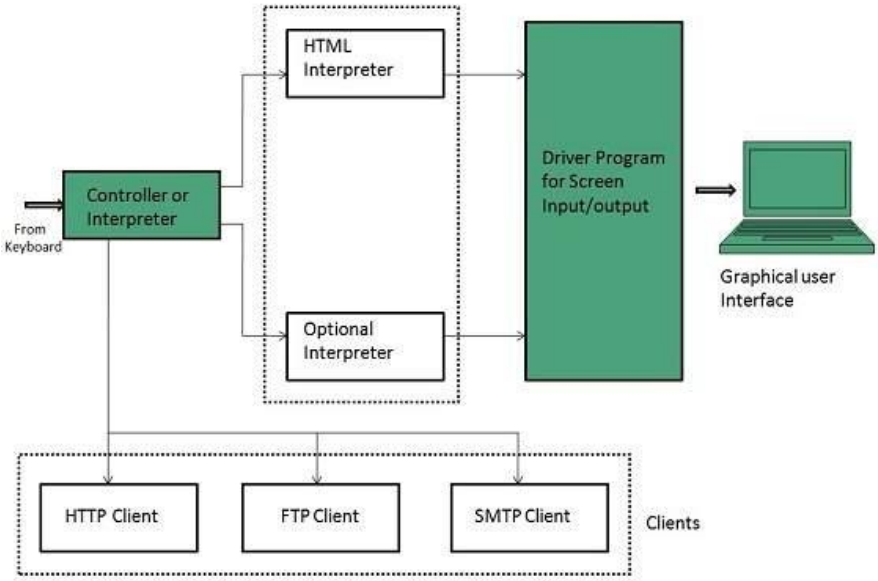
MEDLOG DESIGN DOCUMENT

Table of Contents

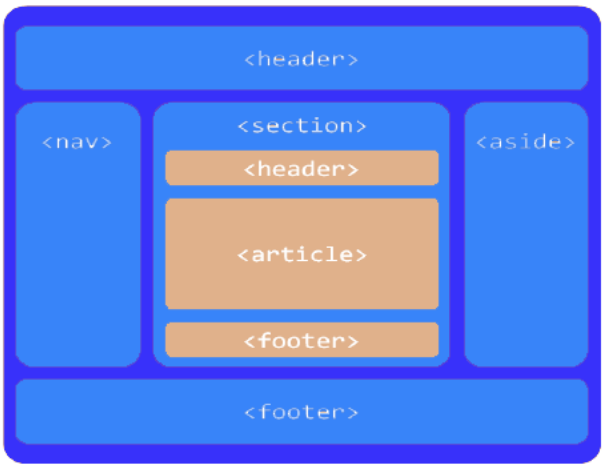
Front-end components:

The front-end incorporates the use of five primary programming language frameworks in developing each component of Medlog regarding the web application platform. HTML, CSS, Javascript, and JQuery are combined to structure client-side dynamics, presenting the patient end-user with the interactive features. HTML provides the standardized structured format for web pages and applications of all varieties in most rudimentary form. CSS provides standardized layout formatting for each element enclosed in tag based formats. Dynamics become impacted by incorporating Javascript framework. All are evaluated by a runtime interpreter.



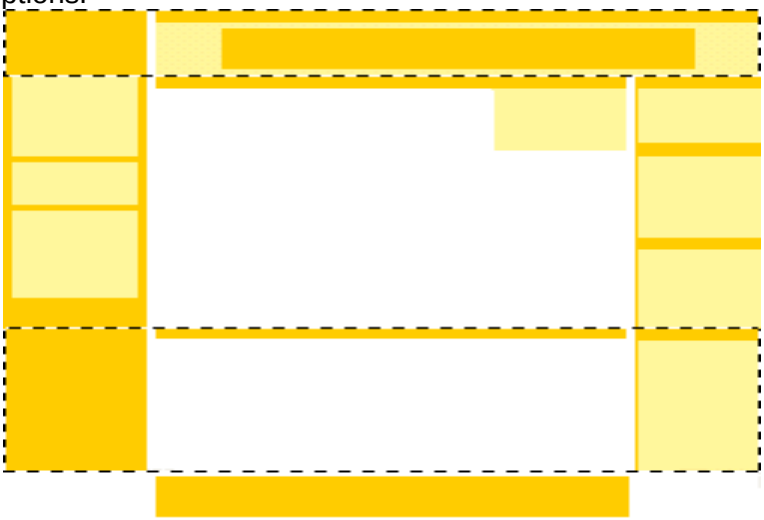
HTML mark-up

HTML mark-up encapsulates web service structure on the most fundamental level on the front-end, as previously mentioned. Use of tags to indicate formatting of text and a variety of objects is achieved through the use of block and inline elements regarding tag based format. Block elements will declare ownership of a certain portion of the page while inline are designed to be uniquely nested formatting within blocks. Generally, tagged elements can be strategically programmed to nest one another or remain separate, having an impact on the generic aesthetic display of the application. Blocks and inline elements are further encapsulated by pages which can potentially provide references to one another. This serves as the fundamental placement structure of each logging component as well as overall application features.



CSS styling

Modified formatting is generally intuitively handled by CSS styling page references for tweaking the interface-able layout. While it is possible to add customized modifications within element based tagging, best practice is understood by referencing unique ID and class selectors for modifying page elements in a separate CSS file. IDs are unique to one element while a class is unique to a group of elements, both of which can be strategically intertwined. CSS files, along with the permitting of scripting variation is enabled through the head tag, which acts as the web service runtime interpreter's preprocessor. It is also best practice to make reference of customized scripting files similarly to using proper CSS techniques, minus strategic programming exceptions.



Javascript

Javascript programming comprises the remainder of the front-end design. Dynamic page formatting is the role Javascript plays, which will allow fluid client-side interaction for each of the Medlog components. Javascript is a dynamically typed object oriented language, and as such, has loosely defined variables broadly associating to object types.

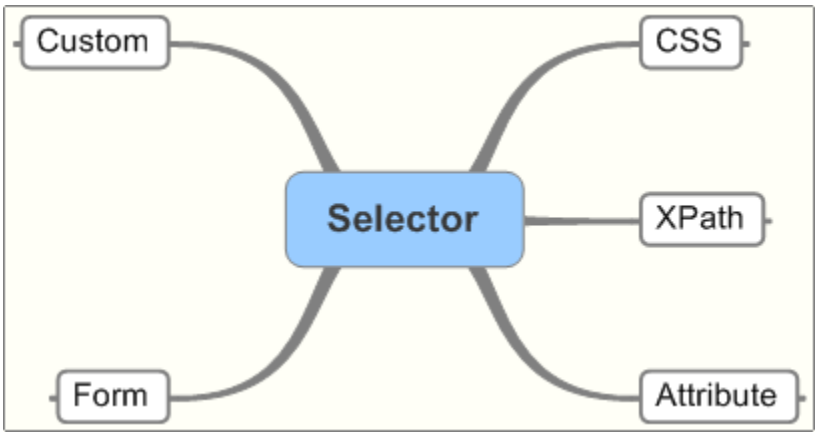
Every instance of a variable, function, or class object, is associated as type var. There is loose interchangeability in recording even arrays of type var. Structured use of this broad object

defined typing allows the ability to manipulate such dynamic behavior of pages. Javascript also has the capability to manipulate the DOM as well as BOM elements of each page. Through manipulating the DOM, Javascript has power over page structure. BOM manipulation allows Javascript to interact with the browser more directly rather than the displayable document contained within. While JSON is included in the Javascript framework, elaboration of such mark-up is covered via back-end. More discussion of algorithmic implementation through Javascript will be covered in the “WIREFRAMES AND ALGORITHMS” section to elaborate understanding of managing each component of Medlog.

JQuery

JQuery simplifies common tasks implemented by multiple lines of Javascript code. The idea behind the use of JQuery is to opt for modularity over redundancy regarding Javascript’s OOP framework as a whole, mainly geared towards AJAX calls and DOM manipulation. Use of JQuery is also extended to CSS manipulation, HTML event methods, effects and animations, and utilities. The most important programmable components of JQuery are selectors and events.

Selectors enable manipulation of HTML elements. These selectors are used to find tagged elements of particular id or class specification, which also happen to be built on top of CSS selectors. These selectors could also be used to traverse through other selectors that are referenced to tags according to ancestor and descendant relations as well as filtering. General use of JQuery is based on event response handling. All client-side user interactions that a web page can respond to are considered events.



JQuery also enables special effects to be incorporated into pages, also known as actions. To eliminate the possibility of errors occurring with effects executing one after another is by using callback functions, which are executed after an effect has been finished. Chaining can also take place within the use of effects, which allows an action to be appended to a previous action.

Back-end components:

Server-
Windows 2008 Server.

Web Server-
Apache Tomcat (v8.x)

SQL Server-

SQL Server 2008R2

MedLog's core features are mapped to database components. A component, in the database, consists of multiple entities (tables). The basic layout of a component, pictured below, consists of:

1. The component's primary entity
2. The join table that links the component to the patient.
    a. This table also contains the log entry specific data, such as date and any incidental information.
3. The join table linking between keywords (tags) and log entries.
4. Supplemental tables
    a. Diary:
        i. User defined field - such as a rating.
    b. Medication
        i. Type - (OTC, Rx, Supplement)
        ii. Sig - Dosage and directions (ex: 200mg, take one pill three times per day).
        iii. FDA Coding - Most OTC/Supplement and all Rx are searchable in the FDA database. Information about interactions and contraindications can be searched using a combination of ids.
        iv. Join for Rx between medication and prescriber.
    c. Healthcare Provider
        i. Joins between:
            1. Patient-medication and Provider
            2. (Inferentially) between patient and provider (for purposes of sending diary)
        ii. Contact history - A log of the diary/log sent to provider by patient.
5. Non-component features of database include
    a. Logging -
        i. System events (such as errors)
        ii. Mobile Sync (trusted devices [table] have permission to sync). When a patient creates offline log entries, those entries may synchronize with the patient's MedLog account when the device re-connects to the internet.

