EverSleep Dashboard

Shea Styer, Alex Michael, Jacob McBee, Jensen Krone

June 20, 2018

Contents

1 Introduction **1**

2 Requirements **2**
2.1 Functional Requirements 2
2.2 Non-Functional Requirements 2

3 System Architecture **2**

4 Technical Design **4**
4.1 Access Control 4
4.2 Overall Design 4

5 Design Decisions **6**

6 Results **6**

7 Appendix - Usage Guide **6**

1 Introduction

EverSleep is a startup company in Golden, CO that is aiming to help the average American better understand their sleep and hopefully improve it. Their product (Figure 1) is a finger sensor that takes multiple measurements every minute, and interfaces with their mobile app to analyze the data and provide coaching to improve sleep. These measurements include pulse, blood oxygen saturation, snoring, movement, and several other metrics.

Figure 1: The EverSleep Tracker being assembled

While the idea of measuring these statistics throughout the night is not a new idea, it has only been implemented in expensive sleep labs that are only available to certain people. Even then, it is far too expensive to constantly attend these sleep labs and the results are rarely representative of how a person sleeps in their own home.

As the company grows and amasses more data, the founders of EverSleep hope they can run statistical models on this data and eventually make advancements in sleep research.

When users dont understand the results the mobile app gives them, they contact EverSleep for help. EverSleep employees then retrieve the users result from the database in order to help them. The previous web app for doing this had very little working functionality, however. Even the basic search function for finding data by user ID did not work properly. The project was to either fix or replace this web app, first getting the basic search function to work, then adding as much functionality as possible within the remaining time. Due to the confusing design of the previous app, the team decided to replace it.

2 Requirements

2.1 Functional Requirements

- Search for reports by any combination of variables, including user ID, date range, duration, and any other parameter stored in the database
- View and download graphs of collected data for a single report
- View numerical statistics for a single report
- View and download graphs of a single user's data over time for five key metrics: sleep quality, blood oxygen saturation, pulse rate, motion, and snoring
- Delete reports from the database
- View platform and version statistics for the mobile app

2.2 Non-Functional Requirements

- Written in JavaScript using Node.js, Express.js, and various Node.js libraries
- Pug (formerly known as Jade) was used as the templating engine. It allows a more concise method for writing HTML, as well as adding support for inline JavaScript.
- Hosted on an AWS server running Ubuntu 16.04 LTS
- Uses an API to interface with a PostgreSQL database, which is hosted on another AWS server
- Source code is hosted on a third AWS server using GitLab

3 System Architecture

The overall system architecture is shown in Figure 2. The mobile app and the EverSleep product, end users sleep data is collected and entered into the database. The Node.js application uses an API to communicate with the database. EverSleep employees can then use the Node.js application to retrieve data in a human-readable format.

