Better Me

An App to Improve your Health and Life

By
Aravindharaj Rajendran
Graduate Student
Dept. of Computer Science
UNC Charlotte

Project Summary and Conclusion

We succeeded in implementing a working system using MySQL and a PHP frontend. We decided to host on the Amazon Web Services platform, a popular 'Cloud' provider. Utilizing the cloud provided us with a convenient common platform to develop on and ensured all of us equal access to resources.
Some lessons learned:

- We could have benefited from a greater volume of test data. We injected a large batch of generated data into the system and relied on that for our user and performance testing, but more time spent using the system would have given us more realistic data to use.
- More data would have given us an opportunity to delve deeper into performance tuning. As it was we created a number of indexes to optimize our platform, but certainly with a real in-use system, we would have other optimizations we'd want to do.
- We used a free set of products from Amazon Web Services but still accumulated some charges. More effort would be spent next time understanding the different tiers of service and how the products are managed and billed.

Future Enhancements:

- If we were to pursue this application as a real product, we would certainly spend more time on the data since one of our main objectives was to make a tool that has a simple interface, we would partner with some food scientists to try and create a manageable dataset of options for the user to choose from.
- Our product would have a very mobile optimized experience, and this would be a major focus of any next work.

As a team we were very pleased with the overall project and had several 'A-ha' moments when working through the EERD diagram that helped us to see for ourselves the benefits of good data modeling up front.

Project Definition

Our product, *Better Me*, is designed with the active user in mind. We are creating an application to track activity related to the user's health and wellbeing and provide tools for the user to analyze patterns and establish improved health habits. There is a rapidly growing market for biometric related tools; we will differentiate ourselves by providing a simple interface with quick and easy choices for the user to make.

Our system is comprised of the user interface, the database backend, and an analytics platform. We will deploy our system in the public Cloud on Amazon Web Services. We will use the widely available LAMP system which is composed of Apache, PHP and MySQL running on Linux. Using the Cloud will provide us with an easily accessible and scalable platform.

The application data is primarily timeseries entries that track the users' activity. Over time, we will provide tools to look for patterns that can help the user to determine steps to improve their health. For example, a user may be able to determine that they feel poorly whenever they eat

dairy, and therefore are dairy intolerant. Another user sees the correlation between their mood and how much sleep they've gotten, and take steps to improve their sleep habits.

Potential stretch goals are creating a mobile interface, creating tie-ins with external health improvement websites, finding a way to drive adspace in the app, and further development of the visualization tools.

Entities

Attributes, primary keys(PK), foreign keys(FK)

Events: We have time stamp for each event

- **Food Event :**
  - **Food_Event_ID** - Varchar(10) **PK**
  - **Category** - Character **FK** [Input: Meals, Desserts, Drinks etc.]
  - **User_ID** - Varchar(20) **FK**
  - Time_stamp - Timestamp
  - Quality - Number [Input: Rate between 1-5]
  - Source - Character [Input: Home/Restaurant]
  - Size - Number [Input:S, M, L]

- **Sleep Event :**
  - **Sleep_Event_ID** - Varchar(20) **PK**
  - **User_ID** - Varchar(20) **FK**
  - Start_Time - Timestamp
  - Stop_Time - Timestamp
  - Duration - Number [Derived column: No. of Hours]
  - Quality - Number [Input: Rate between 1-5]
  - Location - Varchar(50)

- **Exercise Event** :
  - **Exercise_Event_ID** - Varchar(20) **PK**
  - **User_ID** - Varchar(20) **FK**
  - **Type** - Varchar(20) **FK** [Input:Cardio,Running,Biking etc.]
  - Duration - Number [Input: Number of hours]
  - Intensity - Varchar(9) [Input:Easy, Medium,Difficult]
  - Start_Time - Timestamp
  - Stop_Time - Timestamp

- **Mental Mood** -
  - **Mental_Mood_ID** - Varchar(20) **PK**
  - **User_ID** - Varchar(20) **FK**
  - Mood_Condition - Number [Input: Rate between 1-5. Bad(1) to Good(5)]
  - Time_stamp - Timestamp

- **Physical Mood** -
  - **Physical_Mood_ID -** Varchar(20) **PK**
  - **User_ID** - Varchar(20) **FK**
  - Mood_Condition - Number [Input: Rate between 1-5. Bad(1) to Good(5)]
  - Time_stamp - Timestamp

