Implementation Document

for

Elysium

Version 1.0

Prepared by

Group: 13 **Group Name: aleph-7**

Aditi Khandelia	220061	aditikh22@iitk.ac.in
Gottupulla Venkata Aman	220413	gvaman22@iitk.ac.in
Mahaarajan J	220600	mahaarajan22@iitk.ac.in
Kushagra Srivastava	220573	skushagra22@iitk.ac.in
Sankalp Mittal	220963	sankalpm22@iitk.ac.in
Aditya Jagdale	220470	jagdale22@iitk.ac.in
Wattamwar Akanksha	221214	akankshab22@iitk.ac.in
Ritesh Baviskar	220286	baviskars22@iitk.ac.in
Arush Upadhyaya	220213	arushu22@iitk.ac.in
Animesh Madaan	220145	manimesh22@iitk.ac.in

Course: CS253

Mentor TA: Mr. Sarthak Neema

Date: 15 March 2024

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
1.0	Group 13	First draft of Implementation Document	18/03/24

1 Implementation Details

Programming languages and framework

1) Frontend:

Using HTML, CSS, and JavaScript for the frontend has many advantages. They're supported by all browsers, have numerous tools and libraries, and integrate well with any backend framework. Plus, they're lightweight, resulting in faster system performance and they also allow the frontend to change without the user having to manually reload.

We opted for the Vite.js framework using the React.js template due to its numerous advantages like:

1. *Ease of learning and adaptability.*
2. *Offers features like reusable components and virtual DOM, giving rise to well-structured and efficient code.*
3. *Includes useful tools like react-router for effectively managing multi-page websites like ours*
4. *The vite.js framework is focused on speed and performance offering features like hot reload during development.*
5. *Helps to maintain uniformity across work done on different devices.*

2) Backend:

For the backend of our web-app, we are using Node.js. This is due to its numerous advantages like:

1. *The execution time and runtime is very fast because it has been built on Google Chrome's V8, a C++ based open source JavaScript engine.*
2. *A wide range of functionalities available in the form of bundles in the Node Package Manager (npm), these can be imported with ease at any time, during the development process.*
3. *Node.js is totally asynchronous in nature allowing us to build real time and data intensive web applications.*
4. *Node.js is open-source and it is easy to develop projects using Node.js if one is familiar with Javascript.*
5. *Many prominent software companies like Netflix and Paypal use Node.js for their websites.*

Additionally to route our API's we have used Express.js library. The advantages of using Express.js are:

1. *Express.js provides a straightforward and minimalist framework for building web applications and API's with powerful routing mechanisms.*

2. *It makes application development fast, easy and simple due to its highly supportive open source community.*
3. *Express.js is known for its lightweight and fast performance. It is built on top of Node.js, leveraging its asynchronous and event-driven architecture. This results in efficient handling of concurrent requests with the support of I/Q request handling.*
4. *It is highly flexible allowing us to define routes based on HTTP methods and URLs.*

3) Databases:

For the database we have used MongoDB, due to its marked advantages such as:

1. *MongoDB is a NoSQL database, which means it does not require a predefined schema. This flexibility allows developers to store data of different structures within the same collection, making it ideal for applications with evolving data requirements.*
2. *MongoDB is designed to scale horizontally by distributing data across multiple servers. This allows for seamless scaling as application data grows, without significant changes to the database architecture.*
3. *MongoDB employs various optimization techniques such as indexing, sharding, and replication to deliver higher performance in database operations such as efficient querying and retrieval of data.*
4. *MongoDB stores data in flexible JSON-like documents called BSON (Binary JSON). This allows us to represent complex hierarchical relationships and nested data structures easily.*

Codebase

Github Repository : https://github.com/aleph-7/elysium

Code Structure

The project is divided into two parts, the codebase and the databases, based on the model view architecture format.

