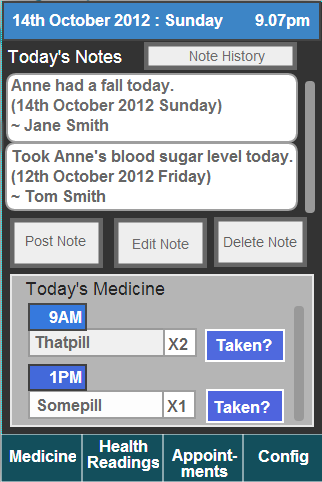
# Introduction

The purpose of this document is to outline the design choices for the web based HTML5 Caring App. It will be detailed in such a way that those outside the project should be able to read and get a strong understanding of how the project would be developed. This document will detail all the technologies involved in the building of this project. In particular the document will detail communication between the Carer web app and Google App Engine.

# GUI Design

The following are examples of the GUI Design. There will be more screens included the app but the following represents the main GUI design types that will appear.

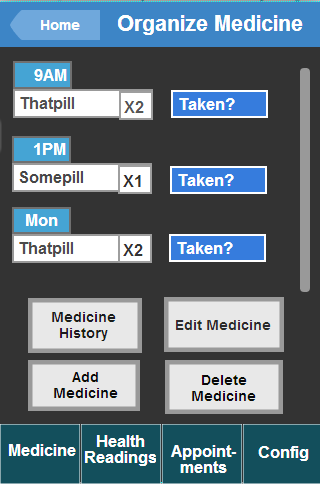
## Main Screen



The main screen allows the user to access a scrolling list of the current day’s medicine. It allows access to a scrolling list of today’s notes. It allows the user to create, edit and delete their notes. Finally a button at the top right allows the user to view Note history, which has a larger scrolling list of notes.

Notes and Medicine are the two most pressing features of my app, so I felt it appropriate that they would make an appearance on the main screen of the app.

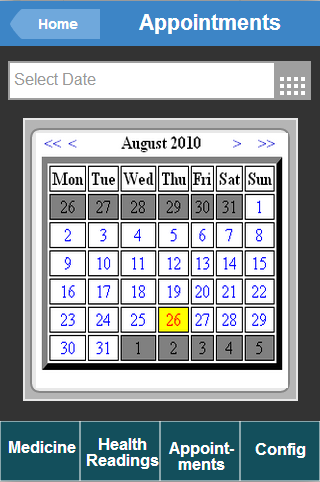
## Medicine Screen



The medicine screen allows the user to view the upcoming medicine that must be taken by the patient. By pressing the “Taken?” button next to each medicine the user confirms that the patient has taken this dosage of medicine and the medicine is removed from the list and is added to medicine history.

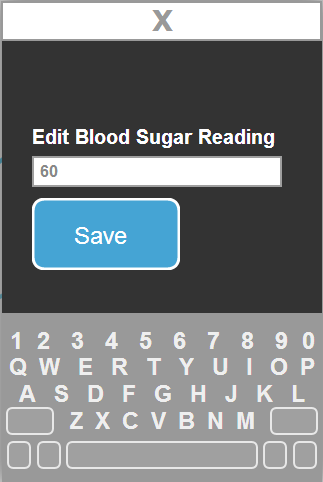
Users can also add new medicine to the system by pressing the add medicine button on this screen. They will also have the option of setting the dosage time and frequency.

## Appointments Screen



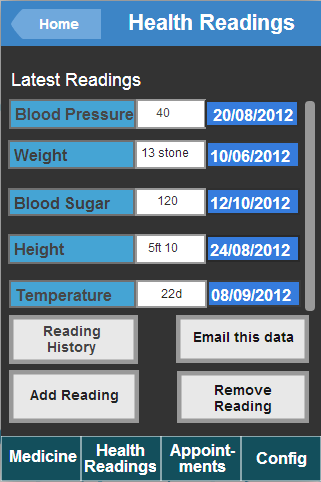
The appointment screen uses the Google Calendar Api to allow the user to keep track of appointments and other important events relating to the patient. Users will be able to create, edit, view and delete appointments using this screen.