- **User** :
  - **User_ID** - Varchar(20) **PK**
  - User_Type - Varchar(20) **(Type Discriminator)**
  - Name - Varchar(50)
  - Email - Varchar(25)
  - Birthdate - Date
  - Age - Number [In years]
  - Sex - Varchar(2) [Input: M/F]
  - Height - Number [In cms]
  - Weight - Number [In cms]
  - Role - Varchar(20)

- **Food** :
  - **Category** - Varchar(20) **PK** [Input: Desserts, Vegetables, Meat, Grams, Dairy, Coffee, Alcohol]
  - Protein - Number
  - Fat - Number
  - Calories - Number

- **Exercise** :
  - **Type** - Varchar(20) **PK** [Input: Running, Biking, Cardio, Weights]
  - Avg_Calories - Number

- **History** (Aggregate of day) :
  - **User_ID** - Varchar(20) **FK**
  - **Time_stamp** - Timestamp
  - Avg_Calories_Gained - Number
  - Avg_Calories_Burnt - Number
  - Overall_Mood_Of_Day - Number [Input: Rate between 1-5. Bad(1) to Good(5)]
  - Overall_Health_Condition - Number [Input: Rate between 1-5. Bad(1) to Good(5)]
    - User_ID, Time_stamp - **PK**

Business Rules / Constraints

- Users will login or be able to register. Registered users may retrieve forgotten passwords.
- A user may have 0 or more events.
- User may have 0 or more mood events.
- User may have 0 or more sleep events.
- The history table contains pre-aggregated statistics rolled up to daily values. This is intended to improve performance for reporting and analytics.
- When a user creates a food event, this event may have one or more foods in it.
- When a user creates an exercise event, this event may have one or more exercises in it.

User Interface Requirements -

1. Login module:

- Whenever the user opens up the application, the system should always display the login page initially.
- If the user is a first-time user and doesn't have an account in the application, he/she should be allowed to create a new account on the login page.
- If not, he/she should be allowed to login to the application once the login credentials are verified.
- If the user does not remember the credentials, he/she should be able to request for a new one (either username or password or both) on the login page.

2. User module:

- When the users log in into the application, they should be presented with the following tabs to navigate to different pages:

  a. **My Profile** – Navigation to profile page where the users can View and Update their user information (such as First Name, Last Name, Age, Date of Birth, Password)

  b. **My Health History** – Navigation to health report page where the users can view health report on daily/weekly/monthly basis generated using the details previously entered by them for each event. The report should be given in the form of graphical representation and should be provided with respective analysis and suggestions for better health.

c. **My Food** – Navigation to Food Event page where the users enter their Food intake details such as the Type of the Food (Meals, Desserts, Drinks etc), Quantity (Small, Regular, Large), Source of Food (Homemade or from Restaurant) and Time of intake.

d. **My Sleep** - Navigation to Sleep Event page where the users enter their Sleep details such as the Time they went to sleep, the Time they woke up, Duration of the sleep, Quality of the sleep (ranging from 1 to 5 and represented by the Smiley faces) and Location (in Home or in Hotel).

e. **My Workout** - Navigation to Exercise Event page where the users enter their exercise details such as the Type of workout, Intensity, Duration and Time of workout.

f. **My Mood** - Navigation to Mental mood event page where the users enter their mood details on the scale of 1 to 5 (One being very bad and five being very good).

g. **My Physical Feeling** - Navigation to Physical mood event page where the users enter their physical mood details on the scale of 1 to 5(One being very bad and five being very good).

3. Admin Module:

The Admin module allows all the Administrators to edit the content present in the event tabs (The Administrators account are created at the backend by the Database Administrator). After logging in to the application, the admin will view a page with all the existing events. The page has navigation buttons to view and edit each existing events. The page also has provision to add a new event. On the edit page of the existing event specification, the page admins are allowed to add a new attribute/field corresponding to that event or change existing attribute or field.

Users and User Privileges

1.USER

A typical end user is the one who uses the application to check on their health by entering different daily event details.

2.ADMINISTRATOR

Administrator is the one who has all privileges at the portal like maintains event specifications in the application for the users.