Each component has two (2) many-to-many relationships. Between itself and the Patient, and between itself and a Tag.

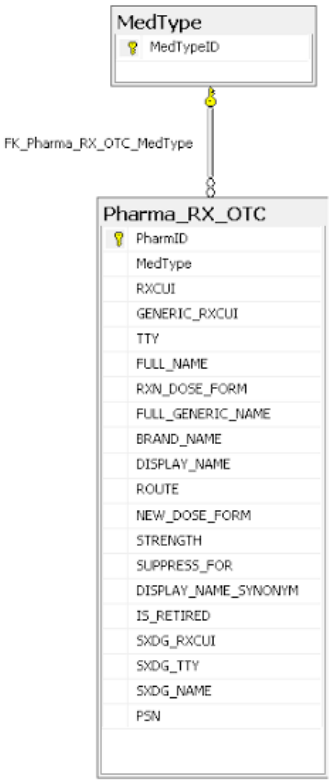
The Patient mapping facilities reuse of component types across users of MedLog.

For example: A user may take multiple medications, and those medications may also be prescribed to many other users of MedLog.

The Tagging system is a feature that enhances search and cross-component relationships. All types of log entries (e.g. diary, medication, activity etc..) can be linked to keyword tags.

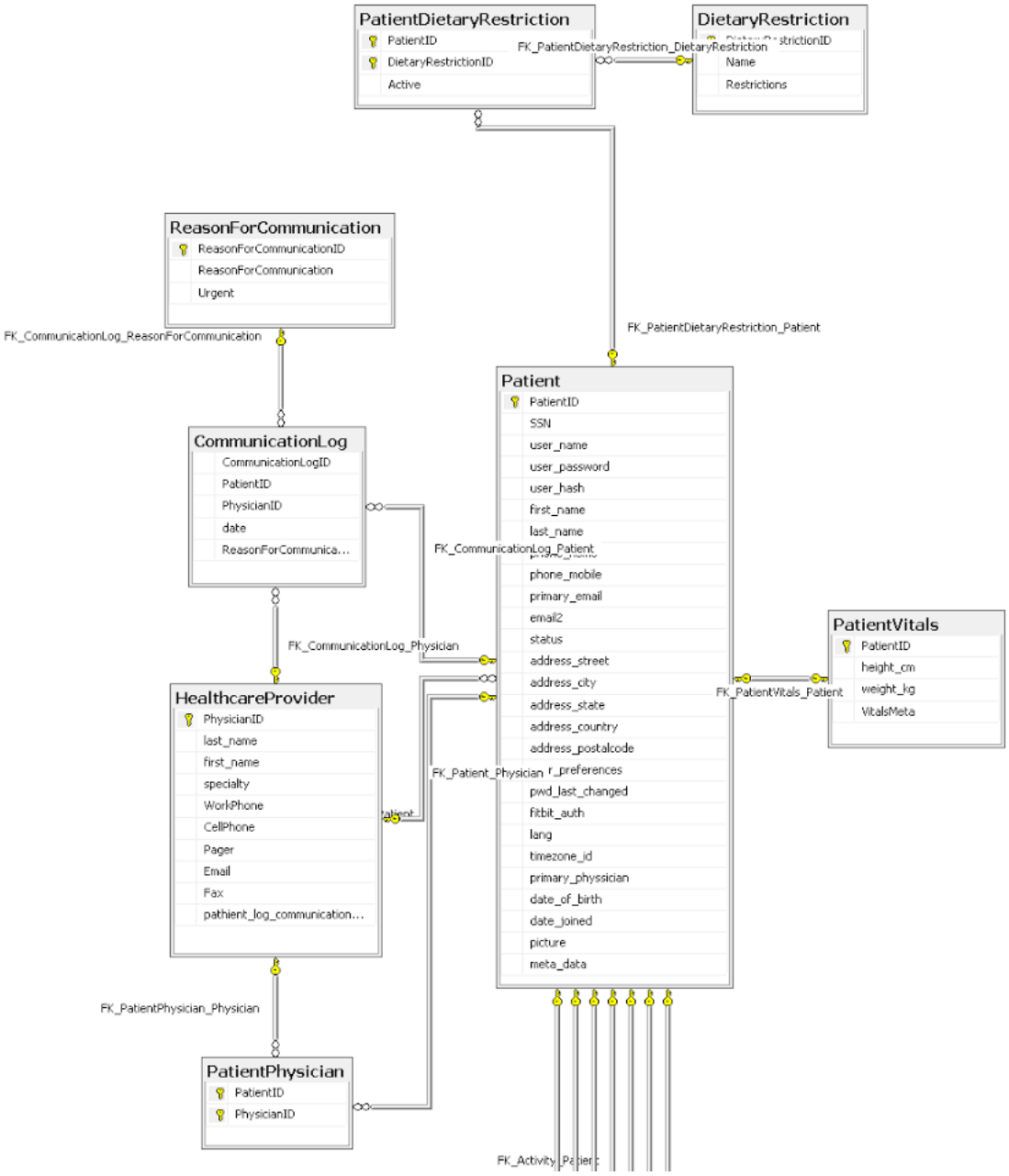
For example: A patient creates a keyword "drowsy." When the patient takes/logs medication -and/or- types a diary entry, they apply the tag drowsy. The search page includes keyword list. Clicking on drowsy, all tagged entries.

The following diagrams are either denoted as components or entities. Components follow the aforementioned pattern. Entities are described above, either as supplemental tables or sub-components.



Medication

[FDA/GoodRx API sub-component]

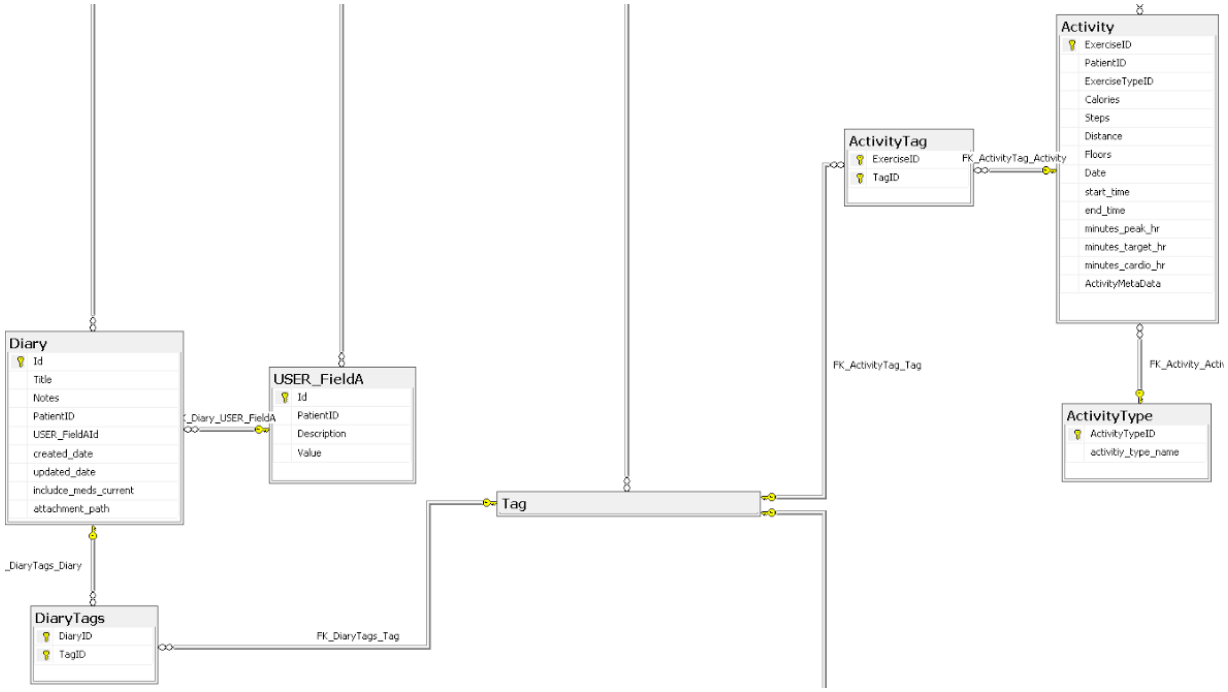


Healthcare Provider

[component] **Patient (user) [entity]**

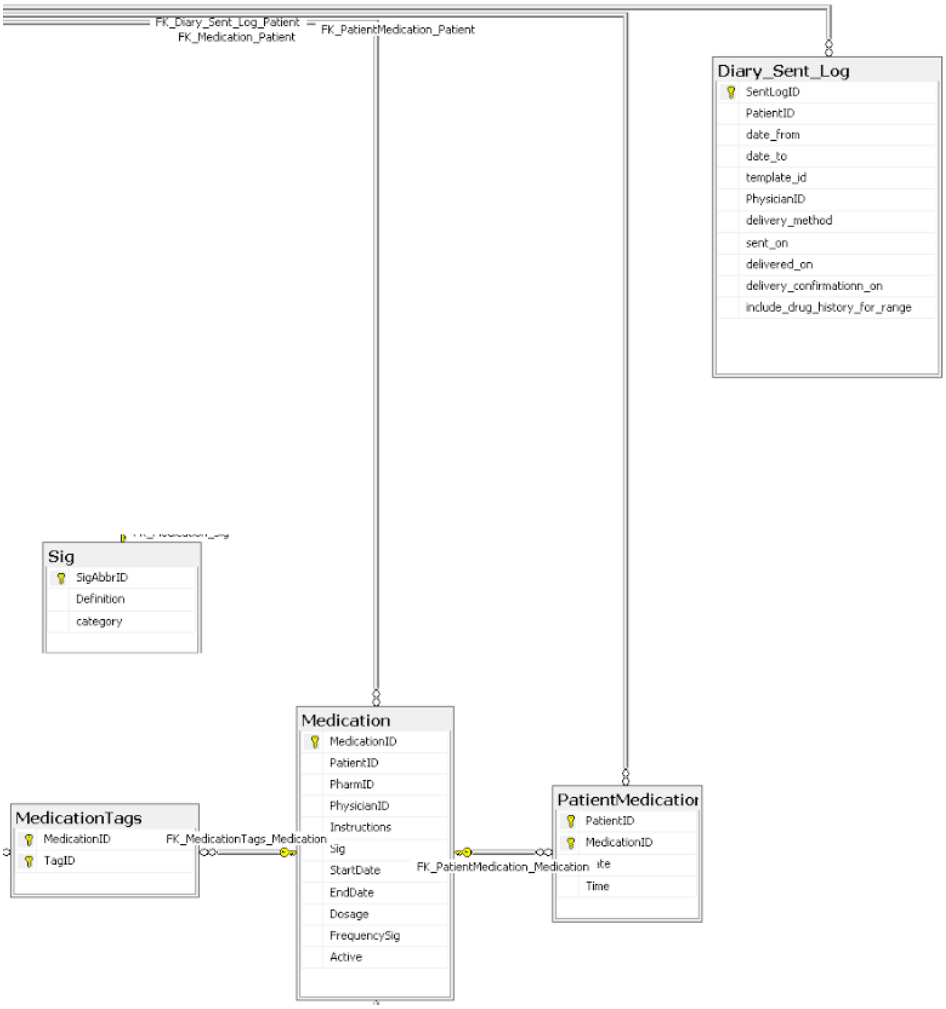
Mobile sync

[entities]

