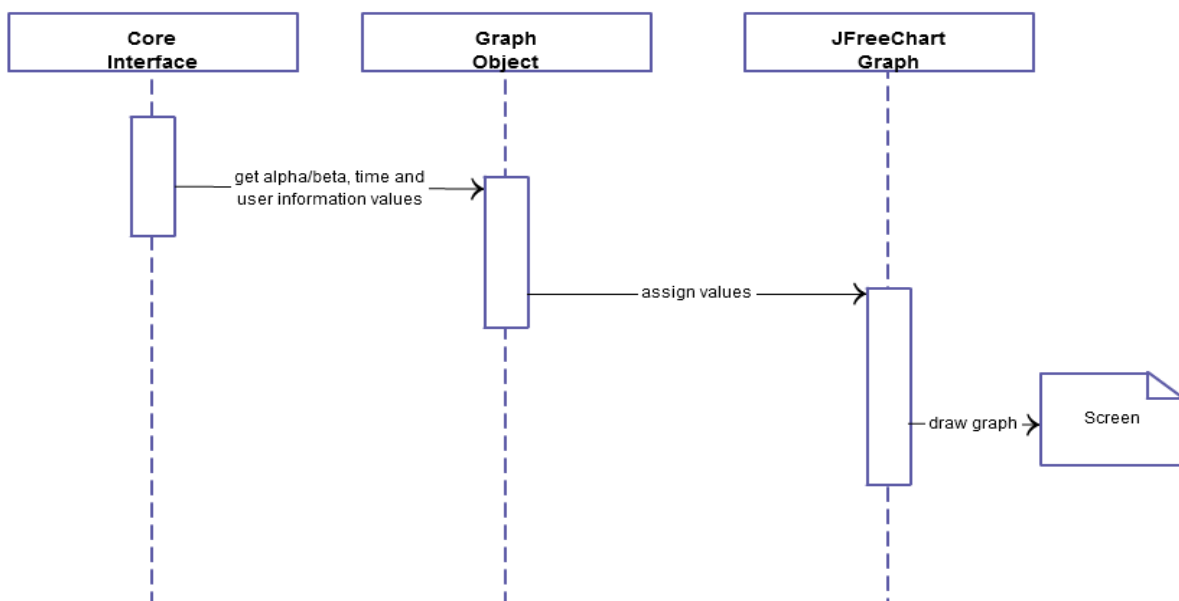
##### *5.2.4.3. Processing Detail for Graphical Interpreter Module*

Graphical Interpreter module should draw line graph for users. Graph can include one user's data or more than one user's data. Basically, it creates a line graph object. It wants the alpha beta values and time interval values from database. Then, it assigns them to the relevant variables of line graph object. Moreover it gets the user's name from database and assigns it to the graph's title. For the case that is shown more than one user's data, lines are assigned to the different colors. Mapping from colors to the users are shown in graph frame. So graph

module needs each user's information. After getting each user's information, they are assigned to the relevant variables of line graph object.

Second functionality of the graph interpreter module is to draw graph of alpha beta values that got from BCID module to show to the user his/her concurrent alpha and beta values. It follows the same procedures except it does not need time interval. Because it is just a line graph that grows with incoming data. Since it is concurrent it is not necessary to specify time.

#### 5.2.4.4 Dynamic Behavior



**Figure #9** Graphical Interpreter Module Sequence Diagram

#### 5.2.5. Database Module

##### 5.2.5.1. Processing Narrative for Database Module

This module is the interface between the database management system and the core module. Other modules use database over this module by the connection of core module. In other words, database only communicates with the core module, and the whole system uses database from the core.

##### 5.2.5.2. Interface Description for Database Module

Interface of this module is divided into two packages, namely the Modal Package and the Controller Package. Class diagrams of these packages are presented in the *Class Diagrams*



section of *Data Design*. Model Package consists of Patient, Administrator, Alpha-Beta, Game and Graph classes while Controller Package consists of Patient Manager, Authentication, AlphaBetaManager, GameManager and GraphManager.

#### *5.2.5.3. Processing Detail for Database Module*

Model package consists of classes that are used for interacting with the database. They represent the entity structure of the database. When data is requested from the database, related class instances will be created for related database tables and will be filled with the requested information to be used for the requested execution.

Classes defined in the controller package will be used for communicating with the database, i.e. retrieving/updating/deleting/inserting data. The functionality beneath the user interface will be realized by means of this group of classes.

#### *5.2.5.4. Dynamic Behavior*

Dynamic behavior of this module is explained in the *Data Design* section.

#### *5.2.6. Logger Module*

Logger module is designed for the maintenance of the whole system. By looking at log files, one can see where the problem is. All error/warning messages are kept in files according to dates. It gets various types of error/warning messages. Moreover all messages have property degrees. Logger module is in communication with core module.

##### *5.2.6.1. Processing Narrative for Logger Module*

Logger module is responsible from maintenance of the system. It should give information to the technician about system. It is responsible to detect which message has higher priority. To illustrate, BCID connection problem should have more priority than login problem.

