*To check for missed headings, delete at end*

Table of Contents

[I. Introduction 2](#_Toc477822876)

[II. Sleepify’s Promise 2](#_Toc477822877)

[III. Background 2](#_Toc477822878)

[A. Sleep 2](#_Toc477822879)

[B. Sleep and thermoregulation 3](#_Toc477822880)

[IV. Related Work 3](#_Toc477822881)

[A. Literature 3](#_Toc477822882)

[V. System Design and Implementation 4](#_Toc477822883)

[A. Overall High Level Design 4](#_Toc477822884)

[B. Sensors 4](#_Toc477822885)

[C. Backend (Server, Database, API) 4](#_Toc477822886)

[1) The Server 4](#_Toc477822887)

[2) The Database 5](#_Toc477822888)

[3) The API 5](#_Toc477822889)

[D. Machine Learning 6](#_Toc477822890)

[1) Sleep quality evalution 6](#_Toc477822891)

[2) Clustering Analysis and Features Extraction 6](#_Toc477822892)

[3) Model Selection 7](#_Toc477822893)

[4) Machine Learning Model – Server Deployment 8](#_Toc477822894)

[5) Testing 8](#_Toc477822895)

[E. Frontend (iOS Application and Website) 8](#_Toc477822896)

[1) The iOS application and Homekit 8](#_Toc477822897)

[2) iOS User Interface Design 9](#_Toc477822898)

[3) Implementation of Communication 9](#_Toc477822899)

[4) Implementation of Security 9](#_Toc477822900)

[5) Implementation of the Feedback System 10](#_Toc477822901)

[6) Application Testing 10](#_Toc477822902)

[F. Web Interface 10](#_Toc477822903)

[VI. Evaluation Criteria and Setup 10](#_Toc477822904)

[VII. Results 10](#_Toc477822905)

[VIII. Discussion 11](#_Toc477822906)

[IX. Conclusion 11](#_Toc477822907)

[X. References 11](#_Toc477822908)

*A****bstract— <placeholder>***

**Sleepify: A system towards personalizing and optimising sleeping environments**

**Jeremy Chan, Tsz Ho Ho, Dominic Kwok, Ho Shun Lo, Nathalie Wong**

Department of Electrical and Electronic Engineering, Imperial College London, SW7 2AZ  
Email: {jc4913, thh13, cyk113, hsl113, nw813}@ic.ac.uk

***“***. In your final report, make sure that the importance of temperature for sleep is reflected also in the abstract.

# Introduction

It is estimated that people spend about one third of their life asleep. Good sleep is important for both the physical and psychological health of a person. For example, sleep aids in the healing and repair of the blood vessels and heart. Studies have shown that sleep deficiency has been linked to an increased risk of stroke, high blood pressure, heart disease and diabetes [2]. Sleep deficiency has also been linked to mental health problems such as depression, bipolar disorder and anxiety disorder. The mechanism of regulating sleep is complex; there are many factors, which affect sleep quality, such as the psychology of a person. In addition, the thermal environment is a key determinant to achieving good quality sleep [3]. Furthermore, disturbed sleep affects not only physical and psychological health status, but also mortality rates in the elderly [4]. Previous studies conducted on human subjects have shown that sleep is strongly linked to thermoregulation - a process that maintains the body’s core internal temperature at a constant level [3]. This mechanism is also controlled by sleep regulation and circadian rhythm. These findings indicate that maintaining a comfortable thermal sleep environment is important for a healthy life. Several other works have also investigated on the effects of room temperature on sleeping pattern in human [3][5].

These findings are our motivation for creating a product that improves sleep quality by monitoring vital physiological data and sleep environment of the user, and applying machine learning on these data to return an optimal and personalized room temperature.

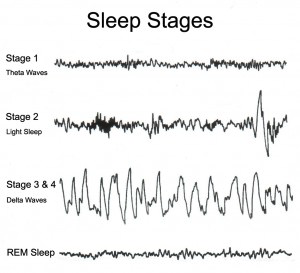
# Sleepify’s Promise

This project aims to provide a better sleeping experience overall from having the room temperature automatically adjust to body and room temperature information from two sensors. Sleepify also promises improved performance and improved machine learning classification accuracy based on prolonged usage of the app. Continued usage of Sleepify is especially important for our machine learning algorithms; thankfully the retention rate of health and fitness apps are the highest among others [1]. Lastly, Sleepify promises to deliver a slick and intuitive app, and web interface for the user to use and interact with, motivating the user to continue using Sleepify regularly; this is crucial to having a low app abandonment rate [2].

# Background

## Sleep

In order to understand how the physical environment affects sleep quality, we must first define what sleep is. Sleep is typically differentiated into five phases; stages 1, 2, 3, 4 and rapid eye movement (REM) [6]. These stages cycle repeatedly during sleep, starting from stage 1 and ending with REM. Electroencephalogram (EEG) measurements are often used to determine these stages as different stages of sleep presents peaks at different region of the signals.



Stage 1 is commonly known as light sleep. During this stage the eyes move very slowly and muscle activity slows. Sometime, we may even experience hypnic myoclonia –a sudden and involuntary muscle contraction. Stage 2 is marked as the onset of sleep where the person becomes disengaged from their surroundings, eye movements stops, heart rate and breathing rate returns to normal, and core body temperature drops.

Stage 3 and 4 is the slow wave sleep (SWS) also called deep sleep stage. In this stage, there are no eye movements or muscle activity. This is the stage where the body repairs and heals itself, muscles are relaxed, and blood supply to muscles increases, blood pressure drops and breathing becomes slower.

REM stage is the final stage of a sleep cycle and it is significantly different from previous stages in that the brain is active. REM EEG waves are very similar to stage 1.

## Sleep and thermoregulation

The sleep-wake rhythm is strongly correlated with the circadian rhythm of the core body temperature (Tcore). Core body temperature decreases upon the onset of sleep due to the circadian rhythm; sleep further enhances this effect by keeping Tcore low [7]. The fundamental driving force behind this decrease in Tcore is due to the peripheral skin temperature. Vasodilatation near the peripheral skin allows rapid decreases in Tcore and promotes onset of sleep [5], [7]–[9]. Studies have concluded that elevated room temperature does degrade sleeping quality [10] [11], as sleep and decreases in skin temperature are related to cardiac activity, it has been suggested that the use of heart rate variability (HRV), skin temperature and galvanic skin response (GSR) can infer to the different stages of sleep and indeed this is how wearable such as Fitbit, and Jawbones detect sleeping patterns.