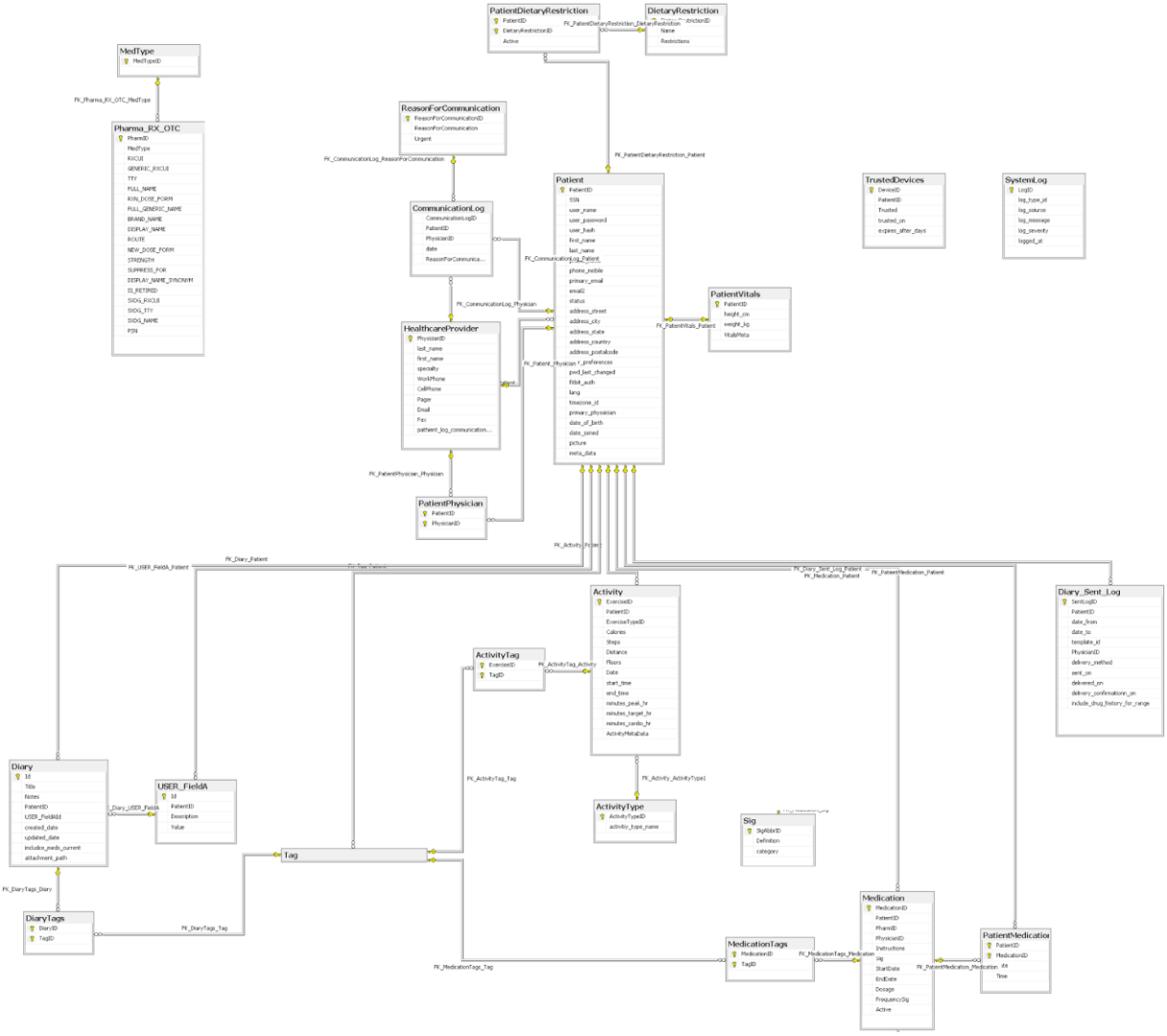


Dietary restrictions

[component] **Activity [component]**



Medication(CONT)



Big Picture ER Diagram

Android OS:

Use of android framework primarily incorporates the use of Java OOP and XML markup. XML markup represents the UI of the functioning application while Java code will program the UI with specific behaviors. In the case of MedLog, these behaviors are limited to initializing a journal entry, making a journal report, and forwarding the journal report information. Gradle compiler mitigates binding to the Java and XML, under the hood of the Dalvik Debug Monitor System.

Java OOP framework is focused around the use of activities components and incorporation of intents to shift from different user navigated tasks (i.e. logging in, journal entry, journal report, report forwarding. Use of event listeners are incorporated for widget use (interactive mechanisms such as buttons and text fields) where the user enters credentials for login, submits an entry, views report info, or forwards report info. MainActivity, is where the app launches and executes its main thread to initiate login credentials, where user can make synchronizable journal entry, view report information, or forward report information to a specified healthcare provider. A variety of other activities are incorporated for submitting an entry, viewing report information, and tracking the user preferences.

Android incorporates binding between the XML and Java code. XML views and ViewGroups provide resource identifiers accessible to the Java binding. This binding is handle for user interaction mechanism widgets incorporated in their specified ViewGroups pertaining to the XML layout for a particular activity. These views within their respective ViewGroups are referenced in the R.java resource file, where they are accessible to the activities that use them associated with layouts, widgets, and interactable display fields of information previously mentioned for event handling via use of listeners.

SQLlite and Content Providers are utilized to synchronize with the Microsoft SQL Server database. MedLog incorporates the use of mobile synch feature in case a patient is out of service temporarily, where once they are in an area service, their entry will synchronize with the database, keeping all existing entries up to date. VO files are also stored for journal entries, patient information, application state information, and the offline synch capability. A util directory is also incorporated for testing offline synch and archiving entries stored that are not yet synchronized.

3RD PARTY APIs:

openFDA

While openFDA has a number of uses, Medlog's main focus is geared towards the use of the drug info API which can identify interaction, reactions, as as well as background information on

a number of prescription and OTC (Over-The-Counter) medications. This API can also be used for querying food related information for the sake of the dietary restriction logger.

- Adverse events layer: Reports of drug side effects, product use errors, product quality problems, and therapeutic failures.
- Drug product labelling layer: Structured product information, including prescribing information, for approved drug products.

openFDA is an elasticsearch API, with a querying focus on drugs, devices, and foods. These categorizations serve as endpoints, which are then utilized in our case for the adverse events as well as drug product labelling layer of medication relative archiving. Intuitive JSON compatibility is supplied with the API, with both metadata, of type meta, matching results querying, of type results. Metadata includes a disclaimer, link to data license, last-updated date, and total matching records, if applicable. Results returns array of matching results, dependent on which endpoint was queried.

Querying involves a number of useful function calls for the purposes of Medlog's medication/supplementation logger, focusing its usefulness towards prescription and OTC medications.

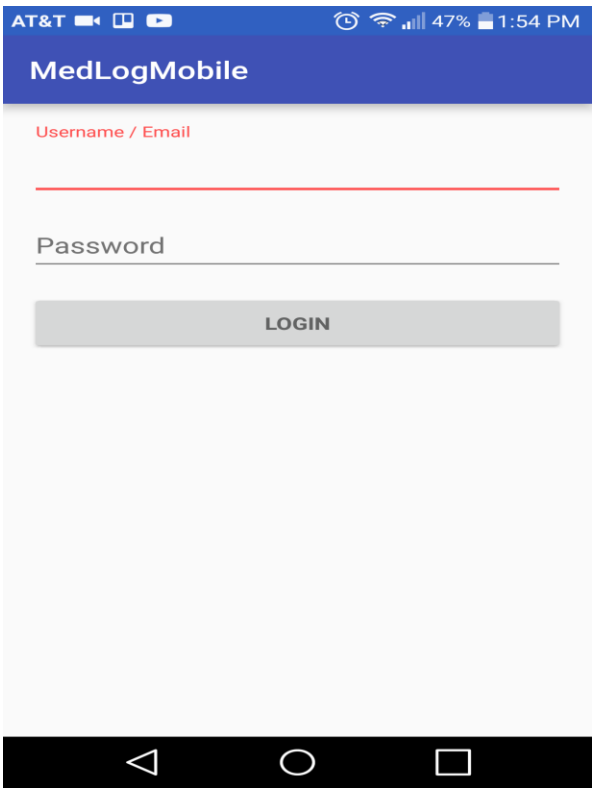
- Meta functions (number type)
  - meta.results.skip - Offset (page) of results, defined by the "skip" query parameter.
  - meta.results.limit - Number of records in this return, defined by the "limit" query parameter. If there is no limit parameter, the API returns one result.
  - meta.results.total - Total number of records matching the search criteria.
- Query parameters
  - search: What to search for, in which fields. If you don't specify a field to search, the API will search in every field.
  - count: Count the number of unique values of a certain field, for all the records that matched the "search" parameter. By default, the API returns the 1000 most frequent values.
  - limit: Return up to this number of records that match the "search" parameter. Large numbers (above 100) could take a very long time, or crash your browser.
  - skip: Skip this number of records that match the "search" parameter, then return the matching records that follow. Use in combination with "limit" to paginate results.