Codebase

The server directory contains the following major files/folders:

- *model folder contains the schema for all the databases to create the models required for making calls to the database*
- *databases folder contains the code to connect the databases to the application via a https connection*
- *server.js file that acts as a server and handles all the requests that are made from the frontend sends*

The entire organisational structure is as follows:

```
|----node_modules
|----public
|----server
|----src
```

The structure of the backend is as follows:

```
|----server
     |---- databases
          |---- bookingsDB.js
          |---- contentDB.js
          |---- courtDB.js
          |---- leaderboardDB.js
          |---- userDB.js
     |---- middleware
          |---- check_auth.js
     |---- models
          |---- bookingsDB.js
          |---- contentDB.js
```

```
          |---- courtDB.js
          |---- leaderboardDB.js
          |---- userDB.js
     |---- node_modules
          |---- routes
          |---- algorithms
               |---- booking.js
          |---- apply_workshop,js
          |---- auth.js
          |---- admin.js
          |---- leaderboard.js
          |---- tutorials.js
          |---- user.js
          |---- workshop.js
          |---- server.js
```

The src directory contains all the frontend files and is divided in the following form:

```
|---- assets
|---- admin
     |---- assets
     |---- attendance.jsx
|---- counsellor
     |---- screens
          |---- button.jsx
          |---- page1.jsx
          |---- page2.jsx
          |---- page3.jsx
          |---- table.jsx
     |---- CardStack.jsx
     |---- Counsellor.jsx
|---- Dashboard
     |---- Coach
          |---- pages
               |---- checkEnrollment
               |---- postWorkshop
               |---- reserveCourt
               |---- statistics
```

```
          |---- Components
                |---- buttons
                |---- datetime
                |---- home
                |---- sidebar
                      |---- SidebarData.js
                      |---- Sidebar.js
                |---- subscribe
                |---- table
                |---- taskbar
                |---- tutorials
                      |---- Tutorials.js
                      |---- items
                |---- viewshow
          |---- CardStack.jsx
          |---- Coach_Dashboard.jsx
     |----Header
     |----Yoga_Instructor
          |----Yoga_Instructor.jsx
          |----CardStack.jsx
          |----screens
                |----PostBlog
                |----PostSession
                |----check_enrollment
                      |----check_enrollment.jsx
                      |----table.jsx
     |---- Gym_Instructor
          |---- assets
          |---- Button
          |---- table
          |---- pages
                |---- check_enrollment
                |---- reserve
                |---- statistics
|---- Gym
     |---- components
          |---- Home
          |---- Subscribe
          |---- Tutorials
```

```
          |---- Viewshow
          |---- assets
          |---- card_stack
     |---- Gym.jsx
|----login
     |----login.jsx
     |----signup.jsx
|---- user
     |---- assets
     |---- badminton
          |----screens
               |---- booking
                    |----active-booking.jsx
                    |----booking.jsx
                    |----pre-booking.jsx
               |----equipment
               |----home
               |----leaderboard
               |----tutorials
               |----workshop
          |---- CardStack.jsx
          |---- Badminton.jsx
     |---- basketball
          |---- screens
               |----home
               |----tutorials
               |----workshop
          |---- CardStack.jsx
          |---- Basketball.jsx
     |----components
          |----tutorials
               |----table.jsx
          |----workshops
               |----table.jsx
               |----table_workshop.jsx
     |---- cricket
          |---- screens
               |---- home
               |---- tutorial
```

```
                |---- workshop
          |---- CardStack.jsx
          |---- CardStack.jsx
     |----football
           |---- screens
                |---- home
                |---- tutorial
                |---- workshop
          |---- CardStack.jsx
          |---- Football.jsx
     |---- history
          |---- components
                |---- greeting.jsx
                |---- heading.jsx
                |---- info.jsx
                |---- table.jsx\
          |----History.jsx
     |---- hockey
          |---- screens
                |---- Home
                |---- Tutorial
                |---- Workshop
          |---- CardStack.jsx
          |---- Hockey.jsx
|---- Landing_Page1
     |---- Button.jsx
     |---- CourtsAvailable.jsx
     |---- LP1.jsx
     |---- LandingPageNewsArticle.jsx
     |---- UpcomingBookings.jsx
     |---- table.jsx
|---- Landing_Page2
     |----App.jsx
     |----LDP.jsx
     |----NewsArticle.jsx

     |----self-help
          |----self-help.jsx
          |----self-help_post.jsx
```

```
|----squash
     |----screens
          |---- booking
               |----active-booking.jsx
               |----booking.jsx
               |----pre-booking.jsx
          |----equipment
          |----home
          |----leaderboard
          |----tutorial
          |----workshop
     |---- CardStack.jsx
     |---- Squash.jsx
|---- swimming
     |----screens
          |---- booking
               |----active-booking.jsx
               |----booking.jsx
               |----pre-booking.jsx
          |----equipment
          |----home
          |----leaderboard
          |----tutorial
          |----workshop
     |---- CardStack.jsx
     |---- Swimming.jsx
|---- tabletennis
     |----screens
          |---- booking
               |----active-booking.jsx
               |----booking.jsx
               |----pre-booking.jsx
          |----equipment
          |----home
          |----leaderboard
          |----tutorials
          |----workshop
     |---- CardStack.jsx
     |---- TableTennis.jsx
```

```
     |---- tennis
          |----screens
               |---- booking
                    |----active-booking.jsx
                    |----booking.jsx
                    |----pre-booking.jsx
               |----equipment
               |----home
               |----leaderboard
               |----tutorials
               |----workshop
          |---- CardStack.jsx
          |---- Tennis.jsx
     |---- volleyball
          |---- screens
               |---- Tutorial
               |---- Workshop
               |---- Home
          |---- CardStack.jsx
          |---- Volleyball.jsx
     |---- yoga
          |---- screens
               |---- Tutorial
               |---- Workshop
               |---- Home
          |---- CardStack.jsx
          |---- Yoga.jsx
     |---- Header.jsx
     |---- SidebarData_MentalWellness.jsx
     |---- SidebarData_PhysicalWellness.jsx
|---- App.jsx
|---- main.jsx
|---- protected_routes_admin.jsx
|---- protected_routes_user.jsx
|---- useAuth.jsx
```