3.SYSTEM

System analyzes the saved records in database on user's event and generates report on users health**.**

User	Roles	Description
User		
	Registration	Allow non-admin users(end users) to register in the application to use the features of application regularly.
	Login	Allow user to login/logout into application.
	Choose Event	Allow user to choose the desired event to enter the details to track their health. Events include food event sleep event exercise and so on.
	Enter and Save Event	Allow user to add event details on the particular selected events and submit it for generating health report on analyzing the patterns.
	View Report	Allow user to view their daily/weekly/monthly health report generated on the basis of their events saved.
Administrator		
	Manage Events	Manages the event specifications that is provided to user to save their day to day events
System		
	Maintain user credentials	Resets password for users on request.
	Generate Report	Analyses user’s events and generate health report to the users.

Project plan with tentative dates for completion

- Sept 15 - Sept 18: Clean up the project report
- Sept 21 - Sept 27: UI Design and Database Design
- Sept 28 - Oct 4: Implement User Table and Exercise Table. User Login, Signup Page
- Oct 5 - Oct 11: Implement Food Table and Food Event Tables
- Oct 12 - Oct 18: Implement Sleep Event Table, Mental Mood Table and Physical Mood Table
- Oct 19 - Oct 25: Implement Exercise Event Table and History Table
- Oct 26 - Nov 1: Finish Logic Part of the APP
- Nov 2 - Nov 9: Wrap up, Testing and Debugging
    - Mid November - Project Submission

Food Event

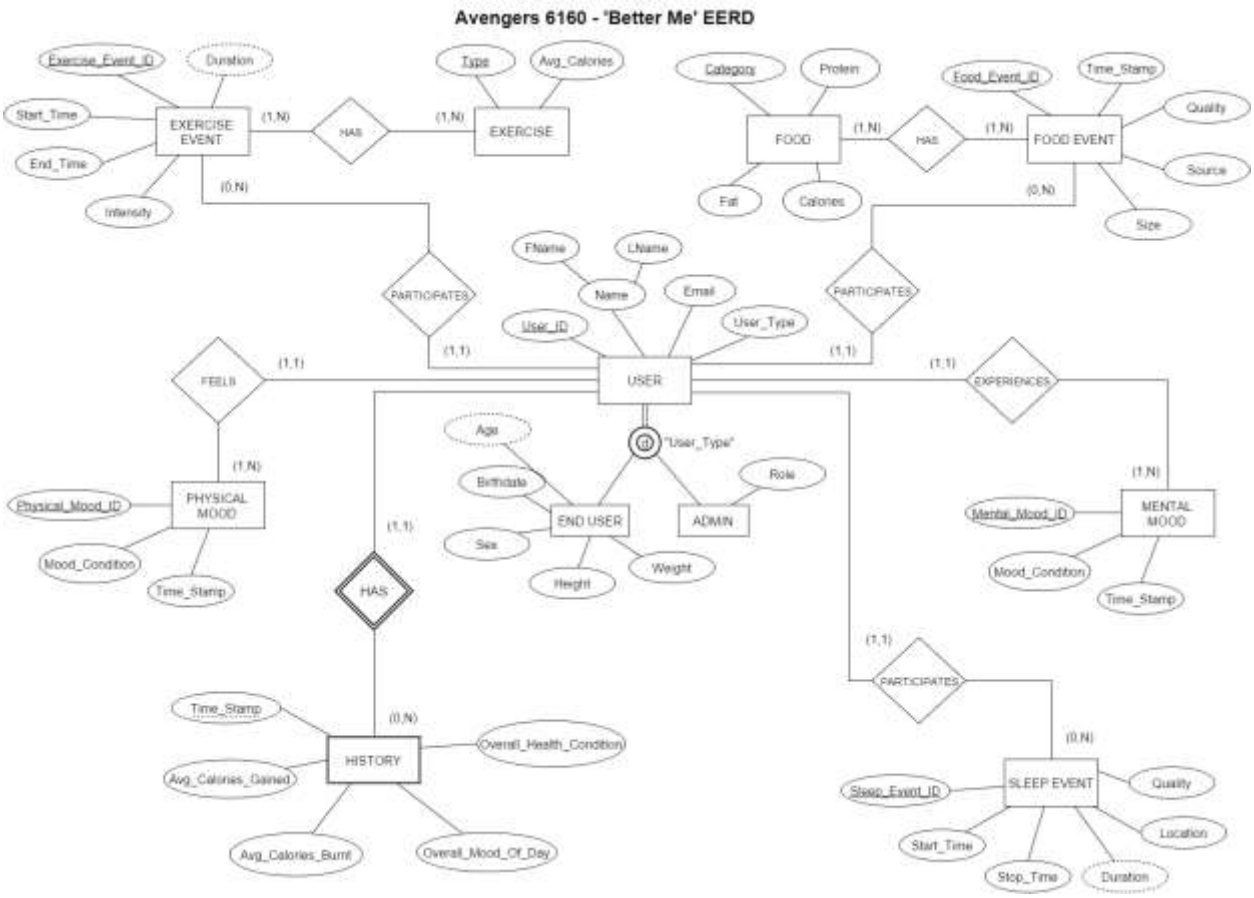
Main Course	Mashed Potato Steak Chili Meatball Sandwich Burger Pizza Rice Noodle
Beverage	Soda Alcohol Juice Coffee Ice Tea
Dessert	Cup cake Brownies Cookie Pudding Ice cream Chocolate Pretzels Popcorn