Look and Feel

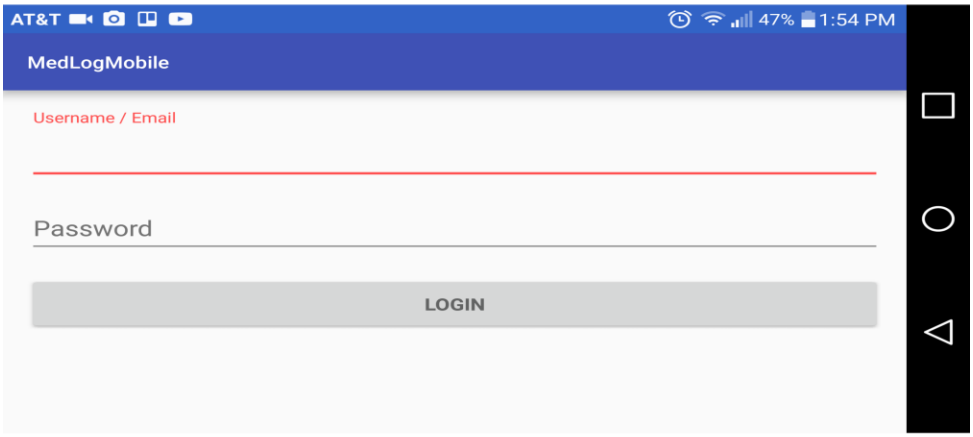
Login/Register

When the user launches MedLog, they are first directed to the login component either on web browser or android device. User is prompt to verify their credentials, but in the android mobile version, cannot register. Credentials are obviously required in order to access the core features of logging a journal entry, accessing their mood and productivity status report based on the average scaling of entries, medication/supplementation logging, and keeping tabs on healthcare provider contact information with the ability to forward their report via mobile device. Registration requires username and password credential input with a variety of personal profile information fields such as first name, last name, date of birth, location information such as state/city/address/zipcode/country of origin, and means of contact such as phone number(s) and email address.