# Related Work

## Literature

The transition from awake to sleep is indistinctive. Ogilve et al reviewed different studies and concluded entry to sleep is a continuous progress that begins with relaxed drowsiness state until entering stage 1 [3]. Once a person has entered sleep, different stages of sleep can be classified using electroencephalogram (EEG). Davis et al demonstrated this in 1937, where they observed different human brain potential during different stages of sleep [4]. Several works have also observed nervous system activity during sleep [5]. Since then, measurements of EEG to detect sleep have become popular and later forms part of todays’ gold standard in sleep study called polysomnography (PSG). PSG is a combined study that measure EEG, electromyography (EMG), electrooculography (EOG) and electrocardiogram (ECG).

Physiological signals that are used for sleep studies can be put into two groups according to where these signals originate from the nervous system. EEG, EOG, EMG signals originates from the central nervous system (CNS), EEG and EOG are obtained from specialized equipment, which would not be ideal for a normal day person to use and therefore not suitable for our application.

Signals such as ECG, blood pressure (BP), skin temperature, skin conductance and respiration originates from the autonomic nervous system (ANS). These signals are relatively easy to obtain as many commercial wearable contains these sensors [6], although clinical trials and validation for these devices are rare hence they are advertised as “activity tracker”. This inaccuracy in the raw data could potentially affect the performance of the system. Currently, there are a few clinical grade wearable; one example is the E4 wristband by Emptica [7]. Nevertheless, there has been increase in using commercial wearable for researches in the mobile healthcare and medical fields.

Heart rate is well known to decrease at sleep onset [9], it can be obtained from two methods, ECG or photoplethysmography (PPG). ECG measurements require at least 3 lead electrodes attached across Einthoven’s triangle, while PPG measures LED absorption when blood flows through the blood vessels. Commercial activity tracker monitors heart rate via PPG sensors, however Lu et al, have shown PPG signals are prone to motion artifacts

[10]. Nevertheless related works primarily focuses on using PPG and other sensors to estimate sleep quality [11], [12].

HRV is a non-invasive and intensively used method to assess the activity level of ANS. It is the inter-change R peaks interval (RR interval) measured from ECG data. It has been observed that the RR interval changes during sleep [13][14][15]. Frequency domain analyses on HRV are common in many works. Power spectrum of high frequency, low frequency and very low frequency were used in Redmond et al works, they used features extracted from these frequency bands to classify different stages of sleep with a subject dependent probabilistic model that achieved accuracy of 87% [16]. The European Society of Cardiology and North American Society of Pacing and Electrophysiology provides a standardized method of HRV, including definitions, methods to obtain features from HRV. These methods are adopted wildly and treated as a gold reference for HRV [17].

Some works have also used accelerometer for sleep quality estimations. Webster et al, developed a scoring based sleep wake recognizer using accelerometer attached to the user’s wrist. Their algorithm summed activity every 2 seconds and was evaluated against sleep / wake status derived from EEG signals. Their algorithm achieved 93.88% accuracy, this suggest the use of accelerometer could be useful [18]. Cole et al, have also presented similar work, their algorithm was able to distinguish sleep from wakefulness for 88% of the time [19].

Body temperature is found to decrease when sleep stage transits from NREM to REM in the third cycle [15]. Secondly, a trend towards a decrease of the low- to high-frequency ratio (LF/HF) derived from the power spectral analysis of Heart Rate Variability is expected to associate with the transition from wakefulness to NREM sleep. Meanwhile, the LF/HF ratio increases during the rapid-eye-movement (REM) sleep[3],[4]. Nevertheless, James W. Mold et.al have concluded that night sweating is associated with several sleep symptoms[5]. Most importantly, body movement monitoring data should be captured as conventional actigraphy method is proved to reach 90% agreement with traditional PSG for nocturnal sleep period [6].

*B. Commercial Products*

TABLE

# System Design and Implementation

## Overall High Level Design

There are many types of software distribution models. Traditionally, users purchase a piece of software either through a retailer or online, and then install it onto their computer. The user then holds the license to use this piece of software, indefinitely. A drawback of this traditional method is that the user normally has to pay an upfront cost, update availability is subject to the package the user bought, and data only exists locally on the user’s machine [7].

Nowadays, the Software-as-a-Service (SAAS) model is the model upon which most companies are building their products around [8]. The SAAS model gives the consumer the ability to use on demand software that is provided by developer via the web or an app. From Heredia et al., as the user normally pays a subscription fee instead of an upfront fee, the SAAS model guarantees that the user will always be using the most updated version of the software as there is no ‘local copy’ of the software to install [9]. Moreover, SAAS removes the burden of having to configure (and control of) infrastructure for the user. However, SAAS solutions often assume that customers will always like new changes as updates are rolled out to all users [10] – this is not applicable to Sleepify as the timeframe of the project means only developing a minimal viable product (MVP).

Following the SAAS model, Sleepify consists of a front-end and a back-end, each consisting of two parts. The front-end is what the user sees and uses, and consists of an iOS application and the web interface. Updates through the App Store and the website ensure the user will always be using the most updated version of Sleepify. Finally, this front-end connects to the sensors for data collection and temperature adjustment. The back-end consists of the servers, databases, APIs, and machine learning modules – these both provide, and accept information from the front-end applications. The user has no information or control on how the back-end is configured; they need not to.

## Sensors

Based on our findings from both academic and commercial sources stated in related works, it is clear that our chosen wearable should provide physiological signals such as heart rate, rr-interval, skin temperature and GSR. The Data-as-a-service platform for healthy lifestyle and preventive medicine (DAPENE) has provided a comprehensive literature on existing wearable technology, it is clear that only products from Fitbit, Jawbone and Microsoft contain the above sensors [6]. Since the release of the first Microsoft Band, researchers have been using it for various works in different fields such as sleep tracking [20]. It can be seen from the table, Microsoft Band 2 provides an extensive range of suitable sensors and APIs, therefore the Microsoft band 2 was chosen.

<NATHALIE TABLE, PLACE ACROSS THE TWO COLUMNS>

## Backend (Server, Database, API)

The backend is responsible for interfacing with the front-end, in accepting and providing it with the information it needs. It consists of a server on which a database resides, and a Representational State Transfer (RESTful) API which allows the iOS app, web interface, and machine learning sections to communicate with the server and by extension, the database.