## Example of a data entry/edit screen



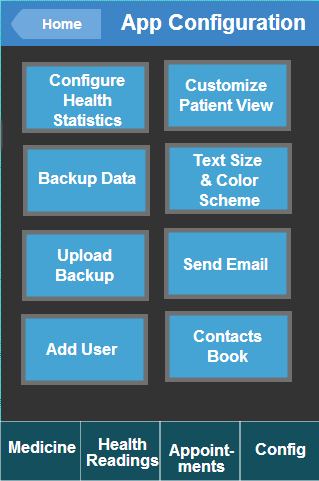
All data entry/edit screens will use a similar format to the above picture. Users will be able to enter values in the available fields and then select the “Save” button to commit their change to the online database. Alternatively the user will be able to click the X button at the top of the screen to return to the screen they came from.

## Health Readings Screen



The health readings screen will allow the user to view the latest readings taken for each Health Statistic. They will be able to add a new reading for a health statistic or remove a reading for a health statistic. This will be done by selecting the Health Statistic in the scrolling list and then pressing either the “Add Reading” or “Remove Reading” button. The user will also be able to select a Health Statistic and view a reading history of that Health Statistic.

## Settings Screen



The settings screen (or config screen) provides the user with various customization and ease of use options. Some of the features displayed in this image are fully intended to be included with the app, while others will be time dependent.

# 

# Mobile <-> Google App Engine communication

## Technology

JSON (Javascript Object Notation) [JSO06] will be used to format data sent over a network connection. JSON is preferable over other data modelling languages such as XML because data formatted with JSON can travel across domains. It also has the benefit of being easily readable.

An example of the JSON data format taken from the main JSON website [EXM11]

:

{

"glossary": {

"title": "example glossary",

"GlossDiv": {

"title": "S",

"GlossList": {

"GlossEntry": {

"ID": "SGML",

"SortAs": "SGML",

"GlossTerm": "Standard Generalized Markup Language",

"Acronym": "SGML",

"Abbrev": "ISO 8879:1986",

"GlossDef": {

"para": "A meta-markup language, used to create markup languages such as DocBook.",

"GlossSeeAlso": ["GML", "XML"]

},

"GlossSee": "markup"

}

}

}

}

}

JSON will also be used with the Google Calendar Api.

An example of JSON format used for a note created by the user (for the caring app project):

{

"author": "Peter Byrne",

"patient": "Mary Byrne",

"content": "Mary needs another reading",

"date": "16/01/2013",

"time": "4.30pm"

};

## Protocol

The protocol that will be used for the connection between the mobile app and Google app engine will be HTTP Secure. [HTT05] Google app engine provides an SSL Certificate for every app it hosts that uses the appspot.com domain.

When using HTTPS the browser receives the domains SSL Certificate. When the browser has confirmed the Certificates validity the date can then be sent to the domain using a public key. This is safe as the public key cannot be used to decrypt the data.

## Security concerns

By using the HTTPS protocol combined with the HTTP Strict Transport Security policy [STS08], the caring app will have greater protection from Man in the middle attacks [MAN04] over the HTTP protocol. In order for the use of the HTTPS protocol to be effective, all user sessions in the app must use the HTTPS protocol at all times. If there is even just one point in the app that uses HTTP, the security of the app will be compromised and users will be open to attack. [OWA07] So in order to protect users HTTP Strict Transport Security declares a website as only accessible through HTTPS.

Users will be left vulnerable if session tokens are not properly addressed. The session token if compromised will allow a user with malicious intent to wrongly access authenticated areas of the app.

Session tokens must be of a reasonable length, if they are too short they risk being compromised through brute force tactics. They should include upper and lower case characters. It will not be enough to create a strong session token, it also must not be easily predicted. If a user with malicious intent manages to uncover a session token they may also be able to identify a pattern in the token and compromise more session tokens. In this way even the vulnerability of one session token could have devastating effects.

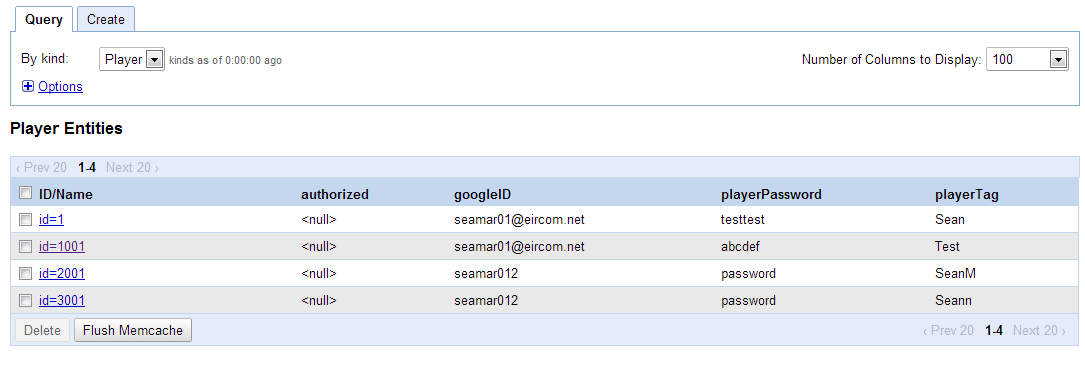
The app will need to take measures to prevent Session Fixation. This is an attack against the user where the attacker attempts to identify the session token (or ID) that the user is using. [FIX10]