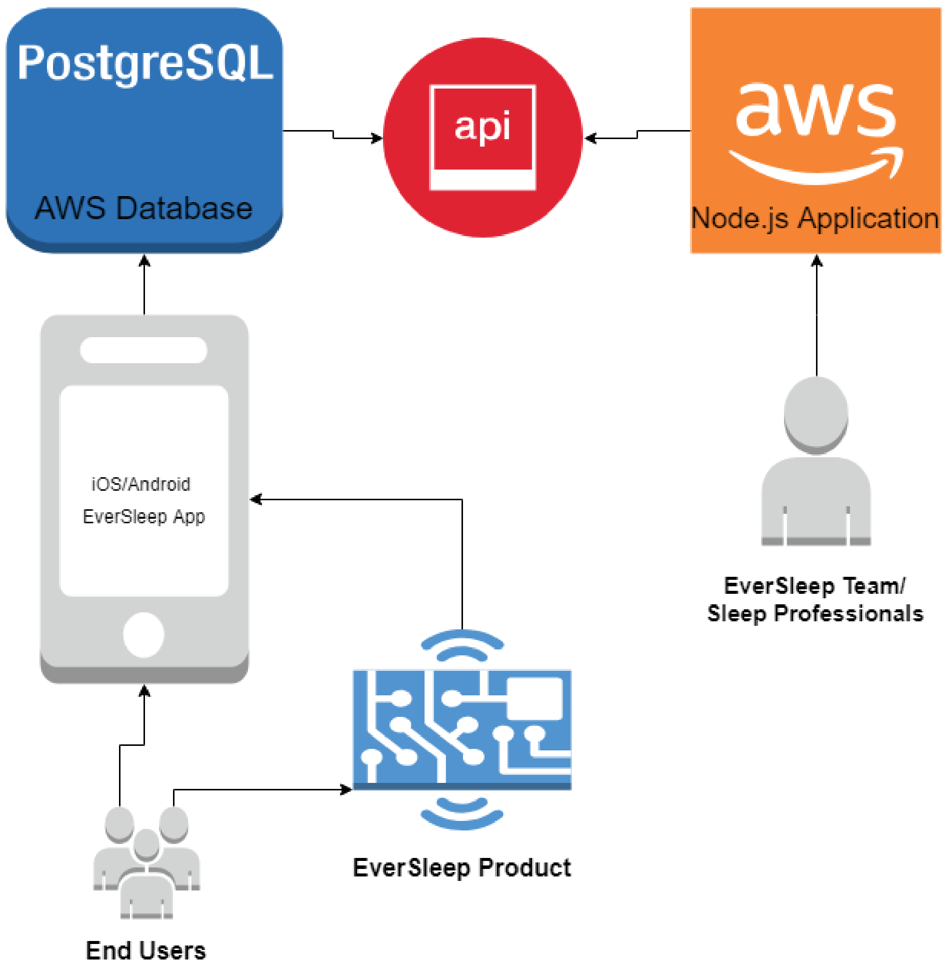


Figure 2: The general architecture of the system

The design of the API is outlined in Table 1. There are four general groups of API actions: records, trends, users, and system. Notice that there is no POST method for records. This is because this tool is only for viewing data collected by the mobile app, not for modifying existing entries or adding new ones.

Action	Method	Path
Retrieve list of records	GET	/api/records
Retrieve individual record info	GET	/api/records/:record_id
Delete individual record	DELETE	/api/records/:record_id
Retrieve trend data for variable	GET	/api/trends
Retrieve list of users	GET	/api/users
Add user to login	POST	/api/users/add
Delete user from login	DELETE	/api/users/:username
Retrieve platform data	GET	/api/system/platform
Retrieve version data	GET	/api/system/version

Table 1: API calls used by the app

4 Technical Design

4.1 Access Control

Access control is handled using three Node.js libraries: Passport.js, bcrypt.js, and Sequelize. Figure 3 shows how these components interact with each other. After the Node.js server receives the users login credentials, it passes them into a Passport.js authentication function. Passport.js uses strategies to define the authentication functions. The team used the local strategy provided by Passport.js, since the usernames and hashed passwords are stored in the same database as the sleep reports. After receiving the user credentials, Passport.js uses Sequelize to query the PostgreSQL database and retrieve the hashed password that matches the entered username. It then passes the entered password and the saved password hash into a bcrypt.js function which compares them. The authentication function then returns a boolean, indicating to Node.js whether the user was authenticated.

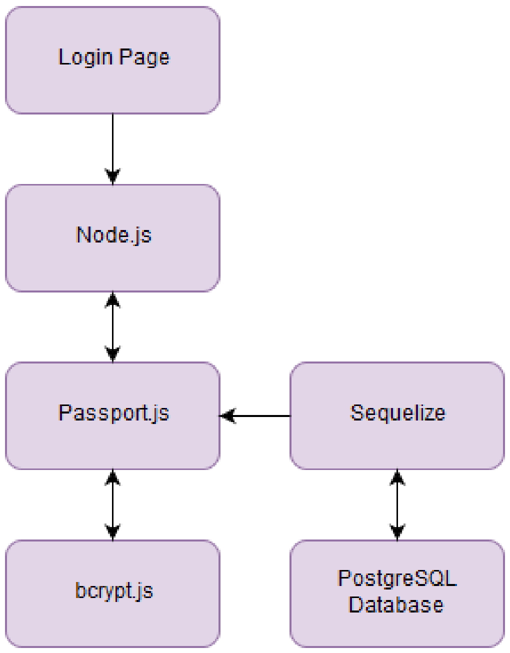


Figure 3: The general design of the access control system

4.2 Overall Design

The UML diagram in Figure 4 shows the overall design of the app. In order to access the database, the request is sent through an api, where the request is handled. Most of the Jade files use a script in order to change the site with interaction, whether that be adding a filter, or changing a date range. Security for the site uses Passport.js and bcrypt.js to ensure that the user has the proper credentials before allowing them to access the site or API.