Fruit	Apple Banana Orange Watermelon Pineapple Grape Peach Avocado

Exercise Event

- Jogging
- Swimming
- Walking
- Cycling
- Yoga
- Hiking

EERD



Database Create Scripts

USER:

```
CREATE TABLE `user` (
  `user_id` int(11) NOT NULL AUTO_INCREMENT,
  `user_type` varchar(20) DEFAULT NULL,
  `user_name` varchar(50) DEFAULT NULL,
  `user_email` varchar(50) DEFAULT NULL,
  `user_birthdate` date DEFAULT NULL,
  `user_age` int(11) DEFAULT NULL,
  `user_sex` varchar(1) DEFAULT NULL,
  `user_height` int(11) DEFAULT NULL,
  `user_weight` int(11) DEFAULT NULL,
  `user_role` varchar(20) DEFAULT NULL,
  `password` varchar(50) DEFAULT NULL,
  PRIMARY KEY (`user_id`),
  KEY `user` (`user_email`,`password`)
);
```

EXERCISE;

```
CREATE TABLE `exercise` (
  `exercise_type` varchar(30) NOT NULL DEFAULT '',
  `avg_calories` int(11) DEFAULT NULL,
  PRIMARY KEY (`exercise_type`)
);
```

FOOD:

```
CREATE TABLE `food` (
  `category` varchar(30) NOT NULL DEFAULT '',
  `protein` int(11) DEFAULT NULL,
  `fat` int(11) DEFAULT NULL,
  `calories` int(11) DEFAULT NULL,
  PRIMARY KEY (`category`)
);
```

EXERCISE_EVENT:

```
CREATE TABLE `exercise_event` (
  `exercise_event_id` varchar(20) NOT NULL DEFAULT '',
  `user_id` int(11) DEFAULT NULL,
  `exercise_type` varchar(20) DEFAULT NULL,
  `duration` int(11) DEFAULT NULL,
  `intensity` varchar(10) DEFAULT NULL,
  `start_time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
  `stop_time` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
  PRIMARY KEY (`exercise_event_id`),
  KEY `exercise_event_fk1` (`user_id`),
  KEY `exercise_event_fk2` (`exercise_type`),
  KEY `exercise` (`exercise_event_id`,`user_id`,`exercise_type`,`duration`,`intensity`),
```

```
  CONSTRAINT `exercise_event_fk1` FOREIGN KEY (`user_id`) REFERENCES `user`
(`user_id`),
  CONSTRAINT `exercise_event_fk2` FOREIGN KEY (`exercise_type`) REFERENCES
`exercise` (`exercise_type`)
);
```