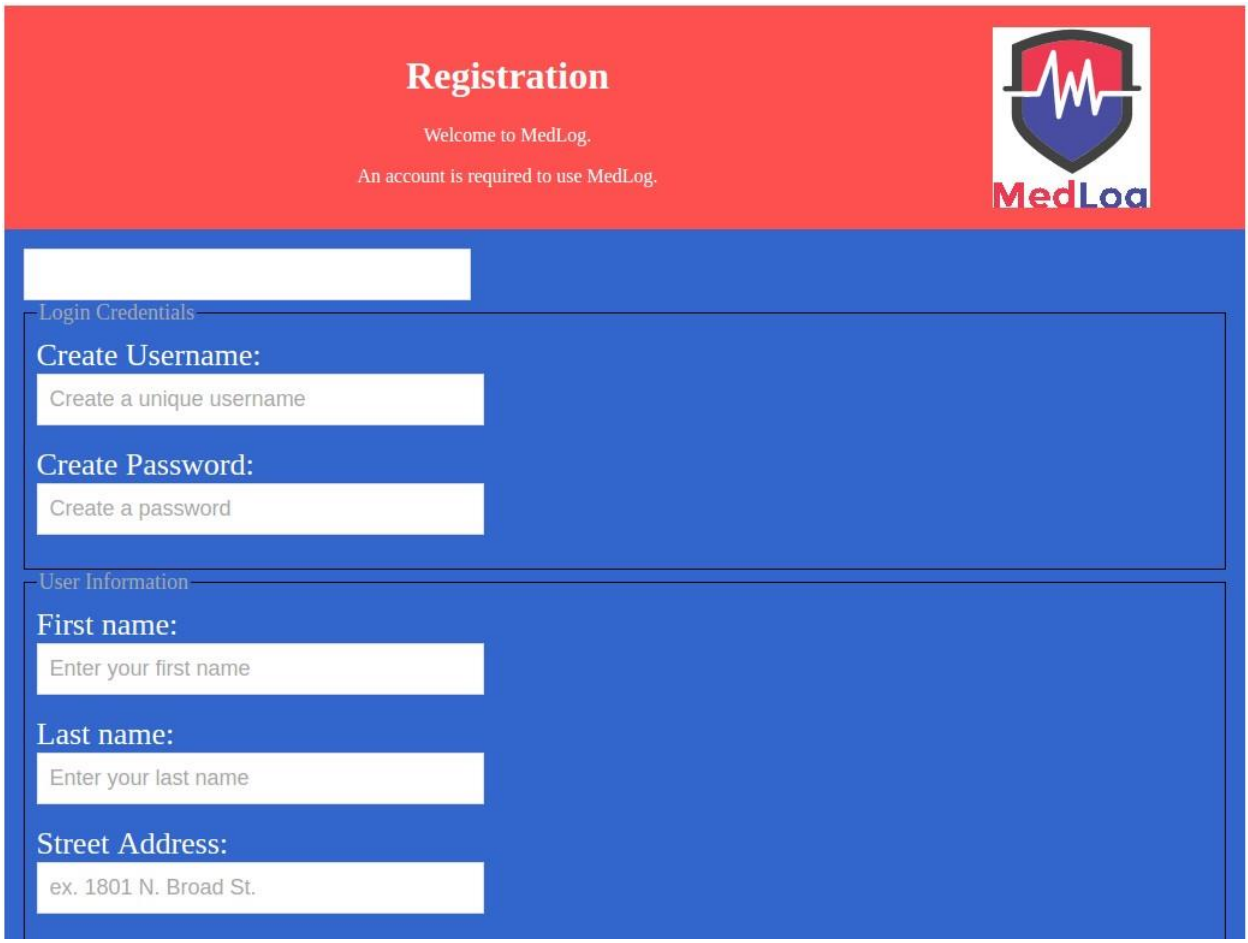
MedLog login page via web



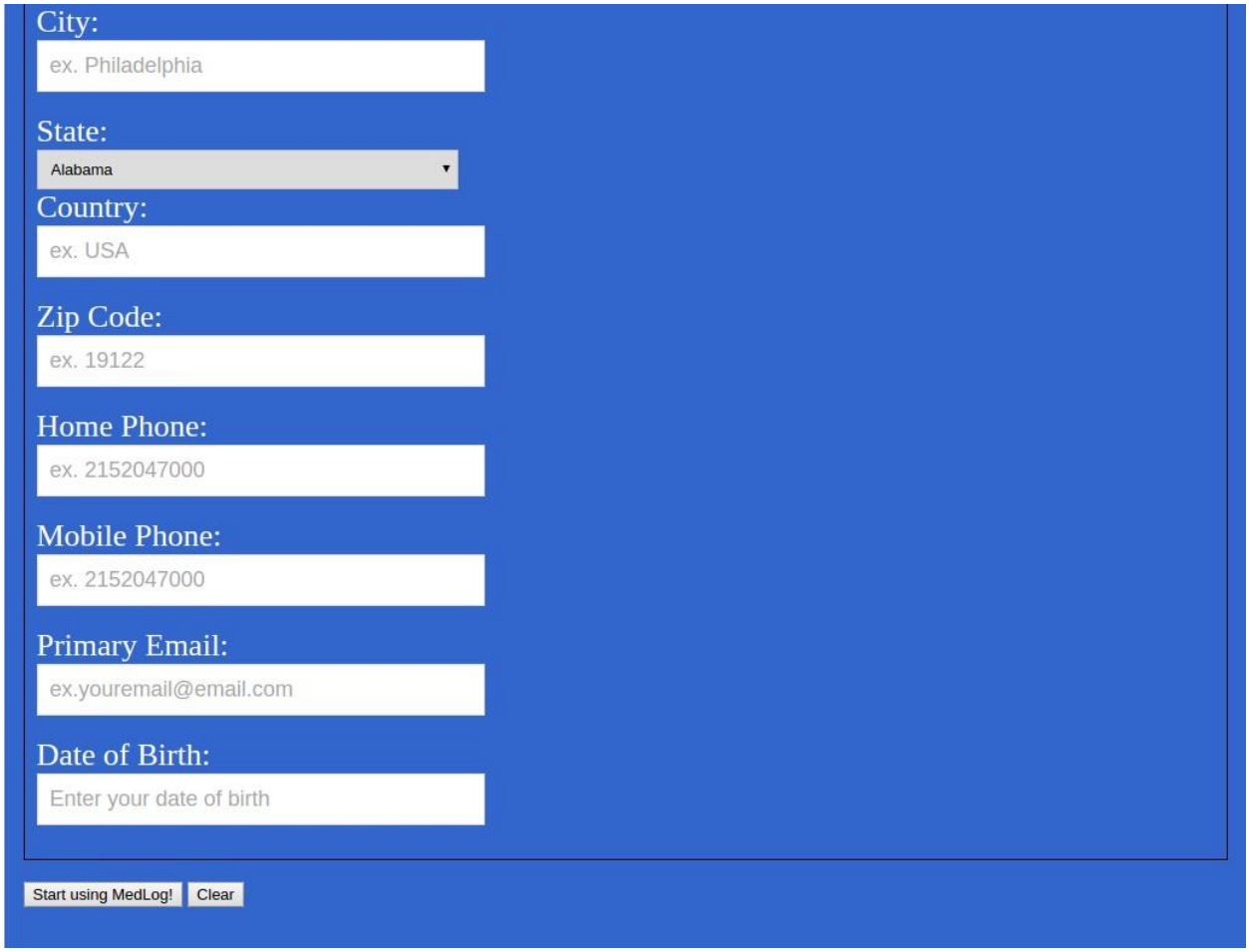
Medlog login portrait mode via mobile



Medlog login landscape mode via mobile



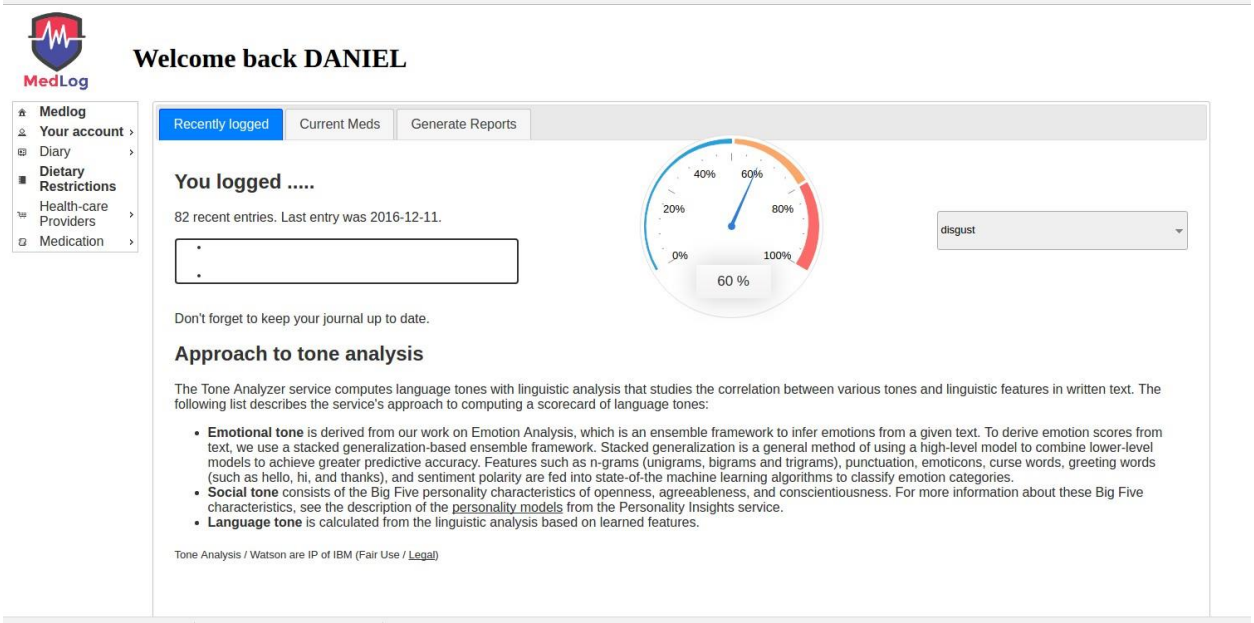
First half of registration page via web



Second half of registration page via web

Dashboard

Medlog dashboard provides the initial UI to access all functional logging components via web, as well as viewing individual weighted tone variable report information (more on this in Tone Report section). The user can navigate to the current meds tab, journal tab, medication tab, view list of medications, generate tone report analysis, and the healthcare providers tab.

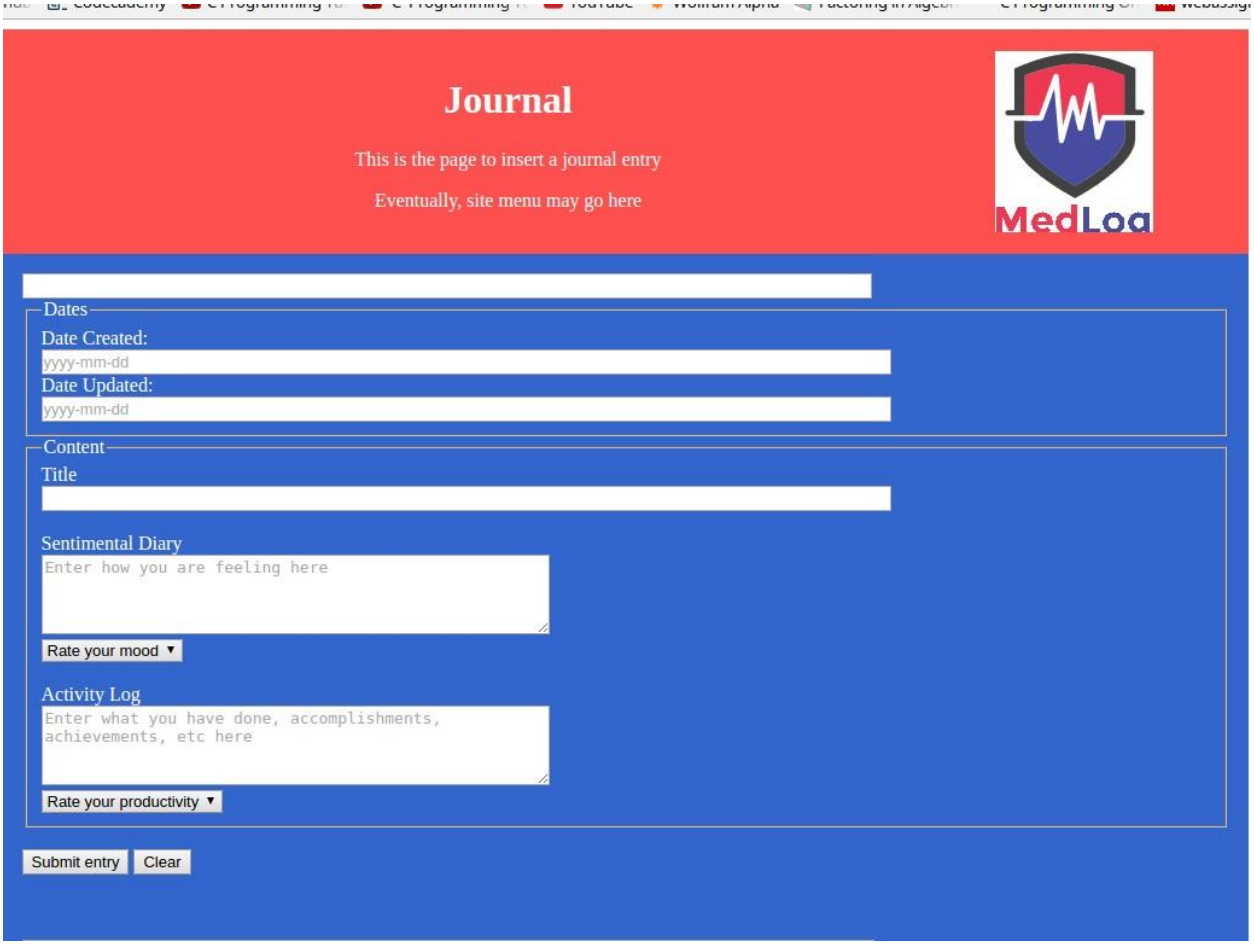


Medlog dashboard redirect upon logging in via web

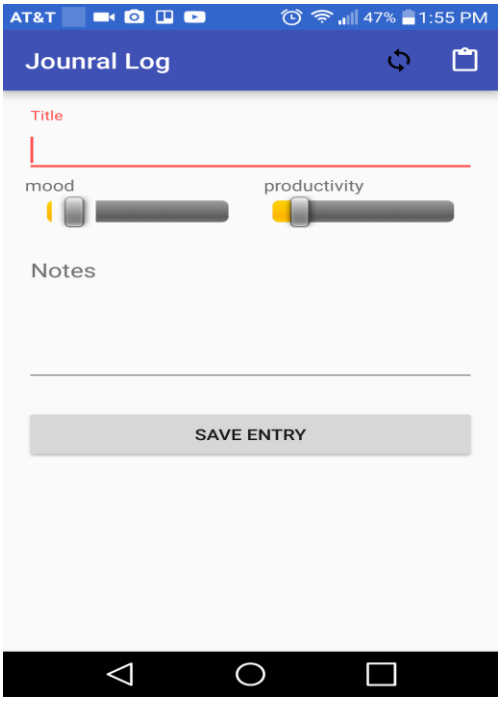
Personal Journal

Through use of the Personal Journal, a user can generally make bi-entry logs based on “how they are feeling” via the Sentimental Diary sub-component based on “what they have accomplished” via the Productivity Logger sub-component, as well as the entry date. Rating their mood and productivity comes respectively with both bi-entry log features important for the journal report portion. Ratings are scaled between 1 to 10 (1 being the worst and 10 being the best). Android platform only allows users to make a Sentimental Diary entry, but still incorporates the mood and productivity rating mechanics. Android also has the functionality of forwarding the report summary. Android platform is also limited in the sense it is unable to fetch the user journal entry list or revise previous entries, both of which the web platform variant is capable of.

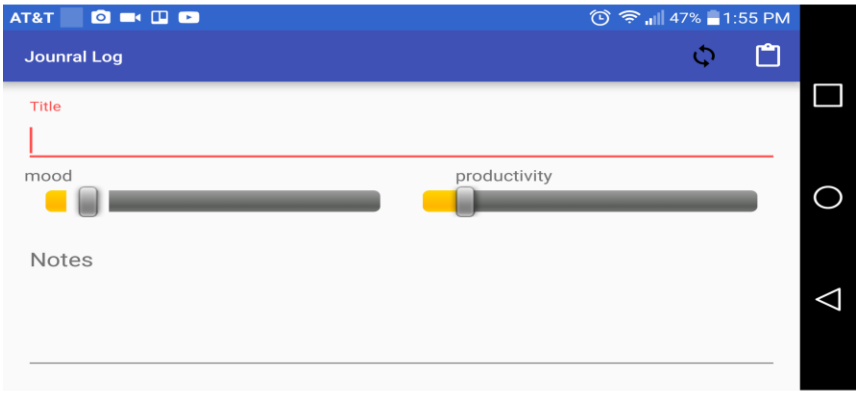
Mood and productivity ratings are archived for purposes of scaling mechanisms for reporting information to draw towards a potential physiological and behavioral analysis based on both bi-entries. This analysis is limited in the sense it can only measure mood and productivity based on journal entries to be integrated into the journal report. We will understand that the phrasing context of entries provides importance for the later emphasized Tone Report algorithm section to predict the mood of the user for their next journal entry. Both mood and productivity data are displayed on a line graph based on the trend of mood to productivity per bi-entry, each with their different depictions of comparative behavior visually depicted together.