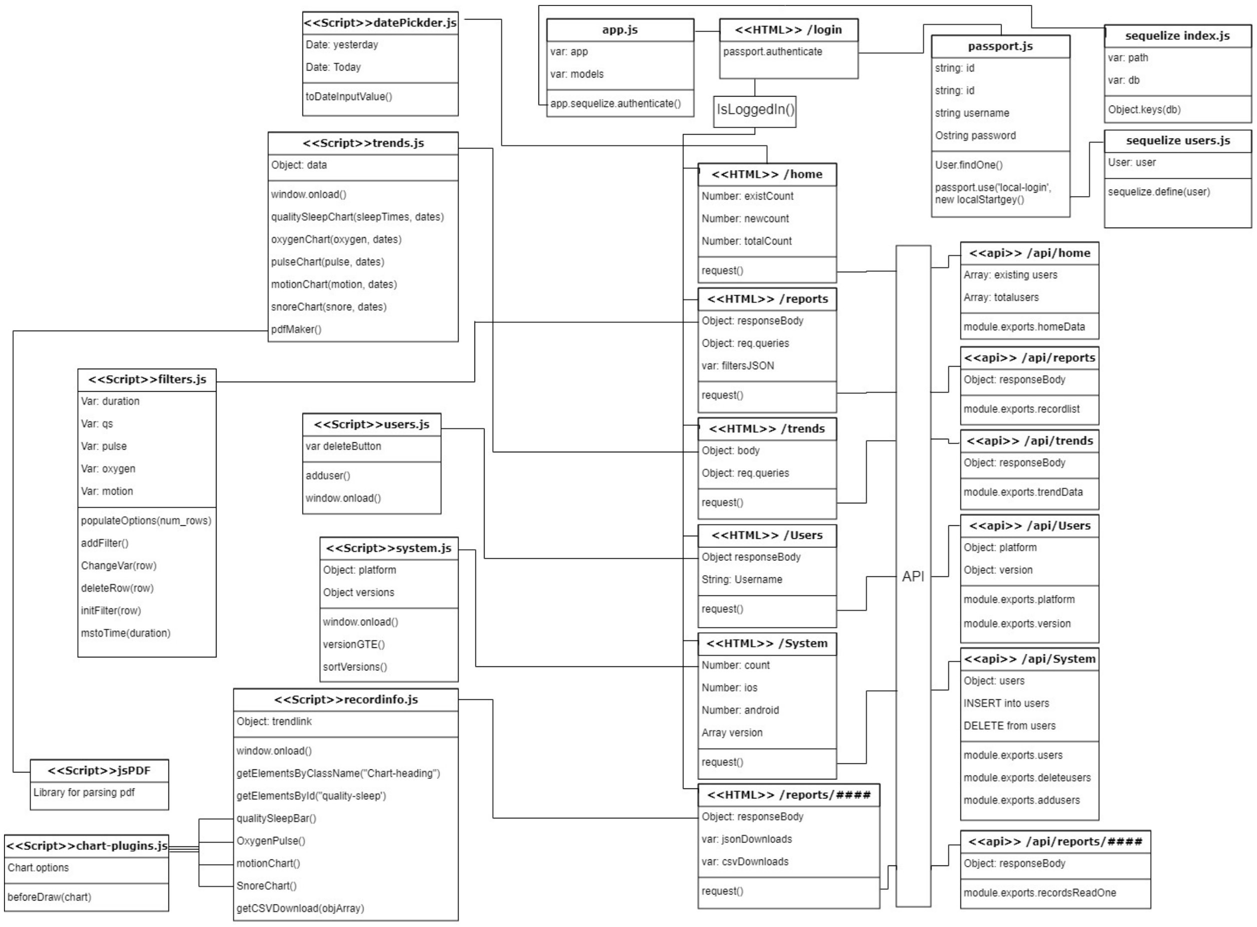


Figure 4: UML diagram showing the design of the app

There are five primary routes used by the app: home, records, trends, users, and system. The pages the routes correspond to are described briefly here. Screenshots and more detailed descriptions can be found in the instructional guide located in the Appendix.

The home page displays some general information about how many people are using the product, showing the new users, returning users, and total users for a given interval. This interval defaults to the previous night, but can be changed using date fields on the page.

The records page, by default, lists the 15 most recent reports. The user can add filters to search by any parameters stored in the database. The records route also has a sub-route which allows viewing information for a single report. This page displays all of the useful data collected for a single user during a single night. Much of this data is displayed graphically.

The trends page shows graphical representations of sleep data for all reports recorded by one user.

The users page allows management of the web apps user accounts. Currently, it allows adding and removing users.

The system page shows charts displaying data about usage of the mobile app. One chart shows how many users are using each platform, the other shows the same information about the mobile app version.

5 Design Decisions

- Node.js, Express.js, and Pug used because they are standard technologies for developing web apps. As a result, they all have good documentation and support, which was useful because only one of the four team members had prior experience with web development.
- Passport.js was used to handle user authentication because it is the standard library to use for that purpose.
- bcrypt.js was used for hashing passwords because it was used in the previous web app. Using the same hashing algorithm allowed the team to avoid resetting all the existing passwords.
- Sequelize and pg-promise were used to interface with the database. Two different libraries were used for this task because Sequelize uses a model system which the team found more useful for querying user information, whereas pg-promise uses a more flexible promise system, which the team found to be better for the dynamic queries used for searching reports
- Chart.js was used for rendering graphs because it is open-source, well-documented, and easy to use
- An API system is used to interface with the database because it will simplify updating the app if the database system is ever changed

6 Results

The goal of the project with EverSleep was to remake their dashboard website so that employees working at EverSleep could easily find user information to help when customers contact them about their sleep results. Overall, this goal was accomplished.