Database:

The databases section contains various databases and collections of the form:

```
|---- bookings
     |---- counsellor_appointments
     |---- equipments
     |---- sport_bookings
     |---- swim_gym_memberships
     |---- tutorials
     |---- tutorials
     |---- yoga_sessions_sport_workshops
|---- content
     |---- blogs_posted_by_counsellors
     |---- homepage_announcements
     |---- sport_workshops
     |---- time_slot_posted_by_counsellors
     |---- time_slots_by_counsellors
     |---- tutorials
     |---- yoga_sessions
|---- courts
     |---- badmintons
     |---- basketball
     |---- cricket
     |---- football
     |---- hockey
     |---- squashes
     |---- table_tennis
     |---- tennis
     |---- volleyball
|---- equipments
     |---- badminton
     |---- basketball
     |---- cricket
     |---- football
     |---- hockey
     |---- squash
     |---- swimming
     |---- table_tennis
```

```
     |---- tennis
     |---- volleyball
|---- leaderboard
     |---- badmintons
     |---- squashes
     |---- table_tennis
     |---- tennis
|---- user
     |---- users
     |---- records
```

API Endpoints

The following is a summary of all the possible API calls that are present in the application

1. Authentication

a. Login

```
URL:  /login

Method: POST

Data : {
      username,
      password
}
Status : {
      If successful: {
            200_OK
      }
      else: {
            401_Unauthorized (Wrong Password / Unregistered)
            500_Internal Server Error (Authentication Failed)
      }
}
```

b. Sign-Up

```
URL: /signup

Method: POST

Data: {
      username,
      password,
      confirm_password,
      email_id
}
```

```
Status : {
      If successful: {
            201_Created (User Created)
      }
      else: {
            400_Bad Request (username/email_id already exists)
            500_Internal Server Error (Registration Failed)
      }
}
```

c. Check_user

URL: /checkUser/:username

Method: GET

Response : Returns true if user exists and false if not

```
Status : {
      If successful: {
            200_OK
      }
      else: {
            500_Internal Server Error
      }
}
```

d. Check routerlied Timeslot

URL: /checkrouterliedTimeslots

Method: POST

```
Data:{
      user_id,
      selectedTime
```

```
}
Status: {
      If successful: {
            OK( routerlied successfully)
            }
       Else: {
            500_Internal Server Error(An error occurred)}
       }
}
```

2. User

a. Profile Page

```
URL: /profile

Method: GET

Response: Authenticate User and show profile page

Status : {
      If successful: {
            201_Created (User Created)
      }
      else: {
            400_Bad Request (username/email_id already exists)
            500_Internal Server Error (Registration Failed)
      }
}
```

b. Booking History

```
URL: /get_booking_history

Method: GET

Response: Acquires the booking history of the user from the
database
```

```
Status : {
      If successful: {
            200_OK (Data Fetched)
      }
      else: {
            //No edit
      }
}
```

3. Sports

a. View leaderboard

Places used: Badminton, basketball,cricket, football,gym, hockey, squash, swimming, tennis, volleyball, table_tennis.