FOOD_EVENT:

```
CREATE TABLE `food_event` (
  `food_event_id` varchar(10) NOT NULL DEFAULT '',
  `category` varchar(30) DEFAULT NULL,
  `user_id` int(11) DEFAULT NULL,
  `time_stamp` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
  `quality` int(11) DEFAULT NULL,
  `food_source` varchar(20) DEFAULT NULL,
  `size` varchar(1) DEFAULT NULL,
  PRIMARY KEY (`food_event_id`),
  KEY `food_event_fk1` (`category`),
  KEY `food_event_fk2` (`user_id`),
  CONSTRAINT `food_event_fk1` FOREIGN KEY (`category`) REFERENCES `food`
(`category`),
  CONSTRAINT `food_event_fk2` FOREIGN KEY (`user_id`) REFERENCES `user` (`user_id`)
);
```

HISTORY:

```
CREATE TABLE `history` (
  `user_id` int(11) NOT NULL,
  `time_stamp` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
  `avg_calories_gained` int(11) DEFAULT NULL,
  `avg_calories_burnt` int(11) DEFAULT NULL,
  `overall_mood_of_day` int(11) DEFAULT NULL,
  `overall_health_condition` int(11) DEFAULT NULL,
  PRIMARY KEY (`user_id`,`time_stamp`),
  CONSTRAINT `history_fk` FOREIGN KEY (`user_id`) REFERENCES `user` (`user_id`)
);
```

MENTAL_MOOD:

```
CREATE TABLE `mental_mood` (
  `mental_mood_id` varchar(20) NOT NULL DEFAULT '',
  `user_id` int(11) DEFAULT NULL,
  `mood_condition` int(11) DEFAULT NULL,
  `time_stamp` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
  PRIMARY KEY (`mental_mood_id`),
  KEY `mental_mood_fk` (`user_id`),
  CONSTRAINT `mental_mood_fk` FOREIGN KEY (`user_id`) REFERENCES `user` (`user_id`)
);
```

PHYSICAL_MOOD:

```
CREATE TABLE `physical_mood` (
```

```
  `physical_mood_id` varchar(20) NOT NULL DEFAULT '',
  `user_id` int(11) DEFAULT NULL,
  `mood_condition` int(11) DEFAULT NULL,
  `time_stamp` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
  PRIMARY KEY (`physical_mood_id`),
  KEY `physical_mood_fk` (`user_id`),
  CONSTRAINT `physical_mood_fk` FOREIGN KEY (`user_id`) REFERENCES `user`
(`user_id`));
SLEEP_EVENT:
CREATE TABLE `sleep_event` (
  `sleep_event_id` varchar(20) NOT NULL DEFAULT '',
  `user_id` int(11) DEFAULT NULL,
  `start_time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
  `stop_time` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
  `duration` varchar(10) DEFAULT NULL,
  `quality` int(11) DEFAULT NULL,
  `location` varchar(50) DEFAULT NULL,
  PRIMARY KEY (`sleep_event_id`),
  KEY `sleep_event_fk` (`user_id`),
  KEY `sleep` (`sleep_event_id`,`user_id`,`duration`,`quality`,`location`),
  CONSTRAINT `sleep_event_fk` FOREIGN KEY (`user_id`) REFERENCES `user` (`user_id`)
);
```

Database Select Statements
Example 1:
Looking for a history of everytime user fred@gmail.com ate meat in October:

```
SELECT user_name, food, timestamp
FROM user
JOIN food_event on user.user_id = food_event.user_id
JOIN food on food.category = food_event.category

WHERE
user_email = 'fred@gmail.com'
and category = 'meat'
and time_stamp BETWEEN '2015-10-01' AND '2015-11-01'
```

Database Insert Statements

```
INSERT INTO `BetterMeMain`.`exercise_event`
(
`user_id`,
`exercise_type`,
`duration`,
`intensity`,
`start_time`,
`stop_time`)
```

```
VALUES
(
5,
'Bowling',
60,
4,
'2015-11-6 12:05:00',
'2015-11-6 12:10:00');

INSERT INTO BetterMeMain.exercise(exercise_type,avg_calories)
values
("Aerobics, high impact",664)

INSERT INTO BetterMeMain.food(category, protein, fat, calories)
VALUES ('Grain',9,3,230)
```