There was one major feature that the team did not have time to implement: trends for multiple users based on filters like age, location, and habits. This was not a feature required by the client; it was one of the low-priority requests.

The team tested the app on both Firefox and Chrome. In addition to verifying that all pages functioned, this testing included attempting to bypass the user authentication and submitting nonsensical queries. A prototype was delivered to the client (in the form of being deployed to AWS) for testing as soon as user authentication was working properly. Since that time, the version on AWS has been updated whenever new features were working on the testing environment. In this way, it has undergone continuous acceptance testing.

In order to ensure usability with future people working with the site, an information guide was made to assist with navigation and explain all the features.

Future work for this product could include adding the trends that allow for multiple users' data to be plotted, as well as writing unit and integration tests.

There were many lessons learned with this project, with statements querying the database for the list of records, it was found to be much faster to only query what was needed instead of everything. Our example of this was that our initial query did SELECT * and for asking more more than 40 results, it would take over 10 seconds for the query to return. We found that if instead we just did select id, it would greatly speed up query times to the point where we could load the entire databases ids.

7 Appendix - Usage Guide

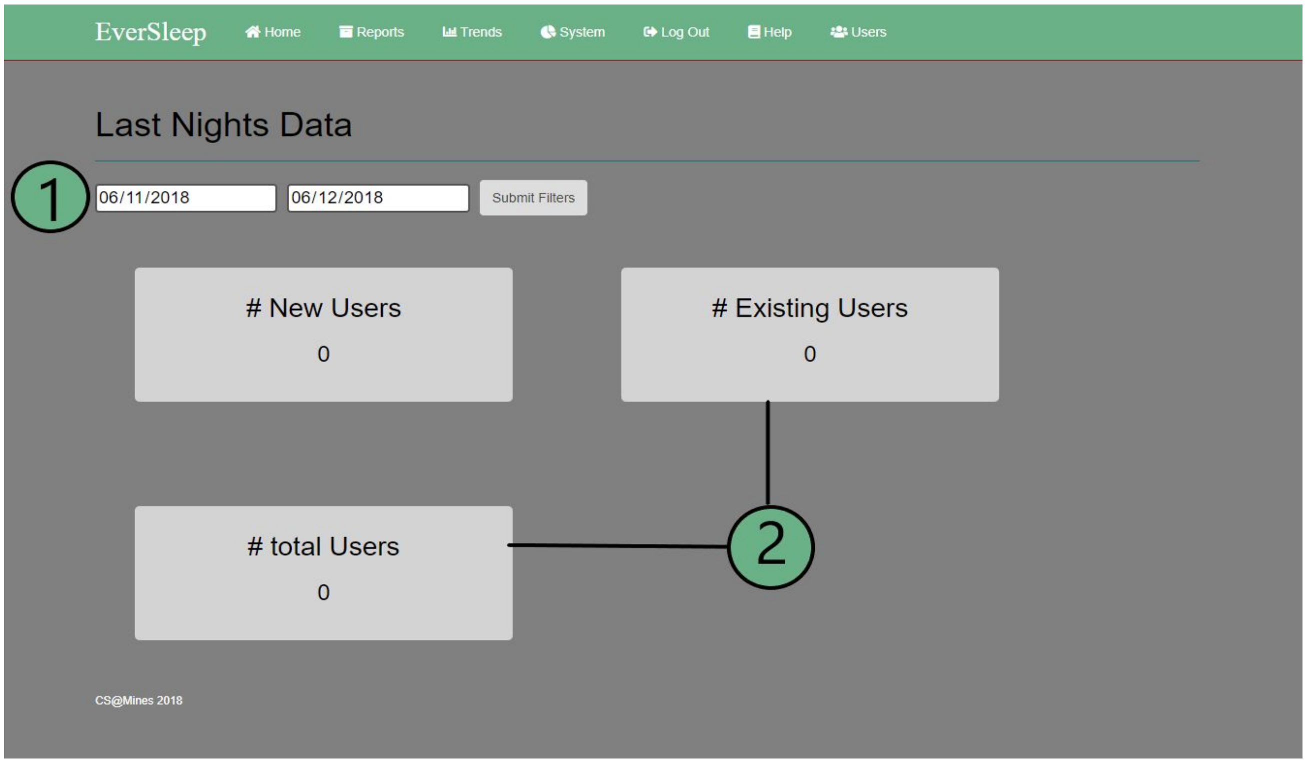
EverSleep

Dashboard Guide

Index:

1. Home
2. Report List
3. Filters
4. Record Page
5. Trends
6. System Info
7. Users

Home Page [Back to index]



Last Nights Data

06/11/2018 06/12/2018

New Users
0

Existing Users
0

total Users
0

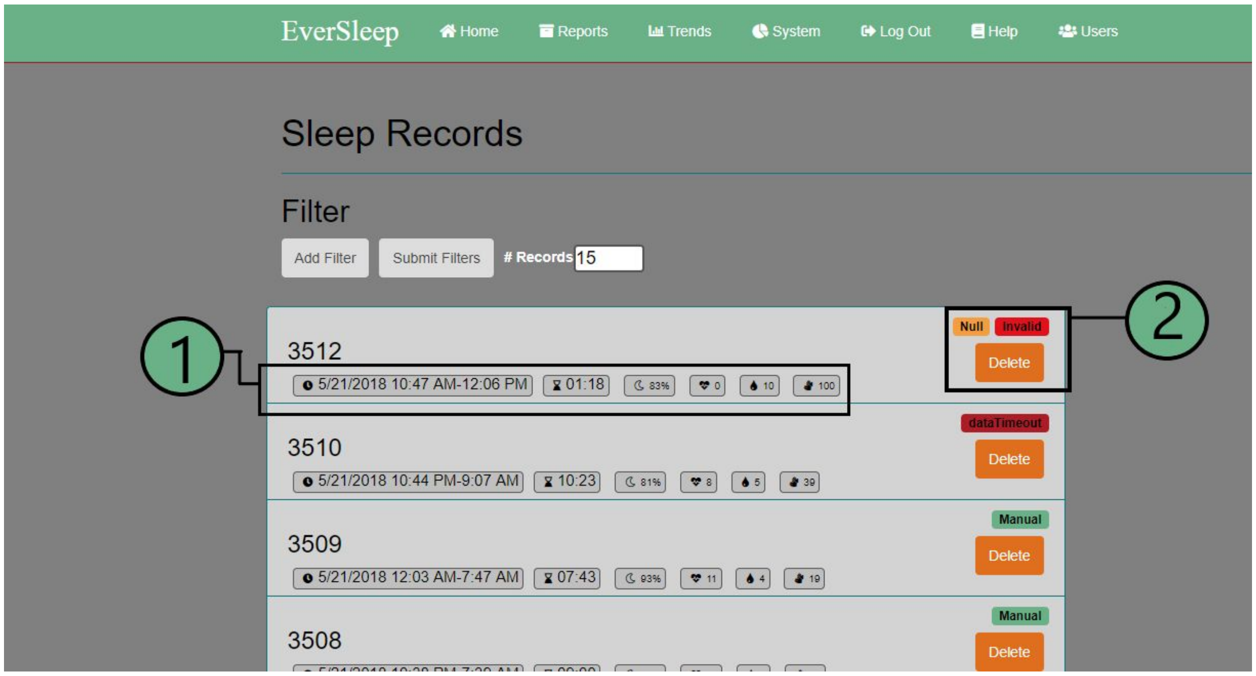
CS@Mines 2018

The home page is designed to show the user quick information about report volume. The default is from the previous day to the beginning of the current day.

1 The values of the date range will begin at yesterday to the current day, but these values can be changed to show any set number of days. This will change the data shown, as described in 2

2 New Users indicates how many new users (by UUID) have used the app Existing Users shows how many returning users submitted a report. (Note that these numbers may be off since the app can reset the uuid if uninstalled and reinstalled.

Report List [Back to index]



Sleep Records

Filter

Add Filter | Submit Filters | # Records 15

3512 — Null | Invalid — Delete
5/21/2018 10:47 AM-12:06 PM | 01:18 | 83% | 0 | 10 | 100

3510 — dataTimeout — Delete
5/21/2018 10:44 PM-9:07 AM | 10:23 | 81% | 8 | 5 | 39