Example:

```
URL: /badminton/leaderboard

Method: GET

Response: Display the leaderboard of badminton

Status:{
      If successful:{
            200_ OK
      }
      Else :{
            401_Unauthorised (Not logged in)
      }
}
```

b. View tutorials

Places used: badminton, basketball,cricket, football,gym, hockey, squash, swimming, tennis, volleyball, table_tennis

Example:

```
URL: /badminton/tutorials

Method: GET

Response: Display the tutorials of badminton

Status:{
      If successful:{
            200_ OK
      }
      Else :{
            401_Unauthorised (Not logged in)
      }
}
```

c. View Workshops

Places used: badminton, basketball, cricket, football, hockey, squash, tennis, volleyball, table tennis, yoga

Example:

```
URL: /badminton/workshop

Method: GET

Response: Display the workshops of badminton

Status:{
      If successful:{
            200_ OK
      }
```

```
        Else :{
                401_Unauthorised (Not logged in)
        }
}
```

d. Active Booking

Places used: badminton, basketball, cricket, football, hockey, squash, tennis, volleyball, table tennis.

Example:

```
URL: /badminton/active_booking

Method: POST

Data: {
        user_id,
        time_slot,
        type_of_sport,
        time_of_booking,
       court_id,
       show_up_status,
       partners_id,
       no_partners,
       booking_status
}

Status : {
        If successful: {
                200_OK (Booking Done)
        }
        else: {
                500_Internal Server Error (Booking failed)
        }
}
```

e. Pre Booking

Places used: badminton, basketball, cricket, football, hockey, squash, tennis, volleyball, table tennis.

Example:

```
URL: /badminton/pre_booking

Method: POST

Data: {
      user_id,
      time_slot,
      type_of_sport,
      time_of_booking,
     court_id,
     date_slot,
     show_up_status,
     partners_id,
     no_partners,
     booking_status
}

Status : {
      If successful: {
            200_OK (Booking added to queue)
      }
      else: {
            500_Internal Server Error (Booking failed)
      }
}
```

f. Self-Help Blogs

```
URL: /self_help

Method: GET
```

```
Response: Display the self-help blogs uploaded by the
counsellor

Status:{
      If successful:{
            200_ OK
      }
      Else :{
            401_Unauthorised (Not logged in)
      }
}
```

g. Apply Workshop

Places used: badminton, basketball, cricket, football, hockey, squash, tennis, volleyball, table tennis

Example:

```
URL: /badminton/apply_workshop

Method: POST

Data: {
      workshop_id,
      user_id
}

Status : {
      If successful: {
            200_OK (Workshop updated successfully)
      }
      else: {
            500_Internal Server Error (Error updating workshop)
            404_Not Found (Workshop not found)
            400_Bad Request (Workshop max strength reached)
      }
}
```

4. Admin

a. Mark Attendance

Places Used: badminton, squash, table tennis, tennis

Example:

```
URL: /badminton/attendance

Method: GET

Response: get the occupancy status of court
```

b. Enter Court Name

Places used: badminton, squash, tennis, table tennis

Example:

```
URL: /court_name_entry

Method: POST

Data: {
      court_name,
      type of sport
}

Status : {
      If successful: {
            200_OK (Court exists for the specific sport)
      }
      else: {
            404_Not Found (Court does not exist for the sport)
            400_Bad Request (Invalid type of sport)
      }
}
```

c. Mark Attendance

```
URL: /mark_attendance

Method: POST

Data: {
      user_1,
      user_2,
      user_3,
      user_4,
      attendance_1,
      attendance_2,
      attendance_3,
      attendance_4,
      position_1,
      position_2,
      position_3,
      position_4,
}

Status : {
      If successful: {
            200_OK (Attendance marked)
      }
}
```

d. Fill Entry

```
URL: /fill_entries

Method: POST

Data: {
      court_name,
```

```
            type of sport,
            court_id,
            time_slot,
            date_slot,
            user_id_1,
            user_id_2,
            user_id_3,
            user_id_4
      }

      Status : {
            If successful: {
                  200_OK (Usernames posted)
            }
            else: {
                  400_Bad Request (Post empty usernames fields)
            }
      }
```

e. Marking metrics

```
URL: /match_metric_marking

Method: POST

Data: {
            username_1,
            username_2,
            username_3,
            username_4,
            attendance_1,
            attendance_2,
            attendance_3,
            attendance_4,
            position_1,
            position_2,
            position_3,
            position_4,
            type_of_sport,
            attributeList
```

```
}
Status:{
      If successful: {
            OK (Document updated)
      }
      Else:{
            Error(Error updating document)
            Error(Document not found)
      }
}
```