### The Server

As the database, the web interface, and the API all reside upon the server, a smart choice needed to be made regarding how the server would be implemented.

Our group had prior experience in setting up a server running a LAMP stack (Linux, Apache, MySQL, PHP) in hacking together a simple custom API and website, but this was judged to be inadequate for Sleepify as trying to hand code PHP without a web framework when creating any sort of advanced web app would take an extremely long time. Laravel, and Yii, both modern PHP frameworks, were initially shortlisted as using a modern web framework would shorten development times drastically. However, the verbose and sometimes confusing syntax of PHP mean getting-things-done is more important than code readability [11]. As Sleepify’s development may continue in the future, reusability and code readability meant the group decided not to go with a PHP framework. In hindsight, further research from Srinivasan et al. also showed PHP to suffer from more security issues compared to other web frameworks [12].

Emphasising code readability and rapid development meant the choice was narrowed down to two programming languages: Python, and Ruby. The most well-known web framework in Python is Django, and in Ruby, Ruby on Rails. Both offer extremely fast prototyping and development, extensive documentation, security measures against common attacks, and multiple libraries to assist development. The final decision was to use Django, the Python web framework, as the ease of use of Python (especially considering the group had extensive C++ and Python experience) and the ample documentation on Django meant decreasing the time needed to create the MVP.

Multiple Django libraries were leveraged to add extra functionality, the most notable being: django-rest-framework, a library which provides the skeleton of the RESTful API; django-rest-auth, which extends Django’s already excellent authentication system with the API; django-bootstrap3, a library which simplifies styling a website using Twitter bootstrap.

### The Database

The main design decision when choosing the database was whether to go with a Structured Query Language (SQL) or a NoSQL database.

SQL databases are known as relational databases, where databases are linked together by keys and values [13], held in entries in database tables. In SQL databases, all the incoming data must match the format of the database table, whilst NoSQL operate on the premise that the incoming data is of a large volume and of a rapidly changing format [14].

The most well-known NoSQL database is MongoDB, and it offers several advantages over SQL databases. MongoDB claims scalability and performance improvements in [15], claiming that NoSQL databases are horizontally scalable (add more servers) instead of vertically scalable (have to make the one server more powerful). However, the flexibility of NoSQL data means there exists consistency issues when dealing with many similar data objects – unacceptable for user data. Another important advantage a NoSQL database has is its data format. Nayak et al. goes into more detail, showing the data being held in a binary Javascript Object Notation (JSON) object [16]; this can be accessed using object oriented methods. However, this advantage is nullified with Django as it has its own object oriented wrapper for any type of database. Django supports its own ‘Models’, which abstract away the complicated SQL statements needed to modify the database [17] in favour of treating database tables as objects, nullifying yet another advantage of NoSQL databases.

Hence, the final decision was to use SQL databases. Having a SQL database means structured data (sensor data is of a set format anyway), and relational relationships mean data can be linked with user profiles very easily. There are a few popular SQL databases, the most popular 3 being SQLite, MySQL, and PostgreSQL. The following table shows the pros and cons of each (taken from [18]); the final decision was made to use SQLite, a SQL database that comes shipped with Django by default, with the main justification coming from portability (copy and paste the database across testing machines, committable on Git), and it supporting enough features to not be considered bloated. Scalability issues have been moved down in priority as according to SQLite, they only occur at high volumes of data [19], an unrealistic target for Sleepify. Lastly, NoSQL support is not part of the official Django development effort, and is only supported via third party forks [20].

With regards to user security, the SQLite database is not encrypted now as the database is not reachable through the internet. However, in the future, a Django library known as django-fernet-fields can be utilised to encrypt database fields (unfortunately, floating point numbers are not supported yet).

Table : Comparison of SQL Databases

|  |  |  |
| --- | --- | --- |
| Database | Pros | Cons |
| SQLite | * Extremely portable as the whole database is one file * Feature rich for development and testing | * No in-built user management * Slow for large number of writes |
| MySQL | * Scalable, more feature rich than SQLite, tested * High security | * Reliability issues and discontinued development * Not fully SQL compliant |
| PostgreSQL | * Reliable in terms of data integrity * Supports complex queries | * Overkill for simple databases * Not portable without spending time replicating the database |

### The API

Creating an API was a top priority for the back-end as it enables a consistent communication format between the front and back-ends. Sleepify’s API exposes URLs in which data can be sent or retrieved, including but not limited to the following: machine learning results, raw sensor data, graphs, calendar events, and sending push notifications.

To create the API, a communication format and architectural style had to be decided. Nurseitov et al. compares the two main communication formats, eXtensible Markup Language (XML), and Javascript Object Notation (JSON) [21]. XML follows a rigid pre-defined structure while JSON does not have any pre-defined structures (large companies such as Google, Yahoo, and Microsoft have a web repository of such pre-defined structures but this is by no means compulsory - <http://schema.org/>), so initially it seemed XML was the way forward. However, since everything in XML is stored in strings, parsing the XML data takes relatively more processing power than that of JSON, which can have single entries or arrays of strings or integers – making JSON much more efficient, especially on mobile platforms as demonstrated by Sumaray et al. [22], making it Sleepify’s choice for the data format.

To decide on the architectural style, the pros and cons of Simple Object Access Protocol (SOAP), Representational State Transfer (REST), and Remote Procedure Call (RPC) were compared in the following table [23], [24], [25].

Table : Comparison of API styles

|  |  |  |
| --- | --- | --- |
| Style | Pros | Cons |
| SOAP | * Versatile, can use different protocols other than HTTP | * Extremely complex payload * Way too verbose for simple tasks * Legacy |
| REST | * Uses standard HTTP verbs * Intuitive and clean looking API URLs * Suited for getting data * Embraced by many as the way to go when designing APIs | * Not suited for calling functions or actions (URLs are nouns) * Not easy to do more than one thing in one request |
| RPC | * Uses standard HTTP verbs * Suited for verbs, functions, actions * Can have a custom verb do more than one action at once | * Getting data using RPC architectures is messy and inconsistent * Naming conventions up to the developer |

As SOAP relied on XML, it was not chosen. Based on these results, Sleepify chose to use a mixture of REST and RPC architectures. Data retrieval and insertion would be done using RESTful nouns such as /user/, /raw\_data/, /stats/, while push notifications and the machine learning training would be done using RPC verbs such as /push\_to\_devices/, /migrate\_features/.

