Online Health Monitoring System

Deliverables II

COP 4331, Fall, 2014

Team Name: Team 14

Team Members:

- Jon Carelli - email - page
- Chris McCue - email - page
- Chris Chaffin - email - page
- Ethan Pitts - email - page
- David Gundler - email - page
- James Luke - email - page

Contents of this Document

Deliverables II

Online Health Monitoring System

High Level Design

COP 4331, Fall, 2014

Modification history:

Version	Date	Who	Comment
v0.0	08/15/00	G. H. Walton	Template
v1.0	10/22/14	C. N. McCue	First Draft

Team Name: Team 14

Team Members:

- Jon Carelli - email - page
- Chris McCue - email - page
- Chris Chaffin - email - page
- Ethan Pitts - email - page
- David Gundler - email - page
- James Luke - email - page

Contents of this Document

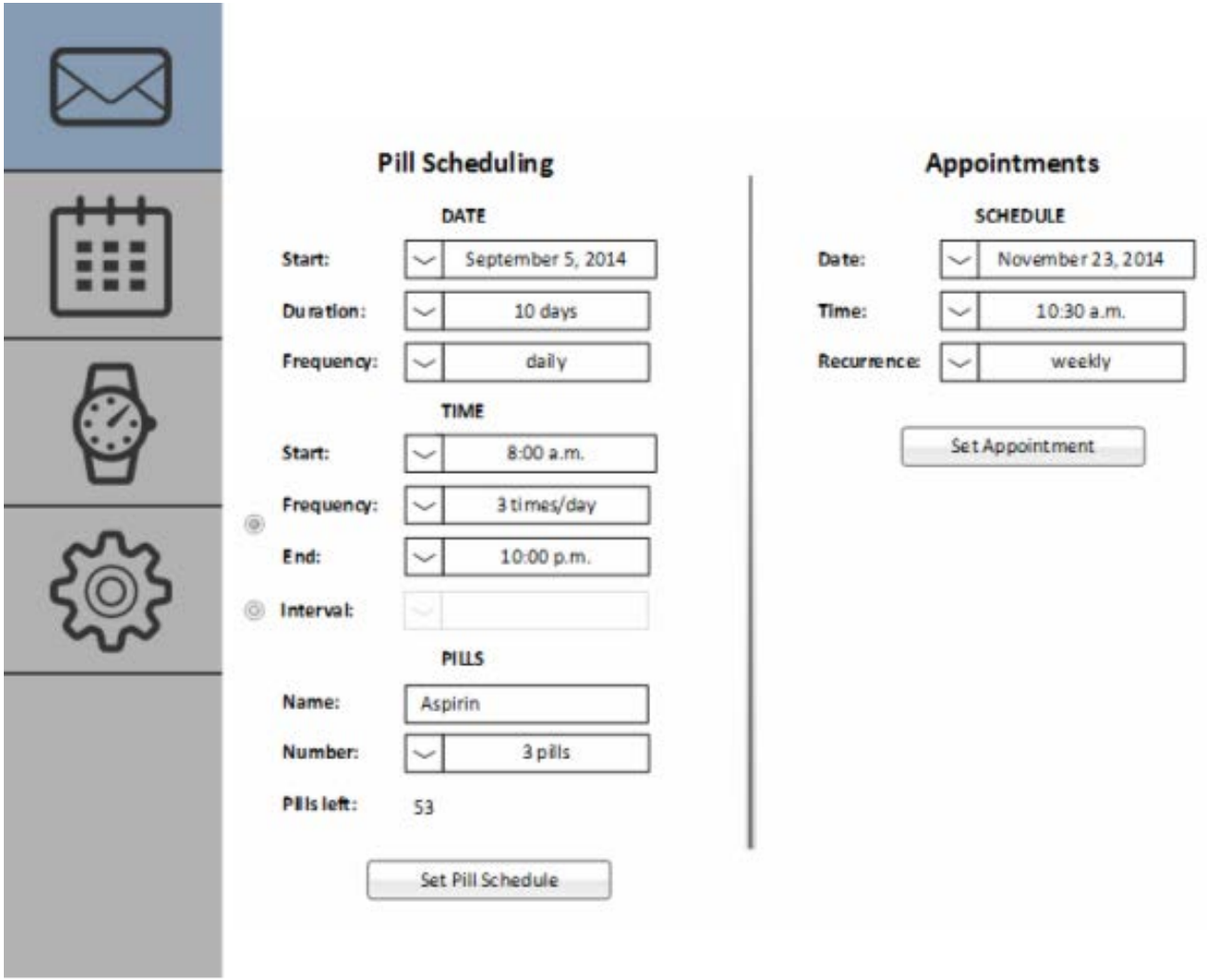
High-level Architecture:

- **Diagrams of Major Components and Interfaces**

  For High-Level design, the team organized the operational features from the Concept of Operations in logical order. Functionality was divided between two users: doctors and patients. Functionality was furthered divided into modules, such as chat and pill/appointment manager. At this point, we essentially have the front end modules.

Operational Features (Concept of Operations) to Modules (High-Level Design)		
Module	**User**	
	Doctor	*Patient*
Pill/Appointment Manager	Ability to create pill schedules	
	Ability to create appointments	
SMS Alerts		Text message reminders to take pills
		Text message reminders of appointments
Chat Manager	Ability for doctor-patient messaging	Ability for patient-doctor messaging
Calendar		Ability to view pill schedules on calendar
		Ability to view appointments on calendar
Video Chat (optional)	Ability to video chat with patient	Ability to video chat with doctor

We then sketched a diagram of the user interfaces and the flow from one to another. Button, list boxes and other controls were discussed and added to these blocks. However, in the detailed design phase, more thought will be given to these controls. Below is a visualization of a user interface screen with the Navigator tabs:



Visualization of User Interface (Doctor View)

In the sketch, it was determined that Settings, Login and New Account management would also be needed. At this point, we have a diagram of the front end.

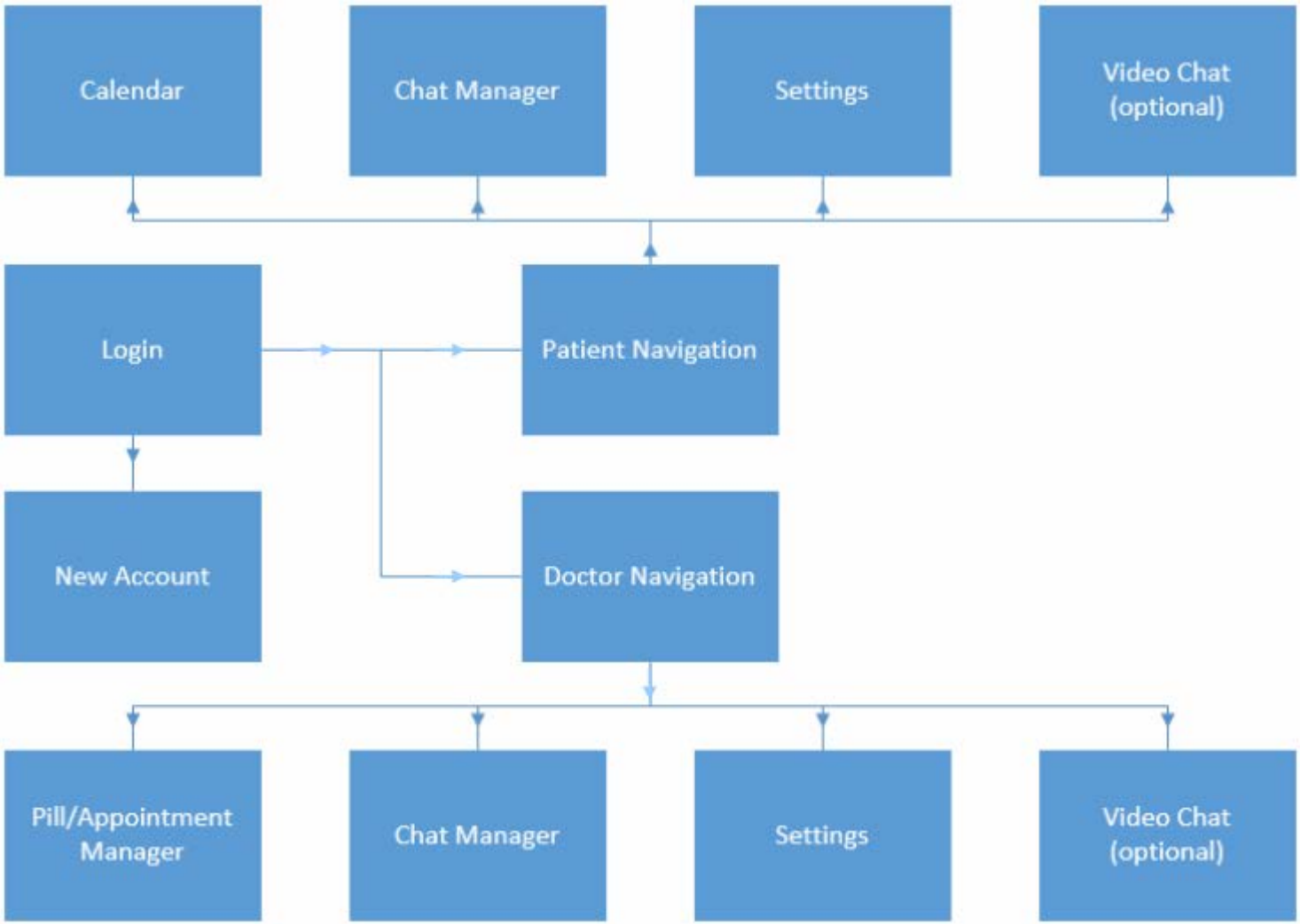


Diagram of Front End User Interface Flow

Additionally, back end modules need to be considered. The Database and Database Manager are central to this system. Also, there must be an SMS Alert module in the back end.

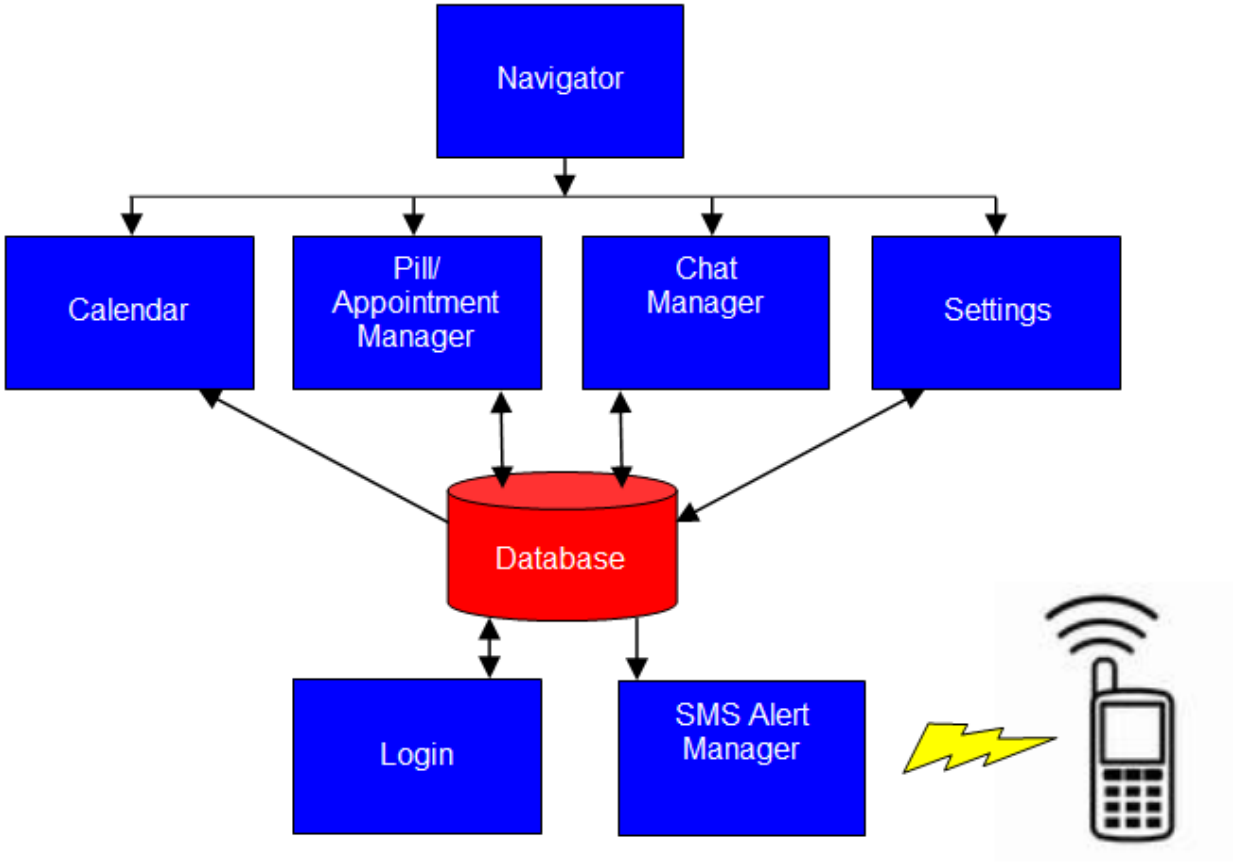


Diagram with Back End Modules

In the end, it was determined that Login is intimately tied with the Database and Settings. Therefore, the Database, Database Manager, Login and Settings are considered one module. This is the central module of the whole system. Hence, there are six modules. These are described in detail in the next section.

- **Description of modules and interfaces**

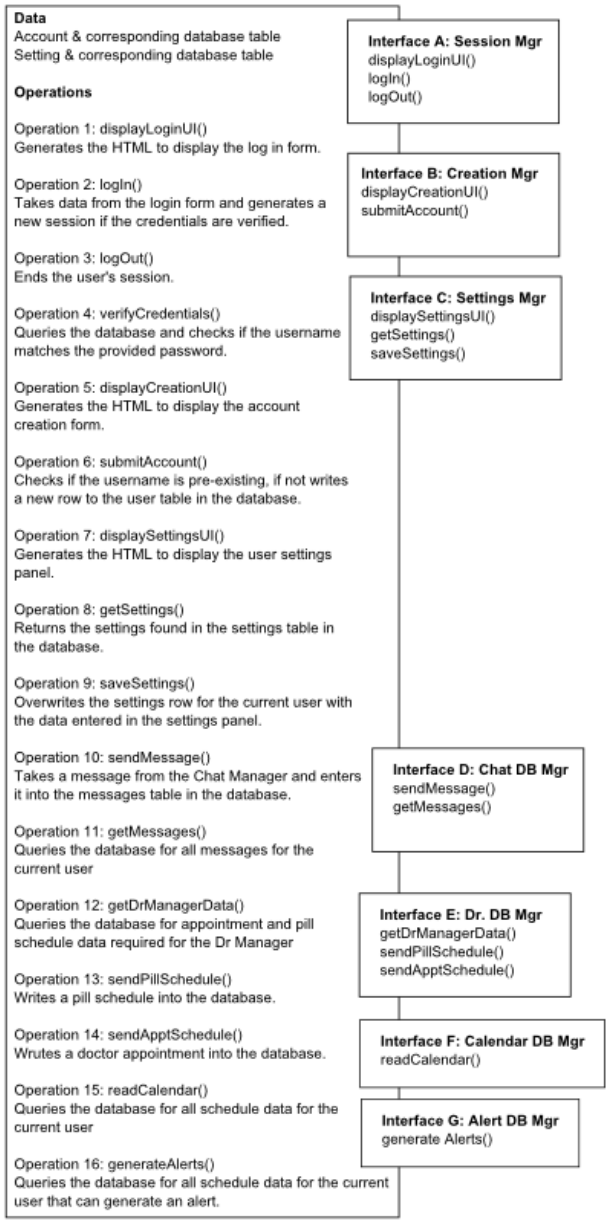
Detailed Design Modules		
Module	**Interfaces**	**Team Member**
Database/Login/Settings	All except Navigator	James
SMS Alerts	Database	David
Navigator	All except SMS Alerts	Ethan
Chat Manager	Database, Navigator	Jon
Calendar	Database, Navigator	Chris Chaffin
Pill/Appointment Manager	Database, Navigator	Chris McCue

1. **Database/Database Manager/Login/Settings** (James)

   Accessibility: Neither doctors nor patients use this back end module directly.

   Functionality:
   - Login: Creates login session. User logs in with id and password.
   - Settings: Users can change settings of this system.
   - Database Manager: Stores and retrieves information from the database.
   - Database: This is the central storage for all modules.

   Interface with:
   - Database interfaces with every module except Video Chat
   - Other team members will use James' work as a "template".