3509 — Manual — Delete
5/21/2018 12:03 AM-7:47 AM | 07:43 | 93% | 11 | 4 | 19

3508 — Manual — Delete

1 The icons following the report info give some brief info:

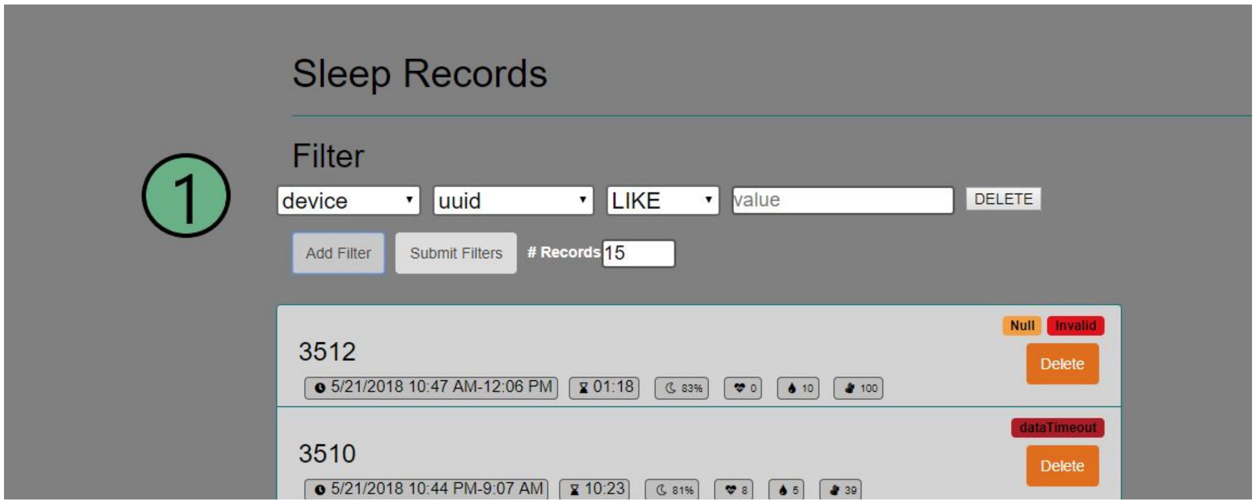
- 🕒 Start and End Times (Date shown is when report was uploaded)
- ⌛ Duration
- ☾ Quality Sleep Percent
- ♥ Heart Events
- 💧 Oxygen Events
- ✋ Motion Events

2 In the top right corners of the report boxes there are certain badges that will appear to help identify report properties quickly.

Invalid — The invalid badge is applied to any record which is considered too low of a time

Manual Null dataTimeout — Other badges indicate the Endreason for the record

Filter [Back to index]



1

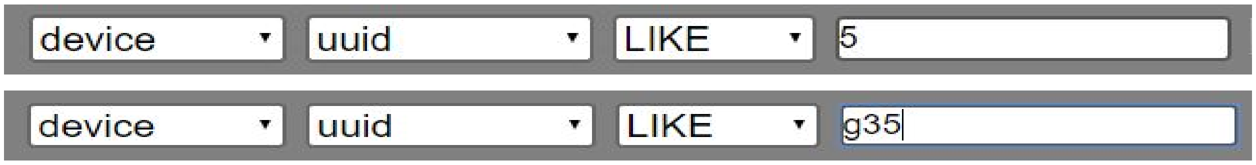
The filter form is divided into filters, the user can add and delete filters as needed. Each individual filter is separated into 4 fields, **Category, Variable, Operator, Input**. Additionally the **# Records** field shows how many total entries to view upon hitting the submit button.

The **Category** and **Variable** are related, in that they are the data type a user wants to isolate. The Variable is what is eventually tested, but Category can be selected to narrow down results.

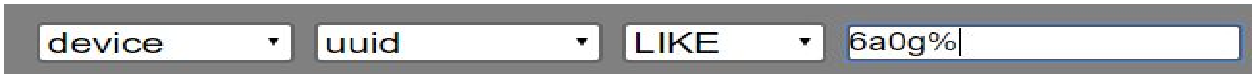
The **Operator** and **Input** field is what the user wants to test their data against.

Important to remember is the IS NULL operator should be used when trying to find null data, rather than '= null'.

The LIKE operator is a search operation that will find a String that contains the information sent. It is set by default to search based on the last character. For instance if the user wants to find a **UUID: 6a0ggw-52g35**,