# Data storage and Google App Engine Datastore

I have chosen to use Datastore supplied by Google App Engine. It is a No-SQL schemaless object datastore. [DAT01] App engine uses the High Replication Datastore as its primary data repository. It uses the Paxos algorithm [PAX02] to replicate data across multiple data centres. Datastore is built on top of Bigtable which is a distributed storage system. [BIG03]

The datastore is fully scalable and does not require tables or fields to be manually added by the administrator.

The following is an example view of the Datastore using the Datastore Viewer on App engine. The fields in this example do not relate to this project, they are just an example used to explain the concept.

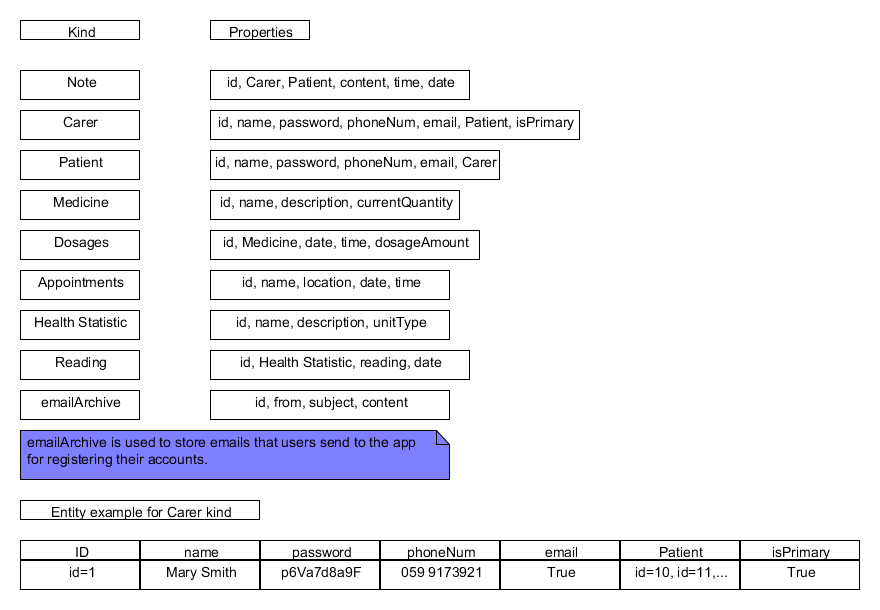


Player is an example of a kind. Each individual row is an example of a Player entity.

ID/Name, authorized, googleID, playerPassword and playerTag are examples of properties relating to the Player entities contained within player kind.

Properties can also contain other entities, for example if there was another property above titled “friend” the friend field may reference one or more player entities.

The following diagram shows the structure of the datastore in regards to the caring app project. It shows the kinds and properties used for storing information. It also gives an example of a Carer entity.



## Reading

Each reading entity contains an **id** (i.e. a unique identification number), a **reference** to the **Health Statistic** it relates to (for example if a reading is taken for a patients height, that reading entity will contain a reference to the Height entity id), a **reading value** (.i.e. a decimal number) and finally a **date** that the reading was taken on.

## Dosages

Each dosages entity contains an **id** (i.e. a unique identification number), a **reference** to the **Medicine** it relates to (for example if a dosage is created for paracetamol, the dosage entity will contain a reference to the Paracetamol entity ID), the **date** which the medicine must be taken on, the time at which the medicine must be taken and finally a **dosage amount** that must be taken. (i.e. a string)

## EmailArchive

EmailArchive will be used to store emails sent to the app administrator for registration purposes. I.e. when they first sign up for an account they will be asked to confirm their registration by responding to an automated email. (I.e. clicking a link they receive in the email) The EmailArchive will then be used to confirm a user’s intention to register for the service.