Data
Account & corresponding database table
Setting & corresponding database table
Operations
Operation 1: displayLoginUI()
Generates the HTML to display the log in form.
Operation 2: logIn()
Takes data from the login form and generates a new session if the credentials are verified.
Operation 3: logOut()
Ends the user's session.
Operation 4: verifyCredentials()
Queries the database and checks if the username matches the provided password.
Operation 5: displayCreationUI()
Generates the HTML to display the account creation form.
Operation 6: submitAccount()
Checks if the username is pre-existing, if not writes a new row to the user table in the database.
Operation 7: displaySettingsUI()
Generates the HTML to display the user settings panel.
Operation 8: getSettings()
Returns the settings found in the settings table in the database.
Operation 9: saveSettings()
Overwrites the settings row for the current user with the data entered in the settings panel.
Operation 10: sendMessage()
Takes a message from the Chat Manager and enters it into the messages table in the database.
Operation 11: getMessages()
Queries the database for all messages for the current user
Operation 12: getDrManagerData()
Queries the database for appointment and pill schedule data required for the Dr Manager
Operation 13: sendPillSchedule()
Writes a pill schedule into the database.
Operation 14: sendApptSchedule()
Wrutes a doctor appointment into the database.
Operation 15: readCalendar()
Queries the database for all schedule data for the current user
Operation 16: generateAlerts()
Queries the database for all schedule data for the current user that can generate an alert.
Interface A: Session Mgr
displayLoginUI()
logIn()
logOut()
Interface B: Creation Mgr
displayCreationUI()
submitAccount()
Interface C: Settings Mgr
displaySettingsUI()
getSettings()
saveSettings()
Interface D: Chat DB Mgr
sendMessage()
getMessages()
Interface E: Dr. DB Mgr
getDrManagerData()
sendPillSchedule()
sendApptSchedule()
Interface F: Calendar DB Mgr
readCalendar()
Interface G: Alert DB Mgr
generate Alerts()

2. **Navigator** (Ethan)

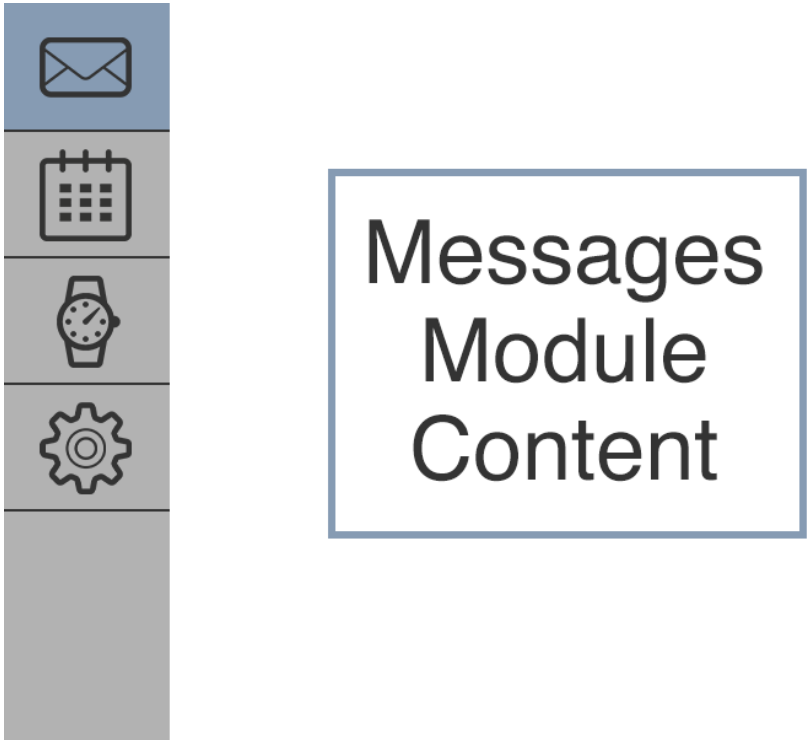
Accessibility: This is the main interface for both doctors and patients alike.

Functionality:

- This will be the "template" into which are modules fit. The Navigator is the style of the user interface more than a module.

Interfaces with:

- The Navigator interfaces with the Chat Manager, Pill/Appointment Manager, Calendar and, possibly, the Video Chat. Integration is as simple as calling up a HTML page.
- The Navigator also interfaces with the Login screen. The login screen will call the Navigator page.



3. **Chat Manager** (Jon)

Accessibility: Both doctor and patient use

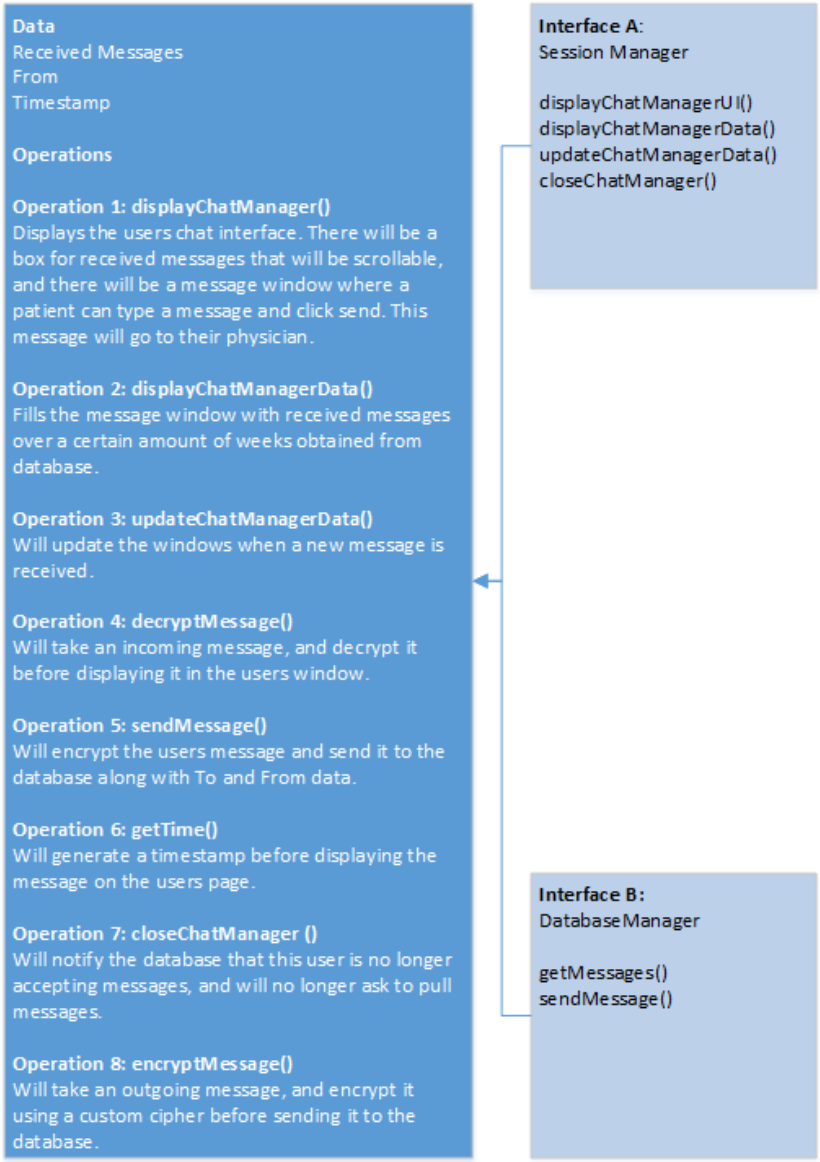
Functionality:

- Patients must be able to read, delete, reply, etc. to one doctor
- Doctors must be able to read, delete, reply, etc. to multiple patients

Interfaces with:

- Database Manager- Receive messages from database. Send messages to database.
- Navigator, or Session Manager- Chat Manager integrates with Navigator.
- See next page for diagram.

Chat Manager Overview:



Development Stages:

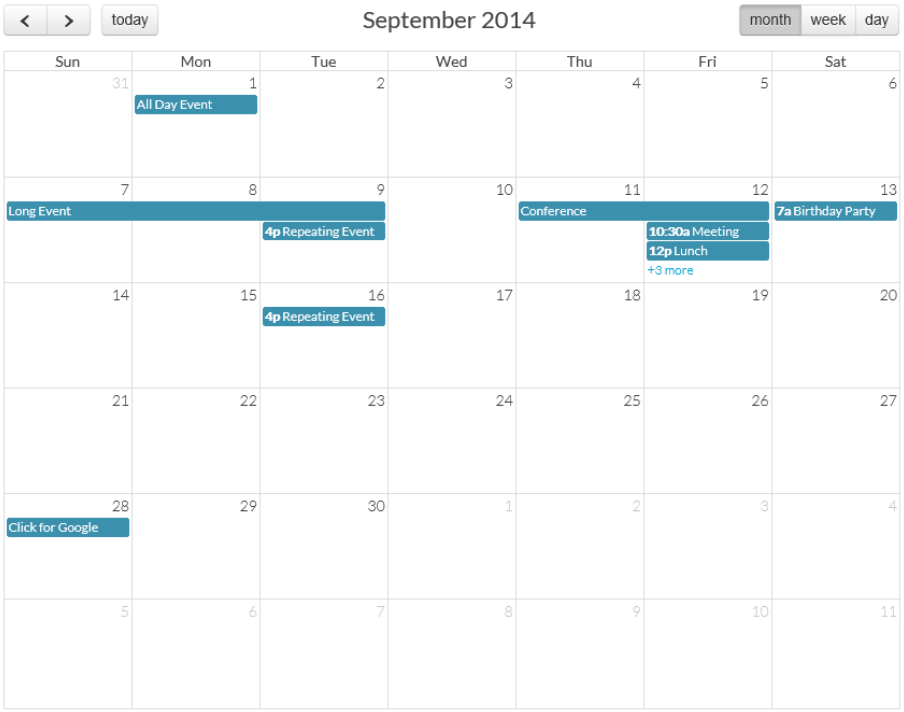
- During prototyping stage, get chat information from data file. Emphasis on functionality, not design. No need to follow Ethan's design scheme.
- During integration stage, get information from database. Use Ethan's design scheme.

4. **Calendar** (Chris C.)

Accessibility: Only patient uses this

Functionality:

- Calendar will be filled in with pill times and medical appointments.
- Navigate forward and backwards to different months.
- This will be implemented with a plug in API calendar. Tentatively, http://arshaw.com/fullcalendar will be implemented. An example of this calendar is shown below:



Interfaces with:

- Database Manager- Read database information and place on calendar. No need to write to database.
- Navigator, or Session Manager- Navigator will place Calendar in blank area of Navigator.

Development Stages:

- During prototype stage, get information from a data file to fill in calendar
- During integration state, will need to get data from database and place in Ethan's design

5. **Pill/Appointment Manager** (Chris M.)

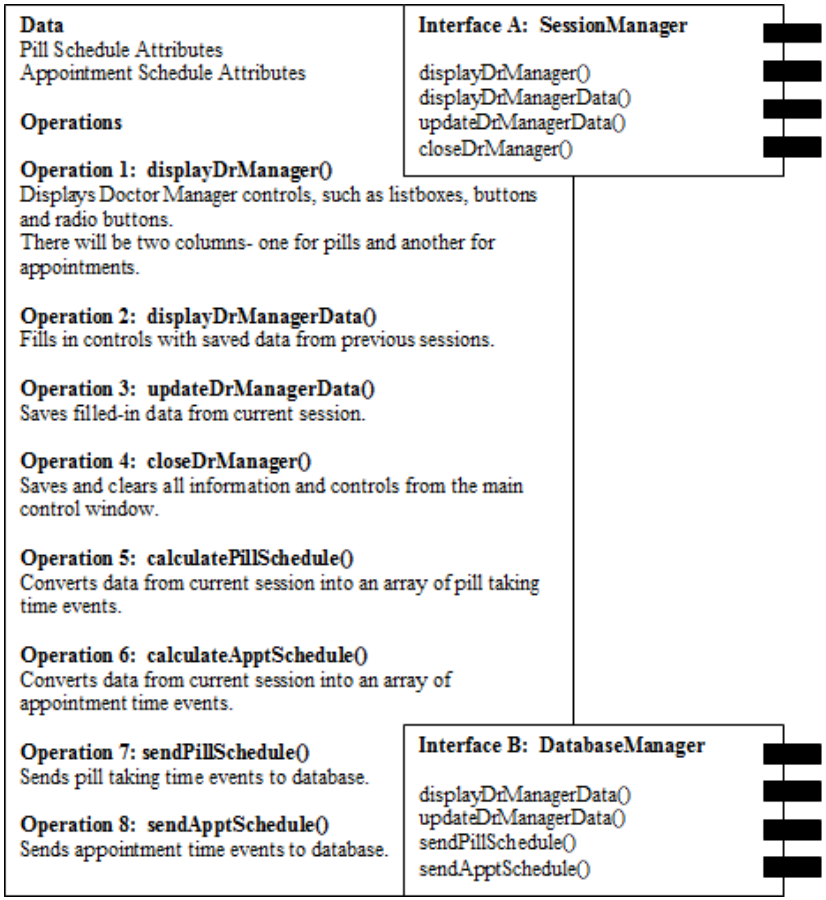
Accessibility: Only doctor uses this.

Functionality:

- Use list boxes, buttons and other controls to create pill schedules.
- Use list boxes, buttons and other controls to create appointments.

Interfaces with:

- Database Manager- Database information is read in to fill in previous appointments and pill schedules. New appointments and pills schedules are stored in the database.
- Navigator, or Session Manager- This module is integrated into the Navigator tab scheme.
- See diagram on next page.



Development Stages:

- During prototype stage, get information to fill in forms from data file and store information in that same data file
- During integration stage, get information from and send information to database. Also, integrate with Ethan's design.

6. **Alert Manager** (David)

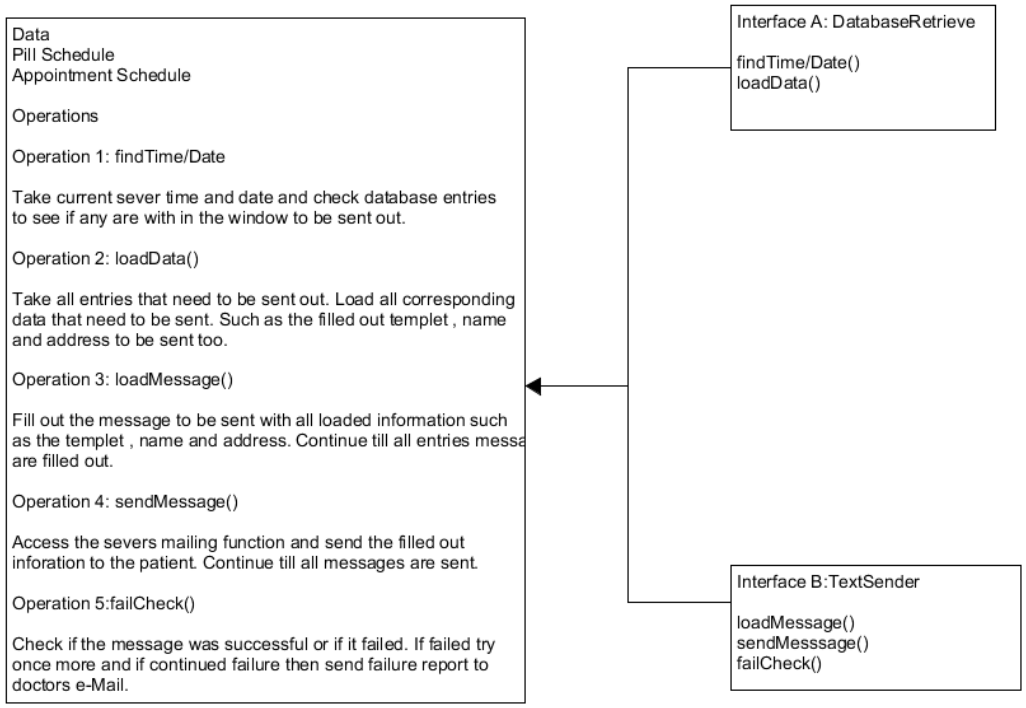
Accessibility: Neither doctors nor patients interact with this. Text alerts are automatically sent to patient's cell phone.

Functionality:

- Send automatic SMS texts to patients based on pill schedule.
- Send automatic SMS texts to patients based on appointments.

Interfaces with:

- Database Manager- Queries database and auto generates alerts accordingly.
- SMS devices- Sends SMS texts to devices, such as cell phones.



Development Stages:

- During prototype stage, the main focus should be on using a computer to send a text to a cell phone.
- During integration stage, constantly query database and send alerts based on these queries.

7. **Video Chat** (optional)

At this point, we will not be implementing video chat.

Design Issues:

Issues relative to this project include safety, ease-of-use, performance and portability. Since this is a health-related product, failure could result in harm or death to the user. Pill schedules can become a life and death issue. Doctors could enter pill schedules wrong or software could fail causing botched pill schedules. Appointments are a crucial issue. If one misses an appointment, it may be weeks or months before one can reschedule. Waiting is not an option when one is sick. It is important that there is no software failure. Otherwise, the patient's life could be endangered.