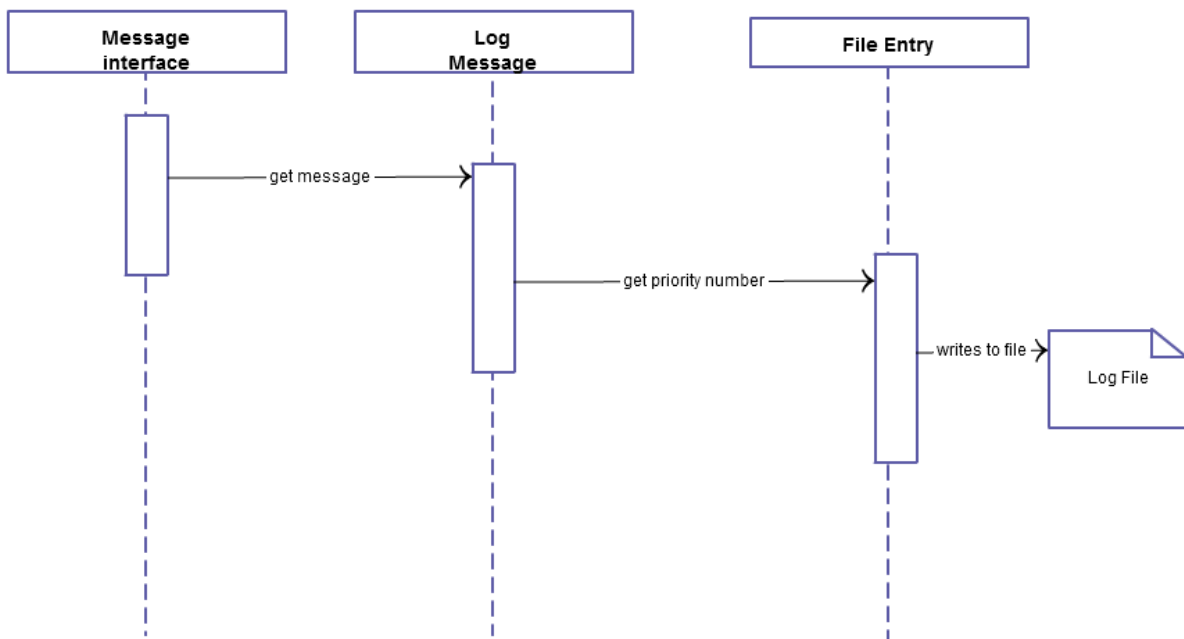
##### *5.2.6.2. Interface Description for Logger Module*

Logger module has interface to get data from core module. Core module sends error/warning messages to the module. Logger module has interface with file I/O system. It sends strings and priority information to the file output to write them to the file.

### 5.2.6.3. Processing Detail for Logger Module

Logger module should save error/warning messages to the log files as described earlier. The module should find the priority of the message whenever module is activated with string (error or warning message). Because some errors could be more important than the others, module needs the priority of the message. When priority of message is found, it is saved to the file with message.

### 5.2.6.4 Dynamic Behavior



**Figure #10** Logger Module Sequence Diagram

## 5.3 Design Rationale

We choose 6 part composition because we need 6 main functionalities. These are getting alpha beta values from device, drawing graph from saved alpha beta values, log saving functionality, database communication, and game. 1 extra module created for core module. This type of design gives a chance to add extra modules to the easily after building whole system. Since supporter of us “Minder” stated that there will be more than one games if it will be done accurately, we choose flow based programming. With this type of implementation, we can easily add or remove modules. Other alternative is just connecting each module to the others without having any core module. However it will be costly to add

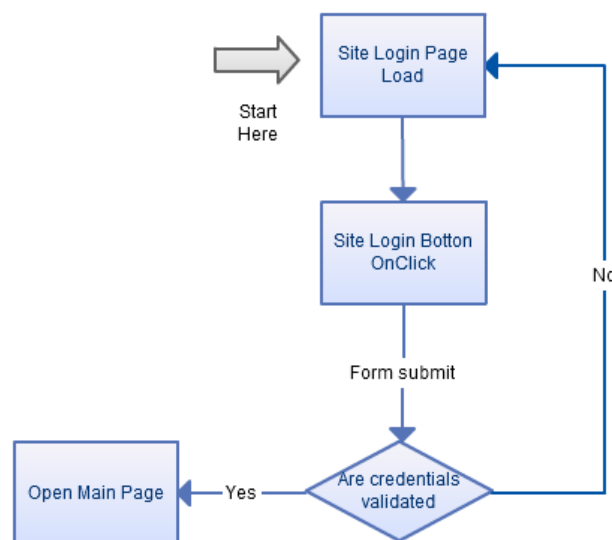
new module to the system when it will be finished. Therefore, we chose this type of module composition.

## 6. User Interface Design

### 6.1. Overview of User Interface

#### 6.1.1. Login Screen

When the program gets started, login screen will appear. In login screen, there will be two text boxes to be filled, namely user name and password, and one button namely login in order to enter the system. The system can assign the cookie information by this login screen. If the user is administrator, the cookie contains the information and permissions of the administrator. If the user is the patient, the cookie contains his/her permissions and information.