Insert entry page via web dashboard



Medlog Journal



WebApp Medlog journal entry landscape mode via mobile

List Journals

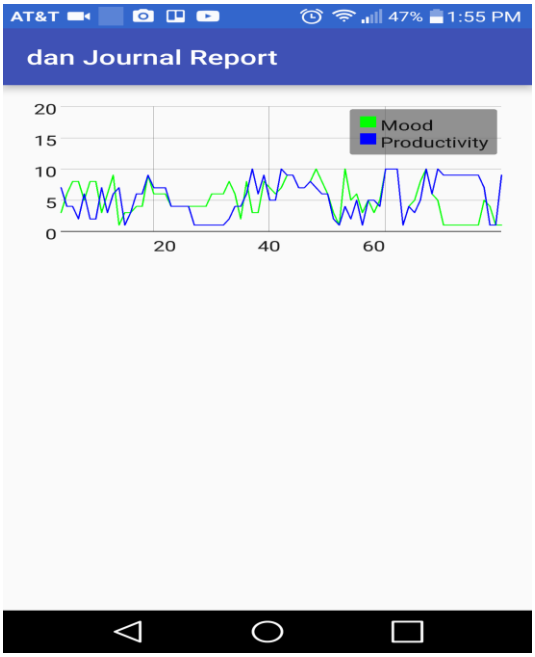
Edit	Journal ID	Date Created	Title	Mood Rating	Productivity Rating	Date Updated
	2	10/25/2016	test	8	8	
	3	10/26/2016	AAAAtest	2	4	
	4	10/26/2016	AAAAtest	3	4	
	5	10/26/2016	AAAAtest	5	5	
	6	10/26/2016	asdfsdfdfdsf	3	7	
	7	10/31/2016	nonoYAAAASSSSS	6	4	01/01/2016
	10	11/01/2016	NONONONON	8	4	
	11	11/02/2016	NONONONON	8	2	
	12	11/03/2016	NONONONON	5	6	
	13	11/05/2016	NONONONON	8	2	
	14	11/05/2016	NONONONON	8	2	
	15	11/06/2016	GTree demo	3	7	
	16	11/06/2016	fdxghk,lkkkk	6	3	
	17	11/07/2016	Title	9	6	
	18	11/07/2016	Stressed about project demo	1	7	
	734	11/23/2016	Some Test Title	3	1	
	735	11/23/2016	Testing title	3	3	
	736	11/23/2016	kkkdjdjjeidj	4	6	
	737	11/23/2016	kkkdjdjjeidj	4	6	
	738	11/23/2016	DLL t truck	9	9	
	739	11/24/2016	yryehdhdhc	6	7	
	740	11/24/2016	yryehdhdhc	6	7	
	741	11/24/2016	yryehdhdhc	6	7	

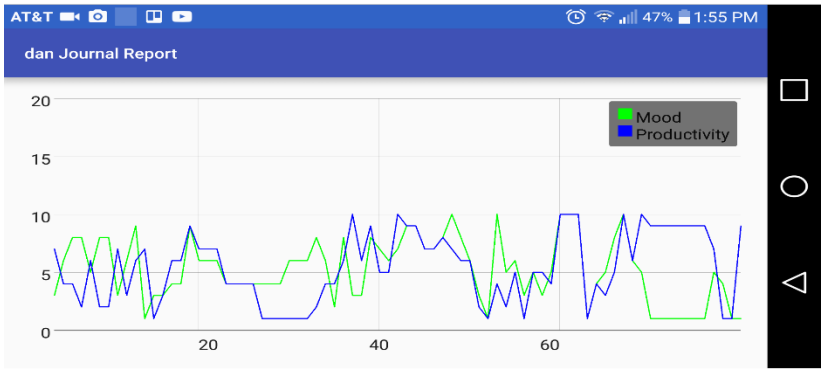
List entries page via web dashboard

Revise entry page via dashboard accessible from Journal List page

Journal Report page via dashboard

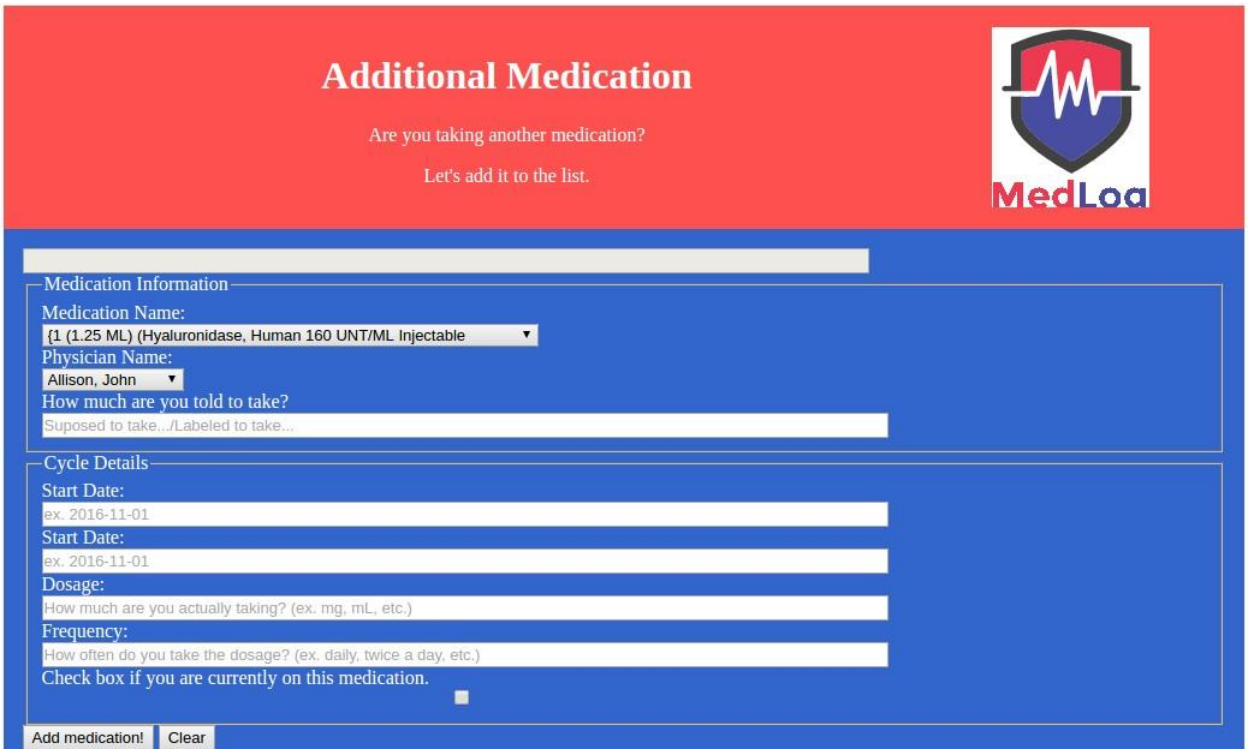
Journal report portrait mode via mobile





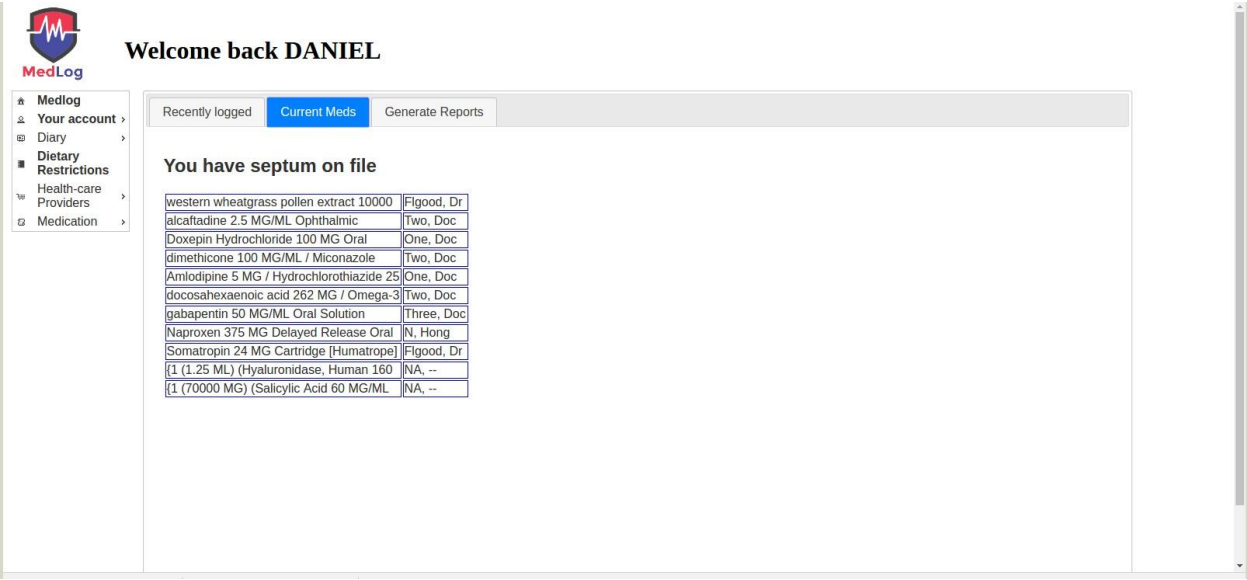
Medication/Supplementation

User is able to log the prescribed medications, supplements, and over-the-counter (OTC) medications as logged. Prescribed medications require a healthcare provider to be tagged in the entry, whereas supplements and OTC meds do not require this extra field, which is just a reference to an existing healthcare provider logged in the healthcare provider contacts log. Additional information includes dosage “How much are you supposed to take”, cycle details inclusive to start and end date, frequency, and whether the user has clarified via checkbox that they still take the medication/supplement/OTC. A user can fetch a medication name from the database archives, generally consisting of openFDA entries, which can be modified with additional entries to be loaded in future additions. If a medication does not exist in the archived database, then the user can add the name of the medication to the logging info, which will then be archived accordingly upon entry. A user can also fetch their listed entries of medications they currently take.