Ease-of-use is related to safety. If the system is easy to use, doctors will be able to enter pill schedule and appointment data correctly. We want to eliminate the possibility of doctors entering bad pill schedules because of complicated controls. In addition, ease-of-use is important in itself. Many patients will be elderly or vulnerable since they are sick. Therefore, the Navigator must be as simple as possible. A user should be able to understand how to use the Online Health Monitoring System in 10 minutes or less. Buttons and selections in the user interface must be very clear so that users will not be confused on what they are selecting. The user interface visualization demonstrates that the user controls are configured in a standard format that users would already be familiar with. Since this is an online application it must be portable. Since this system deals with many patients and patient data, it is important that it have high performance. Storage and retrieval of information should occur within 3 seconds. Otherwise, users will become frustrated with the poor performance.

Future relevant issues include reusability and maintainability. Certainly, other features would be added to this as time goes on. Therefore, it would need to have reusability. The Navigator should allow for more tabs to be added. The database should be capable of handling more data. Maintainability would also be crucial. The pill and appointment data sent and retrieved from the database must be perpetually maintained. This data is safety critical. At some point, database capabilities would need to be expanded. Realistically, a health monitoring product has a great deal of liability and risk. It would be difficult to use this product in the real world. Doctors would not want to waste their time on a new product. Therefore, the issues of reusability and maintainability are not realistic.

Due to time constraints, it is not feasible to consider alternative design strategies. Our prototyping is solely for the purpose of stepping towards integration. Since we are using an incremental phased development approach, we plan to make prototypes of each of the modules. Then we will integrate these modules one by one. The Calendar, Pill/Appointment Manager and Chat Manager will be integrated with the Database Manager. These will then be integrated with the Navigator. In the meantime, the Navigator will be integrated with Login. Finally, the SMS Alert Manager will be integrated with the Database Manager.

We expect most of our technical difficulties to come from integration with the Database Manager. Most of the team is inexperienced with databases. However, James does have database experience. He will help us integrate our modules with the database. He is planning on making wrappers to make integration easier for other team members. Still, this does pose a challenge. Other technical difficulties are team inexperience with JavaScript, SMS messaging and video chat. Our solution to all the difficulties was to start research early. We started learning JavaScript early in the project. David started researching SMS messaging early in the project as well. Although James did some video chat research early in the project, this is optional. Due to time constraints, it is unlikely that we will implement this option.

Our architecture style is a hybrid of client-server and repositories architectures. Since this is an online project, the client-server was an obvious choice of architecture. Due to this system storing and retrieving data from a database, the repositories architecture was also used. There were no design trade-offs in this

selection of architecture. This project is straightforward. We are just using the established tools and techniques for this sort of problem. Again, the technical risks are team member inexperience with databases. However, James does have database experience. He is planning to help each team member with database integration. Communication from other team members and James will be crucial for the coding phase of this system.

Template created by G. Walton (GWalton@mail.ucf.edu) on October 8, 1999 and last modified on Aug 15, 2000

This page last modified by Christopher McCue (christopher.mccue@knights.ucf.edu) on October 22, 2014

Online Health Monitoring System

Detailed Design: Database/Database Manager/Login/Setting

COP 4331, Fall, 2014

Modification history:

Version	Date	Who	Comment
v0.0	08/15/00	G. H. Walton	Template
v1.0	10/9/14	James Luke	First commit

Team Name: Team 14

Team Members:

- Jon Carelli - email - web page
- Chris McCue - email - web page
- Chris Chaffin - email - web page
- Ethan Pitts - email - web page
- David Gundler - email - web page
- James Luke - email - web page

Contents of this Document

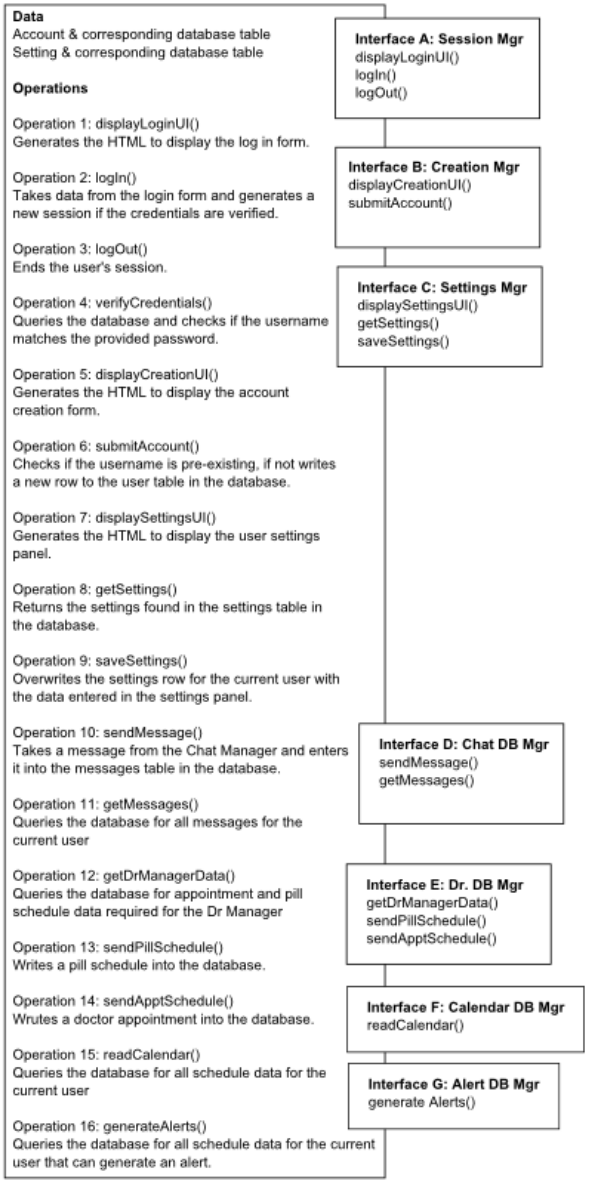
Design Issues:

When storing a user's medical data, security is paramount. We do not want an attacker to be able to steal sensitive information that could be used to harass or blackmail patients. For this reason, we will be encrypting the MySQL database. To protect our user's passwords from being stolen, we will implement salting and hashing in our user account store. These measures, while not bulletproof, will help keep the users safe in the unlikely event that our server is compromised by attackers online or within the server room.

Since most modules will be depending on retrieving and modifying this data, I will need to work closely with all members of the team. In order to reduce work, we will need to agree on uniform modes of access to the system.

Detailed Design Information:

Overview:


Data
Account & corresponding database table
Setting & corresponding database table
Operations
Operation 1: displayLoginUI()
Generates the HTML to display the log in form.
Operation 2: logIn()
Takes data from the login form and generates a new session if the credentials are verified.
Operation 3: logOut()
Ends the user's session.
Operation 4: verifyCredentials()
Queries the database and checks if the username matches the provided password.
Operation 5: displayCreationUI()
Generates the HTML to display the account creation form.
Operation 6: submitAccount()
Checks if the username is pre-existing, if not writes a new row to the user table in the database.
Operation 7: displaySettingsUI()
Generates the HTML to display the user settings panel.
Operation 8: getSettings()
Returns the settings found in the settings table in the database.
Operation 9: saveSettings()
Overwrites the settings row for the current user with the data entered in the settings panel.
Operation 10: sendMessage()
Takes a message from the Chat Manager and enters it into the messages table in the database.
Operation 11: getMessages()
Queries the database for all messages for the current user
Operation 12: getDrManagerData()
Queries the database for appointment and pill schedule data required for the Dr Manager
Operation 13: sendPillSchedule()
Writes a pill schedule into the database.
Operation 14: sendApptSchedule()
Wrutes a doctor appointment into the database.
Operation 15: readCalendar()
Queries the database for all schedule data for the current user
Operation 16: generateAlerts()
Queries the database for all schedule data for the current user that can generate an alert.
Interface A: Session Mgr
displayLoginUI()
logIn()
logOut()
Interface B: Creation Mgr
displayCreationUI()
submitAccount()
Interface C: Settings Mgr
displaySettingsUI()
getSettings()
saveSettings()
Interface D: Chat DB Mgr
sendMessage()
getMessages()
Interface E: Dr. DB Mgr
getDrManagerData()
sendPillSchedule()
sendApptSchedule()
Interface F: Calendar DB Mgr
readCalendar()
Interface G: Alert DB Mgr
generate Alerts()

Data:

This module works closely with most other modules in this project. Much of the functionality acts as a wrapper to the database for these modules. In this way, we will come in contact with many of the data structures defined in their detailed designs:

- Chat Manager - Message
- Dr. Manager - Pill Schedule, Appointment Schedule
- Event Manager - Event
- Calendar Manager - Calendar format

In addition, this module will incorporate two new data structures: Setting and Account.

Setting
String username
String email
String doctorName
Bool isDoctor

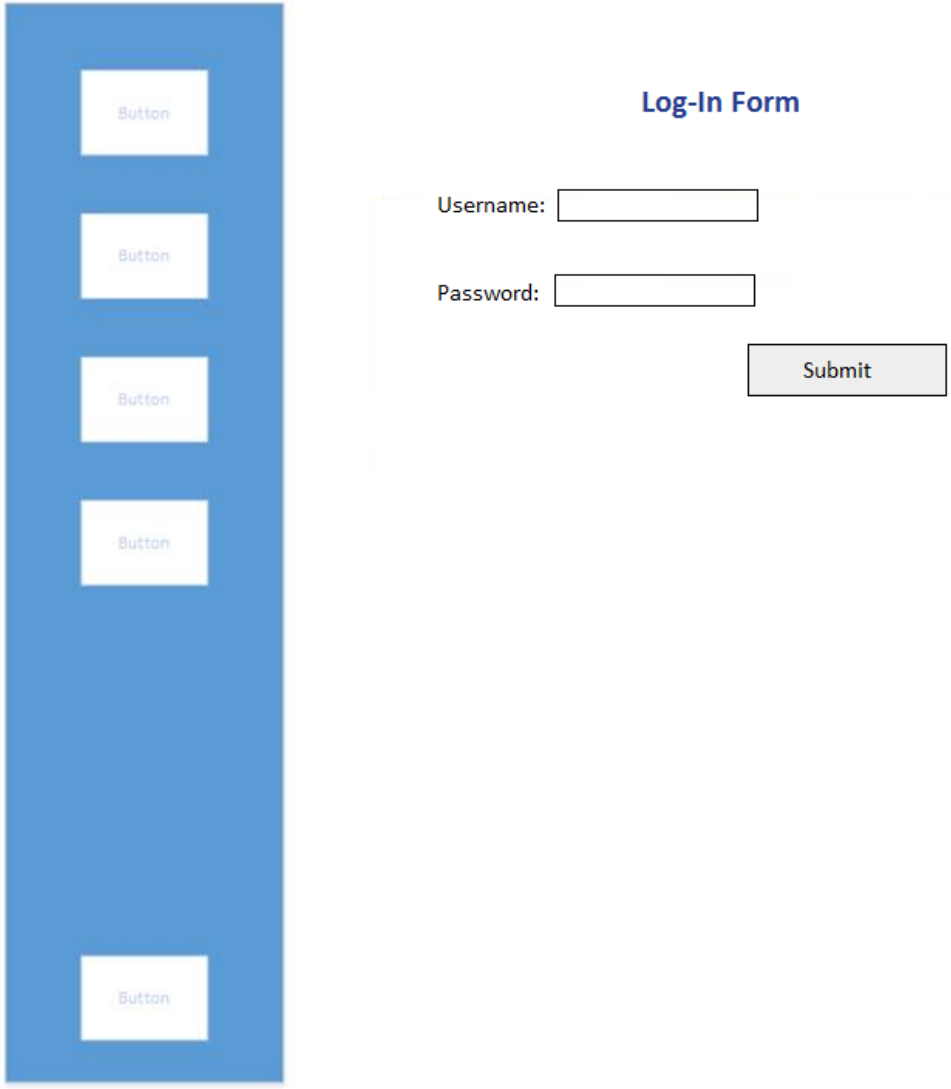
Account
String username
String salt
String hashPassword

In the **Setting** structure, *doctorName* corresponds with the username of the user's doctor. This is used to match doctors and users in a one-to-many fashion. A boolean value of true in *isDoctor* indicates that this account belongs to a doctor.

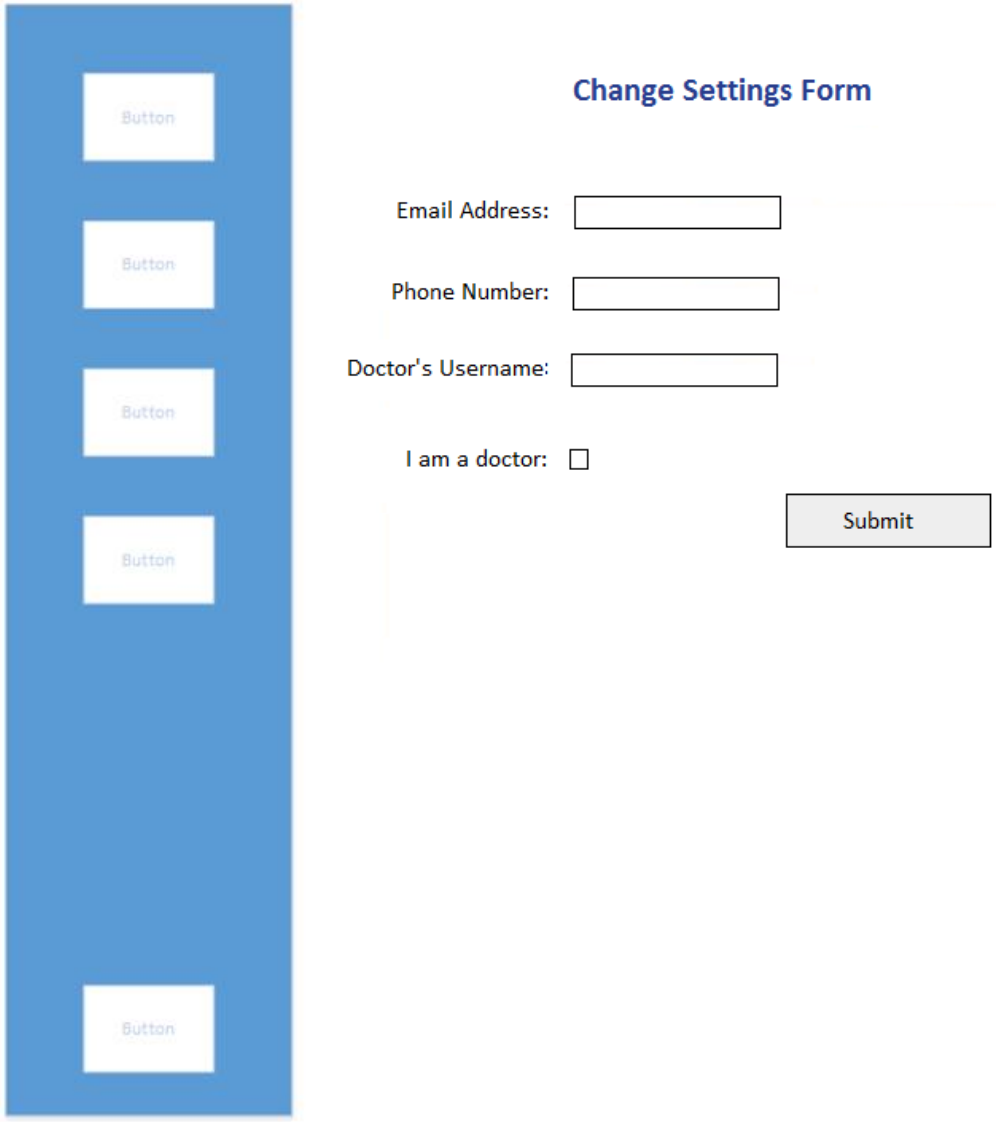
In the **Account** structure, *salt* refers to the random data used in the one way hash algorithm provided by PHP. The *hashPassword* is the result of that hash algorithm. Both are needed to verify a user's real password.

User Interface Visualization

Visual of HTML page produced by displayLoginUI()



Visual of the HTML page produced by displaySettingsUI()



Operations:

As noted above, one of the main functions is to pass data in-between modules and the database. This allows other team members to not have to learn SQL queries and the details of how the data is stored. The other duty this module performs is to keep track of user accounts and the process of logging in and out. Many of these other functions will heavily rely on the built-in capabilities of PHP, such as the native password hashing API.

PHP is a great way to interface with our MySQL installation. It includes a library of functions dedicated to MySQL operations.

We will make the most of the features available to us through PHP to store and retrieve our user data.