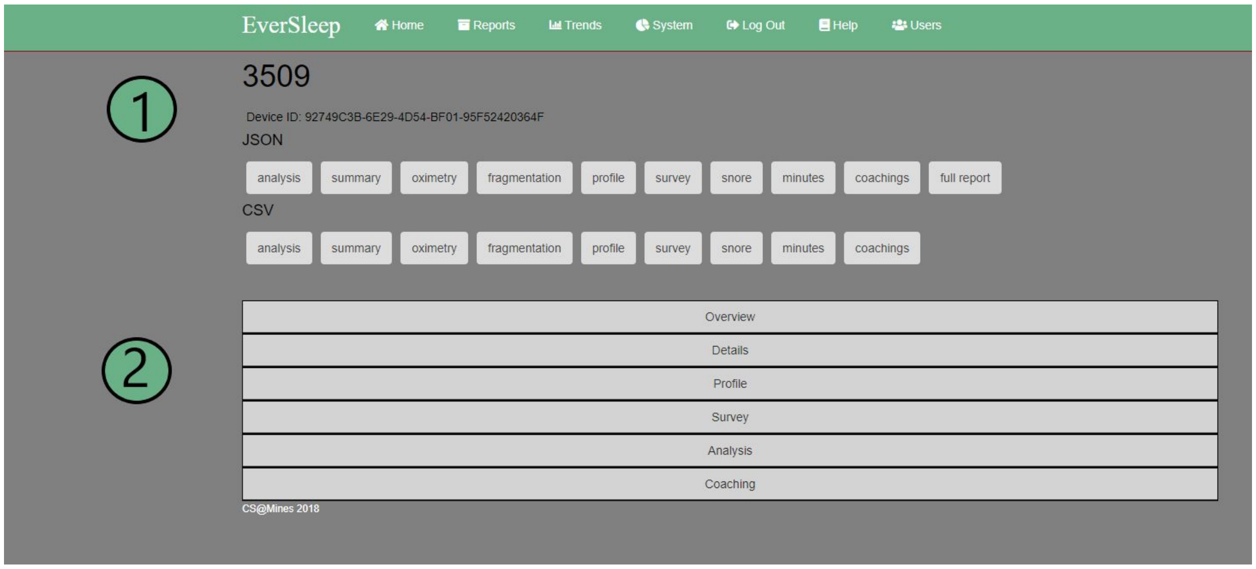


are both valid ways. The less specific however, may also include other users. It is also possible to add a wild card if necessary. As with the above example,



will return the same user.

Record Page [Back to index]


EverSleep

Home Reports Trends System Log Out Help Users

1

3509

Device ID: 92749C3B-6E29-4D54-BF01-95F52420364F

JSON

analysis summary oximetry fragmentation profile survey snore minutes coachings full report

CSV

analysis summary oximetry fragmentation profile survey snore minutes coachings

2

Overview

Details

Profile

Survey

Analysis

Coaching

CS@Mines 2018

The individual record page is a way to quickly view and download necessary files for a specific record

1

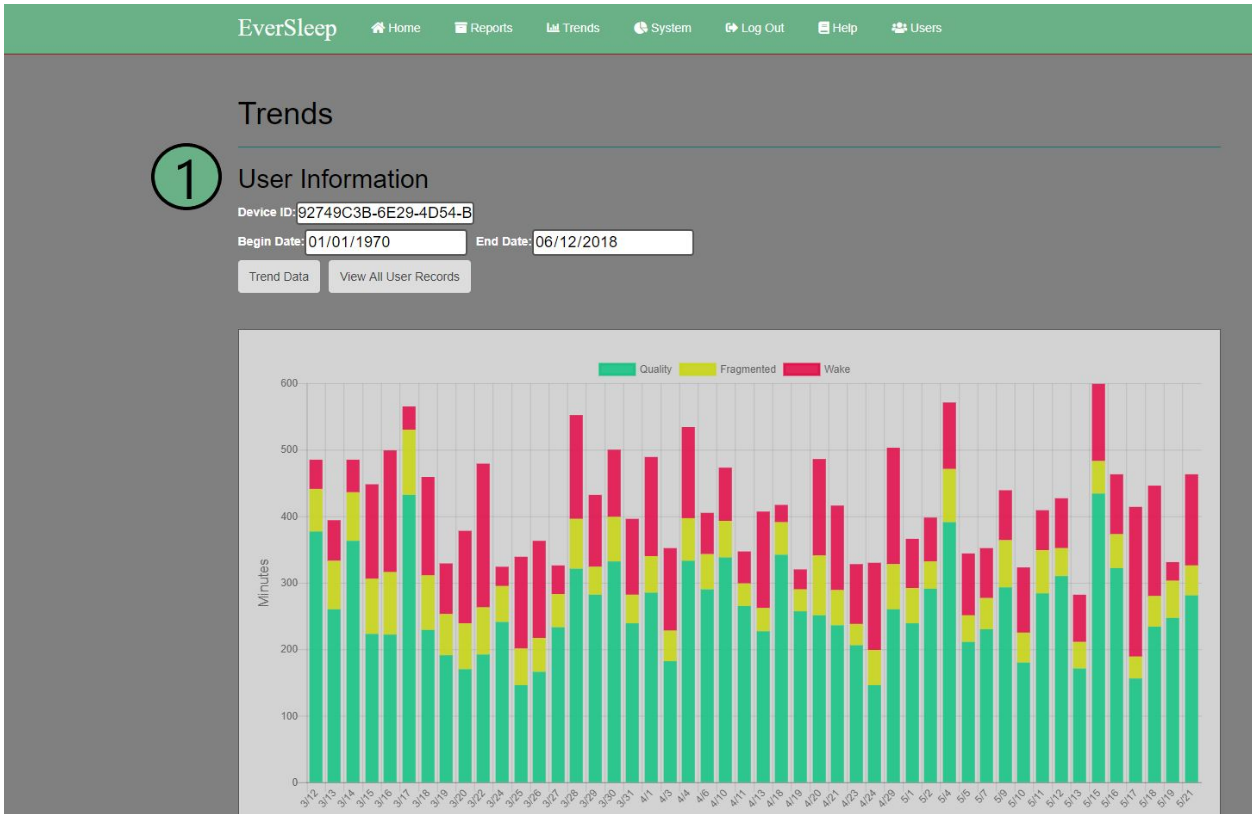
The Device ID is a link that will take the user to the trends page for that specific customer, allowing the user to view multiple records at once.