## Carer, Patient, Medicine and Health Statistic kinds

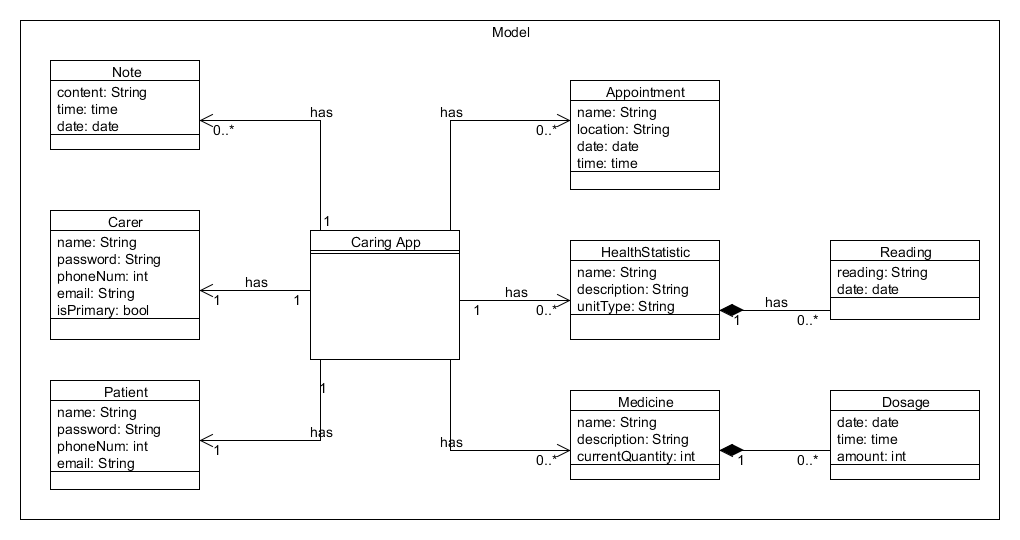
In the above diagram, these four kinds are all used as properties of another kind. The reason for this is that each Note must be aware of the Patient entity it relates to and also the User who created the note (who may be either a Patient or a Carer)

Each Carer entity must be aware of the Patient entity it relates to while each patient should be aware of the Carers it relates to. A dosage entity needs to be aware of the Medicine entity it relates to and the reading entity will need to be aware of the Health Statistic entity it relates to.

Each Carer entity will contain a Patient array with the patient ids of the patients they care for.

# Class Diagram

The following class diagram describes the phone user side of the app.



# References

[DAT01] Datastore, *Datastore Overview*, Google App Engine, Available at: https://developers.google.com/appengine/docs/python/datastore/overview

[Accessed Jan 10 2013].

[PAX02] Tushar Chandra, Robert Griesemer, Joshua Redstone, *Paxos Made Live – An Engineering Perspective*, Available at: http://static.googleusercontent.com/external\_content/untrusted\_dlcp/research.google.com/en/us/archive/paxos\_made\_live.pdf

[Accessed Jan 10 2013].

[BIG03] Google Inc. , *Bigtable: A Distributed Storage System for Structured Data*, Available at: http://static.googleusercontent.com/external\_content/untrusted\_dlcp/research.google.com/en//archive/bigtable-osdi06.pdf

[Accessed Jan 10 2013].

[MAN04] OWASP, *Man-in-the-middle-attack*, OWASP, Available at: https://www.owasp.org/index.php/Man-in-the-middle\_attack

[Accessed Jan 11 2013].

[HTT05] SearchSoftwareQuality, *HTTPS (HTTP over SSL or HTTP Secure)*, Available at: http://searchsoftwarequality.techtarget.com/definition/HTTPS

[Accessed Jan 10 2013].

[OWA07] OWASP, *Broken Authentication and Session Management*, OWASP, Available at: https://www.owasp.org/index.php/Broken\_Authentication\_and\_Session\_Management

[Accessed Jan 11 2013].

[STS08] OWASP, *HTTP Strict Transport Security*, OWASP, Available at: https://www.owasp.org/index.php/HTTP\_Strict\_Transport\_Security

[Accessed Jan 11 2013]

[WHS09] Whitehat Security Blog, *Secure Session Tokens*, Whitehat, Available at: http://blog.whitehatsec.com/secure-session-tokens-2/

[Accessed Jan 11 2013]

[FIX10] OWASP, *Session Fixation,* OWASP, Available at: https://www.owasp.org/index.php/Session\_fixation

[Accessed Jan 11 2013]

[EXM11] JSON, *JSON Example,* Available at:http://*json*.org/example.html

[Accessed Jan 11 2013]