Medication page for adding existing archived medication to the medication log

Medication page for adding a non-existing archived medication to the medication log

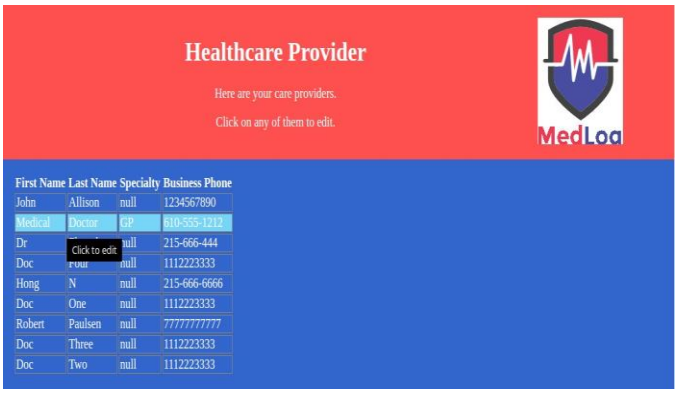


Medication List via dashboard

Healthcare Provider

User's are able to manage their healthcare provider contacts with use of the Healthcare Provider information logger. Functionality includes adding a contact, listing existing contacts, and revising a specific contact. When adding or revising a contact, the information of concern is first name, last name, specialty, location (address/city/state/zipcode), phone number(s) (business/mobile/fax), and email address. Each existing healthcare provider can have journal report information forwarded to them upon request of the user via mobile platform.

Healthcare Provider add page



via dashboard

APPENDIX A
Insight/Repotting BI

Mood Predictor Algorithm

Abstract:

MedLog keeps a record of all journal entries that has a user has reported. Included in the journal entries are mood and productivity ratings as well as entry in text on how a user is feeling. We want to take it a step further by utilizing the tone analyzer with the mood ratings.

Introduction:

The Watson Tone Analyzer provides three components: emotion, language style, and social tendencies. Each category have sub-categories:

- Emotion
  - Anger
  - Disgust
  - Fear
  - Joy
  - Sadness
- Language-style
  - Analytical
  - Confident
  - Tentative
- Social tendencies
  - Openness
  - Conscientiousness
  - Extraversion
  - Agreeableness
  - Emotional range**

Given a body of text, the Tone analyzer will provide a value between 0 and 1 as to how likely each category is present. This feature will be combined with the mood ratings feature. Mood rating is from 1 as the worst to 10 as the best. Each sentence has their own evaluation. Each journal entry would also get its own score. We will then utilize these two components to predict mood rating based for current journal entry based on previous journal entries.

Analysis:

There were several approaches to how the mood can be determined. First and foremost, we want to utilize the history of diary entries. The quantity as well as how descriptive the entries would lead to a more definite prediction of mood. Secondly, we want to rank the categories and values provided by Watson API. This is determine how much each category contributes to the mood prediction. All categories must be utilized.

Step 1: Find regression values, R^2, for all categories.

First, we find r by plotting mood versus each category value provided by Watson.

Then, we ranked the values using r^2, by dividing r^2 by the sum of r^2. This is the adjusted r^2. This is to give us a ranking system based on the best fit line correlation compared to the other correlations. This accounts for how far a value may be from the normalized distribution. We prefer not to have drastic changes to collected entries when a new entry is appended and analyzed.

Step 2: Normal Distribution

With the ranked weight, we normalize the distribution using StatUtils.normalize(). This provides for us the standard error percentage as well as the variance. With the ranked weight normalize, we can obtained the inner quartile range, eliminating the first and last quartile as outliers.

Step 3: PMF and CDF

For each new journal entry, we have a new analysis provided by Watson. That means a different ranked weight. We utilize each particular entry as its own PMF (probability mass function) while the overall history as CDF (common density function).

Step 4: Predicting mood

Current entry values are multiplied by their respective weight, provided by CDF. This provides the CDF contribution of current/new entry. Then the sum of contribution is taken, which is then the value of mood.

Development History:

First Approach:

1. Obtain regression values by plotting all mood values vs each category. Higher regression coefficient suggest higher correlation.
2. Reanalyzed mood values with correlated emotion category:
   Use an integer array with 10 slots (maybe smaller) and increment each slot when emotion category is high.
   Example: Anger as the highest correlated
   Every time Anger value > .5, check mood. Mood value increments in array.
   Say for a week, we have mood values:
   [5 7 9 1 5 6 5]
   For Tone{Ange}r value:
   [.5 .75 .75 0 .75 .5 .5]
   So Array may look like:
   [0, 0, 0, 0, 3, 1, 1, 0, 1, 0]
   Find the highest occurrence value, which is 5 in this case. So mood rating of 5 is most correlated with Anger

Equation:

$$Predicted\ Mood = \frac{correlation\ coefficient}{k} \sum_{i=1}^{k} mood\ value * occurrance$$

k as the number of unique mood value, 1 ≤ k ≤ 10

In the case of example, the correlation coefficient is .75. Then

$$Predicted\ Mood = \frac{.75}{4} * 38 = 7.125 \approx 7$$

Where

$$\sum_{i=1}^{k} mood\ value * occurrance = (5 * 3) + (6 * 1) + (7 * 1) + (9 * 1)$$

$$= 15 + 6 + 8 + 9 = 38$$

This equation is simply taking the average.
However, if we keep the lowest correlated, and note when it is high, we can subtract one (=6) and add when it is low (=8).

Second Approach (current):

1. Watson API will analyze text and provide statistics on each category. Each category will contribute a certain percentage of the predicted mood value.
2. From there, ranked each category in descending order, so that the higher the rank, the bigger the contribution it has to predicted mood. This is detailed in Analysis section.
3. With rank as a factor, we will also need to factor in how old an entry is. The older the entry, the less the contribution. Thus, 75% would be based on recent entries, and 25% will be based on past.

Equation:

$$Predicted\ Mood = 30 * \sum_{i=1}^{13} \frac{rank\ weight_{recent}}{rank\ value_{recent}} + 10 \sum_{i=1}^{13} \frac{rank\ weight_{past}}{rank\ value_{past}}$$

Initial Findings:

Presented below, is a primitive application method on a small sample of 35 journal entries.

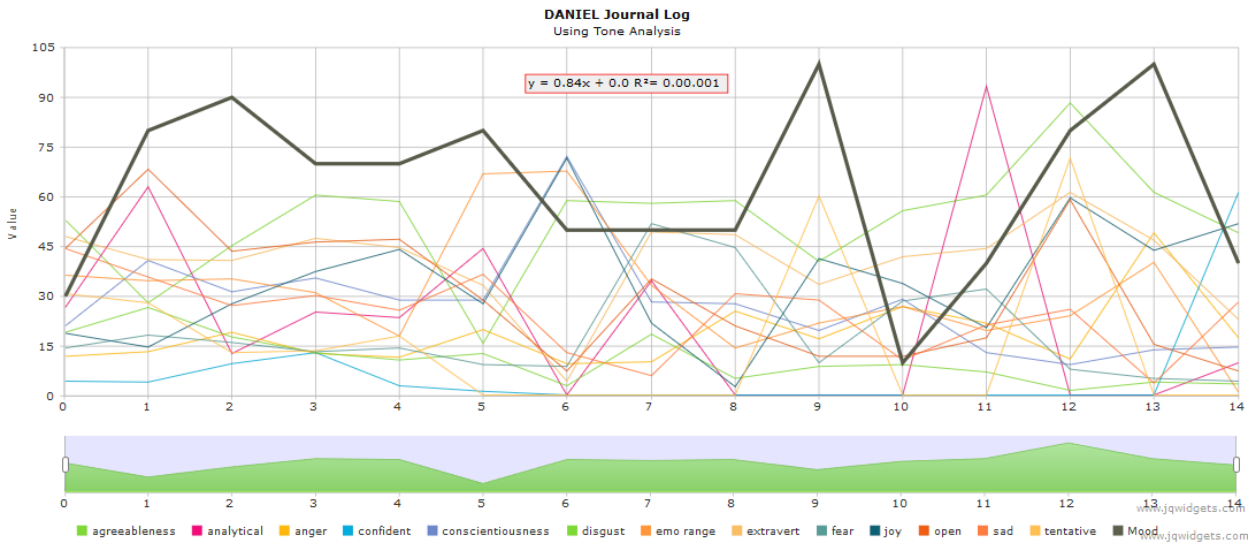
	A	B	C	D
	rank	rank weight	rank weight * 10	rank weight per unit
	9	0.0164	0.164	0.018222222
	5	0.0499	0.499	0.0998
	10	0.0151	0.151	0.0151
	8	0.0201	0.201	0.025125
	11	0.0139	0.139	0.012636364
	12	0.0121	0.121	0.010083333
	7	0.0395	0.395	0.056428571
	4	0.0784	0.784	0.196
0	1	0.4109	4.109	4.109
1	3	0.1399	1.399	0.466333333
2	6	0.0427	0.427	0.071166667
3	13	0.0057	0.057	0.004384615
4	2	0.1551	1.551	0.7755
5				
6		0.9997	9.997	5.859780106
7		Max rank weight	Max Mood value	Predicted mood

Below is a sample of the implementation:

```
com.medlog.webservice.vo.DiaryAnalysisSummaryVO.getIDXFromName(KEY)agreeablen
ess
com.medlog.webservice.vo.DiaryAnalysisSummaryVO.getValFromKeyPct()49.10483333
/ 2.7237583327
ToneKeyValuePair{value=0.007494145199063138, rank=10,
weightedValue=0.0017878043697989374, key=IDX_AGREEABLENESS_BIG5 .....}
com.medlog.webservice.vo.DiaryAnalysisSummaryVO.populateCorrelation() Guess
with 10== 0.3223113980108449

com.medlog.webservice.vo.DiaryAnalysisSummaryVO.getIDXFromName(KEY)analytical
com.medlog.webservice.vo.DiaryAnalysisSummaryVO.getValFromKeyPct()10.06476666
/ 2.7237583327
ToneKeyValuePair{value=0.023328996443750454, rank=9,
weightedValue=0.005565368788207482, key=IDX_ANALYTICAL .............}
com.medlog.webservice.vo.DiaryAnalysisSummaryVO.populateCorrelation() Guess
with 9== 0.527961574671818

com.medlog.webservice.vo.DiaryAnalysisSummaryVO.getIDXFromName(KEY)anger
com.medlog.webservice.vo.DiaryAnalysisSummaryVO.getValFromKeyPct()17.45406666
0000002 / 2.7237583327
ToneKeyValuePair{value=0.9878011969815248, rank=3,
weightedValue=0.23564999737087594, key=IDX_ANGER .................}
com.medlog.webservice.vo.DiaryAnalysisSummaryVO.populateCorrelation() Guess
with 3== 15.628606566413502
```



This displayed the values after step 3 in comparison to mood line (black).