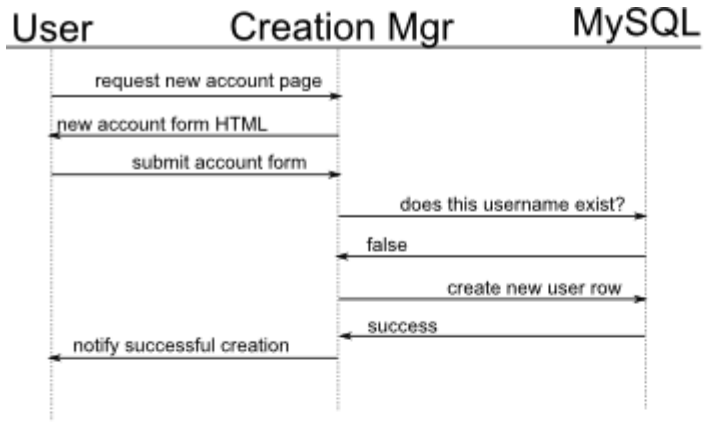
Interfaces

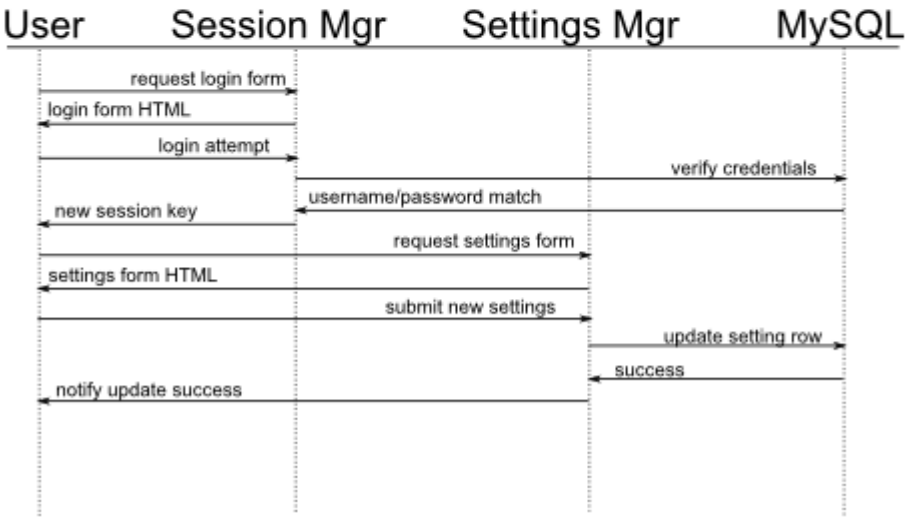
Interfaces are provided for account functionality and for other modules to access the database. Each other module has its own interface to the database. This makes it easier to maintain a low level of dependency between modules. It is also easy to modify the interface to match the needs of each module

- Interface A: Session Mgr - manages logging in and logging out.
- Interface B: Creation Mgr - manages creating a new account.
- Interface C: Settings Mgr - allows users to modify their information in the database
- Interface D: Char DB Mgr - stores and retrieves user messages from the database
- Interface E: Dr. DB Mgr - stores and retrieves data on pill and appointment schedules from the database
- Interface F: Calendar DB Mgr - reads the database to find relevant information for the calendar
- Interface G: Alert DB Mgr - reads the database to find and create new user alerts

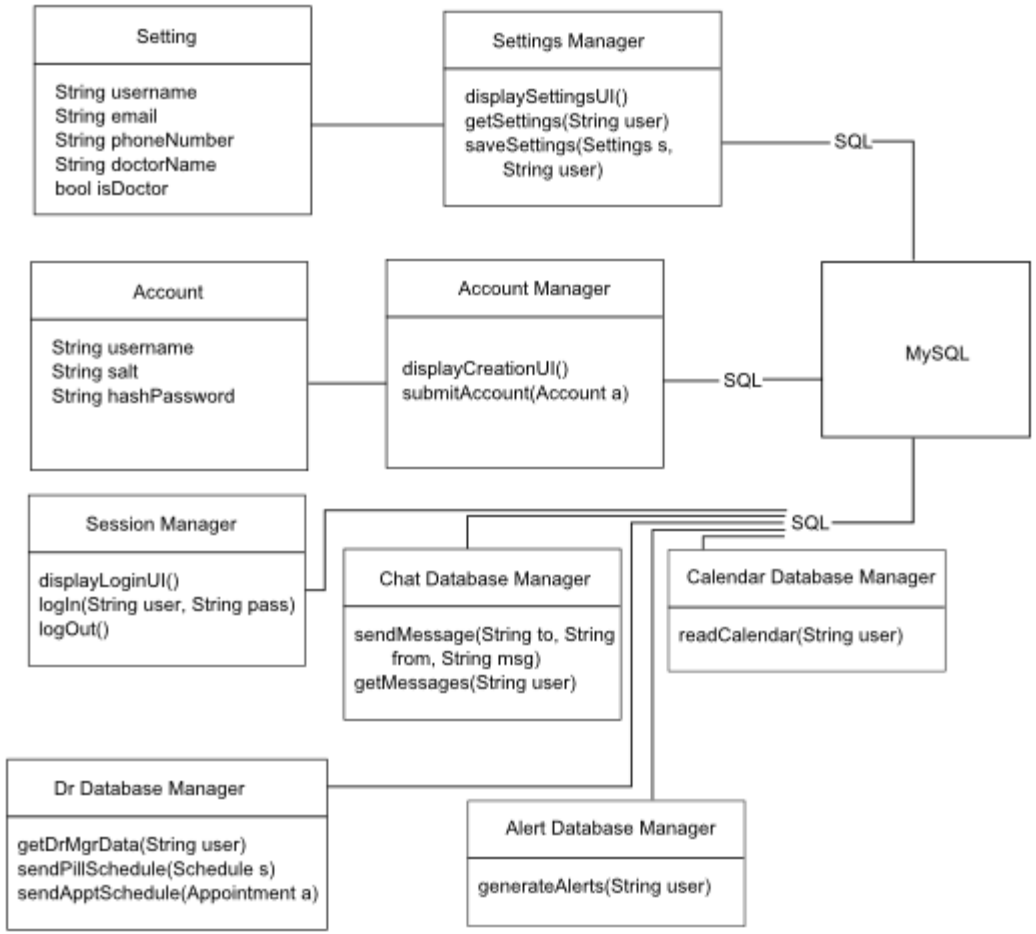
Event Sequence Diagrams

The first diagram represents creating an account. The second represents logging in and changing an account setting.


User
Creation Mgr
MySQL
request new account page
new account form HTML
submit account form
does this username exist?
false
create new user row
success
notify successful creation


User
Session Mgr
Settings Mgr
MySQL
request login form
login form HTML
login attempt
verify credentials
username/password match
new session key
request settings form
settings form HTML
submit new settings
update setting row
success
notify update success

Class Diagram



Trace of Requirements to Design:

Item Description	Detailed Design Section	Document of Origin	Sections of Origin
Overall functionality of Database Manager	Overview	Concept of Operations	Users and Modes of Operation, Operational Features
Log In	Event Sequence Diagrams	Software Requirements Specification	Use Case Descriptions
Crete Account	Event Sequence Diagrams	Software Requirements Specification	Event Table, Use Case Descriptions
Change account settings	Data, Operations	Software Requirements Specification	Use Case Descriptions
Secure user information	Design Issues	Software Requirements Specification	3.8

Template created by G. Walton (GWalton@mail.ucf.edu) on October 8, 1999 and last modified on August 15, 2000

This page last modified by James Luke on October 9, 2014

Online Health Monitoring System

Detailed Design: SMS Alert Manager

COP 4331, Fall, 2014

Modification history:

Version	Date	Who	Comment
v0.0	08/15/00	G. H. Walton	Template
v1.0	10/9/14	David Gundler	Filled Out

Team Name: Team 14

Team Members:

- Jon Carelli - email - web page
- Chris McCue - email - web page
- Chris Chaffin - email - web page
- Ethan Pitts - email - web page
- David Gundler - email - web page
- James Luke - email - web page

Contents of this Document

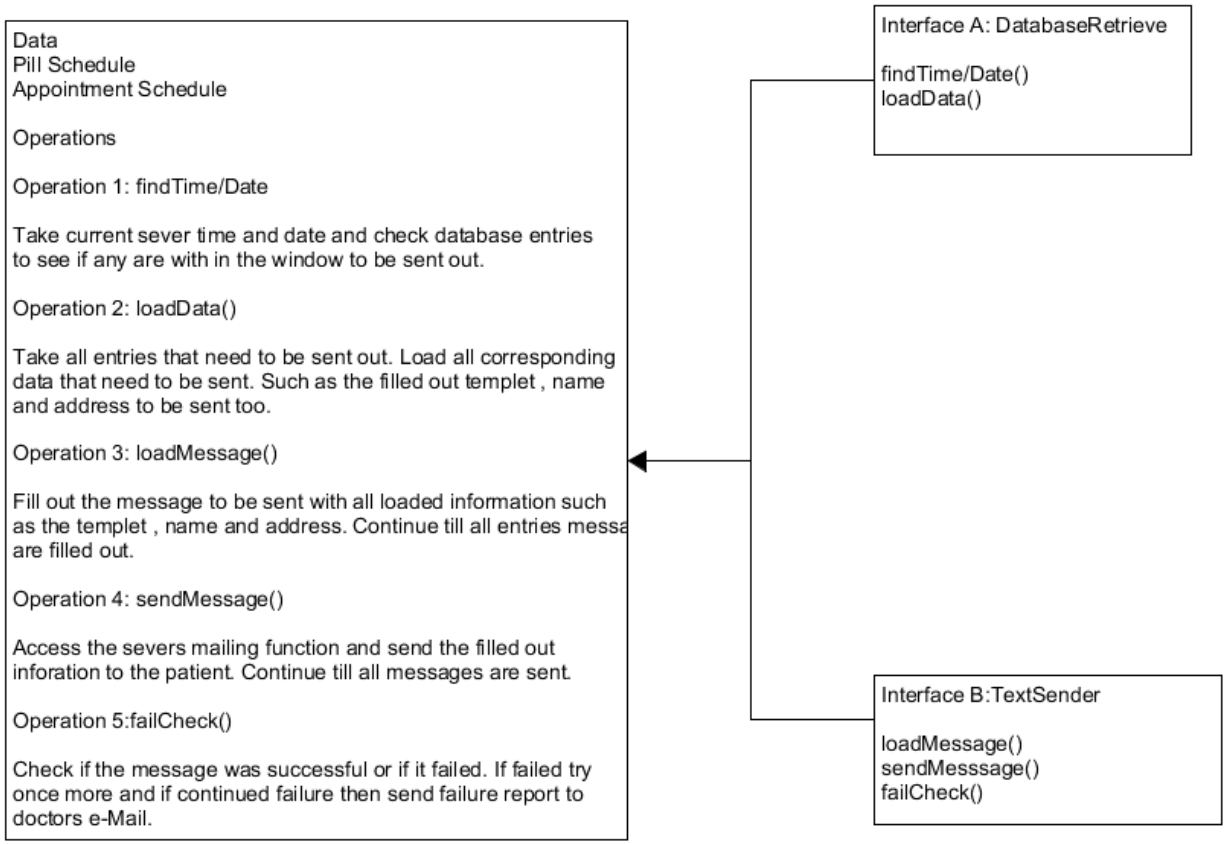
Design Issues:

Design issues to this module include: performance, reliability and, maintainability. The system will retrieve the pill schedule and appointment schedule information from the data base and relay that information to the patient using SMS. The system will access the information from the database, check the time, and send a message to the patient's phone when it is time to take medication or a reminder of an upcoming appointment.

The system must be thoroughly test as it is imperative that the correct recipient receive the right messages from the information stored in the database. The database must hold the necessary data and data must be correct. Data must be sent and received at the correct time. The system must be able to make sure the messages have been sent otherwise an error report will be produced and sent to the doctor's email address so this can be manually addressed.

Detailed Design Information:

SMS Overview



Data:

The information for the messages to be sent out can be broken down as follows

Date:
The dates that are need for taking pills wither once a day or every other day.
The dates use to remind the patient that morning or the day before of an appointment.

Time:
The time that a pill needs to be taken.

The time of a scheduled appointment.

Name:
The First and Last name of the patient.

Address:
The address need to be sent too
.
Template:
The preset form the message will take with the above items inserted with prescription and time or appointment time.

Template for taking pills

```
Dear "Last Name" "First Name"
You're Advil prescription needs to be taken by 1;00PM.

12:30 PM.
```

Template for Appointment Doctor

```
Dear "Last Name" "First Name"
You have an appointment with Doctor Example at 6:00PM 10/15/14

Sent at 12:00PM 10/14/14
```

Template for Appointment Hospital

```
Dear "Last Name" "First Name"
You have an appointment at Example Hospital at 6:00PM 10/15/14

Sent at 12:00PM 10/14/14
```

Data Storage Format:

Date	Time	Name	Address	template
2014:10:15	**13:00**	**patient(Last Name First Name)**	**patient(Cells E-mail Address)**	**The message to be sent.**

Operations:

Daily: Once a day the appointments for the next day will be sent out.

findDate/Time()

Find all of tomorrow's appointments in the database using the server's dates.

loadData()

Load all appointments that need to be sent out and the necessary information name , template and address.

loadMessage()

Fill out the appointments to be sent using the information gathered.

sendMessage()

Send all the appointments that need to go out.

failCheck()

Monitor all Sends. If a send fails then retry. If it fails aging then send failure report to doctors e-mail will Name of patients and message that was to be sent.

Example:

10/17/2014 Check all 10/18/2014

found:

patient 1

patient 4

load:

10/18/2014\ 12:00 PM \Patient One\ Cell address\appointment Doctor

10/18/2014\ 2:00 PM \Patient Four\ Cell address\appointment Hospital

Fill Out:

Dear "Patient" "One"
You have an appointment with Doctor Example at 12:00PM 10/18/14

Sent at 12:00PM 10/17/14

Dear "Patient" "Four"
You have an appointment at Example Hospital at 1:00PM 10/18/14

Sent at 12:00PM 10/17/14

Send:

To: patient One cell number
message

To: patient Four cell number
message

Fail:
No failures in patient One message.
No failures in patient Four message.

Delay: On a delay timer pills and appointments.

findDate/Time()

Find all of message appointments in the database using the server's dates.

loadData()

Load all message that need to be sent out and the necessary information name , template and address.

loadMessage()

Fill out the message to be sent using the information gathered.

sendMessage()

Send all the messages that need to go out.

failCheck()

Monitor all Sends. If a send fails then retry. If it fails aging then send failure report to doctors e-mail will Name of patients and message that was to be sent.

Example:

10/18/2014 11:45AM Check all 10/18/2014 15min window

found:

patient one
patient three
load:

10/18/2014\ 12:00 PM \Patient One\ Cell address\appointment Doctor
10/18/2014\ 11:50 AM \Patient Three\ Cell address\appointment Pill

Fill Out:

Dear "Patient" "One"
You have an appointment with Doctor Example at 12:00PM 10/18/14

Sent at 11:45AM 10/18/14

Dear "Patient" "Three"
You're Advil prescription needs to be taken by 11:50 AM.