The JSON and CSV files allow for downloading files of those specific sections of data. For JSON files of large amounts of data (typically 12+ hours), it is best to save these files and view them in a editor rather than a browser.

2

Each section displays quick information of the record as well as charts that are similar to what the user will see

Trends [Back to index]



Trends

User Information

Device ID: 92749C3B-6E29-4D54-B

Begin Date: 01/01/1970 End Date: 06/12/2018

The trends page allows a user to gather all of a single customer's reports. It contains various charts to contain generalized data for the user over time, as the customer will see it in their app.

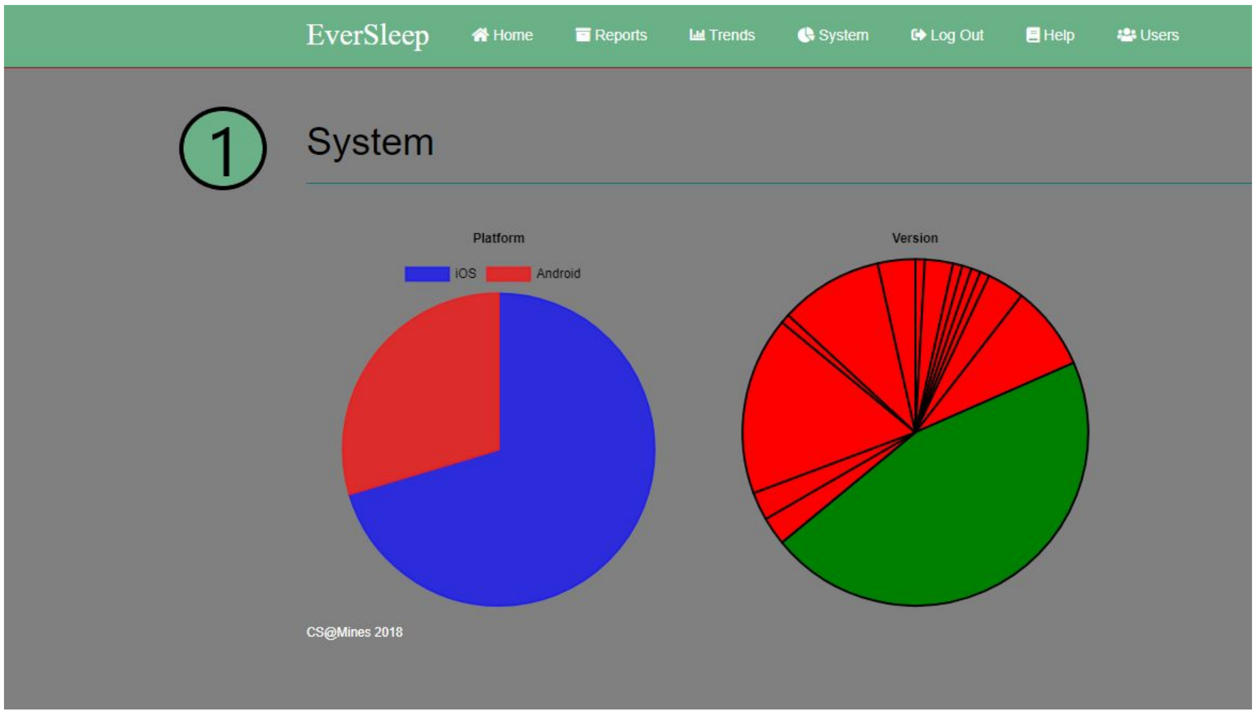
1

The filters in the trends section require a specific user UUID. It searches based on the LIKE operator (explained in the filters section). The default for date filters are from 1-1-1970 to the current date. When ready to filter out, click on the Trend Data to display the customers data.

After the data is displayed, it is possible to view all the individual records by clicking on

which will bring the user to the reports list page with the customer UUID entries.

System Back to index



1 The Systems Page purpose is to display the current Customer Platforms and Software Versions

Users Back to index

The users page shows all the users that are in the database. It also allows someone to delete Other users and create new users if wanted.