**Figure #11** Login Screen User Diagram

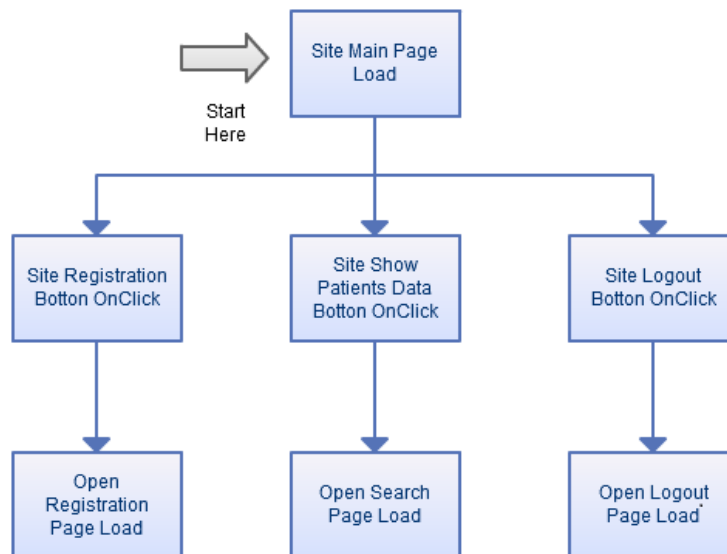
#### 6.1.2. Main Screen

There will be two types of main screen which is due to two types of users. The first main screen belongs to administrator user and the second main screen belongs to patients.

##### 6.1.2.1. Administrator Case

In administrator's case there will be three button namely, registration, show patients data and logout. Registration button will send user to the registration screen which is related with

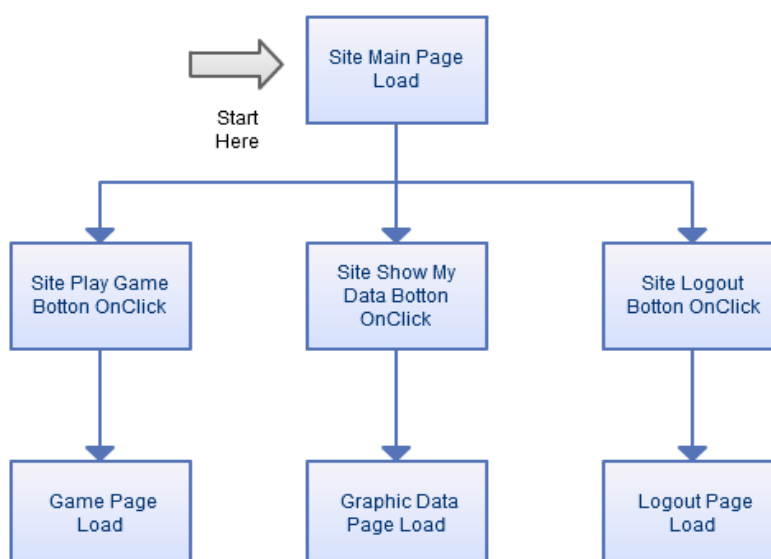
add/edit/delete patient functions that are described below. Show patients data button will send user to the search screen. And logout button will send the user to the login screen.