As the API is built on the Django server, the django-rest-framework was leveraged to provide the skeleton of the API. Converting functions into RESTful and RPC compliant APIs were as simple as wrapping the function in an ‘APIView’ class. The library also provided an appealing interface to display data retrieved from the API without any added custom user styling. Authentication to the API is done through sessions/cookies, and is supported through defining permission classes in the API functions (e.g. statistics only available to logged in users, user registration, log in/out, open to the public). Another style of authentication is using JSON Web Tokens (JWT), but JWT does not allow pushing notifications to logged in clients as there is no way to know whether a user is logged in or not, as opposed to authenticating using sessions [26].

## Machine Learning

### Sleep quality evalution

The prime objective of the sleep quality evaluation module is to allow real time sleep quality evaluation based on sensor data. This section reviews the related methods in sleep quality evaluation and their deliverables to our final implementation.

There are three widely used methods in clinical sleep quality assessment: Pittsburgh Sleep Quality Index (PSQI), Polysomnography, and Actigraphy [4]. First, PSQI is a questionnaire-based assessment focusing on subjective feedback on medium to long-term sleep quality [5]. Based on subjects’ answers, it generates a score that is inversely proportional to sleep quality (lower is better). Due to the limitation of a long assessment interval, PSQI is not suitable for our real-time implementation. However, this method can, and is, used to evaluate the general performance of our system on sleep quality enhancement; the result of which will be discussed in VI.

Polysomnography analyses sleep quality by using electroencephalograms (EEG), electro-oculograms (EOG) and electromyograms (EMG) of the mentalis and libs [7]. It reflects the precise proportion of each sleep stage during a 24-hours assessment interval, and hence provides the most accurate sleep quality evaluation. Despite its accuracy, it has a few critical disadvantages that prevent its application into our system. First, the sensors required are extremely intrusive to user (located around the head) and all signals require intensive processing algorithms to analyse (how complex? We are using cloud, how much needed per second/per user?). Secondly, the data collection process for complete analysis require at least 12 hours, making it not applicable in Sleepify’s case.

Actigraphy monitors the sleep quality by estimating the ratio between ‘sleep’ and ‘awake’ patterns. Conventionally, ‘sleep’ and ‘awake’ patterns are defined as minor and intense body movements during sleep by using a motion sensing device known as an actometer. The core principle is that body muscles are completely paralyzed during deep sleep stages but not in others. By extending this principle along with redefining sleep-awake patterns and combining more sensors, several actigraphy sleep quality evaluation methods have been invented. Mobile applications such as iSleep [8], Sleep as android [9] and Toss ‘N’ Turn [10] use mobile phone as the main sensor to collect data about the sleeping noise, body movement, and background light intensity. They determine the sleep-awake ratio in each night to evaluate sleep quality with a mean accuracy over 80%. Furthermore, research by Ya-Ti Peng et.al has also shown that introducing heart rate data into normal motion tracking can improve sleep-awake pattern classification [11]. These applications act as a proof of concept for actigraphy validity, as well as the correlation between sleep quality and biometrics including body movement, heart rate, etc. Moreover, they demonstrate the method’s compatibility with a mobile phone. Thus, we decided to utilise actigraphy as our detection principle. We additionally leverage machine learning for the implementation to provide short-term sleep quality evaluation, continuously.

### Clustering Analysis and Features Extraction

Given the sensors provided by Microsoft Band 2, the features generated covers 3 modalities, summarized in Table 3. As sensor data is time based a time interval for feature generation is required. Our final windowing size is 10 minutes of data sampled at 1Hz. This decision provided the best compromise between mobile phone hardware capabilities, feature validity, and data usage, giving around 2MB’s worth of data per night of sleep. We justified this by observing that the typical range of sleep stage transition time varied from 7 to 45 minutes. Thus, it would be optimal to generate features every 45 minutes. However, the window width is constrained by the memory available on mobile phone. Storing 7 sensors’ readings at 1Hz for 45 minutes exceeds the available memory and it also causes long delays in the sending of the data to the server. Therefore, a smaller, 10-minute interval was chosen; it also proved to be effective in Toss ‘N’ Turn [10].

Features for each modality are chosen by referencing existing signal processing techniques. Previous research has shown that the exposure to extreme heat and a humid environment can affect sleep quality – Sleepify uses the mean skin temperature to capture this exposure effect on the core body temperature<http://europepmc.org/abstract/med/10505822>. Moreover, as temperature tends to decrease at the night-time sleep onset but increase when awake [15], the standard deviation can be used to capture the fluctuation of body temperature within each time window. The mean and standard deviation of skin temperature were extracted.

Accelerometer readings are used to reflect users’ arm movement. Instead of using the interval average acceleration over time suggested in [18], we extract the data in terms of mean and standard deviation of squared amplitude shown in Equation 1.

Equation :

The motivation behind this is to capture the movement information every second. The suggested method is effective when the time interval is around 100 seconds, which is much smaller than the chosen 10 minutes’ interval. If a similar approach is used, the excessive smoothing on 10 minutes’ worth of data can remove acute magnitude fluctuations caused by sudden arm movements. Therefore, the mean and standard deviations on the mean squared amplitude are used to extract overall movement intensity and frequency.

Heart rate variability is proved to be higher in rapid eye movement sleep stage than others < Zemaityte D, Varoneckas G, Sokolov E. Heart rhythm control during sleep Psychophysiology 1984;21:279e89> while mean and standard deviation on RR intervals are shown to be adequate measures of HRV during sleep stages transition < Heart rate variability, sleep and sleep disorders>. Additionally, instantaneous heart rate data is also analysed in a similar manner. Apart from using the mean and standard deviation, kurtosis is also used to analyse the extremes. From < "The meaning of kurtosis: Darlington reexamined". *The American Statistician>*, kurtosis is a measure on outlier’s population out of the total samples. The higher the kurtosis is, the less distributed the sample is around the statistical median. Hence, it can be a useful tool to quantify the chronic changes in RR interval and heart rate that is removed under mean and standard deviation. Lastly, we have excluded the introduced Galvanic Skin Response reading as input features because of three reasons. First, the sweat production is proven to be independent with galvanic skin response amplitude < INDEPENDENCE OF GALVANIC SKIN RESPONSE AMPLITUDE AND SWEAT PRODTJCTION\* >. Secondly, the galvanic skin response (GSR) sensor required the band be in a calibrated and ‘locked’ mode to provide accurate data; this is controlled independently by the Microsoft Band 2. Thirdly, the variance of GSR across different sleep quality is found to be nearly zero from data collected. This demonstrates its insignificance in sleep quality evaluation, and hence GSR is excluded in the final feature set.