5. Coach

a. Post Workshop

```
URL : /coach/postWorkshop

Method: POST

Response: Post workshop for coach

Data:{
      start_time,
      end_time,
      Description,
      racquet,
      cork,
      shoe
      coach_user_id,
      max_participants,
      date,
      type_of_sports
}

Status:{
      If successful:{
```

```
            200_ OK(Post successful)
      }
      Else :{
            500_Exception(Post failed)
      }
}
```

b. Reserve Court

```
URL: /coach/reserveCourt

Method: POST

Response: Court booking for coach

Data:{
          time_slot,
          date_slot,
          court_id,
          show_up_status,
          type_of_sport,
          time_of_booking,
          booking_status,
          user_id
}

Status:{
      If successful:{
            200_ OK(Reserve successful)
      }
      Else :{
            500_Exception(Reserve failed)
      }
}
```

c. Post Yoga Session

```
URL: /yoga/postSession

Method: POST

Response: Post new yoga session

Data:{
          batch_size,
          content,
          startDate,
          startTime,
          endTime
}

Status:{
     If successful:{
          200_ OK(Post successful)
     }
     Else :{
          500_Exception(Post failed)
     }
}
```

6. Counsellor

a. Post Blog

```
URL: /counsellor/postBlog

Method: POST

Response: Post new blog

Data:{
          content,
          title,
          counsellor_username
}
```

```
Status:{
      If successful:{
            200_ OK(Post successful)
      }
      Else :{
            500_Exception(Post failed)
      }
}
```

b. Availability

```
URL: /counsellor/availability

Method: POST

Response: Update Availability status

Data:{
          day_vector,
          hour_vector,
          date_slot,
          date_slot_time_vector,
          counsellor_user_id
}

Status:{
      If successful:{
            200_ OK(Availability updated successful)
      }
      Else :{
            500_Exception(Availability updating failed)
      }
}
```

c. Get Appointments

```
URL: /counsellor/getAppointments
```

```
Method: POST

Response: Get all pending appointments

Data:{
          booking_id,
          user_id,
          date_slot,
          time_slot,
          booking_status,
          counsellor_user_id
}

Status:{
     If successful:{
          200_ OK (Got pending appointments)
     }
     Else :{
          500_Exception (Unable to fetch appointments)
     }
}
```

d. Get Availability

```
URL: /counsellor/getAvailability

Method: POST

Response: Get available times

Data:{
          hour_vector,
          day_vector,
          date_slot,
          date_slot_time_vector,
          counsellor_user_id
}
```

```
Status:{
      If successful:{
            200_ OK (Got availability)
      }
      Else :{
            500_Exception (Error Availability Fetching)
      }
}
```

e. Delete Day Availability

```
URL: /counsellor/deleteDayAvailabilty

Method: POST

Response: Delete the availability of a day.

Data:{
          day_vector,
          counsellor_user_id
}

Status:{
      If successful:{
            200_ OK (Successfully updated)
      }
      Else :{
            500_Exception (Availability detection failed)
            500_Exception (Availability updating failed)
      }
}
```

f. Delete Date Availability

```
URL: /counsellor/deleteDateAvailabilty

Method: POST

Response: Delete the availability of a day.

Data:{
          date_slot,
          counsellor_user_id
}

Status:{
      If successful:{
            200_ OK (Successfully updated)
      }
      Else :{
            500_Exception (Availability detection failed)
            500_Exception (Availability updating failed)
      }
}
```

g. Accept Appointments

```
URL: /counsellor/acceptAppointments

Method: POST

Response: Accept appointment by changing the status

Data:{
          booking_status,
          appointment_id,
}

Status:{
      If successful:{
            200_ OK(appointment acceptance successful)
      }
```

```
            Else :{
                  404_Exception(Appointment not found)
                  500_Exception(Appointment acceptance failed)
            }
      }
```

Completeness

Implemented Features

1. **Sign-up/Sign-in:** *Created a login/register page which redirects to their respective profile dashboard. It also displays if incorrect credentials are entered.*

2. **Privacy of users:** *We have used **jwt tokens and sha256 hashing** to store the passwords of all the users in the database. By using this technique, the passwords of the users are not visible to those who have access to the database and this ensures the security and privacy of the users.*

3. **Features implemented in User Section:**

    a. **User Dashboard:** *The dashboard will be divided into three sections to assist the users, i.e. IIT Kanpur students. The facilities offered to them are:*
    - *Book a sports facility*
    - *Book a yoga session*
    - *Book an appointment with a counsellor.*
    - *The profile will show the upcoming bookings of the user.*