**Figure #12** Main Screen for Administrator User Diagram

#### 6.1.2.2. Patient Case

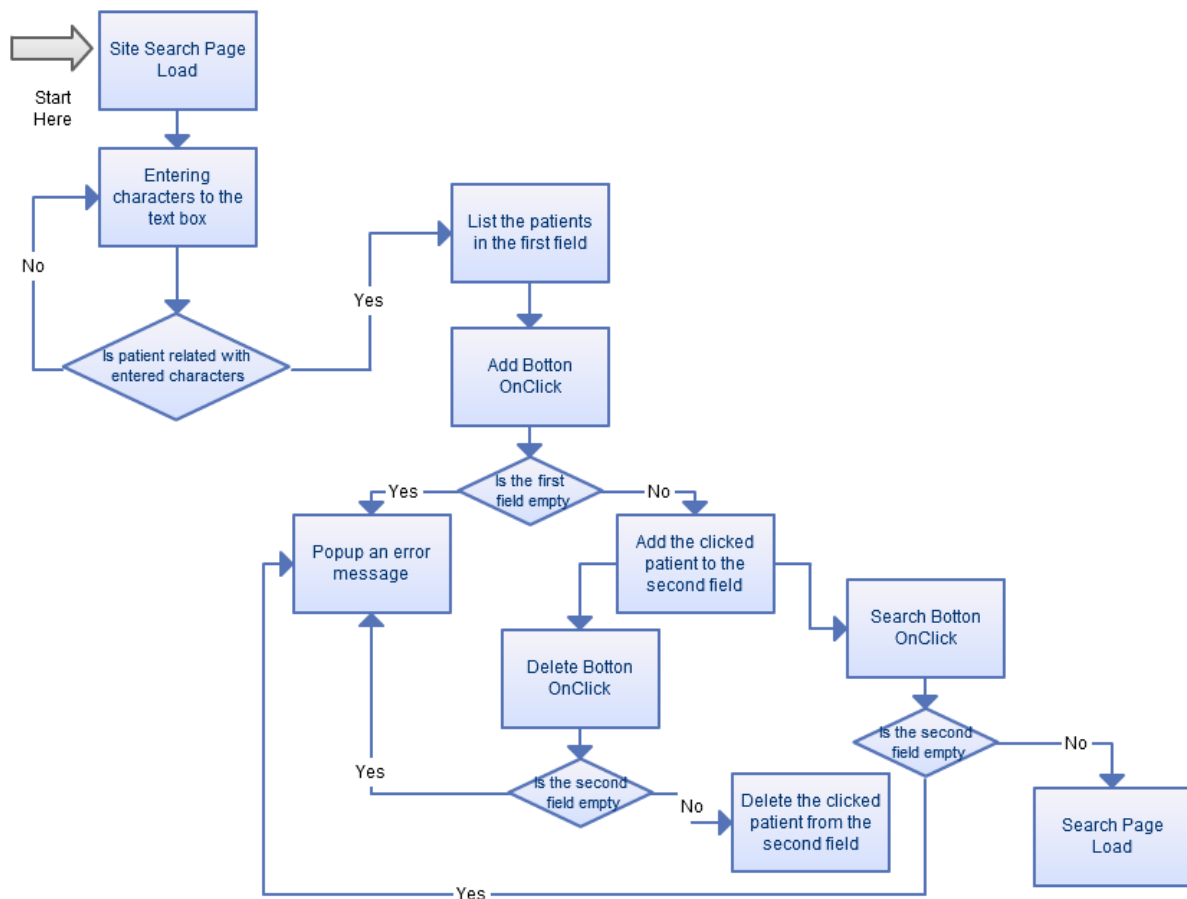
In patient's case there will be three buttons namely, play game, show my data and logout. As can be understood easily, play game will send user to game screen, show my data button will send user to graphic data screen and logout button will send user to login screen. Show my data button is related with showing user data functions that are specified in part 4.



**Figure #13** Main Screen for Patients User Diagram

### 6.1.3. Search Screen

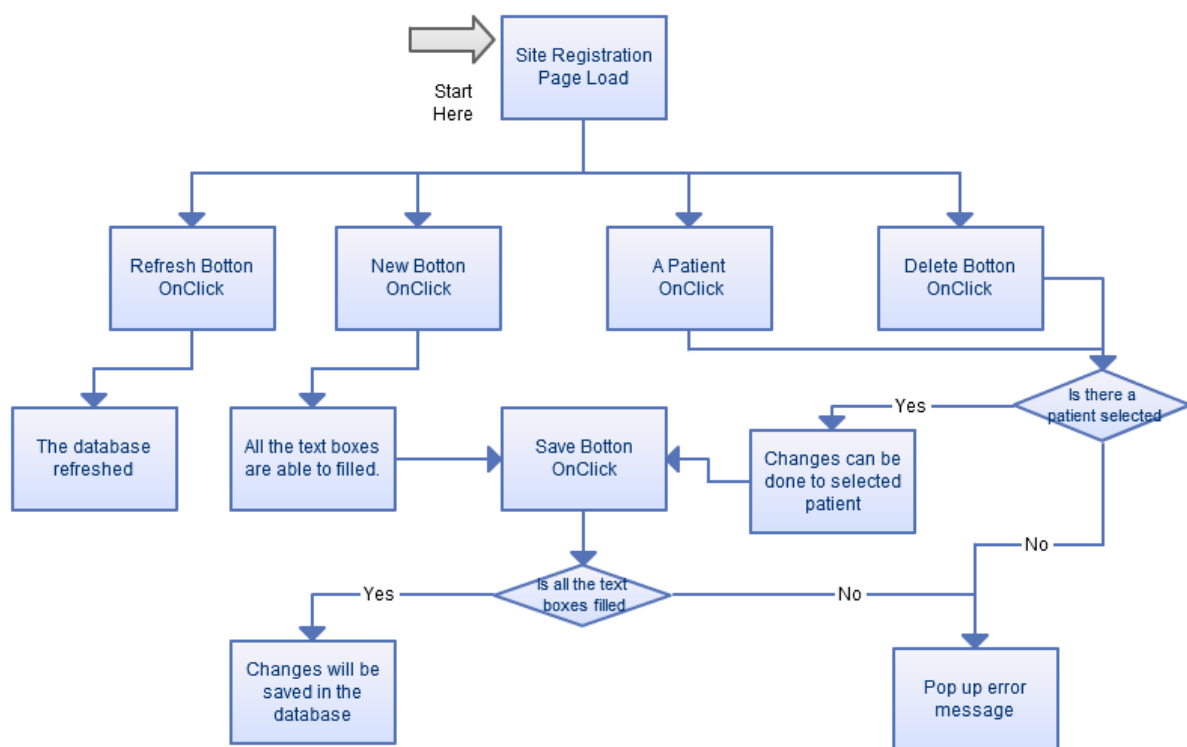
There will be one text box to be filled, 2 list fields and three buttons. While entering characters to the text box, all patient entries will be displayed *related with the entered characters, in the first list field concurrently*. This functionality will use searching *patients' function*. When the user select the patient's entry from the first list field and clicks to the add button, it will be moved to the second list field. If user does not select anything, pop-up screen will appear as warning including "no entry is selected to be added". When user select patient's entry from the second list field, and clicks the delete button, the entry will be moved to the first list field. If user does not select anything, pop-up screen will appear as warning including "no entry is selected to be deleted". Show button will send user to the graph data screen with using patients' data which are in second list field.