Table 3: Initial Feature Set

|  |  |  |
| --- | --- | --- |
| Modality | Sensor Data | Features |
| Temperature | Temperature readings in Celsius | Mean, standard deviation(STD) |
| Movement | 3-axes Accelerometer readings | Mean, STD of root mean squared amplitude |
| Heart rate | Optical Heart Rate readings and RR interval | Mean, STD, Kurtosis |

### 3) Model Selection

We have defined our sleep quality evaluation problem as a binary classification problem after evaluating it from the view of various sleep-awake pattern classifiers < Comparison of Sleep-Wake Classification using Electroencephalogram and Wrist-worn Multi-modal Sensor Data >< https://infoscience.epfl.ch/record/135627/files/EPFL\_TH4391.pdf >. Given the final feature set, it is necessary to select an optimal classifier based on the obtained data. MATLAB’s classification learner is used to perform cross-model benchmarking. From previous testing results as shown in Table 4, a user specific classifier performs better than a unified classifier and hence our final model selection process only focus on optimising models which are trained on a specific subject. To prevent a loss of generality, we carefully selected a subject that has the most uniformly distributed sleep quality. The result is shown in Table 5.

Table 4: Feature Analysis Results

|  |  |  |
| --- | --- | --- |
| Models | All Data Accuracies | Personalized Data Accuracies |
| Best Tree | 78.1% | 90.5% |
| Logistic Regression | 78.1% | 85.7% |
| Best SVM | 75% | 95.2% |
| Best KNN | 84.4% | 90.5% |
| Best Random Forest | 75% | 95.2% |
| Boosted Tree | 81.3% | 85.7% |
| Subspace Discriminant | 60% | 100% |

Table : Model Selection Results

|  |  |
| --- | --- |
| Models | Personalized Data Accuracies |
| Decision Tree | 95.8% |
| Logistic Regression | 85.7% |
| Best SVM | 96.4% |
| Best Random Forest | 96.3% |
| Boosted Tree | 95.6% |

From Table 4, it can be observed that either support vector machine or random forest should be chosen as the implementation model. Apart from the model accuracy, training, and testing time are also considered as model selection criteria. Thus, random forest is chosen to be implemented due to its efficient training and testing principles. Another drawback from this testing result is that it is based on feature sets generated from one specific subject over 1 week due to limited resources such as time and available hardware. Moreover, the limited subject diversity also reduced the available sample size. This is because most of data collected are from good sleepers, which causes an imbalanced sleep quality distribution. Nevertheless, the sleep quality labelling on training data assumes that overall night sleep quality can be interpolated into individual interval sleep quality. This assumption should be abandoned if more time and resources are given to perform clinical testing on overnight sleep quality monitoring with the device.

### Machine Learning Model – Server Deployment

To integrate the machine learning module into the system smoothly, an implementation of the random forest classifier is done on the server side to provide online estimation upon requests from Sleepify’s clients.

As we have chosen Python-Django for the server development platform, Scikit-Learn [27] (a Python library) is used for our machine learning implementation. Model persistence on server is achieved by binary serialisation and recovering using Pickle, a Python object serialization tool. From the preliminary study results, it is necessary that each user requires a specific classification model which is trained by personalized data. Therefore, these binary models files are linked to user entries in the database to allow user specific mapping.

To communicate with the mobile application, the Sleepify API is extended to offer three functions: sensor data storage, machine learning model retraining, and prediction. When the mobile application sends a packet of new sensor data to the server, this interface will first extract the date, time and user details of the packet and store the data into database under specific user accounts. After that, the packet will be pre-processed into a feature sample. Then, binary model files under the specific user are recovered, and prediction methods are called with the generated feature vector. Finally, the prediction outcome is sent back to mobile application in the form of a JSON object. Continuous learning is supported using existing models provided by Scikit-Learn, preventing the situation of having to retrain the model completely. A user specific data tracker is implemented to monitor the accumulated count of untrained data packets. When it exceeds a threshold, the model will be recovered from the binary file to perform online learning. The block diagram for this machine learning server infrastructure is shown in Figure 1.



Figure : Server Architecture for ML model

### Testing

To evaluate the model implementation, offline model testing is done to demonstrate the practical performance of random forest model provided by Scikit-learn. To compare with testing performed by the MATLAB classification learner, identical hyperparameters such as tree depth and tree numbers are used. Moreover, a similar testing and training set splitting ratio is used - 0.2 and 0.8 respectively. The result shows that implemented classifier can achieve 90% accuracy, similar to the MATLAB implementation.

## Frontend (iOS Application and Website)

### The iOS application and Homekit

The iOS platform also has the added advantage of having a *HomeKit* framework, which accommodates the use and incorporation of other *HomeKit* compatible devices into our system. *HomeKit* allows third-party access to the home configuration database, to display, edit the accessories, and perform actions. Please note that there is a *HomeKit* Framework for developing *HomeKit* related products. [1] *Homekit*’s developer guide is available on the Apple official website, which includes the sample codes on application development. [2] They are, however, not yet updated to *Swift 3*. OOPer on github [3] has converted Apples sample code from *Swift 2.3* to *Swift 3*. These codes not only provide a basic framework on the linkage to the *Homekit* products, they are also used in our mobile application.

As the aim of the project is to change the room temperature, we have added units to control this. An ‘additional temperature sensor’ and an ‘on-off smart plug’ are needed to measure the room temperature, and to provide feedback to control the heater. We have selected *HomeKit* compatible devices, i.e. the *Elgato Eve Room* and the *Elgato Eve Energy* for easier collaboration. In addition to the measurement of room-temperature, the *Elgato Eve Room* is a wireless indoor sensor for measuring air quality and humidity; while the *Elgato Eve Energy* functions as a smart plug.

### iOS User Interface Design

Most of the mobile apps such as Sleepbot [28] and Sleepcycle [29] only provides an interface that allows the user to only read data on sleep quality. Our mobile application aims to provide a simple and intuitive interface for the users to control the various hardware that are linked to the system, as well as to read the important data, such as target temperature and the quality of sleep. It has a minimalistic design with a login page and four other main pages as follows:

*(1) Login:* The user would first be prompted to the login page of the mobile application upon launch. Buttons are also present to navigate to the registration page and the ‘forget password’ page for the user to register an account without having the need to access the website. Upon successful logging-in, the user will be prompted to a tab bar controller consisting of four tabs – a control tab, a reminder tab, a data tab and a logout tab.