    The side navigation bar will be divided into two sections, one for physical wellness and the other for mental wellness.
    - *The physical wellness section will enlist the various sports whose facilities are available in IITK. It will also contain yoga*
    - *The mental wellness section will contain links to avail the facilities of the ICS counsellors as well as self-help blogs.*

    b. **Sports Coach Dashboard:** *Every coach of every sport in IIT Kanpur will have a dashboard. This page will allow them to:*
    - *Posts regarding upcoming workshops*
    - *Check enrollment of students.*
    - *Reserve courts for team practice*

    c. Validation dashboard:
    - *Verifies the occupancy of courts for existing bookings*
    - *Checks if players have arrived on time as well*

    d. **Counsellor Dashboard:**
    - *Accept appointments*
    - *Post blogs*
    - *Set their availability*

e. **Yoga Instructors Dashboard:**
- *Put up announcements*
- *Post blogs*

f. Swimming/Gym Instructor Dashboard:
- *Reserve slots*
- *Check enrollment of users*

4. **Features implemented in Sports Section:**

a. **Sports Page:**
- *View tutorials*
- *View leaderboard (if exists for that sport)*
- *Enroll for workshops*
- *Book courts in advance as well as active*
  - *Advance booking implements a FCFS system*
  - *Active booking is to book in case no advance booking is done*
- *Add playmates (if exists for that sport)*
- *Pair players (if exists for that sport)*

b. **Gym/Swimming Page:** *This page will allow users to the following for the gym/swimming:*
- *slots can be booked depending on availability*
- *user can see the respective tutorials*
- *there is an option to show the pass for verification while entry*

5. **Features implemented in Wellness Section:**

a. **Yoga Page:**
- *Enroll for upcoming yoga sessions*
- *View tutorials*

b. **Counsellor Session Booking Page**:
- *Display available timings*
- *Book appointments*
- *Keep a track of upcoming appointments*

c. **Self-help blogs:**
- *Display self help blogs*

Future Development Plans

1. **Adding personalised profile pictures:** *We shall allow users to upload a profile picture at the time of sign-up which will be displayed at all the user pages.*

2. **Viewing past bookings or appointmnets:** *On registering for a workshop or appointment we are yet to implement the feature to allow users to view their respective bookings with the status*

3. **Adding forgot password feature:** *At present there is no feature for users to reset their password if they forget it. We shall be including an option at login to enable users to reset their password*

4. **Issuing equipment:** *We shall be adding the option to issue equipment at various sports facilities. This feature wasn't implemented properly and shall be debugged in the testing phase.*

5. **View empty courts:** *We will be adding a feature to show all the courts that are currently empty so that the active booking feature can be used more conveniently.*

6. **Statistics:** *We will be adding a feature for the coaches to view the statistics of court bookings so that they can book courts for team practice at the appropriate times, and staff can be adjusted based on the people that are coming at a particular time.*

7. *General fixing of bugs and improvement of software through rigorous testing.*

Reserved User_ID's for testing

The following is a list of credentials so that the special dashboards for the coaches can be tested

User name	Password	Role
coach_badminton	coach_badminton	Coach
coach_tabletennis	coach_tabletennis	Coach
coach_tennis	coach_tennis	Coach
chinmay	linguist	Counselor
coach_squash	coach_squash	Coach
yoga_inst	yoga_inst	Yoga Instructor
gym_inst	gym_inst	Gym Instructor
swim_inst	swim_inst	Swim Instructor
admin	admin	Admin

Appendix A - Group Log

Date	Timings	Duration	Minutes
02 Mar	23:00 – 1:30	2hrs 30 min	● Discussed the components to be present and distributed the division of the components
07 Mar	22:00 – 00:00	2hrs	● Built the individual components and discussed further plans regarding the combination of the components.
09 Mar	21:00 – 22:00	1hr	● Revision of the components and distribution of the individual pages of the application.
13 Mar	21:00 – 03:00	6hrs	● Finished the user dashboards. ● Finalising the coach and counsellor dashboards and started working on the databases. ● Started working on the backend.
14 Mar	21:00 – 03:00	6hrs	● Finished the frontend. ● Working on the backend and the integration. ● Started working on the implementation document.
15 Mar	10:00 – 14:00	4hrs	● Finished the implementation doc ● Finishing up the backend and the integration of the frontend and backend.