**Figure #14** Search Screen User Diagram

#### 6.1.4. Registration Screen

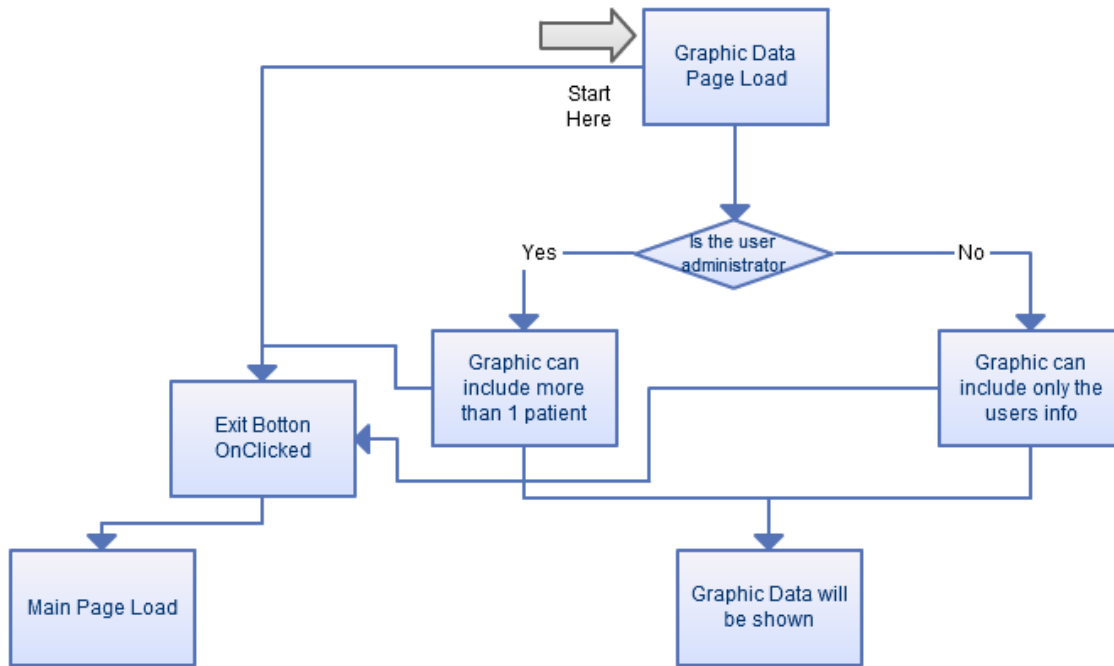
It will be available to administrators only. In this screen, there will be a table which is connected to database with read functions. All patients' entries will be displayed in the table. There will be 4 buttons namely, add, delete, refresh and save. As can easily be understood, new button will add new patient, delete will delete the selected patient, refresh will refresh the table entries, and save will save the modifications that is done in *patients information* to database. If user selects one entry from table all data will be filled to the text boxes to give a chance to edit entries.



**Figure #15** Registration Screen User Diagram

#### 6.1.5. Graphic Data Screen

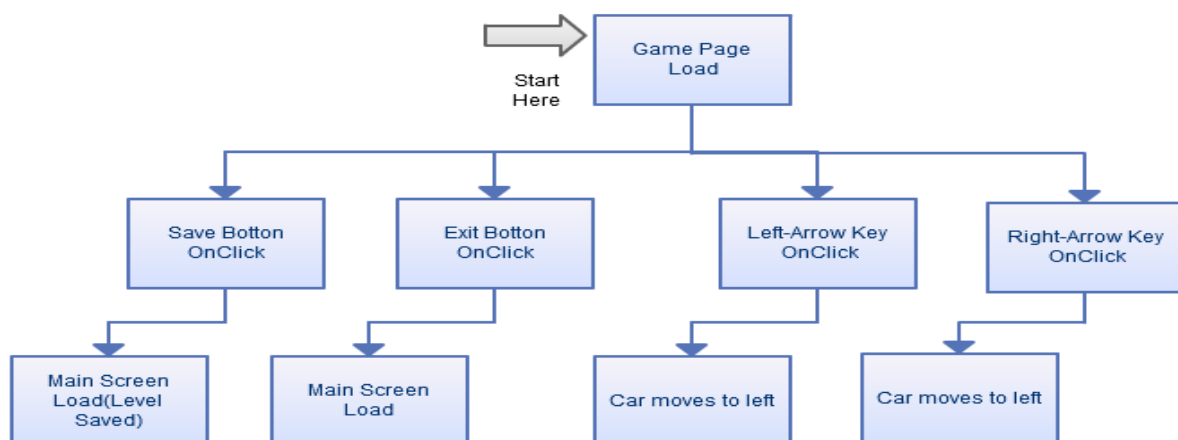
In this screen there will be a graph that is drawn according to selected patient's id before. For administrators, there is a chance to draw more than one patient's data. For patient users, there will be only their data in graph. There will be two buttons namely, exit and logout. Exit button will send user to the main screen.