Database Triggers

```
INSERT_AGE:
#For inserting age based on DOB in the User table
CREATE TRIGGER insert_age
BEFORE INSERT
ON BetterMeMain.user FOR EACH ROW
BEGIN
SET NEW.user_age = (timestampdiff(YEAR, NEW.user_birthdate, curdate()));
END
UPDATE_AGE;
#For updating age based on DOB in the User table
CREATE TRIGGER update_age
BEFORE UPDATE
ON BetterMeMain.user FOR EACH ROW
BEGIN
IF NEW.user_birthdate <> OLD.user_birthdate
THEN
SET NEW.user_age = (timestampdiff(YEAR, NEW.user_birthdate, curdate()));
END IF;
END

INSERT_SLEEP_DURATION:
#For inserting duration based on the start time and stop time in the Sleep_Event table
CREATE TRIGGER insert_sleep_duration
BEFORE INSERT
ON BetterMeMain.sleep_event
FOR EACH ROW
```

```
BEGIN
IF (timestampdiff(MINUTE, NEW.start_time, NEW.stop_time) >= 60) THEN
SET NEW.duration = concat(timestampdiff(HOUR, NEW.start_time,
NEW.stop_time),".",timestampdiff(MINUTE, NEW.start_time, NEW.stop_time)%60);
ELSE
SET NEW.duration = concat(timestampdiff(HOUR, NEW.start_time,
NEW.stop_time),".",timestampdiff(MINUTE, NEW.start_time, NEW.stop_time));
END IF;
END
```

UPDATE_SLEEP_DURATION:

```
#For updating duration based on the start time and stop time in the Sleep_Event table
CREATE TRIGGER update_sleep_duration
BEFORE UPDATE
ON BetterMeMain.sleep_event
FOR EACH ROW
BEGIN
IF (OLD.start_time <> NEW.start_time || OLD.stop_time <> NEW.stop_time)
THEN
IF (timestampdiff(MINUTE, NEW.start_time, NEW.stop_time) >= 60) THEN
SET NEW.duration = concat(timestampdiff(HOUR, NEW.start_time,
NEW.stop_time),".",timestampdiff(MINUTE, NEW.start_time, NEW.stop_time)%60);
ELSE
SET NEW.duration = concat(timestampdiff(HOUR, NEW.start_time,
NEW.stop_time),".",timestampdiff(MINUTE, NEW.start_time, NEW.stop_time));
END IF;
END IF;
END
```

INSERT_EXERCISE_DURATION:

```
#For inserting duration based on the start time and stop time in the Exercise_Event table
CREATE TRIGGER insert_exercise_duration
BEFORE INSERT
ON BetterMeMain.exercise_event
FOR EACH ROW
BEGIN
IF (timestampdiff(MINUTE, NEW.start_time, NEW.stop_time) >= 60) THEN
SET NEW.duration = concat(timestampdiff(HOUR, NEW.start_time,
NEW.stop_time),".",timestampdiff(MINUTE, NEW.start_time, NEW.stop_time)%60);
ELSE
SET NEW.duration = concat(timestampdiff(HOUR, NEW.start_time,
NEW.stop_time),".",timestampdiff(MINUTE, NEW.start_time, NEW.stop_time));
END IF;
END
```

UPDATE_EXERCISE_DURATION:

```
#For updating duration based on the start time and stop time in the Exercise_Event table
CREATE TRIGGER update_exercise_duration
BEFORE UPDATE
ON BetterMeMain.exercise_event
FOR EACH ROW
BEGIN
IF (OLD.start_time <> NEW.start_time || OLD.stop_time <> NEW.stop_time)
THEN
IF (timestampdiff(MINUTE, NEW.start_time, NEW.stop_time) >= 60) THEN
SET NEW.duration = concat(timestampdiff(HOUR, NEW.start_time,
NEW.stop_time),".",timestampdiff(MINUTE, NEW.start_time, NEW.stop_time)%60);
ELSE
SET NEW.duration = concat(timestampdiff(HOUR, NEW.start_time,
NEW.stop_time),".",timestampdiff(MINUTE, NEW.start_time, NEW.stop_time));
END IF;
END IF;
END
```

Database Indices

```
CREATE INDEX user ON user(user_email,password);

CREATE INDEX sleep_event ON sleep_event(sleep_event_id, user_id, duration, quality,
location);

CREATE INDEX exercise_event ON exercise_event(exercise_event_id, user_id, exercise_type,
duration, intensity);

CREATE INDEX food_event ON food_event(food_event_id, user_id, time_stamp, quality);
```