Sent at 11:45AM 10/18/14

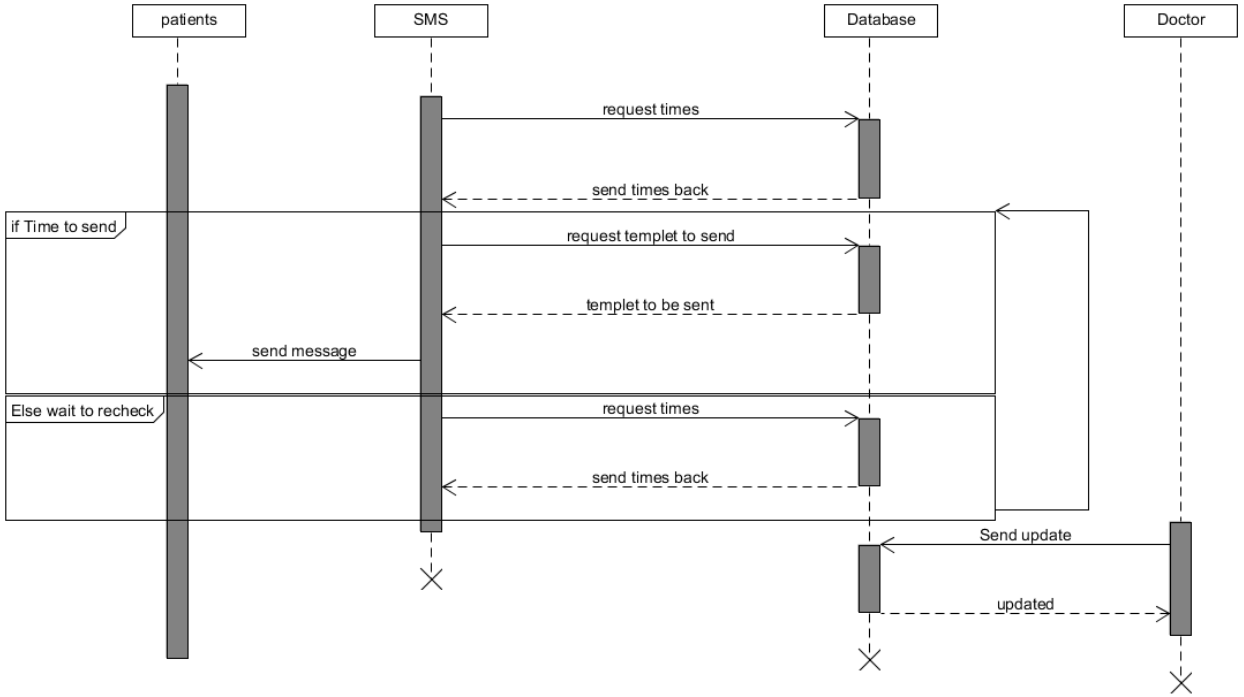
Send:

To: patient One cell number
message

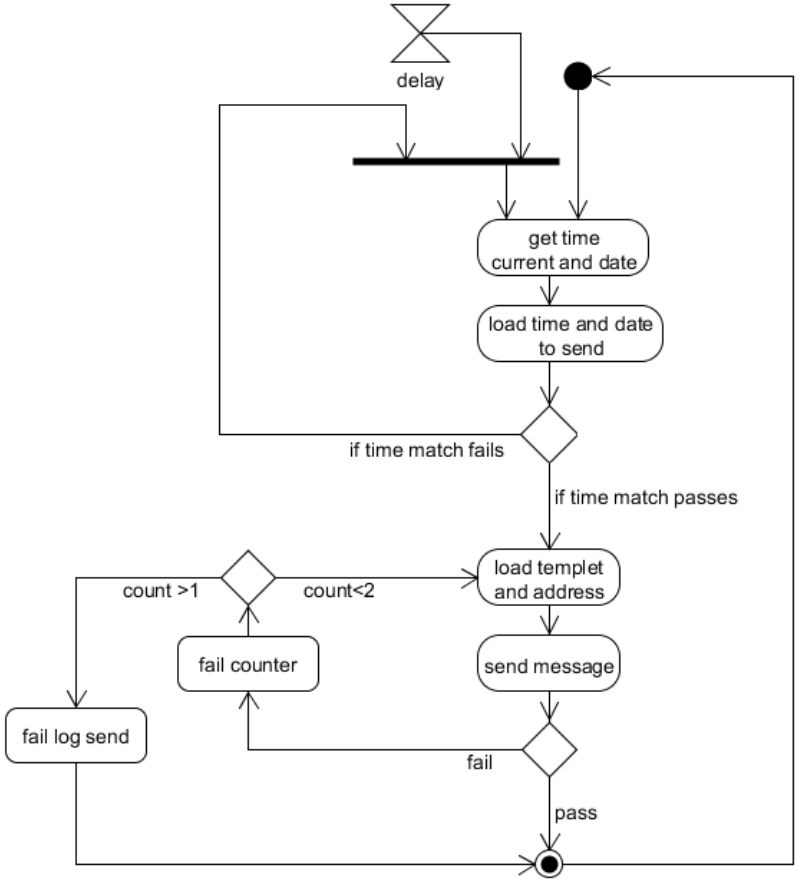
To: patient Three cell number
message

Fail:
No failures in patient One message.
No failures in patient Three message.

Sequence Diagram:



Activity Diagram:



Trace of Requirements to Design:

Item Description	Detailed Design Section	Document of Origin	Sections of Origin
Overall functionality of Pills/Appointments /Messaging	Overview	Concept of Operations	Operational Scenarios, Operational Features
Tools	Overview, Operations	Software Requirements Specification	3.7
SMS messaging	Data, Operations	Software Requirements Specification	3.1
Database access/Data format	Overview, Data, Operations	Software Requirements Specification	3.6

Template created by G. Walton (GWalton@mail.ucf.edu) on October 8, 1999 and last modified on August 15, 2000

This page last modified by David Gundler (d_gundler@knights.ucf.edu) on 10/7/2014

Online Health Monitoring System

Detailed Design: System Navigation

COP 4331, Fall 2014

Modification history:

Version	Date	Who	Comment
v0.0	08/15/00	G. H. Walton	Template
v1.0	10/07/14	Ethan Pitts	First Revision

Team Name: Team 14

Team Members:

- Jon Carelli - email - web page
- Chris McCue - email - web page
- Chris Chaffin - email - web page
- Ethan Pitts - email - web page
- David Gundler - email - web page
- James Luke - email - web page

Contents of this Document

Design Issues:

The primary concerns in this module are usability and reliability. The menu bar is a core system functionality, which serves a dual purpose of providing quick and simple navigation between major features of the system, as well as a context for the user's current screen.

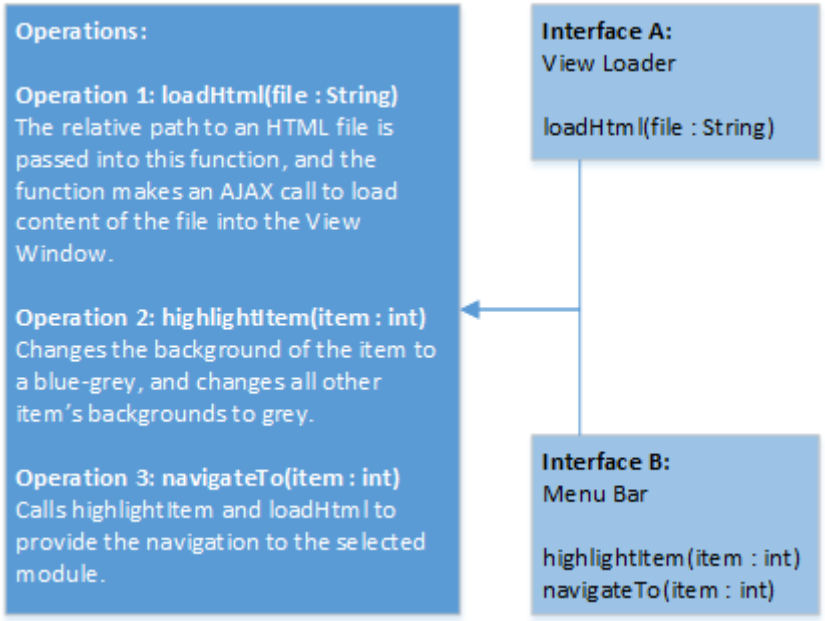
We believe that the menu bar is the optimal way of navigating the system, because it provides separation of the unrelated modules of the system without sacrificing accessibility of features. The menu bar is static on the left side of the screen, and each box in the menu bar will load a different module in the main viewer, which lies just to the right of the menu, and covers most of the screen. To navigate to a new module, the user clicks on one of the menu's boxes, each of which contain an icon, which represents the associated module.

It is also necessary for the user to have a visual indication on the menu bar, which allows them to know which module is currently selected. We have chosen do to this by highlighting the selected menu item by changing the background of the box from grey to a blue-grey.

The menu bar must also be reliable, and free of glitches. It is paramount that the correct modules be loaded when the menu item's box is clicked, since a failure to load the module means that that particular feature will be impossible to access. Given the nature of the system, that could result in injury or even death if a doctor cannot contact his/her patient with vital information.

Detailed Design Information:

System Navigation Overview:

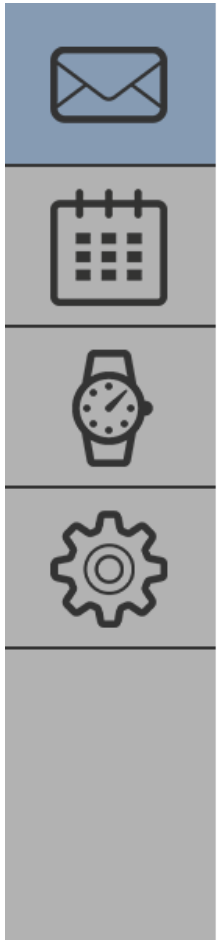


The design of the System Navigation module is driven by functional requirements. Per the requirements, the user will be able to quickly and easily access any of the available modules using the Menu Bar, and view the module in the View Loader.

User Interface Visualization:

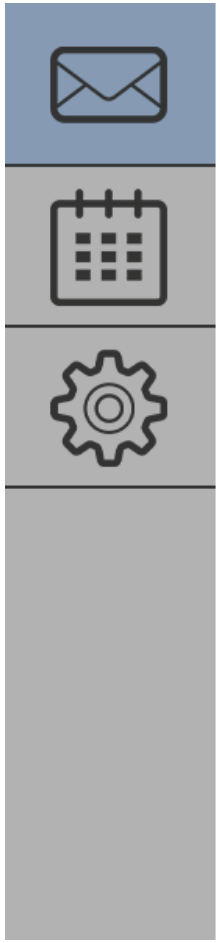
At this point, it is helpful to visualize the functionality-driven user interface.

Doctor Mode:



Messages Module Content

Patient Mode:



Messages Module Content

Operations:

See “System Navigation Overview” section for operations and descriptions.

There are two main operations that need to be noted. These are navigateTo(item : int) and loadHtml(file : String).

The loadHtml operation takes the relative file path to an HTML file, and loads that file in the View Window. This allows us to have a separate HTML file for each module, which in turn simplifies the development process.

The navigateTo operation is called when a selection is made on the Menu Bar. The item's position number that was selected is passed to the function. The menu item clicked should then become "active," which is shown by changing the background color to a grey-blue, and changing all other menu item's background color to grey. The loadHtml operation should then be called to load the corresponding HTML file into the View Window.

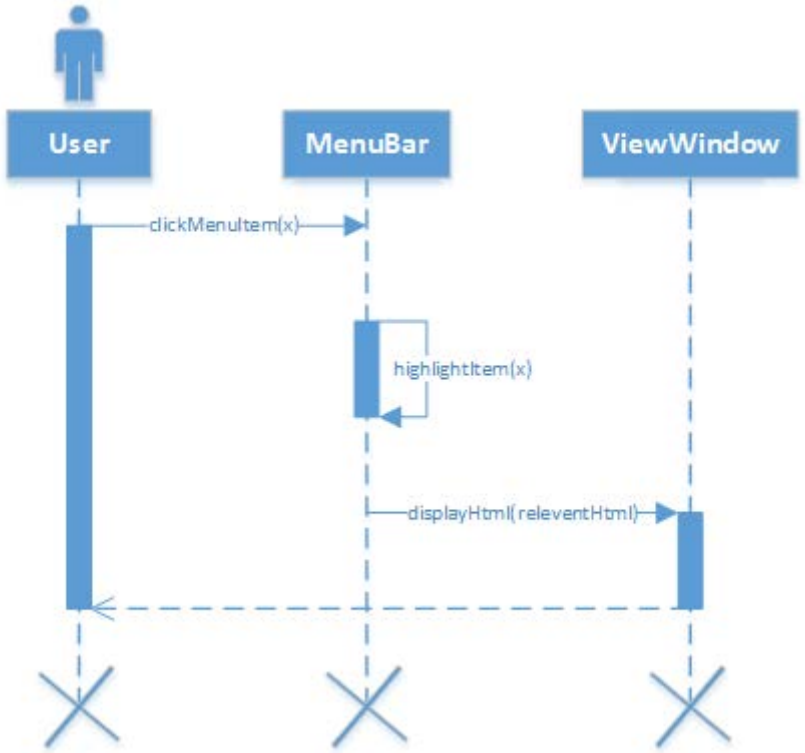
Interfaces:

Interface A: View Loader - The main area to the right of the Menu Bar, where the different modules are loaded.

Interface B: Menu Bar - The navigational entity which selects and displays which module is being used.

Event Sequence Diagram:

The following sequence diagram is a general sequence for any menu item 'x.'



Class Diagram:

Trace of Requirements to Design:

Item Description	Detailed Design Section	Document of Origin	Sections of Origin
Patient Interface Navigation	Specific Requirements	Software Requirements Specification	3.2 No. 10
Physician Interface Navigation	Specific Requirements	Software Requirements Specification	3.2 No. 9

Template created by G. Walton (GWalton@mail.ucf.edu) on October 8, 1999

This page last modified by Ethan Pitts (ethanempe@gmail.com) on October 9, 2014

Online Health Monitoring System

Detailed Design: Chat Manager

COP 4331, Fall, 2014

Modification history:

Version	Date	Who	Comment
v0.0	08/15/00	G. H. Walton	Template
v1.0	10/3/14	Jon Carelli	First draft, Title, Team, Design issues
v1.1	10/4/14	Jon Carelli	Added overview diagram, data
v1.2	10/6/14	Jon Carelli	UI Visualization, Operation, Algorithm
v1.3	10/7/14	Jon Carelli	Sequence Diagram, Tracing, Class Diagram

Team Name: Team 14

Team Members:

- Jon Carelli - email - web page
- Chris McCue - email - web page
- Chris Chaffin - email - web page
- Ethan Pitts - email - web page
- David Gundler - email - web page
- James Luke - email - web page

Contents of this Document

Design Issues:

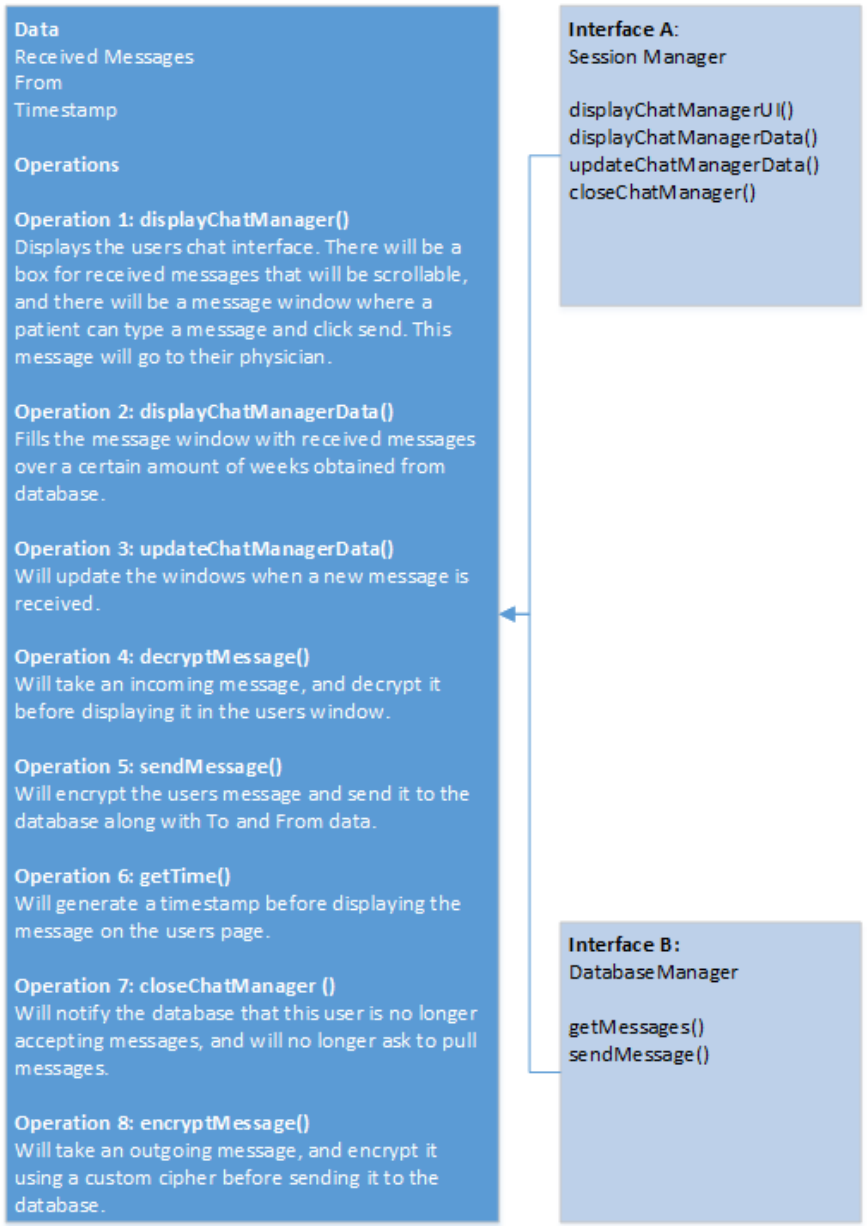
In this module, priority comes down to usability, reliability and security. The chat manager will handle incoming messages and send outgoing messages to the database. Once the database has a new message for another user, the chat manager will update that user's chat window with the supplied message. Currently, the system will only allow a patient to chat with their scheduled physician. The physician may chat with multiple different users concurrently. From the main website, a patient or physician can click the envelope icon to access the chat manager for that user. This means the interface will be clear, as an envelope is the universal symbol for a message. Once the user has accessed the chat manager interface, they may begin sending and receiving messages. Since this is an online application it must be portable. HTML and JavaScript will ensure this is not a problem. From a user perspective, a user should be able to understand how to use this module in less than 5 minutes. The user interface visualization demonstrates that the user controls are configured in a standard format that users would already be familiar with.

This module will be highly influenced by the database, so we must make sure the integration of the two is easy and quick. Security is also an important risk. We don't want patient's information being exposed to anybody but the patient. Encryption will be needed in some form running on the backend to encrypt and decrypt each patient/doctor message. Ensuring that the system isn't compromised is key in creating a system that is reliable and secure.