**Figure #16** Graphic Data Screen User Diagram

#### 6.1.6. Game Screen

In this screen, there will be two internal frames and two buttons. First frame will show the simultaneous alpha and beta values of user's brainwaves that is taken from emotive computer brain interface device. Concurrently, in the second internal frame, the game will continue. First button (exit) will send user to the main screen. Second button will save the level of game, which is being played, to the database with write functions and return the main screen.



**Figure #17** Game Screen User Diagram



## 6.2. Screen Images


### 6.2.1. Login Screen

### Member Login

Username or email:

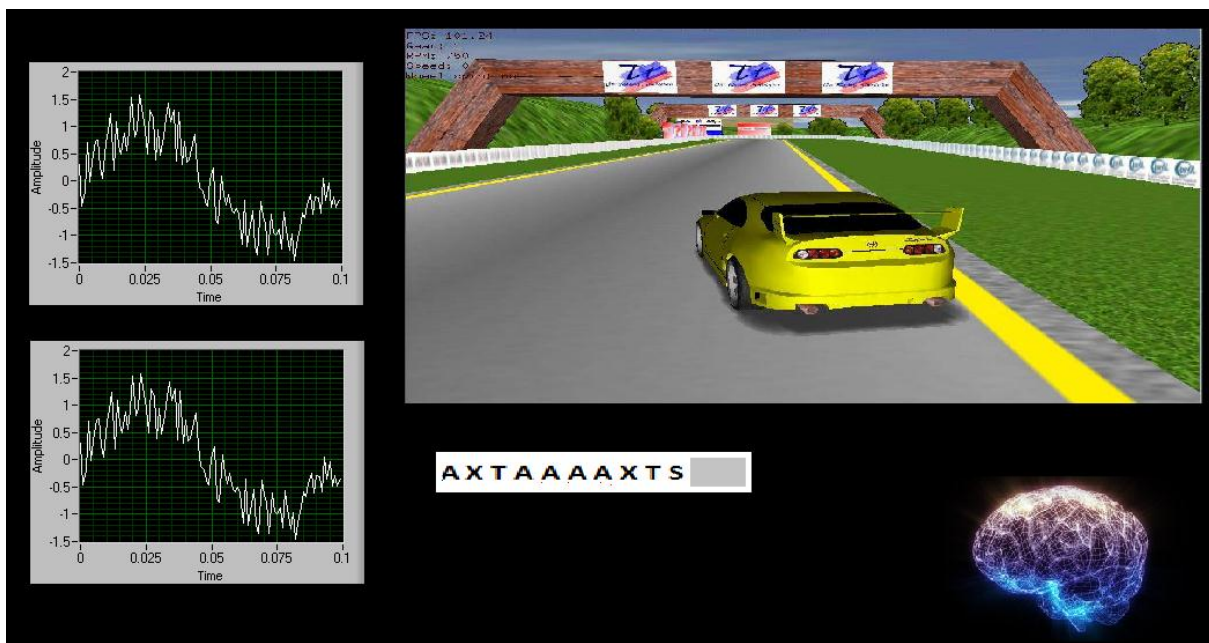
Password:

☐ Remember Me



*ScreenShoot #1*

### 6.2.2. Game Screen



*ScreenShoot #2*



### 6.2.3. Registration Screen

Patient id	Patient name	Patient surname	Patient age
1		Özer	22

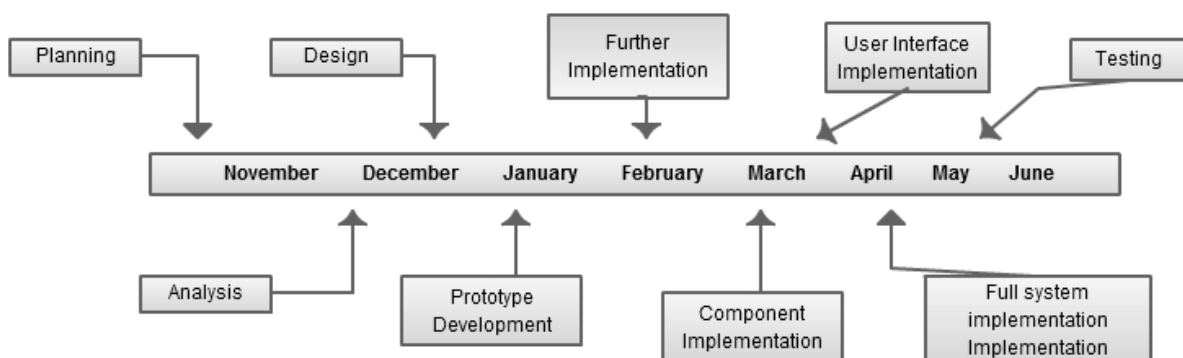
Patient id:   
 Patient name:   
 Patient surname:   
 Patient age:

### ScreenShoot #3

### 6.3. Screen Objects and Actions

On all of the screens, there will be Exit bottom object which's action is returning the main page. And also there will be Log-Out bottom object which's action is to return the login page. In Registration Screen there will be new delete refresh and save bottom objects which explained in 6.1.4. And also there will be play game and show my graphic and search/compare bottom objects are included by the main pages. All the objects are specified in 6.1 in detail.