*(2) Control of HomeKit products*: The control page consists of a table view where the database of the configuration of *HomeKit* products is automatically loaded. It is in a hierarchy structure, where the list of homes are first listed, the user can choose to view the different accessories in each home the home, and choose to show the configurations of each accessory. This “top-down” hierarchy has the advantage of reducing distraction or information overload to the user. [] The accessories listed includes the *Eve Room* and *Eve Energy* that are being used in the temperature control module of our product. Also, the user can perform actions easier on the different products – he/she no longer has to leave the application to view or change the configurations at the ‘Home’ page.

*(3) Data collection*: The data page is the main page of the mobile application. It is where the user can send the data unto the server, read the real time room temperature, body temperature and the target body temperature for sleep optimization. The user interface will have a ‘start’ and a ‘stop’ button for the user to commence recording data from the *Microsoft* band and *Eve Room* when about to sleep, and to end when the user wakes up. Throughout the period when the user is asleep, data will be uploaded to the web server every 10 minutes in a *json* format, to classify the sleeping quality using machine learning. Feedback of the target body temperature and sleeping quality will be sent back to the mobile application in real time and displayed on the user interface. An output textbox at the bottom of the screen is there to keep the user informed of what is happening during the data collection process. [] In case the user does not want to follow the target body temperature generated by the machine learning algorithm, there is an ‘override’ button for the user to set his/her own desired target room temperature using the circular slider on the top of the page, which allows intuitive control of favourable room temperature. Lastly, a switch is present for the user to input whether he/she has had a good night, for further machine learning purposes.

*(4) Calendar integration:* The reminder page is an override function in addition to the automatic calendar integration. This is more convenient as it allows the user to set a reminder to sleep early. There is a ‘date-picker’ in the middle to input the date and time where the user wishes the reminder to be set. The reminder content can be entered in the text field below the ‘date-picker’. A reminder can then be set easily by clicking the ‘set reminder’ button.

*(5) Logout:* The logout page has a ‘logout’ button. Upon successful logging-out, the user will be prompted back to the login page, which is where the application was first launched.

### Implementation of Communication

The internet communication is essential to the *Sleepify*’s mobile application and backend server as the system requires large amount of data transmitted in regular time interval when user is sleeping. Alamofire is an HTTP networking library written in Swift for iOS and Mac OS X. It consists of an elegant interface that simplifies loads of Apple’s Foundation common networking tasks [30]. To obtain and deliver the physiological data and machine learning algorithm output, we need to use chainable request or response methods to deserializing JSON in both the backend and frontend applications. Alamofire provided various type of GET and POST request API for performing data transmission. By encoding data into JSON format, we are able to upload and download data between mobile application and backend server through implemented URL from the backend. We used NSDictionary class in Swift to set for searching key and obtain the corresponding data from backend. The general communication channels are listed as below:

1. When user wants to register a new account, the corresponding email, username and password will be sent to the backend server.
2. When user logon the system successfully, username will be sent to server for future usage.
3. When start button is pressed, counter will measure the data buffer size and upload the collected physiological data to backend once it is full (every 10 minutes).
4. When stop button is pressed, the total physiological array and user sleeping quality feedback will be uploaded to the backend for further classification.

### Implementation of Security

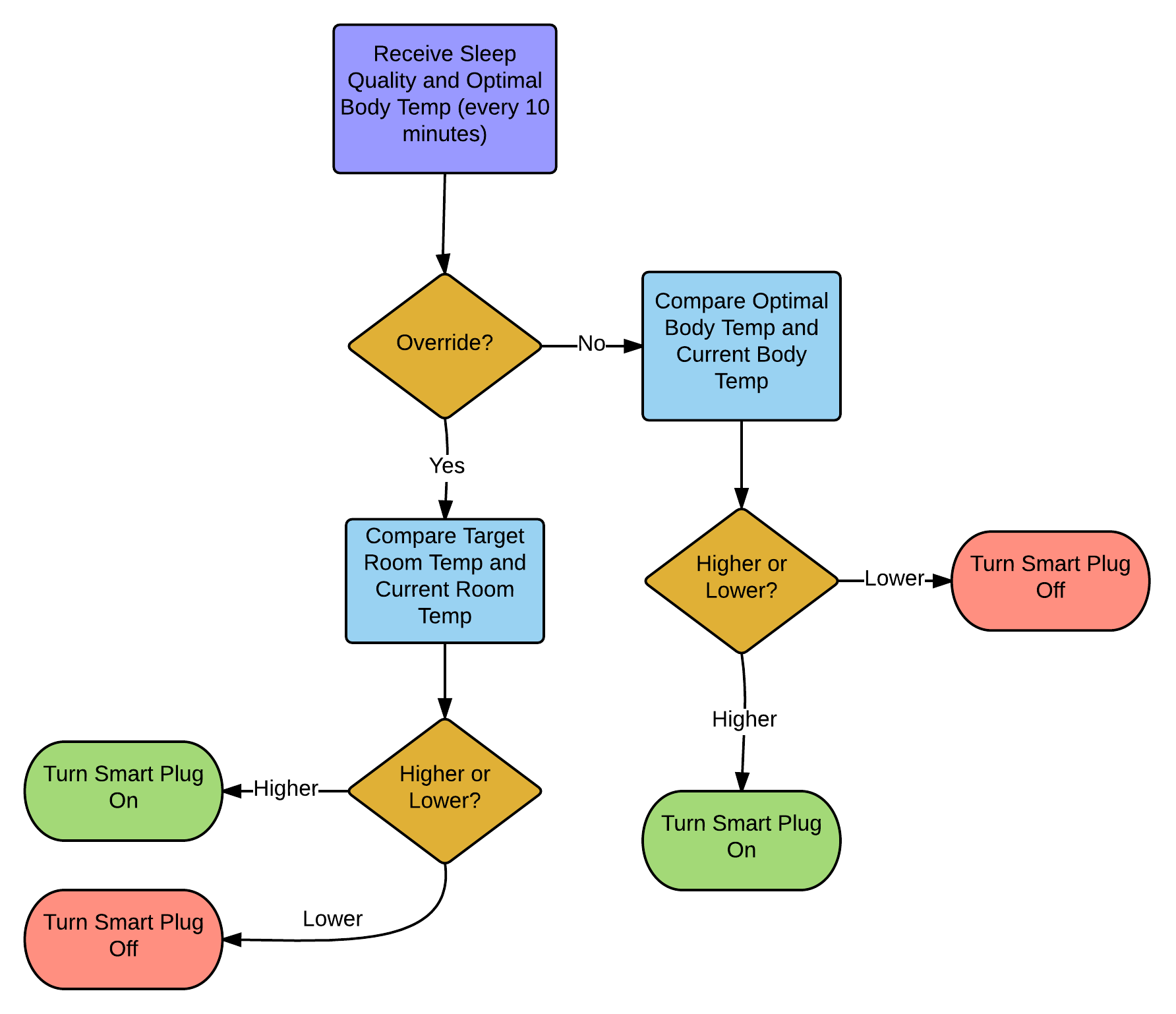
To maintain the network and data usability and integrity, we need to protect the network infrastructure from unauthorized activities. Data encryption and user authentication are the common methods to protect the user data and authorize user to perform any human-to-machine interactions through any type of networks to access the connected systems and resources [31].