When a user sends the message, the message will be encrypted, To, From, and the Message will be sent to the database, the database will then send the message to the corresponding users message manager, a timestamp will be generated, and the information will be displayed on their page.

Detailed Design Information:

Chat Manager Overview:



The design of the Chat Manager module is driven by functional requirements. Per the requirements, the physician may only send their physician a message, but the physician can only send messages to his patients.

Data:

Messages can be broken down to as follows:

Message:
To: Dr. Example
From: Jon Carelli
Message: What is going on im not very good at computer

Time Related Attributes (generated by chat manager)
Date: 10/4/2014
Time: 11:31 AM

Encryption of data
Message: What is going on im not very good at computer
Encrypted: edwae wt dwads dw wg hfj dasc diid ks csotmvz

The message data is as follows:

Data: Categories		
Message	**Time Related Attributes**	**Encryption of Data**
toUserName	dateData	messageString
fromUserName	timeData	messageEncryptedString
messageString		

Data Storage Format:

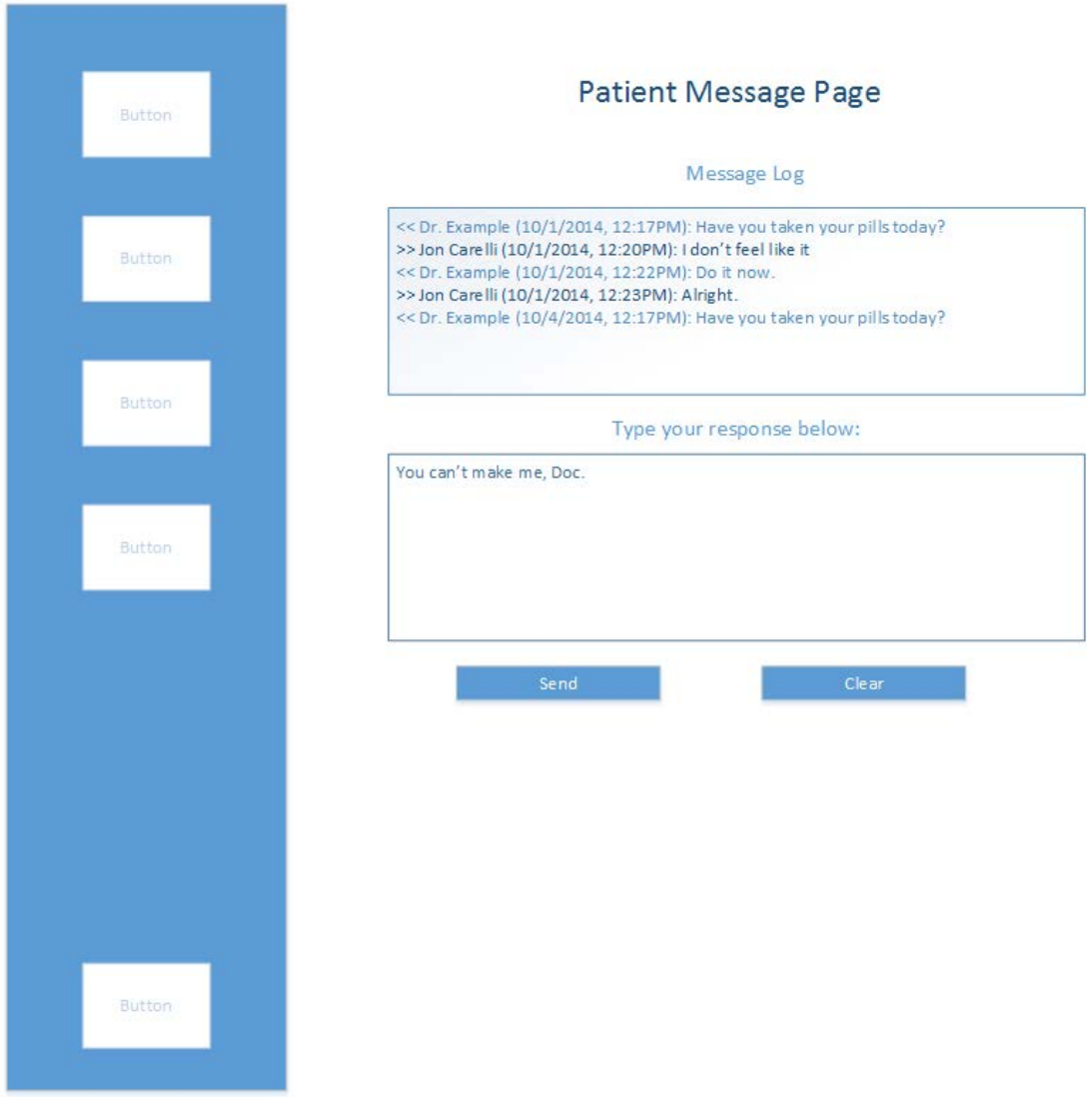
For the example data described above, the data storage format will be as follows:

```
Jon Carelli       Dr. Example                        What is going on im not very good at computer
2014:10:04        11:54:AM
What is going on im not very good at computer        edwae wt dwads dw wg hfj dasc diid ks csotmvz
```

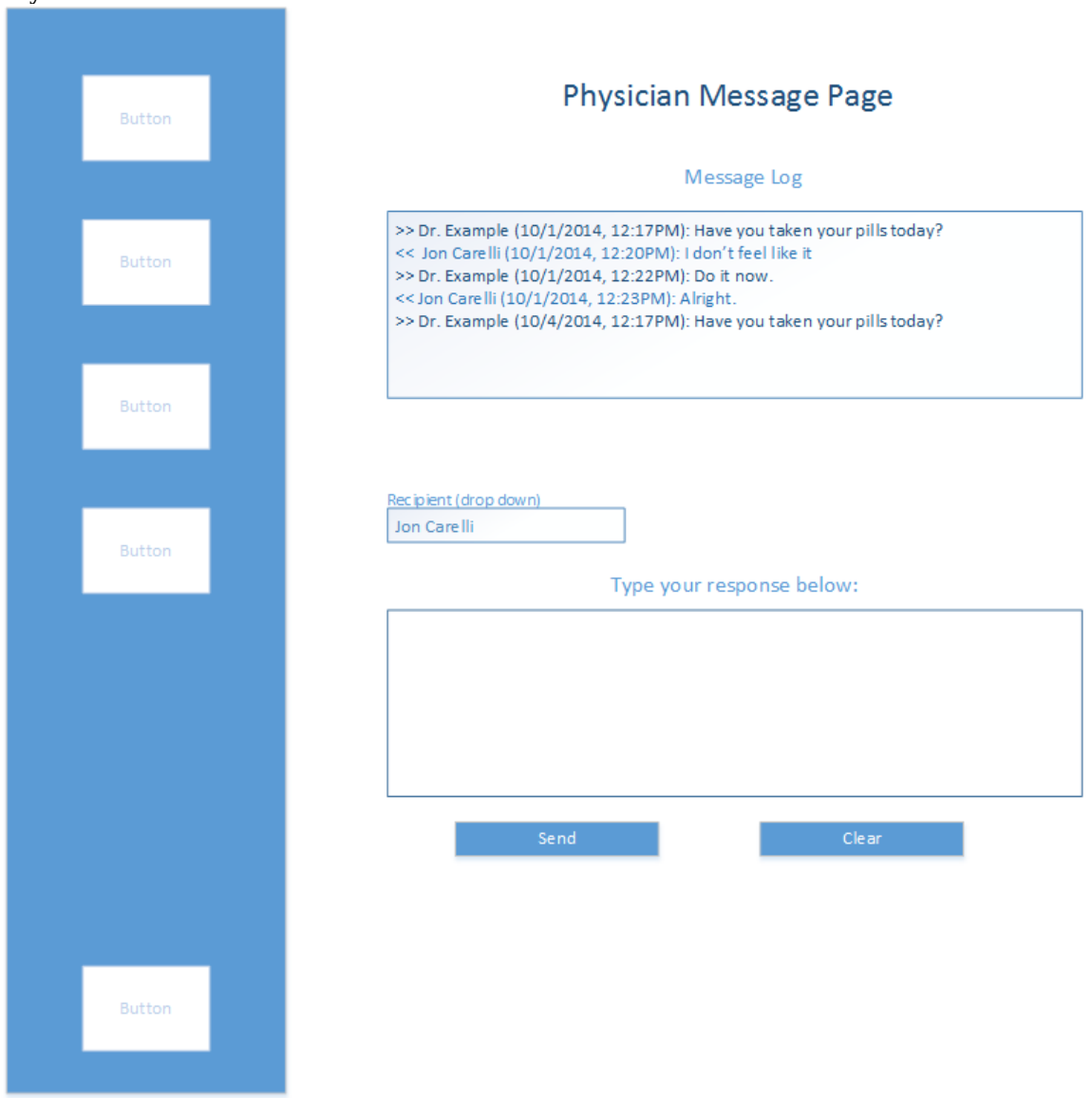
The top row contains message data, From, To, Unencrypted String. The second row contains date related data. The last row contains encryption data. This is the data storage format for sending and receiving messages between users. Time and date will be calculated by the Chat Manager, once calculated they will be appended to the message string and shown on the page. The database manager should not need to calculate this data.

User Interface Visualization:

Patient:



Physician:


Button
Button
Button
Button
Button
Physician Message Page
Message Log
>> Dr. Example (10/1/2014, 12:17PM): Have you taken your pills today?
<< Jon Carelli (10/1/2014, 12:20PM): I don't feel like it
>> Dr. Example (10/1/2014, 12:22PM): Do it now.
<< Jon Carelli (10/1/2014, 12:23PM): Alright.
>> Dr. Example (10/4/2014, 12:17PM): Have you taken your pills today?
Recipient (drop down)
Jon Carelli
Type your response below:
Send
Clear

Operations:

See above diagram for operations and descriptions.
There are a few operations that the Chat Manager will be performing. It will both send and receive messages, with encryption and decryption support. The Chat Manager will also be responsible for generating timestamps and appending them to the message string before showing them on the user's page. This means the Chat Manager will be sending To, From, and encrypted message strings to the database.

Events Generated (example) for sending and receiving messages:

```
Jon Carelli            Dr. Example      dwae wt dwads dw wg hfj dasc diid ks csotmvz
Dr. Example            Jon Carelli      saddsa sad ghhghjdg bbc beet
Jon Carelli            Dr. Example      fdfd qwt 34 zc fqr jjewx
Dr. Example            Jon Carelli      sfs few tuif vs
```

This could be considered four inputs to the database, the string of garbled text represents encrypted text. There should never be unencrypted text being transferred to and from the database, that a packet sniffer or hacker could intercept.

Events Generated (example) for encrypting and decrypting messages:

```
Encrypt: dwae wt dwads dw wg hfj dasc diid ks csotmvz
Decrypt: What is going on im not very good at computer
Encrypt: asdadas gdgdj df h ew dfhd asgxg
Decrypt: Quit hassling me Doc.
```

Events Generated for appending date and time to the string.

```
Jon Carelli            Dr. Example      Quit hassling me Doc.
getDateAndTime(); → 10/6/2014, 12:46PM
Appending: (10/6/2014, 12:46PM): Quit hassling me Doc.
Encrypt (String message)
SendMessage(String To, String From, String message)
```

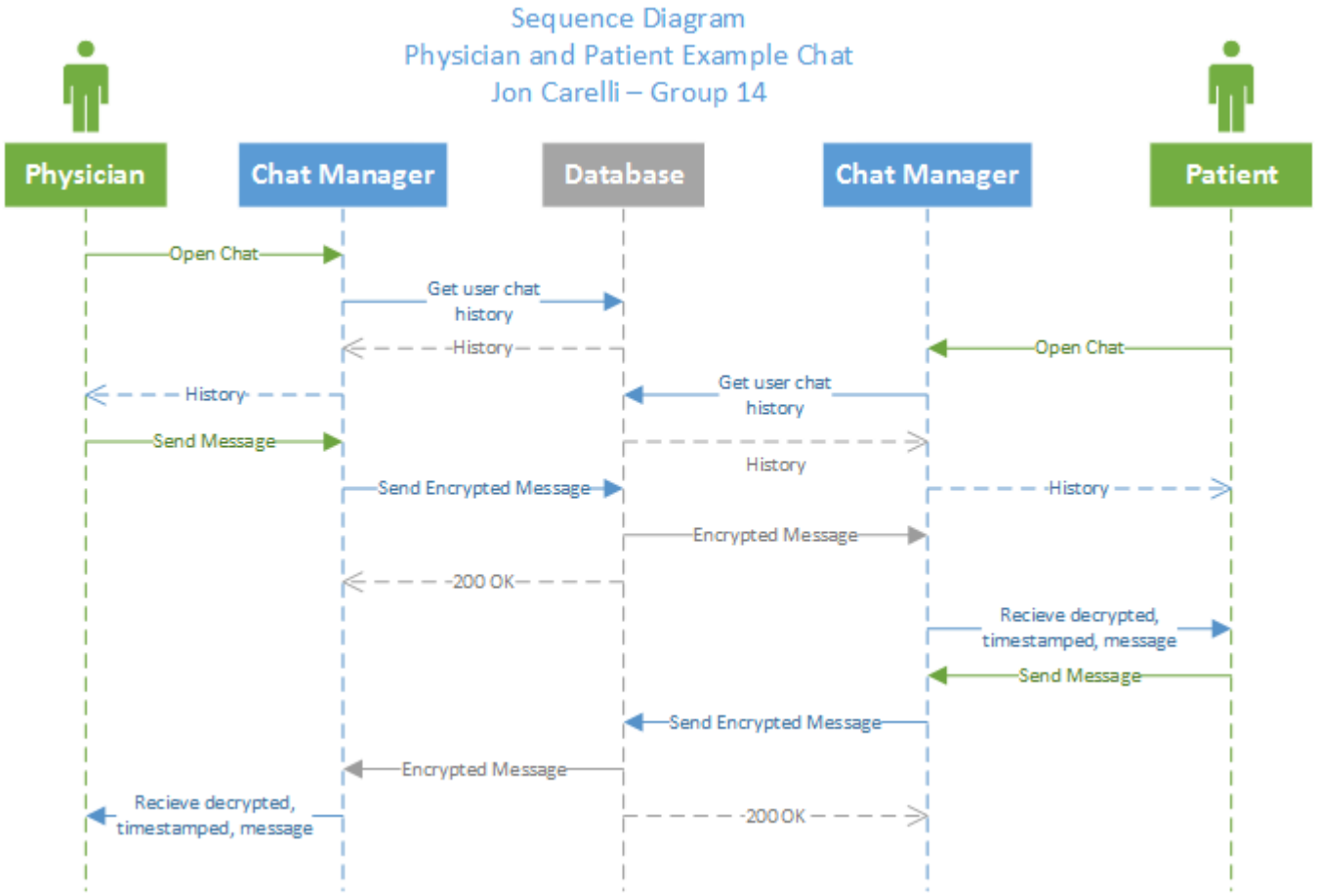
Algorithms:

There would need to be an efficient encryption algorithm, possibly an AES/DES approach. For this project, nothing too sophisticated needs to be implemented, could do a simple Caesar Cipher.