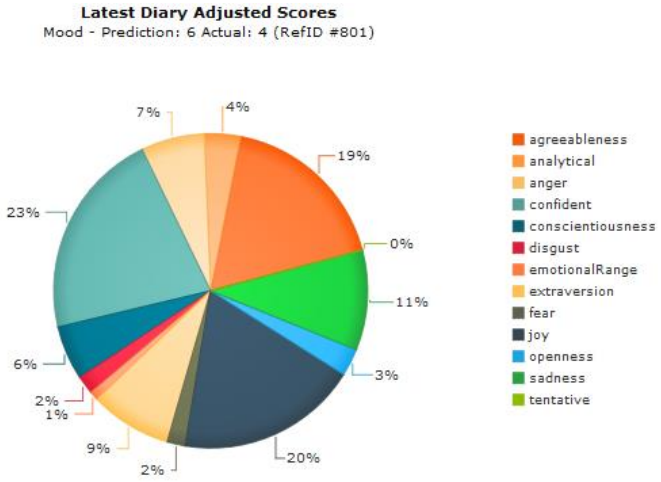


Chart: *key* entry adjusted

The pie chart displays the categories distribution for current diary entry.

Recommendations:

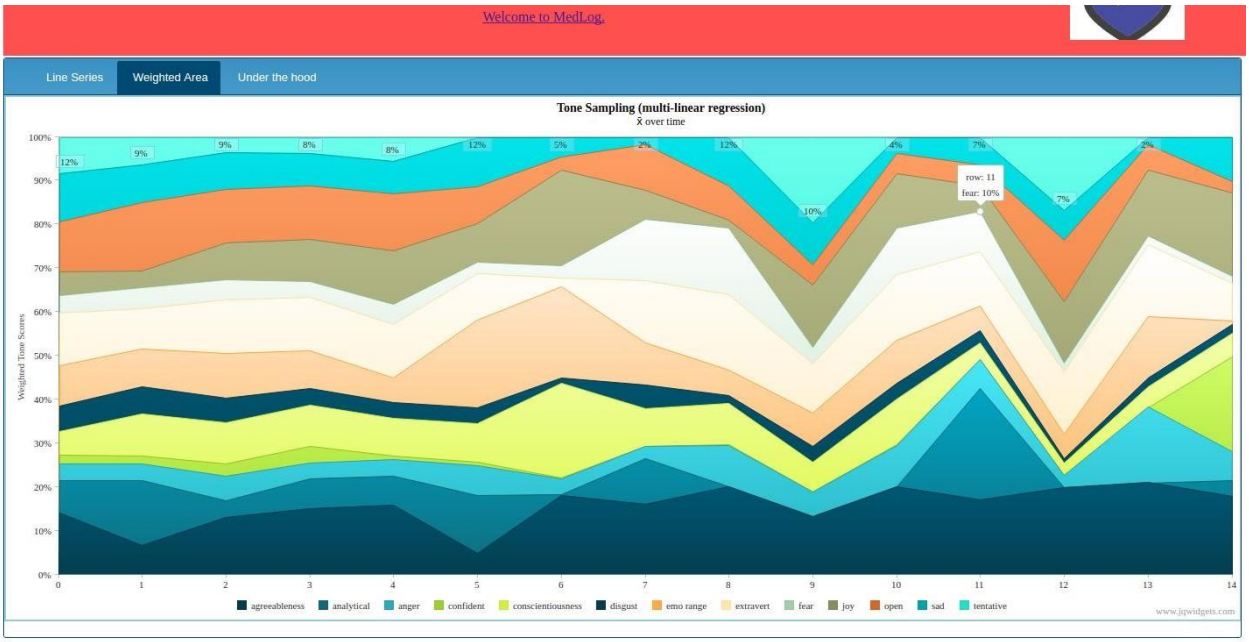
This requires further analyzation of journal entries and inclusion of other factors that may affect the mood, such as medication. Currently, prediction is heavily dependent on current entry with minor contributions from past entries.

Mood predictor was originally planned to work per entry, and overall. Due to time limitations, many bugs are still present. Outliers are data values that falls outside the second and third quartile, when data is presented a stem and leaf plot.

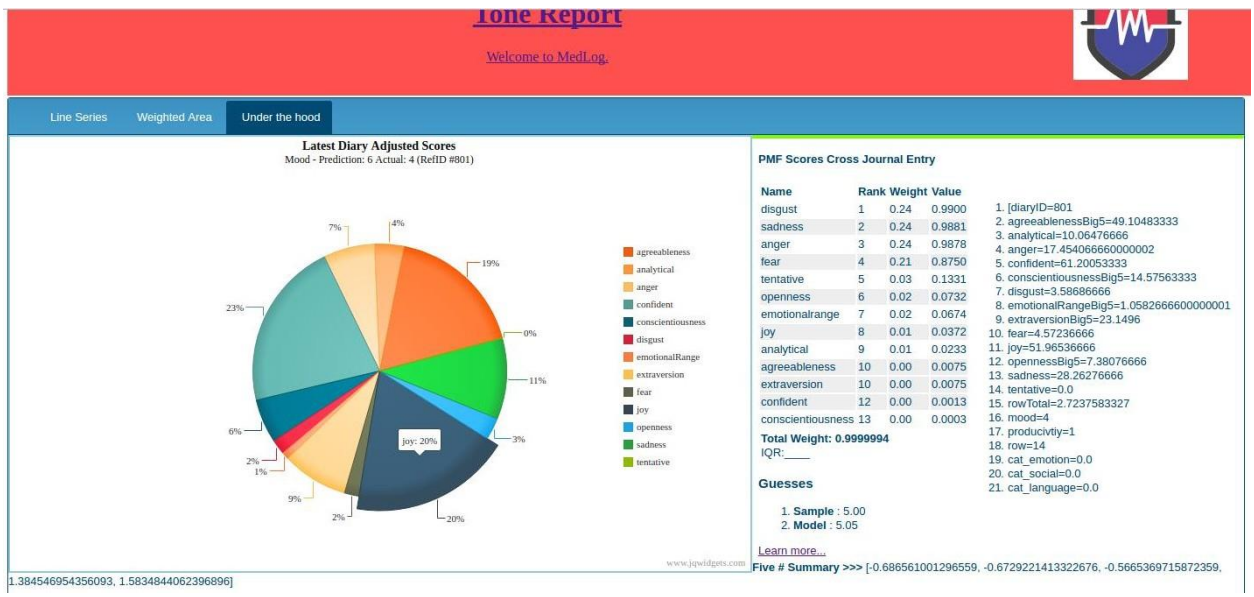
There are also quite a number of limitations. The mood levels are restricted to the range 1-10. The algebra is also constrained due to the number of significant figures it possess. One very important flaw is that the less text a journal entry has, the more likely it would skewed the analysis. One such example is an entry that consist of one angry sentence. Along with flaw, if there are contrasting categories with high values, such as sadness and joy, miscalculate the mood.

There may also be certain trends throughout the years that we may have to consider for a more accurate prediction. For example, one may be most happy during the autumn season months of September to November.

Another glance at Line Series with a glimpse at a weighted tone value



Weighted Area section of Tone Report showing multi-linear regression results and a weighted tone value


Tone Report
Welcome to MedLog.
Line Series
Weighted Area
Under the hood
Latest Diary Adjusted Scores
Mood - Prediction: 6 Actual: 4 (RefID #801)
7%
4%
19%
23%
0%
11%
6%
3%
2%
1%
9%
2%
20%
joy: 20%
agreeableness
analytical
anger
confident
conscientiousness
disgust
emotionalRange
extraversion
fear
joy
openness
sadness
tentative
www.jqwidgets.com
PMF Scores Cross Journal Entry
Name Rank Weight Value
disgust 1 0.24 0.9900
sadness 2 0.24 0.9881
anger 3 0.24 0.9878
fear 4 0.21 0.8750
tentative 5 0.03 0.1331
openness 6 0.02 0.0732
emotionalrange 7 0.02 0.0674
joy 8 0.01 0.0372
analytical 9 0.01 0.0233
agreeableness 10 0.00 0.0075
extraversion 10 0.00 0.0075
confident 12 0.00 0.0013
conscientiousness 13 0.00 0.0003
Total Weight: 0.9999994
IQR:____
Guesses
1. Sample : 5.00
2. Model : 5.05
1. [diaryID=801
2. agreeablenessBig5=49.10483333
3. analytical=10.06476666
4. anger=17.454066660000002
5. confident=61.20053333
6. conscientiousnessBig5=14.57563333
7. disgust=3.58686666
8. emotionalRangeBig5=1.0582666600000001
9. extraversionBig5=23.1496
10. fear=4.57236666
11. joy=51.96536666
12. opennessBig5=7.38076666
13. sadness=28.26276666
14. tentative=0.0
15. rowTotal=2.7237583327
16. mood=4
17. producivtiy=1
18. row=14
19. cat_emotion=0.0
20. cat_social=0.0
21. cat_language=0.0
Learn more...
Five # Summary >>> [-0.686561001296559, -0.6729221413322676, -0.5665369715872359,
1.384546954356093, 1.5834844062396896]