Cross-site request forgery (CSRF) is an attack vector that allow unauthorized commands transmitted by sending arbitrary HTTP requests from a user that is trusted. CSRF vulnerabilities have been known since 2001 and many websites became victim of this type of attack. Therefore, prevention must be made for every communication through internet. There are several CSRF prevention techniques by embedding additional authentication data into requests in order to detect unauthorized usage [32].

To secure everyday user’s private sleeping physiological data, we have designed a login system for each user. With this personal account, users are able to utilize their personal username and password to access *Sleepify*’s web interface and mobile application securely. By implementing the Cookie-to-Header Token with backend server, we are able to use this anti-CSRF technique through including the token in the headers of every HTTP requests. We have also setup the AES256-CBC encryption for further protecting every data in future. Once the system is completed, we can use this library to encrypt the data we want to protect.

### Implementation of the Feedback System

As the users woke up everyday, the mobile application will require them to feedback their sleeping quality as good or bad. So that the backend server can classify the physiological data and calculate the optimal body temperature for future usage. Ideally, when the user is sleeping, the backend server will provide an optimal body temperature for the system to adjust the heater every ten minutes. If the user is having a bad sleeping quality at the moment, this optimal body temperature will then compare with the user’s currently body temperature. The system will control the smart plug connecting to the heater with respect of this comparison to achieve the optimal body temperature. If the override button is on, the system will control the heater according to the comparison between the target room temperature and current room temperature. Any changes on body temperature will be neglected. The flow of the system is shown as below:



### Application Testing

Throughout the implementation, we have reduced the data buffer size from 10 minutes to 10 seconds in order to test our code quicker. By checking the HTTP status code, we are able to detect whether the connection with backend server is successful or failure. The backend server terminal console also allows us to check the correctness of data received. Using a hairdryer instead of heater can change the received body temperature of the Microsoft Band faster.

## Web Interface

As per Sleepify’s promise, the web interface should be modern, intuitive, and easy to use. The following three design principles were followed during development:

1. Compatibility with mobile devices
2. Use existing HTML styling frameworks to assist development, provide compatibility with mobile
3. Simple navigation buttons should provide directions to all parts of the website
4. Prioritise reader comfort and readability of text, use neutral colours in graphs and text

To tackle the compatibility with mobile problem, 1., responsive web design was a priority in the development of the web interface. By utilizing viewports (show different amounts of data based on the device width and display density) as described in [33], [34], a responsive web design was created as a product of extensive testing on different resolutions (from mobile screen resolutions such as 640x960 to quad HD resolutions such as 2560x1440).

To tackle the framework problem of 2., Twitter Bootstrap was used as the library of choice as it is lightweight (only a few static files needed for setup), flexible (allows overriding of default styles with custom ones), and powerful. Bootstrap is also built for mobile first when compared to other styling frameworks such as Foundation and Skeleton. Bootstrap gives predictable websites (which can also lead to a lot of perfectly functional websites looking similar) at the cost of slightly verbose HTML [35]. By following code styles and practices from [36], a navigation bar, a jumbotron (big heading type text at the top of every page to provide context), sidebars, and footers were created as the base template for Sleepify, thereby fulfilling 3.

No. 4 is quite subjective, however, Sleepify adhered to the design principles shown in [37], to create a flat, minimalistic, and stylish website, as opposed to other design styles such as skeuomorphism [37]. By respecting Bootstrap’s grid system and whitespace, as well as choosing a modern font (Josefin Sans), Sleepify achieved 85.7% positive feedback after being shown to 14 non-Sleepify people. Some constructive feedback such as ‘*text too small’*, and ‘*homepage has too much left/right margin whitespace’* were taken into consideration in the final design.

# Evaluation Criteria and Setup

# Results

# Discussion

# Conclusion

In conclusion, this report highlighted the motivation behind in building a system that is capable to alter the users thermal sleep environment to achieve better sleep quality. We have identified that the thermal environment is a key factor in affecting sleep quality, this justifies our rational in controlling this factor in order to provide better sleep quality to users. We have discussed related works, however to the authors’ knowledge there is no work that has developed a complete system to alter sleeping environments. Finally, we have also presented some of our prelimiarly works.

# References

[1] ‘Enter the Matrix: App Retention and Engagement’, *Flurry Blog*. [Online]. Available: http://flurrymobile.tumblr.com/post/144245637325/appmatrix. [Accessed: 19-Mar-2017].

[2] ‘App Retention Improves - Apps Used Only Once Declines to 20%’. [Online]. Available: http://info.localytics.com/blog/app-retention-improves. [Accessed: 19-Mar-2017].

[3] M. Malik, ‘Heart rate variability’, *Ann. Noninvasive Electrocardiol.*, vol. 1, no. 2, pp. 151–181, 1996.

[4] E. Vanoli, P. B. Adamson, null Ba-Lin, G. D. Pinna, R. Lazzara, and W. C. Orr, ‘Heart rate variability during specific sleep stages. A comparison of healthy subjects with patients after myocardial infarction’, *Circulation*, vol. 91, no. 7, pp. 1918–1922, Apr. 1995.

[5] J. W. Mold, J. H. Woolley, and Z. Nagykaldi, ‘Associations between night sweats and other sleep disturbances: An OKPRN study’, *Ann. Fam. Med.*, vol. 4, no. 5, pp. 423–426, Oct. 2006.