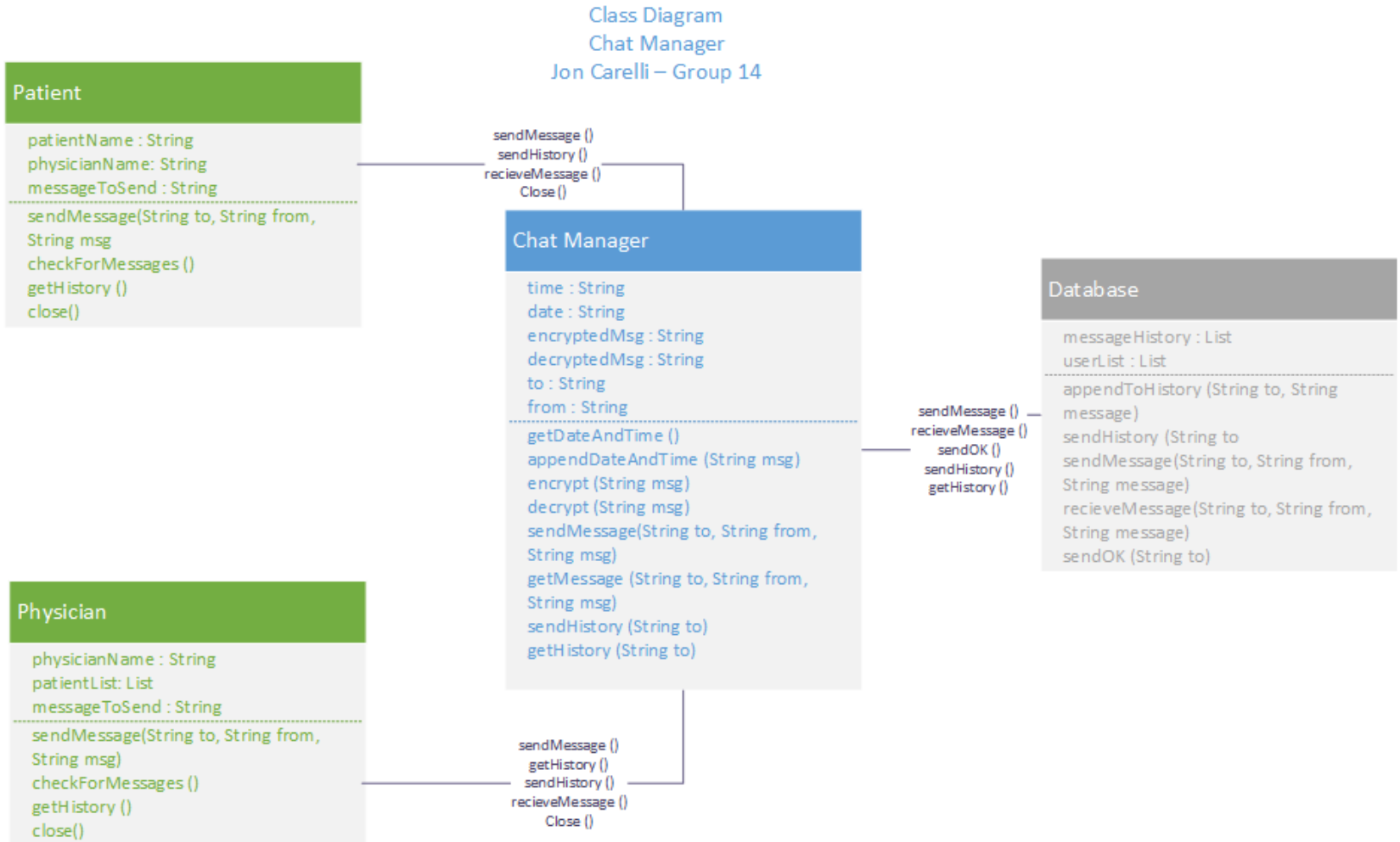
Encryption: $E_n(x) = (x + n) \mod 26.$

Decryption: $D_n(x) = (x - n) \mod 26.$

Sequence Diagram:



Class Diagram:



Trace of Requirements to Design:

Item Description	Detailed Design Section	Document of Origin	Sections of Origin
Overall functionality of Chat Manager	Overview	Concept of Operations	Users and Modes of Operation, Operational Features
Send Message	Event Sequence Diagrams	Software Requirements Specification	Event Table, Use Case Descriptions
Receive Message	Event Sequence Diagrams	Software Requirements Specification	Event Table, Use Case Descriptions
Encrypt Messages	Data, Operations	Software Requirements Specification	3.8
Decrypt Messages	Data, Operations	Software Requirements Specification	3.8
Appointment data specifications	Data, Operations	Software Requirements Specification	3.2
Example data and Data format	Data, Operations	Software Requirements Specification	3.6
Quality attributes	Design Issues	Software Requirements Specification	3.9

Online Health Monitoring System

Detailed Design: Calendar

COP 4331, Fall, 2014

Modification history:

Version	Date	Who	Comment
v0.0	08/15/00	G. H. Walton	Template
v1.0	10/9/14	Chris Chaffin	First Revision

Team Name: Team 14

Team Members:

- Jon Carelli - email - web page
- Chris McCue - email - web page
- Chris Chaffin - email - web page
- Ethan Pitts - email - web page
- David Gundler - email - web page
- James Luke - email - web page

Contents of this Document

Design Issues:

The calendar contains all of the times and dates associated with patient's medicine and appointments with their physician. The interface will support Google's plug in API calendar. This choice was made to reduce development time and to use a familiar calendar scheme that is currently used by millions. Furthermore, this design decision ensures portability.

This module interacts with the database upon selecting the calendar. As with the other modules it is imperative that the load times are limited to three seconds to avoid frustration. The load times for the calendar will occur each time a new month or week is selected. In order to keep the load times less than three seconds communication between the database and the calendar must be efficient.

The design of the calendar will not allow patients to add any events to the calendar. The calendar will only be able to be managed by the doctor to avoid any accidental dosages taken by the patient. This is of the utmost importance as failure to adhere to the calendar could result in severe sickness or death.

Once the prototype is developed it will be integrated with the Database Manager. The biggest risk lies in this step. The calendar is completely dependant on communication with the database, without it the calendar has no functionality. I will have to work closely with the team to eliminate this risk.

Detailed Design Information:

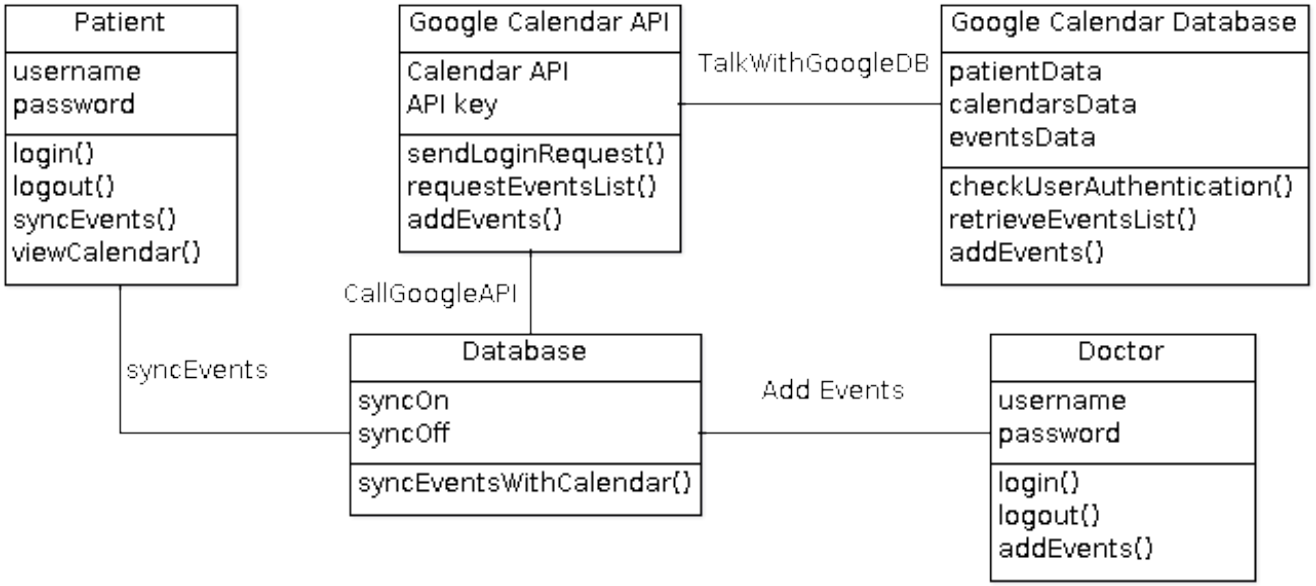
Classes

Patient – In the event of a successful login the patient view the doctor's calendar or sync their own with the doctors, which is stored in the Database

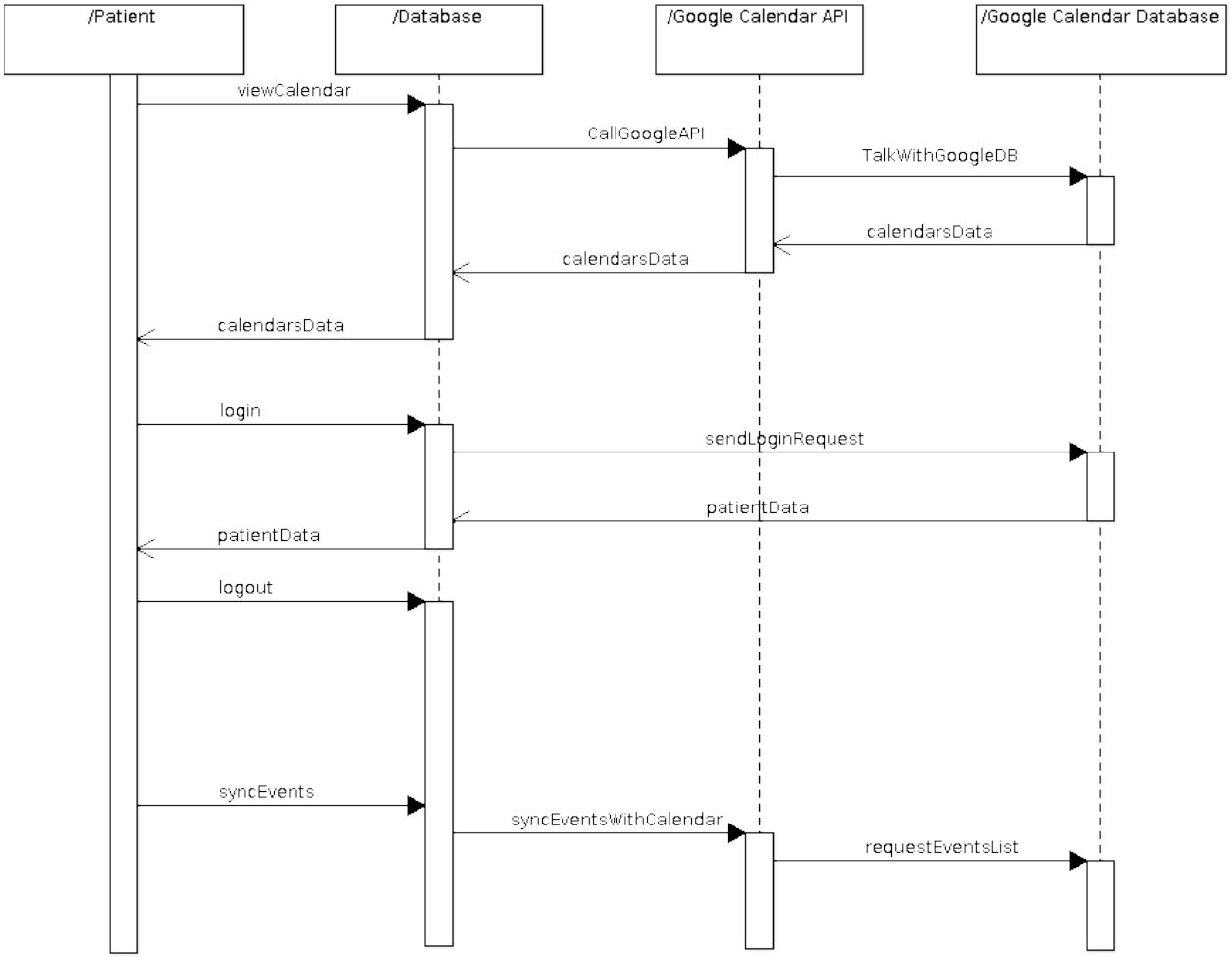
Doctor – The doctor can add events, such as an appointment, and save it to the database for a patient to view.

Database – The database calls the Google Calendar API, which communicates with the Google Calendar Database

Class Diagram



Sequence Diagram



Trace of Requirements to Design:

Item Description	Detailed Design Section	Document of Origin	Sections of Origin
Overall functionality of Calendar	Detailed Design Information	Users of Modes and Operation	Concept of Operations
Add Event	Event Sequence Diagram	Software Requirements Specification	3.4
View Calendar	Event Sequence Diagram	Software Requirements Specification	3.5

Template created by G. Walton (GWalton@mail.ucf.edu) on October 8, 1999 and last modified on August 15, 2000

This page last modified by Chris Chaffin (chris_chaffin4488@yahoo.com) on 10/9/2014

Online Health Monitoring System

Detailed Design: Pill/Appointment Manager

COP 4331, Fall, 2014

Modification history:

Version	Date	Who	Comment
v0.0	08/15/00	G. H. Walton	Template
v1.0	10/03/14	C. N. McCue	First Revision

Team Name: Team 14

Team Members:

- Jon Carelli - email - web page
- Chris McCue - email - web page
- Chris Chaffin - email - web page
- Ethan Pitts - email - web page
- David Gundler - email - web page
- James Luke - email - web page

Contents of this Document

Design Issues:

Relevant design issues to this module include safety, testability, ease-of-use, portability, performance and maintainability. Since this is a health-related product, failure could result in harm or death to the user. Particularly, for this module, pill schedules can become a life and death issue. Buttons and selections in the user interface must be very clear so that users will not be confused on what they are selecting. Therefore, it is important that this is tested well. Key to this module is the Doctor Manager to Database Manager interface. The appropriate data must be stored in the database properly. This will be a key area in the Test Plan. Somewhere, a disclaimer must be posted.

Currently, we are placing the pill scheduling and appointment management in one place. An alternative is to separate these into two tabs on the Session Manager. This may make the user interface easier to use. This alternative will be weighed during the prototyping stage.

Since this is an online application it must be portable. HTML and JavaScript will ensure this is not a problem. From a user perspective, a user should be able to understand how to use this module in less than 5 minutes. The user interface visualization demonstrates that the user controls are configured in a standard format that users would already be familiar with.

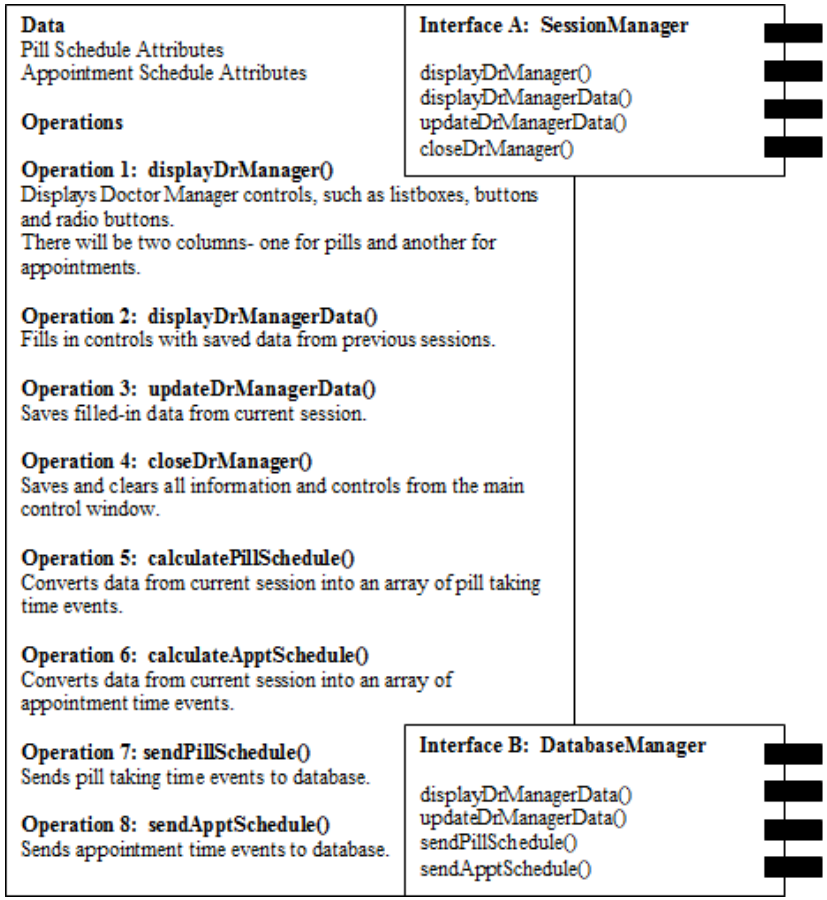
Since this module interfaces with the database, it is important that storage and retrieval of information occurs within 3 seconds. Otherwise, users will become frustrated with the poor performance. The pill and appointment data sent and retrieved from the database must be perpetually maintained.

From the detailed design of this module a prototype will be coded which stands alone. There will be no interface with the Session Manager. There will be no interface with the Database Manager. Instead data files will be used to simulate data storage and retrieval.

After the prototype is developed for this module, it will then be integrated with the Database Manager and Session Manager. This is where the majority of the risk lies. I am depending on James' module to provide a template to follow for database storage and retrieval. I will need to work with James to determine the optimal data format for storage and retrieval. It is anticipated that the integration stage will consist of close teamwork.

Detailed Design Information:

Doctor Manager Overview:



The design of the Doctor Manager module is driven by functional requirements. Per the requirements, the doctor will manage pill schedules and patient appointments.

Data:

Pill schedules can be broken down as follows:

Date Related Attributes

Start: for example, December 3, 2014

Duration: for example, 7 days

Frequency: for example, daily or every other day

Time Related Attributes

Start: for example, 7:30 a.m.

Frequency: for example, 3 times/day OR Interval: every 6 hours

End: for example, 10:00 p.m.