## 7. Time Planning (Gantt Chart)



**Figure #18** Estimation Diagram



### 7.1. Term 1 Gantt Chart

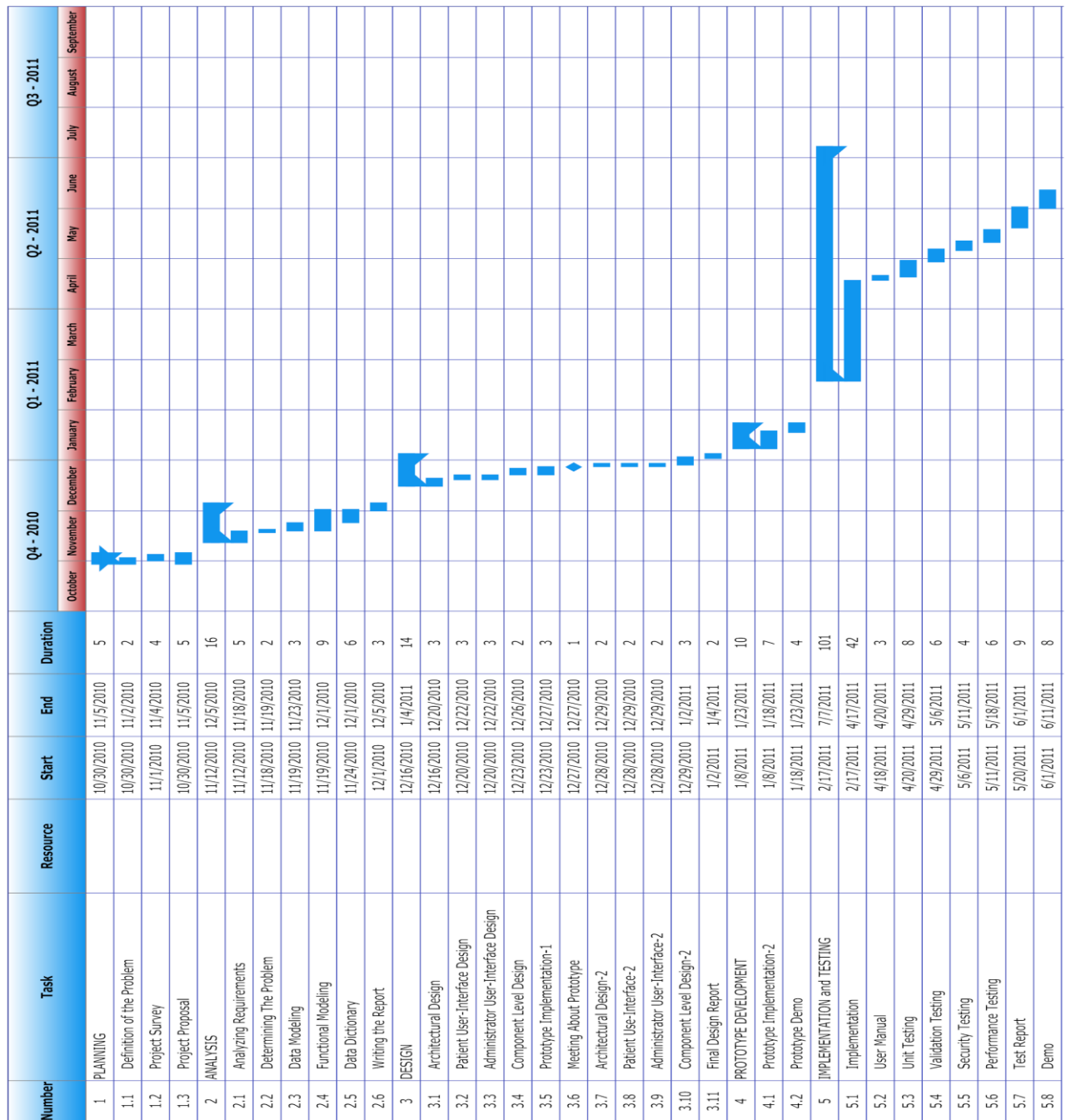


Figure #19 Gantt Chart for Term I



## 7.2. Term 2 Gantt Chart

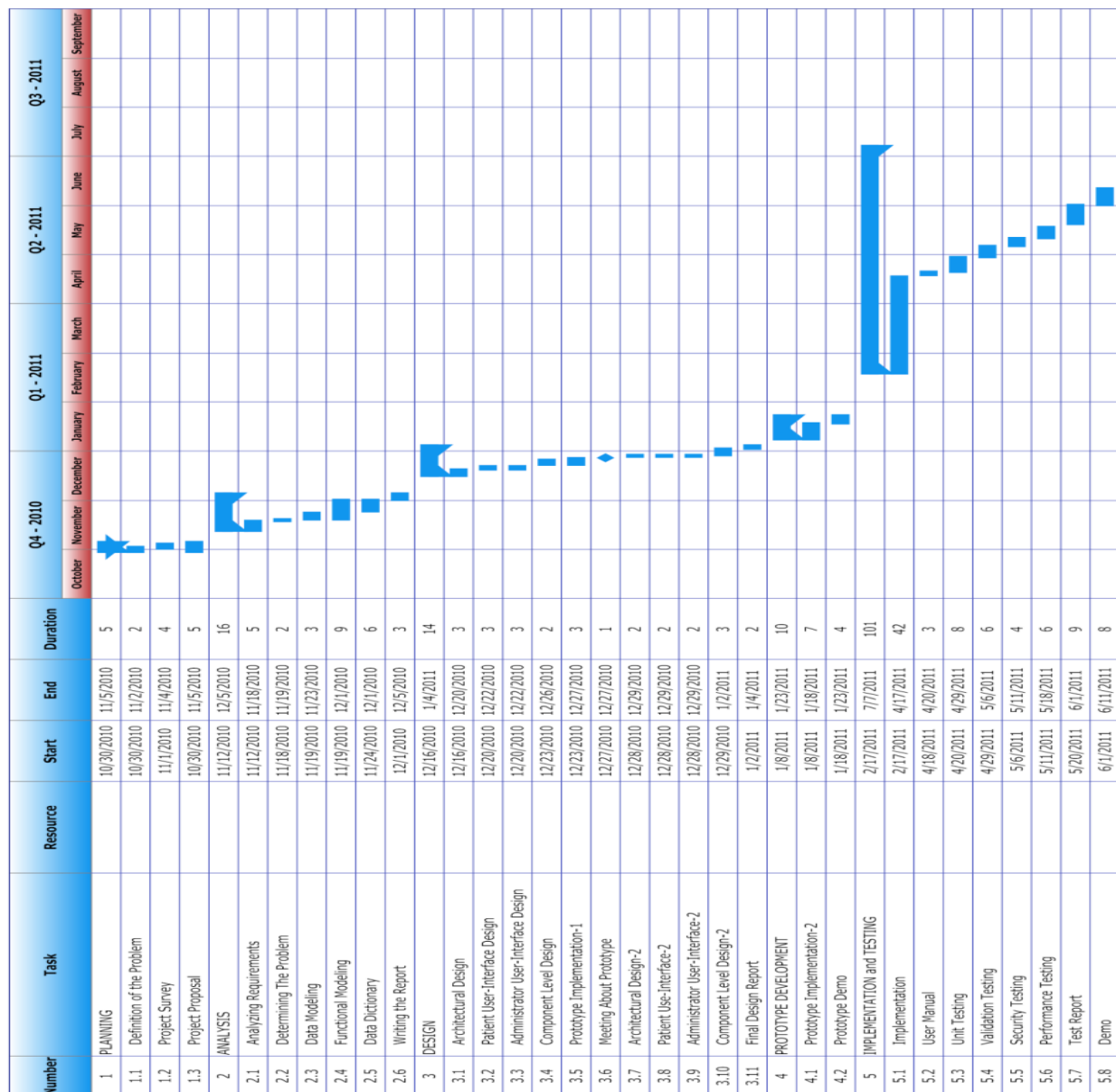


Figure #20 Gantt Chart for Term II

## 8. Libraries and Tools

**Eclipse:** It is an IDE that we are going to use for design. It is really compatible with Java programming language, and with the library provided by Minder to use BCID. Therefore we have chosen this IDE to the design the system.

**Emotiv BCID:** It is a brain computer interface device which gets the alpha, beta, theta etc. values with 50ms period (as default). Our sponsor company provides this device.

**Emotiv Test Bench:** It is a program to see data coming from the BCID. The program controls whether the points that should give alpha/beta values, is correctly placed to users head. Moreover, it concurrently shows the incoming values of data.

**Java:** Because of the libraries provided of use, we are going to use Java.

**jFreeChart:** It is a library to draw various types of graphs (line, pie chart etc.). It is available for Java programming language and distributed with GPL.

**jMonkey:** It is s a game engine made especially for modern 3D development. The entire project is based on Java for high accessibility. Advanced graphics are supported through OpenGL 2 via LWJGL.

**jopenGL:** It is similar to OpenGL in C/C++. It is extended from OpenGL for Java language. As mostly known, jOpenGL gives us a chance to design game environment.

**swing:** It's a library to design frames, windows in given environment. In other words, it is a library to design GUI.

## 9. Conclusion

In conclusion, Initial Design Report for Mindolog gives the definition, purpose and scope of the project. The possible design and other constraints that can be faced are explained. The tools and the libraries that will be used during developing the project are decided. Data flow models, class diagrams, interface features, entity relationship diagrams, possible use cases are given within the document. We have explained the works that we have done so far and within the schedule we give the future work to be done.