[6] A. Sadeh, K. M. Sharkey, and M. A. Carskadon, ‘Activity-based sleep-wake identification: an empirical test of methodological issues’, *Sleep*, vol. 17, no. 3, pp. 201–207, Apr. 1994.

[7] ‘What is SaaS’, *Interoute*, 01-Mar-2017. [Online]. Available: http://www.interoute.com/what-saas. [Accessed: 19-Mar-2017].

[8] J. P. • M. 30 and 2016, ‘Top 25 Most Popular SaaS, Cloud Applications for Business’, *ChannelE2E*, 30-Mar-2016. [Online]. Available: https://www.channele2e.com/2016/03/30/top-25-most-popular-saas-and-cloud-applications/. [Accessed: 19-Mar-2017].

[9] A. Heredia, R. Colomo-Palacios, and A. de Amescua, ‘Software Business Models from a Distribution Perspective: A Systematic Mapping Study’, *Procedia Comput. Sci.*, vol. 64, pp. 395–402, Jan. 2015.

[10] ‘Weighing the Pros and Cons of a SaaS Solution’, *AT&T Networking Exchange Blog*, 28-Jul-2016. [Online]. Available: https://networkingexchangeblog.att.com/enterprise-business/weighing-pros-cons-saas-solution/. [Accessed: 19-Mar-2017].

[11] S. Das, ‘Which is Better, PHP or Python? A Developer’s Take’, *LinkedIn Pulse*, 11-Jun-2015. [Online]. Available: https://www.linkedin.com/pulse/which-better-php-python-developers-take-srikrishna-das. [Accessed: 19-Mar-2017].

[12] S. M. Srinivasan and R. S. Sangwan, ‘Web App Security: A Comparison and Categorization of Testing Frameworks’, *IEEE Softw.*, vol. 34, no. 1, pp. 99–102, Jan. 2017.

[13] tutorialspoint.com, ‘SQL RDBMS Concepts’, *www.tutorialspoint.com*. [Online]. Available: https://www.tutorialspoint.com/sql/sql-rdbms-concepts.htm. [Accessed: 19-Mar-2017].

[14] ‘NoSQL Databases Explained’, *MongoDB*. [Online]. Available: https://www.mongodb.com/nosql-explained. [Accessed: 19-Mar-2017].

[15] ‘MongoDB at Scale’, *MongoDB*. [Online]. Available: https://www.mongodb.com/mongodb-scale. [Accessed: 19-Mar-2017].

[16] A. Nayak, A. Poriya, and D. Poojary, ‘Type of NOSQL Databases and its Comparison with Relational Databases’, *Int. J. Appl. Inf. Syst.*, vol. 5, no. 4, Mar. 2013.

[17] ‘Models | Django documentation | Django’. [Online]. Available: https://docs.djangoproject.com/en/1.10/topics/db/models/. [Accessed: 20-Mar-2017].

[18] ‘SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems’, *DigitalOcean*. [Online]. Available: https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems. [Accessed: 20-Mar-2017].

[19] ‘Implementation Limits For SQLite’. [Online]. Available: https://www.sqlite.org/limits.html. [Accessed: 20-Mar-2017].

[20] ‘NoSqlSupport – Django’. [Online]. Available: https://code.djangoproject.com/wiki/NoSqlSupport. [Accessed: 20-Mar-2017].

[21] N. Nurseitov, M. Paulson, R. Reynolds, and C. Izurieta, ‘Comparison of JSON and XML Data Interchange Formats: A Case Study’, Department of Computer Science Montana State University – Bozeman Bozeman, Montana, 59715, USA.

[22] A. Sumaray and S. K. Makki, ‘A Comparison of Data Serialization Formats for Optimal Efficiency on a Mobile Platform’, in *Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication*, New York, NY, USA, 2012, p. 48:1–48:6.

[23] ‘REST vs XML-RPC vs SOAP’. [Online]. Available: http://effbot.org/zone/rest-vs-rpc.htm. [Accessed: 20-Mar-2017].

[24] ‘REST vs XML-RPC vs SOAP – pros and cons : Max Ivak Personal Site’. .

[25] ‘How REST replaced SOAP on the Web: What it means to you’, *InfoQ*. [Online]. Available: https://www.infoq.com/articles/rest-soap. [Accessed: 20-Mar-2017].

[26] R. Golwalkar, ‘Pros and cons in using JWT (JSON Web Tokens)’, *Rahul Golwalkar*, 21-Aug-2016. .

[27] ‘scikit-learn: machine learning in Python — scikit-learn 0.18.1 documentation’. [Online]. Available: http://scikit-learn.org/stable/. [Accessed: 22-Feb-2017].

[28] SleepBot, *SleepBot - Sleep Cycle Alarm*. SleepBot, 2013.

[29] ‘Sleep Cycle alarm clock on the App Store’, *App Store*. [Online]. Available: https://itunes.apple.com/gb/app/sleep-cycle-alarm-clock/id320606217?mt=8. [Accessed: 01-Feb-2017].

[30] ‘Alamofire Reference’. [Online]. Available: http://cocoadocs.org/docsets/Alamofire/4.3.0/. [Accessed: 21-Mar-2017].

[31] ‘What is user authentication? - Definition from WhatIs.com’, *SearchSecurity*. [Online]. Available: http://searchsecurity.techtarget.com/definition/user-authentication. [Accessed: 21-Mar-2017].

[32] ‘Cross-Site Request Forgery (CSRF) Prevention Cheat Sheet - OWASP’. [Online]. Available: https://www.owasp.org/index.php/Cross-Site\_Request\_Forgery\_(CSRF)\_Prevention\_Cheat\_Sheet. [Accessed: 21-Mar-2017].

[33] C. Sharkie and A. Fisher, *Jump Start Responsive Web Design*, 1 edition. Collingwood, VIC, Australia: SitePoint, 2013.

[34] T. Firdaus, *Responsive Web Design by Example*. Birmingham: Packt Publishing, 2013.

[35] ‘What are the pros and cons of using Bootstrap in web development? - Quora’. [Online]. Available: https://www.quora.com/What-are-the-pros-and-cons-of-using-Bootstrap-in-web-development. [Accessed: 21-Mar-2017].

[36] D. Cochran, *Twitter Bootstrap Web Development How-To*. Birminghan, UK: Packt Publishing, 2012.

[37] A. Pratas, *Creating Flat Design Websites*. Birmingham, UK: Packt Publishing, 2014.