Pills Related Attributes

Pill name: Aspirin

Number to take: 3 pills at a time

Pills left: 15 pills

Hence, the pill data attributes are as follows:

Data: Categories of Pill Schedule Attributes		
Date Related	**Time Related**	**Pill Related**
dateStart	timeStart	pillName
dateDuration	timeFrequency	numberPillsToTake
dateFrequency	timeEnd	pillsLeft
	timeInterval	

In a similar fashion, a table is created for appointment data:

Data: Appointment Schedule Attributes
appointmentDate
appointmentTime
recurringFrequency
endRecurrence

Data Storage Format:

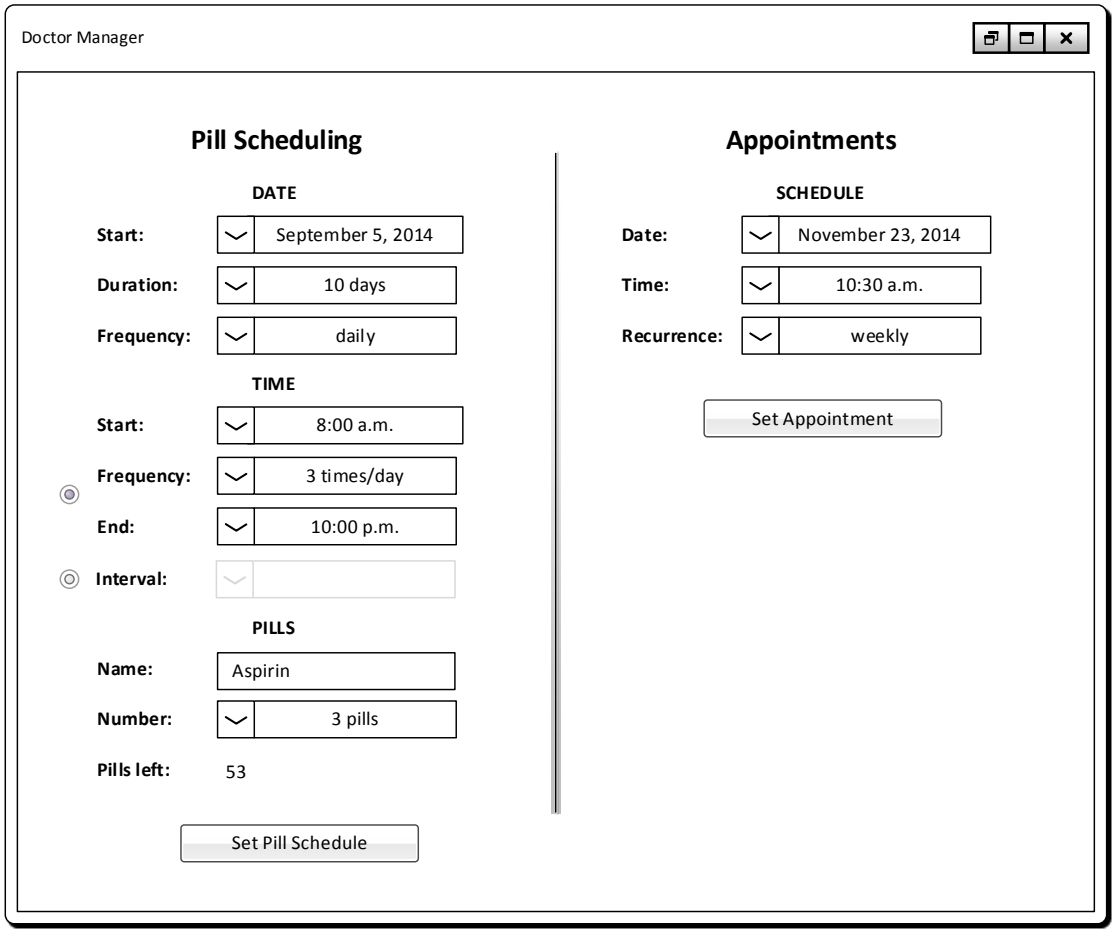
For the example data described above, the data storage format will be as follows:

```
Aspirin          3     15
2014:12:3        7     1
7:30             3     22:00
```

The top row contains pill related data. The second row contains date related data. The last row contains time related data. This is the data storage format for updating and refreshing the Doctor Manager display. Data which will be displayed in the calendar or data for pill alerts will be stored in a different format. The database manager should not need to calculate this data. The doctor manager module will calculate the data and the database will store the data.

User Interface Visualization:

At this point, it is helpful to visualize the functionality-driven user interface.



Operations:

See “Doctor Manager Overview” section for operations and descriptions.

Two operations worth noting are calculatePillSchedule() and calculateApptSchedule(). The pill schedule corresponds with SMS pill alerts. The database must store times of events instead of calculating these events. Therefore, the doctor manager must calculate an array of events to send to the database.

Using our aspirin example, the events generated are listed below:

```
Aspirin          3          15
2014:12:3        7:30
2014:12:3        15:00
2014:12:3        22:00
2014:12:4        7:30
2014:12:4        15:00
2014:12:4        22:00
2014:12:5        7:30
2014:12:5        15:00
2014:12:5        22:00
2014:12:6        7:30
2014:12:6        15:00
2014:12:6        22:00
2014:12:7        7:30
2014:12:7        15:00
2014:12:7        22:00
```

```
2014:12:8          7:30
2014:12:8          15:00
2014:12:8          22:00
2014:12:9          7:30
2014:12:9          15:00
2014:12:9          22:00
```

Similar events would be generated for appointment scheduling.

Calculation Algorithm:

```
for(dateStart; dateStart + dateDuration; dateFrequency++)
        for(timeStart; timeEnd; (timeEnd-timeStart)/(timeFrequency - 1))
                sendEventDateTime();
```

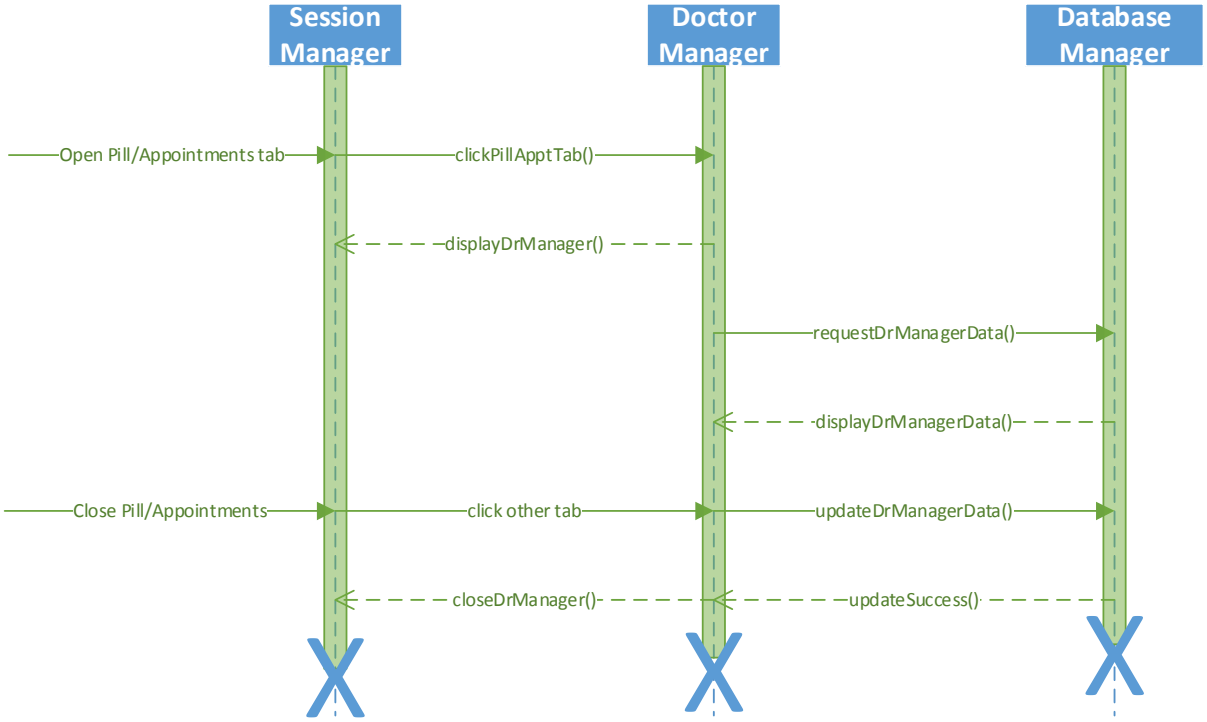
Interfaces:

Interface A: Session Manager- The main interface which uses tabs to determine what is displayed in the main window.

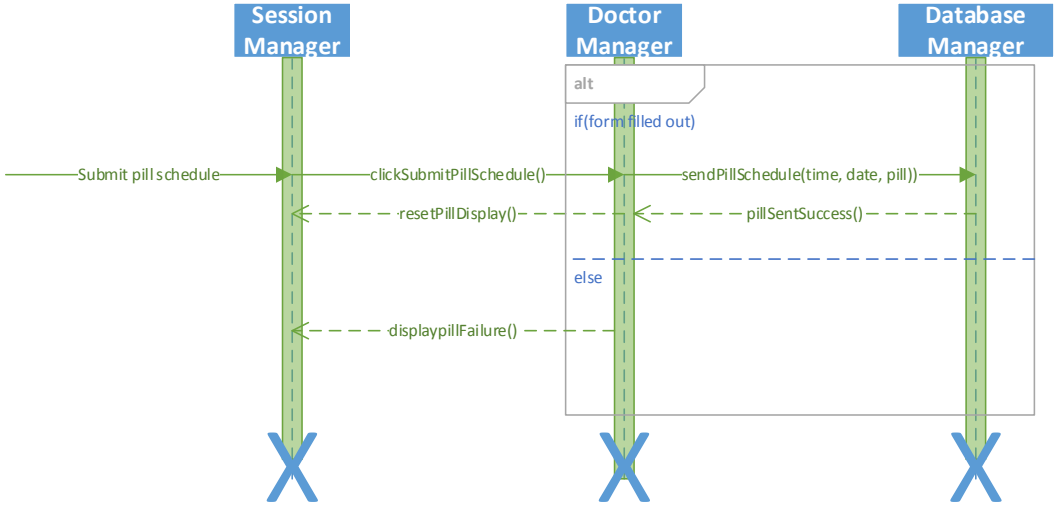
Interface B: Database Manager- The database manager manages the storage and retrieval of data. This follows the client-server format. To see how these interfaces interact with the Doctor Manager see “Event Sequence Diagrams” below.

Event Sequence Diagrams:

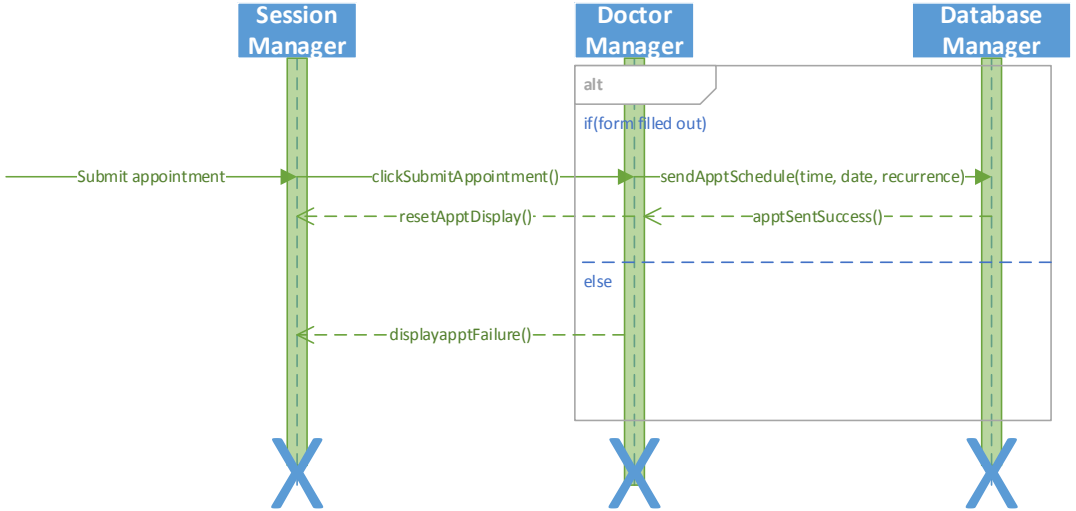
The functionality of the Doctor Manager is driven by events, such as a button click. Because of this, event sequence diagrams are helpful to understand user interaction. First, there is interaction between the Session Manager and the Doctor Manager. This opens and closes the Doctor Manager interface.



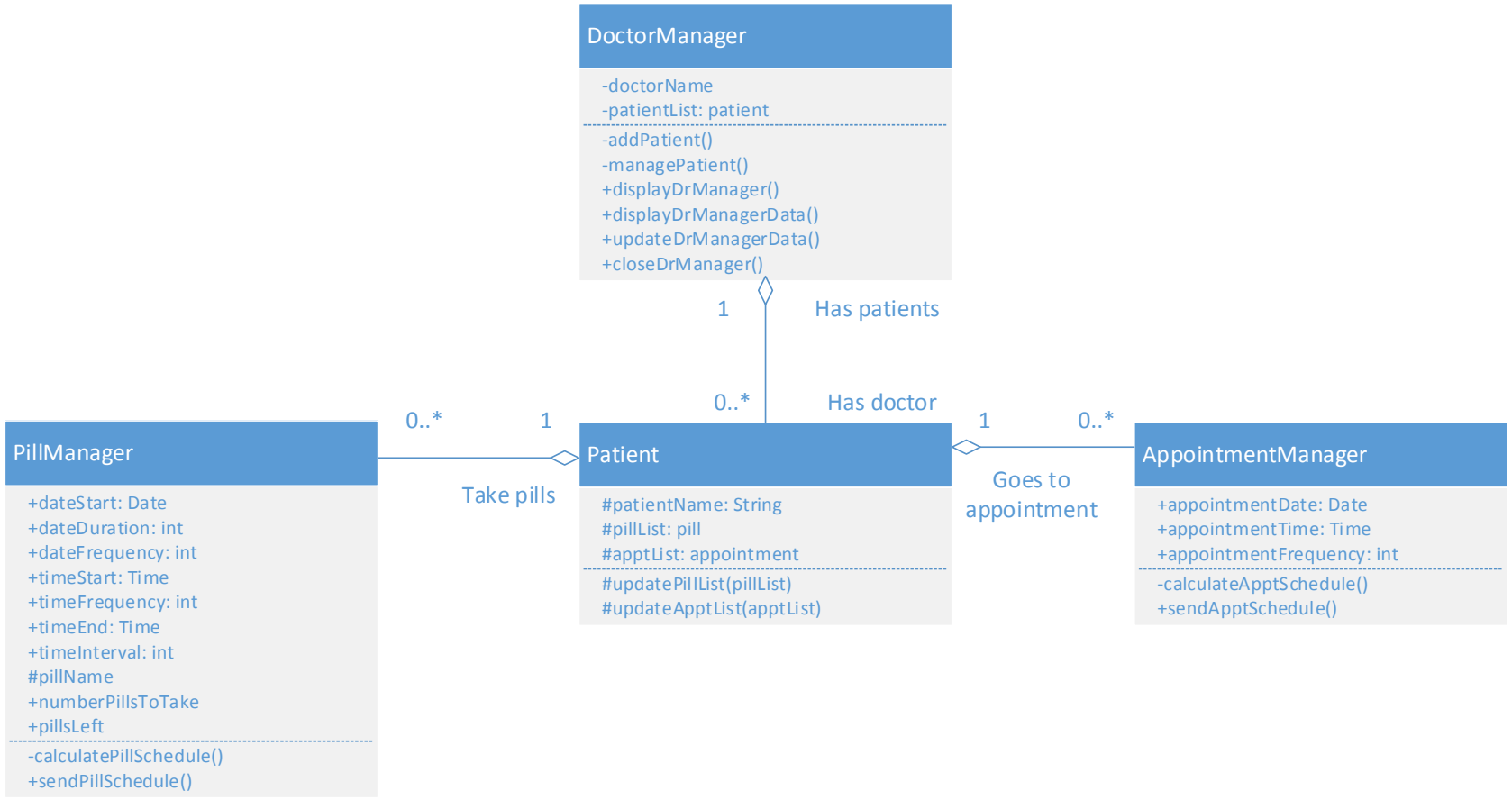
The doctor can submit a pill schedule.



The doctor can also set a patient's appointment.



Class Diagram:



Trace of Requirements to Design:

Item Description	Detailed Design Section	Document of Origin	Sections of Origin
Overall functionality as pills/appointments manager	Overview	Concept of Operations	Users and Modes of Operation, Operational Features
Assign patient medicine	Event Sequence Diagrams	Software Requirements Specification	Event Table, Use Case Descriptions
Assign appointments	Event Sequence Diagrams	Software Requirements Specification	Event Table
Pill/appointment management functionality	Overview, Data	Software Requirements Specification	3.1
Pill data specifications	Data, Operations	Software Requirements Specification	3.2
Appointment data specifications	Data, Operations	Software Requirements Specification	3.2
Example data and Data format	Data, Operations	Software Requirements Specification	3.6
Quality attributes	Design Issues	Software Requirements Specification	3.9